



БИБЛИОТЕКА ПРОГРАММИСТА



П. Дейтел Х. Дейтел Э. Дейтел М. Моргано

Android для разработчиков

- Приложение-ориентированный подход
- Новые функции Android SDK 4.3 и 4.4
- Среды разработки ADT на базе Eclipse и Android Studio
- Отправка приложений в Google Play

ANDROID™ FOR PROGRAMMERS

AN APP-DRIVEN APPROACH

SECOND EDITION

DEITEL® DEVELOPER SERIES

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



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



БИБЛИОТЕКА ПРОГРАММИСТА

П. Дейтел Х. Дейтел Э. Дейтел

Android для разработчиков



Москва · Санкт-Петербург · Нижний Новгород · Воронеж
Ростов-на-Дону · Екатеринбург · Самара · Новосибирск
Киев · Харьков · Минск

2015

ББК 32.973.2-018.1

УДК 004.43

А66

Дейтел П., Дейтел Х., Дейтел Э.

А66 Android для разработчиков. — СПб.: Питер, 2015. — 384 с.: ил. — (Серия «Библиотека программиста»).

ISBN 978-5-496-01517-2

Более миллиона человек во всем мире воспользовались книгами Дейтелей, чтобы освоить Java, C#, C++, C, веб-программирование, JavaScript, XML, Visual Basic, Visual C++, Perl, Python и другие языки программирования.

Эта книга, выходящая уже во втором издании, даст вам всё, что нужно, для начала разработки приложений для Android и быстрой публикации их на Google Play. Авторы используют приложение-ориентированный подход, при котором описание каждой технологии рассматривается на примере 16 полностью протестированных приложений для Android. Кроме описания процесса создания приложений, в книге дано пошаговое руководство по размещению ваших приложений на Google Play и примеры успешных публикаций. Новое издание книги полностью обновлено и содержит информацию о работе с Android 4.3 и 4.4, разработке на Eclipse и новом Android Studio.

12+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.973.2-018.1

УДК 004.43

Права на издание получены по соглашению с Prentice Hall, Inc. Upper Sadle River, New Jersey 07458. Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-0133570922 англ.

ISBN 978-5-496-01517-2

© Prentice Hall

© Перевод на русский язык ООО Издательство «Питер», 2015

© Издание на русском языке, оформление ООО Издательство «Питер», 2015

Оглавление

Предисловие	16
Авторские права и лицензии на код	17
Целевая аудитория	17
Особенности книги	18
Как связаться с авторами книги	19
Благодарности	20
Об авторах	21
Подготовка	23
Требования к аппаратному и программному обеспечению	23
Установка Java Development Kit (JDK)	23
Выбор среды разработки	24
Android 4.3 SDK	25
Создание виртуальных устройств Android (AVD) для использования в эмуляторе Android	26
Настройка устройства Android для разработки	28
Как получить примеры кода	29
Глава 1. Введение в Android	31
1.1. Введение	32
1.2. Android – мировой лидер в области мобильных операционных систем	32
1.3. Особенности Android	33
1.4. Операционная система Android	37
1.4.1. Android 2.2 (Froyo)	38
1.4.2. Android 2.3 (Gingerbread)	39
1.4.3. Android 3.0–3.2 (Honeycomb)	39
1.4.4. Android Ice Cream Sandwich	40
1.4.5. Android 4.1–4.3 (Jelly Bean)	41
1.4.6. Android 4.4 (KitKat)	42

1.5. Загрузка приложений из Google Play	43
1.6. Пакеты.....	45
1.7. Android Software Development Kit (SDK)	46
1.8. Краткий обзор объектно-ориентированного программирования.....	49
1.8.1. Автомобиль как объект	50
1.8.2. Методы и классы	50
1.8.3. Создание экземпляра класса	51
1.8.4. Повторное использование.....	51
1.8.5. Сообщения и вызовы методов	51
1.8.6. Атрибуты и переменные экземпляра класса.....	51
1.8.7. Инкапсуляция	52
1.8.8. Наследование.....	52
1.8.9. Объектно-ориентированный анализ и проектирование.....	52
1.9. Тестирование приложения Doodlz на виртуальном устройстве AVD	53
1.9.1. Запуск приложения Doodlz на AVD для смартфона Nexus 4.....	54
1.9.2. Запуск Doodlz App на планшетном AVD	64
1.9.3. Выполнение приложения Doodlz на устройстве Android.....	64
1.10. Создание успешных Android-приложений	66
1.11. Ресурсы для разработчиков.....	68
1.12. Резюме	70

Глава 2. Приложение Welcome..... 72

2.1. Введение	73
2.2. Обзор применяемых технологий	74
2.2.1. Android Developer Tools IDE	74
2.2.2. TextView и ImageView	74
2.2.3. Ресурсы приложения	74
2.2.4. Доступность.....	74
2.2.5. Интернационализация	75
2.3. Создание приложения	75
2.3.1. Запуск интегрированной среды разработки Android Developer Tools..	75
2.3.2. Создание нового проекта.....	75
2.3.3. Диалоговое окно New Android Application.....	75
2.3.4. Шаг Configure Project	78
2.3.5. Шаг Configure Launcher Icon.....	78
2.3.6. Шаг Create Activity.....	80
2.3.7. Шаг Blank Activity	81

2.4. Окно Android Developer Tools.....	82
2.4.1. Окно Package Explorer	82
2.4.2. Окна редактора.....	83
2.4.3. Окно структуры	84
2.4.4. Файлы ресурсов приложения	84
2.4.5. Макетный редактор	85
2.4.6. Графический интерфейс по умолчанию	85
2.5. Построение графического интерфейса приложения.....	87
2.5.1. Добавление изображений в проект	87
2.5.2. Изменение свойства Id компонентов RelativeLayout и TextView	88
2.5.3. Настройка компонента TextView	89
2.6. Выполнение приложения Welcome.....	96
2.7. Обеспечение доступности приложения	97
2.8. Интернационализация приложения.....	99
2.9. Резюме.....	104

Глава 3. Приложение Tip Calculator 105

3.1. Введение	106
3.2. Тестирование приложения Tip Calculator	107
3.3. Обзор применяемых технологий	108
3.3.1. Класс Activity	108
3.3.2. Методы жизненного цикла активности.....	108
3.3.3. Построение представления с использованием компонентов LinearLayout и GridLayout	109
3.3.4. Создание и настройка графического интерфейса.....	110
3.3.5. Форматирование чисел в соответствии с локальным контекстом	111
3.3.6. Реализация интерфейса TextWatcher для обработки изменений в компоненте EditText	111
3.3.7. Реализация интерфейса OnSeekBarChangeListener для обработки изменения позиции ползунка SeekBar	112
3.3.8. AndroidManifest.xml	112
3.4. Построение графического интерфейса приложения.....	112
3.4.1. Основы GridLayout.....	112
3.4.2. Создание проекта TipCalculator	114
3.4.3. Переключение на GridLayout	115
3.4.4. Добавление компонентов TextView, EditText, SeekBar и LinearLayout	116
3.4.5. Настройка компонентов.....	118

3.5. Включение функциональности в приложение	123
3.6. Файл AndroidManifest.xml	132
3.7. Резюме.....	133

Глава 4. Приложение Twitter® Searches 135

4.1. Введение	136
4.2. Тестирование приложения.....	137
4.2.1. Импортирование и запуск приложения	137
4.2.2. Добавление нового запроса.....	137
4.2.3. Просмотр результатов поиска.....	139
4.2.4. Редактирование запроса	140
4.2.5. Пересылка запроса.....	141
4.2.6. Удаление запроса.....	142
4.2.7. Прокрутка списка сохраненных запросов	143
4.3. Обзор применяемых технологий	144
4.3.1. ListView.....	144
4.3.2. ListActivity.....	145
4.3.3. Настройка макета ListActivity	145
4.3.4. ImageButton.....	145
4.3.5. SharedPreferences	145
4.3.6. Интенты запуска других активностей	146
4.3.7. AlertDialog	147
4.3.8. Файл AndroidManifest.xml	148
4.4. Построение графического интерфейса приложения	148
4.4.1. Создание проекта	148
4.4.2. Файл activity_main.xml	149
4.4.3. Добавление GridLayout и других компонентов.....	150
4.4.4. Панель инструментов макетного редактора	155
4.4.5. Макет варианта ListView: list_item.xml	157
4.5. Построение класса MainActivity.....	158
4.5.1. Команды package и import.....	159
4.5.2. Расширение ListActivity	160
4.5.3. Поля класса MainActivity	161
4.5.4. Переопределение метода активности onCreate	162
4.5.5. Анонимный внутренний класс, реализующий интерфейс OnItemClickListener для сохранения нового или измененного запроса	164
4.5.6. Метод addTaggedSearch.....	167

4.5.7. Анонимный внутренний класс, реализующий интерфейс OnItemClickListener класса ListView для отображения результатов поиска.....	168
4.5.8. Анонимный внутренний класс, реализующий интерфейс OnItemLongClickListener класса ListView для пересылки, изменения и удаления запросов	170
4.5.9. Метод shareSearch	172
4.5.10. Метод deleteSearch	174
4.6. AndroidManifest.xml	175
4.7. Резюме.....	176
Глава 5. Приложение Flag Quiz.....	177
5.1. Введение	178
5.2. Тестирование приложения Flag Quiz	179
5.2.1. Импортирование и запуск приложения.....	179
5.2.2. Настройка викторины	180
5.2.3. Ответы на вопросы викторины	180
5.3. Обзор применяемых технологий	184
5.3.1. Меню.....	184
5.3.2. Фрагменты.....	184
5.3.3. Методы жизненного цикла фрагментов.....	185
5.3.4. Управление фрагментами.....	186
5.3.5. Объекты Preference	186
5.3.6. Папка assets.....	186
5.3.7. Папки ресурсов	187
5.3.8. Поддержка разных размеров экранов и разрешений.....	187
5.3.9. Определение размера экрана.....	188
5.3.10. Вывод временных сообщений	188
5.3.11. Использование обработчика для планирования будущих операций	189
5.3.12. Применение анимации к компонентам	189
5.3.13. Регистрация исключений с помощью Log.e	189
5.3.14. Использование явного интента для запуска другой активности в том же приложении	190
5.3.15. Структуры данных Java	190
5.4. Построение графического интерфейса и файлов ресурсов.....	190
5.4.1. Создание проекта	190
5.4.2. Файл strings.xml и ресурсы форматных строк	191

5.4.3. arrays.xml	192
5.4.4. colors.xml.....	193
5.4.5. dimens.xml	194
5.4.6. Макет activity_settings.xml	194
5.4.7. Макет activity_main.xml для телефонов и планшетов в портретной ориентации	195
5.4.8. Макет fragment_quiz.xml.....	196
5.4.9. Макет activity_main.xml для планшета в альбомной ориентации	198
5.4.10. Определение конфигурации приложения в файле preferences.xml	199
5.4.11. Создание анимации «качающегося» флага.....	201
5.5. Класс MainActivity	203
5.5.1. Команда package, команды import и поля.....	203
5.5.2. Переопределение метода onCreate	204
5.5.3. Переопределение метода onStart	206
5.5.4. Переопределение метода onCreateOptionsMenu	207
5.5.5. Переопределение метода onOptionsItemSelected	208
5.5.6. Анонимный внутренний класс, реализующий интерфейс OnSharedPreferenceChangeListener	208
5.6. Класс QuizFragment	210
5.6.1. Команда package и команды import	210
5.6.2. Поля	211
5.6.3. Переопределение метода onCreateView	212
5.6.4. Метод updateGuessRows	214
5.6.5. Метод updateRegions.....	215
5.6.6. Метод resetQuiz	215
5.6.7. Метод loadNextFlag	217
5.6.8. Метод getCountryName	219
5.6.9. Анонимный внутренний класс, реализующий интерфейс OnClickListener	220
5.6.10. Метод disableButtons	222
5.7. Класс SettingsFragment	223
5.8. Класс SettingsActivity.....	223
5.9. AndroidManifest.xml	224
5.10. Резюме	225
Глава 6. Приложение Cannon Game	227
6.1. Введение	228

6.2. Тестирование приложения Cannon Game.....	229
6.3. Обзор применяемых технологий	230
6.3.1. Присоединение пользовательского представления к макету	230
6.3.2. Использование папки ресурсов raw.....	230
6.3.3. Методы жизненного цикла активности и фрагмента.....	230
6.3.4. Переопределение метода onTouchEvent класса View.....	231
6.3.5. Добавление звука с помощью SoundPool и AudioManager	231
6.3.6. Покадровая анимация с помощью потоков, SurfaceView и SurfaceHolder.....	232
6.3.7. Простое обнаружение столкновений	232
6.3.8. Рисование графики с помощью Paint и Canvas	233
6.4. Создание графического интерфейса приложения и файлов ресурсов.....	233
7.4.1. Создание проекта	233
6.4.2. Файл strings.xml	234
6.4.3. Файл fragment_game.xml	234
6.4.4. Файл activity_main.xml	235
6.4.5. Добавление звуков в приложение	235
6.5. Класс Line.....	236
6.6. Класс MainActivity	236
6.7. Класс CannonGameFragment.....	237
6.8. Класс CannonView	238
6.8.1. Команда package и команды import	239
6.8.2. Переменные экземпляров и константы	239
6.8.3. Конструктор	241
6.8.4. Переопределение метода onSizeChanged класса View	243
6.8.5. Метод newGame	244
6.8.6. Метод updatePositions	245
6.8.7. Метод fireCannonball класса CannonView	249
6.8.8. Метод alignCannon.....	249
6.8.9. Метод drawGameElements.....	250
6.8.10. Метод showGameOverDialog.....	252
6.8.11. Методы stopGame и releaseResources	254
6.8.12. Реализация методов SurfaceHolder.Callback	254
6.8.13. Переопределение метода onTouchEvent	255
6.8.14. Поток CannonThread: использование потока для создания цикла игры	256
6.9. Резюме.....	258

Глава 7. Приложение Doodlz.....	259
7.1. Введение	260
7.2. Обзор применяемых технологий	261
7.2.1. Использование SensorManager для прослушивания событий акселерометра	262
7.2.2. Пользовательские реализации DialogFragment	262
7.2.3. Рисование с использованием Canvas и Bitmap	263
7.2.4. Обработка событий многоточечных касаний и хранение данных линий в объектах Path.....	263
7.2.5. Режим погружения Android 4.4.....	263
7.2.6. GestureDetector и SimpleOnGestureListener	264
7.2.7. Сохранение рисунка в галерее устройства.....	264
7.2.8. Поддержка печати в Android 4.4 и класс PrintHelper из Android Support Library.....	264
7.3. Создание графического интерфейса и файлов ресурсов приложения.....	264
7.3.1. Создание проекта	265
7.3.2. Файл strings.xml.....	265
7.3.3. Файл dimens.xml	266
7.3.4. Меню DoodleFragment	267
7.3.5. Макет activity_main.xml для MainActivity	268
7.3.6. Файл fragment_doodle.xml для фрагмента DoodleFragment	269
7.3.7. Макет fragment_color.xml для фрагмента ColorDialogFragment	269
7.3.8. Макет fragment_line_width.xml для фрагмента LineWidthDialogFragment	272
7.3.9. Добавление класса EraseImageDialogFragment	273
7.4. Класс MainActivity	274
7.5. Класс DoodleFragment.....	275
7.6. Класс DoodleView	282
7.7. Класс ColorDialogFragment	295
7.8. Класс LineWidthDialogFragment	299
7.9. Класс EraseImageDialogFragment	302
7.10. Резюме	304
Глава 8. Приложение Address Book.....	306
8.1. Введение	307
8.2. Тестирование приложения Address Book.....	309
8.3. Обзор применяемых технологий	311
8.3.1. Отображение фрагментов с использованием FragmentTransaction	311

8.3.2. Передача данных между фрагментом и управляющей активностью	311
8.3.3. Метод onSaveInstanceState	312
8.3.4. Определение и применение стилей к компонентам GUI	312
8.3.5. Определение фона для компонентов TextView	312
8.3.6. Расширение класса ListFragment для создания фрагмента, содержащего ListView.....	312
8.3.7. Работа с базой данных SQLite	313
8.3.8. Выполнение операций с базами данных за пределами потока GUI с использованием AsyncTask	313
8.4. Создание графического интерфейса пользователя и файлов ресурсов.....	313
8.4.1. Создание проекта	313
8.4.2. Создание классов приложения.....	314
8.4.3. Файл strings.xml.....	315
8.4.4. Файл styles.xml.....	316
8.4.5. Файл textview_border.xml	317
8.4.6. Файл макета MainActivity: activity_main.xml	317
8.4.7. Файл макета DetailsFragment: fragment_details.xml	318
8.4.8. Макет AddEditFragment: fragment_add_edit.xml	320
8.4.9. Определение меню фрагментов	321
8.5. Класс MainActivity	323
8.6. Класс ContactListFragment.....	329
8.7. Класс AddEditFragment	336
8.8. Класс DetailsFragment	342
8.9. Вспомогательный класс DatabaseConnector.....	350
8.10. Резюме	356
Глава 9. Google Play и коммерческие аспекты разработки	358
9.1. Введение	359
9.2. Подготовка приложений к публикации	359
9.2.1. Тестирование приложения.....	360
9.2.2. Лицензионное соглашение.....	360
9.2.3. Значки и метки.....	360
9.2.4. Контроль версии приложения	361
9.2.5. Лицензирование для управления доступом к платным приложениям	362
9.2.6. Маскировка кода	362
9.2.7. Получение закрытого ключа для цифровой подписи	362

9.2.8. Снимки экрана	363
9.2.9. Информационный видеоролик	364
9.3. Цена приложения: платное или бесплатное?	365
9.3.1. Платные приложения	366
9.3.2. Бесплатные приложения	367
9.4. Монетизация приложений с помощью встроенной рекламы.....	368
9.5. Внутренняя продажа виртуальных товаров	369
9.6. Регистрация в Google Play	371
9.7. Создание учетной записи Google Wallet	371
9.8. Отправка приложений в Google Play	372
9.9. Запуск Play Store из приложения.....	374
9.10. Управление приложениями в Google Play	375
9.11. Другие магазины приложений Android	375
9.12. Другие популярные платформы мобильных приложений	376
9.13. Маркетинг приложения.....	377
9.14. Резюме	382

Посвящается памяти Амара Дж. Боза, профессора МИТ, основателя и председателя Bose Corporation.

Для нас было честью быть вашим студентом, а следующее поколение Дейтелов часто слышало от своего отца, что именно ваши уроки вдохновили его на самые серьезные достижения. Вы научили нас, что, если браться за действительно сложные задачи, — можно достичь выдающихся результатов.

Харви Дейтел

Пол и Эбби Дейтел

Предисловие

Добро пожаловать в динамичный мир разработки приложений для смартфонов и планшетов Android с использованием Android Software Development Kit (SDK), языка программирования Java™, интегрированной среды разработки Android Development Tools на базе Eclipse, а также новой и стремительно развивающейся среды разработки Android Studio.

В этой книге представлены передовые технологии разработки мобильных приложений для профессиональных программистов. В основу книги заложен принцип *разработки, ориентированной на приложения*, — концепции разработки продемонстрированы на примере *полностью работоспособных приложений* Android, а не фрагментов кода. Каждая глава начинается с вводной части, в которой вкратце описано разрабатываемое приложение. Затем приводятся результаты тестирования приложения и обзор технологий, применяемых в процессе его разработки. Далее выполняется подробный анализ исходного кода приложения. Исходный код всех приложений доступен на сайте www.deitel.com/books/AndroidFP2/. Во время чтения книги мы рекомендуем держать исходный код открытым в среде разработки.

Объемы продаж устройств Android и количество загрузок Android-приложений стремительно растут. Мобильные телефоны Android первого поколения появились на рынке в октябре 2008 года. Согласно результатам исследования рынка, проведенного компанией Strategy Analytics, к октябрю 2013 года смартфоны Android занимали 81,3% глобального рынка смартфонов в США, смартфоны Apple iPhone — 13,4%, смартфоны с системами Microsoft — 4,1%, а BlackBerry — 1%¹. По данным отчета IDC, к концу первого квартала 2013 года Android принадлежало 56,5% глобального рынка планшетных устройств, по сравнению с 39,6% у Apple iPad и 3,7% у планшетов на базе Microsoft Windows².

Сейчас в мире используется более миллиарда смартфонов и планшетов на базе Android³, и ежедневно активируется более 1,5 миллиона устройств⁴. По данным IDC, компания Samsung является ведущим производителем устройств на базе

¹ <http://blogs.strategyanalytics.com/WSS/post/2013/10/31/Android-Captures-Record-81-Percent-Share-of-Global-Smartphone-Shipments-in-Q3-2013.aspx>

² <http://www.idc.com/getdoc.jsp?containerId=prUS24093213>

³ <http://www.android.com/kitkat>

⁴ <http://www.technobuffalo.com/2013/04/16/google-daily-android-activations-1-5-million>

Android; в III квартале 2013 года на ее долю приходилось почти 40% поставок таких устройств.

С Google Play – магазина приложений Android компании Google – загружены миллиарды приложений. Возможности, предоставляемые разработчикам приложений Android, поистине безграничны.

Ожесточенная конкуренция среди разработчиков популярных мобильных платформ и мобильных сервисов приводит к быстрому внедрению инноваций и стремительному обвалу цен. Благодаря соперничеству между десятками производителей устройств Android ускоряется внедрение аппаратных и программных инноваций в сообществе Android.

Авторские права и лицензии на код

Весь код и Android-приложения, приведенные в книге, являются собственностью компании Deitel & Associates, Inc. Примеры программ, рассмотренные в книге, распространяются на условиях лицензии Creative Commons Attribution 3.0 Unported License (<http://creativecommons.org/licenses/by/3.0/>) за исключением того, что эти примеры не могут использоваться повторно каким-либо образом в учебниках и руководствах, распространяемых в печатном и цифровом формате. Авторы и издатель не предоставляют никаких прямых или косвенных гарантий относительно программ или документации, содержащихся в книге. Авторы и издатель ни при каких условиях не несут ответственности за случайные или предсказуемые убытки в связи с комплектацией, производительностью или использованием этих программ. Разрешается использование рассмотренных в книге приложений и их функционала в качестве основы для ваших собственных приложений. Если у вас возникают какие-либо вопросы, обращайтесь по адресу deitel@deitel.com.

Целевая аудитория

Предполагается, что читатели этой книги знают язык Java и имеют опыт объектно-ориентированного программирования, знакомы с XML. Благодаря совершенствованию средств разработки Android-приложений в этом издании нам удалось почти полностью исключить разметку XML. В книге осталось всего два небольших и понятных XML-файла, с которыми вам придется работать. Мы используем только завершенные рабочие приложения, поэтому, даже не зная Java, но имея опыт объектно-ориентированного программирования на C#/.NET, Objective-C/Сocoa либо C++ (с библиотеками классов), вы сможете быстро освоить излагаемый в книге материал, а заодно узнать много полезного о Java и объектно-ориентированном программировании.

Эта книга *не является* учебником по Java, но вместе с тем содержит значительный объем материала по этим технологиям в контексте разработки Android-приложений. Если вы хотите глубже освоить Java, обратите внимание на следующие книги:

- ❑ Java for Programmers, 2/e (www.deitel.com/books/javafp2)
- ❑ Java Fundamentals: Parts I and II LiveLessons videos (www.deitel.com/books/LiveLessons)
- ❑ Java How to Program, 10/e (www.deitel.com/books/jhtp10)

Если вы не знакомы с XML, обращайтесь к онлайновым учебникам:

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

Особенности книги

Разработка, ориентированная на приложения. В каждой из глав 2–8 представлено одно полное приложение — рассмотрены функции приложения, приведены снимки экрана выполняющегося приложения, результаты тестовых запусков и обзор технологий и архитектуры, используемых при создании приложения. Затем мы строим графический интерфейс приложения, представляем его полный исходный код и проводим подробный анализ этого кода; обсуждаем концепции, применяемые в программировании, и демонстрируем функциональность Android API, используемую при создании приложения.

Android SDK 4.3 и 4.4. В книге рассматривается множество новых функций, включенных в состав пакета Android SDK (Software Development Kit) 4.3 и 4.4.

Фрагменты. Начиная с главы 5 мы будем использовать фрагменты для создания и управления частями графического интерфейса каждого фрагмента. Объединяя несколько фрагментов, можно создавать интерфейсы, эффективно использующие пространство экрана планшетов. Разработчик может легко заменять фрагменты, что делает графический интерфейс более динамичным; пример переключения фрагментов рассматривается в главе 8.

Поддержка разных размеров и разрешений экрана. В главах приложений будет продемонстрировано применение средств автоматического выбора ресурсов Android (макетов, изображений и т. д.) на основании размеров и ориентации устройства.

Описание среды разработки ADT (Android Development Tools) на базе Eclipse в печатной версии книги. Бесплатная интегрированная среда разработки (IDE) Android Development Tools (ADT), включающая Eclipse и плагин ADT, в сочетании с бесплатным пакетом JDK (Java Development Kit) предоставляет все необходимое для создания, запуска и отладки приложений Android, поддержки их распространения (например, отправки в магазин Google Play™) и т. д.

Android Studio. Перспективная среда разработки для будущей разработки приложений Android. Так как среда Android Studio быстро развивается, ее обсуждение размещено в сети по адресу

<http://www.deitel.com/books/AndroidFP2>

Мы покажем, как импортировать готовые проекты для тестирования приложений. Также будет продемонстрировано создание новых приложений, построение графического интерфейса, правка файлов ресурсов и тестирование приложений. Если у вас возникнут вопросы, свяжитесь с нами по адресу deitel@deitel.com.

Режим погружения. Панель состояния в верхней части экрана и кнопки меню в нижней части можно скрыть, чтобы ваши приложения могли использовать большую часть экрана. Чтобы получить доступ к панели состояния, пользователь проводит пальцем от верхнего края экрана, а к системной панели с кнопками Back, Home и Recent Apps — от нижнего края.

Инфраструктура печати. Android 4.4 KitKat позволяет добавить в приложение поддержку печати: поиск доступных принтеров по Wi-Fi или в облаке, выбор размера листа, выбор печатаемых страниц и т. д.

Тестирование на смартфонах Android, планшетах и в эмуляторе. Для достижения оптимального результата приложения следует тестировать на физических смартфонах и планшетах Android. Полезную информацию также можно получить при тестировании в эмуляторе Android (см. раздел «Подготовка»), однако эмуляция создает существенную нагрузку на процессор и может работать медленно, особенно в играх с большим количеством подвижных объектов. В главе 1 перечислены некоторые функции Android, не поддерживаемые эмулятором.

Мультимедиа. В приложениях используются разнообразные мультимедийные возможности Android, включая графику, изображения, покадровую анимацию, анимацию и работу с аудио.

Отправка приложений в Google Play. В главе 9 описан процесс регистрации в Google Play и настройки учетной записи для продажи приложений. Вы узнаете, как подготовить приложение к отправке в Google Play, как установить цену на приложение, и познакомитесь с возможностями монетизации приложений через размещение рекламы и внутренние продажи. Также будут представлены ресурсы, которые могут использоваться для маркетинга приложений. Главу 9 можно читать после главы 1.

Как связаться с авторами книги

Мы ждем ваши комментарии, критические замечания, исправления и предложения по улучшению книги. С вопросами и предложениями обращайтесь по адресу deitel@deitel.com.

Исправления и уточнения к материалу книги публикуются по адресу

www.deitel.com/books/AndroidFP2

а также в Facebook, Twitter, Google+, LinkedIn и Deitel® Buzz Online.

Посетите сайт www.deitel.com, здесь вы сможете:

- ❑ загрузить примеры кода;
- ❑ ознакомиться с постоянно расширяющимся списком ресурсов по программированию;
- ❑ получить обновления к материалу книги;
- ❑ подписаться на бесплатный бюллетень Deitel® Buzz Online, распространяемый по электронной почте, по адресу www.deitel.com/newsletter/subscribe.html;
- ❑ получить информацию об учебных курсах серии Dive Into® Series, проводимых на территории клиента по всему миру.

Благодарности

Спасибо Барбаре Дейтел (Barbara Deitel) за многочасовую работу над проектом — она создавала наши ресурсные центры Android, терпеливо разбираясь во всех технических подробностях.

Эта книга стала результатом сотрудничества между академическим и профессиональным подразделением Pearson. Мы высоко ценим экстраординарные усилия и 18-летнее наставничество нашего друга и профессионала Марка Л. Тауба (Mark L. Taub), главного редактора издательской группы Pearson Technology Group. Марк со своей группой работает над всеми нашими профессиональными книгами и видеокурсами. Ким Бедигхаймер (Kim Boedigheimer) привлекла к работе заслуженных участников сообщества Android и организовала работу группы рецензирования по контенту, относящемуся к Android. Мы выбрали иллюстрацию для обложки, а Чати Презерсит (Chuti Prasertsith) и Сандра Шредер (Sandra Schroeder) разработали ее дизайн. Джон Фуллер (John Fuller) руководил процессом публикации всех наших книг из серии Deitel Developer Series.

Мы хотим выразить свою благодарность Трейси Джонсон (Tracy Johnson), ответственному редактору по направлению компьютерных технологий. Трейси и ее группа работали над всеми нашими учебниками. Кэрол Снайдер (Carole Snyder) подбирала научных рецензентов и руководила процессом рецензирования. Боб Энглхардт (Bob Engelhardt) управляет выпуском наших научных работ.

Мы благодарим Майкла Моргано, нашего бывшего коллегу по Deitel & Associates, Inc., а теперь Android-разработчика в компании Imerj™, соавтора первых изданий этой книги, а также книги «iPhone for Programmers: An App-Driven Approach». Майкл — исключительно одаренный программист.

Рецензенты книг «Android for Programmers: An App-Driven Approach» и «Android How to Program»

Мы хотим поблагодарить рецензентов первого и второго издания книги. Они тщательно проверили текст и предоставили множество рекомендаций по его улучшению: Пол Бойстеръен (Paul Beusterien), главный специалист компании Mobile Developer Solutions; Эрик Дж. Боуден (Eric J. Bowden), главный управляющий компании Safe Driving Systems, LLC; Тони Кантрелл (Tony Cantrell) (Северо-западный технический колледж штата Джорджия); Иан Дж. Клифтон (Ian G. Clifton), независимый подрядчик и разработчик приложений Android; Даниэль Гэлпин (Daniel Galpin), энтузиаст Android и автор книги «Intro to Android Application Development»; Джим Хэзевэй (Jim Hathaway), разработчик из компании Kellogg; Дуглас Джонс (Douglas Jones), старший инженер-программист, компания Fullpower Technologies; Чарльз Ласки (Charles Lasky), муниципальный колледж Нагаутук; Энрике Лопес-Манас (Enrique Lopez-Manas), старший специалист по архитектуре Android и преподаватель информатики в университете Алькала, Мадрид; Себастиан Никопп (Sebastian Nykopp), главный архитектор, компания Reaktor; Майкл Пардо, разработчик Android, компания Mobiata; Ронан «Зеро» Шварц (Ronan «Zero» Schwarz), директор по информационным технологиям, компания OpenIntents; Ариджит Сенгупта (Arijit Sengupta), университет Wright State; Дональд Смит (Donald Smith), Колумбийский колледж; Хесус Убальдо (Jesus Ubaldo), Кеведо Торреро, университет штата Висконсин, Парксайд; Дон Уик (Dawn Wick), Юго-западный муниципальный колледж; Фрэнк Ксю (Frank Xu), университет Гэннон.

Итак, свершилось! Эта книга поможет вам быстро освоить разработку приложений Android. Мы надеемся, что от чтения этой книги вы получите не меньше удовольствия, чем мы — от ее написания!

Пол Дейтел
Харви Дейтел
Эбби Дейтел

Об авторах

Пол Дж. Дейтел (Paul J. Deitel), генеральный и технический директор компании Deitel & Associates, Inc., окончил Массачусетский технологический институт (MIT) по специальности «Информационные технологии» (Information Technology). Обладатель сертификатов Java Certified Programmer, Java Certified Developer и Oracle Java Champion. В Deitel & Associates, Inc. он провел сотни занятий по всему миру для корпоративных клиентов, включая Cisco, IBM, Siemens, Sun Microsystems, Dell, Fidelity, NASA (Космический центр имени Кеннеди), Национальный центр прогнозирования сильных штормов, ракетный полигон Уайт-Сэндз, Rogue Wave Software, Boeing, SunGard Higher Education, Stratus, Cambridge Technology Partners, One Wave, Hyperion Software, Adra Systems, Entergy, CableData Systems, Nortel Networks, Puma, iRobot, Invensys и многих других. Пол и его соавтор, д-р Харви

М. Дейтел, являются авторами всемирно известных бестселлеров — учебников по языкам программирования, предназначенных для начинающих и для профессионалов, а также видеокурсов.

Харви М. Дейтел (Dr. Harvey M. Deitel), председатель и главный стратег компании Deitel & Associates, Inc., имеет 50-летний опыт работы в области информационных технологий. Он получил степени бакалавра и магистра Массачусетского технологического института и степень доктора философии Бостонского университета. В 1960-е годы он работал в группах, занимавшихся созданием различных операционных систем IBM, в Advanced Computer Techniques и Computer Usage Corporation, а в 1970-е годы занимался разработкой коммерческих программных систем. Харви имеет огромный опыт преподавания в колледже и занимал должность председателя отделения информационных технологий Бостонского колледжа. В 1991 году вместе с сыном — Полом Дж. Дейтелем — он основал компанию Deitel & Associates, Inc. Харви с Полом написали несколько десятков книг и выпустили десятки видеокурсов LiveLessons. Написанные ими книги получили международное признание и были изданы на китайском, корейском, японском, немецком, русском, испанском, французском, польском, итальянском, португальском, греческом, турецком языках и на языке урду. Дейтел провел сотни семинаров по программированию в крупных корпорациях, академических институтах, правительственные и военных организациях.

Эбби Дейтел (Abbey Deitel), президент компании Deitel & Associates, Inc., закончила школу менеджмента Террер при университете Карнеги-Мелон и получила степень бакалавра в области промышленного менеджмента. Она курирует коммерческие операции в компании Deitel & Associates, Inc. на протяжении 16 лет. Эбби автор либо соавтор многочисленных публикаций в Deitel & Associates и вместе с Полом и Харви участвовала в написании книг «Android for Programmers: An App-Driven Approach, 2/e», «iPhone for Programmers: An App-Driven Approach», «Internet & World Wide Web How to Program, 5/e», «Visual Basic 2012 How to Program, 6/e» и «Simply Visual Basic 2010, 5/e».

Подготовка

Благодаря этому разделу ваш компьютер будет правильно настроен и подготовлен к выполнению упражнений, приведенных в книге. Средства разработчика Android часто обновляются. Прежде чем читать этот раздел, посетите сайт книги

<http://www.deitel.com/books/AndroidFP2/>

и проверьте, не был ли опубликован обновленный вариант.

Требования к аппаратному и программному обеспечению

Для разработки приложений Android необходима операционная система Windows®, Linux либо Mac OS X. Новейшие требования к операционной системе доступны по адресу

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

(прокрутите до раздела SYSTEM REQUIREMENTS). Для разработки приложений, представленных в книге, используется следующее программное обеспечение:

- ❑ Java SE 7 Software Development Kit;
- ❑ Android SDK/ADT Bundle на базе среды разработки Eclipse;
- ❑ Android SDK, версии 4.3 и 4.4.

О том, как установить все эти компоненты, подробно рассказано ниже.

Установка Java Development Kit (JDK)

Для разработки Android-приложений требуется пакет *Java Development Kit (JDK)* версии 7 (JDK 7) или 6 (JDK 6). Мы используем JDK 7. Чтобы загрузить JDK для Windows, OS X или Linux, перейдите на сайт

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

Для разработки потребуется только JDK. Выберите 32- или 64-разрядную версию в зависимости от типа процессора вашего компьютера и операционной системы. На новейших компьютерах используется 64-разрядное оборудование — проверьте спецификацию своей системы. Если вы используете 32-разрядную операционную систему, используйте 32-разрядную версию JDK. Выполните инструкции по установке по адресу

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

Выбор среды разработки

В настоящее время Google предоставляет два варианта среды разработки приложений Android.

- ❑ Пакет Android SDK/ADT — версия среды разработки Eclipse с заранее настроенной новейшей версией Android Software Development Kit (SDK) и новейшей версией плагина Android Development Tools (ADT). На момент написания книги это были версии Android SDK 4.4 и ADT 22.3.
- ❑ Android Studio — новая среда разработки приложений Android на базе IntelliJ® IDEA; предполагается, что в будущем именно эта среда станет основной.

Пакет Android SDK/ADT широко использовался в разработке приложений Android в течение нескольких лет. Среда Android Studio, выпущенная в мае 2013 года, существует в ознакомительной версии и стремительно развивается. По этой причине в книге мы будем использовать популярный вариант Android SDK/ADT.

Установка пакета Android SDK/ADT

Чтобы загрузить пакет Android SDK/ADT, откройте страницу

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

и нажмите Download the SDK ADT Bundle. Когда загрузка завершится, распакуйте содержимое ZIP-файла в своей системе. Полученная папка содержит вложенную папку `eclipse` со средой разработки Eclipse и вложенную папку `sdk` с Android SDK. Как и в случае с JDK, можно выбрать 32- или 64-разрядную версию. 32-разрядная версия пакета Android SDK/ADT должна использоваться с 32-разрядной версией JDK, а 64-разрядная — с 64-разрядной версией 64-bit JDK.

Установка Android Studio

Во всех инструкциях, относящихся к среде разработки, в печатной версии книги используется пакет Android SDK/ADT. Вы также можете установить и использовать среду Android Studio. Чтобы загрузить Android Studio, откройте страницу

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

и нажмите Download Android Studio. Когда загрузка завершится, запустите программу установки и выполните инструкции для ее завершения. [Примечание: для

разработки Android 4.4 в Android Studio поддерживаются новые языковые средства Java SE 7, включая оператор `<>`, множественные исключения в `catch`, `String` в конструкциях `switch` и `try` с ресурсами.]

Настройка уровня соответствия компилятора Java и вывод номеров строк

Android не обладает полной поддержкой Java SE 7. Чтобы приведенные в книге примеры правильно компилировались, настройте Eclipse для построения файлов, совместимых с Java SE 6. Для этого выполните следующие действия.

1. Откройте среду Eclipse ( или ) из папки `eclipse` внутри папки, в которую был установлен пакет Android SDK/ADT.
2. Когда на экране появится окно `Workspace Launcher`, нажмите `OK`.
3. Выполните команду `Window > Preferences`, чтобы открыть окно `Preferences`. В Mac OS X следует выполнить команду `ADT > Preferences...`
4. Откройте узел `Java` и выберите узел `Compiler`. В разделе `JDK Compliance` задайте уровень соответствия (`Compiler compliance level`) равным 1.6 (тем самым вы указываете, что среда Eclipse должна генерировать откомпилированный код, совместимый с Java SE 6).
5. Откройте узел `General > Editors` и выберите узел `TextEditors`. Убедитесь в том, что режим `Show line numbers` включен. Нажмите `OK`.
6. Закройте Eclipse.

Android 4.3 SDK

В примерах книги использовались пакеты Android SDK версий 4.3 и 4.4. На момент написания этой главы версия 4.4 включалась в комплект поставки Android SDK/ADT и Android Studio. Вам также стоит установить Android 4.3 (и все остальные версии, которые вы захотите поддерживать в своих приложениях). Чтобы установить другие версии платформы Android, выполните следующие действия (пропустите пункты 1 и 2, если среда Eclipse уже открыта).

1. Откройте Eclipse. В зависимости от платформы значок может иметь вид  или .
2. Когда на экране появится окно `Workspace Launcher`, нажмите `OK`.
3. Если в Mac OS X появится окно с сообщением «`Could not find SDK folder '/Users/YourAccount/android-sdk-macosx/'`», нажмите `Open Preferences`, затем `Browse...` и выберите папку `sdk` в том месте, где была выполнена распаковка Android SDK/ADT.
4. Выполните команду `Window > Android SDK Manager`; открывается окно `Android SDK Manager` (рис. 0.1).

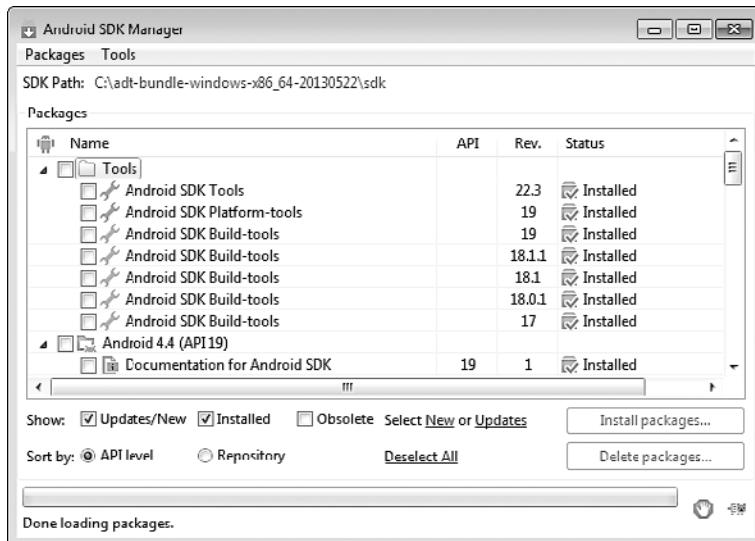


Рис. 0.1. Окно Android SDK Manager

5. В столбце **Name** перечислены все инструменты, версии платформ и дополнительные средства (например, API для взаимодействия с сервисами Google — такими, как Maps), которые вы можете установить в своей системе. Снимите флажок **Installed**. Если в списке **Packages** присутствуют категории **Tools**, **Android 4.4 (API19)**, **Android 4.3 (API18)** и **Extras**, убедитесь в том, что эти пакеты помечены, и нажмите **Install # packages...** (# — количество выбранных пакетов), чтобы открыть окно **Choose Packages to Install**. Многие пакеты в категории **Extras** не обязательны. В этой книге вам понадобится библиотека **Android Support Library** и службы **Google Play**. Пакет **Google USB Driver** необходим для пользователей Windows, желающих тестировать приложения на устройствах Android.
6. В окне **Choose Packages to Install** прочитайте лицензионные соглашения по каждому пакету. Завершив чтение, установите переключатель **Accept License** и нажмите **Install**. Состояние процесса установки отображается в окне **Android SDK Manager**.

Создание виртуальных устройств Android (AVD) для использования в эмуляторе Android

Эмулятор *Android*, включенный в состав *Android SDK*, позволяет тестировать приложения *Android* в эмулированной среде на компьютере, а не на реальном устройстве *Android*. Такая возможность может быть полезна, если вы изучаете программирование для *Android*, не имея доступа к физическому устройству, но эмулятор может работать *очень* медленно, так что физическое устройство все же предпочтительнее. Некоторые средства аппаратного ускорения могут повысить быстродействие эмулятора (developer.android.com/tools/devices/emulator.html#acceleration). Прежде чем

запускать приложение в эмуляторе, создайте *виртуальное устройство* Android Virtual Device (AVD). Оно определяет характеристики реального устройства, для которого должно тестируться приложение: размер экрана (в пикселях), плотность пикселов, физический размер экрана, объем карты памяти SD, используемой в качестве хранилища данных, и ряд других параметров. Если нужно протестировать приложения на нескольких устройствах Android, создайте отдельное устройство AVD для каждого конкретного физического устройства. В этой книге мы используем AVD для эталонных устройств Android (телефон Nexus 4, малый планшет Nexus 7 и большой планшет Nexus 10) с немодифицированными версиями Android. Чтобы создать устройства AVD, выполните следующие действия.

1. Откройте Eclipse.
2. Выполните команды Window ▶ Android Virtual Device Manager. В открывшемся окне Android Virtual Device Manager перейдите на вкладку Device Definitions (рис. 0.2).

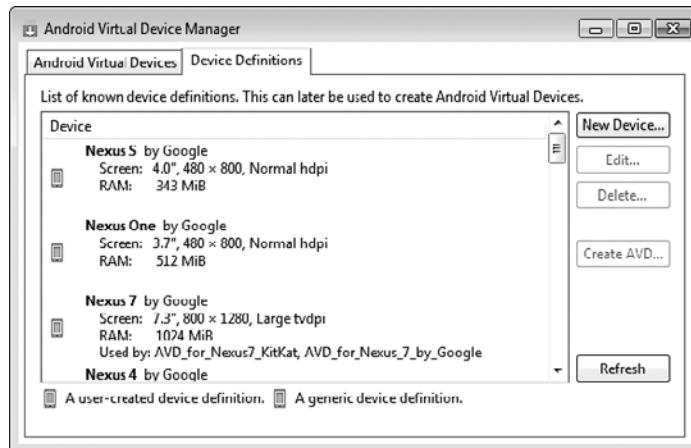


Рис. 0.2. Окно Android Virtual Device Manager

3. Google предоставляет готовые конфигурации, которые могут использоваться для создания AVD. Выберите вариант Nexus 4 by Google и нажмите Create AVD..., чтобы открыть окно Create new Android Virtual Device (AVD) (рис. 0.3). Задайте значения параметров так, как показано на иллюстрации, и нажмите OK, чтобы создать AVD. Если установить флажок Hardware keyboard present, вы сможете использовать клавиатуру компьютера для ввода данных в приложениях, работающих в AVD, но это может помешать отображению виртуальной клавиатуры на экране. Если у вашего компьютера нет камеры, выберите значение Emulated в списках Front Camera и Back Camera. Каждое создаваемое устройство AVD имеет много других параметров, хранимых в файле config.ini. Вы можете изменить этот файл, чтобы он более точно соответствовал аппаратной конфигурации вашего устройства; за дополнительной информацией обращайтесь по адресу

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

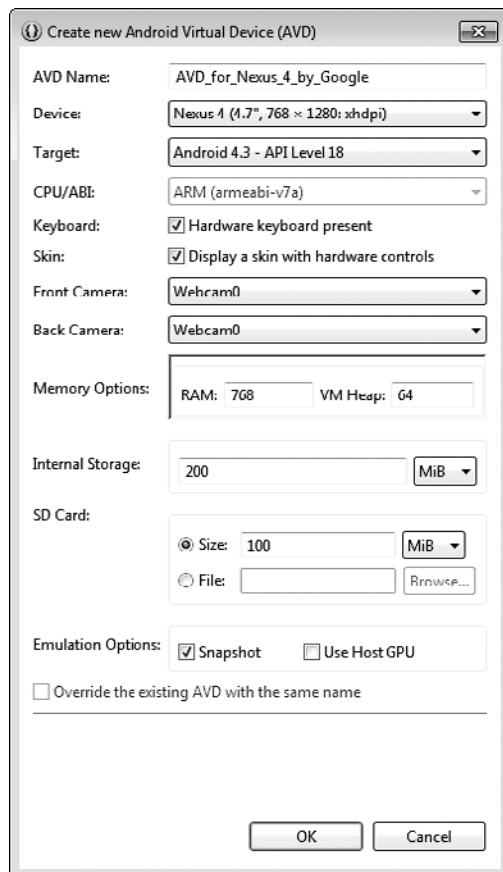


Рис. 0.3. Настройка AVD смартфона Nexus 4 для Android 4.3

- Мы также настроили AVD для платформы Android 4.3, представляющие устройства Nexus 7 и Nexus 10, для тестирования планшетных приложений. Параметры этих устройств показаны на рис. 0.4. Также были созданы AVD для платформы Android 4.4 для Nexus 4, Nexus 7 и Nexus 10, которым были присвоены следующие имена: AVD_for_Nexus_4_KitKat, AVD_for_Nexus_7_KitKat и AVD_for_Nexus_10_KitKat.

Настройка устройства Android для разработки

Как упоминалось ранее, в AVD приложения могут работать медленно. Если у вас имеется физическое устройство Android, тестируйте приложение на этом устройстве. Кроме того, некоторые функции могут тестироваться только на физических устройствах. За информацией о том, как выполнить приложение на устройстве Android, обращайтесь по адресу

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

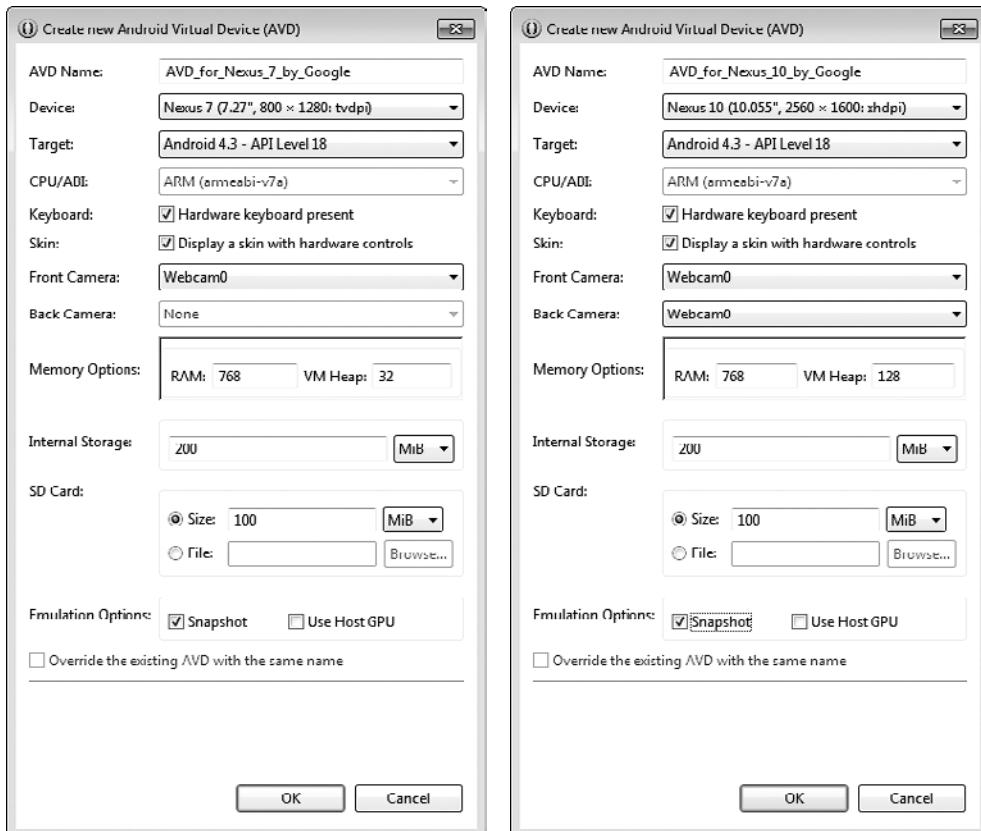


Рис. 0.4. Настройка AVD для планшетов Nexus 7 и Nexus 10

Если разработка ведется на платформе Microsoft Windows, вам также понадобится драйвер Windows USB для устройств Android. В некоторых случаях могут также понадобиться USB-драйверы, предназначенные для конкретного устройства. Со списком USB-драйверов, предназначенных для различных устройств, можно ознакомиться по адресу

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

Как получить примеры кода

Все примеры кода, рассматриваемые в книге, доступны по адресу

<http://www.deitel.com/books/AndroidFP2/>

Если вы еще не зарегистрированы на нашем веб-сайте, перейдите на сайт www.deitel.com и щелкните на ссылке Register. Введите необходимую информацию. Регистрация бесплатна, а введенная вами информация не будет передаваться третьим лицам.

Проверьте правильность адреса электронной почты — после регистрации на сайте вы получите подтверждающее сообщение с верификационным кодом. Работа на сайте *deitel.com* возможна только после того, как вы щелкнете на ссылке верификации в этом сообщении. Настройте ваш клиент электронной почты на разрешение приема электронных сообщений от *deitel.com*, чтобы верификационное письмо не попало в папку нежелательной почты. Мы рассылаем только сообщения, предназначенные для управления учетной записью, если вы не подписались дополнительно на бесплатный бюллетень Deitel® Buzz Online по адресу

www.deitel.com/newsletter/subscribe.html

Посетите сайт *www.deitel.com* и зарегистрируйтесь, щелкнув на ссылке *Login* под нашим логотипом в левом верхнем углу страницы. Откройте страницу *http://www.deitel.com/books/AndroidFP2/*. Щелкните на ссылке *Examples*, чтобы загрузить Zip-архив с примерами на ваш компьютер. Дважды щелкните на файле, чтобы распаковать его. Запомните, где именно будет сохранено распакованное содержимое архива.

О средствах разработчика Android

Google часто обновляет средства разработки для Android. Нередко это создает проблемы с компиляцией даже в том случае, если приложение не содержит ни единой ошибки. Если вы импортировали приложение в Eclipse или Android Studio и оно не компилируется, вероятно, дело в какой-то незначительной проблеме с конфигурацией. Пожалуйста, свяжитесь с нами по электронной почте *deitel@deitel.com* или задайте свой вопрос:

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

Мы поможем вам решить проблему.

Итак, вы установили все необходимые программы и загрузили примеры кода, которые вам понадобятся для изучения разработки Android-приложений и написания собственных программ. Успехов!

1 Введение в Android

В этой главе...

- История Android и Android SDK
- Загрузка приложений из магазина Google Play Store
- Пакеты Android, используемые в книге для создания приложений Android
- Основные концепции объектной технологии
- Ключевые программные продукты, применяемые для разработки приложений Android, в том числе Android SDK, Java SDK, интегрированная среда разработки Eclipse и Android Studio
- Основная документация по Android
- Тестирование приложения Android для рисования на экране в Eclipse
- Характеристики профессиональных приложений Android

1.1. Введение

Добро пожаловать в мир разработки приложений Android! Мы надеемся, что книга «Android для разработчиков. Изд. 2-е» покажется вам познавательной и интересной, и вы не пожалеете о потраченном времени.

Материал книги ориентирован на *программистов Java*. В книге используются только завершенные рабочие приложения, поэтому, даже не зная Java, но имея опыт объектно-ориентированного программирования на другом языке (C#/ Objective-C/Cocoa либо C++ с библиотеками классов), вы сможете быстро освоить излагаемый в книге материал, а также изучить Java и объектно-ориентированное программирование в стиле Java в процессе изучения разработки приложений Android.

Все новые технологии в книге рассматриваются в контексте разработки завершенных рабочих приложений Android, каждое из которых описано в отдельной главе. Мы описываем приложение и *тестируем* его. Затем вкратце описываются ключевые технологии *Eclipse* (интегрированной среды разработки), Java и *Android SDK* (Software Development Kit), используемые для создания приложения. Если разрабатываемые приложения используют графический интерфейс, описывается процесс его *визуального проектирования* в Eclipse. Далее приводятся листинги исходного кода с нумерацией строк и выделением ключевых фрагментов кода. Результаты выполнения кода проиллюстрированы одним-двумя экранными снимками. Код подробно анализируется, причем особое внимание уделяется новым концепциям программирования, использованным в приложении. Исходный код всех приложений, рассматриваемых в книге, может быть загружен с веб-сайта www.deitel.com/books/AndroidFP2/.

Для каждой главы мы также предоставляем версии всех инструкций, специфических для Eclipse, адаптированные для среды разработки Android Studio. Поскольку Android Studio пока существует в ознакомительной версии и быстро развивается, мы разместили инструкции для Android Studio на сайте книги. Это позволит нам поддерживать их в актуальном состоянии.

1.2. Android — мировой лидер в области мобильных операционных систем

Стремительный рост продаж устройств на базе Android открывает выдающиеся возможности перед разработчиками приложений Android.

- ❑ Первое поколение телефонов на базе Android было выпущено в октябре 2008 года. В октябре 2013 года отчет Strategy Analytics показывал, что Android принадлежит

81,3% глобального рынка смартфонов — по сравнению с 13,4% у Apple, 4,1% у Microsoft и 1% у Blackberry.¹

- ❑ По данным отчета IDC, к концу первого квартала 2013 года Android принадлежало 56,5% глобального рынка планшетных устройств — по сравнению с 39,6% у Apple iPad и 3,7% у планшетов на базе Microsoft Windows.²
- ❑ В апреле 2013 года ежедневно активировалось более 1,5 миллиона устройств на базе Android (включая смартфоны, планшеты и т. д.).³
- ❑ Во время работы над книгой было зарегистрировано более миллиарда активированных Android-устройств.⁴
- ❑ На платформе Android сейчас работают смартфоны, планшеты, электронные книги, роботы, реактивные двигатели, спутники NASA, игровые приставки, холодильники, телевизоры, камеры, медицинские устройства, «умные часы», автомобильные информационные системы (для управления радио, GPS, телефонами, терmostатами и т. д.) и многие другие устройства.⁵

1.3. Особенности Android

Одно из главных преимуществ платформы Android — ее открытость. Операционная система Android построена на основе *открытого исходного кода* и находится в свободном распространении. Это позволяет разработчикам получить доступ к исходному коду Android и понять, каким образом реализованы свойства и функции приложений. Любой пользователь может принять участие в совершенствовании операционной системы Android. Для этого достаточно отправить отчет об обнаруженных ошибках (<http://source.android.com/source/report-bugs.html>) либо принять участие в одной из групп дискуссий Open Source Project (<http://source.android.com/community/index.html>). В Интернете доступны различные приложения Android с открытым исходным кодом, предлагаемые компанией Google и рядом других производителей (табл. 1.1). В табл. 1.2 показано, где можно получить исходный код Android, узнать об идеологии, заложенной в основу операционной системы с открытым кодом, и получить лицензионную информацию.

¹ <http://blogs.strategyanalytics.com/WSS/post/2013/10/31/Android-Captures-Record-81-Percent-Share-of-Global-Smartphone-Shipment-in-Q3-2013.aspx>

² <http://www.idc.com/getdoc.jsp?containerId=prUS24093213>

³ <http://www.technobuffalo.com/2013/04/16/google-daily-android-activations-1-5-million>

⁴ <http://venturebeat.com/2013/09/03/android-hits-1b-activations-and-will-be-called-kitkat-in-next-version>

⁵ <http://www.businessweek.com/articles/2013-05-29/behind-the-internet-of-things-is-android-and-its-everywhere>

Таблица 1.1. Ресурсы приложений и библиотек Android с открытым исходным кодом

URL	Описание
http://en.wikipedia.org/wiki/List_of_open_source_Android_applications	Обширный список приложений с открытым исходным кодом, разбитых по категориям (игры, утилиты и пр.)
http://developer.android.com/tools/samples/index.html	Примеры приложений Google для платформы Android; включает свыше 60 приложений и игр (таких, как «Посадка на Луну», «Змейка» и «Крестики-нолики»)
http://github.com/	Репозитарий GitHub позволяет распространять ваши приложения и исходный код, а также участвовать в проектах с открытым кодом других разработчиков
http://sourceforge.net	SourceForge тоже позволяет распространять ваши приложения и исходный код, а также участвовать в проектах с открытым кодом других разработчиков
http://f-droid.org/	Сотни бесплатных и распространяемых с открытым кодом приложений Android, включая блокировщик рекламы Adblock Plus, навигационную систему для общественного транспорта aMetro, экранную клавиатуру AnySoftKeyboard (для нескольких языков), музыкальный проигрыватель Apollo, игру «Китайские шашки», систему контроля веса DroidWeight, «живые обои» Earth Live Wallpaper и многие другие
http://blog.interstellr.com/post/39321551640/14-great-android-apps-that-are-also-open-source	14 приложений Android, распространяемых с открытым кодом, со ссылками на код
http://www.openintents.org/en/libraries	Около 100 библиотек с открытым кодом, расширяющих функциональность приложений
http://www.androidviews.net	Адаптированные элементы графического интерфейса, улучшающие внешний вид приложения
http://www.stackoverflow.com	Сайт Stack Overflow предназначен для публикации вопросов и ответов для программистов. Пользователи могут голосовать, и лучшие ответы занимают первые места

Таблица 1.2. Ресурсы с исходным кодом операционной системы Android

Название	URL
Исходный код Android	http://source.android.com/source/downloading.html
Governance Philosophy	http://source.android.com/about/philosophy.html
Лицензии	http://source.android.com/source/licenses.html
Списки FAQ	http://source.android.com/source/faqs.html

Открытость платформы способствует быстрому обновлению. В отличие от закрытой системы iOS компании Apple, доступной только на устройствах Apple, система Android доступна на устройствах десятков производителей оборудования (OEM, Original Equipment Manufacturer) и телекоммуникационных компаний по всему миру. Все они конкурируют между собой, что идет на пользу конечному потребителю.

Java

При разработке приложений Android используется Java — один из наиболее распространенных языков программирования. Использование Java стало логичным выбором для платформы Android, потому что это мощный, свободный и открытый язык, известный миллионам разработчиков. Опытные программисты Java могут быстро освоить Android-программирование, используя интерфейсы Google Android API (Application Programming Interface) и другие разработки независимых фирм.

Язык Java является объектно-ориентированным, предоставляет разработчикам доступ к мощным библиотекам классов, ускоряющих разработку приложений. Программирование графического интерфейса пользователя является *управляемым событием* — в этой книге приведены примеры приложений, которые реагируют на инициируемые пользователями *события*, такие как *касания экрана*. Помимо непосредственного написания кода приложений, можно воспользоваться средами разработки Eclipse и Android Studio, позволяющими собирать графический интерфейс из готовых объектов, таких как кнопки и текстовые поля, перетаскивая их в определенные места экрана, добавляя подписи и изменения их размеры. Эти среды разработки позволяют быстро и удобно создавать, тестировать и отлаживать приложения Android.

Мультисенсорный экран

Многие современные смартфоны Android сочетают в себе функции мобильных телефонов, интернет-клиентов, MP3-плееров, игровых консолей, цифровых фотоаппаратов и многое другое. Эти портативные устройства оборудованы полноцветными *мультисенсорными экранами*. Простые прикосновения пальцев позволяют легко переключаться между использованием телефона, запуском приложений, воспроизведением музыки, просмотром веб-страниц и т. д. На экране может отображаться клавиатура для ввода электронной почты и текстовых сообщений, а также ввода данных в приложениях (некоторые устройства Android также оснащаются физическими клавиатурами).

Жесты

Мультисенсорный экран позволяет управлять устройством с помощью касаний и *жестов*, как показано в табл. 1.3.

Встроенные приложения

В комплект поставки устройств Android входят различные встроенные приложения, набор которых зависит от устройства, производителя или оператора мобильной

Таблица 1.3. Жесты, применяемые на устройствах Android

Название	Физическое действие	Применение
Касание	Коснитесь один раз экрана	Открытие приложения, «нажатие» кнопки или элемента меню
Двойное касание	Дважды коснитесь экрана	Увеличение или уменьшение масштаба просмотра изображений, карт Google Maps и веб-страниц. Повторное двойное касание возвращает к прежнему масштабу
Длинное нажатие	Нажмите выбранную область экрана и удерживайте палец в этой позиции	Выбор элементов в списке.
Смахивание	Нажмите, быстро проведите пальцем вдоль экрана в нужном направлении, после чего отпустите палец	Прокрутка серии объектов (например, галереи фотографий). Смахивание автоматически останавливается у следующего объекта
Перетаскивание	Нажмите пальцем и перетащите его вдоль экрана	Перемещение объектов или значков либо точная прокрутка веб-страницы или списка
Масштабирование двумя пальцами	Коснитесь экрана двумя пальцами, а потом сведите или разведите их для изменения масштаба	Увеличение или уменьшение масштаба просмотра экрана (увеличение или уменьшение текста и рисунков)

связи. Обычно это приложения Телефон (Phone), Контакты (People), Почта (EMail), Браузер (Browser), Камера (Camera) и ряд других приложений.

Веб-службы

Веб-службы (web services) представляют собой программные компоненты, хранящиеся на одном компьютере, к которым могут обращаться приложения (или другие программные компоненты) с другого компьютера по Интернету. На базе веб-служб могут создаваться мэшапы (mashups), ускоряющие разработку приложений путем комбинирования веб-служб, используемых в различных организациях, с различными типами вводимой информации. Например, сайт 100 Destinations (www.100destinations.co.uk) объединяет фотографии и публикации в Твиттере с картами Google Maps, чтобы вы могли познакомиться с разными странами по фотографиям других пользователей.

На сайте Programmableweb (<http://www.programmableweb.com/>) размещен каталог 9400 API и 7000 мэшапов, а также руководства и примеры кода по самостоятельному созданию мэшапов. В табл. 1.4 перечислены некоторые популярные веб-службы. По данным Programmableweb, самыми популярными API для создания мэшапов являются Google Maps, Twitter и YouTube. В главе 4 мы воспользуемся веб-службами Twitter.

Таблица 1.4. Некоторые популярные веб-службы (www.programmableweb.com/apis/directory/1?sort=mashups)

Веб-службы	Применение
Google Maps	Картографические службы
Twitter	Микроблоги
YouTube	Поиск видео
Facebook	Социальные сети
Instagram	Публикация фотографий в Интернете
Foursquare	Мобильная регистрация местоположения
LinkedIn	Социальные бизнес-сети
Groupon	Социальная коммерция
Netflix	Аренда фильмов
eBay	Интернет-аукционы
Wikipedia	Коллективная энциклопедия
PayPal	Платежи
Last.fm	Интернет-радио
Amazon eCommerce	Покупка книг и множества других продуктов
Salesforce.com	Управление отношениями с клиентами и партнерами
Skype	Интернет-телефония
Microsoft Bing	Поисковая машина Bing
Flickr	Публикация фотографий в Интернете
Zillow	Цены на недвижимость
Yahoo Search	Поисковая машина Yahoo!
WeatherBug	Прогноз погоды

1.4. Операционная система Android

Операционная система Android была разработана компанией Android, Inc., которая в 2005 году была приобретена компанией Google. В ноябре 2007 был сформирован консорциум Open Handset Alliance™, который в настоящее время объединяет уже 84 компании (http://www.openhandsetalliance.com/oha_members.html). В задачи этого консорциума входит разработка, сопровождение и развитие Android, внедрение инноваций в мобильных технологиях, а также повышение удобства работы с устройствами Android при одновременном снижении затрат.

Имена версий Android

Каждой новой версии Android присваивалось название, соответствующее названию какого-либо десерта (табл. 1.5).

Таблица 1.5. Номера версий Android и соответствующие названия

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

1.4.1. Android 2.2 (Froyo)

Версия Android 2.2, которая также называется «Froyo» (замороженный йогурт), вышла в мае 2010 года. В ней появилась поддержка внешней памяти, позволявшая хранить приложения на внешнем устройстве (вместо внутренней памяти устройства Android). С помощью службы C2DM (Android Cloud to Device Messaging, обмен сообщениями с устройствами через облако) разработчики приложений могут использовать программы и данные, хранящиеся в «облаке» — то есть на удаленных компьютерах (серверах) в Интернете, вместо того чтобы хранить их на настольном компьютере, ноутбуке или мобильном устройстве. Облачные технологии предоставляют гибкие средства наращивания или сокращения вычислительных ресурсов под конкретные потребности в любой момент времени. Это делает их более экономически эффективными по сравнению с приобретением дорогостоящего оборудования, гарантирующего достаточную вычислительную мощность и емкость памяти для периодических пиковых нагрузок. Android C2DM позволяет разработчикам приложений пересыпать данные со своих серверов на приложения, установленные на устройствах Android, даже если эти приложения в данный момент *не активны*. Сервер оповещает приложения, предлагая им подключиться к нему для приема обновления или пользовательских данных¹. Сейчас технология C2DM считается устаревшей — на смену ей пришла технология Google Cloud Messaging.

Информация о других новых возможностях Android 2.2 — графических средствах OpenGL ES 2.0, медиафреймворках и т. д. — доступна по адресу

<http://developer.android.com/about/versions/android-2.2-highlights.html>

¹ <http://code.google.com/android/c2dm/>

1.4.2. Android 2.3 (Gingerbread)

Версия Android 2.3 Gingerbread («имбирный пряник»), появившаяся в декабре 2010 года, предлагает новые возможности для пользователя — более удобную клавиатуру, улучшенные навигационные функции, более эффективное использование батареи и ряд других преимуществ. Также в ней добавились новые средства разработчика для передачи данных (например, технологии, упрощающие телефонные звонки и ответы на них в приложениях), мультимедийные технологии (новые API для работы со звуком и графикой) и игровые средства (повышение быстродействия и новые датчики — например, гироскоп для обработки перемещений в пространстве).

Одним из самых значительных нововведений Android 2.3 стала поддержка NFC (Near-Field Communication) — технологии беспроводной высокочастотной связи малого радиуса действия, которая дает возможность обмена данными между устройствами, находящимися на расстоянии нескольких сантиметров. Уровень поддержки и функциональность NFC зависит от конкретного устройства. NFC может использоваться для электронных платежей (например, когда вы прикасаетесь устройством Android с поддержкой NFS к платежному блоку на торговом автомате), передачи данных (контактов, фотографий и т. д.), сопряжении устройств и аксессуаров и т. д.

Полный список средств разработчика Android 2.3 доступен по адресу

<http://developer.android.com/about/versions/android-2.3-highlights.html>

1.4.3. Android 3.0–3.2 (Honeycomb)

В версии Android 3.0 Honeycomb («пчелиные соты») появились усовершенствования пользовательского интерфейса, предназначенные для устройств с большим экраном (то есть планшетов): усовершенствованная клавиатура для более эффективного ввода данных, трехмерный пользовательский интерфейс, упрощение переходов между экранами в приложении и ряд других улучшений. Некоторые новые средства разработчика Android 3.0:

- ❑ фрагменты, описывающие части пользовательского интерфейса приложения, которые могут объединяться в один экран или использоваться на разных экранах;
- ❑ постоянная панель действий в верхней части экрана, предоставляющая дополнительные средства для взаимодействия с приложениями;
- ❑ возможность добавления макетов для большого экрана к существующим приложениям, рассчитанным на малые экраны, чтобы оптимизировать приложение для разных вариантов размера экрана;
- ❑ привлекательный и более функциональный пользовательский интерфейс «Holo», имитирующий «голографический» эффект;
- ❑ усовершенствованные графические и мультимедийные возможности;

- ❑ возможность использования многоядерных процессоров для повышения быстродействия;
- ❑ улучшенная поддержка Bluetooth (например, возможность проверки подключенных устройств — гарнитуры, клавиатуры и т. д.);
- ❑ фреймворк для анимации пользовательского интерфейса или графических объектов.

Список средств Android 3.0 для пользователей и разработчиков, а также платформенных технологий доступен по адресу

<http://developer.android.com/about/versions/android-3.0-highlights.html>

1.4.4. Android Ice Cream Sandwich

Версия Android 4.0 Ice Cream Sandwich («сэндвич с мороженым»), вышедшая в 2011 году, объединила Android 2.3 (Gingerbread) и Android 3.0 (Honeycomb) в одну операционную систему, предназначенную для использования на всех устройствах Android. Эта версия Android позволяет включить функции, поддерживаемые в версии Honeycomb и ранее доступные только для планшетов («голографический» интерфейс пользователя, новый лаунчер и т. д.), в приложения для смартфонов. Так обеспечивается простое масштабирование приложений, позволяющее использовать их на различных устройствах. В версии Ice Cream Sandwich также появился ряд новых API для улучшения обмена данными между устройствами, технологий для пользователей с ограниченными возможностями (например, с ослабленным зрением), поддержки социальных сетей и т. д. (табл. 1.6). За полным списком API для Android 4.0 обращайтесь по адресу

<http://developer.android.com/about/versions/android-4.0.html>

Таблица 1.6. Некоторые средства разработчика Android Ice Cream Sandwich (<http://developer.android.com/about/versions/android-4.0.html>)

Функция	Описание
Face detection (Распознавание лиц)	С помощью камеры совместимые устройства могут определять положение глаз, носа и рта пользователя. Камера может также отслеживать направление взгляда пользователя, позволяя создавать приложения, которые изменяют перспективу в зависимости от направления взгляда пользователя
Virtual camera operator (Виртуальный оператор камеры)	В процессе видеосъемки камера автоматически фокусируется на говорящем человеке
Android Beam	Технология Android Beam позволяет передавать контент (контакты, фото, видео) между двумя соприкасающимися устройствами Android

Функция	Описание
Wi-Fi Direct	Wi-Fi P2P (Peer-to-Peer) API позволяет связать несколько устройств Android по Wi-Fi. В этом случае беспроводная связь между устройствами может осуществляться на большем расстоянии, чем при использовании Bluetooth
Social API	Получение доступа и обмен контактными данными между социальными сетями и приложениями (с разрешения пользователя)
Calendar API	Добавление событий, обмен событиями между приложениями, управление сигналами, участниками и т. д.
Accessibility API	Новые вспомогательные API повышают доступность ваших приложений для людей с ограниченными возможностями (ослабленным зрением и т. д.). Режим «Explore-by-touch» позволяет слабовидящим пользователям прикоснуться к любой точке экрана и прослушать голосовое описание контента
Android@Home framework (Фреймворк Android@Home)	Обеспечивает создание приложений Android, управляющих бытовыми устройствами в доме пользователя: терmostатами, системами полива, освещения и т. д.
Bluetooth Health Devices	Создание приложений, взаимодействующих с медицинскими устройствами с поддержкой Bluetooth: весами, пульсометрами и т. д.

1.4.5. Android 4.1–4.3 (Jelly Bean)

Версия Android Jelly Bean («жевательная конфета»), выпущенная в 2012 году, включает поддержку внешних экранов, усовершенствованные средства безопасности, улучшенное оформление (например, виджеты приложений с изменяемыми размерами и крупные оповещения) и функции более плавного переключения между приложениями и экранами (табл. 1.7). За полным списком возможностей Jelly Bean обращайтесь по адресу

<http://developer.android.com/about/versions/jelly-bean.html>

Таблица 1.7. Некоторые возможности Android Jelly Bean (<http://developer.android.com/about/versions/jelly-bean.html>)

Функция	Описание
Android Beam	Технология Android Beam может использоваться для простого связывания смартфона или планшета с беспроводными динамиками на базе Bluetooth® или специальными наушниками

Таблица 1.7 (окончание)

Функция	Описание
Виджеты блокировки экрана	Создание виджетов, которые отображаются на экране пользователя во время блокировки устройства, или изменение существующих виджетов главного экрана, чтобы они тоже были видны во время блокировки
Photo Sphere	API для работы с новыми панорамными фотографиями позволяют создавать обзорные фото, сходные с теми, которые используются в режиме Google Maps Street View
Daydreams	Интерактивные заставки, которые активизируются при зарядке или нахождении устройства в док-станции. Заставки Daydreams способны воспроизводить аудио и видео, а также реагировать на действия пользователя
Поддержка языков	Новые функции, ориентированные на пользователей из других стран (двусторонний вывод текста — слева направо и справа налево, клавиатуры для других языков, дополнительные раскладки и т. д.)
Средства разработчика	Новые средства диагностики и отладки — например, возможности отправки отчетов об ошибках, содержащие экранные снимки и информацию о состоянии устройства

1.4.6. Android 4.4 (KitKat)

Версия Android 4.4 KitKat, выпущенная в октябре 2013 года, включает ряд усовершенствований, обеспечивающих работу операционной системы на любых устройствах Android, включая старые устройства с ограниченной памятью, особенно популярные в развивающихся странах.¹

Переход большего количества пользователей на KitKat сократит «фрагментацию» версий Android на рынке. Фрагментация создавала проблемы для разработчиков, которые были вынуждены проектировать приложения для разных версий операционной системы или ограничивать круг потенциальных пользователей, разрабатывая приложение для конкретной версии операционной системы.

Android KitKat также включает усовершенствования безопасности и доступности, улучшенные графические и мультимедийные возможности, средства анализа памяти и т. д. В табл. 1.8 перечислены некоторые из ключевых новых возможностей KitKat. За полным списком обращайтесь по адресу

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

¹ <http://techcrunch.com/2013/10/31/android-4-4-kitkat-google/>

Таблица 1.8. Некоторые возможности Android KitKat
(<http://developer.android.com/about/versions/kitkat.html>)

Функция	Описание
Режим погружения (Immersive mode)	Панель состояния у верхнего края экрана и кнопки меню у нижнего края можно скрыть, чтобы ваши приложения могли занимать большую часть экрана. Пользователь может вызвать панель состояния, смахнув вниз от верхнего края экрана, а системную панель (с кнопками Back, Home и Recent Apps) — смахиванием вверх от нижнего края
Инфраструктура печати	Встраивание функциональности печати в приложения, включая поиск доступных принтеров по Wi-Fi или в облаке, выбор размера бумаги и печатаемых страниц
Доступ к документам	Создание провайдеров хранения документов, позволяющих просматривать, создавать и редактировать файлы (например, документы и изображения) в разных приложениях
Сервис SMS	Создание приложений SMS (Short Message Service) и MMS (Multimedia Messaging Service) с использованием нового поставщика SMS и API. Теперь пользователи могут выбирать приложение передачи сообщений по умолчанию
Переходы	Новая инфраструктура упрощает создание переходных анимаций
Запись с экрана	Запись видеороликов работающих приложений для создания интерактивных руководств и маркетинговых материалов
Улучшения доступности	Captioning Manager API позволяет приложению проверить пользовательские настройки субтитров (язык, оформление текста и т. д.)
Chromium WebView	Поддержка новейших стандартов вывода веб-контента, включая HTML5, CSS3 и ускоренную версию JavaScript
Датчик и счетчик шагов	Создание приложений, которые определяют, идет ли пользователь, бежит или поднимается по лестнице, и подсчитывают количество сделанных шагов
Host Card Emulator (HCE)	Технология HCE позволяет любому приложению выполнять безопасные транзакции NFC (например, мобильные платежи) без обязательного присутствия элемента безопасности на SIM-карте под управлением оператора мобильной связи

1.5. Загрузка приложений из Google Play

На время написания этой книги в Google Play было доступно более миллиона приложений, и это число быстро росло.¹ Приложения можно загрузить из приложения Play Store, установленном на устройстве. Вы также можете войти в учетную

¹ en.wikipedia.org/wiki/Google_Play

запись Google Play по адресу: <http://play.google.com> через браузер, а затем выбрать устройство Android, на котором должно быть установлено приложение. Приложение загружается на устройство через подключение WiFi или 3G/4G. В главе 9 будут рассмотрены другие магазины, предоставляющие приложения (бесплатно или за деньги), цены и т. д.

Таблица 1.9. Некоторые популярные Android-приложения в Google Play

Категория	Некоторые популярные приложения в этой категории
Книги и справочники	Kindle, Wikipedia, Audible for Android, Google Play Books
Бизнес	Office Suite Pro 7, Job Search, Square Register, GoToMeeting
Комиксы	ComicRack, Memedroid Pro, Marvel Comics, Comic Strips
Связь	Facebook Messenger, Skype™, GrooVe IP
Образование	Duolingo: Learn Languages Free, TED, Mobile Observatory
Развлечения	SketchBook Mobile, Netflix, Fandango® Movies, iFunny :)
Финансы	Mint.com Personal Finance, Google Wallet, PayPal
Игры: аркады и экшны	Minecraft—Pocket Edition, Fruit Ninja, Angry Birds
Игры: головоломки	Where's My Water?, Draw Something, Can You Escape
Игры: азартные	Solitaire, Slots Delux, UNO™ & Friends, DH Texas Poker
Игры: логические	Candy Crush Saga, Hardest Game Ever 2, Game Dev Story
Здоровье и фитнес	RunKeeper, Calorie Counter, Workout Trainer, WebMD®
Стиль жизни	Zillow Real Estate, Epicurious Recipe App, Family Locator
Живые обои	PicsArt, GO Launcher EX, Beautiful Widgets Pro
Видео и мультимедиа	MX Player, YouTube, KeepSafe Vault, RealPlayer®
Медицина	Epocrates, ICE: In Case of Emergency, Medscape®
Музыка и аудио	Pandora®, Shazam, Spotify, Ultimate Guitar Tabs & Chords
Новости и журналы	Flipboard, Pulse News, CNN, Engadget, Drippler
Персонализация	Beautiful Widgets Pro, Zedge™, GO Launcher EX
Фотография	Camera ZOOM FX, Photo Grid, InstaPicFrame for Instagram
Производительность	Adobe® Reader®, Dropbox, Google Keep, SwiftKey Keyboard
Покупки	eBay, Amazon Mobile, Groupon, The Coupons App
Социальные сети	Facebook®, Instagram, Vine, Twitter, Snapchat, Pinterest
Спорт	SportsCenter for Android, NFL'13, Team Stream™
Инструменты	Titanium Backup PRO, Google Translate, Tiny Flashlight®
Транспорт	Uber, Trapster, Lyft, Hailo™, Ulysse Speedometer
Путешествия	Waze, GasBuddy, KAYAK, TripAdvisor, OpenTable®
Погода	WeatherBug, AccuWeather, The Weather Channel
Виджеты	Zillow, DailyHoroscope, Starbucks, Family Locator

1.6. Пакеты

В Android используется целая коллекция *пакетов*, являющихся группами связанных предварительно определенных классов. Некоторые из пакетов специфичны для Android, другие относятся к Java и Google. Пакеты обеспечивают удобный доступ к функциям операционной системы Android, а также включение этих функций в приложения. Они упрощают создание Android-приложений, использующих особенности оформления и соответствующие стилевым рекомендациям Android (<http://developer.android.com/design/index.html>). В табл. 1.10 перечислены пакеты, которые будут рассмотрены в книге. Полный перечень пакетов Android приводится на веб-сайте developer.android.com/reference/packages.html.

Таблица 1.10. Пакеты Android и Java, используемые в книге, с указанием главы, в которой они впервые встречаются

Пакет	Описание
android.app	Включает классы высокого уровня в модели приложения Android (приложение Tip Calculator в главе 3)
android.content	Добавление и публикация данных на устройстве (приложение Cannon Game в главе 6)
android.content.res	Классы, предназначенные для обеспечения доступа к ресурсам приложения (например, к медиафайлам, цветам, рисункам и прочим ресурсам), а также к информации о конфигурации устройства, влияющей на поведение приложения (приложение Flag Quiz в главе 5)
android.database	Обработка данных, возвращаемых провайдером контента (приложение Address Book в главе 8)
android.database.sqlite	Управление базами данных SQLite для частных баз данных (приложение Address Book в главе 8)
android.graphics	Графические инструменты, применяемые для рисования на экране (приложение Flag Quiz в главе 5 и приложение Doodlz в главе 7)
android.hardware	Поддержка аппаратного обеспечения устройств (приложение Doodlz в главе 7)
android.media	Классы, предназначенные для обработки медиаинтерфейсов аудио и видео (приложение Cannon Game в главе 6)
android.net	Классы доступа к сети (приложение Twitter® Searches в главе 4)
android.os	Сервис операционной системы (приложение Tip Calculator в главе 3)
android.preference	Работа с пользовательскими настройками приложения (приложение Flag Quiz в главе 5)
android.provider	Работа с провайдерами контента Android (приложение Doodlz в главе 7)

Таблица 1.10 (окончание)

Пакет	Описание
android.support.v4.print	Функции Android Support Library для использования инфраструктуры печати Android 4.4 (приложение Doodlz в главе 7)
android.text	Вывод и отслеживание текста на экране устройства (приложение Tip Calculator в главе 3)
android.util	Служебные методы и вспомогательные средства XML (приложение Cannon Game в главе 6)
android.widget	Классы интерфейса пользователя, предназначенные для виджетов (приложение Tip Calculator в главе 3)
android.view	Классы интерфейса пользователя, предназначенные для взаимодействия с пользователем и построения макетов (приложение Favorite Twitter® Searches в главе 4)
java.io	Потоки, сериализация и доступ к файловой системе для устройств ввода и вывода (приложение Flag Quiz в главе 5)
java.text	Классы форматирования текста (приложение Twitter® Searches в главе 4)
java.util	Вспомогательные классы (приложение Twitter® Searches в главе 4)
android.graphics.drawable	Классы, предназначенные только для отображаемых на экране элементов (например, градиентов) (приложение Flag Quiz в главе 5)

1.7. Android Software Development Kit (SDK)

В состав Android SDK включены инструменты, необходимые для создания Android-приложений. Этот набор инструментов доступен на веб-сайте Android Developers (на бесплатной основе). В разделе «Подготовительные действия» предыдущей главы описан порядок загрузки из Интернета всех инструментов, применяемых для разработки Android-приложений, включая Java SE, пакет Android SDK/ADT (включающий интегрированную среду разработки Eclipse) и Android Studio IDE.

Android SDK/ADT

Пакет Android SDK/ADT (включающий Eclipse) — самая распространенная интегрированная среда разработки Android-приложений. Некоторые разработчики предпочитают использовать для создания Android-приложений только текстовый редактор и инструменты командной строки. Среда разработки Eclipse включает:

- ❑ редактор кода с поддержкой цветового выделения синтаксиса и нумерации строк;
- ❑ автоматические отступы и автозавершение (то есть подсказки при вводе кода);
- ❑ отладчик;

-
- ❑ система контроля версий;
 - ❑ поддержка рефакторинга.

Среда Eclipse применяется в разделе 1.9 для тестирования приложения Doodlz. Начиная с главы 2, где разрабатывается приложение Welcome, мы будем использовать среду Eclipse для построения приложений.

Android Studio

Android Studio, новая интегрированная среда разработки Android на основе среды JetBrains IntelliJ IDEA (<http://www.jetbrains.com/idea/>), была анонсирована в 2013 году. Компания Google считает ее основной средой разработки Android-приложений в будущем. На момент написания книги среда Android Studio существовала только в виде ознакомительной версии — многие ее возможности все еще находились в процессе разработки. Для каждой главы на сайте книги приведены версии инструкций для Eclipse, адаптированные для Android Studio:

<http://www.deitel.com/books/AndroidFP2>

За дополнительной информацией об Android Studio, установке среды и переходе на нее с Eclipse, посетите страницу

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

Плагин Android Development Tools (ADT) для Eclipse

Плагин ADT (Android Development Tools, Инструменты разработки Android-приложений, часть пакета Android SDK/ADT) для Eclipse — расширение интегрированной среды разработки Eclipse. С помощью этого подключаемого модуля можно создавать, выполнять и отлаживать приложения Android, экспортить их для дальнейшего распространения (например, выгружать в Google Play) и выполнять ряд других операций. Плагин ADT также включает визуальный конструктор графического интерфейса — компоненты графического интерфейса перетаскиваются и размещаются в нужных местах без написания кода. Более подробно плагин ADT рассматривается в главе 2.

Эмулятор Android

Эмулятор Android, включенный в состав Android SDK, позволяет смоделировать среду для запуска приложений Android под управлением Windows, Mac OS X либо Linux. Эмулятор отображает вполне реалистичное окно интерфейса пользователя Android. Он особенно полезен, если у разработчика нет доступа к устройствам Android для прямого тестирования. Безусловно, перед отправкой в Google Play приложения желательно протестировать на различных Android-устройствах.

Перед запуском приложения на выполнение следует создать *AVD* (Android Virtual Device, Виртуальное устройство Android). Это устройство определяет характеристики реального устройства Android, на котором нужно тестировать приложения: аппаратное обеспечение, системный образ, размер экрана, хранилище данных и ряд других характеристик. Если нужно тестировать приложения на нескольких

устройствах Android, необходимо создать отдельные AVD для каждого уникального физического устройства или воспользоваться сервисом (таким, как *testdroid.com* или *appthwack.com*), позволяющим провести тестирование на многих устройствах.

Большинство экранных снимков в книге было сделано именно в эмуляторе, а не на реальных устройствах Android. В эмуляторе можно имитировать большинство жестов Android (табл. 1.11) и элементов управления (табл. 1.12), используя клавиатуру и мышь компьютера. Конечно, возможности воспроизведения жестов с помощью эмулятора несколько ограничены, поскольку компьютер не в состоянии смоделировать все аппаратные функции Android. Например, чтобы протестировать GPS-приложения в эмуляторе, нужно создать файлы, имитирующие данные GPS. Хотя можно смоделировать изменения ориентации (переключение в *портретный/альбомный* режим), для моделирования показаний *акселерометра* (это устройство оценивает ориентацию и наклон устройства) необходимы функции, отсутствующие в эмуляторе. Существует эмулятор *Sensor Simulator*, который может использоваться для отправки AVD смоделированной информации датчиков для тестирования этой и других функций:

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

Таблица 1.11. Имитация жестов Android на эмуляторе

Жест	Действие в эмуляторе
Касание	Щелкните кнопкой мыши (приложение Tip Calculator в главе 3)
Двойное касание	Дважды щелкните кнопкой мыши (приложение Cannon Game в главе 6)
Длинное нажатие	Щелкните кнопкой мыши и удерживайте ее
Перетаскивание	Щелкните левой кнопкой мыши, удерживайте ее и переместите мышь (приложение Cannon Game в главе 6)
Смахивание	Щелкните кнопкой мыши и, удерживая ее, переместите указатель мыши в направлении смахивания, а потом отпустите кнопку мыши (приложение Address Book в главе 8)
Масштабирование двумя пальцами	Нажмите и удерживайте Ctrl. Появятся две окружности, имитирующие касание экрана двумя пальцами. Переместите окружности в начальную позицию, щелкните кнопкой мыши и, удерживая ее, переместите окружности в конечную позицию

Таблица 1.12. Имитация органов управления Android в эмуляторе
(имитация других органов управления описана на сайте
<http://developer.android.com/tools/help/emulator.html>)

Орган управления устройства Android	Действие в эмуляторе
Back (Назад)	Esc
Вызов/набор номера	F3
Камера	Ctrl-KEYPAD_5, Ctrl-F3

Орган управления устройства Android	Действие в эмуляторе
Завершить вызов	F4
Home (Домой)	Home button
Меню (левая программаная кнопка)	F2 or Page Up button
Питание	F7
Поиск	F5
* (правая программаная кнопка)	Shift-F2 or Page Down button
Вращение влево	KEYPAD_7, Ctrl-F11
Вращение вправо	KEYPAD_9, Ctrl-F12
Включение/выключение сети мобильной связи	F8
Кнопка увеличения громкости	KEYPAD_PLUS, Ctrl-F5
Кнопка уменьшения громкости	KEYPAD_MINUS, Ctrl-F6

В табл. 1.13 перечислены функциональные аспекты Android, недоступные в эмуляторе. Впрочем, вы всегда можете загрузить свое приложение на устройство Android для их тестирования. Мы начнем создавать виртуальные устройства AVD и использовать эмулятор для разработки Android-приложений в приложении Welcome главы 2.

Таблица 1.13. Функциональность Android, недоступная в эмуляторе
(<http://developer.android.com/tools/devices/emulator.html>)

Функциональность Android, недоступная в эмуляторе
Реальные телефонные звонки (эмодулятор позволяет только моделировать их)
Bluetooth
Подключения USB
Подключение гарнитуры
Определение состояния подключения телефона
Определение текущего уровня батареи или состояния зарядки
Обнаружение вставки/извлечения SD-карты
Датчики (акселерометр, барометр, компас, датчик освещенности, датчик расстояния)

1.8. Краткий обзор объектно-ориентированного программирования

В программировании для Android используются объектно-ориентированные технологии, поэтому в данном разделе мы приведем обзор объектных технологий. Все эти концепции будут часто встречаться в книге.

При разработке современных и эффективных программ достаточно трудно выполнить такие требования, как быстрота разработки, правильность работы и экономичность. *Объекты* (а точнее, как мы увидим в главе 3, — *классы*, на основе которых создаются объекты) по сути представляют собой *повторно используемые* программные компоненты. В качестве объектов может использоваться дата, время, видео, человек, автомобиль и другие предметы материального мира. Практически каждое *существительное* может быть адекватно представлено программным объектом в понятиях *атрибутов* (например, имя, цвет и размер) и *поведений* (например, вычисление, перемещение и передача данных). Разработчики программ видят, что использование модульной структуры и объектно-ориентированного проектирования при разработке приложений повышает продуктивность работы. Этот подход пришел на смену применявшемуся ранее структурному программированию — объектно-ориентированный код проще понять и изменить.

1.8.1. Автомобиль как объект

Чтобы лучше понять суть объектов и их содержимого, воспользуемся простой аналогией. Представьте себе, что вы *находитесь за рулем автомобиля и нажимаете педаль газа, чтобы набрать скорость*. Что должно произойти до того, как вы получите такую возможность? Прежде чем вы *поместите автомобиль*, кто-то должен его *спроектировать*. Изготовление любого автомобиля начинается с инженерных чертежей, которые подробно описывают устройство автомобиля. В частности, на этих чертежах показано устройство педали акселератора. За этой педалью *скрываются* сложные механизмы, которые непосредственно ускоряют автомобиль — подобно тому, как педаль тормоза *скрывает* механизмы, тормозящие автомобиль, а руль *скрывает* механизмы поворота. Благодаря этому люди, не имеющие понятия о внутреннем устройстве автомобиля, могут легко им управлять.

Подобно тому как невозможно готовить пищу на кухне, которая лишь изображена на листе бумаги, нельзя водить автомобиль, существующий лишь в чертежах. Прежде чем вы сядете за руль машины, ее нужно *построить* на основе инженерных чертежей. Воплощенный в металле автомобиль имеет *реальную* педаль газа, с помощью которой он может ускоряться, но и это не все — он не может это делать самостоятельно (к счастью!), а только после того, как водитель *нажмет* на педаль.

1.8.2. Методы и классы

Воспользуемся примером с автомобилем для демонстрации некоторых ключевых концепций объектно-ориентированного программирования. Для выполнения операции в программе требуется *метод*, в котором «скрываются» инструкции программы, непосредственно выполняющие операцию. Метод скрывает эти инструкции от пользователя подобно тому, как педаль газа автомобиля скрывает от водителя механизмы, вызывающие ускорение автомобиля. Программная

единица, именуемая классом, включает методы, выполняющие задачи класса. Например, класс, представляющий банковский счет, может включать три метода, один из которых пополняет счет, второй — снимает средства со счета, а третий — запрашивает текущий баланс. С концептуальной точки зрения класс подобен инженерному чертежу, в котором изображено устройство педали газа, рулевого колеса и других механизмов.

1.8.3. Создание экземпляра класса

Водитель не сядет за руль автомобиля, пока его не *построят* по чертежам, так и *построение объекта* класса необходимо для выполнения задач, определяемых методами класса. Этот процесс называется *созданием экземпляра*. Полученный при этом объект называется *экземпляром класса*.

1.8.4. Повторное использование

На основе одних и тех же чертежей можно создать много автомобилей, а на основе одного класса можно создать много объектов. Использование существующих классов для создания новых классов экономит время и силы разработчика. *Повторное использование* также облегчает создание более надежных и эффективных систем, поскольку ранее созданные классы и компоненты обычно проходят тщательное *тестирование, отладку и оптимизацию*. Подобно тому как концепция использования взаимозаменяемых частей легла в основу промышленной революции, повторно используемые классы играют ключевую роль в программной революции, которая была инициирована внедрением объектных технологий.

1.8.5. Сообщения и вызовы методов

Когда вы ведете машину, нажатие педали газа отправляет автомобилю *сообщение* с запросом на выполнение определенной задачи (ускорение автомобиля). Подобным же образом отправляются сообщения объекту. Каждое сообщение представляется вызовом метода, который «сообщает» методу объекта о необходимости выполнения некоторой задачи. Например, программа может вызвать метод `deposit` объекта банковского счета, чтобы пополнить банковский счет.

1.8.6. Атрибуты и переменные экземпляра класса

Любой автомобиль помимо возможности выполнять определенные операции также обладает *атрибутами*, такими как цвет, количество дверей, запас топлива в баке, показания спидометра и одометра. По аналогии с операциями, атрибуты автомобиля представляются на инженерных диаграммах (в качестве атрибутов

автомобиля могут выступать одометр и указатель уровня бензина). При вождении автомобиля его атрибуты перемещаются вместе с ним. Каждый автомобиль содержит *собственный* набор атрибутов. Например, каждый автомобиль «знает» о том, сколько бензина осталось в его баке, но ему ничего не известно о запасах горючего в баках других автомобилей.

Объект, как и автомобиль, имеет собственный набор атрибутов, которые он «переносит» с собой при использовании этого объекта в программах. Эти атрибуты определяются как часть объекта класса. Например, объект `bankaccount` имеет атрибут баланса, представляющий количество средств на банковском счете. Каждый объект `bankaccount` «знает» о количестве средств на собственном счете, но ничего не «знает» о размерах *других* банковских счетов. Атрибуты определяются с помощью других *переменных экземпляра* класса.

1.8.7. Инкапсуляция

Классы *инкапсулируют* атрибуты и методы в объекты (атрибуты и методы объекта между собой тесно связаны). Объекты могут обмениваться информацией между собой, но обычно они не «знают» о деталях реализации других объектов, которые скрыты внутри самих объектов. Подобное сокрытие информации жизненно важно в практике хороших программных архитектур.

1.8.8. Наследование

С помощью наследования можно быстро и просто создать новый класс объектов. При этом новый класс наследует характеристики существующего класса, которые при этом могут частично изменяться. Также в новый класс добавляются уникальные характеристики, присущие только этому классу. Если вспомнить аналогию с автомобилем, «трансформер» является объектом более обобщенного класса «автомобиль», у которого может подниматься или опускаться крыша.

1.8.9. Объектно-ориентированный анализ и проектирование

А теперь ответьте на вопрос, каким образом вы собираетесь программировать? Скорее всего, таким же образом, как и большинство других программистов, — включите компьютер и начнете вводить исходный код программы. Подобный подход годится при создании маленьких программ, но что делать в том случае, когда приходится создавать крупный программный комплекс, который, например, управляет тысячами банкоматов крупного банка? Либо если вам придется возглавлять команду из 1000 программистов, занятых разработкой системы управления воздушным движением следующего поколения? В таких крупных и сложных проектах нельзя просто сесть за компьютер и начать набирать код.

Чтобы выработать наилучшее решение, следует провести детальный анализ *требований* к программному проекту (то есть определить, *что* должна делать система) и разработать архитектуру, которая будет соответствовать этим требованиям (то есть определить, *как* система будет выполнять свои задачи). В идеале перед началом создания кода следует выполнить эту процедуру и тщательно проанализировать проект (либо поручить выполнение этой задачи другим профессионалам). Если в ходе выполнения этого процесса происходит анализ и проектирование системы с применением объектно-ориентированного подхода, значит, мы имеем дело с процессом объектно-ориентированного анализа и проектирования (ООАП). Языки программирования, подобные Java, называются объектно-ориентированными. Программирование на таких языках, называемое *объектно-ориентированным программированием* (ООП), реализует объектно-ориентированные проекты в виде работоспособных систем.

1.9. Тестирование приложения Doodlz на виртуальном устройстве AVD

В этом разделе вы запустите на выполнение и будете «общаться» со своим первым приложением Android. С помощью приложения Doodlz пользователь может «рисовать» на экране, выбирая различные цвета и кисти разных размеров в *меню* приложения. Рассматривать код приложения не нужно — мы построим это приложение и изучим его код в главе 7.

Следующая пошаговая инструкция показывает, как импортировать проект приложения в среду Eclipse и протестировать приложение на виртуальном устройстве (AVD) для Nexus 4 (создание и настройка AVD описаны в разделе «Подготовка», находящемся во вводной части книги). Позднее будет рассмотрено, каким образом можно выполнять это приложение на реальном устройстве Android. Когда приложение выполняется в AVD, для создания нового рисунка достаточно «простирать пальцем» по экрану; «касания» имитируются нажатием кнопки мыши.

Android SDK/ADT и Android Studio

Экранные снимки, используемые в качестве иллюстраций, были созданы на компьютере, на котором установлены Windows 7, Java SE 7 SDK и Android SDK/ADT (см. раздел «Подготовка»). Поскольку Android Studio существует только в ознакомительной версии и стремительно развивается, инструкции по запуску и тестированию приложения в Android Studio размещены на сайте книги:

www.deitel.com/books/AndroidFP2

Это позволит нам оперативно обновлять инструкции в ответ на изменения, вносимые разработчиками из Google. Android SDK/ADT и Android Studio используют один эмулятор Android, поэтому когда приложение заработает в AVD, дальнейшие действия идентичны.

1.9.1. Запуск приложения Doodlz на AVD для смартфона Nexus 4

Чтобы протестировать приложение Doodlz, выполните следующие действия.

- Проверка конфигурации системы.** Убедитесь в том, что ваш компьютер настроен в полном соответствии с требованиями, изложенными в разделе «Подготовка», находящемся во вводной части книги.
- Запуск среды Eclipse.** Откройте папку `eclipse` в папке, в которой был установлен пакет Android SDK/ADT, и дважды щелкните на значке Eclipse ({} или ⚙) в зависимости от платформы).
- Выбор папки для хранения рабочей области.** Когда на экране появится окно `Workspace Launcher`, укажите, где должны храниться создаваемые приложения, и нажмите **OK**. Мы использовали вариант по умолчанию — папку с именем `workspace` каталога текущего пользователя. *Рабочей областью* (*workspace*) называется набор проектов, каждый из которых обычно представляет приложение или библиотеку, совместно используемую разными приложениями. Каждая рабочая область пространства также обладает собственными настройками — например, определяющими, где должны отображаться те или иные служебные окна Eclipse. Вы можете открыть несколько рабочих областей и переключаться между ними для разных задач разработки — например, создать разные рабочие области для разработки Android-приложений, Java-приложений и веб-приложений, каждое из которых настраивается отдельно от других. Если Eclipse запускается первый раз, появится вкладка `Welcome!`, показанная на рис. 1.1.



Рис. 1.1. Страница Welcome! в Eclipse

- Запуск Nexus 4 AVD.** В нашем тестировании будет использоваться виртуальное устройство для смартфона Nexus 4, настроенное для Android 4.4 (KitKat) в разделе «Подготовка»; в разделе 1.9.2 будет показано приложение, работающее на AVD для планшета. Загрузка AVD может занять несколько минут, поэтому

рекомендуется запустить его заранее и оставить в фоновом режиме на время построения и тестирования приложений. Чтобы запустить Nexus 4 AVD, выполните команду Window ▶ Android Virtual Device Manager; на экране появится диалоговое окно Android Virtual Device Manager (рис. 1.2). Выберите вариант Nexus 4 AVD для Android KitKat и нажмите Start..., затем нажмите Launch в открывшемся диалоговом окне Launch Options. Не пытайтесь запускать приложение до завершения загрузки AVD. Когда на экране появится окно AVD (рис. 1.3), разблокируйте AVD, протащив указатель мыши от значка блокировки к краю экрана.

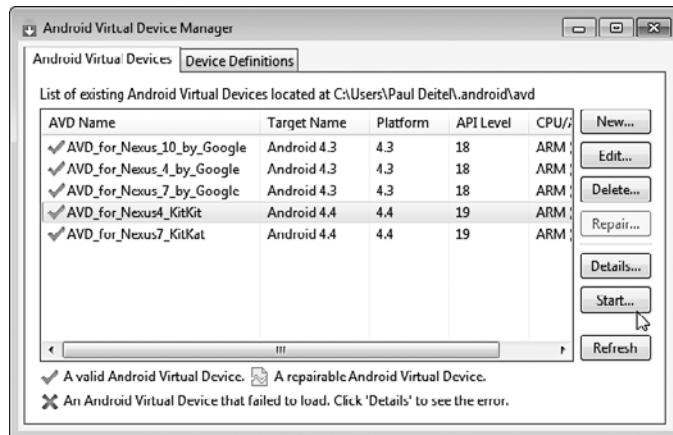


Рис. 1.2. Диалоговое окно Android Virtual Device Manager

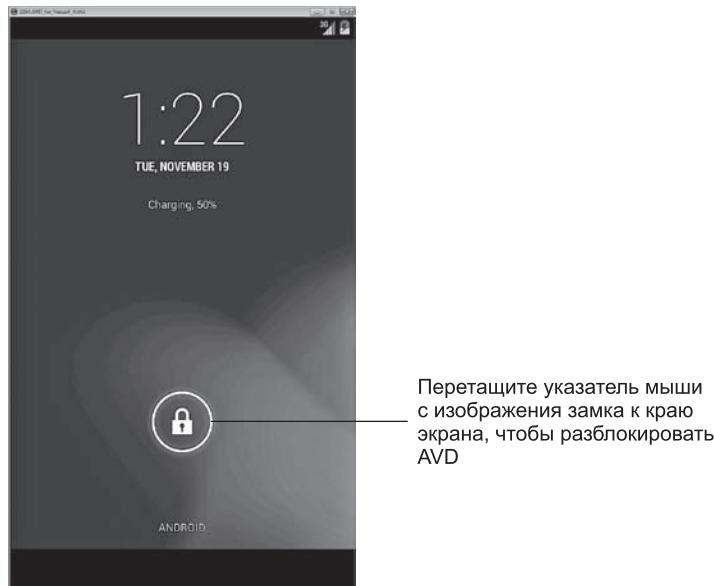


Рис. 1.3. Главный экран Nexus 4 AVD (для Android 4.4) после завершения загрузки AVD

5. Импортирование проекта приложения Doodlz. Выполните команду File > Import..., чтобы открыть диалоговое окно Import (рис. 1.4, а). Раскройте узел General, выберите вариант Existing Projects into Workspace и нажмите Next>, чтобы перейти к окну Import Projects (рис. 1.4, б). Нажмите Browse... справа от текстового поля Select root

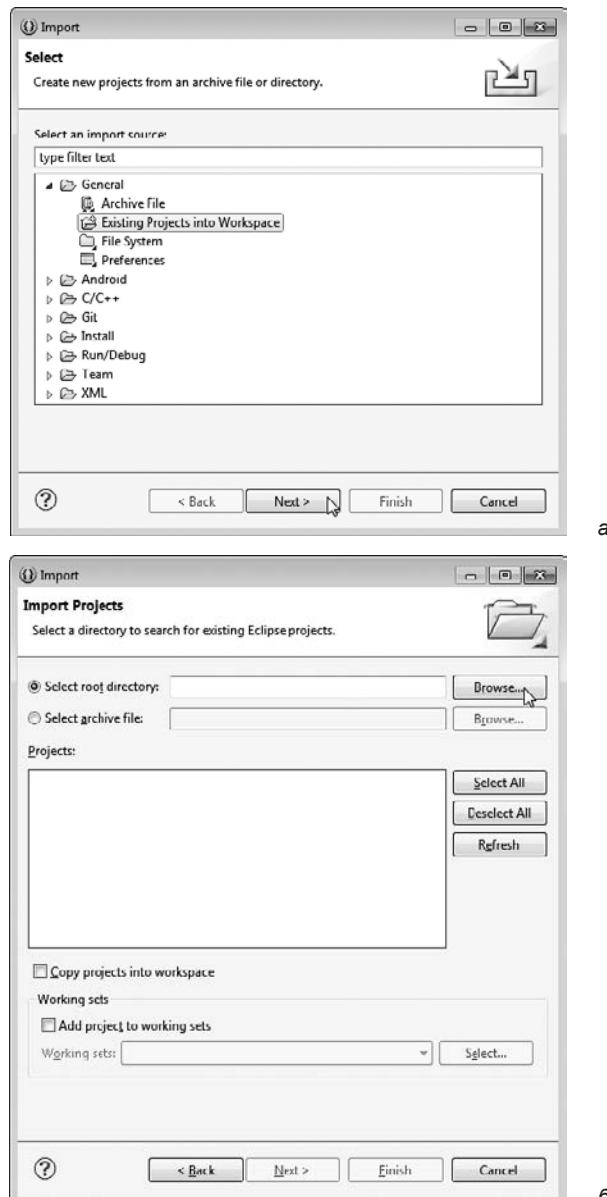


Рис. 1.4. Импортирование существующего проекта:
а — диалоговое окно Import; б — окно Import Projects

directory. В диалоговом окне Browse For Folder найдите папку Doodlz, которая находится в папке примеров книги, выберите ее и нажмите Open. Нажмите кнопку Finish, чтобы импортировать проект в Eclipse. Проект появится в окне Package Explorer в левой части окна Eclipse (рис. 1.5). Если окно Package Explorer не отображается, откройте его командой Window ▶ Show View ▶ Package Explorer.

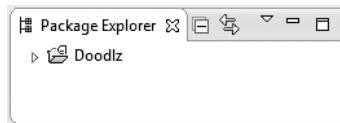


Рис. 1.5. Окно Package Explorer

6. **Запуск приложения Doodlz.** В среде Eclipse щелкните правой кнопкой мыши на проекте Doodlz в окне Package Explorer и выберите команду Run As ▶ Android Application (рис. 1.6). В результате приложение Doodlz запускается на виртуальном устройстве, загруженном на шаге 4 (рис. 1.7).
7. **Знакомство с AVD и режимом погружения.** В нижней части экрана AVD находятся различные программные кнопки, отображаемые на сенсорном экране устройства. Нажатие этих кнопок (имитируемое мышью в AVD) осуществляется

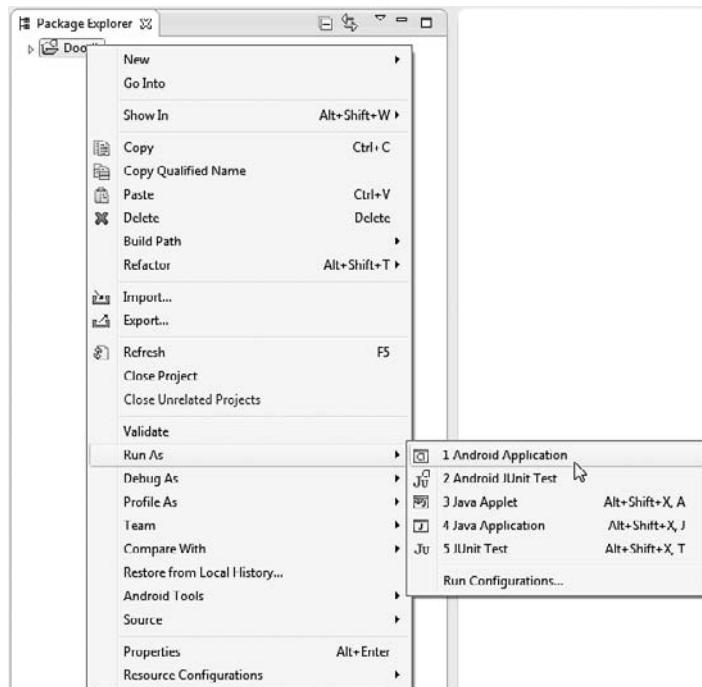


Рис. 1.6. Запуск приложения Doodlz

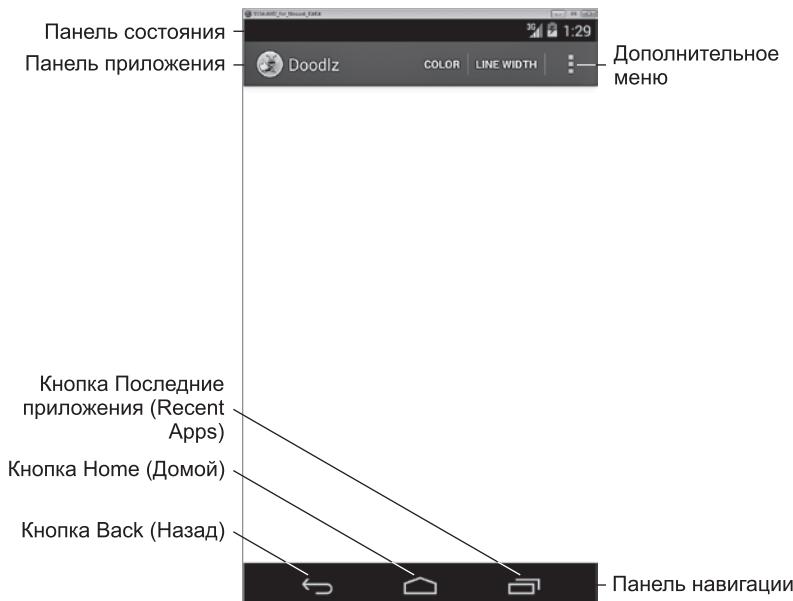


Рис. 1.7. Приложение Doodlz на виртуальном устройстве Android (AVD)

различные взаимодействия с приложениями и ОС Android. Кнопка Back (Назад) выполняет возврат к предыдущему экрану приложения — или предыдущему приложению, если текущим является начальный экран приложения. Кнопка Home (Домой) возвращает пользователя к главному экрану устройства. Кнопка Recent Apps (Последние приложения) позволяет просмотреть список недавно использовавшихся приложений для быстрого возврата к ним. У верхнего края экрана располагается панель приложения (app bar), на которой выводится значок и название приложения, а также могут отображаться другие программные кнопки конкретного приложения — некоторые выводятся на панели (COLOR и LINE WIDTH на рис. 1.7), другие открываются в дополнительном меню приложения (⋮). Количество кнопок на панели приложения зависит от размера устройства — эта тема рассматривается в главе 7. Android 4.4 поддерживает новый режим погружения (immersive mode), в котором приложение может занимать весь экран. В нашем приложении вы можете коснуться белой области рисования, чтобы скрыть панели состояния и навигации, а также панель действий приложения. Чтобы снова вызвать их на экран, прикоснитесь к области рисования или смахните от верхнего края экрана.

8. **Знакомство с меню приложения.** Чтобы вывести список команд, которые не отображаются на панели приложения, коснитесь значка меню (⋮) (то есть щелкните на нем в эмуляторе). На рис. 1.8, *a* изображена панель действий и дополнительное меню для Nexus 4 AVD, а на рис. 1.8, *б* показаны они же на устройстве Nexus 7 AVD — команды на панели действий выводятся малыми

прописными буквами (капителью). Если вы коснетесь команды COLOR, на экране появляется интерфейс изменения цвета линии. Команда LINE WIDTH открывает интерфейс для изменения толщины рисуемой линии. Команда Eraser выбирает белый цвет, чтобы при рисовании на цветных областях изображение стиралось. Команда Clear сначала требует подтвердить, что вы действительно хотите стереть все изображение, а потом очищает область рисования (если действие не было отменено). Команда Save Image сохраняет изображение в галерее устройства. В Android 4.4 команда Print открывает интерфейс для выбора принтера, чтобы вы могли распечатать рисунок или сохранить его в виде документа PDF (используется по умолчанию). Вскоре все эти команды будут описаны более подробно.

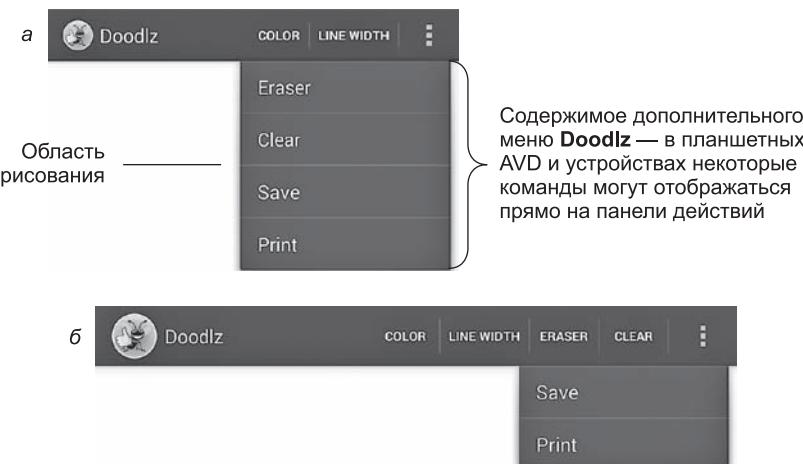


Рис. 1.8. Раскрытое меню приложения Doodlz: а — панель действий и раскрытое меню в Nexus 4 AVD; б — панель действий и раскрытое меню в Nexus 7 AVD

- Изменение цвета кисти на красный.** Чтобы изменить цвет кисти, сначала коснитесь команды Color для вызова диалогового окна выбора цвета (рис. 1.9). Цвета определяются с помощью цветовой схемы ARGB, в которой с помощью целочисленных цветовых значений (от 0 до 255) определены следующие цветовые компоненты: *альфа-канал* (прозрачность), красный, зеленый и синий. Для альфа-канала значение 0 соответствует полной прозрачности, а 255 — полной непрозрачности. Для красной, зеленой и синей составляющей 0 означает нулевую интенсивность, а 255 — максимальную интенсивность этого цвета. Интерфейс настройки цвета состоит из четырех полос SeekBar, с помощью которых можно выбирать степень прозрачности и интенсивности красного, зеленого и синего цветов. Цвета изменяются перетаскиванием ползунка SeekBar, при этом образец нового цвета отображается под полосами. Выберите красный цвет, перетащив ползунок Red в крайнее правое положение, как на рис. 1.9. Прикоснитесь к кнопке Set Color, чтобы вернуться в область рисования.

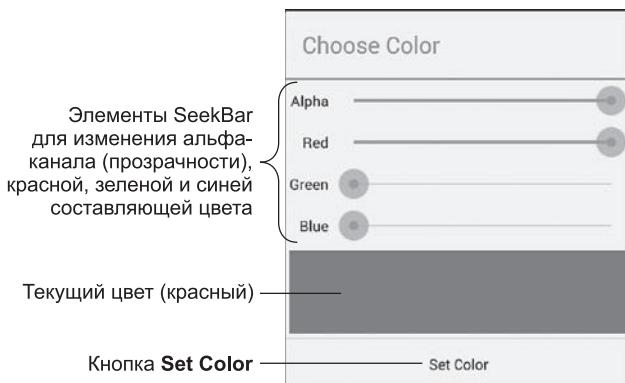


Рис. 1.9. Выбор красного цвета для рисования

10. **Изменение толщины линии.** Чтобы изменить толщину линии, коснитесь команды LINE WIDTH на панели действий; на экране появляется диалоговое окно Choose Line Width. Перетащите полосуSeekBar, предназначенную для настройки толщины линии, вправо (рис. 1.10). Прикоснитесь к кнопке Set Line Width, чтобы вернуться в область рисования.



Рис. 1.10. Изменение толщины линии

11. **Рисование лепестков.** Коснитесь экрана, чтобы войти в режим погружения, и перетащите «палец» (мышь при использовании эмулятора) по области рисования, чтобы нарисовать лепестки цветка (рис. 1.11).
12. **Изменение цвета кисти на темно-зеленый.** Коснитесь экрана, чтобы выйти из режима погружения, затем коснитесь команды COLOR для вызова диалогового окна Choose Color. Выберите темно-зеленый цвет, перетащив ползунок полосы GreenSeekBar в крайнее правое положение, а ползунки полос Red и Blue — в крайнее левое положение (рис. 1.12, а).
13. **Изменение толщины линии для рисования стебля и листьев.** Чтобы изменить толщину линии, коснитесь команды LINE WIDTH. На экране появляется диалоговое окно Choose Line Width. Перетащите полосуSeekBar, предназначенную для настройки толщины линии, вправо (рис. 1.12, б). Коснитесь экрана, чтобы войти в режим погружения, и нарисуйте стебель цветка и листья. Повторите шаги 12 и 13 для выбора более светлого оттенка зеленого и меньшей толщины линии, и нарисуйте траву (рис. 1.13).

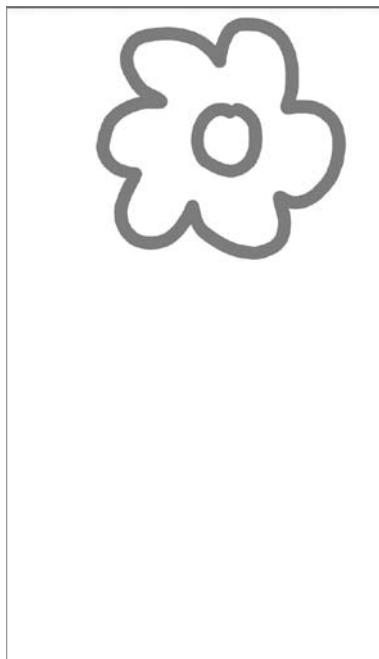


Рис. 1.11. Рисование лепестков



Рис. 1.12. Переключение на темно-зеленый цвет и увеличение толщины линии:
а — выбор темно-зеленого цвета линии; б — увеличение толщины линии

14. **Завершение рисунка.** Коснитесь экрана, чтобы выйти из режима погружения. Выберите синий цвет линии (рис. 1.14, а) и меньшую толщину (рис. 1.14, б).

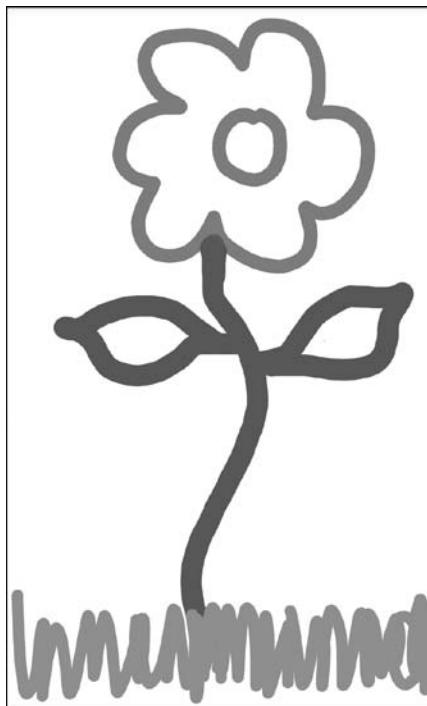


Рис. 1.13. Рисование стебля и травы

Коснитесь экрана, чтобы войти в режим погружения, и нарисуйте капли дождя (рис. 1.15).

15. **Сохранение изображения.** При желании можно сохранить изображение в приложении *Gallery*. Для этого выберите команду *Save* из дополнительного меню . Чтобы просмотреть это и другие изображения, хранящиеся на устройстве, откройте приложение *Gallery*.
16. **Печать изображения.** Чтобы распечатать изображение, выберите команду *Print* в дополнительном меню. На экране появляется диалоговое окно печати, которое по умолчанию предлагает сохранить изображение в виде документа PDF. Чтобы выбрать принтер, выберите режим *Save as PDF* и выберите один из доступных принтеров. Если в списке нет ни одного принтера, возможно, вам придется настроить сервис Google Cloud Print для вашего принтера. Дополнительная информация доступна на странице
<http://www.google.com/cloudprint/learn/>
17. **Возвращение на главный экран.** Вернитесь на главный экран AVD, коснувшись кнопки *Home* () в AVD. Чтобы просмотреть рисунок в приложении *Gallery*, коснитесь кнопки () для открытия списка приложений, установленных в AVD. Откройте приложение *Gallery* и просмотрите рисунок.



Рис. 1.14. Выбор синего цвета и меньшей толщины линии: а — выбор синего цвета линии; б — уменьшение толщины линии



Рис. 1.15. Рисование дождевых капель с выбором другого цвета и толщины линии

1.9.2. Запуск Doodlz App на планшетном AVD

Чтобы протестировать приложение в планшетном AVD, сначала запустите AVD, как описано на шаге 4 в разделе 1.9.1, но вместо Nexus 4 AVD выберите Nexus 7 AVD. Затем щелкните правой кнопкой мыши на проекте Doodlz в окне Eclipse Package Explorer и выберите команду Run As ▶ Android Application. Если при запуске приложения запущено сразу несколько AVD, появляется диалоговое окно Android Device Chooser (рис. 1.16), в котором выбирается AVD для установки и выполнения приложения. В нашей системе работали сразу два AVD для Nexus 4 и Nexus 7, поэтому были доступны два виртуальных устройства Android для запуска приложения. Выберите Nexus 7 AVD и нажмите OK. Это приложение запускается в портретной ориентации (ширина меньше высоты) на телефонах и малых планшетах. Если запустить приложение на AVD большого планшета (или большом планшетном устройстве), то оно запустится в альбомной ориентации (ширина больше высоты). На рис. 1.17 показано приложение в Nexus 7 AVD. Если высота AVD слишком велика для вида экрана, вы можете изменить ориентацию AVD нажатием Ctrl+F12 (на Mac используется комбинация fn+control+F12). На некоторых клавиатурах клавиша Ctrl имеет пометку Control.

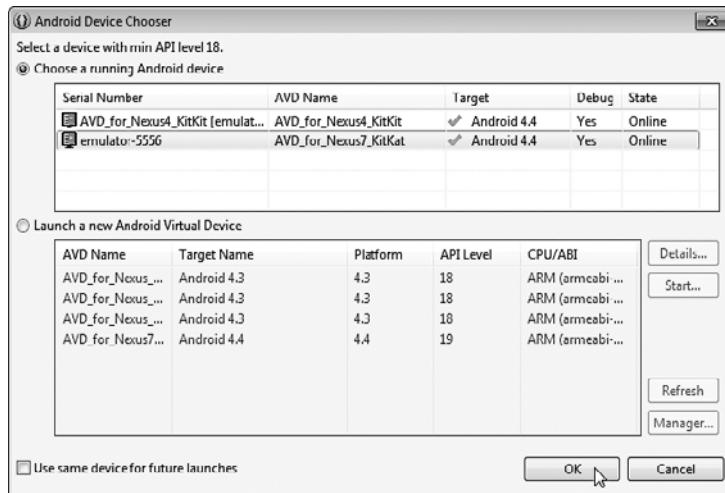


Рис. 1.16. Диалоговое окно Android Device Chooser

1.9.3. Выполнение приложения Doodlz на устройстве Android

Если у вас имеется устройство Android, вы можете легко протестировать приложение на этом устройстве.

- Активизируйте средства разработчика на устройстве.** Прежде всего нужно включить для устройства режим отладки. Запустите приложение *Settings*,



Рис. 1.17. Рисование в Nexus 7 AVD

установленное на устройстве, затем выберите раздел About Phone (или About Tablet), найдите пункт Build Number (в конце списка) и последовательно нажимайте на нем, пока на экране не появится сообщение You are now a developer. При этом в приложении Settings появляется раздел Developer Options.

2. **Включите режим отладки на устройстве.** Вернитесь в приложение Settings, выберите раздел Developer Options и убедитесь в том, что флагок USB debugging установлен — это происходит по умолчанию при первом включении средств разработчика на устройстве.
3. **Подключите устройство.** Подключите устройство к компьютеру с помощью кабеля USB, входящего в комплект поставки вашего устройства. Если вы пользователь Windows, вспомните, о чем говорилось в разделе «Подготовка» — возможно, вам потребуется установить драйвер USB для вашего устройства. Подробную информацию можно найти на следующих двух страницах:
developer.android.com/tools/device.html
developer.android.com/tools/extras/oem-usb.html
4. **Запустите Doodlz на устройстве Android.** В среде Eclipse перейдите в окно Package Explorer, щелкните правой кнопкой мыши на проекте Doodlz и выберите команду Run As > Android Application. Если в системе нет открытых AVD, но имеется подключенное устройство Android, среда автоматически установит приложение на вашем устройстве и выполнит его. Если в системе открыто одно или несколько виртуальных устройств AVD и/или имеются подключенные физические устройства, на экране появляется диалоговое окно Android Device Chooser (рис. 1.16). В нем следует выбрать устройство или AVD для установки и запуска приложения.

Подготовка к распространению приложений

Если вы создаете приложения, предназначенные для распространения через Google Play или другие магазины, протестируйте их на возможно большем числе реальных устройств. Не забывайте о том, что некоторые функции могут быть протестированы только на реальных устройствах. Если вы не можете найти реальные устройства в достаточном количестве, создайте устройства AVD, имитирующие различные устройства, на которых должно выполняться созданное вами приложение. Просмотрите в Интернете спецификации реальных устройств, а потом настройте соответствующим образом каждое виртуальное устройство AVD. Также можно изменить файл *config.ini* виртуального устройства AVD, как описано в разделе «Setting hardware emulation options» на сайте

developer.android.com/guide/developing/tools/avd.html

Файл *config.ini* включает параметры, которые не настраиваются в Android Virtual Device Manager. Настройка этих параметров позволяет добиться более точного соответствия выбранного виртуального устройства Android с реальным.

1.10. Создание успешных Android-приложений

Сейчас в Google Play доступно свыше 800 000 приложений.¹ Как создать приложение Android, которое пользователи найдут, загрузят, будут использовать и рекомендовать другим? Подумайте, что сделает ваше приложение интересным, полезным, привлекательным и долгоживущим. Эффектное название, привлекательный значок и заманчивое описание могут привлечь людей к вашему приложению в Google Play или одном из многих других магазинов приложений Android. Но когда пользователь загрузит ваше приложение, что заставит его регулярно работать с приложением и порекомендовать его другим? В табл. 1.14 перечислены некоторые характеристики успешных приложений.

Таблица 1.14. Характеристики успешных приложений

Характеристики успешных приложений
Успешные игры
Интересны и увлекательны
Требуют усилий со стороны игрока
Предоставляют разные уровни сложности
Выводят счет и ведут таблицы рекордов
Предоставляют звуковую и визуальную обратную связь
Реализуют качественную анимацию

¹ <http://www.pureoxygenmobile.com/how-many-apps-in-each-app-store/>

Характеристики успешных приложений

Выделяют операции ввода/вывода и интенсивные вычисления в отдельные программные потоки для повышения скорости отклика интерфейса и быстродействия приложения

Используют инновационную технологию дополненной реальности, насыщая реальное окружение виртуальными компонентами; данная возможность особенно популярна в приложениях на базе видео

Утилиты

Предоставляют полезную функциональность и точную информацию

Повышают персональную производительность и эффективность бизнеса

Упрощают выполнение операций (ведение списков задач, управление расходами и т. д.)

Повышают информированность пользователя

Предоставляют тематическую информацию (последние биржевые котировки, новости, предупреждения о штормах, оперативная информация о потоке транспорта и т. д.)

Используют географическую привязку для предоставления локального сервиса (купоны для местных магазинов, лучшие цены на бензин, доставка еды и т. д.)

Общие характеристики

Используют новейшие возможности Android, но совместимы с разными версиями Android для поддержки максимально широкой аудитории

Работают корректно

Ошибки быстро исправляются

Следуют стандартным правилам графического интерфейса приложений Android

Быстро запускаются

Быстро реагируют на действия пользователя

Не требуют слишком большого объема памяти, передачи больших объемов данных или заряда батареи

Оригинальны и необычны

Выдерживают испытание временем (то есть пользователи периодически возвращаются к ним)

Используют значки профессионального уровня, которые отображаются в Google Play и на устройстве пользователя

Используют качественную графику, анимации, аудио и видео

Интуитивно понятны и просты в использовании (не требуют обширной справочной документации)

Доступны для людей с ограниченными возможностями (<http://developer.android.com/guide/topics/ui/accessibility/index.html>)

Представляют причины и средства для передачи другим пользователям информации о приложении (например, дают возможность опубликовать игровой результат в Facebook или Twitter)

Таблица 1.14 (окончание)

Характеристики успешных приложений
Предоставляют дополнительный контент там, где это уместно (игровые уровни, статьи, головоломки и т. д.)
Локализуются (см. главу 2) для каждой страны, в которой распространяются (например, перевод текстовых и звуковых файлов, использование разной графики в зависимости от местонахождения пользователя и т. д.)
Превосходят приложения конкурентов по быстродействию, функциональности или простоте использования
Эффективно используют встроенные возможности устройства
Не требуют лишних привилегий
Проектируются для оптимального выполнения в широком спектре устройств Android
Не теряют актуальности с выходом новых устройств — разработчик точно указывает набор аппаратных возможностей, используемых приложением, чтобы магазин Google Play мог фильтровать и отображать приложение только для совместимых устройств (http://android-developers.blogspot.com/2010/06/future-proofing-your-app.html)

1.11. Ресурсы для разработчиков

В табл. 1.15 перечислены некоторые важнейшие документы с сайта Android Developer. В ходе углубленного изучения разработки Android-приложений у вас могут возникнуть вопросы по инструментам, особенностям проектирования, безопасности и т. д. Существует целый ряд новостных групп и форумов для разработчиков Android, на которых можно найти новейшие объявления или задать вопросы (табл. 1.16). В табл. 1.17 приведены некоторые сайты, на которых можно найти рекомендации, видеоролики и ресурсы по программированию для Android.

Таблица 1.15. Важная электронная документация для Android-разработчиков

Название	URL
Компоненты приложений	http://developer.android.com/guide/components/index.html
Использование эмулятора Android	http://developer.android.com/tools/devices/emulator.html
Справочник пакетов	http://developer.android.com/reference/packages.html
Справочник классов	http://developer.android.com/reference/classes.html
Дизайн приложений	http://developer.android.com/design/index.html
Резервное копирование	http://developer.android.com/guide/topics/data/backup.html
Советы по безопасности	http://developer.android.com/training/articles/security-tips.html
Управление проектами из Eclipse с ADT	http://developer.android.com/guide/developing/projects/projects-eclipse.html

Название	URL
Основы Android Studio	http://developer.android.com/sdk/installing/studio.html
Отладка	http://developer.android.com/tools/debugging/index.html
Справочная информация по инструментам	http://developer.android.com/tools/help/index.html
Рекомендации по производительности	http://developer.android.com/training/articles/perf-tips.html
Обеспечение быстрой реакции интерфейса	http://developer.android.com/training/articles/perf-anr.html
Контрольный список публикации (для Google Play)	http://developer.android.com/distribute/googleplay/publish/preparing.html
Основы публикации приложений	http://developer.android.com/distribute/googleplay/publish/register.html
Управление памятью приложения	http://developer.android.com/training/articles/memory.html
Соглашение о распространении приложений через Google Play для разработчиков	http://play.google.com/about/developer-distribution-agreement.html

Таблица 1.16. Новостные группы и форумы Android

Название	Подписка	Описание
Android Discuss	Подписка через Google Groups: android-discuss Подписка по электронной почте: android-discuss-subscribe@googlegroups.com	Общая конференция по программированию для Android, в которой можно получить ответы на вопросы по разработке приложений
Stack Overflow	http://stackoverflow.com/questions/tagged/android	Список вопросов начального уровня по разработке для Android, включающий азы работы с Java и Eclipse, и основные приемы программирования
Android Developers	http://groups.google.com/forum/?fromgroups#!forum/android-developers	Вопросы диагностики, проектирования графического интерфейса, оптимизации и т. д. для опытных разработчиков
Android Forums	http://www.androidforums.com	Здесь можно задать вопросы, обменяться информацией с другими разработчиками и найти форумы конкретных устройств на базе Android

Таблица 1.17. Новостные группы и форумы Android

Информация, видео, ресурсы	URL
Примеры Android-приложений от Google	http://code.google.com/p/apps-for-android/
Статья «Десять советов по разработке Android-приложений» (O'Reilly)	http://answers.oreilly.com/topic/862-ten-tips-for-android-application-development/
Сайт Bright Hub™ с рекомендациями для Android-программистов и практическими руководствами	http://www.brighthub.com/mobile/google-android.aspx
The Android Developers Blog	http://android-developers.blogspot.com/
The Sprint® Application Developers Program	http://developer.sprint.com/site/global/develop/mobile_platforms/android/android.jsp
HTC's Developer Center for Android	http://www.htcdev.com/
Сайт Android-разработки компании Motorola	http://developer.motorola.com/
Top Android Users on Stack Overflow	http://stackoverflow.com/tags/android/topusers
AndroidDev Weekly Newsletter	http://androiddevweekly.com/
Codependent (блог Чета Хаазе)	http://graphics-geek.blogspot.com/
Блог Сирила Мотье	http://cyrilmottier.com/
Блог Ромена Гая	http://www.curious-creature.org/category/android/
Канал для Android-разработчиков на YouTube®	http://www.youtube.com/user/androiddevelopers
Видеоролики по программированию для Android	http://developer.android.com/develop/index.html
Новости инструментария для Android-разработчиков	http://www.youtube.com/watch?v=Imv1dTnhLH4
Видео с конференции Google I/O 2013 Developer Conference	http://developers.google.com/events/io/sessions

1.12. Резюме

В этой главе вашему вниманию была представлена краткая история Android и функциональные свойства этой платформы. Были представлены ссылки на некоторую ключевую документацию в Интернете, группы новостей и форумы, которые

позволят вам связаться с сообществом разработчиков. Мы рассмотрели возможности операционной системы Android Market и привели ссылки на некоторые приложения в Google Play — как бесплатные, так и распространяемые на коммерческой основе. Были изложены начальные сведения о пакетах Java, Android и Google, которые позволяют использовать аппаратные и программные свойства для создания Android-приложений; многие из этих пакетов будут использованы в книге. Также были рассмотрены язык программирования Java и Android SDK, жесты Android, а также способы их выполнения на устройствах Android и эмуляторах. Вашему вниманию был предложен краткий обзор концепций объектной технологии, в том числе классы, объекты, атрибуты и аспекты поведения. Мы протестировали приложение Doodlz на эмуляторах смартфона и планшета Android. В главе 2 мы построим свое первое Android-приложение, используя только средства визуального программирования. Приложение выводит текст и два изображения. Также будут рассмотрены средства обеспечения доступности и интернационализации Android-приложений.

2 Приложение Welcome

Знакомство со средствами разработчика

В этой главе...

- Изучение основных средств Android-разработчика (интегрированной среды разработки Eclipse и плагина ADT), применяемых для написания, выполнения и отладки Android-приложений
- Создание нового проекта приложения в среде разработки
- Построение графического интерфейса пользователя (без программирования) в макетном редакторе
- Изменение свойств компонентов графического интерфейса
- Создание простого приложения Android и его выполнение в эмуляторе
- Повышение доступности приложения для пользователей с ослабленным зрением посредством назначения строк, используемых функциями Android TalkBack и Explore-by-Touch
- Поддержка интернационализации, чтобы приложение могло выводить локализованные строки на разных языках

2.1. Введение

В этой главе мы создадим приложение Welcome. Это простое приложение выводит приветственное сообщение и два изображения, и для его создания мы не напишем *ни одной строчки кода*. Мы воспользуемся средствами *ADT* для создания приложения, работающего на телефонах Android. В следующих главах будет показано, что вы также можете создавать приложения, работающие на планшетах или как на телефонах, так и на планшетах. Мы создадим простое Android-приложение (рис. 2.1) в *макетном редакторе*, который позволяет строить графический интерфейс пользователя простым перетаскиванием. И наконец, мы запустим приложение на *эмulateоре Android* (и на телефоне Android, если он у вас есть). Наконец, вы узнаете, как сделать приложение доступным для людей с ограниченными возможностями и как организовать вывод строк, локализованных для разных языков. На сайте книги – <http://www.deitel.com/books/AndroidFP2> – приведена версия приложения этой главы для *интегрированной среды разработки Android Studio*. Предполагается, что вы ознакомились с предисловием, разделом «Подготовка» и разделом 1.9.

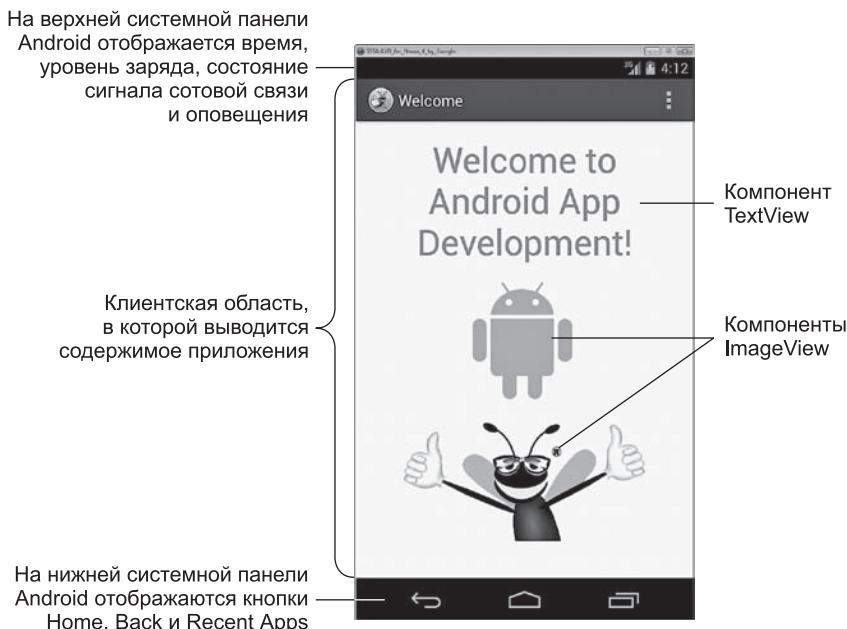


Рис. 2.1. Приложение Welcome в эмуляторе Android

2.2. Обзор применяемых технологий

В этом разделе представлены технологии, о которых вы узнаете в этой главе.

2.2.1. Android Developer Tools IDE

В этой главе будет рассмотрена интегрированная среда разработки Android Developer Tools (ADT). Мы воспользуемся ею для создания нового проекта (см. раздел 2.3). Как вы увидите, среда создает простейший графический интерфейс с текстом “Hello world!”. Далее мы воспользуемся макетным редактором среды и окном свойств для построения простого графического интерфейса с текстом и двумя изображениями (см. раздел 2.5).

2.2.2. TextView и ImageView

Для вывода текста в приложении будет использоваться компонент `TextView`, а графика будет отображаться в компонентах `ImageView`. Графический интерфейс, созданный для приложения по умолчанию, содержит компонент `TextView`, различные параметры которого — текст, размер шрифта, цвет текста и т. д. — настраиваются в окне свойств среды разработки (см. раздел 2.5.3). Затем мы перетащим компоненты `ImageView` с палитры в макет графического интерфейса (см. раздел 2.5.4).

2.2.3. Ресурсы приложения

Все строки и числовые значения рекомендуется определять в файлах ресурсов, которые размещаются во вложенных папках папки `res` проекта. В разделе 2.5.3 вы узнаете, как создавать ресурсы для строк (например, для текста в компоненте `TextView`) и метрик (например, размера шрифта). Также будет продемонстрировано использование встроенных цветовых ресурсов Android для определения цвета шрифта `TextView`.

2.2.4. Доступность

Android содержит разнообразные специальные средства, упрощающие использование устройств людьми с ограниченными возможностями. Например, слабовидящие пользователи могут воспользоваться функцией TalkBack — экранным диктором, читающим текст на экране (или текст, предоставленный разработчиком) для лучшего понимания назначения и содержимого компонента. Функция Android Explore by Touch позволяет прикоснуться к экрану, чтобы диктор TalkBack зачитал информацию о содержимом экрана рядом с точкой касания. В разделе 2.7 показано, как включить эти функции и как настроить компоненты графического интерфейса приложения для улучшения доступности.

2.2.5. Интернационализация

Устройства на базе Android используются во всем мире. Чтобы ваши приложения охватывали как можно большую аудиторию, подумайте о том, чтобы адаптировать их для разных культур и языков — этот процесс называется *интернационализацией*. В разделе 2.8 показано, как в приложении Welcome предоставить текст на испанском языке для компонента `TextView` и строки доступности для компонентов `ImageView` и как затем протестировать приложение на виртуальном устройстве AVD, настроенном для испанского языка.

2.3. Создание приложения

Примеры книги разрабатывались с использованием версий Android Developer Tools (22.x) и Android SDK (4.3 и 4.4), актуальных на момент написания книги. Предполагается, что вы прочитали раздел «Подготовка» и установили Java SE Development Kit (JDK) и интегрированную среду Android Developer Tools, которая использовалась для тестового запуска приложения в разделе 1.9. В этом разделе будет показано, как создать новый проект в IDE, а другие функции IDE будут представлены в книге.

2.3.1. Запуск интегрированной среды разработки Android Developer Tools

Чтобы запустить IDE, откройте папку `eclipse` в папке установки пакета Android SDK/ADT и сделайте двойной щелчок на значке Eclipse icon ({} или ●, в зависимости от платформы).

При первом запуске среды разработки открывается страница `Welcome` (см. рис. 2.1). Если она не появилась, откройте ее командой `Help > Android IDE`.

2.3.2. Создание нового проекта

Проект — это группа связанных файлов (например, файлы кода и графические файлы), образующих приложение. Работа над приложением начинается с создания проекта. Нажмите `New Android Application...` на странице `Welcome`, чтобы открыть диалоговое окно `New Android Application` (рис. 2.2). То же самое можно сделать другими способами: выберите команду меню `File > New > Android Application Project` или откройте список рядом с кнопкой `New` на панели инструментов (⋮) и выберите команду `Android Application Project`.

2.3.3. Диалоговое окно New Android Application

Введите следующую информацию на первом шаге мастера `New Android Application` (рис. 2.2), после чего нажмите `Next>`:

- Application Name: — имя приложения. Введите в этом поле строку `Welcome`.

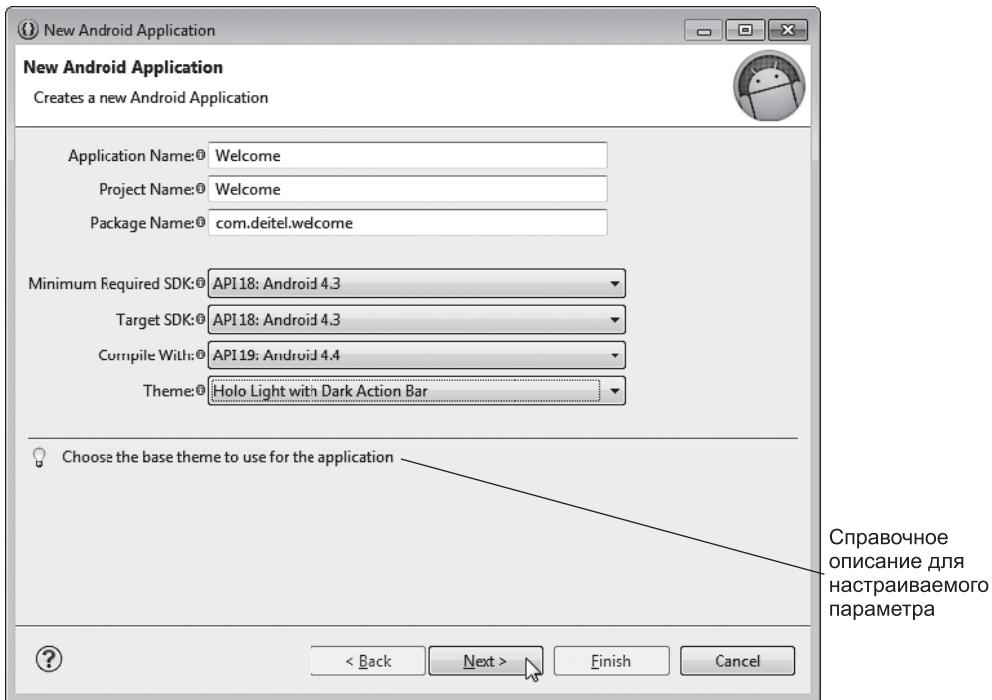


Рис. 2.2. Диалоговое окно New Android Application

- Project Name: — имя проекта, отображаемое в корневом узле проекта на панели Package Explorer в IDE. По умолчанию IDE выбирает в качестве имени проекта имя проекта без пробелов, в котором каждое слово записывается с прописной буквы, — так, для приложения с именем `Address Book` будет выбрано имя проекта `AddressBook`. Если вы предпочитаете использовать другое имя, введите его в поле Project Name:.
- Package Name: — имя пакета Java для исходного кода приложения. Android и магазин Google Play используют это имя в качестве уникального идентификатора приложения, которое должно оставаться постоянным во всех версиях приложения. Имя пакета обычно начинается с доменного имени вашей компании или учреждения, записанного в обратном порядке. Например, мы используем доменное имя `deitel.com`, поэтому имена наших пакетов начинаются с префикса `com.deitel`. Далее обычно следует имя приложения. По общепринятым соглашениям в имени пакета используются только буквы нижнего регистра. По умолчанию IDE выбирает имя пакета, начинающееся с префикса `com.example`, — этот префикс используется только в учебных целях и его необходимо изменить, если вы собираетесь распространять свое приложение.

4. **Minimum Required SDK:** — минимальный уровень Android API, необходимый для запуска приложения. Этот параметр разрешает выполнять приложение на устройствах с заданным уровнем API и выше. Мы используем уровень 18, соответствующий Android 4.3 — меньшей из двух версий, используемых в книге. В табл. 2.1 перечислены версии Android SDK и соответствующие им уровни API. Другие версии SDK объявлены устаревшими и не должны использоваться. Информацию о процентном соотношении устройств на базе Android, использующих каждую платформу, можно найти по адресу

<http://developer.android.com/about/dashboards/index.html>

Таблица 2.1. Версии Android SDK и уровни API
(<http://developer.android.com/about/dashboards/index.html>)

Версия SDK	Уровень API
4.4	19
4.3	18
4.2.x	17
4.1.x	16
4.0.3-4.0.4	15
4.0.1	14
3.2	13
2.3.3-2.3.7	10
2.2	8
2.1	7
1.6	4

5. **Target SDK:** — предпочтительный уровень API. В приложениях книги используется уровень 19 (Android 4.4). На момент написания на 26% устройств на базе Android все еще использовался уровень 10. При разработке приложений для распространения желательно ориентироваться на максимально широкий круг устройств. Например, чтобы приложение предназначалось для Android 2.3.3 и выше (98% всех устройств Android), следует выбрать в поле **Minimum Required SDK** значение 10. Если в этом поле выбран более ранний уровень API, чем в поле **Target SDK**, проследите за тем, чтобы приложение не использовало функции уровней API выше **Minimum Required SDK** или чтобы оно проверяло уровень API устройства и соответствующим образом изменяло свою функциональность. Служебная программа Android Lint, которую IDE выполняет в фоновом режиме, сообщает о попытках использования неподдерживаемых функций.
6. **Compile With:** — версия API, используемая при компиляции приложения. Обычно совпадает с версией в поле **Target SDK**, но также может быть более ранней версией с поддержкой всех API, используемых в приложении.

7. **Theme:** — тема оформления по умолчанию, соответствующая стандартам Android. Выбрать можно из трех тем — Holo Light, Holo Dark и Holo Light with Dark Action Bars (выбирается IDE по умолчанию). При построении графического интерфейса можно выбрать одну из многих разновидностей тем Holo Light и Holo Dark. В этой главе используется тема по умолчанию, а другие варианты более подробно рассматриваются в последующих главах. За дополнительной информацией о каждой теме с примерами экранов обращайтесь по адресу:

<http://developer.android.com/design/style/themes.html>

2.3.4. Шаг Configure Project

На втором шаге мастера New Android Application — Configure Project (рис. 2.3) — оставьте значения по умолчанию и нажмите **Next >**. Эти настройки позволяют выбрать значок приложения и активность (Activity) — класс, управляющий выполнением приложения.

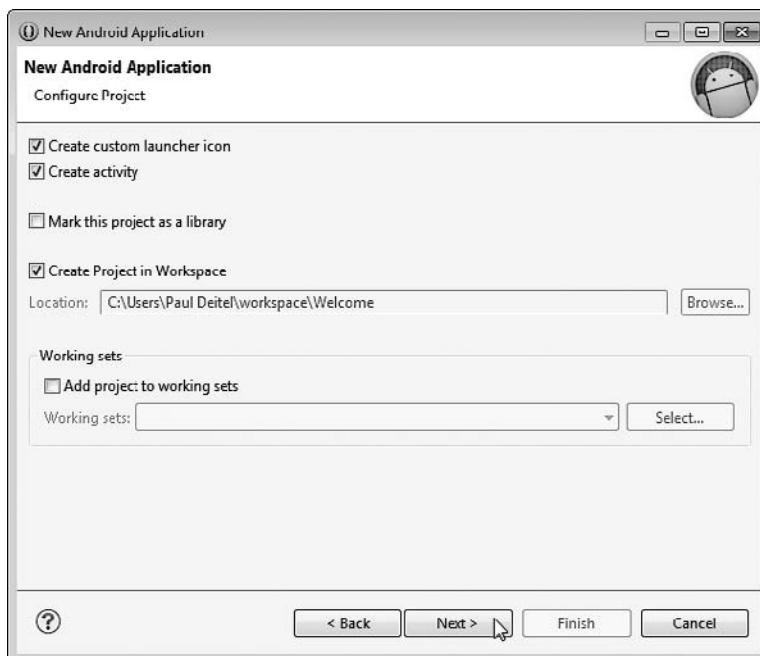


Рис. 2.3. Диалоговое окно New Android Application (шаг 2)

2.3.5. Шаг Configure Launcher Icon

Когда ваше приложение устанавливается на устройстве, его значок и имя появляются вместе с остальными установленными приложениями в лаунчере — оболочке,

вызываемой значком  на главном экране устройства. Android работает на множестве устройств с разными размерами и разрешениями экрана. Чтобы графика хорошо смотрелась на всех устройствах, следует предоставить несколько версий каждого изображения, используемого устройством. Android автоматически выбирает правильное изображение в зависимости от разных характеристик, таких как размеры экрана (ширина и высота в пикселях) или количество точек на дюйм (DPI, dots per inch). Мы рассмотрим эти механизмы начиная с главы 3. За дополнительной информацией о проектировании интерфейса для разных размеров экрана и разрешений обращайтесь по адресу

<http://developer.android.com/training/multiscreen/index.html>

а о значках вообще — по адресу

<http://developer.android.com/design/style/iconography.html>

На шаге Configure Launcher Icon (рис. 2.4) можно выбрать значок приложения из существующих изображений, графических заготовок или текста. Для выбранного значка создаются версии, масштабированные до размеров 48×48, 72×72, 96×96 и 144×144 для поддержки разных разрешений экрана. В нашем приложении используется изображение из файла *DeitelOrange.png*. Чтобы использовать его, нажмите Browse... справа от поля Image File:, перейдите в папку images в папке примеров, выберите

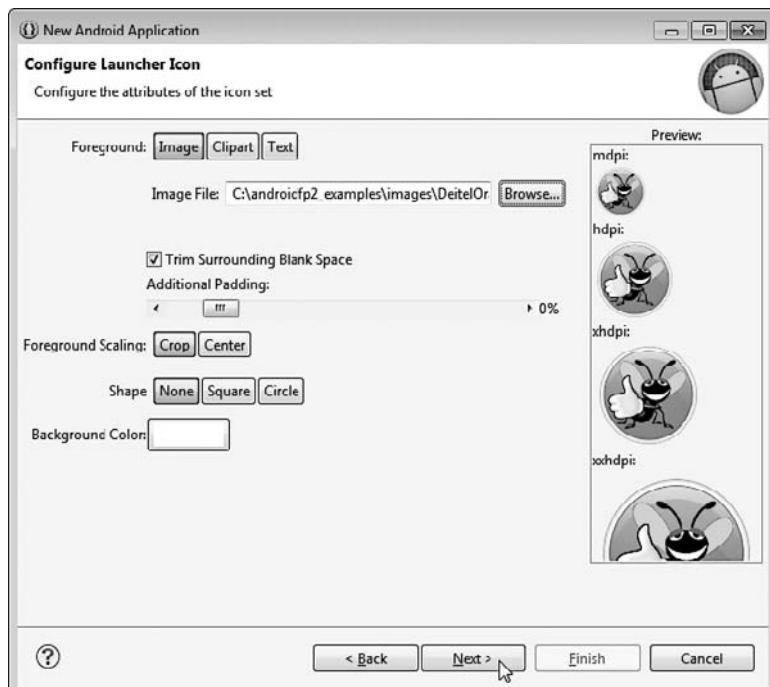


Рис. 2.4. Диалоговое окно New Android Application (шаг 3)

файл DeitelOrange.png и нажмите Open. Предварительные версии масштабированных изображений отображаются в области Preview диалогового окна. Эти изображения помещаются в соответствующие папки проекта приложения. Изображения не всегда хорошо масштабируются. Для приложений, которые вы собираетесь продавать через Google Play, стоит обратиться к профессиональному дизайнеру за версиями значков для разных разрешений. В главе 9 рассматривается процесс отправки приложений в магазин Google Play и приводится список нескольких компаний, предоставляющих услуги по дизайну значков (бесплатные или платные). Нажмите Next>, чтобы перейти к следующему шагу.

2.3.6. Шаг Create Activity

На шаге Create Activity (рис. 2.5) выбирается шаблон *активности* (*Activity*) приложения. Шаблоны экономят время разработчика, предоставляя заранее настроенные отправные точки для часто используемых вариантов дизайна приложения. В табл. 2.2 кратко описаны три шаблона, изображенных на рис. 2.5. Для нашего приложения выберите пустой шаблон *Blank Activity* и нажмите Next>. Остальные шаблоны будут использоваться в других главах.

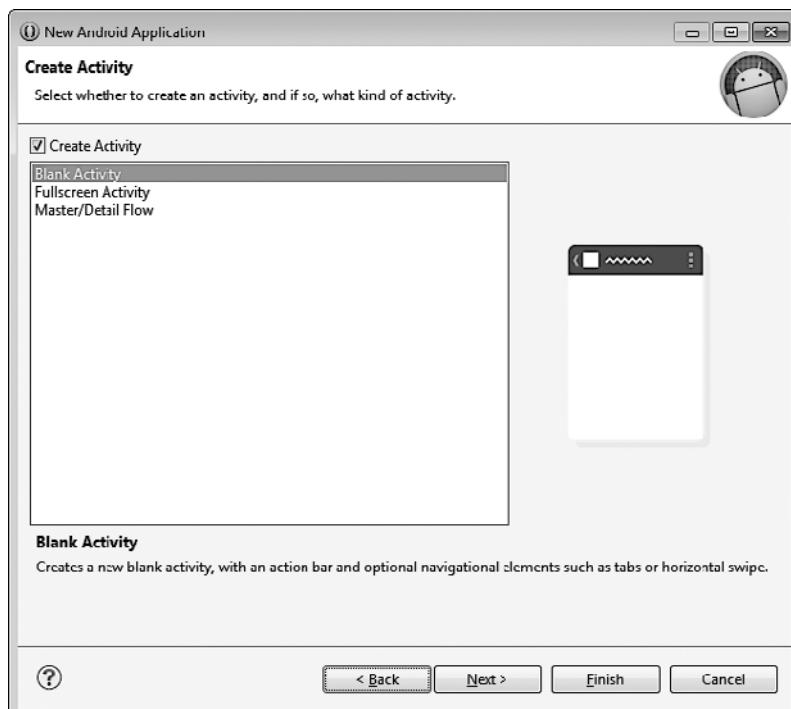


Рис. 2.5. Диалоговое окно New Android Application (шаг 4)

Таблица 2.2. Шаблоны Activity

Шаблон	Описание
Blank Activity	Используется для одноэкранных приложений, в которых большая часть графического интерфейса строится разработчиком. Включает панель действий в верхней части приложения, в которой выводится имя приложения, а также могут отображаться элементы управления для взаимодействия с приложением
Fullscreen Activity	Используется для одноэкранных приложений (по аналогии с Blank Activity), занимающих весь экран, но с возможностью переключения режима видимости панели состояния устройства и панели действий приложения
Master/Detail Flow	Используется для приложений с главным списком, из которого пользователь может выбрать один вариант для просмотра подробной информации (как во встроенных приложениях Email и People). На планшетах главный список и детализация выводятся рядом друг с другом на одном экране. На телефонах главный список выводится на одном экране, а при выборе варианта детализация отображается на отдельном экране

2.3.7. Шаг Blank Activity

Этот шаг (рис. 2.6) зависит от выбора шаблона на предыдущем шаге. Для шаблона Blank Activity он позволяет задать:

- Activity Name** — имя активности (по умолчанию IDE предлагает имя `MainActivity`). Это имя субкласса `Activity`, управляющего выполнением приложения. Начиная с главы 3 мы будем изменять этот класс для реализации функциональности приложения.
- Layout Name** — имя файла макета (по умолчанию IDE предлагает имя файла `activity_main.xml`). В файле хранится представление графического интерфейса приложения в формате XML. В этой главе мы построим интерфейс (раздел 2.5) визуальными средствами.
- Navigation Type** — тип навигации (по умолчанию IDE предлагает значение `None`). Приложение `Welcome` не предоставляет никакой функциональности. В приложении, поддерживающем взаимодействие с пользователем, можно выбрать другой режим для просмотра контента вашего приложения. Режимы навигации будут более подробно рассмотрены в следующих приложениях.

Нажмите `Finish`, чтобы завершить создание проекта.



Рис. 2.6. Диалоговое окно New Android Application (шаг 5)

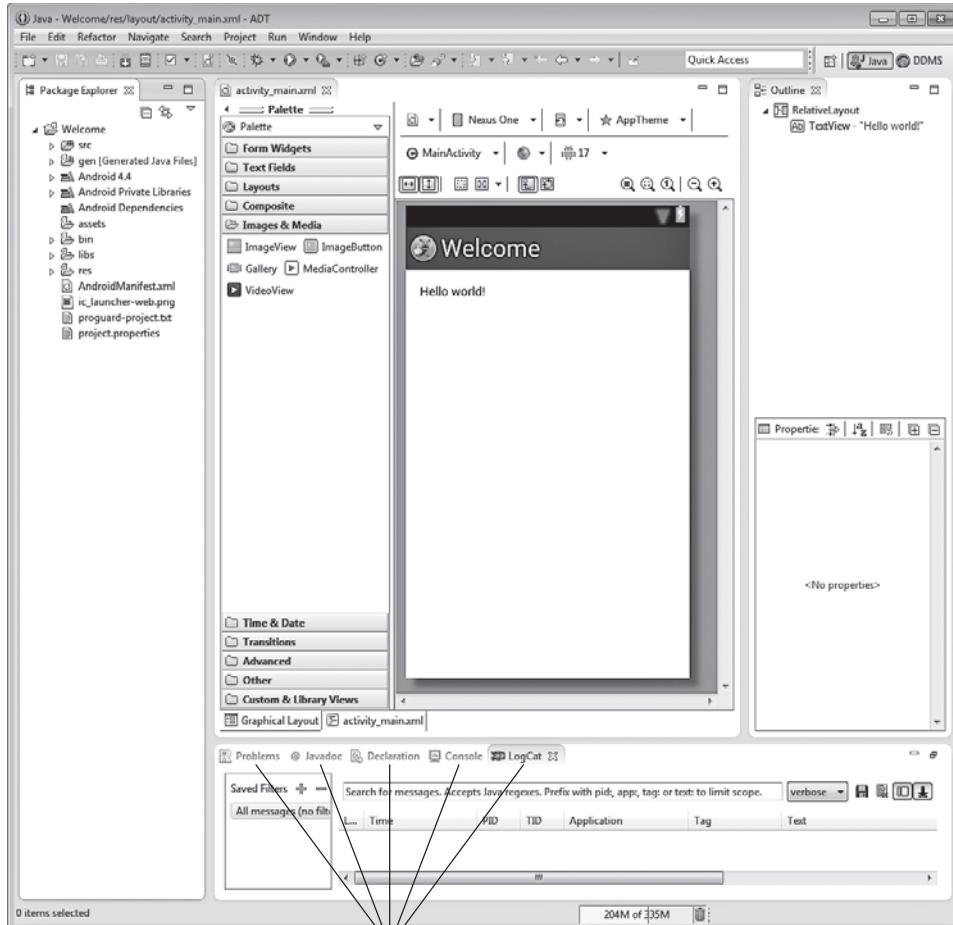
2.4. Окно Android Developer Tools

При завершении создания проекта IDE открывает файлы `MainActivity.java` и `activity_main.xml`. Закройте файл `MainActivity.java`, чтобы окно IDE выглядело так, как показано на рис. 2.7. В IDE отображается макетный редактор для построения графического интерфейса приложения. В этой главе мы ограничимся средствами, необходимыми для построения приложения `Welcome`. Другие возможности IDE будут представлены позднее.

2.4.1. Окно Package Explorer

Окно `Package Explorer` предоставляет доступ ко всем файлам проекта. На рис. 2.8 изображен проект приложения `Welcome` в окне `Package Explorer`. Узел `Welcome` представляет весь проект. В IDE можно одновременно открыть несколько проектов — каждый будет представлен узлом верхнего уровня. Внутри узла проекта все содержимое делится на папки и файлы. В этой главе используются только файлы из папки `res`, которая будет рассмотрена в разделе 2.4.4; другие папки будут описаны в последующих главах, когда мы начнем их использовать.

Здесь находятся окна редактора (например, макетный редактор для файла activity_main.xml) Панель Outline со списком свойств выделенного элемента



Центральный столбец занят интерфейсом со вкладками окон
Problems, Javadoc, Declaration, Console и LogCat

Рис. 2.7. Проект Welcome в Android Developer Tools

2.4.2. Окна редактора

Справа от окна Package Explorer на рис. 2.7 расположено окно макетного редактора (Graphical Layout). Если сделать двойной щелчок на файле в окне Package Explorer, его содержимое открывается в окне соответствующего редактора (в зависимости от типа файла). Для файлов Java открывается редактор исходного кода Java. Для файла XML, представляющего графический интерфейс (такого, как activity_main.xml), открывается макетный редактор.

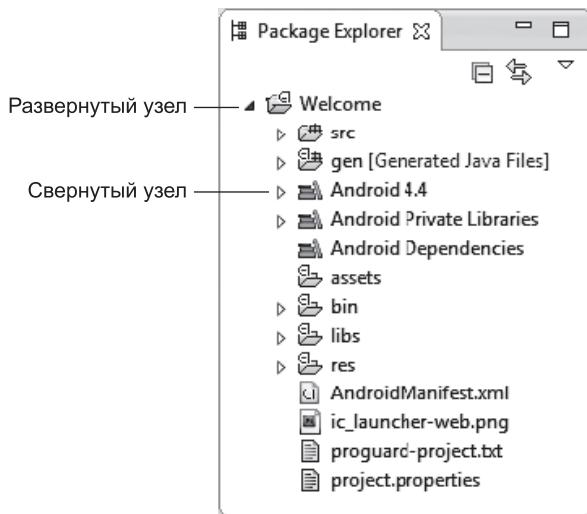


Рис. 2.8. Окно Project Explorer

2.4.3. Окно структуры

Окно структуры Outline отображается в правой части IDE (рис. 2.7). В этом окне отображается информация о редактируемом файле. Для графического интерфейса в нем выводятся все элементы, образующие интерфейс. Для класса Java выводится имя класса, все его методы и поля.

2.4.4. Файлы ресурсов приложения

Файлы макетов — такие, как `activity_main.xml` (из папки проекта `res/layout`), — считаются *ресурсами приложения* и хранятся в папке `res` проекта. В этой папке находятся вложенные папки для разных типов ресурсов. В табл. 2.3 перечислены папки, используемые в проекте Welcome; другие папки (`menu`, `animator`, `anim`, `color`, `raw` и `xml`) будут представлены позже, когда они будут использоваться в книге.

Таблица 2.3. Папки ресурсов, используемые в этой главе

Папка	Описание
<code>drawable</code>	Папки с именами, начинающимися с <code>drawable</code> , обычно содержат изображения. В них также могут храниться файлы XML, описывающие геометрические фигуры и другие типы экранных объектов (например, изображения, представляющие нажатое и ненажатое состояние кнопки)
<code>layout</code>	Папки с именами, начинающимися с <code>layout</code> , содержат файлы XML с описанием графического интерфейса — например, файл <code>activity_main.xml</code>

Папка	Описание
values	Папки с именами, начинающимися с values, содержат файлы со значениями массивов (<code>arrays.xml</code>), цветов (<code>colors.xml</code>), размеров (<code>dimen.xml</code> ; такие значения, как ширина, высота и размеры шрифтов), строки (<code>strings.xml</code>) и стили (<code>styles.xml</code>). Эти имена файлов используются по общепринятым соглашениям, но не являются обязательными — на самом деле все ресурсы этих типов можно разместить в одном файле. Все эти жестко фиксированные данные рекомендуется определять в виде ресурсов, чтобы их можно было изменить без модификации исходного кода Java. Например, если ссылки на некую метрику встречаются в нескольких местах кода, проще один раз отредактировать файл ресурсов, чем искать все вхождения фиксированного значения в исходных файлах Java

2.4.5. Макетный редактор

При создании проекта IDE открывает файл `activity_main.xml` приложения в макетном редакторе (рис. 2.9). Также макетный редактор можно открыть двойным щелчком на файле `activity_main.xml` в папке `res/layout` проекта приложения.

Выбор типа экрана

Приложения Android могут работать на многих видах устройств. В этой главе мы спроектируем графический интерфейс для телефона на базе Android. Как упоминалось в разделе «Подготовка», для этой цели будет использоваться виртуальное устройство (AVD), эмулирующее телефон Google Nexus 4. Макетный редактор поддерживает многочисленные конфигурации устройств для разных размеров и разрешений экранов, которые могут использоваться для проектирования интерфейса. В этой главе мы воспользуемся заранее определенным эмулятором Nexus 4, который выбирается в списке устройств на рис. 2.9. Это не означает, что приложение может выполняться только на устройстве Nexus 4, просто этот дизайн используется для устройств, сходных по размеру экрана и разрешению с Nexus 4. В следующих главах вы узнаете, как спроектировать графический интерфейс, нормально масштабируемый на широком диапазоне устройств.

2.4.6. Графический интерфейс по умолчанию

Графический интерфейс по умолчанию (рис. 2.9) для шаблона Blank Page состоит из компонента `RelativeLayout` со светло-серым фоном (определяется темой, выбранной при создании проекта) и компонентом `TextView` с текстом "Hello world!". Компонент `RelativeLayout` размещает компоненты графического интерфейса *относительно друг друга или самого макета* — например, при помощи `RelativeLayout` можно указать, что компонент должен находиться ниже другого компонента, с горизонтальным выравниванием по центру. Компонент `TextView` предназначен

для вывода текста. Все упомянутые компоненты будут более подробно рассмотрены в разделе 2.5.

Палитра содержит виджеты (компоненты графического интерфейса), макеты и другие элементы, которые можно перетащить на холст

Раскрывающийся список устройств, для которых предназначен дизайн приложения (выберите **Nexus 4** для этой главы)

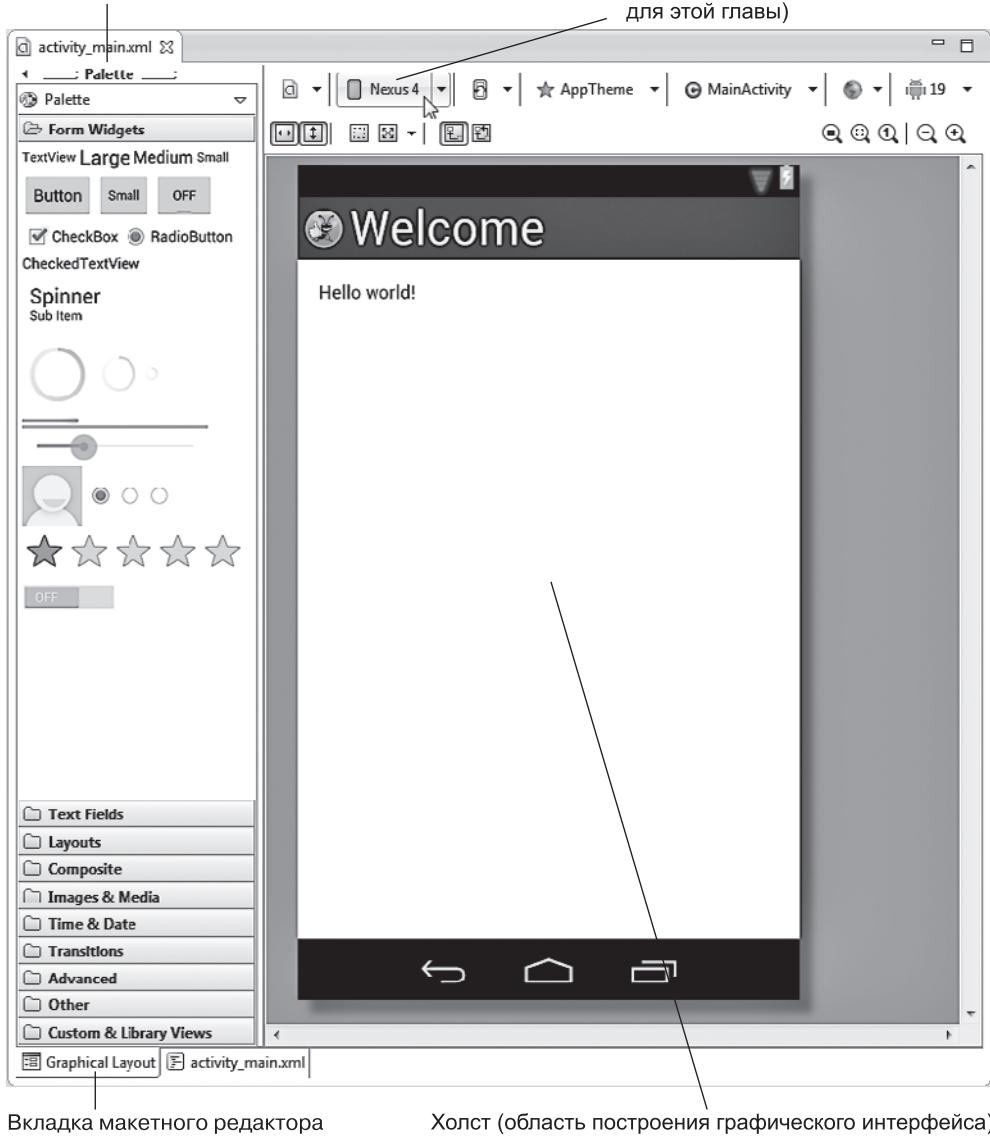


Рис. 2.9. Графический интерфейс приложения по умолчанию
в макетном редакторе

2.5. Построение графического интерфейса приложения

А теперь создадим графический интерфейс пользователя для приложения Welcome. Макетный редактор (*Graphical Layout*) позволяет создать графический интерфейс пользователя путем перетаскивания в окно приложения компонентов GUI, таких как `Button`, `TextView`, `ImageView` и др. По умолчанию, описание макета для шаблона Blank App хранится в XML-файле с именем `activity_main.xml` в папке `res/layout`. В этой главе мы воспользуемся макетным редактором и окном структуры `Outline` для построения приложения и *не будем* изучать генерированную разметку. Инструментарий разработчика Android усовершенствовался до такого состояния, что в большинстве случаев вам не нужно напрямую работать с разметкой.

2.5.1. Добавление изображений в проект

Для нашего приложения в проект необходимо включить «фирменного» жука Deitel (`bug.png`) и логотип Android (`android.png`). Эти изображения хранятся в папке `images/Welcome` примеров книги. Имена файлов графических ресурсов (а также всех остальных ресурсов, о которых вы узнаете в последующих главах) должны записываться *в нижнем регистре*.

Так как экраны устройств Android обладают разными размерами, разрешением и плотностью пикселов (DPI, Dot Per Inch), разработчик обычно предоставляет изображения с разными разрешениями, а операционная система выбирает графику на основании плотности пикселов устройства. По этой причине папка `res` вашего проекта содержит несколько вложенных папок, имена которых начинаются с префикса `drawable`. В этих папках хранятся изображения для разных плотностей пикселов (табл. 2.4).

Таблица 2.4. Плотности пикселов устройств Android

Плотность	Описание
<code>drawable-ldpi</code>	Низкая плотность — приблизительно 120 точек на дюйм
<code>drawable-mdpi</code>	Средняя плотность — приблизительно 160 точек на дюйм
<code>drawable-hdpi</code>	Высокая плотность — приблизительно 240 точек на дюйм
<code>drawable-xhdpi</code>	Очень высокая плотность — приблизительно 320 точек на дюйм
<code>drawable-xxhdpi</code>	Сверхвысокая плотность — приблизительно 480 точек на дюйм
<code>drawable-xxxhdpi</code>	Ультравысокая плотность — приблизительно 640 точек на дюйм

Графика для устройств, сходных по плотности пикселов с телефоном Google Nexus 4, который мы используем для нашего AVD, размещается в папке `drawable-hdpi`. Изображения для устройств с более высокой плотностью пикселов (например, некоторых телефонов и планшетов) находятся в папках `drawable-xhdpi` или `drawable-xxhdpi`.

Изображения для экранов со средней и низкой плотностью пикселов старых устройств Android размещаются в папках `drawable-mdpi` и `drawable-ldpi` соответственно.

В этом приложении мы предоставляем только одну версию каждого изображения. Если Android не находит изображения в подходящей папке `drawable`, то версия изображения из другой папки масштабируется до нужной плотности (вверх или вниз, как потребуется).



ПРИМЕЧАНИЕ 2.1

Изображения низкого разрешения плохо масштабируются. Чтобы графика выглядела normally, устройству с высокой плотностью пикселов нужны изображения с более высоким разрешением, чем устройству с низкой плотностью.



ПРИМЕЧАНИЕ 2.2

За подробной информацией о поддержке разных размеров экранов в Android обращайтесь по адресу http://developer.android.com/guide/practices/screens_support.html.

Чтобы добавить изображения в проект, выполните следующие действия.

1. В окне Package Explorer откройте папку `res` проекта.
2. Найдите и откройте папку `Welcome` в папке `images` в вашей файловой системе. Перетащите файлы изображений в папку `res/drawable-hdpi`. Убедитесь в том, что в открывшемся диалоговом окне `File Operation` выбран режим `Copy Files`, и нажмите `OK`. Желательно использовать изображения в формате PNG, но форматы JPG и GIF тоже поддерживаются.

Теперь скопированные файлы можно использовать в приложении.

2.5.2. Изменение свойства Id компонентов `RelativeLayout` и `TextView`

Когда графический интерфейс отображается в макетном редакторе, вы можете использовать окно свойств (`Properties`) в нижней части окна `Outline` для настройки свойств выбранного макета или компонента без прямого редактирования XML. Чтобы выбрать макет или компонент, щелкните на нем в макетном редакторе или выделите соответствующий узел в окне `Outline` (рис. 2.10).



Рис. 2.10. Графический интерфейс приложения по умолчанию в макетном редакторе

Переименуйте все макеты и компоненты и присвойте им содержательные имена, особенно если вы собираетесь работать с ними на программном уровне (как мы будем делать в будущих приложениях). Имя каждого объекта определяется его свойством `Id`. Это свойство может использоваться для обращения к компонентам и их изменения, не зная их точного местоположения в XML. Как вы вскоре увидите, свойство `Id` также может применяться для относительного позиционирования компонентов в `RelativeLayout`.

Выделите компонент `RelativeLayout`, а затем в верхней части окна свойств задайте его свойству `Id` значение

```
@+id/welcomeRelativeLayout
```

Символ `+` в конструкции `@+id` определяет создание нового идентификатора (имя переменной), который указан справа от символа `/`. Затем выделите компонент `TextView` и задайте его свойству `Id` значение

```
@+id/welcomeTextView
```

Окно свойств должно выглядеть так, как показано на рис. 2.11.

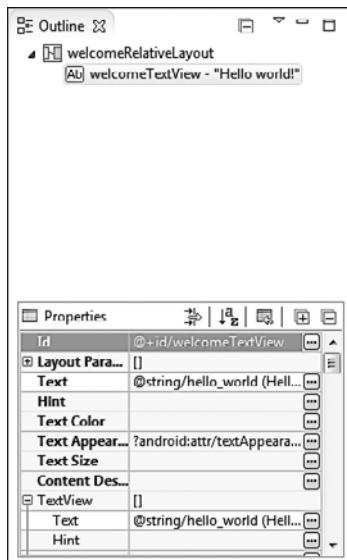


Рис. 2.11. Окно свойств после изменения свойств `Id` компонентов `RelativeLayout` и `TextView` в графическом интерфейсе по умолчанию

2.5.3. Настройка компонента `TextView`

Графический интерфейс по умолчанию приложения `Welcome` уже содержит компонент `TextView`, поэтому мы просто изменим его свойства.

Настройка свойства Text компонента TextView с помощью строковых ресурсов

Согласно документации Android по ресурсам приложений (<http://developer.android.com/guide/topics/resources/index.html>), считается хорошим тоном хранить строки, массивы строк, изображения, цвета, размеры шрифтов, метрики и другие ресурсы приложения во вложенных папках в папках `res` проекта, чтобы их можно было использовать отдельно от Java-кода приложения. Такой способ хранения называется **экстернализацией**. Например, при экстернализации цветовых значений для перекраски всех компонентов, использующих один и тот же цвет, достаточно просто изменить значение в центральном файле ресурса.

Если вы хотите локализовать свое приложение на нескольких языках, сохраните строки *отдельно* от кода приложения, чтобы в дальнейшем их можно было легко изменить. В папке `res` проекта находится подпапка `values`, в которой помещен файл `strings.xml`. Этот файл предназначен для хранения строк. Чтобы сформировать локализованные строки для других языков, создайте отдельные папки `values` для каждого используемого языка (раздел 2.8).

Чтобы установить значение свойства Text компонента `TextView`, создадим новый строковый ресурс в файле `strings.xml`.

1. Убедитесь в том, что выделен компонент `TextView`.
2. В окне свойств найдите свойство `Text`, щелкните на значении по умолчанию, затем щелкните на кнопке с многоточием. Эта кнопка, расположенная справа от значения, открывает диалоговое окно `Resource Chooser`.
3. В диалоговом окне `Resource Chooser` нажмите `New String...`, чтобы отобразить диалоговое окно `Create New Android String`, показанное на рис. 2.12.
4. Заполните поля `String` и `New R.string` (см. рис. 2.12), установите флажок `Replace in all XML file for different configurations`, потом нажмите `OK`, чтобы закрыть диалоговое окно и вернуться к окну `Resource Chooser`. В поле `String` содержится текст, который будет отображаться в компоненте `TextView`, а в поле `R.string` — имя строкового ресурса, чтобы на него можно было ссылаться из свойства `Text` компонента `TextView`.
5. Новый строковый ресурс `welcome` автоматически выделяется. Нажмите `OK`, чтобы использовать этот ресурс.

Свойство `Text` в окне свойств должно выглядеть так, как показано на рис. 2.13. Запись в форме `@string` свидетельствует о том, что существующий строковый ресурс будет выбран в файле `strings.xml` (из папки `res/values` проекта), а имя `welcome` определяет выбираемый строковый ресурс.

Настройка свойства Text Size компонента TextView — пиксели, независимые от плотности и масштабирования

Размеры компонентов GUI и текста на экране Android могут определяться с помощью различных единиц измерения (табл. 2.5). Документация, описывающая различные размеры экранов, находится на веб-сайте по адресу

developer.android.com/guide/practices/screens_support.html

и рекомендует для определения размеров компонентов GUI и других экраных элементов использовать *пиксели, независимые от плотности*, а размеры шрифтов задавать с помощью *пикселов, независимых от масштабирования*.

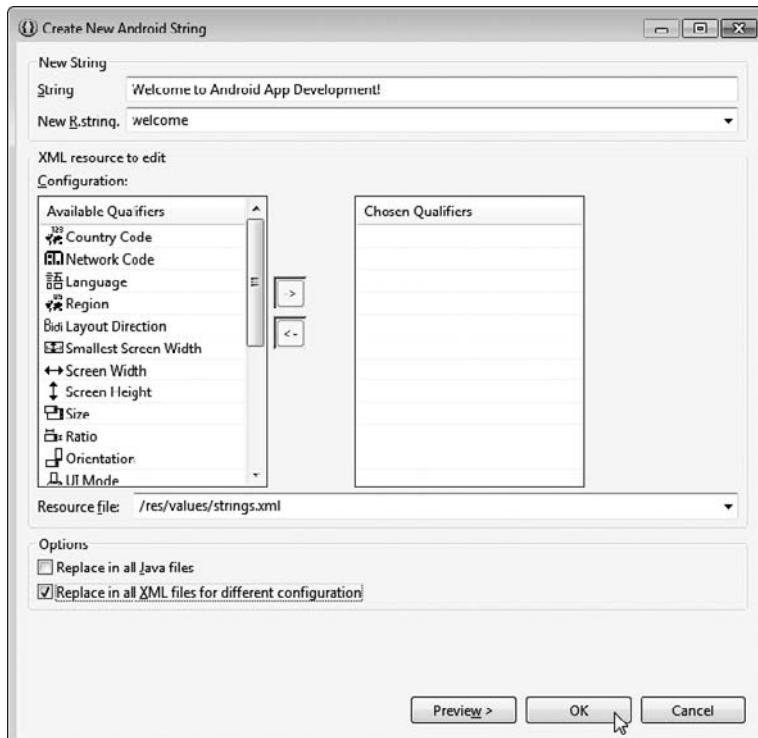


Рис. 2.12. Диалоговое окно Create New Android String



Рис. 2.13. Окно свойств после изменения свойства Text компонента TextView

Таблица 2.5. Единицы измерения

Единица измерения	Описание
px	Пиксель
dp или dip	Пиксель, независимый от плотности
sp	Пиксель, независимый от масштабирования
in	Дюймы
mm	Миллиметры

Задание размеров в пикселях, независимых от плотности (`dpx` или `dip`), позволяет платформе Android автоматически *масштабировать* графический интерфейс пользователя в зависимости от плотности пикселов экрана физического устройства. Размер пикселя, независимого от плотности, эквивалентен размеру физического пикселя на экране с разрешением 160 dpi (точек на дюйм). На экране с разрешением 240 dpi размер пикселя, независимого от плотности, будет масштабироваться с коэффициентом 240/160 (то есть 1,5). Таким образом, компонент, размер которого составляет 100 пикселов, независимых от плотности, будет масштабирован до размера в 150 *физических пикселов* на таком экране. На экране с разрешением 120 точек на дюйм каждый независимый от плотности пиксель масштабируется с коэффициентом 120/160 (то есть 0,75). Значит, 100 независимых от плотностей пикселов превратятся на таком экране в 75 физических пикселов. Пиксели, *независимые от масштабирования*, масштабируются так же, как и пиксели, независимые от плотности, но их масштаб зависит также и от *предпочитаемого размера шрифта*, выбиремого пользователем.

А теперь увеличим размер шрифта для компонента `TextView` и добавим небольшой отступ над компонентом `TextView`, чтобы отделить текст от края экрана устройства. Чтобы изменить размер шрифта:

1. Убедитесь в том, что в проекте выбран компонент `welcomeTextView`.
2. Найдите свойство `Text Size` в окне свойств. Щелкните на кнопке с многоточием справа от значения свойства, чтобы открыть диалоговое окно `Resource Chooser`.
3. В диалоговом окне `Resource Chooser` нажмите `New Dimension...`
4. В открывшемся диалоговом окне введите в поле `Name` имя `welcome_textsize`, а в поле `Value` — значение `40sp`. Нажмите `OK`, чтобы закрыть диалоговое окно и вернуться к окну `Resource Chooser`. Буквы `sp` в значении `40sp` означают, что речь идет о пикселях, *независимых от масштабирования*. Буквы `dp` в значении (например, `10dp`) обозначают *пиксели, независимые от плотности*.

Новый ресурс с именем `welcome_textsize` выбирается автоматически. Нажмите `OK`, чтобы использовать этот ресурс.

Настройка дополнительных свойств `TextView`

Используйте окно свойств для задания следующих дополнительных свойств компонента `TextView`.

- ❑ Задайте свойству `Text Color` значение `@android:color/holo_blue_dark`. В Android существуют заранее определенные цветовые ресурсы. Когда вы вводите строку `@android:color/` в поле значения свойства `Text Color`, открывается список цветовых ресурсов (рис. 2.14). Выберите в списке пункт `@android:color/holo_blue_dark`, чтобы текст был ярко-синим.

Чтобы текст, состоящий из нескольких строк, выравнивался по центру в `TextView`, задайте его свойству `Gravity` значение `center`. Для этого щелкните на поле `Value` данного свойства, после чего щелкните на кнопке с многоточием, чтобы вызвать диалоговое

окно Select Flag Values с вариантами значений Gravity (рис. 2.15). Щелкните на кнопке на флагажке center, после чего нажмите OK, чтобы сохранить введенное значение.

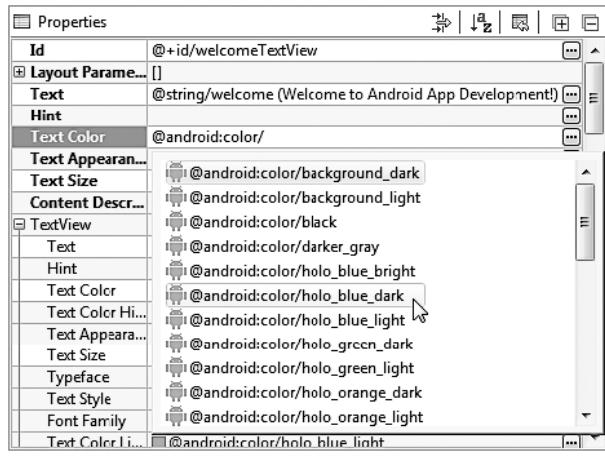


Рис. 2.14. Задание свойству Text Color компонента TextView значения @android:color/holo_blue_dark

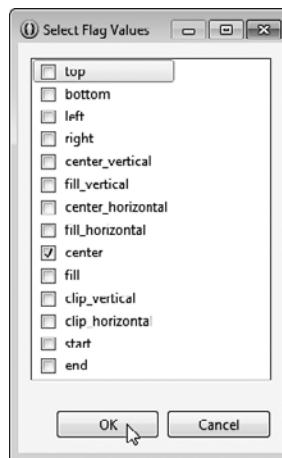


Рис. 2.15. Допустимые значения свойства Gravity объекта

Окно макетного редактора должно выглядеть так, как показано на рис. 2.16.

Добавление компонентов ImageView для вывода изображений

А теперь в графический интерфейс пользователя добавим два компонента ImageView, с помощью которых отображаются два изображения, добавленные в проект в разделе 2.5.1. Для этого следует перетащить компоненты ImageView из

раздела *Images&Media* палитры в графический интерфейс приложения. Выполните следующие действия.

1. Откройте категорию *Images & Media* на палитре и перетащите компонент *ImageView* на холст (рис. 2.17). Новый компонент *ImageView* появится ниже узла *welcomeTextView*. При перетаскивании компонента в область холста макетный редактор отображает зеленые линейки, а на экране появляется подсказка. Линейки упрощают позиционирование компонентов в графическом интерфейсе, а подсказка сообщает, как будет настроен компонент, если отпустить его в текущей позиции мыши. Подсказка на рис. 2.17 сообщает, что компонент *ImageView* будет выровнен по центру в родительском макете (также обозначается пунктирной линейкой, проходящей от верха до низа графического интерфейса) и будет размещен ниже компонента *welcomeTextView* (обозначается пунктирной линейкой со стрелкой на конце).

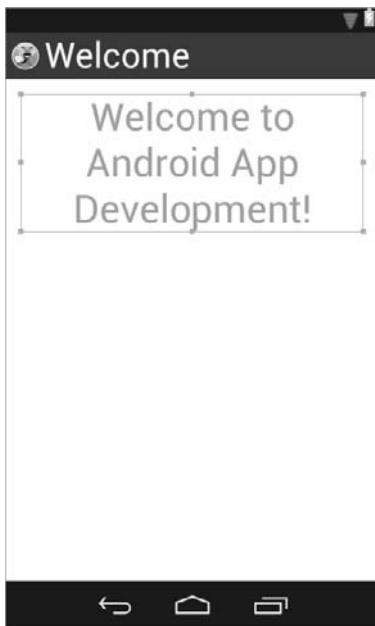


Рис. 2.16. Графический интерфейс после завершения настройки *TextView*



Рис. 2.17. Перетаскивание компонента *ImageView* в графический интерфейс

2. При отпускании кнопки мыши появляется диалоговое окно *Resource Chooser* (рис. 2.18), в котором можно выбрать ресурс изображения. Для каждого изображения, помещенного в папку *drawable*, IDE генерирует идентификатор ресурса (то есть имя ресурса), по которому можно ссылаться на изображение в макете и в коде. Идентификатор ресурса представляет собой имя файла

изображения без расширения — так, для файла android.png будет использоваться идентификатор ресурса `android`. Выберите идентификатор `android` и нажмите **OK**, чтобы выводилось изображение робота. При добавлении в графический интерфейс нового компонента он автоматически выделяется, а его свойства отображаются в окне свойств.

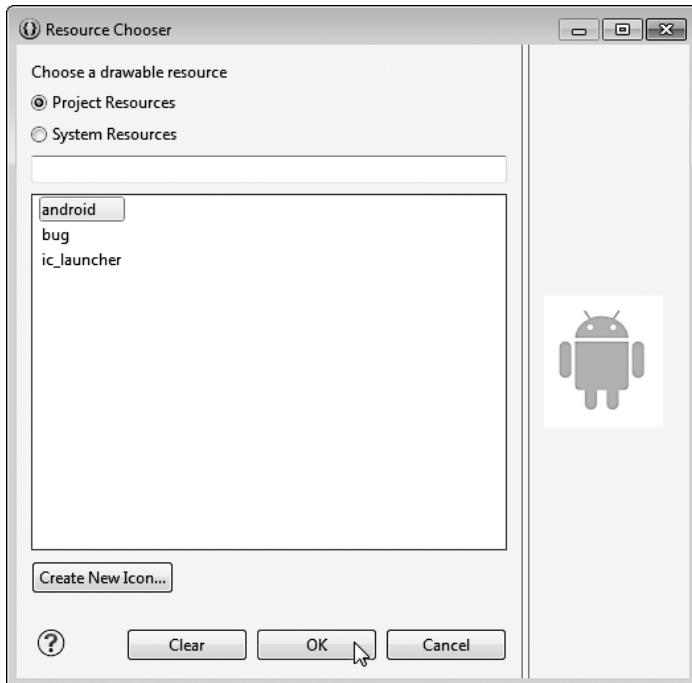


Рис. 2.18. Выбор ресурса изображения `android` в диалоговом окне `Resource Chooser`

3. IDE по умолчанию задает свойству `Id` компонента `ImageView` значение `@+id/imageView1`. Измените его на `@+id/droidImageView`. На экране появляется диалоговое окно `Update References?` с предложением подтвердить операцию переименования. Нажмите **Yes**. Открывается окно `Rename Resource` с перечнем всех вносимых изменений. Нажмите **OK**, чтобы завершить операцию переименования.
4. Повторите пункты 1–3 для создания компонента `bugImageView`. Для этого компонента разместите компонент `ImageView` под `droidImageView`, выберите ресурс с изображением жука в диалоговом окне `Resource Chooser` и задайте его свойству `Id` значение `@+id/bugImageView` в окне свойств. Сохраните файл.

Теперь графический интерфейс приложения должен выглядеть так, как показано на рис. 2.19.

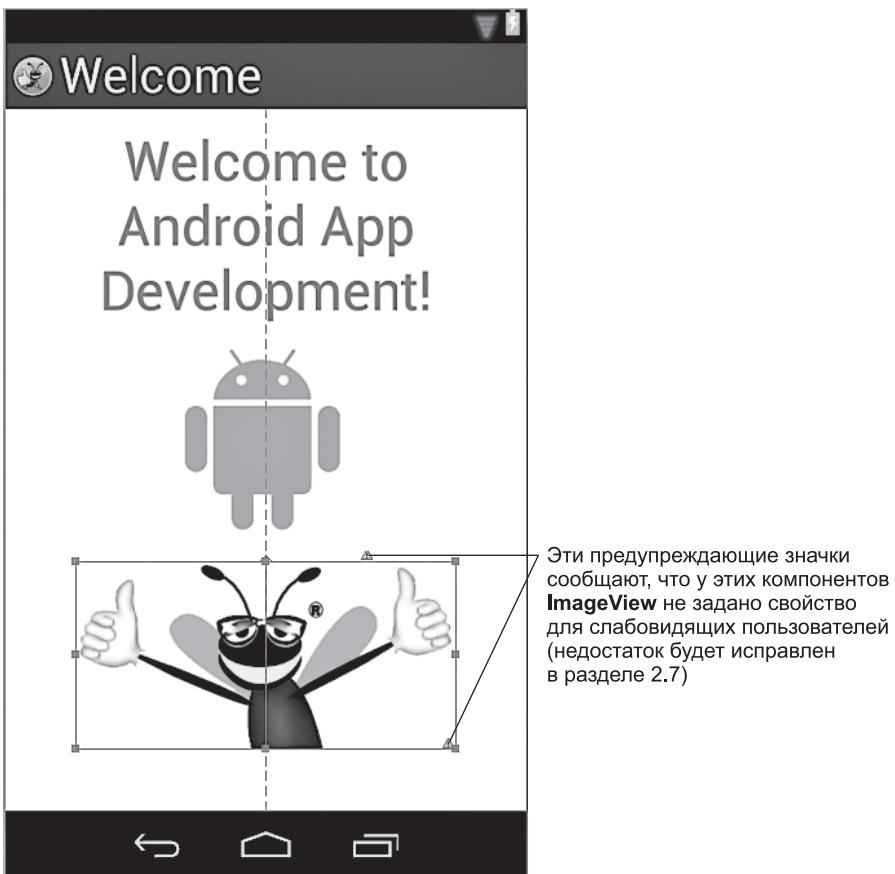


Рис. 2.19. Готовый графический интерфейс

2.6. Выполнение приложения Welcome

Чтобы выполнить приложение на виртуальном устройстве AVD для телефона, выполните действия из раздела 1.9.1. На рис. 2.20 показано запущенное приложение на виртуальном устройстве Nexus 4 AVD, настроенном в разделе «Подготовка». Приложение представлено в *портретной* ориентации (высота устройства больше его ширины). Хотя устройство или AVD можно развернуть в альбомную ориентацию (ширина больше высоты), графический интерфейс приложения не рассчитан на эту ориентацию. В следующей главе будет показано, как ограничить ориентацию приложения, а в дальнейших главах вы научитесь создавать динамичные графические интерфейсы, поддерживающие оба варианта ориентации.

При желании повторите действия из раздела 1.9.3, чтобы выполнить приложение на устройстве Android. Хотя приложение будет работать в AVD планшетных устройств

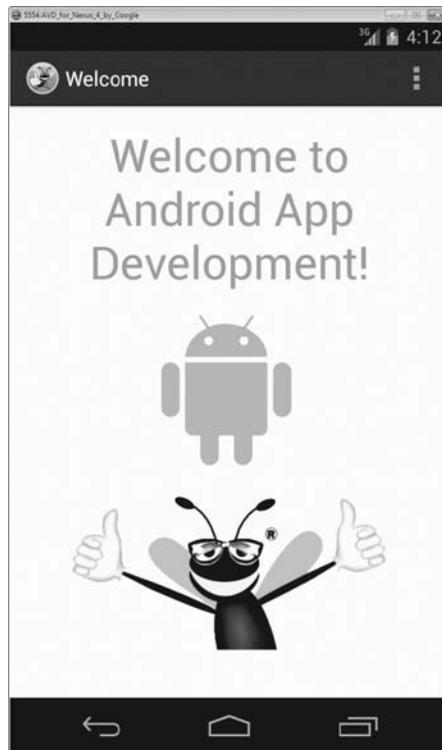


Рис. 2.20. Приложение Welcome в AVD

или на физических планшетах, оно будет занимать лишь небольшую часть экрана. Как правило, в приложениях, способных работать на телефонах и на планшетах, также определяется планшетный макет, более эффективно использующий пространство экрана; вскоре вы увидите, как это делается.

2.7. Обеспечение доступности приложения

В Android реализованы средства доступности, упрощающие работу с устройствами для пользователей с ограниченными возможностями. Для слабовидящих пользователей функция Android TalkBack проговаривает экранный текст (или текст, предоставленный разработчиком при визуальном или программном построении графического интерфейса), чтобы помочь пользователю понять назначение компонента. Также существует функция Explore by Touch, с помощью которой пользователь может услышать, что находится на экране в точке касания. Если пользователь касается компонента в режиме TalkBack, воспроизводится вспомогательный текст, а устройство вибрирует, чтобы предоставить обратную связь пользователям с ослабленным слухом. Все стандартные компоненты графического интерфейса Android

поддерживают средства доступности. Если компонент выводит текст, то TalkBack по умолчанию читает этот текст — например, когда пользователь касается компонента `TextView`, TalkBack проговаривает текущее содержимое поля. Функция TalkBack включается в приложении `Settings` (раздел `Accessibility`). Здесь же можно включить и другие средства доступности Android — например, средства доступности по умолчанию (такие, как увеличенный размер текста по умолчанию и применение жестов для увеличения области экрана). К сожалению, функция TalkBack в настоящее время не поддерживается в AVD, и чтобы услышать, как TalkBack читает текст, приложение необходимо запустить на физическом устройстве. При включении TalkBack Android предоставляет возможность просмотреть руководство по использованию функций TalkBack и `Explore by Touch`.

Включение функции TalkBack для компонентов `ImageView`

В приложении `Welcome` нам не нужен более содержательный текст для `TextView`, потому что TalkBack читает содержимое `TextView`. Однако у компонента `ImageView` не существует текста, который бы читался экранным диктором, если только вы не предоставите его сами. В Android рекомендуется обеспечить возможность использования каждого экранного компонента с TalkBack — для этого следует задать значение свойства `Content Description` каждого компонента, не отображающего текст. По этой причине IDE предупреждает нас о проблеме, отображая маленькие значки () в макетном редакторе рядом с каждым компонентом `ImageView`. Эти предупреждения (они генерируются служебной программой `Android Lint`) сообщают, что мы не задали свойство `Content Description` каждого изображения. Текст этого свойства должен помочь пользователю понять назначение компонента. Для компонентов `ImageView` текст должен описывать изображение.

Чтобы добавить описание `Content Description` для каждого компонента `ImageView` (и избавиться от предупреждений `Android Lint`), выполните следующие действия.

1. Выделите компонент `droidImageView` в макетном редакторе.
2. В окне свойств щелкните на кнопке с многоточием справа от свойства `Content Description`, чтобы открыть диалоговое окно `Resource Chooser`.
3. Нажмите `New String...`, чтобы отобразить диалоговое окно `Create New Android String`.
4. В поле `String` введите текст `"Android logo"`, а в поле `R.string` — текст `android_logo`. Нажмите кнопку `OK`.
5. Новый строковый ресурс `android_logo` выбирается в диалоговом окне `Resource Chooser`. Нажмите `OK`, чтобы назначить этот ресурс значением свойства `Content Description` компонента `droidImageView`.
6. Повторите эти действия для компонента `bugImageView`, но в диалоговом окне `Create New Android String` введите в поле `String` текст `"Deitel double-thumbs-up bug logo"`, а в поле `R.string` — текст `"deitel_logo"`. Сохраните файл.

С заполнением поля `Content Description` каждого компонента `ImageView` в макетном редакторе исчезает предупреждающий значок () для этого компонента `ImageView`.

Тестирование приложения с включенной функцией TalkBack

Запустите это приложение на устройстве с включенной функцией TalkBack. Коснитесь компонента `TextView` и каждого из компонентов `ImageView` и прослушайте соответствующий текст.

Подробнее о доступности

Некоторые приложения динамически генерируют компоненты графического интерфейса в ответ на действия пользователя. Для таких компонентов текст доступности может задаваться на программном уровне. Следующие страницы документации разработчика Android содержат дополнительную информацию о средствах доступности Android и контрольный список, по которому следует действовать при разработке доступных приложений:

<http://developer.android.com/design/patterns/accessibility.html>

<http://developer.android.com/guide/topics/ui/accessibility/index.html>

<http://developer.android.com/guide/topics/ui/accessibility/checklist.html>

2.8. Интернационализация приложения

Как известно, устройства Android используются во всем мире. Чтобы ваши приложения охватывали как можно большую аудиторию, подумайте о том, чтобы адаптировать их для разных культур и языков — этот процесс называется *интернационализацией*. Например, если вы собираетесь распространять свое приложение во Франции, переведите его ресурсы (текст, аудиофайлы и т. д.) на французский язык. Также в зависимости от локального контекста в приложении могут использоваться другие цвета, другая графика и звуки. Для каждого локального контекста создается новый специализированный набор ресурсов. Когда пользователь запускает приложение, Android автоматически находит и загружает ресурсы, соответствующие настройкам локального контекста устройства.

Локализация

Важнейшее преимущество определения строковых значений в виде строковых ресурсов (как в нашем приложении) заключается в том, что вы можете легко локализовать свое приложение, создавая дополнительные файлы ресурсов в формате XML для других языков. Во всех файлах используются одни и те же имена ресурсов строк, но с разными переводами. Android выбирает ресурсный файл в зависимости от основного языка, выбранного на устройстве пользователя.

Имена папок с локализованными ресурсами

XML-файлы ресурсов, содержащие локализованные строки, размещаются во вложенных папках папки `res` проекта. Android использует специальные правила назначения имен папок для автоматического выбора правильных локализованных ресурсов — например, папка `values-fr` содержит файл `strings.xml` для французского

языка, а папка `values-es` содержит файл `strings.xml` для испанского языка. В именах папок также может присутствовать региональная информация — обе папки, `values-en-rUS` и `values-en-rGB` — содержат файл `strings.xml` для английского языка, но первая предназначена для США, а вторая — для Великобритании. Если локализованные ресурсы для нужного локального контекста отсутствуют, Android использует ресурсы приложения по умолчанию (то есть ресурсы из папки `values` в папке `res`). Правила назначения имен папок с альтернативными ресурсами будут более подробно рассмотрены далее.

Добавление папки локализации в проект приложения

Прежде чем добавлять в приложение `Welcome` локализованную версию файла `strings.xml`, содержащую строки на испанском языке, необходимо добавить в проект папку `values-es`. Это делается так:

1. В окне `Package Explorer` щелкните правой кнопкой мыши на папке `res` проекта и выберите команду `New > Folder`, чтобы открыть диалоговое окно `New Folder`.
2. В поле `Folder name:` диалогового окна введите строку `values-es`. Нажмите `Finish`.

Эти шаги повторяются для каждого языка, который вы намерены поддерживать, с соответствующим именем папки `values-XX`.

Копирование файла `strings.xml` в папку `values-es`

Затем файл `strings.xml` копируется из папки `values` в папку `values-es`.

Это делается так:

1. В окне `Package Explorer` откройте папку `values` в папке `res`, щелкните правой кнопкой мыши на файле `strings.xml` и выберите команду `Copy`.
2. Щелкните правой кнопкой мыши на папке `values-es` и выберите команду `Paste`, чтобы вставить в папку копию файла `strings.xml`.

Локализация строк

Графический интерфейс нашего приложения содержит один компонент `TextView`, в котором выводится текстовое сообщение, и два компонента `ImageView` со строками, описывающими их содержимое. Все эти строки определяются как ресурсы в файле `strings.xml`. Теперь мы можем перевести эти строки в новой версии файла `strings.xml`. Компании, занимающиеся разработкой приложений, часто имеют в штате переводчиков или пользуются услугами других компаний для выполнения перевода. Более того, на сайте `Google Play Developer Console` (используемом для публикации приложений в магазине `Google Play`) приведет список компаний, предоставляющих услуги перевода.

За дополнительной информацией о `Google Play Developer Console` обращайтесь к главе 9 и по адресу

developer.android.com/distribute/googleplay/publish/index.html

В нашем приложении строки

"Welcome to Android App Development!"

"Android logo"

"Deitel double-thumbs-up bug logo"

будут заменены строками на испанском языке

"¡Bienvenido al Desarrollo de App Android!"

"Logo de Android"

"El logo de Deitel que tiene el insecto con dedos pulgares hacia arriba"

Это делается так:

1. В окне Package Explorer сделайте двойной щелчок на файле strings.xml в папке values-es, чтобы открыть редактор ресурсов Android. Выберите ресурс welcome (рис. 2.21).

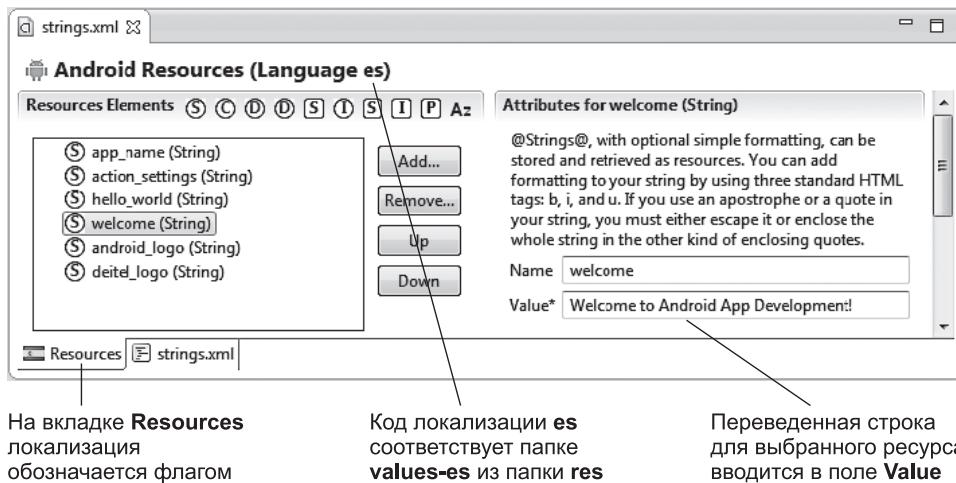


Рис. 2.21. Редактор ресурсов Android с выбранным ресурсом строки welcome

2. В поле Value замените английскую строку "Welcome to Android App Development!" испанской строкой "¡Bienvenido al Desarrollo de App Android!". Если у вас возникнут трудности с вводом специальных испанских символов с клавиатуры, скопируйте испанские строки из нашего файла res/values-es/strings.xml в финальной версии приложения Welcome (папка WelcomeInternationalized в примерах главы). Чтобы вставить испанскую строку в поле Value, выделите английский текст, щелкните на нем правой кнопкой мыши и выберите команду Paste.
3. Выделите ресурс android_logo и замените строку в поле Value на "Logo de Android".
4. Наконец, выделите ресурс deitel_logo и замените строку в поле Value на "El logo de Deitel que tiene el insecto con dedos pulgares hacia arriba".

5. Удалите ресурсы строк `app_name`, `action_settings` и `hello_world`, последовательно выделяя их и щелкая на кнопке `Remove....`. Вам будет предложено подтвердить каждую операцию удаления. Эти три ресурса были включены в файл `strings.xml` по умолчанию при создании проекта. В проекте используется только ресурс `app_name`. Вскоре мы объясним, почему удалили его.
6. Сохраните файл `strings.xml` командой `File > Save` или щелчком на кнопке  на панели инструментов.

Тестирование приложения с испанской локализацией

Чтобы протестировать приложение с испанской локализацией, необходимо изменить языковые настройки в эмуляторе Android (или на вашем устройстве).

1. Нажмите кнопку `Home` () в эмуляторе или на устройстве.
2. Нажмите кнопку лаунчера () , найдите и коснитесь значка приложения `Settings` () .
3. В приложении `Settings` перейдите к разделу `PERSONAL`, затем откройте категорию `Language & input`.
4. Откройте пункт `Language` (первый элемент списка) и выберите в списке строку `Español (España)`.

Эмулятор или устройство переключается на испанский язык и возвращается к настройкам `Language & input`, которые теперь отображаются на испанском.

Теперь запустите приложение `Welcome` из IDE; при этом устанавливается и запускается интернационализированная версия. На рис. 2.22 показано приложение Android, локализованное для испанского языка. В самом начале работы приложения Android проверяет языковые настройки AVD (или устройства), определяет, что в AVD (или на устройстве) включен испанский язык, и использует строковые ресурсы `welcome`, `android_logo` и `deitel_logo`, определенные в файле `res/values-es/strings.xml` работающего приложения. Однако следует обратить внимание на то, что имя приложения на панели действий в верхней части приложения все равно выводится на английском языке. Дело в том, что мы не предоставили локализованную версию строкового ресурса `app_name` в файле `res/values-es/strings.xml`. Вспомните, о чем говорилось ранее: если Android не может найти локализованную версию строкового ресурса, используется версия по умолчанию из файла `res/values/strings.xml`.



ВНИМАНИЕ 2.1

Изменение имен ресурсов может привести к ошибкам времени выполнения. При загрузке локализованных ресурсов Android использует имена ресурсов по умолчанию. Создавая локализованный файл ресурсов, следите за тем, чтобы изменялись только значения ресурсов, но не их имена.

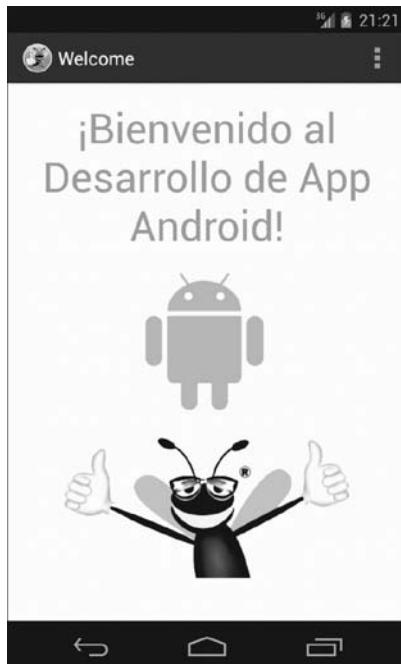


Рис. 2.22. Приложение Welcome на испанском языке в AVD для Nexus 4

Возвращение AVD (или устройства) к английскому языку

Чтобы снова переключить AVD (или устройство) на английский язык:

1. Нажмите кнопку Home (⌂) в эмуляторе или на устройстве.
2. Нажмите кнопку лаунчера (●), найдите и коснитесь значка приложения *Settings* (⚙) — теперь это приложение называется *Ajustes*.
3. Перейдите к разделу *Idioma y entrada de texto*, чтобы вызвать языковые настройки.
4. Откройте пункт *Idioma* и выберите в списке строку *English (United States)*.

TalkBack и локализация

Функция TalkBack в настоящее время поддерживает английский, испанский, итальянский, французский и немецкий языки. Если запустить приложение Welcome на устройстве с испанским языком и включенной функцией TalkBack, то TalkBack будет зачитывать испанские строки при касании компонентов графического интерфейса.

Когда вы впервые переключите устройство на испанский язык и включите TalkBack, Android автоматически загрузит ядро синтеза речи. Если TalkBack не произносит

испанские строки, значит, загрузка ядра еще не завершилась и установка продолжается. В таком случае попробуйте запустить приложение позднее.

Контрольный список локализации

За дополнительной информацией о локализации ресурсов приложения обращайтесь к документации Android Localization Checklist по адресу:

developer.android.com/distribute/googleplay/publish/localizing.html

2.9. Резюме

В этой главе мы воспользовались средой разработки Android Developer Tools IDE для построения приложения Welcome app, которое отображает текстовое сообщение и два изображения без написания какого-либо кода. Мы создали простой графический интерфейс в макетном редакторе и настроили свойства компонентов в окне свойств.

Приложение выводит текст в компоненте `TextView`, а изображения — в компонентах `ImageView`. Мы изменили компонент `TextView` графического интерфейса по умолчанию, чтобы текст выравнивался по центру, увеличенным шрифтом и в одном из цветов стандартной темы. Компоненты `ImageView` перетаскивались мышью из палитры компонентов. Как и положено, все строки и числовые значения были определены в файлах ресурсов в папке `res` проекта.

Также были представлены средства обеспечения доступности, упрощающие использование устройств людьми с ограниченными возможностями. Вы узнали, как использовать функцию TalkBack — экраный диктор, который помогает пользователю лучше понять назначение и содержимое компонента. Функция Android Explore by Touch позволяет прикоснуться к экрану, чтобы диктор TalkBack зачитал информацию о содержимом экрана рядом с точкой касания. Для компонентов `ImageView` приложения были предоставлены описания контента, которые могут использоваться функциями TalkBack и Explore by Touch.

Наконец, вы научились пользоваться средствами интернационализации Android для того, чтобы ваши приложения охватывали как можно большую аудиторию. Приложение Welcome было локализовано испанскими строками для текста `TextView` и описаниями для компонентов `ImageView`, после чего оно было протестировано на виртуальном устройстве AVD, настроенном для испанского языка.

Разработка Android сочетает в себе визуальное построение графического интерфейса с программированием на языке Java. В следующей главе будет разработано простое приложение Tip Calculator для подсчета размера чаевых в ресторанах; при этом макетный редактор будет использоваться для построения графического интерфейса, а программирование на Java — для определения поведения.

3 Приложение Tip Calculator

Знакомство с GridLayout, LinearLayout, EditText, SeekBar, обработкой событий, NumberFormat и определением функциональности приложения на языке Java

В этой главе...

- Создание графического интерфейса пользователя с помощью макетов LinearLayout И GridLayout
- Использование окна Outline среды разработки для добавления компонентов GUI в макеты LinearLayout И GridLayout
- Использование компонентов TextView, EditText И SeekBar
- Применение объектно-ориентированных возможностей Java, включая классы, объекты, интерфейсы, анонимные внутренние классы и наследование, для расширения функциональности приложений Android
- Изменение отображаемого текста путем программного взаимодействия с компонентами GUI
- Использование обработки событий при взаимодействии с пользователем с помощью компонентов EditText И SeekBar
- Постоянное отображение виртуальной клавиатуры при выполнении приложения
- Ограничение приложения портретной ориентацией

3.1. Введение

Приложение Tip Calculator (рис. 3.1, а) вычисляет и отображает чаевые на основании счета в ресторане. После ввода пользователем выставленного счета с цифровой клавиатурой приложение вычисляет и отображает размер чаевых и величину итогового счета (с учетом 15% чаевых). Пользователь может указать собственную ставку в диапазоне от 0% до 30% (по умолчанию 18%). Для этого нужно переместить ползунок компонента Seekbar, после чего обновляется величина процентной ставки, которая отображается в правой части компонента Seekbar (рис. 3.1, б). Мы выбрали пользовательский процент чаевых 18%, потому что многие рестораны в США используют эту процентную ставку для компаний, состоящих из шести человек и более. Цифровая клавиатура на рис. 3.1 может выглядеть по-другому в зависимости от версии Android вашего виртуального устройства AVD или физического устройства, а также от установки и выбора нестандартной клавиатуры для вашего устройства.

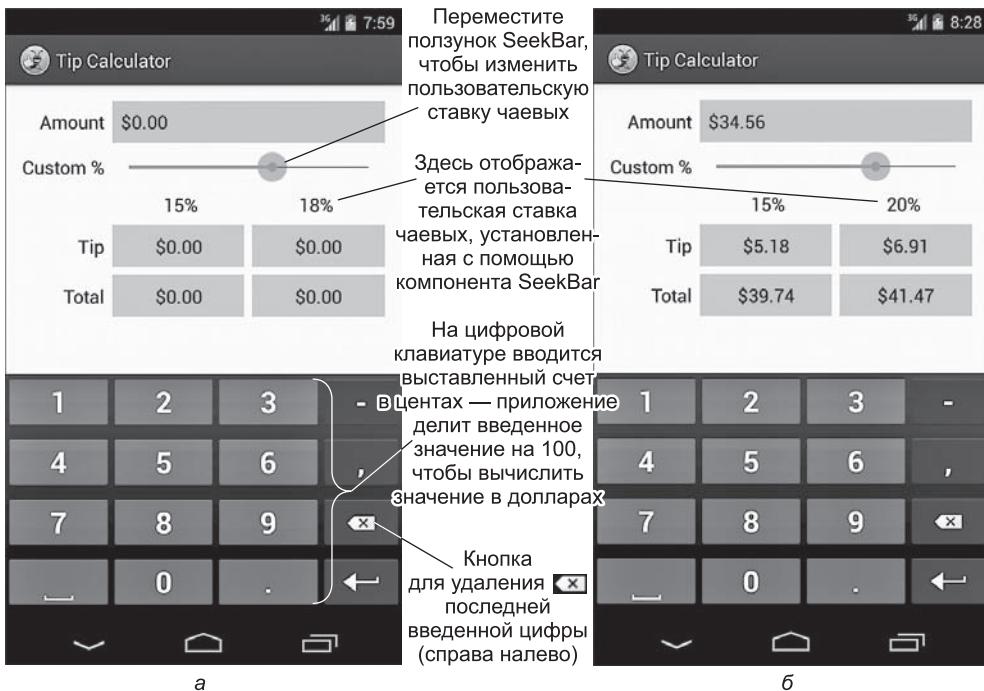


Рис. 3.1. Ввод выставленного счета и вычисление чаевых: а — исходный вид графического интерфейса; б — графический интерфейс после ввода суммы 34,56 и выбора пользовательской ставки чаевых 20%

Начнем с тестирования приложения с вычислением чаевых по стандартной и пользовательской ставке. Затем будет приведен краткий обзор технологий, применяемых для создания приложения. Графический интерфейс пользователя

(GUI) будет построен с помощью макетного редактора (Graphical Layout editor) интегрированной среды Android Developer Tools и окна Outline. Напоследок мы рассмотрим окончательный код приложения и проведем его подробный анализ. Версии разделов 3.2 и 3.4 для Android Studio размещены на сайте книги <http://www.deitel.com/books/AndroidFP2>.

3.2. Тестирование приложения Tip Calculator

Запуск и выполнение приложения

Откройте интегрированную среду Android Developer Tools и импортируйте проект Tip Calculator. Выполните следующие действия.

1. Для тестового запуска нам понадобится виртуальное устройство AVD для смартфона Nexus 4, настроенное в разделе «Подготовка». Чтобы запустить Nexus 4 AVD, выполните команду Window ▶ Android Virtual Device Manager; на экране появится диалоговое окно Android Virtual Device Manager. Выберите Nexus 4 AVD и нажмите Start..., затем нажмите Launch в открывшемся диалоговом окне Launch Options.
2. Выполните команду File ▶ Import, чтобы открыть диалоговое окно Import.
3. Импортируйте проект приложения Tip Calculator. В диалоговом окне Import раскройте узел General, выберите режим Existing Projects into Workspace и нажмите Next> для перехода к шагу Import Projects. Установите флажок Select root directory и нажмите Browse.... В диалоговом окне Browse For Folder найдите папку TipCalculator (в папке примеров книги), выделите ее и нажмите OK. Проследите за тем, чтобы флажок Copy projects into workspace не был установлен. Нажмите Finish, чтобы завершить импортацию. Проект появляется в окне Package Explorer.
4. Запустите приложение Tip Calculator. Щелкните правой кнопкой мыши на проекте TipCalculator в окне Package Explorer, затем в появившемся меню выберите команду Run As ▶ Android Application.

Ввод суммы

Введите на цифровой клавиатуре сумму 34,56. Просто наберите 3456 — приложение автоматически расположит центы в дробной части. Если при вводе будет допущена ошибка, нажмите кнопку удаления (), чтобы стереть крайнюю правую цифру. В компонентах TextView под надписями с процентами (15% и пользовательский процент — 18% по умолчанию) выводится величина чаевых и общая сумма счета с чаевыми. Эти компоненты TextView обновляются при каждом вводе или удалении цифры.

Выбор пользовательского процента чаевых

Компонент Seekbar задает пользовательский процент чаевых. Перетащите ползунок Seekbar до отметки 20% (рис. 3.1, б). В процессе перетаскивания происходит непрерывное обновление величины чаевых и общего счета для текущего процента. По

умолчанию компонент `SeekBar` позволяет выбирать значения в диапазоне от 0 до 100, но в нашем приложении шкала ограничивается максимальным значением 30.

3.3. Обзор применяемых технологий

В этом разделе представлены возможности Java и Android, используемые для построения приложения Tip Calculator. Предполагается, что вы *уже* знакомы с объектно-ориентированным программированием на языке Java. В частности, мы будем:

- ❑ использовать различные классы Android для создания объектов;
- ❑ вызывать методы объектов и классов Android;
- ❑ определять и вызывать пользовательские методы;
- ❑ использовать наследование для субклассирования класса Android `Activity`, определяющего функциональность класса Tip Calculator;
- ❑ использовать обработку событий, анонимные внутренние классы и интерфейсы для обработки взаимодействий пользователя с интерфейсом.

3.3.1. Класс Activity

В отличие от многих приложений Java, приложения Android не содержат метода `main`. Вместо этого в них используются четыре типа исполняемых компонентов — *активности* (activities), *службы* (services), *провайдеры контента* и *широковещательные приемники* (broadcast receivers). В этой главе рассматриваются активности, определяемые как субклассы `Activity` (пакет `android.app`). Пользователи взаимодействуют с активностями через *представления* (views), то есть компоненты GUI.

До выхода Android 3.0 с каждым экраном приложения обычно связывалась отдельная активность. Как будет показано в главе 5, активность может управлять несколькими *фрагментами* (fragments). На телефоне каждый фрагмент обычно занимает целый экран, а активность переключается между фрагментами на основании взаимодействий пользователя. На планшетах активности часто отображают несколько фрагментов на экран, чтобы более эффективно использовать доступное пространство.

3.3.2. Методы жизненного цикла активности

На протяжении своего существования активность может находиться в одном из нескольких состояний — *активном* (то есть выполняемом), *приостановленном* или *остановленном*. Переходы активностей между этими состояниями происходят в ответ на различные события:

- ❑ «Активная активность» отображается на экране и «обладает фокусом» — то есть взаимодействует с пользователем.
- ❑ Приостановленная активность находится на экране, но не *обладает фокусом* (например, на время отображения диалогового окна с сообщением).
- ❑ Остановленная активность *не отображается* на экране и, вероятно, будет уничтожена системой, когда потребуется освободить занимаемую ею память. Активность останавливается, когда другая активность переходит в активное состояние.

При переходах активности между этими состояниями исполнительная среда Android вызывает различные методы жизненного цикла (все эти методы определяются в классе `Activity`):

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

В ваших приложениях для *каждой* активности будет переопределяться метод `onCreate`. Этот метод вызывается исполнительной системой Android при запуске активности — то есть когда ее графический интерфейс готов к отображению, чтобы пользователь мог взаимодействовать с активностью. Также у активностей существуют другие методы жизненного цикла: `onStart`, `onPause`, `onRestart`, `onResume`, `onStop` и `onDestroy`. Большинство этих методов будет рассмотрено в дальнейших главах. Каждый переопределяемый вами метод жизненного цикла активности должен вызывать версию метода из суперкласса; в противном случае происходит *исключение*. Вызов версии суперкласса необходим, потому что каждый метод жизненного цикла в суперклассе `Activity` содержит код, который должен выполняться помимо кода, определяемого вами в переопределенных методах жизненного цикла.

3.3.3. Построение представления с использованием компонентов `LinearLayout` и `GridLayout`

Макеты (`layouts`) предназначены для построения представлений в графическом интерфейсе. Компонент `LinearLayout` (пакет `android.widget`) размещает компоненты либо горизонтально (по умолчанию), либо вертикально, с возможностью пропорционального изменения размеров своих представлений. Мы воспользуемся им для размещения двух компонентов `TextView` горизонтально, чтобы каждый компонент занимал половину доступного пространства.

Компонент `GridLayout` (пакет `android.widget`) появился в Android 4.0 как новая разновидность макета для размещения компонентов в ячейках прямоугольной таблицы. Ячейки могут занимать сразу несколько строк и столбцов, что позволяет строить достаточно сложные макеты. Во многих случаях `GridLayout` может заменить старый и иногда менее эффективный компонент `TableLayout`, который размещает компоненты по строкам и столбцам; каждая строка обычно определяется объектом `TableRow`, а количество столбцов определяется максимальным количеством ячеек в наборе объектов `TableRow`. Обычно для использования `GridLayout` требуется API уровня 14 и выше. Однако библиотека Android Support Library предоставляет

альтернативные версии `GridLayout` и многих других компонентов графического интерфейса, что позволяет использовать их в старых версиях Android. За дополнительной информацией об этой библиотеке и о том, как использовать ее в приложениях, обращайтесь по адресу

<http://developer.android.com/tools/support-library/index.html>

Компонент `GridLayout` не позволяет указать, что горизонтальное пространство некоторой строки должно пропорционально распределяться между несколькими компонентами. По этой причине в нашем приложении два компонента `TextView` будут объединяться в горизонтальный компонент `LinearLayout`. Это позволит нам разместить два компонента `TextView` в одной ячейке `GridLayout` так, чтобы пространство ячейки равномерно распределялось между ними. Макеты и представления более подробно рассматриваются в дальнейших главах — за полным списком обращайтесь по адресу

<http://developer.android.com/reference/android/widget/package-summary.html>

3.3.4. Создание и настройка графического интерфейса

Мы создадим интерфейс с компонентами `TextView`, `EditText` и `SeekBar` в макетном редакторе IDE (см. главу 2) и окне `Outline`, а затем настроим их в окне свойств IDE — это окно отображается под окном `Outline` при редактировании графического интерфейса в макетном редакторе. Настройка будет осуществляться без прямой правки XML-файлов, хранящихся в папке `res` проекта.

Компонент `EditText` (часто называемый *текстовым полем* в других технологиях) представляет собой субкласс `TextView` (см. главу 2); он предназначен для вывода текста и получения текста, вводимого пользователем. Можно создать специализированную версию `EditText` для цифрового ввода, разрешить пользователям вводить только цифры и ограничить максимальное количество вводимых цифр.

Компонент `SeekBar` представляет целое число (по умолчанию в диапазоне 0–100). Перемещая ползунок, пользователь может выбрать число в диапазоне допустимых значений. Мы настроим `SeekBar` так, чтобы пользователь мог выбрать процент чаевых только из ограниченного диапазона от 0 до 30.

В окне свойств чаще всего настраиваемые свойства компонентов обычно отображаются в начале списка, а их имена выделяются жирным шрифтом (рис. 3.2). Все свойства в окне свойств также делятся на категории. Например, класс `TextView` наследует многие свойства от класса `View`, поэтому в окне свойств присутствует категория `TextView` со свойствами, специфическими для `TextView`, а за ней следует категория `View` со свойствами, унаследованными от класса `View`.

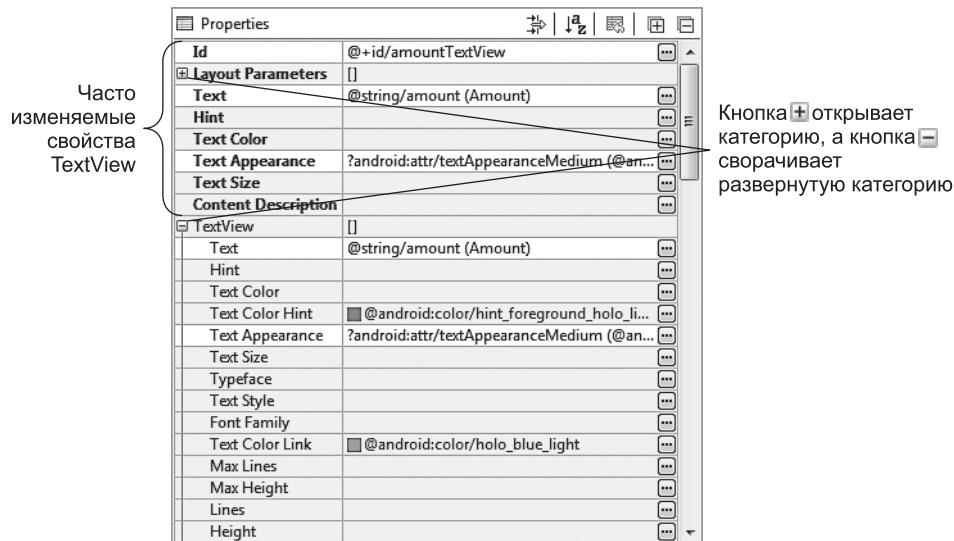


Рис. 3.2. Окно свойств с часто изменяемыми свойствами TextView

3.3.5. Форматирование чисел в соответствии с локальным контекстом

Класс `NumberFormat` (пакет `java.text`) используется для создания строк денежных величин и процентов в формате локального контекста — это важная составляющая интернационализации. Вы также можете добавить строки доступности и интернационализировать приложение средствами, описанными в разделах 2.7–2.8, хотя в нашем приложении это не сделано.

3.3.6. Реализация интерфейса `TextWatcher` для обработки изменений в компоненте `EditText`

Мы воспользуемся анонимным внутренним классом для реализации интерфейса `TextWatcher` (из пакета `android.text`), чтобы обеспечить реакцию на события при изменении текста в поле `EditText` данного приложения. В частности, метод `onTextChanged` будет использоваться для отображения величины счета, отформатированной по правилам денежных сумм, для вычисления чаевых и общей суммы при вводе пользователем каждой цифры.

3.3.7. Реализация интерфейса OnSeekBarChangeListener для обработки изменения позиции ползунка SeekBar

Мы реализуем интерфейс `SeekBar.OnSeekBarChangeListener` (из пакета `android.widget`), чтобы реагировать на перемещения ползунка `SeekBar`. В частности, метод `onProgressChanged` будет использоваться для отображения пользовательской процентной ставки, вычисления чаевых и общей суммы при перемещении ползунка `SeekBar`.

3.3.8. AndroidManifest.xml

Файл `AndroidManifest.xml` генерируется средой разработки при создании нового проекта приложения. В файле хранятся многие настройки, вводимые в диалоговом окне `New Android Application`: имя приложения, имя пакета, целевой и минимальный уровень SDK, имя класса активности, тема оформления и т. д. Мы воспользуемся *редактором манифеста* (`Android Manifest editor`) для того, чтобы добавить в манифест новый параметр, включающий постоянное отображение виртуальной клавиатуры на экране. Также будет указано, что приложение поддерживает только портретную *ориентацию* (то есть вертикальной является более длинная сторона).

3.4. Построение графического интерфейса приложения

В этом разделе будет описан процесс построения графического интерфейса `Tip Calculator`. Только после его завершения графический интерфейс будет выглядеть так, как показано на рис. 3.1. Объем изложенной информации может показаться большим, но многие операции стандартны, и вы быстро привыкнете к ним в процессе использования IDE.

3.4.1. Основы GridLayout

В приложении используется объект `GridLayout` (рис. 3.3) для расположения компонентов GUI в пяти *строках* и двух *столбцах*. Каждая ячейка `GridLayout` либо остается пустой, либо содержит одно или несколько *представлений*, которыми могут быть макеты, *содержащие* другие компоненты. Представление может охватывать несколько строк или столбцов, хотя в данном приложении эта возможность не используется. Количество строк и столбцов в `GridLayout` задается в окне свойств.

Высота каждой строки определяется максимальной высотой представлений в этой строке. Аналогичным образом ширина каждого *столбца* определяется максимальной шириной представлений в этом столбце. По умолчанию представления добавляются

	столбец 0	столбец 1
строка 0	Amount	\$0.00
строка 1	Custom %	<input type="range"/>
строка 2		15% 18%
строка 3	Tip	\$0.00 \$0.00
строка 4	Total	\$0.00 \$0.00

В каждой из этих трех строк второй столбец (т. е. столбец 1) содержит горизонтальный компонент LinearLayout с двумя компонентами TextView

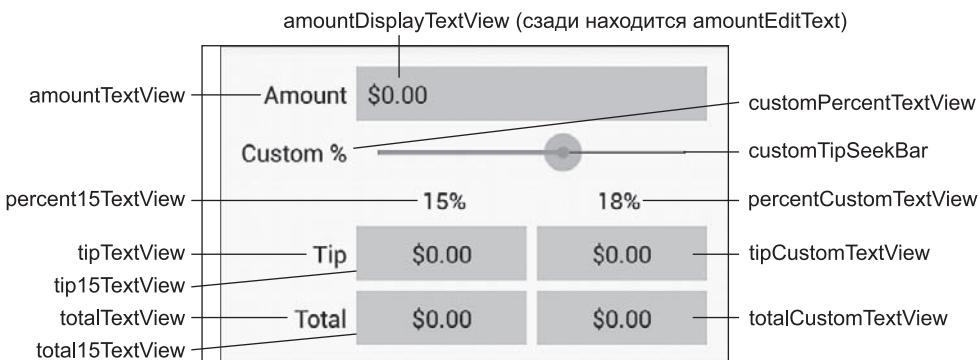
Рис. 3.3. Компонент GridLayout в приложении Tip Calculator

в строку слева направо. Как вы увидите, существует возможность задать строку и столбец для размещения представления. Другие возможности GridLayout будут продемонстрированы при описании действий по построению графического интерфейса. За дополнительной информацией о классе GridLayout обращайтесь по адресу

<http://developer.android.com/reference/android/widget/GridLayout.html>

Значения свойства Id компонентов данного приложения

На рис. 3.4 приведены значения свойств Id компонентов данного приложения. В нашей схеме имя класса компонента используется в свойстве Id компонента и имени переменной Java.

**Рис. 3.4.** Компоненты графического интерфейса Tip Calculator со значениями их свойств Id

В правом столбце первой строки в действительности находятся два компонента — amountDisplayTextView закрывает компонент amountEditText, который получает ввод пользователя. Как будет вскоре показано, пользовательский ввод ограничивается цифрами, чтобы пользователь не мог ввести некорректное значение. Однако при

этом пользователь должен видеть сумму счета в виде денежной величины. При вводе каждой цифры значение делится на 100, а результат в формате денежной суммы отображается в `amountDisplayTextView`. В локальном контексте США при вводе значения 3456 в `amountDisplayTextView` будут последовательно отображаться суммы \$0.03, \$0.34, \$3.45 и \$34.56.

Значения свойств Id для компонентов LinearLayout

На рис. 3.5 приведены значения свойств `Id` трех горизонтальных компонентов `LinearLayout` из правого столбца `GridLayout`.

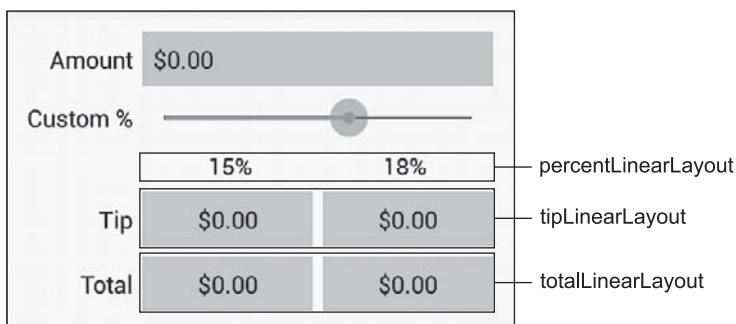


Рис. 3.5. Компоненты `LinearLayout` со значениями их свойств `Id`

3.4.2. Создание проекта TipCalculator

Среда разработки Android Developer Tools позволяет иметь только один проект с конкретным именем в каждой рабочей области, поэтому прежде чем создавать новый проект, удалите проект `TipCalculator`, который тестировался в разделе 3.2. Для этого щелкните на нем правой кнопкой мыши и выберите команду `Delete`. Убедитесь в том, что в открывшемся диалоговом окне флажок `Delete project contents on disk` не установлен, и нажмите `OK`. Проект удаляется из рабочей области, но его папка и файлы остаются на диске на случай, если позднее вы решите снова вернуться к исходному приложению.

Создание нового проекта приложения

Создайте новый проект приложения Android. На первом шаге мастера `New Android Project` введите следующие значения, после чего нажмите `Next >`:

- Application Name: Tip Calculator
- Project Name: TipCalculator
- Package Name: com.deitel.tipcalculator

- Minimum Required SDK: API18: Android 4.3
- Target SDK: API19: Android 4.4
- Compile With: API19: Android 4.4
- Theme: Holo Light with Dark Action Bar
- Create Activity: TipCalculator
- Build Target: проследите за тем, чтобы была выбрана версия Android 4.3

Во втором окне мастера New Android Project оставьте значения по умолчанию и нажмите кнопку **Next>**. На шаге Configure Launcher Icon нажмите **Browse...**, выберите файл **DeitelGreen.png** (из папки **images** в примерах книги) и нажмите **Open**, затем нажмите **Next>**. На шаге **Create Activity** выберите вариант **Blank Activity** (оставьте имя активности по умолчанию) и нажмите кнопку **Next>**. На шаге **Blank Activity** оставьте настройки по умолчанию, после чего нажмите кнопку **Finish** для создания проекта. В макетном редакторе выберите в списке типов экрана **Nexus 4** (см. рис. 2.9) — мы снова возьмем это устройство за основу для построения приложения.

3.4.3. Переключение на GridLayout

По умолчанию в графическом интерфейсе шаблона **Blank App** используется компонент **RelativeLayout**. В нашем примере он будет заменен компонентом **GridLayout**. Щелкните правой кнопкой мыши на компоненте **TextView** в окне **Outline** и выберите команду **Delete**, чтобы удалить компонент из макета. Затем щелкните правой кнопкой мыши на компоненте **RelativeLayout** в окне **Outline** и выберите команду **Change Layout...**. В диалоговом окне **Change Layout** выберите **GridLayout** и нажмите **OK**. Среда разработки изменяет компонент и назначает ему идентификатор **GridLayout1**. Замените его значением **gridLayout** в поле **Id** окна свойств. По умолчанию свойству **Orientation** компонента **GridLayout** задается значение **horizontal**; это означает, что его содержимое будет размещаться по горизонтали (строка за строкой).

Настройка компонента GridLayout с двумя столбцами и отступами по умолчанию

Итак, графический интерфейс на рис. 3.3 состоит из двух столбцов. Чтобы назначить этот параметр, выделите компонент **gridLayout** в окне **Outline**, затем задайте его свойству **Column Count** значение **2** (в группе **GridLayout** окна свойств). По умолчанию ячейки **GridLayout** не имеют отступов (пространство, разделяющее компоненты). Задайте свойству **Use Default Margins** компонента **GridLayout** значение **true**, чтобы вокруг ячеек создавались отступы. По умолчанию **GridLayout** использует рекомендуемые промежутки между представлениями (**8dp**), в соответствии с рекомендациями

<http://developer.android.com/design/style/metrics-grids.html>

3.4.4. Добавление компонентов TextView, EditText, SeekBar и LinearLayout

Перейдем к построению графического интерфейса на рис. 3.3. Мы начнем с создания базового макета и компонентов. В разделе 3.4.5 интерфейс будет завершен настройкой свойств компонентов. При добавлении каждого компонента в интерфейс немедленно задайте его свойство Id в соответствии с рис. 3.4–3.5. Чтобы задать значение Id выделенного компонента, либо воспользуйтесь окном свойств, либо щелкните на компоненте правой кнопкой мыши (в макетном редакторе или окне Outline), выберите команду Edit ID... и отредактируйте свойство Id в открывшемся диалоговом окне Rename Resource.

В следующем описании окно Outline используется для добавления новых компонентов в GridLayout. При работе с макетами бывает трудно увидеть многоуровневую структуру макетов и разместить компоненты в правильных местах, перетаскивая их в окно макетного редактора. Окно Outline упрощает выполнение таких операций, потому что в нем показана иерархия графического интерфейса. Выполняйте следующие действия точно в указанном порядке — в противном случае последовательность отображения компонентов будет нарушена. Если это произойдет, переупорядочите компоненты перетаскиванием их в окне Outline.

Шаг 1: Добавление компонентов в первую строку

Первая строка состоит из компонента amountTextView в первом столбце и компонента amountEditText за компонентом amountDisplayTextView во втором столбце. Каждый раз, когда вы перетаскиваете компонент в gridLayout в окне Outline, он размещается в следующей открытой ячейке макета, если только его позиция не задается явно при помощи свойств Row и Column. Мы воспользуемся этой возможностью, чтобы компоненты amountEditText и amountDisplayTextView разместились в одной ячейке.

Все компоненты TextView в этом приложении используют средний шрифт из темы приложения. Палитра макетного редактора предоставляет готовые варианты TextView с именами Large, Medium и Small (в разделе Form Widgets) для соответствующих размеров текста темы. В каждом случае среда разработки настраивает свойство Text Appearance компонента TextView соответствующим образом.

Выполните следующие действия, чтобы добавить два компонента, TextView и EditText.

1. Перетащите компонент Medium TextView из раздела Form Widgets палитры и разместите его на компоненте gridLayout в окне Outline. Среда разработки создает новый компонент TextView с именем textView1 как вложенный в узел gridLayout. В макетном редакторе выводится текст по умолчанию "Medium Text". Измените значение свойства Id компонента TextView на amountTextView. Текст компонента будет изменен на шаге 6 (раздел 3.4.5).
2. Приложение позволяет вводить только неотрицательные целые числа, которые делятся на 100 для вывода суммы счета. В разделе Text Fields палитры находится

много готовых разновидностей `EditText` для ввода разных данных (числа, время, даты, адреса, телефонные номера). Когда пользователь взаимодействует с `EditText`, на экране отображается клавиатура, соответствующая типу данных `EditText`. Если навести указатель мыши на компонент `EditText` в палитре, на экране появляется подсказка с типом вводимых данных. Найдите в палитре раздел `Text Fields` и перетащите компонент `Number EditText` (обозначенный числом 42) на узел `gridLayout` в окне `Outline`. Задайте свойству `Id` компонента `EditText` значение `amountEditText`. Компонент `EditText` размещается во втором столбце первой строки таблицы `GridLayout`.

3. Перетащите другой компонент `Medium TextView` на узел `gridLayout` в окне `Outline` и измените значение его свойства `Id` на `amountDisplayTextView`. Новый компонент `TextView` изначально размещается в первом столбце второй строки `GridLayout`. Чтобы поместить его во второй столбец первой строки `GridLayout`, задайте свойствам `Row` и `Column` компонента `TextView` (из раздела `Layout Parameters` окна свойств) значения 0 и 1 соответственно.

Шаг 2: Добавление компонентов во вторую строку

Затем в `GridLayout` добавляются компоненты `TextView` и `SeekBar`. Это делается так:

1. Перетащите компонент `Medium TextView (customPercentTextView)` из раздела `Form Widgets` палитры на узел `gridLayout` в окне `Outline`.
2. Перетащите компонент `SeekBar (customTipSeekBar)` из раздела `Form Widgets` палитры на узел `gridLayout` в окне `Outline`.

Шаг 3: Добавление компонентов в третью строку

На следующем шаге в `GridLayout` добавляется макет `LinearLayout` с двумя компонентами `TextView`.

1. Перетащите компонент `Linear Layout (Horizontal) (percentLinearLayout)` на узел `gridLayout` в окне `Outline`.
2. Перетащите компонент `Medium TextView (percent15TextView)` на узел `percentLinearLayout` в окне `Outline`. В результате новый компонент `TextView` будетложен в `LinearLayout`.
3. Перетащите другой компонент `Medium TextView (percentCustomTextView)` на узел `percentLinearLayout` в окне `Outline`.
4. Компонент `percentLinearLayout` и два его вложенных компонента `TextView` должны находиться во втором столбце `GridLayout`. Чтобы разместить их в нужном месте, выделите компонент `percentLinearLayout` в окне `Outline` и задайте его свойству `Column` значение 1.

Шаг 4: Добавление компонентов в четвертую строку

На следующем шаге в `GridLayout` добавляется компонент `TextView` и `LinearLayout`, содержащий еще два компонента `TextView`:

- Перетащите компонент Medium TextView (`tipTextView`) на узел `gridLayout` в окне Outline.
- Перетащите компонент Linear Layout (Horizontal) (`tipLinearLayout`) на узел `gridLayout`.
- Перетащите два компонента Medium TextView (`tip15TextView` и `tipCustomTextView`) на узел `tipLinearLayout`.

Шаг 5: Добавление компонентов в пятую строку

Чтобы создать последнюю строку графического интерфейса, повторите шаг 4, используя следующие значения Id: `totalTextView`, `totalLinearLayout`, `total15TextView` и `totalCustomTextView`.

Текущее состояние приложения

Примерный вид графического интерфейса и окна Outline показан на рис. 3.6. Предупреждающие значки в макетном редакторе и окне Outline исчезнут после завершения макета в разделе 3.4.5.

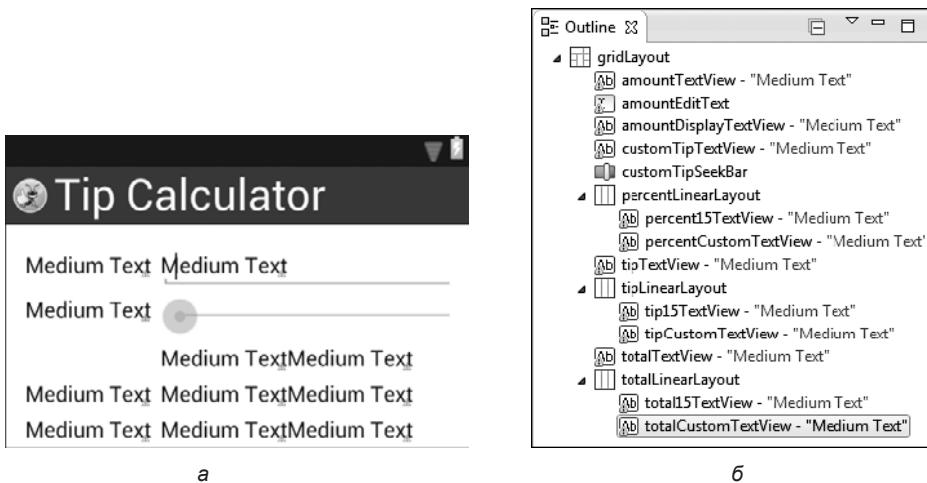


Рис. 3.6. Графический интерфейс и окно Outline после добавления компонентов в GridLayout: а — текущее состояние графического интерфейса; б — окно Outline с компонентами Tip Calculator

3.4.5. Настройка компонентов

Чтобы завершить построение приложения, мы настроим свойства компонентов, создадим несколько ресурсов для строк и метрик. Как упоминалось в разделе 2.5, строковые литералы следует размещать в файлах ресурсов `strings.xml`. Аналогичным образом числовые литералы, задающие метрики компонентов (ширина, высота, интервалы и т. д.), рекомендуется размещать в файле ресурсов `dimens.xml`.

Шаг 6: Ввод текстовых литералов

Задайте текстовые литералы для компонентов `amountTextView`, `customPercentTextView`, `percent15TextView`, `percentCustomTextView`, `tipTextView` и `totalTextView`:

1. Выделите компонент `amountTextView` в окне Outline.
2. В окне свойств щелкните на кнопке с многоточием рядом со свойством `Text`.
3. В диалоговом окне Resource Chooser нажмите New String....
4. В диалоговом окне Create New Android String введите текст `Amount` в поле String и текст `amount` в поле New R.string, нажмите OK.
5. В диалоговом окне Resource Chooser нажмите OK, чтобы задать свойству `Text` компонента `amountTextView` строку ресурса с идентификатором `amount`.

Повторите описанные операции для всех остальных компонентов `TextView`, используя значения из табл. 3.1.

Таблица 3.1. Значения и идентификаторы строковых ресурсов

Компонент	String	New R.String
<code>customPercentTextView</code>	Custom %	<code>custom_tip_percentage</code>
<code>percent15TextView</code>	15%	<code>fifteen_percent</code>
<code>percentCustomTextView</code>	18%	<code>eighteen_percent</code>
<code>tipTextView</code>	Tip	<code>tip</code>
<code>totalTextView</code>	Total	<code>total</code>

Шаг 7: Выравнивание компонентов `TextView` по правому краю левого столбца

На рис. 3.3 все компоненты `TextView` в левом столбце выровнены по правому краю. Для компонентов `amountTextView`, `customPercentTextView`, `tipTextView` и `totalTextView` задайте *макетному* свойству `Gravity` (раздел `Layout Parameters` окна свойств) значение `right`.

Шаг 8: Настройка свойства `Label For` компонента `amountTextView`

Как правило, каждый компонент `EditText` снабжается компонентом `TextView` с описанием, которое помогает пользователю понять смысл `EditText` (а также повышает доступность приложения) — в противном случае *Android Lint* выдает предупреждение. Чтобы исправить недостаток, задайте идентификатор `Id` связанного компонента `EditText` свойству `Label For` компонента `TextView`. Выделите компонент `amountTextView` и задайте его свойству `Label For` (из раздела `View` окна свойств) значение `@+id/amountEditText`.

Знак `+` необходим, потому что компонент `TextView` в графическом интерфейсе определяется *до* `EditText`, так что `EditText` еще не существует при преобразовании XML-разметки в графический интерфейс.

Шаг 9: Настройка amountEditText

В приложении компонент `amountEditText` скрывается за `amountDisplayTextView` и настраивается так, чтобы пользователь мог вводить только цифры. Выделите `amountEditText` и задайте следующие свойства.

1. В разделе `Layout Parameters` окна свойств задайте свойствам `Width` и `Height` значение `wrap_content`. Это означает, что размеры компонента `EditText` выбираются по размерам содержимого, включая все отступы.
2. Удалите значение макетного свойства `Gravity` (`fill_horizontal`), оставив значение свойства пустым. Значение `fill_horizontal` описано ниже.
3. Удалите значение свойства `Ems`, которое обозначает ширину `EditText` в единицах, соответствующих ширине буквы М (в верхнем регистре) шрифта компонента. В нашем компоненте `GridLayout` второй столбец получается слишком узким, поэтому мы удалили значение по умолчанию.
4. В разделе `TextView` окна свойств задайте свойству `Digits` значение `0123456789` — оно разрешает вводить *только* цифры, хотя цифровая клавиатура также содержит кнопки для ввода минуса, запятой, точки и пробела. По умолчанию, свойство `Digits` не отображается в окне свойств, потому что оно относится к категории «расширенных» свойств. Чтобы вывести его в окне, установите переключатель `Show Advanced Properties` () в верхней части окна свойств.
5. Максимальная сумма счета ограничивается шестью цифрами, так что наибольшая сумма, поддерживаемая в приложении, равна 9999,99. В разделе `TextView` окна свойств задайте свойству `Max Length` значение 6.

Шаг 10: Настройка amountDisplayTextView

Чтобы завершить форматирование компонента `amountDisplayTextView`, выберите его и задайте следующие свойства.

1. В разделе `Layout Parameters` окна свойств задайте свойствам `Width` и `Height` значение `wrap_content`. Это означает, что размеры компонента `TextView` выбираются по размерам содержимого.
2. Удалите значение свойства `Text` — выводимый текст будет определяться на программном уровне.
3. В разделе `Layout Parameters` окна свойств задайте свойству `Gravity` значение `fill_horizontal`. Это означает, что компонент `TextView` должен занимать все оставшееся горизонтальное место в строке `GridLayout`.
4. В разделе `View` задайте свойству `Background` значение `@android:color/holo_blue_bright`. Это один из нескольких заранее определенных цветов (их имена начинаются с `@android:color`) в теме Android Holo. Когда вы начинаете вводить значение свойства `Background`, на экране появляется список доступных цветов темы. Вы также можете использовать любой цвет, определяемый комбинацией красной, зеленой и синей составляющей — так называемых *значений RGB*. Каждая

составляющая задается целым числом в диапазоне 0–255. Пользовательские цвета определяются в шестнадцатеричном формате (по основанию 16), так что составляющие RGB лежат в диапазоне 00–FF. Android также поддерживает составляющую альфа-канала (уровня прозрачности) в диапазоне от 0 (полная прозрачность) до 255 (полная непрозрачность). Чтобы использовать альфа-канал, задайте цвет в формате #AARRGGBB, где первые две шестнадцатеричные цифры (AA) представляют альфа-канал. Если обе цифры каждой цветовой составляющей совпадают, допускается использование сокращенных форматов #RGB или #ARGB. Например, запись #9AC интерпретируется как #99AACC, а #F9AC — как #FF99AACC.

5. Остается добавить отступы вокруг `TextView`. Для этого мы создадим новый ресурс метрики с именем `textview_padding`, который будет использоваться в разных местах GUI. Свойство `Padding` компонента задает величину отступов со всех сторон содержимого компонента. В разделе `View` окна свойств щелкните на кнопке с многоточием рядом со свойством `Padding`. Нажмите `New Dimension...`, чтобы создать новый ресурс метрики. Введите в поле `Name` текст `textview_padding`, в поле `Value` — значение `8dp`, и нажмите `OK`. Выделите новый ресурс и нажмите `OK`.

Шаг 11: Настройка компонента `customPercentTextView`

Обратите внимание: компонент `customPercentTextView` выравнивается по верхнему краю ползунка `customTipSeekBar`. Макет будет смотреться лучше, если выровнять компонент вертикально по центру. Для этого в разделе `Layout Parameters` окна свойств измените значение `Gravity` с `right` на `right|center_vertical`.

Вертикальная черта (|) используется для разделения *нескольких* значений `Gravity` — в данном случае этот символ означает, что компонент `TextView` должен выравниваться по правому краю, а по вертикали — по центру. Также задайте свойствам `Width` и `Height` компонента `customPercentTextView` значение `wrap_content`.

Шаг 12: Настройка компонента `customTipSeekBar`

По умолчанию, компонент `SeekBar` использует диапазон допустимых значений от 0 до 100, а его текущее значение обозначается свойством `Progress`. Приложение поддерживает проценты по шкале от 0 до 30, а значение по умолчанию равно 18. Задайте свойству `Max` компонента `SeekBar` значение 30, а свойству `Progress` — значение 18. Также задайте свойствам `Width` и `Height` значение `wrap_content`.

Шаг 13: Настройка компонентов `percent15TextView` и `percentCustomTextView`

Вспомните, что компонент `GridLayout` не позволяет указать размеры представления относительно других представлений строки. Именно поэтому мы поместили компоненты `percent15TextView` и `percentCustomTextView` в контейнер `LinearLayout`, поддерживающий пропорциональное определение размеров. Свойство `Weight` компонента (в некоторых макетах — таких, как `LinearLayout`) задает его вес (относительную величину) по сравнению с другими компонентами в макете. По умолчанию, все компоненты имеют нулевое значение `Weight`.

В нашем макете мы зададим значение `Weight`, равное 1, для `percent15TextView` и `percentCustomTextView` — это означает, что они обладают одинаковыми весами, а следовательно, должны иметь равные размеры. По умолчанию, при добавлении компонента `percentLinearLayout` в `GridLayout` его *макетному* свойству `Gravity` было задано значение `fill_horizontal`, так что макет заполняет все оставшееся место в третьей строке. Когда `LinearLayout` растягивается для заполнения остатка строки, каждый из компонентов `TextView` занимает *половину* ширины `LinearLayout`.

Мы также хотим, чтобы текст каждого компонента `TextView` выравнивался по центру. Для этого свойству `Gravity` (раздел `TextView` окна свойств) задается значение `center`. Тем самым мы определяем способ выравнивания текста в `TextView`, тогда как *макетное* свойство `Gravity` определяет способ выравнивания компонента относительно макета.

Шаг 14: Настройка компонентов `tip15TextView`, `tipCustomTextView`, `total15TextView` и `totalCustomTextView`

В завершение настройки четырех компонентов `TextView` выполните с каждым из них следующие операции.

1. Выделите компонент `TextView`.
2. Удалите значение свойства `Text` — выводимый текст будет определяться на программном уровне.
3. Задайте свойству `Background` значение `@android:color/holo_orange_light`.
4. Задайте макетному свойству `Gravity` значение `center`.
5. Задайте макетному свойству `Weight` значение 1.
6. Задайте макетному свойству `Width` значение `0dp` — тем самым вы разрешаете макету использовать свойство `Weight` для определения ширины представления.
7. Задайте свойству `Gravity` компонента `TextView` значение `center`.
8. Задайте свойству `Padding` компонента `TextView` значение `@dimen/textview_padding` (ресурс, созданный на предыдущем шаге).

Обратите внимание на отсутствие горизонтального промежутка между компонентами `TextView` в `tipLinearLayout` и `totalLinearLayout`. Чтобы исправить этот недостаток, мы зададим правый отступ величиной `8dp` для `tip15TextView` и `total15TextView`. В разделе `Layout Parameters` окна свойств откройте раздел `Margin` и задайте свойству `Right` значение `8dp`; для этого создайте новый ресурс метрики с именем `textview_margin`. Затем используйте этот ресурс для задания свойства `Right` компонента `total15TextView`.

Шаг 15: Вертикальное выравнивание по центру компонентов `tipTextView` и `totalTextView`

Чтобы обеспечить вертикальное выравнивание по центру компонентов `tipTextView` и `totalTextView` с другими представлениями в их строках, измените макетное свойство `Gravity` с `right` на `right|center_vertical`.

Когда вы проделаете это для компонента `totalTextView`, компонент `GridLayout` выравнивает его по вертикали в оставшемся пространстве от пятой строки до низа экрана. Для решения этой проблемы перетащите компонент `Space` (из раздела `Layout` палитры) на узел `gridLayout` в окне `Outline`. При этом создается шестая строка, занимающая остаток экрана. Как нетрудно догадаться по имени, компонент `Space` занимает пространство в графическом интерфейсе. Графический интерфейс должен выглядеть так, как показано на рис. 3.7.

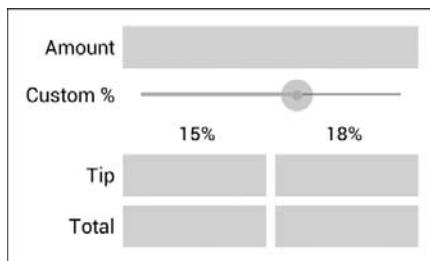


Рис. 3.7. Окончательный вид графического интерфейса

3.5. Включение функциональности в приложение

Класс `MainActivity` (листинги 3.1–3.8) реализует функциональность приложения Tip Calculator. Он вычисляет чаевые из расчета 15% и пользовательской ставки, вычисляет общую сумму и выводит результаты в формате денежной суммы для действующего локального контекста. Чтобы просмотреть файл, откройте папку `src/com.deitel/tipcalculator` и сделайте двойной щелчок на файле `MainActivity.java`. Вам придется ввести большую часть кода в листингах 3.1–3.8.

Команды `package` и `import`

В листинге 3.1 приведены команды `package` и `import` из файла `MainActivity.java`. Команда `package` в строке 3 была вставлена при создании проекта. Когда файл Java открывается в среде разработки, команды `import` сворачиваются — отображается только одна команда со значком слева. Щелкнув на , вы сможете просмотреть список команд `import`.

Листинг 3.1. Команды `package` и `import` из файла `MainActivity`

```

1 // MainActivity.java
2 // вычисление чаевых из расчета 15% и пользовательской ставки.
3 package com.deitel.tipcalculator;
4
5 import java.text.NumberFormat; // Для форматирования денежных сумм
6
7 import android.app.Activity; // Базовый класс активностей
8 import android.os.Bundle; // Для сохранения информации состояния
9 import android.text.Editable; // Для обработки событий EditText
10 import android.text.TextWatcher; // Слушатель EditText

```

```
11 import android.widget.EditText; // Для ввода суммы счета
12 import android.widgetSeekBar; // Для изменения пользовательского %
13 import android.widgetSeekBar.OnSeekBarChangeListener; // СлушательSeekBar
14 import android.widget.TextView; // Для вывода текста
15
```

В строках 5–14 импортируются классы и интерфейсы, используемые приложением:

- ❑ Класс `NumberFormat` из пакета `java.text` (строка 5) предоставляет средства числового форматирования (например, форматы денежных сумм и процентов по правилам локального контекста).
- ❑ Класс `Activity` из пакета `android.app` (строка 7) предоставляет базовые методы жизненного цикла приложения — вскоре мы рассмотрим их подробнее.
- ❑ Класс `Bundle` из пакета `android.os` (строка 8) представляет информацию состояния приложения. Android дает приложению возможность сохранить свое состояние перед тем, как другое приложение появится на экране (например, когда пользователь запустит другое приложение или примет телефонный звонок). Приложение, находящееся на экране, работает в активном режиме (пользователь может взаимодействовать с ним, приложение потребляет вычислительные ресурсы), а другие приложения работают в фоновом режиме (пользователь не может взаимодействовать с ними, и обычно они не занимают процессор). Когда другое приложение переходит в активный режим, приложению, которое ранее было активным, предоставляется возможность сохранить свое состояние перед переходом в фоновой режим.
- ❑ Интерфейс `Editable` из пакета `android.text` (строка 9) позволяет изменять содержимое и разметку текста в графическом интерфейсе.
- ❑ Интерфейс `TextWatcher` из пакета `android.text` (строка 10) реализуется для обработки событий при изменении пользователем текста в `EditText`.
- ❑ Пакет `android.widget` (строки 11–14) содержит виджеты (то есть визуальные компоненты) и макеты, используемые в графическом интерфейсе Android. В приложении используются виджеты `EditText` (строка 11), `SeekBar` (строка 12) и `TextView` (строка 14).
- ❑ Интерфейс `SeekBar.OnSeekBarChangeListener` из пакета `android.widget` (строка 13) реализуется для обработки событий, возникающих при перемещении ползунка `SeekBar`.

При написании кода с различными классами и интерфейсами можно воспользоваться командой `Source > Organize Imports` среды разработки, чтобы она могла вставить команды `import` за вас. Если имя класса или интерфейса встречается в нескольких пакетах, среда разработки позволяет выбрать нужную команду `import`.

Активность приложения Tip Calculator и жизненный цикл активности

Класс `MainActivity` (листинги 3.2–3.8) является субклассом `Activity` для приложения Tip Calculator. При создании проекта Tip Calculator среда разработки генерировала

этот класс как субкласс `Activity` и предоставила переопределенную версию унаследованного от `Activity` метода `onCreate` (листинг 3.3). Каждый субкласс `Activity` должен переопределять этот метод. Код класса `MainActivity` по умолчанию также включает метод `onCreateOptionsMenu`, который мы удалили, потому что он не используется в приложении. Метод `onCreate` будет рассмотрен далее.

Листинг 3.2. Класс `MainActivity` является субклассом `Activity`

```
16 // Класс MainActivity приложения Tip Calculator
17 public class MainActivity extends Activity
18 {
```

Переменные класса

В строках 20–32 листинга 3.3 объявляются переменные класса `MainActivity`. Объекты `NumberFormat` (строки 20–23) используются для форматирования денежных сумм и процентов соответственно. Статический метод `getCurrencyInstance` класса `NumberFormat` возвращает объект `NumberFormat`, который форматирует значения как денежные суммы с использованием *локального контекста по умолчанию*, назначенного для устройства. Аналогичный статический метод `getPercentInstance` форматирует значения как проценты с использованием локального контекста по умолчанию.

Листинг 3.3. Переменные экземпляров класса `MainActivity`

```
19     // Форматировщики денежных сумм и процентов
20     private static final NumberFormat currencyFormat =
21         NumberFormat.getCurrencyInstance();
22     private static final NumberFormat percentFormat =
23         NumberFormat.getPercentInstance();
24
25     private double billAmount = 0.0; // Величина счета, введенная пользователем
26     private double customPercent = 0.18; // Исходный пользовательский %
27     private TextView amountDisplayTextView; // Для отформатированной суммы счета
28     private TextView percentCustomTextView; // Для пользовательского % чаевых
29     private TextView tip15TextView; // Для вывода 15% чаевых
30     private TextView total15TextView; // Для вывода суммы с 15% чаевых
31     private TextView tipCustomTextView; // Для вывода пользовательских чаевых
32     private TextView totalCustomTextView; // Для вывода суммы
                                         // с пользовательскими чаевыми
33
```

Величина счета, введенная пользователем в поле `amountEditText`, считывается и хранится в формате `double` в переменной `billAmount` (строка 25). Пользовательский процент чаевых (целое число в диапазоне 0–30), введенный посредством перемещения ползунка `SeekBar`, умножается на 0,01; полученное значение `double` используется в вычислениях и сохраняется в переменной `customPercent` (строка 26). Например, если выбрать с помощью компонента `SeekBar` значение 25, то в переменной `customPercent` будет сохранено значение 0,25, а приложение будет умножать величину счета на 0,25 для вычисления 25% чаевых.

В строке 27 объявляется компонент `TextView` для отображения выставленного счета, отформатированного в виде денежной суммы. В строке 28 объявляется компонент `TextView` для вывода пользовательского процента чаевых, полученного на основании положения ползунка `SeekBar` (18% на рис. 3.1, а). Переменные в строках 29–32 ссылаются на компоненты `TextView`, в которых приложение выводит вычисленные чаевые и сумму счета.

Переопределение метода `onCreate` класса `Activity`

Метод `onCreate` (см. листинг 3.4) генерируется автоматически при создании проекта приложения и вызывается системой после запуска класса активности. Метод `onCreate` обычно инициализирует переменные экземпляра `Activity` и компоненты GUI. Этот метод должен быть предельно упрощен, чтобы приложение загружалось быстро. Фактически в случае, если загрузка приложения занимает более пяти секунд, операционная система отображает диалоговое окно ANR (Application Not Responding, приложение не отвечает). В этом окне пользователю предоставляется возможность принудительно завершить приложение. О том, как устранить эту проблему, рассказано в главе 8.

Листинг 3.4. Переопределение метода `onCreate` класса `Activity`

```
34 // Вызывается при первом создании активности
35 @Override
36 protected void onCreate(Bundle savedInstanceState)
37 {
38     super.onCreate(savedInstanceState); // Вызов версии суперкласса
39     setContentView(R.layout.activity_main); // Заполнение GUI
40
41     // Получение ссылок на компоненты TextView, с которыми
42     // MainActivity взаимодействует на программном уровне
43     amountDisplayTextView =
44         (TextView) findViewById(R.id.amountDisplayTextView);
45     percentCustomTextView =
46         (TextView) findViewById(R.id.percentCustomTextView);
47     tip15TextView = (TextView) findViewById(R.id.tip15TextView);
48     total15TextView = (TextView) findViewById(R.id.total15TextView);
49     tipCustomTextView = (TextView) findViewById(R.id.tipCustomTextView);
50     totalCustomTextView =
51         (TextView) findViewById(R.id.totalCustomTextView);
52
53     // Обновление GUI по данным billAmount и customPercent
54     amountDisplayTextView.setText(
55         currencyFormat.format(billAmount));
56     updateStandard(); // Обновление TextView с 15%-ными чаевыми
57     updateCustom(); // Обновление TextView с пользовательскими чаевыми
58
59     // Назначение TextWatcher для amountEditText
60     EditText amountEditText =
61         (EditText) findViewById(R.id.amountEditText);
62     amountEditText.addTextChangedListener(amountEditTextWatcher);
63
64     // Назначение OnSeekBarChangeListener для customTipSeekBar
```

```
65     SeekBar customTipSeekBar =
66         (SeekBar) findViewById(R.id.customTipSeekBar);
67     customTipSeekBar.setOnSeekBarChangeListener(customSeekBarListener);
68 } // Конец метода onCreate
69
```

Параметр **Bundle** метода **onCreate**

Во время выполнения пользователь может изменить конфигурацию устройства, повернув его или выдвинув аппаратную клавиатуру. Качественное приложение должно легко справляться с такими изменениями конфигурации. При вызове системой метода **onCreate** передается аргумент **Bundle** с сохраненным состоянием **Activity** (если оно имеется). Как правило, состояние сохраняется в методах **onPause** или **onSaveInstanceState** активности (эта возможность продемонстрирована в последующих приложениях). Стока 38 вызывает метод **onCreate** суперкласса; этот вызов обязателен при переопределении **onCreate**.

Сгенерированный класс **R** содержит идентификаторы ресурсов

При построении графического интерфейса приложения и добавлении в приложение ресурсов (например, строк в файле **strings.xml** или компонентов в файле **activity_main.xml**) среда разработки генерирует класс с именем **R**, который содержит вложенные классы, представляющие каждый тип ресурсов из папки **res** вашего проекта. Класс **R** находится в папке **gen** вашего проекта, содержащей сгенерированные файлы с исходным кодом. Вложенные классы объявляются статическими, так что вы можете обращаться к ним в коде с использованием синтаксиса **R.ИмяКласса**. файлы исходного кода. В классах, вложенных в класс **R**, создаются константы **static final int**, с помощью которых можно обращаться к ресурсам из кода приложения (вскоре вы увидите, как это делается). Некоторые классы, вложенные в класс **R**:

- ❑ Класс **drawable** — содержит константы всех элементов **drawable** (например, изображений), которые находятся в различных папках **drawable** в папке приложения **res**.
- ❑ Класс **id** — содержит константы для *представлений* (визуальных компонентов), используемые в *файлах XML-разметки*.
- ❑ Класс **layout** — содержит константы, которые представляют *макетные файлы* проекта (например, **activity_main.xml**).
- ❑ Класс **string** — константы, используемые для определения строк в файле **strings.xml**.

Заполнение графического интерфейса

При вызове **setContentView** (строка 39) передается константа **R.layout.activity_main**, определяющая XML-файл с графическим интерфейсом **MainActivity**, — в данном случае она определяет файл **main.xml**. Метод **setContentView** использует эту константу для загрузки соответствующего документа XML, который разбирается и преобразуется в графический интерфейс приложения. Этот процесс называется *заполнением* (*inflating*) графического интерфейса.

Получение ссылок на виджеты

После того как заполнение графического интерфейса будет завершено, вы сможете получать ссылки на конкретные виджеты для выполнения программных операций с ними. Для этого используется метод `findViewById` класса `Activity`. Метод получает константу `int`, представляющую идентификатор конкретного виджета, и возвращает ссылку на него. Имя константы `R.id` каждого виджета определяется свойством `Id`, заданным при проектировании графического интерфейса. Например, компоненту `amountEditText` соответствует константа `R.id.amountEditText`.

Строки 43–51 получают ссылки на компоненты `TextView`, изменяемые приложением. Строки 43–44 получают ссылку на компонент `amountDisplayTextView`, обновляемый при вводе выставленного счета. Строки 45–46 получают ссылку на компонент `percentCustomTextView`, который обновляется при изменении пользовательского процента чаевых. В строках 47–51 получаются ссылки на компоненты `TextView`, в которых отображаются вычисленные чаевые и итоговая сумма.

Вывод исходных значений в `TextView`

В строках 54–55 компонент `amountDisplayTextView` заполняется исходным значением `billAmount` (0.00), отформатированным по правилам денежных сумм для текущего локального контекста (для этого вызывается метод `format` объекта `currencyFormat`). Затем в строках 56–57 вызываются методы `updateStandard` (листинг 3.5) и `updateCustom` (листинг 3.6) для вывода исходных значений чаевых и общей суммы в компонентах `TextView`.

Регистрация слушателей событий

В строках 60–61 мы получаем ссылку на компонент `amountEditText`, а в строке 62 вызывается его метод `addTextChangedListener` для регистрации слушателя `TextChangeListener`, который будет реагировать на события, генерируемые при изменении текста в `EditText`. В нашей программе этот слушатель (листинг 3.8) определяется как объект анонимного внутреннего класса, который присваивается переменной экземпляра `amountEditTextWatcher`.

В строках 65–66 мы получаем ссылку на компонент `customTipSeekBar`, а в строке 67 вызывается его метод `setOnSeekBarChangeListener` для регистрации слушателя `OnSeekBarChangeListener`, который будет реагировать на события, генерируемые при перемещении ползунка `customTipSeekBar` для изменения пользовательского процента чаевых. Слушатель (листинг 3.7) определяется как объект анонимного внутреннего класса, который присваивается переменной экземпляра `customSeekBarListener`.

Метод `updateStandard` класса `MainActivity`

Метод `updateStandard` (листинг 3.5) обновляет компоненты `TextView` с величиной 15% чаевых и общей суммой счета каждый раз, когда пользователь изменяет выставленный счет. Метод использует значение `billAmount` для вычисления размера чаевых и суммы счета с чаевыми. В строках 78–79 значения выводятся в формате денежной суммы.

Листинг 3.5. Метод updateStandard вычисляет и выводит 15% чаевых и общую сумму счета

```

70 // Обновление компонентов TextView для 15% чаевых
71 private void updateStandard()
72 {
73     // Вычисление 15% чаевых и общей суммы
74     double fifteenPercentTip = billAmount * 0.15;
75     double fifteenPercentTotal = billAmount + fifteenPercentTip;
76
77     // Вывод 15% чаевых и общей суммы в формате денежной величины
78     tip15TextView.setText(currencyFormat.format(fifteenPercentTip));
79     total15TextView.setText(currencyFormat.format(fifteenPercentTotal));
80 } // Конец метода updateStandard
81

```

Метод updateCustom класса MainActivity

Метод updateCustom (листинг 3.6) обновляет компоненты TextView с пользовательскими чаевыми, выбранными пользователем при помощи компонента customSeekBar, и общей суммой счета с учетом пользовательских чаевых. В строке 86 тексту percentCustomTextView назначается значение customPercent, отформатированное в процентах. В строках 89–90 вычисляются значения переменных customTip и customTotal, после чего в строках 93–94 значения выводятся в формате денежной суммы.

Листинг 3.6. Метод updateCustom вычисляет и выводит пользовательские чаевые и общую сумму счета

```

82 // Обновляет компоненты TextView пользовательских чаевых и общей суммы
83 private void updateCustom()
84 {
85     // Вывод в percentCustomTextView отформатированного значения
86     // customPercent %
87     percentCustomTextView.setText(percentFormat.format(customPercent));
88
89     // Вычисление пользовательских чаевых и общей суммы
90     double customTip = billAmount * customPercent;
91     double customTotal = billAmount + customTip;
92
93     // Вывод пользовательских чаевых и суммы в денежном формате
94     tipCustomTextView.setText(currencyFormat.format(customTip));
95     totalCustomTextView.setText(currencyFormat.format(customTotal));
96 } // Конец метода updateCustom

```

Анонимный внутренний класс, реализующий интерфейс OnSeekBarChangeListener

В строках 98–120 листинга 3.7 создается объект анонимного внутреннего класса customSeekBarListener, реагирующий на события customSeekBar. Если вы еще не знакомы с анонимными внутренними классами, посетите следующую страницу:

<http://bit.ly/AnonymousInnerClasses>

В строке 67 (листинг 3.4) `customSeekBarListener` регистрируется как объект обработки события `OnSeekBarChangeListener` для компонента `customTipSeekBar`. Для наглядности мы определяем так все объекты обработки событий, кроме самых простых, чтобы не загромождать метод `onCreate` этим кодом.

Листинг 3.7. Анонимный внутренний класс, реализующий интерфейс `OnSeekBarChangeListener` для обработки событий `customSeekBar`

```

97     // Вызывается при изменении позиции SeekBar
98     private OnSeekBarChangeListener customSeekBarListener =
99         new OnSeekBarChangeListener()
100        {
101            // Обновить customPercent, затем вызвать updateCustom
102            @Override
103            public void onProgressChanged(SeekBar seekBar, int progress,
104                boolean fromUser)
105            {
106                // Переменной customPercent присваивается позиция ползунка SeekBar
107                customPercent = progress / 100.0;
108                updateCustom(); // Обновление TextView с пользовательскими чаевыми
109            } // Конец метода onProgressChanged
110
111            @Override
112            public void onStartTrackingTouch(SeekBar seekBar)
113            {
114            } // Конец метода onStartTrackingTouch
115
116            @Override
117            public void onStopTrackingTouch(SeekBar seekBar)
118            {
119            } // Конец метода onStopTrackingTouch
120        }; // Конец OnSeekBarChangeListener
121

```

Переопределение метода `onProgressChanged` интерфейса `OnSeekBarChangeListener`

В строках 102–119 реализуются методы интерфейса `OnSeekBarChangeListener`. Метод `onProgressChanged` вызывается при каждом изменении позиции ползунка `SeekBar`. Стока 107 вычисляет `customPercent` с использованием параметра `progress` метода — значения типа `int`, представляющего позицию ползунка `SeekBar`. Пользовательский процент чаевых вычисляется делением значения на 100. В строке 108 вызывается метод `updateCustom` для пересчета и отображения пользовательских чаевых и общей суммы.

Переопределение методов `onStartTrackingTouch` и `onStopTrackingTouch` интерфейса `OnSeekBarChangeListener`

В языке Java каждый метод реализуемого интерфейса должен переопределяться в программе. Так как нашему приложению не нужно знать, когда пользователь начинает перемещать ползунок (`onStartTrackingTouch`) или прекращает перемещать его (`onStopTrackingTouch`), мы просто предоставляем пустое тело для каждого метода (строки 111–119), чтобы выполнить контракт интерфейса.

Анонимный внутренний класс, реализующий интерфейс TextWatcher

В строках 123–156 листинга 3.8 создается объект анонимного внутреннего класса `amountEditTextWatcher`, реагирующий на события `amountEditText`. В строке 62 этот объект регистрируется для прослушивания событий `amountEditText`, происходящих при изменении текста.

Листинг 3.8. Анонимный внутренний класс, реагирующий на события amountEditText

```
122     // Объект, реагирующий на события amountEditText
123     private TextWatcher amountEditTextWatcher = new TextWatcher()
124     {
125         // Вызывается, если пользователь ввел число
126         @Override
127         public void onTextChanged(CharSequence s, int start,
128             int before, int count)
129         {
130             // Преобразование amountEditText в тип double
131             try
132             {
133                 billAmount = Double.parseDouble(s.toString()) / 100.0;
134             } // Завершение try
135             catch (NumberFormatException e)
136             {
137                 billAmount = 0.0; // Для обработки исключений
138             } // Завершение catch
139
140             // display currency formatted bill amount
141             amountDisplayTextView.setText(currencyFormat.format(billAmount));
142             updateStandard(); // Обновление чаевых
143             updateCustom(); // обновление общей суммы
144         } // Завершение метода onTextChanged
145
146         @Override
147         public void afterTextChanged(Editable s)
148         {
149             } // Завершение метода afterTextChanged
150
151         @Override
152         public void beforeTextChanged(CharSequence s, int start, int count,
153             int after)
154         {
155             } // Завершение метода beforeTextChanged
156     }; // Завершение amountEditTextWatcher
157 } // Завершение класса MainActivity
```

Переопределение метода onTextChanged интерфейса TextWatcher

Метод `onTextChanged` (строки 126–144) вызывается при каждом изменении текста, отображаемого в компоненте `amountEditText`. Метод получает четыре параметра. В рассматриваемом примере используется только параметр `CharSequence s`, который содержит копию текста `amountEditText`. Другие параметры сообщают о том, что

текст, количество символов которого определено параметром `count`, начинающийся в позиции `start`, заменил фрагмент прежнего текста длины `before`.

В строке 133 текст, введенный пользователем в компоненте `amountEditText`, преобразуется в тип `double`. Поддерживается ввод только целых чисел, поэтому для получения фактического выставленного счета преобразованное значение делится на 100 – например, если пользователь ввел 2495, то величина счета составит 24,95. Строки 142–143 вызывают методы `updateStandard` и `updateCustom` для пересчета и отображения чаевых и общей суммы.

Другие методы `amountEditTextWatcher`

Нашему приложению не нужно знать, когда начинается внесение изменений в текст (`beforeTextChanged`) или что текст уже был изменен (`afterTextChanged`), мы просто переопределяем каждый из этих методов интерфейса `TextWatcher` пустым телом (строки 146–155), чтобы выполнить контракт интерфейса.

3.6. Файл `AndroidManifest.xml`

В этом разделе мы отредактируем файл `AndroidManifest.xml`, чтобы указать, что активность приложения поддерживает только портретную ориентацию устройства, а виртуальная клавиатура должна отображаться постоянно. Для определения этих настроек будет использоваться редактор манифеста.

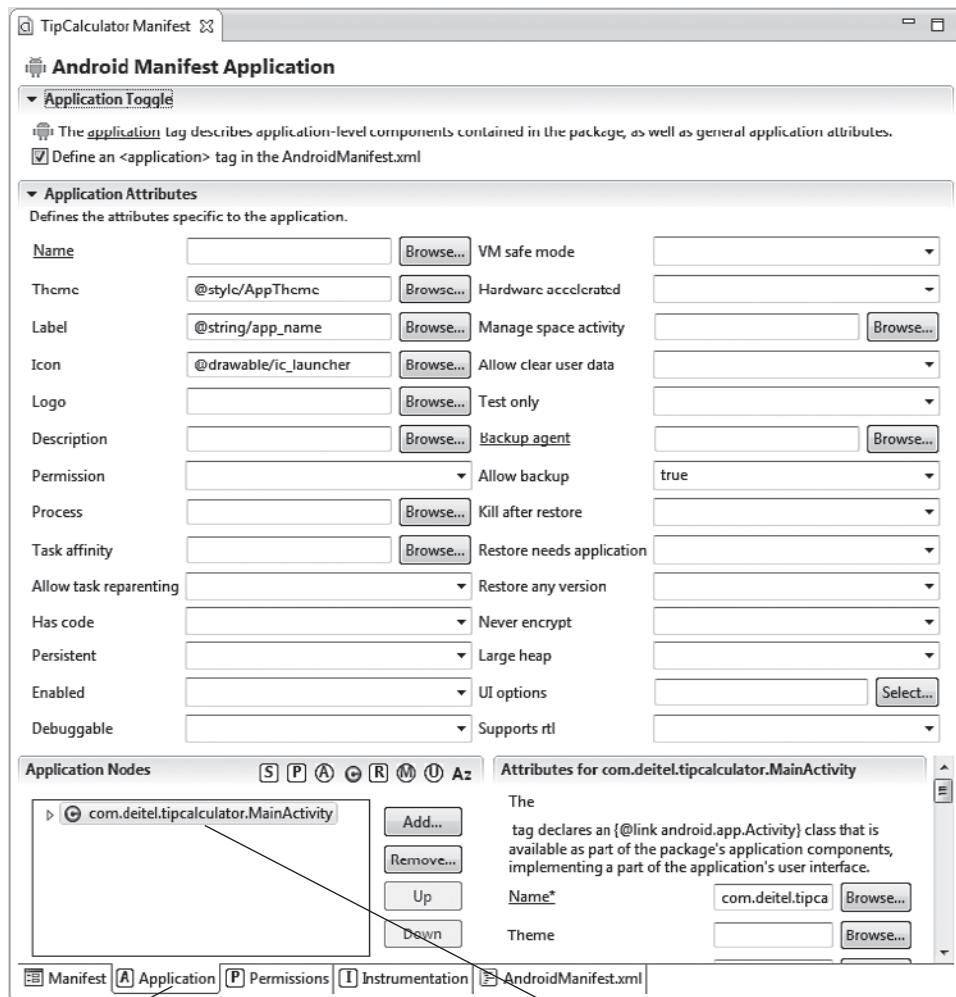
Чтобы открыть редактор манифеста (Android Manifest editor), сделайте двойной щелчок на файле `AndroidManifest.xml` приложения в окне Package Explorer. Щелкните на вкладке Application (рис. 3.8), затем выделите узел `MainActivity` в разделе Application Nodes в нижней части окна. Настройки `MainActivity` выводятся в разделе Attributes for `com.deitel.tipcalculator.MainActivity`.

Настройка `MainActivity` для портретной ориентации

Большинство приложений поддерживает как портретную, так и альбомную ориентацию. В портретной ориентации высота устройства больше его ширины. В альбомной ориентации ширина устройства больше высоты. В приложении Tip Calculator переход устройства в альбомную ориентацию на типичном телефоне приведет к тому, что цифровая клавиатура закроет большую часть графического интерфейса. По этой причине мы настроим класс `MainActivity` так, чтобы он поддерживал только портретную ориентацию. Найдите в разделе Attributes for `com.deitel.tipcalculator.MainActivity` редактора манифеста параметр `Screen orientation` и выберите значение `portrait`.

Постоянное отображение виртуальной клавиатуры для `MainActivity`

В приложении Tip Calculator виртуальная клавиатура должна появиться сразу же после запуска приложения и постоянно оставаться на экране. Найдите в разделе Attributes for `com.deitel.tipcalculator.MainActivity` редактора манифеста параметр `Window`



Вкладка Application

Выделите этот узел, чтобы задать настройки MainActivity

Рис. 3.8. Вкладка Application редактора манифеста

soft input mode и выберите значение stateAlwaysVisible. Учтите, что при наличии аппаратной клавиатуры в этом режиме виртуальная клавиатура отображаться не будет.

3.7. Резюме

С помощью этой главы вы создали свое первое *интерактивное* приложение Android — Tip Calculator. Сначала были вкратце рассмотрены функции этого

приложения, потом в целях тестирования вычислялось значение стандартных и пользовательских чаевых на основе введенного счета. Вы выполнили подробные пошаговые инструкции по созданию графического интерфейса пользователя приложения в макетном редакторе среды разработки Android Developer Tools, окне Outline и окне свойств. Мы также рассмотрели код `MainActivity` — субкласса `Activity`, определяющего функциональность приложения.

При разработке графического интерфейса приложения компонент `GridLayout` использовался для распределения компонентов GUI по строкам и столбцам. Текст выводился в компонентах `TextView`, а для ввода данных использовались компоненты `EditText` и `SeekBar`.

В классе `MainActivity` задействованы многие средства объектно-ориентированного программирования на языке Java, включая классы, объекты, интерфейсы, анонимные внутренние классы и наследование. Также была представлена концепция заполнения графического интерфейса, то есть преобразования содержимого файла XML в его экранное представление. Вы познакомились с классом Android `Activity` и жизненным циклом активности. В частности, мы переопределили метод `onCreate` для инициализации приложения при запуске. В методе `onCreate` метод `findViewById` класса `Activity` использовался для получения ссылок на визуальные компоненты, с которыми приложение взаимодействует на программном уровне. Мы определили анонимный внутренний класс, реализующий интерфейс `TextWatcher`, чтобы приложение могло вычислять новые чаевые и общую сумму при изменении текста в `EditText`.

Также был определен анонимный внутренний класс, реализующий интерфейс `OnSeekBarChangeListener`, чтобы приложение могло вычислить новые чаевые и общую сумму при изменении пользовательского процента чаевых перемещением ползунка `SeekBar`.

Наконец, мы открыли файл `AndroidManifest.xml` в редакторе манифеста среды разработки, чтобы указать, что `MainActivity` поддерживает только портретную ориентацию, а класс `MainActivity` всегда должен отображать виртуальную клавиатуру.

Используя макетный редактор среды разработки, окно `Outline`, окно свойств и редактор манифеста, мы построили приложение без прямого редактирования разметки XML в файлах ресурсов приложения и файле `AndroidManifest.xml`.

В следующей главе мы познакомимся с коллекциями в процессе создания приложения Twitter® Searches. Многие мобильные приложения отображают списки. Для решения этой задачи можно воспользоваться классом `ListActivity`, содержащим элемент `ListView`, связанный с `ArrayList<String>`. Также будет рассмотрено сохранение данных приложения в пользовательских настройках и запуск браузера для отображения веб-страницы.

4 Приложение Twitter® Searches

Настройки SharedPreferences, коллекции, ImageButton, ListView, ListActivity, ArrayAdapter, неявные интенты и AlertDialog

В этой главе...

- Поддержка обеих ориентаций устройства (портретной и альбомной)
- Расширение класса ListActivity для создания активности, отображающей список вариантов в ListView
- Взаимодействие пользователей с приложением посредством кнопок ImageButton
- Использование компонента ScrollView для отображения объектов, не помещающихся на экране
- Работа с коллекциями данных
- Использование SharedPreferences для хранения данных, связанных с приложением, в виде пар «ключ/значение»
- Изменение пар «ключ/значение» в данных приложения с помощью SharedPreferences.Editor
- Использование класса ArrayAdapter для определения данных ListView
- Создание диалоговых окон AlertDialog с помощью объекта AlertDialog.Builder
- Программное открытие веб-сайта в окне браузера с помощью интентов
- Использование неявного интента для вывода списка приложений
- Программное скрытие виртуальной клавиатуры

4.1. Введение

Поисковый механизм Твиттера упрощает отслеживание актуальных тем, обсуждаемых более 500 миллионами пользователей. Поисковые запросы Твиттера могут настраиваться с помощью *поисковых операторов* Твиттера (см. раздел 4.2), при этом следует иметь в виду, что более сложные запросы имеют большую длину, и вводить их на мобильных устройствах трудно и неудобно. Приложение Twitter Searches (рис. 4.1) позволяет сохранить избранные поисковые запросы пользователя на устройстве с короткими, легко запоминающимися именами (тегами) (рис. 4.1, а). Касаясь кнопки поиска, вы можете быстро и легко отслеживать сообщения по заданной теме (рис. 4.1, б). Как вы вскоре увидите, приложение также позволяет пересылать, редактировать и удалять сохраненные запросы.

Приложение поддерживает как портретную, так и альбомную ориентацию устройства. В некоторых приложениях разработчик предоставляет отдельный макет для каждой ориентации. В этом приложении мы обеспечим поддержку обеих ориентаций, спроектировав графический интерфейс с возможностью динамического изменения размеров компонентов GUI для текущей ориентации.

Начнем с тестового запуска приложения, а затем приведем краткий обзор технологий, использованных для его построения. Далее будет спроектирован графический интерфейс приложения. Глава завершается описанием полного исходного кода приложения, при этом основное внимание будет уделяться его новым возможностям.

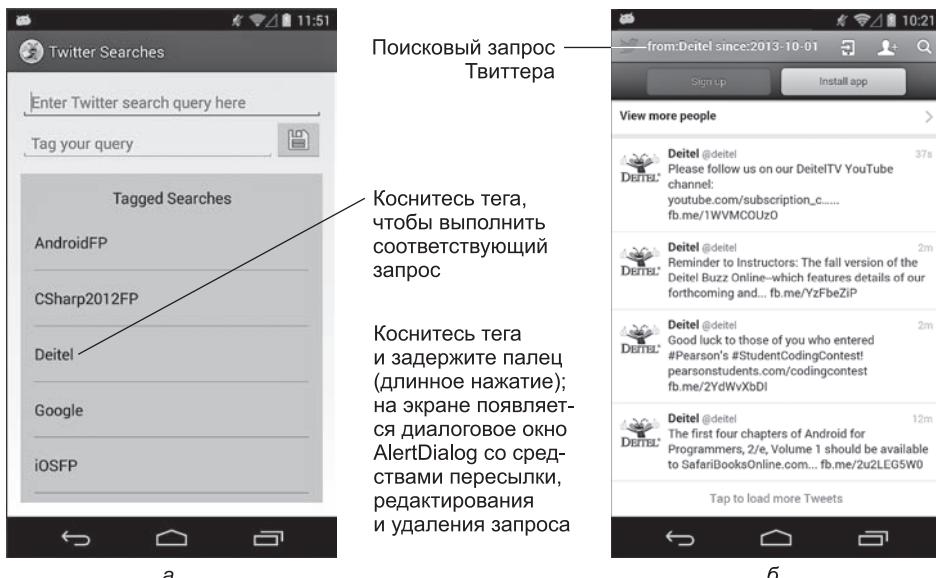


Рис. 4.1. Приложение Twitter Searches: а — приложение с несколькими сохраненными поисковыми запросами; б — приложение после касания кнопки Deitel

4.2. Тестирование приложения

В этом разделе мы опробуем приложение Twitter Searches в деле. Откройте среду разработки Android Developer Tools и импортируйте проект приложения Twitter Searches. Как было сделано в главе 3, запустите Nexus 4 AVD (или подключите устройство Android к компьютеру) для тестирования. Экранные снимки, приведенные в этой главе, были сделаны на телефоне Nexus 4.

4.2.1. Импорт приложения

Выполните следующие действия.

1. Откройте диалоговое окно Import командой **File > Import...**
2. Импортируйте проект приложения Twitter Searches. Раскройте узел **General** и выберите пункт **Existing Projects into Workspace**, потом нажмите **Next >** для перехода к шагу **Import Projects**. Установите флажок **Select root directory**, потом нажмите **Browse....** Найдите папку **TwitterSearches**, находящуюся в папке примеров книги, выделите ее и нажмите **OK**. Проследите за тем, чтобы флажок **Copy projects into workspace** не был установлен. Нажмите **Finish** для завершения импортирования. Проект появляется в окне **Package Explorer**.
3. Запустите приложение Twitter Searches. Щелкните правой кнопкой мыши на проекте **TwitterSearches** в окне **Package Explorer** и выберите команду **Run As > Android Application** для среды AVD. Команда строит проект и запускает приложение (рис. 4.2).

4.2.2. Добавление нового запроса

Коснитесь верхнего компонента **EditText** и введите поисковый запрос *from:deitel* — оператор *from:* включает поиск сообщений от заданного пользователя Твиттера. В табл. 4.1 представлены примеры использования поисковых операторов Твиттера; в сложных запросах можно задействовать сразу несколько операторов. Полный список доступен по адресу <http://bit.ly/TwitterSearchOperators>.

Таблица 4.1. Некоторые поисковые операторы Твиттера

Пример	Критерий поиска сообщений
deitel iOS6	Неявный оператор «логическое И» — находит сообщения, содержащие текст <i>deitel</i> и текст <i>iOS6</i>
deitel OR iOS6	Логический оператор «ИЛИ» — находит сообщения, содержащие текст <i>deitel</i> или текст <i>iOS6</i> (или оба текста сразу)
“how to program”	Строка в кавычках — находит сообщения, содержащие текст в данном виде (« <i>how to program</i> »)

Таблица 4.1 (окончание)

Пример	Критерий поиска сообщений
deitel ?	Вопросительный знак — находит сообщения с текстом deitel, в которых задается вопрос
deitel -sue	Знак «минус» — находит сообщения, содержащие текст deitel, но не содержащие sue
deitel :)	:) — находит <i>позитивные</i> сообщения, содержащие текст deitel
deitel :(:(— находит <i>негативные</i> сообщения, содержащие текст deitel
since:2013-10-01	Находит сообщения, опубликованные в указанный день или после него (дата задается в формате ГГГГ-ММ-ДД)
near:"New York City"	Находит сообщения, отправленные рядом с заданным местом
from:deitel	Находит сообщения от пользователя Твиттера @deitel
to:deitel	Находит сообщения, адресованные пользователю Твиттера @deitel

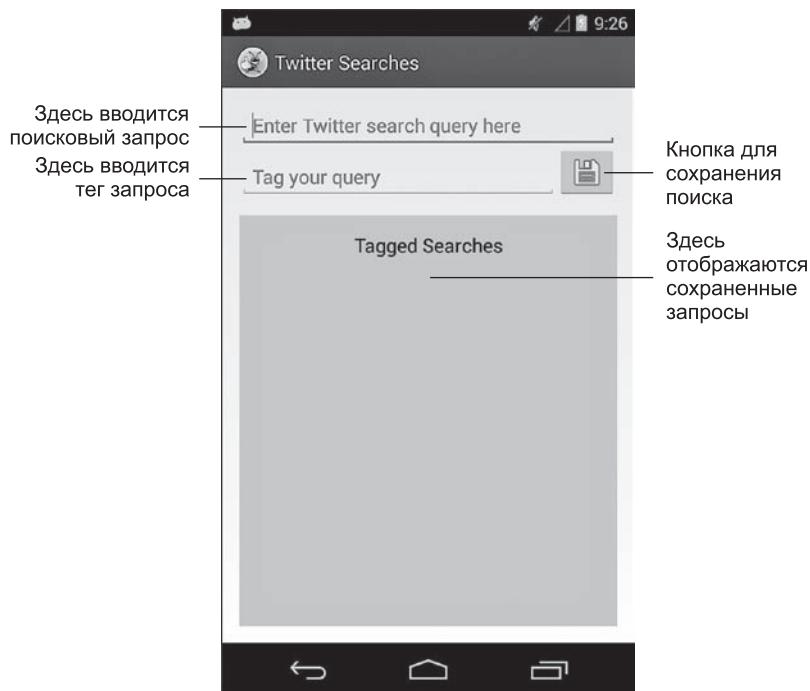


Рис. 4.2. Приложение Twitter Searches при первом запуске

В нижнем компоненте `EditText` введите тег запроса `Deitel` (рис. 4.3, а). Это короткое имя будет отображаться в списке сохраненных запросов. Коснитесь кнопки 

чтобы сохранить запрос, — тег «Deitel» появляется в списке под заголовком Tagged Searches (рис. 4.3, б). При сохранении запроса виртуальная клавиатура исчезает, чтобы список был виден на экране, — о том, как скрыть виртуальную клавиатуру, рассказано в разделе 4.5.5.

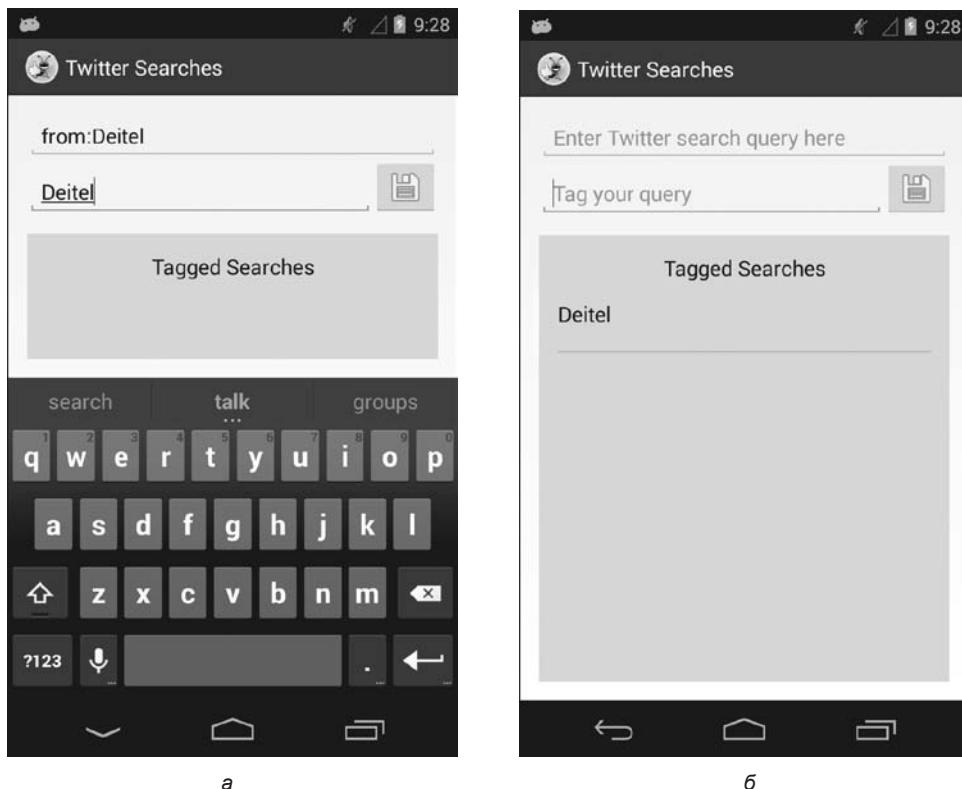


Рис. 4.3. Ввод поискового запроса: а — ввод запроса и тега (короткого имени); б — приложение после сохранения запроса и тега

4.2.3. Просмотр результатов поиска

Чтобы просмотреть результаты поиска, коснитесь тега Deitel. На устройстве запускается браузер, которому передается URL-адрес, представляющий сохраненный поисковый запрос. Твиттер получает поисковый запрос из URL и возвращает сообщения, соответствующие запросу (если они обнаружены), в виде веб-страницы. Затем браузер выводит страницу результатов (рис. 4.4). Просмотрев результаты, коснитесь кнопки Back (◀), чтобы вернуться к приложению Twitter Searches, в котором вы можете сохранить новые запросы, а также отредактировать, удалить и опубликовать ранее сохраненные запросы.

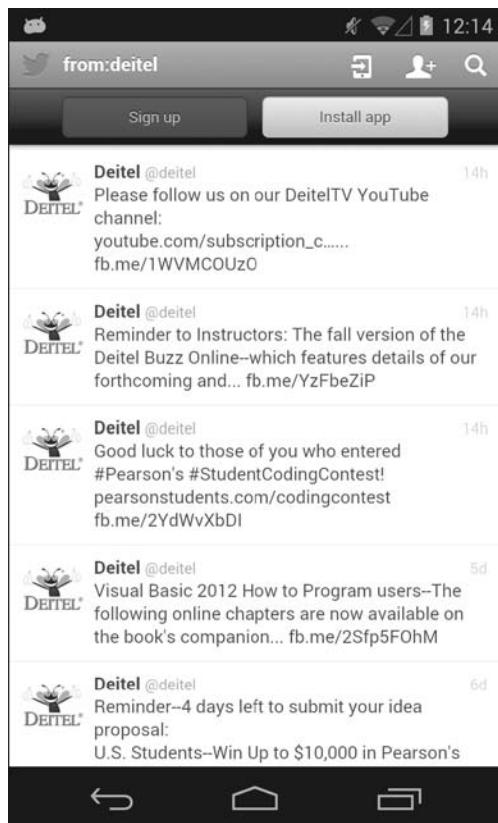


Рис. 4.4. Просмотр результатов поиска

4.2.4. Редактирование запроса

Запросы также можно *пересылать, редактировать и удалять*. Чтобы увидеть эти команды, сделайте длинное нажатие на теге запроса (коснитесь тега и задержите палец у экрана). Если вы используете AVD, щелкните и удерживайте левую кнопку мыши на теге, чтобы имитировать длинное нажатие. При длинном нажатии на теге Deitel открывается диалоговое окно AlertDialog (рис. 4.5, а) с командами Share, Edit и Delete для поискового запроса с тегом Deitel. Если вы не хотите выполнить ни одну из этих операций, коснитесь кнопки Cancel.

Чтобы изменить запрос с тегом Deitel, коснитесь команды Edit в диалоговом окне. Приложение загружает запрос и тег в компоненты EditText для редактирования. Чтобы ограничить поиск сообщениями, отправленными после 1 октября 2013 года, добавьте фрагмент since:2013-10-01 в конец запроса (рис. 4.5, б) в верхнем компоненте EditText. Оператор since: ограничивает результаты поиска сообщениями,

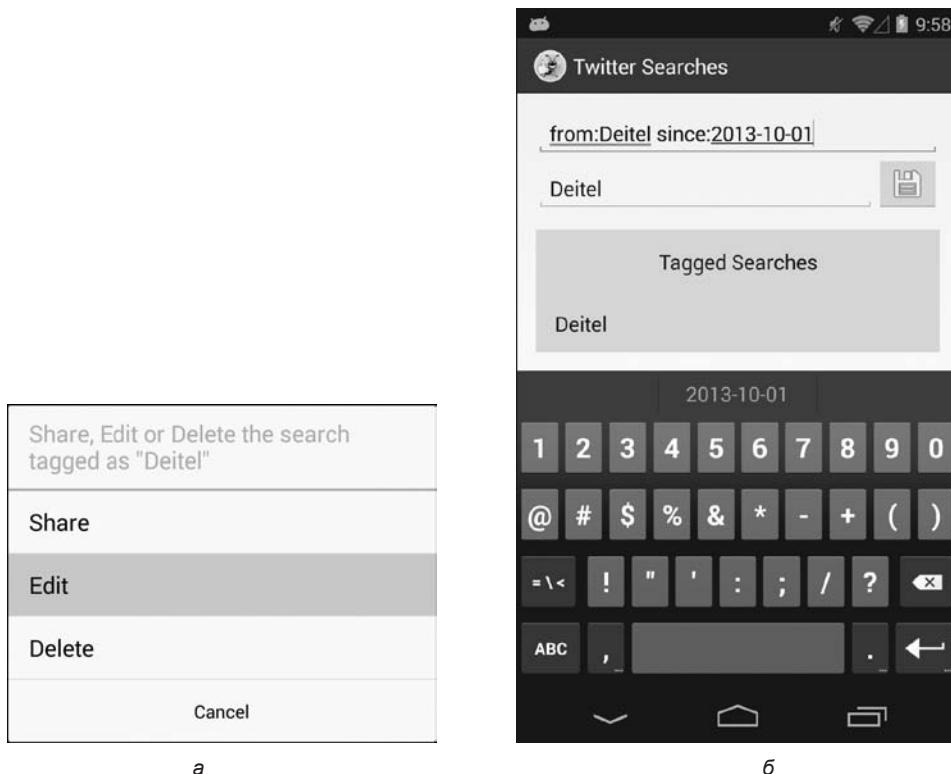


Рис. 4.5. Редактирование сохраненного запроса: а — выбор команды Edit для редактирования существующего запроса; б — редактирование сохраненного запроса Deitel

отправленными в заданный день или после него (в формате ГГГГ-ММ-ДД). Коснитесь кнопки , чтобы обновить сохраненный запрос, и просмотрите обновленные результаты (рис. 4.6), коснувшись тега Deitel в разделе Tagged Searches приложения. [Примечание: изменение имени тега приводит к созданию новой кнопки поиска — эта возможность удобна для создания новых запросов на базе ранее сохраненного запроса.]

4.2.5. Пересылка запроса

Android упрощает распространение разных видов информации из приложений по электронной почте, через системы мгновенной передачи сообщений (SMS), Facebook, Google+ и т. д. Наше приложение позволяет переслать информацию о поисковом запросе — сделайте длинное нажатие на теге запроса и выберите команду Share в открывшемся окне AlertDialog. На экране появляется окно выбора (рис. 4.7, а), содержимое которого зависит от типа передаваемой информации и приложений,

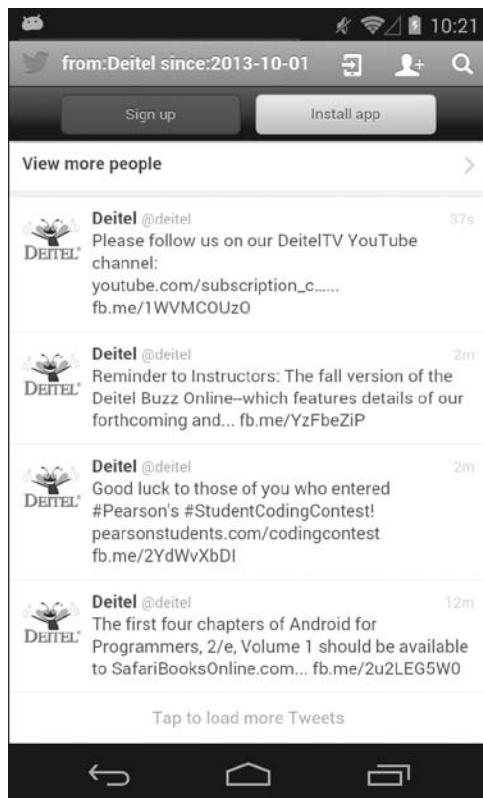


Рис. 4.6. Просмотр обновленных результатов поиска «Deitel»

способных эту информацию обрабатывать. В нашем приложении публикуется текстовая информация, а окно выбора на нашем телефоне (не в AVD!) содержит список приложений, способных обрабатывать текст: Facebook, Gmail, Google+, Messaging (мгновенные сообщения) и Twitter. Если информация не может быть обработана ни одним приложением, в окне выбора выводится соответствующее сообщение. Если информация обрабатывается только одним приложением, это приложение запускается без отображения промежуточного окна выбора. На рис. 4.7, б показан экран Compose приложения Gmail с заполненным полем темы и телом сообщения. Gmail также выводит ваш адрес электронной почты над полем То (мы удалили его на экранном снимке по соображениям конфиденциальности).

4.2.6. Удаление запроса

Чтобы удалить запрос, выполните длинное нажатие на теге запроса и выберите в открывшемся окне AlertDialog команду Delete. Приложение предлагает подтвердить удаление запроса (рис. 4.8) — кнопка Cancel возвращает приложение к главному экрану без удаления, а кнопка Delete удаляет запрос.

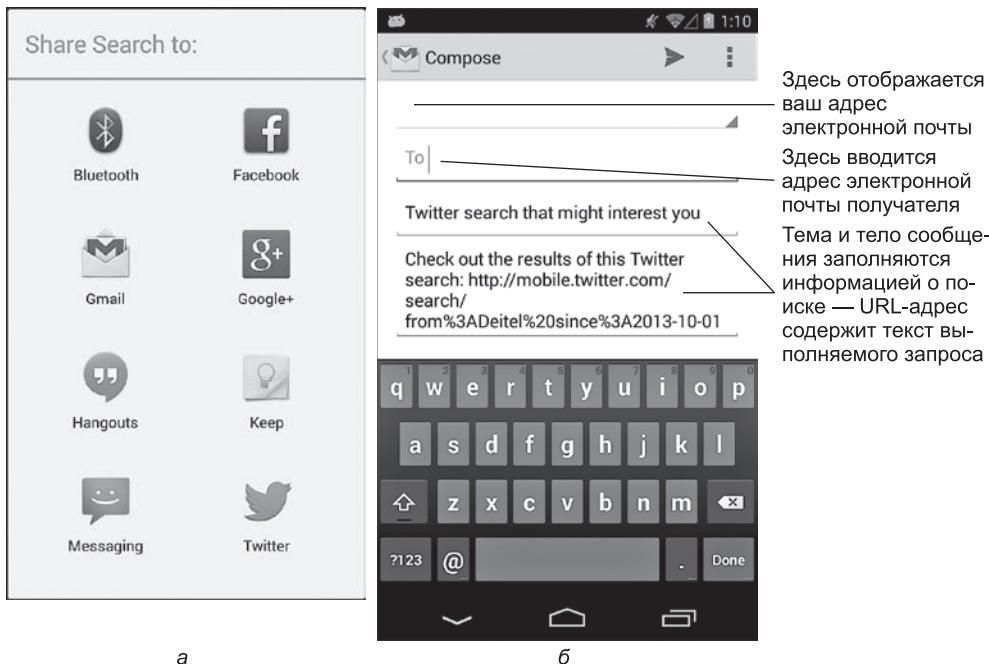


Рис. 4.7. Пересылка запроса по электронной почте: *а* — окно выбора с вариантами пересылки информации; *б* — экран Compose приложения Gmail для поискового запроса

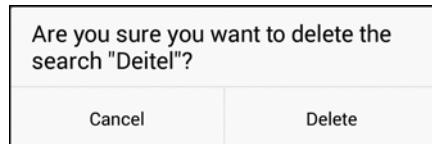


Рис. 4.8. Окно AlertDialog с предложением подтвердить удаление

4.2.7. Прокрутка списка сохраненных запросов

На рис. 4.9 изображено приложение после сохранения 10 запросов, только пять из которых видны в настоящий момент. Приложение дает возможность прокрутить список запросов, если он не помещается на экране полностью. Для вывода списка используется компонент `ListView` (см. раздел 4.3.1). Чтобы прокрутить список, проведите пальцем (или мышью в AVD) по списку `Tagged Searches`. Также попробуйте повернуть устройство в альбомную ориентацию и убедитесь в том, что графический интерфейс динамически подстраивается под новые размеры.

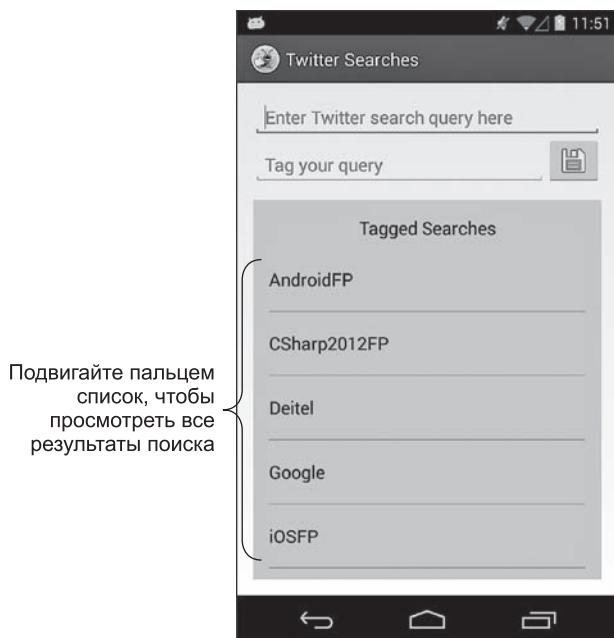


Рис. 4.9. Приложение с длинным списком запросов, не помещающимся на экране

4.3. Обзор применяемых технологий

В этом разделе представлены некоторые функции, используемые в приложении.

4.3.1. ListView

Во многих мобильных приложениях информация выводится в виде списка. Например, почтовый клиент выводит список новых сообщений; адресная книга выводит список контактов; программа чтения новостей выводит список заголовков и т. д. В каждом случае пользователь касается элемента списка, чтобы получить дополнительную информацию — содержимое выбранного сообщения, расширенную информацию о выбранном контакте, текст выбранной новости и т. д. В нашем приложении компонент `ListView` (пакет `android.widget`) используется для вывода списка запросов, который может *прокручиваться*, если полный список не помещается на экране. Вы можете указать способ форматирования каждого элемента `ListView`; в нашем приложении тег каждого запроса будет отображаться как строка в компоненте `TextView`. В следующих приложениях вы научитесь полностью изменять содержимое, отображаемое для элементов списка `ListView` — включать в них графику, текст и кнопки.

4.3.2. ListActivity

Если главной задачей активности является вывод прокручиваемого списка элементов, вы можете расширить класс `ListActivity` (пакет `android.app`). Этот класс по умолчанию использует макет, в котором компонент `ListView` занимает все приложение. `ListView` является субклассом `AdapterView` (пакет `android.widget`) — компонентом GUI, связанным с источником данных через объект `Adapter` (пакет `android.widget`). В этом приложении класс `ArrayAdapter` (пакет `android.widget`) используется для создания объекта, заполняющего `ListView` данными из объекта коллекции `ArrayList`. Подобное связывание называется *привязкой к данным* (*data binding*). Существует несколько разновидностей `AdapterView`, поддерживающих привязку к данным через `Adapter`. В главе 8 вы научитесь связывать `ListView` с базами данных. За дополнительной информацией о привязке к данным в Android и учебными руководствами обращайтесь к

<http://developer.android.com/guide/topics/ui/binding.html>

4.3.3. Настройка макета ListActivity

Графический интерфейс `ListActivity` по умолчанию содержит только компонент `ListView`, заполняющий клиентскую область экрана между верхней и нижней системными панелями Android (см. рис. 2.1). Если для графического интерфейса `ListActivity` достаточно стандартной реализации `ListView`, вам не придется определять отдельный макет для своего субкласса `ListActivity subclass`.

Класс `MainActivity` приложения Twitter Searches отображает несколько компонентов графического интерфейса, поэтому для `MainActivity` будет создан нестандартный макет. При построении графического интерфейса субкласса `ListActivity` макет должен содержать компонент `ListView`, у которого атрибуту `Id` задано значение "`@android:id/list`" — имя, которое используется классом `ListActivity` для ссылок на `ListView`.

4.3.4. ImageButton

Пользователи часто нажимают кнопки, чтобы выполнить некоторую операцию в программе. Сохранение пары «запрос—тег» в этом приложении осуществляется кнопкой `ImageButton` (пакет `android.widget`). `ImageButton` — субкласс `ImageView` с дополнительными возможностями, которые позволяют использовать изображение как объект `Button` (пакет `android.widget`) для выполнения операции.

4.3.5. SharedPreferences

С приложением может быть связан один или несколько файлов, содержащих пары «ключ/значение» (ключ обеспечивает быстрый доступ к соответствующему значению). В нашем приложении используется файл с именем `searches`, в котором хранятся

пары из тегов (ключи) и поисковых запросов Твиттера (значения), созданных пользователем. Для чтения пар «ключ—значение» из этого файла используются объекты `SharedPreferences.Editor` (пакет `android.content`). Ключами должны быть строки (`String`), а значениями — строки или значения примитивных типов.

Приложение читает сохраненные запросы в методе `onCreate` класса активности — такое решение допустимо только при относительно небольшом объеме загружаемых данных. Когда приложение будет запущено, Android создает главный программный поток, называемый *потоком пользовательского интерфейса*, или *UI-потоком*; он обрабатывает все взаимодействие с графическим интерфейсом. Вся обработка GUI должна осуществляться в этом потоке. Продолжительные операции ввода/вывода (например, загрузка данных из файлов и баз данных) не следует выполнять в UI-потоке, потому что такие операции замедляют отклик приложения на действия пользователя. В следующих главах мы покажем, как выполнять ввод/вывод в отдельных потоках.

4.3.6. Интенты запуска других активностей

Для передачи информации между активностями (внутри одного приложения или разных приложений) в Android используются *интенты* (intents). Каждая активность может определять *фильтры интентов*, обозначающие действия, которые могут выполняться этой активностью. Фильтры интентов определяются в файле `AndroidManifest.xml`. Собственно, во всех приложениях, которые мы создавали ранее, среда разработки генерировала фильтр интентов для единственной активности приложения; этот фильтр сообщает, что он может реагировать на заранее предопределено действие с именем `android.intent.action.MAIN`, указывающее, что активность может использоваться для запуска приложения.

Интент используется для запуска активности — он указывает, какое действие должно выполняться и с какими данными оно должно выполняться. В нашем приложении, когда пользователь касается тега запроса, строится URL-адрес, содержащий поисковый запрос Твиттера. Страница поиска загружается в браузере, для чего приложение создает новый объект `Intent` для просмотра URL и передает его методу `startActivity`, неявно унаследованному нашим приложением от класса `Activity`. Для просмотра URL метод `startActivity` запускает браузер устройства для вывода содержимого — в данном случае результатов поиска в Твиттере.

Мы также используем `Intent` и метод `startActivity` для отображения *окна выбора* — окна со списком приложений, способных обработать указанный интент. Это делается при пересылке сохраненного поиска, чтобы пользователь мог выбрать, каким именно способом должна передаваться информация.

Явные и неявные интенты

Интенты, используемые в этом приложении, относятся к категории *неявных* (*implicit*) — разработчик не указывает компонент для отображения веб-страницы, а позволяет Android запустить наиболее подходящую активность в зависимости

от типа данных. Если действие и информация, переданная `startActivity`, могут быть обработаны несколькими активностями, система выводит диалоговое окно, в котором пользователь выбирает активность. Если система не может найти активность для обработки действия, метод `startActivity` инициирует исключение `ActivityNotFoundException`. В общем случае рекомендуется предусмотреть обработку этого исключения в программах. В нашем приложении это не делается, потому что в устройствах Android, на которых будет установлено приложение, наверняка присутствует браузер, способный вывести веб-страницу.

В будущих приложениях мы также будем использовать *явные* интенты, которые точно определяют запускаемую активность. За дополнительной информацией об интентах обращайтесь по адресу

<http://developer.android.com/guide/components/intents-filters.html>

4.3.7. AlertDialog

Объекты `AlertDialog` используются для вывода сообщений, вариантов действий и их подтверждения. Пока диалоговое окно находится на экране, пользователь не может работать с приложением — такие диалоговые окна называются *модальными*. Как вы вскоре увидите, приложение сначала задает параметры диалогового окна с помощью объекта `AlertDialog.Builder`, а затем использует этот объект для создания `AlertDialog`.

Объекты `AlertDialog` могут отображать кнопки, флажки, переключатели и списки, при помощи которых пользователь реагирует на сообщение в диалоговом окне. Стандартный объект `AlertDialog` может содержать до трех кнопок, представляющих следующие действия.

- ❑ Негативное действие — отменяет операцию диалогового окна (часто эта кнопка помечается надписью *Отмена* или *Нет*). Если диалоговое окно содержит несколько кнопок, эта кнопка находится в крайней левой позиции.
- ❑ Позитивное действие — подтверждает операцию диалогового окна (часто кнопка помечается надписью *OK* или *Да*). Если диалоговое окно содержит несколько кнопок, эта кнопка находится в крайней правой позиции.
- ❑ Нейтральное действие — кнопка указывает, что пользователь не хочет ни отменять, ни подтверждать операцию. Например, приложение, которое предлагает пользователю зарегистрироваться для получения доступа к расширенной функциональности, может предоставить нейтральную кнопку *Напомнить позднее*.

В нашем приложении объекты `AlertDialog` используются для следующих целей.

- ❑ Для вывода сообщения о том, что компонент `EditText` тега или запроса не содержит данных. Это диалоговое окно содержит только позитивную кнопку.
- ❑ Для отображения команд пересылки, редактирования и удаления запроса. Это диалоговое окно содержит список вариантов и негативную кнопку.

- Для подтверждения удаления запроса (на случай, если пользователь нечаянно коснулся кнопки Delete).

Дополнительную информацию о диалоговых окнах Android можно найти по адресу <http://developer.android.com/guide/topics/ui/dialogs.html>

4.3.8. Файл AndroidManifest.xml

Как упоминалось в главе 3, файл AndroidManifest.xml генерируется при создании приложения. Для нашего приложения в манифест будет добавлена новая настройка, предотвращающая отображение виртуальной клавиатуры при первой загрузке приложения. За подробной информацией об AndroidManifest.xml обращайтесь по адресу

<http://developer.android.com/guide/topics/manifest/manifest-intro.html>

4.4. Построение графического интерфейса приложения

В этом разделе мы построим графический интерфейс приложения Twitter Searches. Также будет создан второй макет XML; компонент ListView будет динамически заполнять его и использовать для отображения каждого элемента списка.

4.4.1. Создание проекта

Вспомните, что Android Developer Tools IDE не позволяет включить в рабочую область два одноименных проекта, поэтому перед созданием нового проекта удалите проект TwitterSearches, протестированный в разделе 4.2. Для этого щелкните на нем правой кнопкой мыши и выберите команду Delete. Проследите за тем, чтобы в открывшемся диалоговом окне не был установлен флажок Delete project contents on disk, и нажмите OK. Проект удаляется из рабочей области, но папка и файлы проектов остаются на диске на случай, если вы вдруг захотите снова протестировать исходное приложение.

Создание нового проекта пустого приложения

Создайте новый проект Android Application Project. Введите следующие значения на первом шаге мастера New Android Project (New Android Application) и нажмите кнопку Next>:

- Application Name: Twitter Searches
- Project Name: TwitterSearches

- Package Name: com.deitel.twittersearches
- Minimum Required SDK: API18: Android 4.3
- Target SDK: API19: Android 4.4
- Compile With: API19: Android 4.4
- Theme: Holo Light with Dark Action Bar

На втором шаге мастера New Android Project (New Android Application) оставьте значения по умолчанию и нажмите кнопку **Next >**. На шаге Configure Launcher Icon нажмите **Browse...** и выберите значок приложения (находится в папке **images** примеров книги), нажмите **Open** и нажмите кнопку **Next >**. На шаге Create Activity выберите **Blank Activity** и нажмите кнопку **Next >**. На шаге **Blank Activity** оставьте значения по умолчанию и нажмите **Finish**, чтобы создать проект. В макетном редакторе выберите в списке типов экрана **Nexus 4** (см. рис. 2.9) — как и в предыдущем случае, это устройство будет взято за основу для построения интерфейса.

4.4.2. Файл activity_main.xml

Как и в главе 3, в файле **activity_main.xml** этого приложения используется компонент **GridLayout** (рис. 4.10). В нашем приложении компонент **GridLayout** содержит три строки и один столбец. На рис. 4.11 показаны имена компонентов GUI.

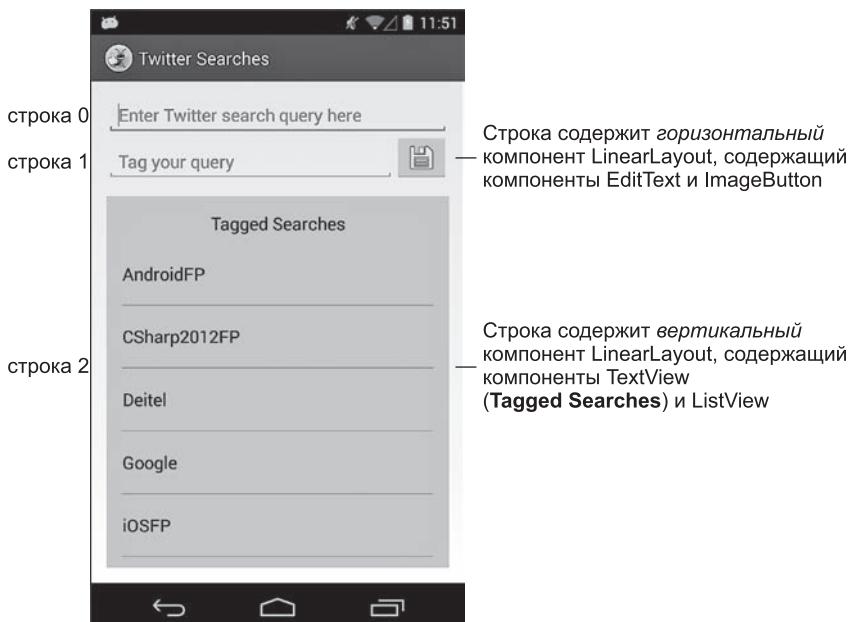


Рис. 4.10. Строки и столбцы в компоненте GridLayout приложения Twitter Searches

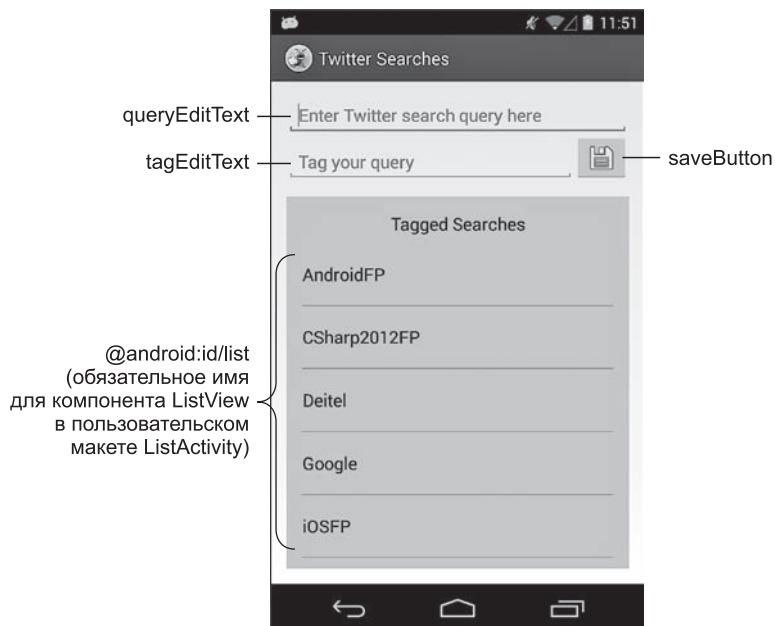


Рис. 4.11. Компоненты графического интерфейса приложения Twitter Searches со значениями свойств Id

4.4.3. Добавление GridLayout и других компонентов

Мы построим графический интерфейс на рис. 4.10–4.11 с использованием средств, о которых вы узнали в главе 3. Все дальнейшие действия предполагают, что вы работаете в макетном редакторе среды разработки. Стоит напомнить, что нужный компонент GUI часто оказывается пропущен в окне Outline.

Мы начнем с построения базовой структуры и элементов управления, а затем займемся настройкой свойств элементов. Для добавления компонентов в нужные строки `GridLayout` используйте окно Outline. Добавляя компоненты GUI, задайте их идентификаторы в соответствии с рис. 4.11; некоторым компонентам макета идентификаторы не нужны, так как приложение нигде не обращается к ним из кода Java. Также не забудьте, что все литературальные строки должны определяться в файле `strings.xml` (в папке `res/values`).

Шаг 1: Переход от `RelativeLayout` к `GridLayout`

Повторите действия из раздела 3.4.3, чтобы удалить компонент `TextView` из макета приложения и переключиться с компонента `RelativeLayout` на `GridLayout`.

Шаг 2: Настройка `GridLayout`

В окне Outline выберите компонент `GridLayout` и задайте следующие свойства — для каждого свойства, вложенного в узел в окне свойств, за именем свойства указывается имя узла в круглых скобках:

- Id: @+id/gridLayout
- Column Count (узел GridLayout): 1 — каждый компонент GUI, вложенный непосредственно в GridLayout, добавляется как новая строка.

Компонент GridLayout заполняет всю клиентскую область экрана, потому что среда разработки задает каждому из свойств Width и Height (из раздела Layout Parameters окна свойств) значение `match_parent`.

По умолчанию среда разработки задает свойствам Padding Left и Padding Right значение `@dimen/activity_horizontal_margin` — заранее определенный ресурс метрики из файла dimens.xml в папке res/values проекта. Это значение равно 16dp, так что слева и справа от компонента GridLayout создаются отступы величиной 16dp. Среда разработки создает этот ресурс при создании проекта приложения. Аналогичным образом свойствам Padding Top и Padding Bottom задается значение `@dimen/activity_vertical_margin` — другой заранее определенный ресурс метрики со значением 16dp. Итак, сверху и снизу от GridLayout тоже создаются отступы величиной 16dp.



ПРИМЕЧАНИЕ 4.1

В рекомендациях по дизайну приложений Android дается совет оставлять промежуток в 16dp между контентом приложения и краями всей области экрана; тем не менее многие приложения (например, игры) используют весь экран.

Шаг 3: Создание первой строки GridLayout

Эта строка содержит только компонент EditText. Перетащите элемент Plain Text из раздела Text Fields палитры на компонент GridLayout в окне Outline, затем задайте его свойству Id значение `@+id/queryEditText`. В окне свойств под узлом TextView удалите значение свойства Ems, не используемое в этом приложении. Затем воспользуйтесь окном свойств, чтобы задать следующие значения свойств:

- Width (узел Layout Parameters): `wrap_content`
- Height (узел Layout Parameters): `wrap_content`
- Gravity (узел Layout Parameters): `fill_horizontal` — гарантирует, что при повороте устройства queryEditText заполнит все доступное горизонтальное пространство. Аналогичные значения Gravity используются для других компонентов, чтобы графический интерфейс приложения поддерживал как портретную, так и альбомную ориентацию.
- Hint: `@string/queryPrompt` — создайте строковый ресурс (как это делалось в предыдущих приложениях) и задайте ему значение "Enter Twitter search query here". Этот атрибут отображает в пустом компоненте EditText подсказку, которая помогает пользователю понять смысл поля. Этот текст также зачитывается диктором Android TalkBack для пользователей с ослабленным зрением, поэтому постарайтесь формулировать подсказки для полей EditText так, чтобы ваше приложение стало более доступным.



ПРИМЕЧАНИЕ 4.2

В рекомендациях по дизайну приложений Android сказано, что текст, отображаемый в графическом интерфейсе, должен быть коротким, простым и понятным, а важные слова должны находиться в начале. За подробностями о рекомендуемом стиле обращайтесь по адресу <http://developer.android.com/design/style/writing.html>.

- ❑ IME Options (узел `TextView`): `actionNext` — это значение указывает, что клавиатура `EditText` должна содержать кнопку `Next` для передачи фокуса следующему компоненту ввода (в нашем приложении это компонент `tagEditText`). Такой способ передачи фокуса упрощает заполнение нескольких компонентов ввода на форме. Когда следующим компонентом является другое поле `EditText`, соответствующая клавиатура отображается автоматически — пользователю не нужно касаться компонента `EditText`, чтобы передать ему фокус.

Шаг 4: Создание второй строки GridLayout

Эта строка содержит горизонтальный компонент `LinearLayout` с `EditText` и `ImageButton`. Чтобы построить графический интерфейс строки, выполните следующие действия.

1. Перетащите компонент `LinearLayout (Horizontal)` из раздела `Layouts` палитры на компонент `GridLayout` в окне `Outline`.
2. Перетащите компонент `Plain Text` из раздела `Text Fields` палитры на `LinearLayout` и задайте его свойству `Id` значение `@+id/tagEditText`.
3. Перетащите компонент `ImageButton` из раздела `Images & Media` палитры на `LinearLayout`. На экране появляется диалоговое окно `Resource Chooser` (рис. 4.12), в котором можно выбрать изображение для кнопки. По умолчанию в окне установлен переключатель `Project Resources`, чтобы изображения выбирались из ресурсов проекта (такие изображения хранятся в папках `res/drawable` проекта). В нашем приложении используется стандартный значок сохранения Android (в правой части рис. 4.12). Чтобы выбрать его, щелкните на переключателе `System Resources`, выберите ресурс `ic_menu_save` и нажмите `OK`. Затем задайте свойству `Id` значение `@+id/saveButton`.

Выделите компонент `tagEditText` и удалите значение свойства `Ems`, не используемое в этом приложении. Затем задайте следующие свойства.

- ❑ `Width` (узел `Layout Parameters`): `0dp` — среда разработки рекомендует использовать это значение, если вы также задаете свойство `Weight`, чтобы компоненты можно было разместить более эффективно.
- ❑ `Height` (узел `Layout Parameters`): `wrap_content`.
- ❑ `Gravity` (узел `Layout Parameters`): `bottom|fill_horizontal` — это значение выравнивает `tagEditText` с `saveButton` по нижнему краю и указывает, что компонент `tagEditText` должен заполнять все доступное горизонтальное пространство.

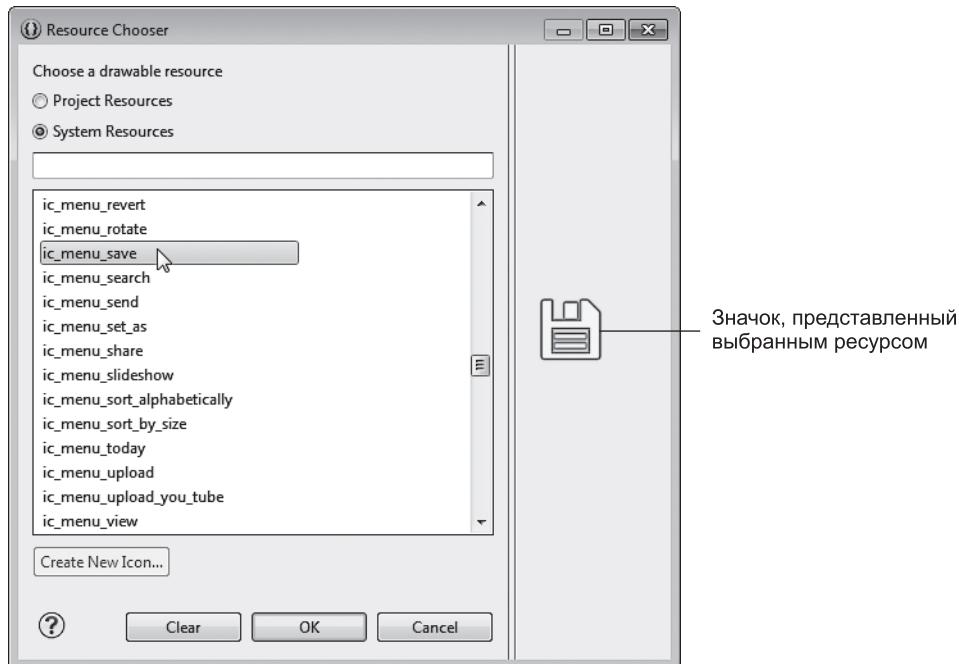


Рис. 4.12. Диалоговое окно Resource Chooser

- ❑ Weight (узел Layout Parameters): 1 — компоненту `tagEditText` назначается больший вес, чем кнопке `saveButton` в этой строке. Когда Android формирует макет строки, `saveButton` будет занимать только фактически необходимое место, а `tagEditText` займет все оставшееся горизонтальное пространство.
- ❑ Hint: `@string/tagPrompt` — создайте строковый ресурс со значением "Tag your query".
- ❑ IME Options (узел TextView): `actionDone` — значение указывает, что клавиатура `queryEditText` будет содержать кнопку Done, при помощи которой пользователь сможет убрать клавиатуру с экрана.

Выделите компонент `saveButton` и удалите значение Weight (узел Layout Parameters), после чего задайте следующие свойства.

- ❑ Width (узел Layout Parameters): `wrap_content`.
- ❑ Height (узел Layout Parameters): `wrap_content`.
- ❑ Content Description: `@string/saveDescription` — создайте строковый ресурс со значением "Touch this button to save your tagged search".

Шаг 5: Создание третьей строки GridLayout

Эта строка содержит вертикальный компонент `LinearLayout` с `TextView` и `ListView`. Чтобы построить графический интерфейс строки, выполните следующие действия:

1. Перетащите компонент **LinearLayout (Vertical)** из раздела **Layouts** палитры на компонент **GridLayout** в окне **Outline**.
2. Перетащите компонент **Medium Text** из раздела **Form Widgets** палитры на компонент **LinearLayout**. При этом создается компонент **TextView**, предварительно настроенный для вывода текста шрифтом среднего размера выбранной темы.
3. Перетащите компонент **ListView** из раздела **Composite** палитры на компонент **LinearLayout**, затем задайте свойству **Id** значение `@android:id/list` — напомним, что этот идентификатор обязателен для **ListView** в пользовательских макетах **ListAdapter**.



ПРИМЕЧАНИЕ 4.3

Вспомните, что в Android рекомендуется строить приложения так, чтобы каждый компонент GUI мог использоваться с TalkBack. Для компонентов, не имеющих текстовых описаний (например, **ImageButton**), текст включается в свойство **Content Description** компонента.

Выделите вертикальный компонент **LinearLayout** и задайте следующие свойства:

- Height** (узел **Layout Parameters**): `0dp` — фактическая высота определяется свойством **Gravity**.
- Gravity** (узел **Layout Parameters**): `fill` — приказывает **LinearLayout** заполнить все доступное пространство по горизонтали и вертикали.
- Top** (в узле **Margins** узла **Layout Parameters**): `@dimen/activity_vertical_margin` — отделяет верх вертикального компонента **LinearLayout** от горизонтального компонента **LinearLayout** во второй строке GUI.
- Background** (узел **View**): `@android:color/holo_blue_bright` — один из заранее определенных цветовых ресурсов темы Android приложения.
- Padding Left/Right** (узел **View**): `@dimen/activity_horizontal_margin` — гарантирует, что компоненты, находящиеся в вертикальном компоненте **LinearLayout**, будут отделены от левого и правого края макета промежутком в `16dp`.
- Padding Top** (узел **View**): `@dimen/activity_vertical_margin` — гарантирует, что верхний компонент, находящийся в вертикальном компоненте **LinearLayout**, будет отделен от верхнего края макета промежутком в `16dp`.

Выделите вертикальный компонент **TextView** и задайте следующие значения свойств:

- Width** (узел **Layout Parameters**): `match_parent`.
- Height** (узел **Layout Parameters**): `wrap_content`.
- Gravity** (узел **Layout Parameters**): `fill_horizontal` — заставляет **TextView** заполнить всю ширину вертикального компонента **LinearLayout** (за вычетом внешних отступов).

- ❑ Gravity (узел TextView): center_horizontal — выравнивает текст по центру TextView.
- ❑ Text: @string/taggedSearches — создайте строковый ресурс со значением "Tagged Searches".
- ❑ Padding Top (узел View): @dimen/activity_vertical_margin — гарантирует, что верхний компонент, находящийся в вертикальном компоненте LinearLayout, будет отделен от верхнего края макета промежутком в 16dp.

Выделите компонент ListView и задайте следующие значения свойств.

- ❑ Width (узел Layout Parameters): match_parent.
- ❑ Height (узел Layout Parameters): 0dp — среда разработки рекомендует использовать это значение, если вы также задаете свойство Weight, чтобы компоненты можно было разместить более эффективно.
- ❑ Weight (узел Layout Parameters): 1.
- ❑ Gravity (узел Layout Parameters): fill — компонент ListView должен заполнять все доступное пространство по горизонтали и вертикали.
- ❑ Padding Top (узел View): @dimen/activity_vertical_margin — гарантирует, что верхний компонент, находящийся в вертикальном компоненте LinearLayout, будет отделен от верхнего края макета промежутком в 16dp.
- ❑ Top и Bottom (в узле Margins узла Layout Parameters): @dimen/tagged_searches_padding — создайте новый ресурс метрики tagged_searches_padding dimension, щелкнув на кнопке с многоточием справа от свойства Top. В диалоговом окне Resource Chooser нажмите New Dimension..., чтобы создать новый ресурс метрики. Задайте свойству Name значение tagged_searches_padding, а свойству Value — значение 8dp. Нажмите OK, выберите новый ресурс метрики и еще раз нажмите OK. Для свойства Bottom просто выберите новый ресурс метрики. Эти свойства гарантируют, что между TextView и верхним краем ListView, а также между нижним краем ListView и вертикальным компонентом LinearLayout останется зазор в 8dp.

4.4.4. Панель инструментов макетного редактора

Построение графического интерфейса MainActivity завершено. На панели инструментов макетного редактора (рис. 4.13) расположены кнопки, позволяющие получить представление о том, как будет выглядеть экран с другим размером или ориентацией. В частности, вы можете просмотреть миниатюрные изображения для многих размеров и ориентаций, щелкнув на стрелке рядом с кнопкой  и выбрав команду Preview Representative Sample или Preview All Screen Sizes. Рядом с каждой миниатюрой расположены кнопки + и -, с помощью которых можно увеличивать и уменьшать изображение. В табл. 4.2 показаны некоторые из кнопок на панели инструментов макетного редактора.

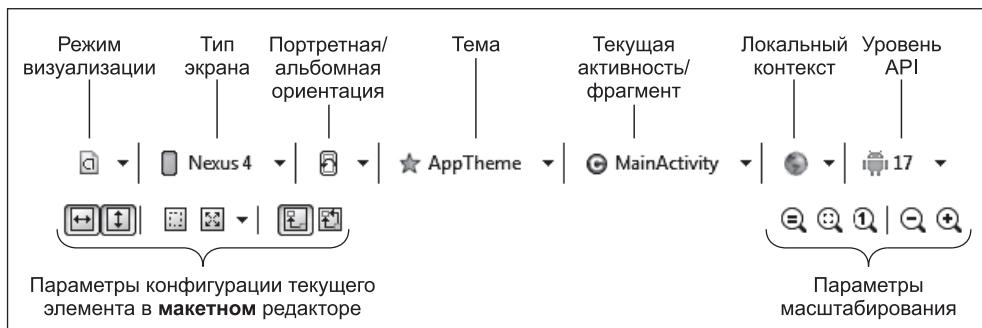


Рис. 4.13. Кнопки на панели инструментов

Таблица 4.2. Кнопки панели инструментов

Кнопка	Описание
Режим визуализации	Последовательный просмотр или одновременный просмотр дизайна для разных размеров экрана
Тип экрана	Android работает на множестве разных устройств, поэтому макетный редактор поддерживает разные конфигурации устройств, представляющие разные размеры экранов и разрешения, которые могут использоваться при построении графического интерфейса. В этой книге мы используем заранее определенные экраны Nexus 4, Nexus 7 и Nexus 10 (в зависимости от приложения). На рис. 4.13 выбран экран Nexus 4
Портретная/альбомная ориентация	Переключение области построения интерфейса между портретной и альбомной ориентацией
Тема	Используется для назначения темы графического интерфейса
Текущая активность/фрагмент	Класс Activity или Fragment, соответствующий создаваемому интерфейсу
Локальный контекст	В интернационализированных приложениях (см. раздел 2.8) позволяет выбрать конкретную локализацию — например, чтобы увидеть, как приложение будет выглядеть со строками разных языков
Уровень API	Целевой уровень API для приложения. С каждым новым уровнем API обычно появляются новые возможности графического интерфейса. В окне макетного редактора отображаются только возможности, доступные для выбранного уровня API

4.4.5. Макет варианта ListView: list_item.xml

При заполнении списка `ListView` данными необходимо указать формат, который будет применяться к каждому элементу списка. В нашем приложении в каждом элементе списка выводится имя тега (в формате `String`) для одного сохраненного поиска. Чтобы определить форматирование каждого элемента списка, мы создадим новый макет, который содержит только компонент `TextView` с соответствующим форматированием. Выполните следующие действия.

1. В окне `Package Explorer` откройте папку `res` проекта, щелкните правой кнопкой мыши на папке `layout` и выполните команду `New > Other...`; на экране появляется диалоговое окно `New`.
2. В категории `Android` выберите вариант `Android XML Layout File` и нажмите `Next >`; на экране появляется диалоговое окно, показанное на рис. 4.14. Настройте параметры файла так, как показано на иллюстрации. Новый макет хранится в файле `list_item.xml`, а корневым элементом макета является компонент `TextView`.
3. Нажмите `Finish`, чтобы создать файл.

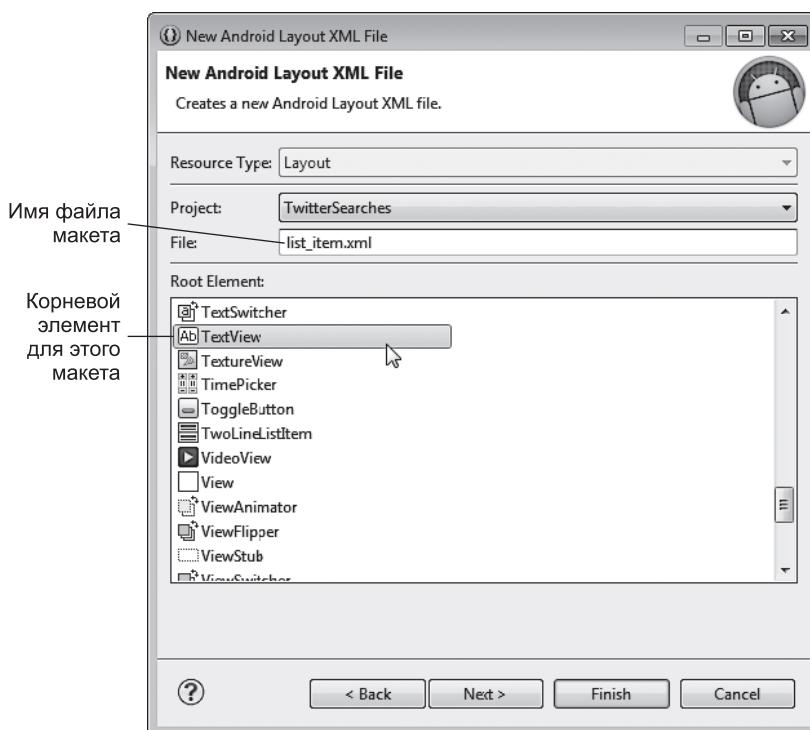


Рис. 4.14. Кнопки на панели инструментов

Среда разработки открывает новый макет в макетном редакторе. Выделите компонент `TextView` в окне `Outline` и задайте следующие свойства.

- `Id: @+id/textView` — идентификаторы компонентов GUI принято начинать со строчной буквы.
- `Height` (узел `Layout Parameters`): `?android:attr/listPreferredItemHeight` — заранее определенный ресурс Android, представляющий предпочтительную высоту элемента списка для правильной реакции на касания пользователя с минимальным риском выбора неправильного элемента.



ПРИМЕЧАНИЕ 4.4

В рекомендациях по дизайну приложений Android указано, что минимальный размер экранного объекта, который может выбираться касанием, составляет 48×48 dp. За дополнительной информацией об изменении размеров и интервалах между компонентами обращайтесь по адресу <http://developer.android.com/design/style/metrics-grids.html>.

- `Gravity` (узел `Layout Parameters`): `center_vertical` — компонент `TextView` должен выравниваться по центру внутри компонента `ListView`.
- `Text Appearance` (узел `TextView`): `?android:attr/textAppearanceMedium` — заранее определенный ресурс темы, который задает размер шрифта для текста среднего размера.

Элементы списка, в которых отображаются несколько объектов данных

Если в элементе списка должны отображаться несколько фрагментов данных, вам понадобится макет элемента списка, состоящий из нескольких элементов; при этом каждому элементу понадобится атрибут `android:id`.

Другие заранее определенные ресурсы Android

Существует много заранее определенных ресурсов Android наподобие тех, которые использовались для задания `Height` и `Text Appearance` для элемента списка. Полный список можно просмотреть по адресу

<http://developer.android.com/reference/android/R.attr.html>

Чтобы использовать такое значение в макете, используйте формат `?android:attr/имяРесурса`.

4.5. Построение класса `MainActivity`

В листингах 4.1–4.10 реализуется логика приложения Twitter Searches в классе `MainActivity`, расширяющем `ListActivity`. Код по умолчанию класса `MainActivity` включал метод `onCreateOptionsMenu`, который мы удалили, потому что он не используется в этом приложении, — метод `onCreateOptionsMenu` рассматривается в главе 5. В этом разделе предполагается, что необходимые ресурсы `String` будут создаваться там, где мы встречаем их в описании кода.

4.5.1. Команды package и import

В листинге 4.1 приведены команды `package` и `import` приложения. Команда `package` (вставленная в строке 4 средой разработки при создании проекта) указывает, что класс в этом файле является частью пакета `com.deitel.twittersearches`. В строках 6–26 импортируются классы и интерфейсы, используемые приложением.

Листинг 4.1. Команды package и import класса MainActivity

```
1 // MainActivity.java
2 // Управление поисковыми запросами Твиттера с ускоренным
3 // вызовом и отображением результатов в браузере устройства
4 package com.deitel.twittersearches;
5
6 import java.util.ArrayList;
7 import java.util.Collections;
8
9 import android.app.AlertDialog;
10 import android.app.ListActivity;
11 import android.content.Context;
12 import android.content.DialogInterface;
13 import android.content.Intent;
14 import android.content.SharedPreferences;
15 import android.net.Uri;
16 import android.os.Bundle;
17 import android.view.View;
18 import android.view.View.OnClickListener;
19 import android.view.inputmethod.InputMethodManager;
20 import android.widget.AdapterView;
21 import android.widget.AdapterView.OnItemClickListener;
22 import android.widget.AdapterView.OnItemLongClickListener;
23 import android.widget.ArrayAdapter;
24 import android.widget.EditText;
25 import android.widget.ImageButton;
26 import android.widget.TextView;
27
```

В строках 6–7 импортируются классы `ArrayList` и `Collections` из пакета `java.util`. Класс `ArrayList` используется для ведения списка тегов сохраненных запросов, а класс `Collections` сортирует теги, чтобы они выводились в алфавитном порядке. Из всех прочих команд `import` мы рассмотрим только те, которые относятся к возможностям, представленным в этой главе:

- ❑ Класс `AlertDialog` из пакета `android.app` (строка 9) используется для отображения диалоговых окон.
- ❑ Класс `ListActivity` из пакета `android.app` (строка 10) является суперклассом для класса `MainActivity`, предоставляющего список `ListView` приложения и методы для работы с ним.
- ❑ Класс `Context` из пакета `android.content` (строка 11) предоставляет доступ к информации о среде, в которой выполняется приложение, и предоставляет средства

для использования разных сервисов Android. Мы используем константу из этого класса, когда скрываем виртуальную клавиатуру из программы после того, как пользователь сохранит поисковый запрос.

- ❑ Класс `DialogInterface` из пакета `android.content` (строка 12) содержит вложенный интерфейс `OnClickListener`. Мы реализуем этот интерфейс для обработки событий, происходящих при касании кнопки в окне `AlertDialog`.
- ❑ Класс `Intent` из пакета `android.content` (строка 13) используется для создания объекта, задающего выполняемое действие и данные для его выполнения, — Android использует интенты для запуска соответствующих активностей. В нашем приложении этот класс используется для запуска браузера с результатами поиска и отображения окна выбора, в котором пользователь выбирает способ пересылки информации запроса.
- ❑ Класс `SharedPreferences` из пакета `android.content` (строка 14) используется для работы с парами «ключ/значение» из файлов, связанных с приложением.
- ❑ Класс `Uri` из пакета `android.net` (строка 15) обеспечивает преобразование URL в формат, необходимый для объекта `Intent`, запускающего браузер на устройстве.
- ❑ Класс `View` из пакета `android.view` (строка 17) используется в различных методах обработки событий; он представляет компонент GUI, с которым взаимодействовал пользователь для инициирования события.
- ❑ Класс `View` содержит вложенный интерфейс `OnClickListener` (строка 18). Мы реализуем этот интерфейс для обработки события, инициируемого при касании кнопки `ImageButton` для сохранения запроса.
- ❑ Класс `InputMethodManager` из пакета `android.view.inputmethod` (строка 19) позволяет скрыть виртуальную клавиатуру при сохранении запроса пользователем.
- ❑ Пакет `android.widget` (строки 20–26) содержит компоненты и макеты, используемые в графических интерфейсах Android. Класс `AdapterView` (строка 20) является базовым классом для `ListView`; он используется при настройке адаптера `ListView` (поставляющего данные элементов `ListView`). Вы реализуете интерфейс `AdapterView.OnItemClickListener` (строка 21) для обработки касаний элементов `ListView`. Интерфейс `AdapterView.OnItemLongClickListener` (строка 22) реализуется для обработки длинных нажатий на элементах `ListView`. Класс `ArrayAdapter` (строка 23) используется для привязки элементов к `ListView`. Класс `ImageButton` (строка 25) представляет кнопку с графическим изображением.

4.5.2. Расширение `ListActivity`

`MainActivity` (листинги 4.2–4.10) — единственный класс активности в приложении Twitter Searches. Когда вы создавали проект TwitterSearches, среда разработки сгенерировала `MainActivity` как субкласс `Activity` и предоставила заготовку переопределенного метода `onCreate`, который должен переопределяться каждым

субклассом `Activity`. Мы меняем суперкласс на `ListActivity` (листинг 4.2, строка 28). При внесении этого изменения среда разработки не опознает класс `ListActivity`, поэтому вы должны обновить свои команды `import`. В среде разработки для их обновления можно воспользоваться командой `Source > Organize Imports`. Eclipse подчеркивает все неопознанные имена классов и интерфейсов. Если навести указатель мыши на имя класса или интерфейса, появляется список возможных исправлений. Если имя известно среде разработки, она предлагает недостающие команды `import`, которые необходимо добавить в программу, — просто щелкните на имени, чтобы импортировать его.

Листинг 4.2. Класс `MainActivity` является субклассом `ListActivity`

```
28 public class MainActivity extends ListActivity  
29 {
```

4.5.3. Поля класса `MainActivity`

В листинге 4.3 содержатся переменные класса `MainActivity` (статические и переменные экземпляров). Строковая константа `SEARCHES` (строка 31) представляет имя файла, в котором будут храниться данные запросов. В строках 33–34 объявляются переменные компонентов `EditText`, в которых пользователь вводит запросы и теги. В строке 35 объявляется переменная `savedSearches` типа `SharedPreferences`; она будет использоваться для операций с парами «ключ/значение», представляющими сохраненные запросы пользователей.

В строке 36 объявляется коллекция `ArrayList<String>` для отсортированных имен тегов пользовательских запросов. В строке 37 объявляется коллекция `ArrayAdapter<String>`, использующая содержимое `ArrayList<String>` как источник данных, отображаемых в компоненте `ListView` класса `MainActivity`.



ХОРОШИЙ СТИЛЬ 4.1

Чтобы программу было проще читать и модифицировать, используйте константы `String` для представления имен файлов (и других строковых литералов), которые не нужно локализовать (а следовательно, определять в `strings.xml`).

Листинг 4.3. Поля класса `MainActivity`

```
30 // Имя XML-файла с данными SharedPreferences  
31 private static final String SEARCHES = "searches";  
32  
33 private EditText queryEditText; // EditText для ввода запроса  
34 private EditText tagEditText; // EditText для ввода тега  
35 private SharedPreferences savedSearches; // запросы пользователя  
36 private ArrayList<String> tags; // список тегов для запросов  
37 private ArrayAdapter<String> adapter; // связывает теги с ListView  
38
```

4.5.4. Переопределение метода активности onCreate

Метод `onCreate` (листинг 4.4) вызывается системой:

- ❑ при загрузке приложения;
- ❑ если процесс приложения уничтожается операционной системой во время нахождения в фоновом режиме, после чего приложение должно восстановиться;
- ❑ при любом изменении конфигурации — например, когда пользователь поворачивает устройство, открывает или закрывает физическую клавиатуру.

Метод инициализирует переменные экземпляра `Activity` и компоненты GUI — мы сделали его по возможности простым, чтобы приложение быстро загружалось. В строке 43 размещается обязательный вызов версии `onCreate` суперкласса. Как и в предыдущем приложении, вызов `setContentView` (строка 44) передает константу `R.layout.activity_main` для заполнения графического интерфейса из файла `activity_main.xml`.

Листинг 4.4. Переопределение метода onCreate класса Activity

```
39 // Вызывается при первом создании MainActivity
40 @Override
41 protected void onCreate(Bundle savedInstanceState)
42 {
43     super.onCreate(savedInstanceState);
44     setContentView(R.layout.activity_main);
45
46     // Получение ссылок на EditText
47     queryEditText = (EditText) findViewById(R.id.queryEditText);
48     tagEditText = (EditText) findViewById(R.id.tagEditText);
49
50     // Получение объекта SharedPreferences с сохраненными запросами
51     savedSearches = getSharedPreferences(SEARCHES, MODE_PRIVATE);
52
53     // Сохранение тегов в ArrayList и их сортировка
54     tags = new ArrayList<String>(savedSearches.getAll().keySet());
55     Collections.sort(tags, String.CASE_INSENSITIVE_ORDER);
56
57     // Создание объекта ArrayAdapter и привязка тегов к ListView
58     adapter = new ArrayAdapter<String>(this, R.layout.list_item, tags);
59     setListAdapter(adapter);
60
61     // Регистрация слушателя для сохранения запроса
62     ImageButton saveButton =
63         (ImageButton) findViewById(R.id.saveButton);
64     saveButton.setOnClickListener(saveButtonListener);
65
66     // Регистрация слушателя для поиска в Твиттере
67     getListView().setOnItemClickListener(itemClickListener);
68
69     // Назначение слушателя, позволяющего удалить или изменить запрос
70     getListView().setOnItemLongClickListener(itemLongClickListener);
71 } // Конец метода onCreate
```

Получение ссылок на EditText

В строках 47–48 мы получаем ссылки на компоненты `queryEditText` и `tagEditText` для инициализации соответствующих переменных экземпляров.

Получение объекта SharedPreferences

В строке 51 используется метод `getSharedPreferences` (унаследованный от класса `Context`) для получения объекта `SharedPreferences`, который может читать существующие пары «тег—запрос» из файла `SEARCHES` (если они есть, конечно). Первый аргумент обозначает имя файла с данными. Второй аргумент задает режим доступа к файлу; в нем может передаваться одно из следующих значений.

- `MODE_PRIVATE` — файл доступен только для этого приложения. Этот режим используется в большинстве случаев.
- `MODE_WORLD_READABLE` — любое приложение на устройстве может читать данные из файла.
- `MODE_WORLD_WRITEABLE` — любое приложение на устройстве может записывать данные в файл.

Эти константы могут объединяться поразрядным оператором ИЛИ (`|`). В нашем приложении объем читаемых данных невелик, поэтому загрузку запросов можно выполнить в `onCreate`.



COBET 4.1

Продолжительные операции с данными не должны выполняться в UI-потоке, иначе приложение выдаст диалоговое окно ANR (Application Not Responding) — как правило, если пользователь не мог взаимодействовать с приложением в течение 5 секунд. За информацией о построении приложений, быстро реагирующих на действия пользователя, обращайтесь по адресу <http://developer.android.com/guide/practices/design/responsiveness.html>.

Получение ключей из объекта SharedPreferences

Теги запросов стоит выводить в алфавитном порядке, чтобы пользователю было проще найти нужный запрос. Сначала строка 54 получает объекты `String`, представляющие ключи из объекта `SharedPreferences`, и сохраняет их в `tags` (`ArrayList<String>`). Метод `getAll` класса `SharedPreferences` возвращает все сохраненные запросы в формате `Map` (пакет `java.util`) — коллекции пар «ключ/значение». Затем мы вызываем для этого объекта метод `keySet`, чтобы получить все ключи в виде `Set` (пакет `java.util`) — коллекции уникальных значений. Результат используется для инициализации `tags`.

Сортировка тегов

В строке 55 метод `Collections.sort` используется для сортировки тегов. Так как пользователь может вводить теги в произвольной комбинации символов верхнего и нижнего регистра, мы выполняем сортировку без учета регистра, для чего во

втором аргументе `Collections.sort` передается заранее определенный объект `String.CASE_INSENSITIVE_ORDER` (реализация `Comparator<String>`).

Использование `ArrayAdapter` для заполнения `ListView`

Чтобы вывести данные в `ListView`, мы создаем новый объект `ArrayAdapter<String>` (строка 58), который связывает содержимое `tags` с компонентами `TextView`, отображаемыми в компоненте `ListView` активности `MainActivity`. Конструктор `ArrayAdapter<String>` получает следующие аргументы.

- ❑ Контекст (`this`), в котором отображается `ListView`; в данном случае это объект `MainActivity`.
- ❑ Идентификатор ресурса (`R.layout.list_item`) макета, который должен использоваться для отображения элементов в `ListView`.
- ❑ Контейнер `List<String>` с данными; коллекция `tags` относится к типу `ArrayList<String>`, который реализует интерфейс `List<String>`, поэтому `tags` является частным случаем `List<String>`.

В строке 59 унаследованный от `ListActivity` метод `setListAdapter` используется для связывания `ListView` с `ArrayAdapter`, чтобы компонент `ListView` мог вывести данные.

Регистрация слушателей для `saveButton` и `ListView`

В строках 62–63 приложение получает ссылку на компонент `saveButton`, а строка 64 регистрирует его *слушателя* — переменная экземпляра `saveButtonListener` содержит ссылку на объект анонимного внутреннего класса, реализующий интерфейс `OnClickListener` (листинг 4.5). Страна 67 использует унаследованный от `ListActivity` метод `getListView` для получения ссылки на компонент `ListView` текущей активности, после чего регистрирует слушателя `OnItemClickListener` компонента `ListView` — переменная экземпляра `itemClickListener` содержит ссылку на объект анонимного внутреннего класса, реализующий этот интерфейс (листинг 4.8). Аналогичным образом строка 70 регистрирует слушателя `OnItemLongClickListener` компонента `ListView` — переменная экземпляра `itemLongClickListener` содержит ссылку на объект анонимного внутреннего класса, реализующий этот интерфейс (листинг 4.9).

4.5.5. Анонимный внутренний класс, реализующий интерфейс `OnClickListener` для сохранения нового или измененного запроса

В листинге 4.5 объявляется и инициализируется переменная экземпляра `saveButtonListener`, в которой хранится ссылка на объект анонимного внутреннего класса, реализующего интерфейс `OnClickListener`. Страна 64 (листинг 4.4) регистрирует `saveButtonListener` как обработчик события `saveButton`. В строках 76–109 определяется метод `onClick` интерфейса `OnClickListener`. Если пользователь ввел запрос *и* тег (строки 80–81), то в строках 83–84 вызывается метод `addTaggedSearch`

(листинг 4.7) для сохранения пары «тег—запрос», а строки 85–86 очищают два компонента EditText. Строки 88–90 скрывают виртуальную клавиатуру.

Листинг 4.5. Анонимный внутренний класс, реализующий интерфейс OnClickListener кнопки saveButton для сохранения нового или измененного запроса

```

73 // saveButtonListener сохраняет пару "тег-запрос" в SharedPreferences
74 public OnClickListener saveButtonListener = new OnClickListener()
75 {
76     @Override
77     public void onClick(View v)
78     {
79         // Создаем тег, если в queryEditText и tagEditText есть данные
80         if (queryEditText.getText().length() > 0 &&
81             tagEditText.getText().length() > 0)
82         {
83             addTaggedSearch(queryEditText.getText().toString(),
84                             tagEditText.getText().toString());
85             queryEditText.setText(""); // Очистка queryEditText
86             tagEditText.setText(""); // Очистка tagEditText
87             ((InputMethodManager) getSystemService(
88                 Context.INPUT_METHOD_SERVICE)).hideSoftInputFromWindow(
89                 tagEditText.getWindowToken(), 0);
90
91         }
92         else // Вывод сообщения с предложением ввести запрос и тег
93         {
94             // Создание объекта AlertDialog Builder
95             AlertDialog.Builder builder =
96                 new AlertDialog.Builder(MainActivity.this);
97
98             // Назначение заголовка и сообщения диалогового окна
99             builder.setMessage(R.string.missingMessage);
100
101            // Кнопка OK просто закрывает диалоговое окно
102            builder.setPositiveButton(R.string.OK, null);
103
104            // Создание объекта AlertDialog на базе AlertDialog.Builder
105            AlertDialog errorDialog = builder.create();
106            errorDialog.show(); // Вывод модального диалогового окна
107        }
108    } // Конец метода onClick
109 }; // Конец анонимного внутреннего класса OnClickListener
110

```

Конфигурация AlertDialog

Если пользователь не ввел запрос и/или тег, в строках 92–108 выводится окно AlertDialog с предложением ввести оба значения. Объект AlertDialog.Builder (строки 95–96) упрощает создание и настройку AlertDialog. В аргументе конструктора передается контекст, в котором должно отображаться диалоговое окно, — в данном случае это объект MainActivity, обозначенный ссылкой this. Чтобы использовать

this в анонимном внутреннем классе, вы должны полностью уточнить this именем внешнего класса. В строке 99 сообщение диалогового окна ("Enter both a Twitter search query and a tag") задается строковым ресурсом R.string.missingMessage.



ПРИМЕЧАНИЕ 4.5

Заголовок AlertDialog (который выводится над сообщением диалогового окна) можно назначить методом setTitle класса AlertDialog.Builder. Согласно рекомендациям по дизайну приложений Android (<http://developer.android.com/design/building-blocks/dialogs.html>), для большинства диалоговых окон заголовки не обязательны. Диалоговое окно должно отображать заголовок для «рискованных операций, сопряженных с возможным риском потери данных, разрывом связи, дополнительными затратами и т. д.». Кроме того, диалоговые окна, в которых отображаются списки вариантов, используют заголовки для описания предназначения диалогового окна.

Добавление строковых ресурсов в strings.xml

Чтобы создать строковый ресурс (такой, как R.string.missingMessage), откройте файл strings.xml из папки res/values. Среда разработки отображает этот файл в редакторе ресурсов, состоящем из двух вкладок — Resources и strings.xml. На вкладке Resources кнопка Add... открывает диалоговое окно на рис. 4.15. Если выбрать тип String и щелкнуть на кнопке OK, открываются текстовые поля Name и Value, в которых можно ввести имя нового строкового ресурса (например, missingMessage) и его значение. Сохраните файл strings.xml после внесения изменений. На вкладке Resources также можно выбрать существующий строковый ресурс, чтобы изменить его имя и значение.

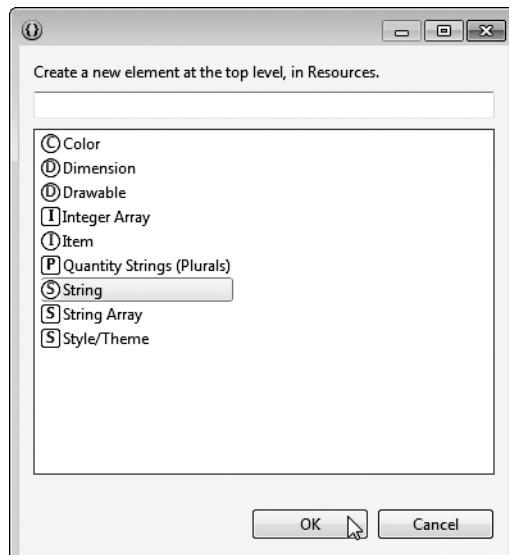


Рис. 4.15. Добавление строкового ресурса

Назначение позитивной кнопки AlertDialog

В этом окне `AlertDialog` понадобится только одна кнопка, при помощи которой пользователь подтверждает сообщение. Эта кнопка назначается *позитивной кнопкой* диалогового окна (рис. 4.14, строка 102) — касание этой кнопки означает, что пользователь подтверждает сообщение, выведенное в окне. Методу `setPositiveButton` передается надпись на кнопке (заданная строковым ресурсом `R.string.OK`) и ссылка на обработчик события кнопки. Нашему диалоговому окну обрабатывать это событие не нужно, поэтому вместо обработчика события передается `null`. Когда пользователь касается кнопки, диалоговое окно просто исчезает с экрана.

Создание и отображение AlertDialog

Объект `AlertDialog` создается методом `create` класса `AlertDialog.Builder` (строка 105), а модальное диалоговое окно отображается методом `show` класса `AlertDialog` (строка 106).

4.5.6. Метод addTaggedSearch

Обработчик события из листинга 4.4 вызывает метод `addTaggedSearch` класса `MainActivity` (листинг 4.6) для добавления нового запроса в `savedSearches` или изменения существующего запроса.

Листинг 4.6. Метод addTaggedSearch класса MainActivity

```

111     // Добавление нового запроса в файл и обновление всех кнопок
112     private void addTaggedSearch(String query, String tag)
113     {
114         // Получение объекта Sharedpreferences.Editor для сохранения новой пары
115         Sharedpreferences.Editor preferencesEditor = savedSearches.edit();
116         preferencesEditor.putString(tag, query); // Сохранение текущего запроса
117         preferencesEditor.apply(); // Сохранение обновленных данных
118
119         // Если тег только что создан – добавить и отсортировать теги
120         if (!tags.contains(tag))
121         {
122             tags.add(tag); // Добавление нового тега
123             Collections.sort(tags, String.CASE_INSENSITIVE_ORDER);
124             adapter.notifyDataSetChanged(); // Повторное связывание с ListView
125         }
126     }
127 }
```

Изменение содержимого объекта Sharedpreferences

Чтобы изменить содержимое объекта `Sharedpreferences`, необходимо сначала вызвать для него метод `edit`, чтобы получить объект `Sharedpreferences.Editor` (строка 115). В этот объект можно добавлять пары «ключ/значение», удалять из него пары «ключ/значение», а также с его помощью изменять значение, связанное

с конкретным ключом в файле `SharedPreferences`. В строке 116 метод `putString` класса `SharedPreferences.Editor` вызывается для сохранения тега (ключ) и запроса (соответствующее значение) — если тег уже существует в `SharedPreferences`, происходит обновление связанного с ним значения. Стока 117 закрепляет изменения вызовом метода `apply` класса `SharedPreferences.Editor` для внесения изменений в файл.

Оповещение `ArrayAdapter` об изменении данных

При добавлении нового запроса необходимо обновить компонент `ListView`, чтобы запрос появился в списке. Строки 120–125 определяют, был ли создан новый запрос. Если запрос был создан, то строки 122–123 добавляют его тег в коллекцию `tags`, после чего сортируют `tags`. Стока 124 вызывает метод `notifyDataSetChanged` объекта-адаптера `ArrayAdapter`, чтобы сообщить об изменении данных `tags`. Адаптер оповещает `ListView` о необходимости обновить отображаемый список.

4.5.7. Анонимный внутренний класс, реализующий интерфейс `OnItemClickListener` класса `ListView` для отображения результатов поиска

В листинге 4.7 объявляется и инициализируется переменная экземпляра `itemClickListener`, которая содержит ссылку на объект анонимного внутреннего класса, реализующий интерфейс `OnItemClickListener`. Стока 67 (листинг 4.4) регистрирует `ItemClickListener` в качестве обработчика события `ListView`, реагирующего на касания элементов списка `ListView`. В строках 131–145 переопределяется метод `onItemClick` интерфейса `OnItemClickListener`. Аргументы метода:

- ❑ Объект `AdapterView`, в котором было сделано касание. Знак `?` в `AdapterView<?>` является метасимволом параметризованных типов Java, который указывает, что метод `onItemClick` может получать объект `AdapterView` для отображения произвольного типа данных — в данном случае `ListView<String>`.
- ❑ Компонент `View`, которого коснулся пользователь в `AdapterView`, — в данном случае компонент `TextView` для вывода тега.
- ❑ Индекс элемента списка, которого коснулся пользователь (индексы начинаются с нуля!)
- ❑ Идентификатор строки для элемента списка, которого коснулся пользователь, — используется прежде всего для информации, загруженной из базы данных (см. главу 8).

Листинг 4.7. Анонимный внутренний класс, реализующий интерфейс `OnItemClickListener` компонента `ListView` для вывода результатов поиска

```
128 // itemClickListener запускает браузер для вывода результатов
129 OnItemClickListener itemClickListener = new OnItemClickListener()
130 {
```

```
131     @Override
132     public void onItemClick(AdapterView<?> parent, View view,
133             int position, long id)
134     {
135         // Получение строки запроса и создание URL-адреса для запроса
136         String tag = ((TextView) view).getText().toString();
137         String urlString = getString(R.string.searchURL) +
138             Uri.encode(savedSearches.getString(tag, ""), "UTF-8");
139
140         // Создание интента для запуска браузера
141         Intent webIntent = new Intent(Intent.ACTION_VIEW,
142             Uri.parse(urlString));
143
144         startActivity(webIntent); // Запуск браузера для просмотра
145             // результатов
146     }
147 } // Конец объявления itemClickListener
```

Получение строковых ресурсов

Строка 136 получает текст компонента `View`, которого пользователь коснулся в `ListView`. В строках 137–138 создается строка, содержащая URL-адрес для поиска в Твиттере с выполняемым запросом. Сначала строка 137 вызывает унаследованный от `Activity` метод `getString` с одним аргументом для получения строкового ресурса с именем `searchURL`, который содержит URL-адрес страницы поиска в Твиттере:

http://mobile.twitter.com/search/

Этот ресурс, как и все остальные строковые ресурсы в этом приложении, следует добавить в файл `strings.xml`.

Получение строк из объекта `SharedPreferences`

Результат из строки 138 присоединяется к URL-адресу поиска для завершения `urlString`. Метод `getString` класса `SharedPreferences` возвращает запрос, связанный с тегом. Если тег еще не существует, возвращается второй аргумент (" " в данном случае). Стока 138 передает запрос методу `encode` класса `Uri`, который экранирует все специальные символы в URL (например, ?, /, : и т. д.) и возвращает так называемую строку в кодировке `URL`. Важно проследить за тем, чтобы веб-сервер, получающий запрос, смог нормально разобрать URL-адрес для получения поискового запроса.

Создание интента для запуска браузера

В строках 141–142 создается новый объект `Intent`, который будет использоваться для запуска браузера на устройстве и вывода результатов поиска. Интенты могут использоваться для запуска других активностей в том же или в других приложениях. В первом аргументе конструктора `Intent` передается константа с описанием выполняемого действия. `Intent.ACTION_VIEW` означает, что мы хотим отобразить представление данных. В классе `Intent` определяются и другие константы для таких действий, как поиск, выбор, отправка и воспроизведение. Во втором аргументе (строка 142) передается объект `Uri` (Uniform Resource Identifier), представляющий

данные, с которыми выполняется действие. Метод `parse` класса `Uri` преобразует строку, представляющую URL (Uniform Resource Locator), в объект `Uri`.

Запуск активности для интента

Строка 144 передает объект `Intent` унаследованному от `Activity` методу `startActivity`, который запускает указанное действие с указанными данными. В нашем случае, поскольку мы выбрали просмотр URI, интент запускает на устройстве браузер для отображения соответствующей веб-страницы. На странице выводится результат поиска в Твиттере.

4.5.8. Анонимный внутренний класс, реализующий интерфейс `OnItemClickListener` класса `ListView` для пересылки, изменения и удаления запросов

В листинге 4.8 объявляется и инициализируется переменная экземпляра `itemLongClickListener`, которая содержит ссылку на объект анонимного внутреннего класса, реализующий интерфейс `OnItemClickListener`. Стока 70 (листинг 4.4) регистрирует `itemLongClickListener` в качестве обработчика события `ListView`, реагирующего на длинные нажатия на элементах списка `ListView`. В строках 153–210 переопределяется метод `onItemLongClick` интерфейса `OnItemClickListener`.

Листинг 4.8. Анонимный внутренний класс, реализующий интерфейс `OnItemClickListener` компонента `ListView` для пересылки, изменения и удаления запросов

```
148 // itemLongClickListener отображает диалоговое окно для удаления
149 // или изменения сохраненного запроса
150 OnItemClickListener itemLongClickListener =
151     new OnItemClickListener()
152 {
153     @Override
154     public boolean onItemLongClick(AdapterView<?> parent, View view,
155         int position, long id)
156     {
157         // Получение тега, на котором было сделано длинное нажатие
158         final String tag = ((TextView) view).getText().toString();
159
160         // Создание нового объекта AlertDialog
161         AlertDialog.Builder builder =
162             new AlertDialog.Builder(MainActivity.this);
163
164         // Создание заголовка AlertDialog
165         builder.setTitle(
166             getString(R.string.shareEditDeleteTitle, tag));
167
168         // Назначение списка вариантов для вывода в диалоговом окне
169         builder.setItems(R.array.dialog_items,
170             new DialogInterface.OnClickListener()
```

```

171
172     {
173         // Реагирует на действие пользователя пересылкой,
174         // изменением или удалением сохраненного запроса
175         @Override
176         public void onClick(DialogInterface dialog, int which)
177         {
178             switch (which)
179             {
180                 case 0: // Пересылка
181                     shareSearch(tag);
182                     break;
183                 case 1: // Изменение
184                     // Заполнение EditText тегом и запросом
185                     tagEditText.setText(tag);
186                     queryEditText.setText(
187                         savedSearches.getString(tag, ""));
188                     break;
189                 case 2: // Удаление
190                     deleteSearch(tag);
191                     break;
192             }
193         } // Конец DialogInterface.OnClickListener
194     }; // Конец builder.setItems
195
196     // Назначение негативной кнопки AlertDialog
197     builder.setNegativeButton(getString(R.string.cancel),
198         new DialogInterface.OnClickListener()
199         {
200             // вызывается при нажатии кнопки "Cancel"
201             public void onClick(DialogInterface dialog, int id)
202             {
203                 dialog.cancel(); // Закрытие окна AlertDialog
204             }
205         });
206     ); // Конец setNegativeButton
207
208     builder.create().show(); // Отображение AlertDialog
209     return true;
210 } // Конец метода onItemLongClick
211 }; // Конец объявления OnItemLongClickListener
212

```

Локальные переменные для анонимных внутренних классов

Строка 158 получает текст элемента списка, на котором пользователь сделал длинное нажатие, и присваивает его локальной переменной `tag` со спецификатором `final`. Любые локальные переменные или параметры методов, которые будут использоваться в анонимном внутреннем классе, должны быть объявлены со спецификатором `final`.

Окно AlertDialog для вывода списка

В строках 161–166 создается объект `AlertDialog.Builder`, а заголовком диалогового окна назначается отформатированная строка, в которой значение `tag` заменяет

спецификатор формата в ресурсе `R.string.shareEditDeleteTitle` (представляющем строку "Share, Edit or Delete the search tagged as \"%s\"").

Строка 166 вызывает унаследованный от `Activity` метод `getString`, который получает несколько аргументов. Первый аргумент содержит идентификатор строкового ресурса, представляющего форматную строку, а в остальных передаются значения, которые должны заменить спецификаторы формата в форматной строке. Кроме кнопок, окно `AlertDialog` может вывести список вариантов в компоненте `ListView`. Строки 169–194 указывают, что в окне должно отображаться содержимое массива строк `R.array.dialog_items` (строки "Share", "Edit" и "Delete"); кроме того, в них определяется анонимный внутренний класс для обработки касаний к элементам списка.

Добавление ресурса массива строк в strings.xml

Массив строк определяется в виде ресурса массива в файле `strings.xml`. Чтобы добавить ресурс массива строк в файл `strings.xml`, выполните следующие действия.

1. Выполните действия из раздела 4.5.5, чтобы добавить строковый ресурс, но выберите пункт `String Array` вместо `String` в диалоговом окне на рис. 4.15 и нажмите `OK`.
2. Введите в поле `Name` имя массива (`dialog_items`).
3. Выделите массив в списке ресурсов в левой стороне редактора ресурсов.
4. Щелкните на кнопке `Add...` и на кнопке `OK`, чтобы добавить в массив новый элемент.
5. Укажите значение нового элемента в текстовом поле `Value`.

Выполните эти действия для команд `Share`, `Edit` и `Delete` (в этом порядке), после чего сохраните файл `strings.xml`.

Обработчик событий для списка элементов диалогового окна

Анонимный внутренний класс в строках 170–193 определяет, какой элемент был выбран пользователем из списка диалогового окна, и выполняет соответствующее действие. Если пользователь выбрал команду `Share`, то вызывается функция `shareSearch` (строка 180). Если пользователь выбрал команду `Edit`, то в строках 184–186 выводится запрос и тег в компонентах `EditText`. Если пользователь выбрал команду `Delete`, то вызывается `deleteSearch` (строка 189).

Настройка негативной кнопки и отображение диалогового окна

В строках 197–206 настраивается негативная кнопка диалогового окна, которая закрывает диалоговое окно, если пользователь отказался от пересылки, изменения или удаления запроса. Стока 208 создает и отображает диалоговое окно.

4.5.9. Метод shareSearch

Метод `shareSearch` (листинг 4.9) вызывается обработчиком события из листинга 4.8, когда пользователь решает переслать запрос. В строках 217–218 создается объект

`String`, представляющий пересылаемый запрос. Строки 221–227 создают и настраивают объект `Intent` для отправки URL-адреса с использованием активности, которая может обработать `Intent.ACTION_SEND` (строка 222).

Листинг 4.9. Метод shareSearch класса MainActivity

```

213 // Выбор приложения для пересылки URL-адреса сохраненного запроса
214 private void shareSearch(String tag)
215 {
216     // Создание URL-адреса, представляющего запрос
217     String urlString = getString(R.string.searchURL) +
218         Uri.encode(savedSearches.getString(tag, ""), "UTF-8");
219
220     // Создание объекта Intent для пересылки urlString
221     Intent shareIntent = new Intent();
222     shareIntent.setAction(Intent.ACTION_SEND);
223     shareIntent.putExtra(Intent.EXTRA_SUBJECT,
224         getString(R.string.shareSubject));
225     shareIntent.putExtra(Intent.EXTRA_TEXT,
226         getString(R.string.shareMessage, urlString));
227     shareIntent.setType("text/plain");
228
229     // Вывод списка приложений с возможностью пересылки текста
230     startActivity(Intent.createChooser(shareIntent,
231         getString(R.string.shareSearch)));
232 }
233

```

Включение дополнительной информации в интент

Объект `Intent` содержит контейнер `Bundle` с *дополнениями* (*extras*), которые должны передаваться объекту `Activity`, обрабатывающему интент. Например, активность электронной почты может получать дополнения, представляющие собой тему сообщения, адреса CC и BCC, а также текст сообщения. В строках 223–226 метод `putExtra` класса `Intent` используется для добавления пары «ключ/значение» в контейнер `Bundle`, связанный с `Intent`. В первом аргументе метода содержится ключ `String`, представляющий назначение дополнения, а во втором — дополнительные данные. Дополнения могут быть значениями примитивных типов, массивами примитивных типов, объектами `Bundle` и не только — за полным списком перегруженных версий `putExtra` обращайтесь к документации класса `Intent`.

Дополнение в строках 223–224 задает тему сообщения в виде строкового ресурса `R.string.shareSubject ("Twitter search that might interest you")`. Для активности, не использующей тему (например, при публикации в социальной сети), это дополнение игнорируется. Дополнение в строках 225–226 представляет пересылаемый текст — отформатированная строка, в которой значение `urlString` подставляется в строковый ресурс `R.string.shareMessage ("Check out the results of this Twitter search: %s")`. Стока 227 назначает интенту тип MIME `text/plain` — такие данные могут обрабатываться любой активностью, способной отправлять простые текстовые сообщения.

Выбор интента

Чтобы вывести окно выбора интента, показанное на рис. 4.8, *a*, мы передаем объект `Intent` и строку заголовка статическому методу `createChooser` класса `Intent` (строка 230). Ресурс `R.string.shareSearch ("Share Search to:")` используется как заголовок окна выбора интента. Важно задать этот заголовок, чтобы напомнить пользователю о выборе соответствующей активности. Вы не можете управлять ни приложениями, установленными на телефоне пользователя, ни фильтрами интентов, которые могут запускать эти приложения, поэтому в окне выбора могут появиться несовместимые активности. Метод `createChooser` возвращает объект `Intent`, который передается `startActivity` для отображения окна выбора.

4.5.10. Метод `deleteSearch`

Обработчик события в листинге 4.8 вызывает метод `deleteSearch` (листинг 4.10), когда пользователь выполняет длинное нажатие на теге и выбирает команду `Delete`. Перед удалением запроса приложение выводит окно `AlertDialog` для подтверждения операции. В строках 241–242 в заголовке диалогового окна назначается отформатированная строка, в которой значение `tag` заменяет форматный спецификатор в строковом ресурсе `R.string.confirmMessage ("Are you sure you want to delete the search \"%s\"?")`. В строках 245–254 настраивается негативная кнопка для закрытия диалогового окна. Строковый ресурс `R.string.cancel` представляет текст `"Cancel"`. В строках 257–275 настраивается позитивная кнопка для удаления запроса. Ресурс `R.string.delete` представляет текст `"Delete"`. В строке 263 тег удаляется из коллекции `tags`, а строки 266–269 используют объект `SharedPreferences.Editor` для удаления запроса из объекта `SharedPreferences` приложения. Затем строка 272 оповещает `ArrayAdapter` об изменении используемых данных, чтобы компонент `ListView` мог обновить свой список.

Листинг 4.10. Метод `deleteSearch` класса `MainActivity`

```
234 // Удаление запроса после подтверждения операции пользователем
235 private void deleteSearch(final String tag)
236 {
237     // Создание нового объекта AlertDialog
238     AlertDialog.Builder confirmBuilder = new AlertDialog.Builder(this);
239
240     // Назначение сообщения AlertDialog
241     confirmBuilder.setMessage(
242         getString(R.string.confirmMessage, tag));
243
244     // Назначение негативной кнопки AlertDialog
245     confirmBuilder.setNegativeButton(getString(R.string.cancel),
246         new DialogInterface.OnClickListener()
247     {
248         // Вызывается при нажатии кнопки "Cancel"
249         public void onClick(DialogInterface dialog, int id)
250     {
```

```

251             dialog.cancel(); // Закрытие диалогового окна
252         }
253     }
254 ); // Конец setNegativeButton
255
256 // Назначение позитивной кнопки AlertDialog
257 confirmBuilder.setPositiveButton(getString(R.string.delete),
258     new DialogInterface.OnClickListener()
259     {
260         // Вызывается при нажатии кнопки "Cancel"
261         public void onClick(DialogInterface dialog, int id)
262         {
263             tags.remove(tag); // Удаление тега из коллекции tags
264
265             // Получение SharedPreferences.Editor для удаления запроса
266             SharedPreferences.Editor preferencesEditor =
267                 savedSearches.edit();
268             preferencesEditor.remove(tag); // Удаление запроса
269             preferencesEditor.apply(); // Сохранение изменений
270
271             // Повторное связывание для вывода обновленного списка
272             adapter.notifyDataSetChanged();
273         }
274     } // Конец OnClickListener
275 ); // Конец setPositiveButton
276
277     confirmBuilder.create().show(); // Вывод AlertDialog
278 } // Конец метода deleteSearch
279 } // Конец класса MainActivity

```

4.6. AndroidManifest.xml

В разделе 3.6 мы внесли два изменения в файл `AndroidManifest.xml`.

- ❑ Первое указывает, что приложение Tip Calculator поддерживает только портретную ориентацию.
- ❑ Второе обеспечивает отображение виртуальной клавиатуры в начале работы приложения, чтобы пользователь мог немедленно ввести сумму счета в приложении Tip Calculator.

Это приложение поддерживает как портретную, так и альбомную ориентацию. Никакие дополнительные изменения в этом случае не требуются, поскольку все приложения по умолчанию поддерживают обе ориентации.

Как правило, пользователь будет запускать приложение Twitter Searches для выполнения одного из сохраненных запросов. Если первым компонентом GUI является компонент `EditText`, Android передает этому компоненту фокус при загрузке приложения. Как вы уже знаете, при получении фокуса компонентом `EditText` отображается соответствующая виртуальная клавиатура (при отсутствии аппаратной

клавиатуры у устройства). В этом приложении виртуальная клавиатура не должна отображаться, пока пользователь не коснется одного из компонентов `EditText`. Для этого выполните действия из раздела 3.6 по настройке параметра `Window soft input mode`, но задайте ему значение `stateAlwaysHidden`.

4.7. Резюме

В этой главе мы создали приложение Twitter Searches. На первом этапе был создан графический интерфейс пользователя (GUI). Вы познакомились с компонентом `ListView`, который позволяет вывести список произвольной длины с возможностью прокрутки. С каждым запросом связывается элемент списка `ListView`, после касания которой запрос передается браузеру устройства. Вы также узнали о том, как создать строковые ресурсы в коде Java.

Поисковые пары «тег—запрос» были сохранены в файле `SharedPreferences`, связанном с приложением. Мы рассмотрели скрытие виртуальной клавиатуры на программном уровне. Объект `SharedPreferences.Editor` использовался для сохранения, изменения и удаления значений из файла `SharedPreferences`. В ответ на касание пользователем запроса приложение загружало `Uri`-ссылки в окно браузера устройства путем создания нового интента с дальнейшей передачей методу `startActivity` из класса `Context`. Мы также использовали объект `Intent` для отображения окна выбора, в котором пользователь может выбрать активность для пересылки запроса.

Объекты `AlertDialog.Builder` были применены для конфигурирования и создания окон `AlertDialog`, в которых отображаются сообщения для пользователя. И наконец, был рассмотрен файл `AndroidManifest.xml` и показано, как настроить приложение, чтобы виртуальная клавиатура не отображалась после запуска приложения.

В главе 5 будет создано приложение Flag Quiz, на экране которого отображается флаг определенной страны, название которой предстоит выбрать пользователю из трех, шести или девяти вариантов. С помощью меню и флагжков вы сможете настроить приложение, ограничив количество отображаемых флагов и стран определенными регионами мира.

5

Приложение Flag Quiz

Фрагменты, меню, AssetManager, анимация с переходами, обработчики, явные интенты и макеты для разных ориентаций устройства

В этой главе...

- Использование фрагментов для эффективной организации пространства в графическом интерфейсе Activity на телефонах и планшетах
- Отображение меню на панели действий для настройки параметров приложения
- Использование объектов PreferenceFragment для автоматического управления настройками пользователя
- Использование папок assets для организации графических ресурсов и работы с ними средствами AssetManager
- Определение анимаций и их применение к View
- Планирование выполняемых в будущем действий с помощью Handler
- Использование объектов Toast для кратковременного отображения сообщений
- Запуск конкретной активности с помощью явного интента
- Коллекции из пакета java.util
- Определение макетов для разных ориентаций устройства
- Использование механизма регистрации Android для регистрации сообщений об ошибках

5.1. Введение

Приложение Flag Quiz проверяет знание пользователем 10 флагов различных стран (рис. 5.1). После запуска приложения на экране появляется изображение флага и три кнопки с вариантами возможного ответа. Один из вариантов правильный, а остальные — неправильные — выбираются случайным образом без повторений. Приложение выводит информацию о ходе игры, отображая номер вопроса (из десяти возможных) в компоненте `TextView`, находящемся над изображением текущего флага. Как будет показано ниже, приложение также позволяет управлять сложностью игры: пользователь может выбрать между тремя, шестью или девятью вариантами ответа, а также выбором регионов, которые должны быть включены в опрос.

Формат отображения вариантов зависит от устройства, на котором выполняется приложение, и ориентации экрана — приложение поддерживает портретную ориентацию на любом устройстве, но альбомная ориентация поддерживается только на планшетах. В портретной ориентации приложение выводит на панели действий меню с командой `Settings`. Когда пользователь выбирает эту команду, приложение выводит активность для определения количества вариантов ответа и регионов. На планшете в альбомной ориентации (рис. 5.2) используется другой макет, при котором настройки приложения отображаются в левой части экрана, а флаги — справа.



Рис. 5.1. Приложение Flag Quiz на смартфоне в портретной ориентации

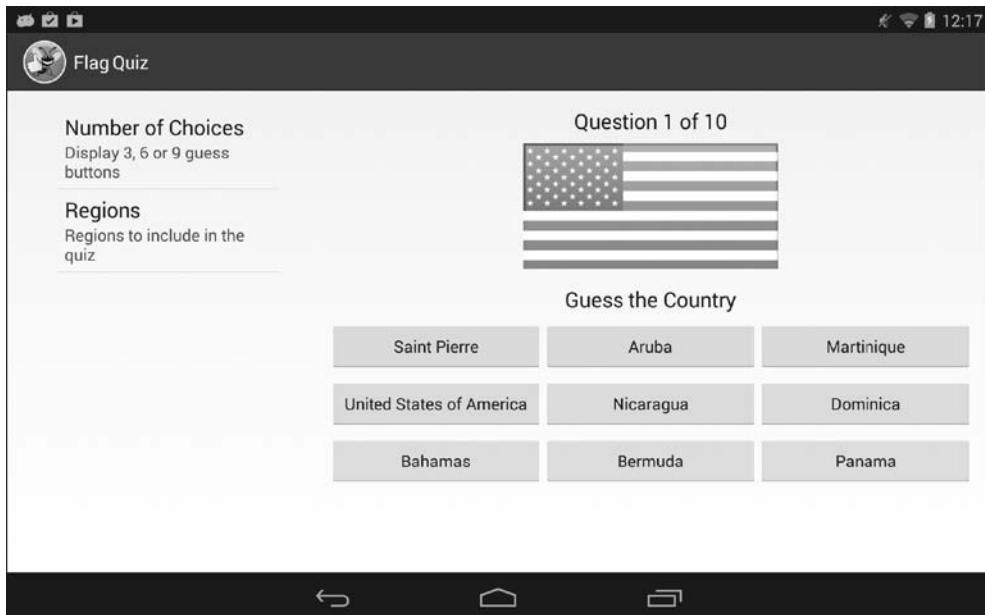


Рис. 5.2. Приложение Flag Quiz на планшете в альбомной ориентации

Начнем с тестового запуска, после чего будет приведен обзор технологий, задействованных при его создании. Затем мы построим графический интерфейс приложения. Глава завершается анализом полного исходного кода приложения с более подробным анализом новых возможностей.

5.2. Тестирование приложения Flag Quiz

Откройте среду разработки и импортируйте проект приложения Flag Quiz. Приложение можно протестировать на виртуальном устройстве (AVD) телефона или планшета или на физическом устройстве. Экранные снимки в этой главе сделаны на телефоне Nexus 4 и планшете Nexus 7.

5.2.1. Импорт и запуск приложения

Выполните следующие действия, чтобы импортировать приложение в IDE:

1. Откройте диалоговое окно Import. Выполните команду File > Import....
2. Импортируйте проект приложения Flag Quiz. В диалоговом окне Import раскройте узел General и выберите параметр Existing Projects into Workspace. Затем нажмите Next > для перехода к шагу Import Projects. Убедитесь в том, что установлен

переключатель **Select root directory**, и нажмите **Browse...** Найдите папку **FlagQuiz** в папке примеров, выделите ее и нажмите **OK**. Убедитесь в том, что переключатель **Copy projects into workspace** не установлен. Нажмите **Finish**, чтобы импортировать проект; он появляется в окне **Package Explorer**.

- Запустите приложение **Flag Quiz**. Щелкните правой кнопкой мыши на проекте **FlagQuiz** и выберите в открывшемся меню команду **Run As ▶ Android Application**, чтобы выполнить приложение в **AVD** или на устройстве. Команда строит проект и запускает приложение (рис. 5.1 и 5.2).

5.2.2. Настройка викторины

При установке и первом запуске приложение настраивается так, чтобы для каждого вопроса выводились три варианта ответов и флаги из *всех* регионов. В этом тестовом запуске мы изменим конфигурацию так, чтобы выводились флаги только из Северной Америки; для количества вариантов будет оставлено значение по умолчанию (три). На телефоне, планшете или **AVD** в портретной ориентации коснитесь кнопки **меню** (≡, рис. 5.1). Выберите в открывшемся меню команду **Settings**; на экране появляется диалоговое окно **Flag Quiz Settings** (рис. 5.3, а). На планшетном устройстве или **AVD** в *альбомной* ориентации команды меню настроек отображаются в левой части экрана (рис. 5.2). Чтобы открыть диалоговое окно для выбора количества вариантов ответа, отображаемых для каждого флага (рис. 5.3, б), коснитесь кнопки **Number of Choices**. (На планшетном устройстве или **AVD** в *альбомной* ориентации весь экран окрашивается в серый цвет, а диалоговое окно отображается в центре экрана.) По умолчанию, для каждого флага отображаются три варианта ответа. Чтобы повысить сложность викторины, выберите вариант 6 или 9 и коснитесь кнопки **OK**; если вы не хотите изменять сложность, коснитесь кнопки **Cancel**, чтобы вернуться к окну **Flag Quiz Settings**. В тестовом запуске будет использоваться количество вариантов по умолчанию — три.

Коснитесь кнопки **Regions** (рис. 5.4, а), чтобы отобразить набор флажков, представляющих регионы мира (см. рис. 5.4, б). По умолчанию после первого запуска приложения отображаются все регионы, поэтому для викторины случайным образом выбираются флаги любых стран мира. Коснитесь флажков **Africa**, **Asia**, **Europe**, **Oceania** и **South America**, чтобы исключить эти регионы из викторины. Коснитесь кнопки **OK**, чтобы начать новую игру с изменившимися настройками. На телефонах, планшетах или **AVD** в портретной ориентации коснитесь кнопки **Back** (⟲), чтобы вернуться к викторине. На планшетном устройстве или **AVD** в *альбомной* ориентации в правой части экрана немедленно отображается очередной вопрос с обновленной конфигурацией.

5.2.3. Ответы на вопросы викторины

Начинается новая викторина с выбранным количеством вариантов и флагами, ограниченными североамериканским регионом. Попробуйте ответить на вопрос — коснитесь кнопки той страны, которой (по вашему мнению) принадлежит этот флаг.

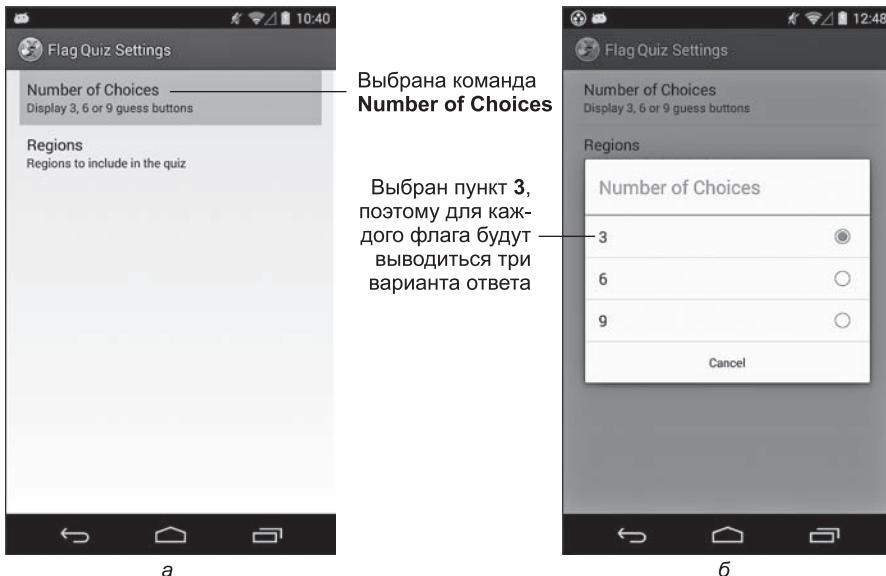


Рис. 5.3. Экран настроек Flag Quiz и диалоговое окно Number of Choices: *а* — меню, в котором пользователь выбирает команду Number of Choices; *б* — диалоговое окно с вариантами количества ответов

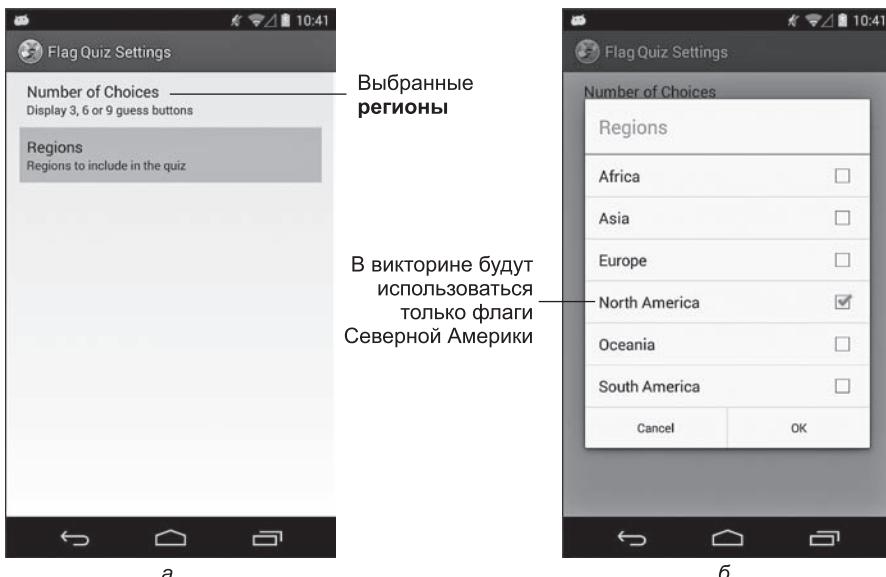


Рис. 5.4. Экран настроек Flag Quiz и диалоговое окно Regions: *а* — меню, в котором пользователь выбирает команду Regions; *б* — диалоговое окно с перечнем регионов

Выбирается правильный ответ

Если вариант был выбран правильно (рис. 5.5, а), приложение блокирует все кнопки ответов и выводит название страны зеленым шрифтом в нижней части экрана (рис. 5.5, б). После непродолжительной задержки приложение загружает следующий флаг с новым вариантом ответов.

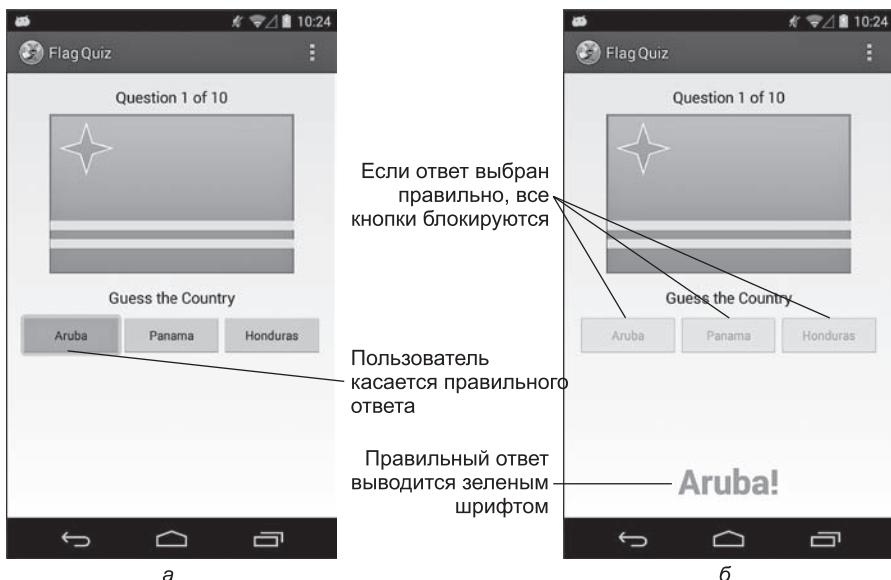


Рис. 5.5. Пользователь выбрал правильный ответ: а — выбор правильного ответа; б — правильный ответ выводится на экране

Выбирается неправильный ответ

Если вариант выбран неправильно (рис. 5.6, а), приложение блокирует кнопку с названием страны, использует анимацию, чтобы «покачать» флагом, и выводит в нижней части экрана сообщение *Incorrect!* красным шрифтом (рис. 5.6, б). Продолжайте попытки, пока не найдете правильный ответ для этого флага.

Завершение викторины

После успешного отгадывания стран, которым принадлежат десять отображаемых флагов, на экране появляется окно *AlertDialog*, в котором будет указано количество попыток отгадывания и процент правильных ответов (см. рис. 5.7). Кнопка *Reset Quiz* начинает новую игру с текущими настройками.

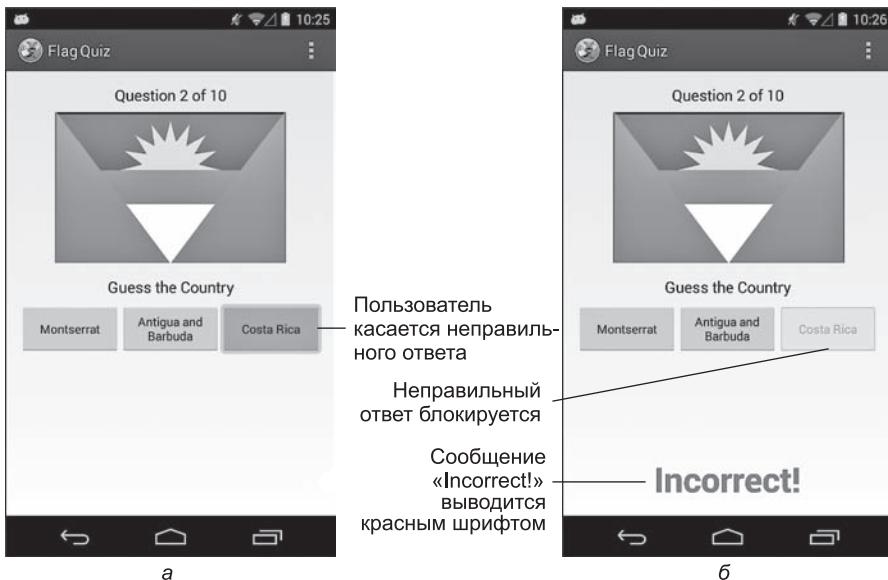


Рис. 5.6. Блокировка неправильного ответа в приложении Flag Quiz: а — выбор неправильного ответа; б — выводится сообщение Incorrect

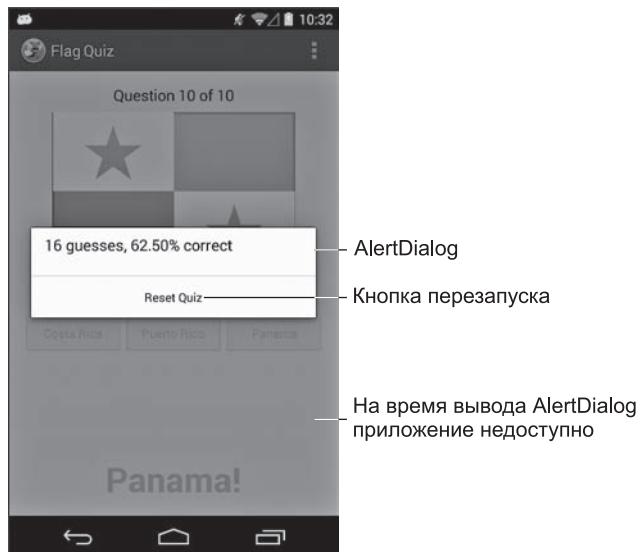


Рис. 5.7. Вывод результатов после завершения викторины

5.3. Обзор применяемых технологий

В этом разделе представлены новые возможности, которые будут использоваться при построении приложения Flag Quiz.

5.3.1. Меню

При создании проекта приложения в среде разработки класс `MainActivity` настраивается для отображения меню (☰) в правой части панели действий. Меню содержит команду `Settings`, которая обычно используется для отображения настроек приложения. В приложениях следующих глав вы узнаете, как создать новые команды меню и выбрать, какие команды должны отображаться непосредственно на панели действий или в меню.

Меню представляется объектом класса `Menu` (пакет `android.view`). Чтобы назначить команды `Menu`, следует переопределить метод `onCreateOptionsMenu` класса `Activity` (раздел 5.5.4) и добавить нужные команды в аргумент `Menu` метода. Когда пользователь выбирает команду меню, метод `onOptionsItemSelected` класса `Activity` (раздел 5.5.5) реагирует на выбор.

5.3.2. Фрагменты

Фрагмент (`fragment`) обычно представляет повторно используемую часть пользовательского интерфейса активности, но он также может представлять повторно используемый блок программной логики. Приложение использует фрагменты для создания частей графического интерфейса и управления ими. Фрагменты можно объединять для создания интерфейсов, эффективно использующих размер экрана планшета. Кроме того, простой механизм замены фрагментов сделает интерфейс приложения более динамичным (см. главу 8).

Базовым классом всех фрагментов является класс `Fragment` (пакет `android.app`). Приложение Flag Quiz определяет следующие субклассы `Fragment` (с прямым или косвенным наследованием).

- ❑ Класс `QuizFragment` (раздел 5.6) — прямой субкласс `Fragment`; он формирует графический интерфейс и определяет логику викторины. По аналогии с `Activity` каждый объект `Fragment` имеет собственный макет, который обычно определяется как ресурс макета, но может создаваться динамически. Графический интерфейс `QuizFragment` строится в разделе 5.4.8. Он будет использоваться в макетах `MainActivity` — для устройств в портретной ориентации и для планшетных устройств в альбомной ориентации.
- ❑ Класс `SettingsFragment` (раздел 5.7) является субклассом `PreferenceFragment` (пакет `android.preference`), способного автоматически поддерживать пользовательские

настройки в файле `SharedPreferences`. Как вы увидите, разработчик может создать XML-файл с описанием настроек, который используется классом `PreferenceFragment` для построения соответствующего интерфейса (рис. 5.3–5.4).

- ❑ Когда пользователь ответит на все вопросы, класс `QuizFragment` создает анонимный внутренний класс, расширяющий `DialogFragment` (пакет `android.app`), и использует его для отображения окна `AlertDialog` с результатами (раздел 5.6.9).

Фрагменты *должны* находиться под управлением активностей — они не могут выполняться автономно. Когда приложение работает на планшете в альбомной ориентации, `MainActivity` управляет всеми фрагментами. В портретной ориентации (на любом устройстве) класс `SettingsActivity` (раздел 5.8) управляет `SettingsFragment`, а `MainActivity` — всеми остальными фрагментами.

Хотя фрагменты появились в Android 3.0, они (а также некоторые более поздние возможности Android) могут использоваться в ранних версиях с поддержкой библиотеки Android Support Library. За дополнительной информацией обращайтесь по адресу

<http://developer.android.com/tools/support-library/index.html>

5.3.3. Методы жизненного цикла фрагментов

Каждый фрагмент, как и активность, имеет свой жизненный цикл и предоставляет методы, которые можно переопределять для обработки событий жизненного цикла. В этом приложении будут переопределены следующие методы.

- ❑ `onCreate` — этот метод (переопределяется в классе `SettingsFragment`) вызывается при создании фрагмента. Объекты `QuizFragment` и `SettingsFragment` создаются при заполнении макетов их родительских активностей, а объект `DialogFragment`, выводящий результаты, создается после завершения викторины.
- ❑ `onCreateView` — этот метод (переопределяется в классе `QuizFragment`) вызывается после `onCreate`; он должен построить и вернуть объект `View` с графическим интерфейсом фрагмента. Как вы вскоре увидите, он получает объект `LayoutInflater`, который используется для программного заполнения графического интерфейса фрагмента по компонентам, заданным в заранее определенном макете в формате XML.

Другие методы жизненного цикла фрагментов будут рассмотрены далее в книге. За полной информацией о подробностях жизненного цикла обращайтесь по адресу

<http://developer.android.com/guide/components/fragments.html>

5.3.4. Управление фрагментами

Родительская активность использует для управления своими фрагментами объект `FragmentManager` (пакет `android.app`), возвращаемый методом `getFragmentManager` класса `Activity`. Если активности потребуется взаимодействовать с фрагментом, который объявлен в макете активности и обладает идентификатором Id, то для получения ссылки на заданный фрагмент активность может вызвать метод `findFragmentById` класса `FragmentManager`. Как будет показано в главе 8, `FragmentManager` может использовать объекты `FragmentTransaction` (пакет `android.app`) для динамического добавления, удаления и переключения между фрагментами.

5.3.5. Объекты Preference

В разделе 5.2.2 мы изменили настройки приложения. Класс `PreferenceFragment` использует объекты `Preference` (пакет `android.preference`) для управления этими настройками. Приложение использует субкласс `Preference` с именем `ListPreference` для управления количеством вариантов ответов, отображаемых для каждого флага, а субкласс `Preference` с именем `MultiSelectListPreference` — для управления регионами, включаемыми в викторину. Варианты `ListPreference` являются взаимоисключающими, тогда как в `MultiSelectListPreference` можно выбрать любое количество вариантов. Для обращения к настройкам и работы с ними используется объект `PreferenceManager` (пакет `android.preference`).

5.3.6. Папка assets

Изображения флагов в приложении¹ загружаются приложением только тогда, когда в них возникнет необходимость. Они находятся в папке `assets` приложения. Чтобы добавить изображения в проект, мы перетащили в `assets` папки всех регионов из нашей файловой системы. Изображения можно найти в папке `images/FlagQuizImages` из папки примеров книги.

В отличие от папок `drawable`, в которых графический контент должен находиться на корневом уровне в каждой папке, папка `assets` может включать файлы произвольного типа, которые могут быть упорядочены по вложенным папкам (изображения флагов стран каждого региона находятся в отдельной подпапке). Доступ к файлам, находящимся в папках `assets`, обеспечивается с помощью диспетчера `AssetManager` (пакет `android.content.res`), который может предоставить список всех имен файлов в заданной папке, вложенной в `assets`, а также может применяться для обращения к каждому отдельному изображению.

¹ Изображения флагов были получены с веб-сайта www.free-country-flags.com.

5.3.7. Папки ресурсов

В разделе 2.4.4 вы узнали о папках `drawable`, `layout` и `values`, находящихся в папке `res` приложения. В этом приложении также будут использоваться папки ресурсов `menu`, `anim` и `xml`.

В табл. 5.1 приведены краткие описания этих папок (а также папок `animator`, `color` и `raw`).

Таблица 5.1. Другие папки, находящиеся в папке `res` проекта

Вложенная папка	Описание
<code>anim</code>	Папки с именами, начинающимися с <code>anim</code> , содержат XML-файлы с определениями анимаций с переходами, способных изменять прозрачность, размеры и позицию объектов, а также поворачивать их. Мы определим такую анимацию в разделе 5.4.11, а затем воспроизведем ее в разделе 5.6.9, чтобы создать визуальный эффект «дрожания»
<code>animator</code>	Папки с именами, начинающимися с <code>animator</code> , содержат XML-файлы с определениями анимаций свойств, изменяющими свойство объекта с течением времени. В Java свойства обычно реализуются в классах переменными экземпляров, имеющими <code>set-</code> и <code>get-</code> методы доступа
<code>color</code>	Папки с именами, начинающимися с <code>color</code> , содержат XML-файлы с определениями списков цветов для различных состояний (например, состояний кнопки — свободное, нажатое, заблокированное и т. д.)
<code>raw</code>	Папки с именами, начинающимися с <code>raw</code> , содержат файлы ресурсов (например, аудиоролики), которые читаются приложением как поток байтов. Такие ресурсы будут использоваться в главе 6 для воспроизведения звука
<code>menu</code>	Папки с именами, начинающимися с <code>menu</code> , содержат XML-файлы с описаниями содержимого меню. При создании проекта IDE автоматически определяет меню с командой <code>Settings</code>
<code>xml</code>	Папки с именами, начинающимися с <code>xml</code> , содержат XML-файлы, не относящиеся ни к одной другой категории ресурсов. Часто это низкоуровневые файлы данных в формате XML, используемые приложением. В разделе 5.4.10 мы создадим файл XML для настроек приложения, отображаемых классом <code>SettingsFragment</code>

5.3.8. Поддержка разных размеров экранов и разрешений

В разделе 2.5.1 вы узнали, что экраны устройств Android могут иметь разные размеры, разрешения и плотности пикселов (DPI, Dots Per Inch). Также вы узнали, что графика и другие визуальные ресурсы обычно предоставляются в нескольких разрешениях, чтобы система Android могла выбрать оптимальный ресурс для плотности пикселов устройства. Кроме того, в разделе 2.8 было рассказано о том, как

определять строковые ресурсы для разных языков и регионов. Android использует схему уточнения имен папок ресурсов, чтобы выбирать подходящие изображения на основании плотности пикселов устройства и строки правильных языков — на основании настроек локального контекста и региональных стандартов. Этот механизм также может использоваться для выбора ресурсов из папок, упомянутых в разделе 5.3.7.

Для класса `MainActivity` этого приложения используемый макет определяется квалификаторами размера и ориентации — один вариант для портретной ориентации на телефонах и планшетах, другой — для альбомной ориентации только на планшетах. Соответственно, для `MainActivity` определяются два макета:

- `activity_main.xml` из папки `res/layout` проекта — макет по умолчанию.
- `activity_main.xml` из папки `res/layout-large-land` проекта используется только на больших устройствах (например, планшетах), находящихся в альбомной (`land`) ориентации.

Уточненные имена папок ресурсов имеют формат:

имя-квалификаторы

где часть *квалификаторы* состоит из одного или нескольких квалификаторов, разделенных дефисами (-). Существует 18 видов квалификаторов, которые могут добавляться к именам папок ресурсов. Смысл других квалификаторов будет объясняться далее в книге. За полным списком всех квалификаторов вложенных папок `res` и правилами их определения в именах папок обращайтесь по адресу

<http://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources>

5.3.9. Определение размера экрана

В этом приложении меню отображается только в том случае, если приложение выполняется на телефоне или планшете в портретной ориентации (раздел 5.5.4). Чтобы получить эту информацию, мы воспользуемся классом `Android WindowManager` (пакет `android.view`) для получения объекта `Display` с текущей шириной и высотой экрана. Объект изменяется в зависимости от ориентации устройства — в портретной ориентации ширина устройства меньше его высоты.

5.3.10. Вывод временных сообщений

Класс `Toast` (пакет `android.widget`) ненадолго выводит сообщение, которое вскоре исчезает. Объекты `Toast` часто используются для отображения второстепенных сообщений об ошибках или информации (например, что викторина

будет перезапущена после изменения настроек). Когда пользователь изменяет конфигурацию приложения, на экране появляется временное окно `Toast` с сообщением о перезапуске. Также объекты `Toast` используются для предупреждения о необходимости выбрать хотя бы один регион, если пользователь исключит все регионы, — в этом случае приложение назначает Северную Америку регионом по умолчанию для викторины.

5.3.11. Использование обработчика для планирования будущих операций

Если пользователь выбрал правильный вариант, приложение выводит правильный ответ на две секунды, прежде чем вывести следующий флаг. Для этой цели используется объект `Handler` (пакет `android.os`). Метод `postDelayed` класса `Handler` в аргументах получает выполняемую задачу `Runnable` и величину задержки в миллисекундах. После истечения задержки задача `Runnable` выполняется в том же программном потоке, который создал `Handler`. Операции, взаимодействующие с графическим интерфейсом или изменяющие его, должны выполняться в GUI-потоке, потому что компоненты GUI не являются потоково-безопасными.

5.3.12. Применение анимации к компонентам

Если пользователь выбирает неверный вариант, флаг покачивается, для чего к компоненту `ImageView` применяется анимация `Animation` (пакет `android.view.animation`). Мы используем статический метод `loadAnimation` класса `AnimationUtils` для загрузки анимации из XML-файла, задающего параметры анимации. Также указывается количество повторений анимации (метод `setRepeatCount` класса `Animation`). Выполнение же самой анимации в `ImageView` реализуется путем вызова метода `startAnimation` класса `View` (с аргументом `Animation`) для компонента `ImageView`.

5.3.13. Регистрация исключений с помощью `Log.e`

Возникающие исключения можно *регистрировать* (в целях отладки) с помощью встроенного в Android механизма регистрации. В Android существует класс `Log` (пакет `android.util`) с набором статических методов, которые представляют различные аспекты сообщений об исключениях. Зарегистрированные сообщения можно просмотреть на вкладке `LogCat` в нижней части IDE, а также воспользоваться утилитой `Android logcat`. Дополнительную информацию о регистрации сообщений можно найти на веб-сайте

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

5.3.14. Использование явного интента для запуска другой активности в том же приложении

Когда приложение работает в портретной ориентации, настройки приложения отображаются в `SettingsActivity` (раздел 5.8). В главе 4 мы показали, как использовать неявный интент для отображения URL-адреса в браузере устройства. В разделе 5.5.5 вы узнаете, как использовать явный интент для запуска конкретной активности в том же приложении.

5.3.15. Структуры данных Java

В этом приложении использованы различные структуры данных из пакета `java.util`. Приложение динамически загружает имена графических файлов и сохраняет их в массиве `ArrayList<String>`. Метод `shuffle` класса `Collections` случайным образом изменяет порядок имен файлов в массиве `ArrayList<String>` для каждой новой игры. Второй массив `ArrayList<String>` используется для хранения имен файлов изображений, соответствующих 10 странам для текущей викторины. Объект `Set<String>` используется для хранения регионов текущей игры. Для обращения к объекту `ArrayList<String>` используется переменная интерфейсного типа `List<String>` — это полезный прием программирования на Java, который позволяет легко изменять структуры данных, не затрагивая при этом остальной код приложения.

5.4. Построение графического интерфейса и файлов ресурсов

В этом разделе мы создадим проект и настроим ресурсы (строки, массивы, цвета, метрики, макеты и анимации) для приложения Flag Quiz.

5.4.1. Создание проекта

Прежде чем создавать новый проект, удалите проект FlagQuiz, который тестиировался в разделе 5.2, — щелкните на нем правой кнопкой мыши и выберите команду Delete. В открывшемся диалоговом окне проследите за тем, чтобы переключатель `Delete project contents on disk` не был установлен, и нажмите OK.

Создание нового проекта приложения

Затем создайте новый проект Android Application Project. Введите следующие значения на первом шаге мастера New Android Project (New Android Application) и нажмите кнопку `Next >`:

- Application Name: Flag Quiz
- Project Name: FlagQuiz
- Package Name: com.deitel.flagquiz
- Minimum Required SDK: API18: Android 4.3
- Target SDK: API19: Android 4.4
- Compile With: API19: Android 4.4
- Theme: Holo Light with Dark Action Bar

На втором шаге мастера New Android Project (New Android Application) оставьте настройки по умолчанию и нажмите кнопку **Next >**. На шаге Configure Launcher Icon нажмите **Browse...** и выберите значок приложения (из папки **images** в папке примеров книги), нажмите **Open** и затем **Next >**. На шаге Create Activity выберите вариант **Blank Activity** и нажмите **Next >**. На шаге **Blank Activity** оставьте настройки по умолчанию и нажмите **Finish**, чтобы создать проект. В макетном редакторе выберите в списке типов экрана **Nexus 4** — как и в предыдущих примерах, это устройство будет взято за основу для построения интерфейса.

5.4.2. Файл strings.xml и ресурсы форматных строк

Так как строковые ресурсы уже создавались в предыдущих главах, сейчас мы приведем только таблицу с именами ресурсов и соответствующими значениями (табл. 5.2). Сделайте двойной щелчок на файле **strings.xml** из папки **res/values**, чтобы запустить редактор для создания этих строковых ресурсов.

Таблица 5.2. Строковые ресурсы, используемые в приложении Flag Quiz

Имя ресурса	Значение
settings_activity	Flag Quiz Settings
number_of_choices	Number of Choices
number_of_choices_description	Display 3, 6 or 9 guess buttons
world_regions	Regions
world_regions_description	Regions to include in the quiz
guess_country	Guess the Country
results	%1\$d guesses, %2\$.02f% correct
incorrect_answer	Incorrect!
default_region_message	Setting North America as the default region. One region must be selected.
restarting_quiz	Quiz will restart with your new settings

Таблица 5.2 (окончание)

Имя ресурса	Значение
ok	OK
question	Question %1\$d of %2\$d
reset_quiz	Reset Quiz
image_description	Image of the current flag in the quiz
default_region	North_America

Форматные строки как строковые ресурсы

Ресурсы `results` и `question` содержат форматные строки, которые используются с методом `format` класса `String`. Если ресурс `String` содержит несколько спецификаторов формата, их необходимо пронумеровать. В ресурсе `results` запись `1$` в `%1$d` обозначает *первый* аргумент метода `format` класса `String` после форматной строки, который должен заменить спецификатор `%1$d`. Аналогичным образом `2$` в `%2$.02f` обозначает *второй* аргумент метода `format` после форматной строки, который должен заменить спецификатор `%2$.02f`. Символ `d` в первом спецификаторе означает, что подставляемое значение должно форматироваться как целое число, а `f` во втором спецификаторе представляет значение в формате с плавающей точкой. В локализованных версиях `strings.xml` порядок спецификаторов формата `%1$d` и `%2$.02f` может измениться. Первый аргумент после форматной строки заменит спецификатор `%1$d`, а второй аргумент заменит `%2$.02f` независимо от их положения в форматной строке.

5.4.3. arrays.xml

В разделе 4.5.8 в файле `strings.xml` был создан ресурс строкового массива. Формально все ресурсы приложения из папки `res/values` могут определяться в одном файле. Тем не менее для удобства управления разными типами ресурсов для каждого типа обычно используется отдельный файл. Например, ресурсы массивов обычно определяются в `arrays.xml`, цветовые ресурсы — в `colors.xml`, строковые — в `strings.xml` и числовые — в `values.xml`. В этом приложении используются три ресурса строковых массивов, определяемые в `arrays.xml`:

- ❑ `regions_list` определяет названия регионов, с разделением слов символами подчеркивания — эти значения используются для загрузки имен файлов изображений из соответствующих папок, а также как выбираемые значения в `SettingsFragment`;
- ❑ `regions_list_for_settings` определяет имена регионов, с разделением слов пробелами — эти значения используются в `SettingsFragment` для вывода названий регионов для пользователя;
- ❑ `guesses_list` определяет строки 3, 6 и 9 — эти значения используются в `SettingsFragment` для выбора количества предлагаемых вариантов ответа.

В табл. 5.3 представлены имена и значения элементов этих трех массивов.

Таблица 5.3. Ресурсы строковых массивов, определяемые в arrays.xml

Имя массива ресурса	Значение
regions_list	Africa, Asia, Europe, North_America, Oceania, South_America
regions_list_for_settings	Africa, Asia, Europe, North America, Oceania, South America
guesses_list	3, 6, 9

Чтобы создать файл и настроить ресурсы массивов, выполните следующие действия.

1. В папке `res` проекта щелкните правой кнопкой мыши на папке `values` и выполните команду `New > Android XML File`. На экране появляется диалоговое окно `New Android XML File`. Так как щелчок был сделан на папке `values`, диалоговое окно заранее настраивается для добавления файла ресурсов `Values` в папке `values`.
2. Введите в поле `File` значение `arrays.xml` и нажмите `Finish`, чтобы завершить создание файла.
3. Если среда разработки открывает новый файл в режиме разметки XML, щелкните на вкладке `Resources` в нижней части окна, чтобы открыть его в редакторе ресурсов.
4. Чтобы создать ресурс строкового массива, нажмите `Add...`, выберите вариант `String Array` и нажмите `OK`.
5. В поле `Name` введите значение `regions_list` и сохраните файл.
6. Выделите новый ресурс строкового массива и воспользуйтесь кнопкой `Add`, чтобы добавить элементы для каждого из значений в табл. 5.3.
7. Повторите шаги 4–6 для массивов `regions_list_for_settings` и `guesses_list`. Когда вы щелкаете на кнопке `Add...` для создания дополнительных ресурсов строковых массивов, сначала необходимо установить переключатель `Create a new element at the top level in Resources`.

5.4.4. colors.xml

Приложение выводит правильные ответы зеленым шрифтом, а неправильные — красным. Цвета, как и все остальные ресурсы, следует определять в разметке XML, чтобы вы могли легко изменить цвета без модификации исходного кода Java. Обычно цвета определяются в файле с именем `colors.xml`, который необходимо создать. Как вы узнали в разделе 3.4.5, цвета определяются в схемах RGB или ARGB.

Чтобы создать файл и настроить два цветовых ресурса, выполните следующие действия.

1. В папке `res` проекта щелкните правой кнопкой мыши на папке `values` и выполните команду `New > Android XML File`. На экране появляется диалоговое окно `New Android XML File`.

2. Введите в поле **File** значение `colors.xml` и нажмите **Finish**, чтобы завершить создание файла.
3. Если среда разработки открывает новый файл в режиме разметки XML, щелкните на вкладке **Resources** в нижней части окна, чтобы открыть его в редакторе ресурсов.
4. Чтобы создать цветовой ресурс, нажмите **Add...**, выберите вариант **Color** и нажмите **OK**.
5. В полях **Name** и **Value** введите `correct_answer` и `#00CC00` соответственно, затем сохраните файл.
6. Повторите шаги 4 и 5 для значений `incorrect_answer` и `#FF0000`.

5.4.5. dimens.xml

Так как ресурсы метрик уже создавались в предыдущих главах, сейчас мы приведем только таблицу с именами ресурсов и соответствующими значениями (табл. 5.4). Откройте файл `dimens.xml` из папки `res/values`, чтобы открыть редактор для создания этих ресурсов. Ресурс `spacing` используется в макетах для определения промежутков между компонентами GUI, а ресурс `answer_size` задает размер шрифта для `answerTextView`. Как говорилось в разделе 2.5.3, размеры шрифта должны задаваться в пикселях, независимых от масштаба (`sp`), чтобы шрифты в приложении также могли масштабироваться под предпочтительный размер шрифта пользователя (заданный в параметрах устройства).

Таблица 5.4. Ресурсы метрик, используемые в приложении Flag Quiz

Имя массива ресурса	Значение
<code>spacing</code>	<code>8dp</code>
<code>answer_size</code>	<code>40sp</code>

5.4.6. Макет activity_settings.xml

В этом разделе мы создадим макет для активности `SettingsActivity` (раздел 5.8), которая отображает фрагмент `SettingsFragment` (раздел 5.7). Макет `SettingsActivity` будет состоять из единственного компонента `LinearLayout`, содержащего графический интерфейс `SettingsFragment`. Как вы вскоре увидите, при добавлении фрагмента в макет среда разработки может создать класс фрагмента за вас. Чтобы создать макет, выполните следующие действия.

1. В папке `res` проекта щелкните правой кнопкой мыши на папке `layout` и выполните команду `New > Android XML File`. На экране появляется диалоговое окно `New Android`

XML File. Так как щелчок был сделан на папке layout, диалоговое окно заранее настраивается для добавления файла ресурсов Layout.

2. Введите в поле **File** значение `activity_settings.xml`.
3. В разделе **Root Element** выберите вариант `LinearLayout` и нажмите **Finish**, чтобы создать файл.
4. Перетащите фрагмент из раздела **Layouts** палитры в область построения графического интерфейса или на узел `LinearLayout` в окне **Outline**.
5. Предыдущий шаг открывает диалоговое окно **Choose Fragment Class**. Если класс фрагмента был определен ранее его макета, вы сможете выбрать его в этом окне. Нажмите **Create New...**, чтобы отобразить диалоговое окно **New Java Class**.
6. Введите в поле **Name** значение `SettingsFragment` и замените содержимое поля **Superclass** значением `android.preference.PreferenceFragment`. Нажмите **Finish**, чтобы создать класс. Среда разработки открывает файл с кодом класса на языке Java, который пока можно закрыть.
7. Сохраните файл `activity_settings.xml`.

5.4.7. Макет `activity_main.xml` для телефонов и планшетов в портретной ориентации

В этом разделе будет создан макет для активности `MainActivity` (раздел 5.5), используемый в портретной ориентации на любых устройствах. Макет для альбомной ориентации на планшетах будет определен в разделе 5.4.9. Чтобы создать этот макет (в котором будет отображаться только `QuizFragment` – раздел 5.6), выполните следующие действия.

1. Откройте файл `activity_main.xml` из папки `res/layout`.
2. Перетащите компонент `Fragment` из раздела **Layouts** палитры на узел `RelativeLayout` в окне **Outline**.
3. В диалоговом окне **Choose Fragment Class** нажмите **Create New...**, чтобы открыть диалоговое окно **New Java Class**.
4. В поле **Name** введите значение `QuizFragment` и нажмите **Finish**, чтобы создать класс. Среда разработки открывает файл с кодом класса на языке Java, который пока можно закрыть.
5. В файле `activity_main.xml` выделите `QuizFragment` в окне **Outline**. Задайте свойству **Id** значение `@+id/quizFragment`, после чего задайте свойствам **Width** и **Height** из раздела **Layout Parameters** значение `match_parent`.
6. Сохраните файл `activity_main.xml`.

5.4.8. Макет fragment_quiz.xml

Обычно разработчик определяет макет для каждого из фрагментов своего приложения. Для каждого макета фрагмента он добавляет в папку(-и) `res/layout` XML-файл макета и указывает, с каким классом фрагмента связан этот макет. Для класса `SettingsFragment` этого приложения определять макет не нужно, потому что его графический интерфейс генерируется автоматически унаследованными средствами класса `PreferenceFragment`.

В этом разделе представлен макет фрагмента `QuizFragment` (`fragment_quiz.xml`). В папке `res/layout` приложения его файл макета будет определен только один раз, потому что один макет будет использоваться для всех устройств и всех ориентаций. На рис. 5.8 показаны имена компонентов `QuizFragment`.



Рис. 5.8. Компоненты графического интерфейса Flag Quiz и значения их свойств Id

Создание макета fragment_quiz.xml

Чтобы создать макет `fragment_quiz.xml`, выполните следующие действия.

1. В папке `res` проекта щелкните правой кнопкой мыши на папке `layout` и выполните команду `New > Android XML File`. На экране появляется диалоговое окно `New Android XML File`.
2. Введите в поле `File` значение `fragment_quiz.xml`.
3. В разделе `Root Element` выберите вариант `LinearLayout (Vertical)` и нажмите `Finish`, чтобы создать файл макета.

4. Используйте макетный редактор и окно Outline для создания структуры макета, изображенной на рис. 5.9. При создании компонентов графического интерфейса задайте им соответствующие значения ID. Для questionNumberTextView и guessCountryTextView мы использовали компоненты Medium Text из раздела Form Widgets палитры. Для кнопок были выбраны компоненты Small Button, использующие более мелкий шрифт, чтобы на них поместились больше текста.

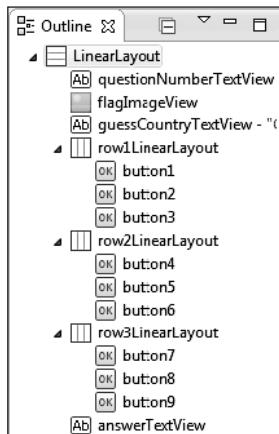


Рис. 5.9. Окно Outline для макета fragment_quiz.xml

5. После завершения шага 4 задайте значения свойств компонентов GUI в соответствии с табл. 5.5. Задавая свойство Height компонента flagImageView равным 0dp, а свойство Weight — равным 1, вы разрешаете этому компоненту изменять размеры по вертикали так, чтобы он занимал все свободное место, не занятое другими компонентами. Аналогичным образом кнопки со значениями свойств Width=0dp и Weight=1 равномерно распределяют горизонтальное пространство внутри компонента LinearLayout. Значение fitCenter свойства Scale Type компонента flagImageView масштабирует изображение так, чтобы оно заполняло ImageView по ширине или высоте с сохранением пропорций исходного изображения. Задавая свойству Adjust View Bounds компонента ImageView значение true, вы указываете, что компонент ImageView сохраняет пропорции своего объекта Drawable.

Таблица 5.5. Значения свойств компонентов в fragment_quiz.xml

Компонент GUI	Свойство	Значение
questionNumberTextView	Layout Parameters – Width – Height – Gravity	wrap_content wrap_content center_horizontal
	Other Properties – Text	@string/question

Таблица 5.5 (окончание)

Компонент GUI	Свойство	Значение
flagImageView	Layout Parameters – Width – Height – Gravity – Weight – Margins – Left/Right – Top/Bottom	wrap_content 0dp center 1 @dimen/activity_horizontal_margin @dimen/activity_vertical_margin
	Other Properties – Adjust View Bounds – Content Description – Scale Type	true @string/image_description fitCenter
guessCountryTextView	Layout Parameters – Width – Height – Gravity	wrap_content wrap_content center_horizontal
	Other Properties – Text	@string/guess_country
LinearLayout (все)	Layout Parameters – Width – Height – Margins – Bottom	match_parent wrap_content @dimen_spacing
Button (все)	Layout Parameters – Width – Height – Weight	0dp fill_parent 1
answerTextView	Layout Parameters – Width – Height – Gravity	wrap_content wrap_content center bottom
	Other Properties – Gravity – Text Size – Text Style	center_horizontal @dimen/answer_size bold

5.4.9. Макет activity_main.xml для планшета в альбомной ориентации

В разделе 5.4.7 определяется макет `MainActivity` для портретной ориентации, который содержит только фрагмент `QuizFragment`. Теперь мы определим макет

`MainActivity` для планшетов в альбомной ориентации, который будет содержать как `SettingsFragment`, так и `QuizFragment`.

Выполните следующие действия.

1. Щелкните правой кнопкой мыши на папке `res` проекта и выберите команду `New ▶ Folder`. Введите в поле `Folder` значение `layout-large-land` и нажмите `Finish`. Квалификаторы `large` и `land` означают, что все макеты, определяемые в этой папке, будут использоваться только на устройствах с большими экранами, на которых приложение выполняется в альбомной ориентации.
2. Щелкните правой кнопкой мыши на папке `layout-large-land` и выберите команду `New ▶ Android XML File`. На экране появляется диалоговое окно `New Android XML File`. Введите в поле `File` значение `activity_main.xml`. В разделе `Root Element` выберите вариант `LinearLayout (Horizontal)` и нажмите `Finish`, чтобы создать файл.
3. Выделите компонент `LinearLayout` и задайте его свойству `Base Aligned` значение `false`.
4. Перетащите компонент `Fragment` из раздела `Layouts` графического макета на узел `LinearLayout` в окне `Outline`. В диалоговом окне `Choose Fragment Class` выберите класс `SettingsFragment` и нажмите `OK`.
5. Повторите шаг 4 для класса `QuizFragment`, нажмите `OK`.
6. Выделите узел `SettingsFragment` в окне `Outline`. В разделе `Layout Parameters` задайте свойству `Width` значение `0dp`, свойству `Height` — значение `match_parent`, а свойству `Weight` — значение 1.
7. Выделите узел `QuizFragment` в окне `Outline`. В разделе `Layout Parameters` задайте свойству `Width` значение `0dp`, свойству `Height` — значение `match_parent`, а свойству `Weight` — значение 2.

Так как веса (`Weight`) фрагментов `QuizFragment` и `SettingsFragment` равны 2 и 1 соответственно, `QuizFragment` будет занимать 2/3 горизонтального пространства.

5.4.10. Определение конфигурации приложения в файле `preferences.xml`

В этом разделе мы создадим файл `preferences.xml`, используемый классом `SettingsFragment` для отображения настроек приложения. Выполните следующие действия:

1. Щелкните правой кнопкой мыши на папке `res` проекта и выберите команду `New ▶ Folder`. Введите в поле `Folder` значение `xml` и нажмите `Finish`.
2. Щелкните правой кнопкой мыши на папке `xml` и выберите команду `New ▶ Android XML File`. На экране появляется диалоговое окно `New Android XML File`.
3. Введите в поле `File` значение `preferences.xml`.

4. Убедитесь в том, что в разделе Resource Type установлен переключатель Preference, а в разделе Root Element выбрано значение PreferenceScreen, представляющее экран с перечнем настроек.
5. Нажмите Finish, чтобы создать файл. Если в среде разработки отображается низкоуровневая разметка XML, щелкните на вкладке Structure в нижней части окна.
6. В левой части окна выберите PreferenceScreen и нажмите Add....
7. В открывшемся диалоговом окне выберите ListPreference и нажмите OK. Этот вариант создает список взаимоисключающих вариантов.
8. В левой части окна выберите PreferenceScreen и нажмите Add....
9. В открывшемся диалоговом окне выберите MultiSelectListPreference и нажмите OK. Этот вариант создает список с возможностью одновременного выбора нескольких вариантов. Весь набор выбранных вариантов сохраняется как значение этого параметра.
10. Выделите компонент ListPreference и задайте его свойства в соответствии с табл. 5.6.
11. Выделите компонент MultiSelectListPreference и задайте его свойства в соответствии с табл. 5.7.
12. Сохраните и закройте файл preferences.xml.

Таблица 5.6. Значения свойств ListPreference

Свойство	Значение	Описание
Entries	@array/guesses_list	Массив строк, которые будут отображаться в виде списка вариантов
Entry values	@array/guesses_list	Массив значений, связанных с элементами из свойства Entries. Значение выбранного элемента будет храниться в объекте SharedPreferences
Key	pref_numberOfChoices	Имя параметра, хранящегося в SharedPreferences
Title	@string/number_of_choices	Заголовок параметра, отображаемый в GUI
Summary	@string/number_of_choices_description	Краткое описание параметра, отображаемое под заголовком
Persistent	true	Признак сохранения параметра после завершения приложения — если он равен true, класс PreferenceFragment немедленно сохраняет значение параметра при каждом его изменении
Default value	3	Элемент из свойства Entries, которое выбирается по умолчанию

Таблица 5.7. Значения свойств MultiSelectListPreference

Свойство	Значение	Описание
Entries	@array/regions_list_for_settings	Массив строк, которые будут отображаться в виде списка вариантов
Entry values	@array/regions_list	Массив значений, связанных с элементами из свойства Entries. Значения всех выбранных элементов будут храниться в объекте SharedPreferences
Key	pref_regionsToInclude	Имя параметра, хранящегося в SharedPreferences
Title	@string/world_regions	Заголовок параметра, отображаемый в GUI
Summary	@string/world_regions_description	Краткое описание параметра, отображаемое под заголовком
Persistent	true	Признак сохранения параметра после завершения приложения
Default value	@array/regions_list	Массив значений по умолчанию этого параметра — в данном случае по умолчанию будут выбраны все элементы

5.4.11. Создание анимации «качающегося» флага

В этом разделе мы создадим анимацию «качающегося флага», используемую в случае неправильного выбора варианта. В разделе 5.6.9 будет показано, как использовать эту анимацию в приложении. Для создания этой анимации выполните следующие действия.

1. Щелкните правой кнопкой мыши на папке res макета и в контекстном меню выберите команду New ▶ Folder, в поле Folder name введите значение anim и нажмите Finish.
2. Щелкните правой кнопкой мыши на узле anim и выберите команду New ▶ Android XML File для отображения диалогового окна New Android XML File.
3. В текстовом поле File введите имя incorrect_shake.xml.
4. Убедитесь в том, что переключатели Resource Type is Tween Animation и Root Element установлены.
5. Нажмите Finish, чтобы создать файл. Файл немедленно открывается в режиме разметки XML.
6. К сожалению, среда разработки не предоставляет редактор для анимаций, поэтому вам придется задать содержимое файла в формате XML в соответствии с листингом 5.1.

Листинг 5.1. Анимация «качающегося флага» (incorrect_shake.xml), применяемая к флагу в случае неправильного выбора варианта

```
1 <?xml version="1.0" encoding="utf-8"?>
2
3 <set xmlns:android="http://schemas.android.com/apk/res/android"
4     android:interpolator="@android:anim/decelerate_interpolator">
5
6     <translate android:fromXDelta="0" android:toXDelta="-5%p"
7         android:duration="100"/>
8
9     <translate android:fromXDelta="-5%p" android:toXDelta="5%p"
10        android:duration="100" android:startOffset="100"/>
11
12    <translate android:fromXDelta="5%p" android:toXDelta="-5%p"
13        android:duration="100" android:startOffset="200"/>
14 </set>
```

В рассматриваемом примере используются анимации `View` для создания эффекта «качающегося флага». Эта анимация фактически состоит из трех анимаций, объединенных в *набор анимаций* (строки 3–14) — группу, образующих большую анимацию. Наборы анимаций могут включать произвольную комбинацию *анимаций с переходами* — `alpha` (прозрачность), `scale` (изменение размеров), `translate` (перемещение) и `rotate` (поворот). Анимация «качающегося флага» состоит из трех анимаций `translate`. Анимация `translate` перемещает компонент `View` внутри родительского компонента. Android также поддерживает *анимации свойств*, позволяя задавать анимацию любого свойства произвольного объекта.

Первая анимация `translate` (строки 6–7) перемещает компонент `View` из начальной позиции в конечную через заданный период времени. Атрибут `android:fromXDelta` определяет смещение компонента `View` (в начальной позиции анимации), а атрибут `android:toXDelta` — смещение `View` в конечной позиции анимации. Эти атрибуты могут включать:

- абсолютные значения (в пикселях);
- проценты от размера анимированного компонента `View`;
- проценты от размера *родительского* компонента `View`.

Атрибуту `android:fromXDelta` было присвоено абсолютное значение 0. Атрибуту `android:toXDelta` было задано значение `-5%p`, означающее, что компонент `View` должен быть перемещен *влево* (знак «минус») на 5% от ширины родительского компонента (на это указывает буква `p`). Если потребуется выполнить перемещение на величину, равную 5% от ширины компонента `View`, букву `p` не указывайте. Атрибут `android:duration` определяет продолжительность анимации (в миллисекундах). Таким образом, анимация, определенная в строках 6–7, переместит компонент `View` влево на 5% относительно ширины родительского компонента в течение 100 миллисекунд.

Вторая анимация (строки 9–10) продолжается с того места, где была завершена первая. Она перемещает компонент `View` с позиции, заданной смещением `-5%`, в позицию, заданную смещением `%5p`, в течение 100 миллисекунд. По умолчанию анимации, включенные в набор анимаций, применяются параллельно, но с помощью атрибута `android:startOffset` можно задать величину задержки (в миллисекундах) перед началом анимации. Задержка может использоваться для последовательного выполнения анимаций. В нашем примере вторая анимация начинается через 100 миллисекунд после завершения первой. Третья анимация (строки 12–13) совпадает со второй, но выполняется в обратном направлении и начинается через 200 миллисекунд после завершения первой анимации.

5.5. Класс MainActivity

Класс `MainActivity` (листинги 5.2–5.7) управляет фрагментом `QuizFragment` при выполнении приложения в портретной ориентации и фрагментами `SettingsFragment` и `QuizFragment` — когда приложение выполняется на планшете в альбомной ориентации.

5.5.1. Команда `package`, команды `import` и поля

В листинге 5.2 приведены команды `package` и `import`, а также поля класса `MainActivity`. В строках 6–21 импортируются различные классы и интерфейсы Java и Android, используемые приложением. Мы выделили новые команды `import`; соответствующие классы и интерфейсы рассматриваются в разделе 5.3, а также о них упоминается в разделах 5.5.2–5.5.6.

Листинг 5.2. Команда `package`, команды `import` и поля класса `MainActivity`

```
1 // MainActivity.java
2 // Класс управляет фрагментом QuizFragment на телефоне и фрагментами
3 // QuizFragment и SettingsFragment на планшете
4 package com.deitel.flagquiz;
5
6 import java.util.Set;
7
8 import android.app.Activity;
9 import android.content.Intent;
10 import android.content.SharedPreferences;
11 import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
12 import android.content.pm.ActivityInfo;
13 import android.content.res.Configuration;
14 import android.graphics.Point;
15 import android.os.Bundle;
16 import android.preference.PreferenceManager;
17 import android.view.Display;
```

```
18 import android.view.Menu;
19 import android.view.MenuItem;
20 import android.view.WindowManager;
21 import android.widget.Toast;
22
23 public class MainActivity extends Activity
24 {
25     // Ключи для чтения данных из SharedPreferences
26     public static final String CHOICES = "pref_numberOfChoices";
27     public static final String REGIONS = "pref_regionsToInclude";
28
29     private boolean phoneDevice = true; // Включение портретного режима
30     private boolean preferencesChanged = true; // Настройки изменились?
31 }
```

В строках 26–27 определяются константы для ключей параметров, созданных в разделе 5.4.10. Ключи будут использоваться для обращения к соответствующим значениям параметров. Логическая переменная `phoneDevice` (строка 29) указывает, выполняется ли приложение на телефоне, — и если выполняется, то приложение разрешает только портретную ориентацию. Логическая переменная `preferencesChanged` (строка 30) указывает, изменились ли настройки приложения, — и если изменились, то метод жизненного цикла `onStart` класса `MainActivity` (раздел 5.5.3) вызывает методы `updateGuessRows` (раздел 5.6.4) и `updateRegions` (раздел 5.6.5) класса `QuizFragment` для изменения конфигурации викторины на основании новых настроек. Изначально этой переменной присваивается значение `true`, чтобы при первом запуске использовалась конфигурация с параметрами по умолчанию.

5.5.2. Переопределение метода `onCreate`

Переопределяемый метод `onCreate` класса `Activity` (листинг 5.3) вызывает метод `setContentView` (строка 36) для назначения графического интерфейса `MainActivity`. Android выбирает файл `activity_main.xml` из папки `res/layout`, если приложение выполняется в портретной ориентации, или из папки `res/layout-large-land`, если приложение выполняется на планшете в альбомной ориентации.

Листинг 5.3. Переопределенный метод `onCreate` класса `MainActivity`

```
32     @Override
33     protected void onCreate(Bundle savedInstanceState)
34     {
35         super.onCreate(savedInstanceState);
36         setContentView(R.layout.activity_main);
37
38         // Значения по умолчанию в SharedPreferences
39         PreferenceManager.setDefaultValues(this, R.xml.preferences, false);
40
41         // Регистрация слушателя для изменений SharedPreferences
42         PreferenceManager.getDefaultSharedPreferences(this).
43             registerOnSharedPreferenceChangeListener(
```

```
44         preferenceChangeListener);
45
46     // Определение размера экрана
47     int screenSize = getResources().getConfiguration().screenLayout &
48         Configuration.SCREENLAYOUT_SIZE_MASK;
49
50     // На планшете присвоить phoneDevice значение false
51     if (screenSize == Configuration.SCREENLAYOUT_SIZE_LARGE ||
52         screenSize == Configuration.SCREENLAYOUT_SIZE_XLARGE )
53         phoneDevice = false; // Не телефон
54
55     // На телефоне разрешена только портретная ориентация
56     if (phoneDevice)
57         setRequestedOrientation(
58             ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
59 } // Конец метода onCreate
60
```

Определение значений по умолчанию и регистрация слушателя изменений

При установке и первом запуске приложения в строке 39 назначаются параметры конфигурации приложения по умолчанию, для чего вызывается метод `setDefaultValues` класса `PreferenceManager` — он создает и инициализирует файл `SharedPreferences` приложения, используя значения по умолчанию, заданные в `preferences.xml`. Метод получает три аргумента.

- Контекст — активность (`this`), для которой задаются значения по умолчанию.
- Идентификатор ресурса для XML-файла настроек (`R.xml.preferences`), создаваемого в разделе 5.4.10.
- Логический признак, определяющий, должны ли значения по умолчанию сбрасываться при каждом вызове метода `setDefaults`, — значение `false` указывает, что значения настроек по умолчанию должны задаваться только при первом вызове этого метода.

Каждый раз, когда пользователь изменяет настройки приложения, объект `MainActivity` должен вызвать метод `updateGuessRows` или `updateRegions` класса `QuizFragment` (в зависимости от того, какой параметр был изменен). `MainActivity` регистрирует слушателя `OnSharedPreferenceChangeListener` (строки 42–44), чтобы получать оповещения обо всех изменениях настроек. Метод `getDefaults` класса `PreferenceManager` возвращает ссылку на объект `SharedPreferences`, представляющий параметры конфигурации приложения, а метод `registerOnSharedPreferenceChangeListener` класса `SharedPreferences` регистрирует слушателя, определяемого в разделе 5.5.6.

Настройка портретной ориентации на телефонах

Строки 47–53 определяют, на каком устройстве выполняется приложение: на планшете или на телефоне? Унаследованный метод `getResources` возвращает объект `Resources` приложения (пакет `android.content.res`), который может использоваться

для обращения к ресурсам приложения и получения информации о среде выполнения. Метод `getConfiguration` класса `Resources` возвращает объект `Configuration` (пакет `android.content.res`) с открытой (`public`) переменной экземпляра `screenLayout`, по которой можно определить категорию размера экрана устройства. Для этого значение `screenLayout` объединяется с маской `Configuration.SCREENLAYOUT_SIZE_MASK` при помощи поразрядного оператора И (`&`). Затем результат сравнивается с константами `SCREENLAYOUT_SIZE_LARGE` и `SCREENLAYOUT_SIZE_XLARGE` класса `Configuration` (строки 51–52). Если одна из проверок даст положительный результат, значит приложение выполняется на планшетном устройстве. Наконец, если устройство является телефоном, в строках 57–58 вызывается унаследованный от `Activity` метод `setRequestedOrientation`, который заставляет приложение отображать `MainActivity` только в портретной ориентации.

5.5.3. Переопределение метода `onStart`

Переопределяемый метод жизненного цикла активности `onStart` (листинг 5.4) вызывается в следующих двух ситуациях.

- При первом запуске приложения метод `onStart` вызывается после `onCreate`. В этом случае вызов `onStart` гарантирует, что приложение будет правильно инициализировано в состоянии по умолчанию при установке и первом запуске или в соответствии с обновленной конфигурацией пользователя при последующих запусках.
- Если приложение выполняется в портретной ориентации, а пользователь открывает `SettingsActivity`, активность `MainActivity` приостанавливается на время отображения `SettingsActivity`. Когда пользователь возвращается к `MainActivity`, снова вызывается метод `onStart`. На этот раз вызов обеспечивает необходимое изменение конфигурации, если пользователь внес изменения в настройки.

В обоих случаях, если переменная `preferencesChanged` равна `true`, `onStart` вызывает методы `updateGuessRows` (раздел 5.6.4) и `updateRegions` (раздел 5.6.5) класса `QuizFragment` для изменения конфигурации. Чтобы получить ссылку на объект `QuizFragment` для вызова его методов, в строках 71–72 используется унаследованный от `Activity` метод `getFragmentManager` для получения объекта `FragmentManager`, после чего вызывается его метод `findFragmentById`. Затем в строках 73–76 вызываются методы `updateGuessRows` и `updateRegions` класса `QuizFragment`, при этом в аргументах передается объект `SharedPreferences` приложения, чтобы эти методы могли загрузить текущую конфигурацию. Стока 77 сбрасывает состояние игры.

Листинг 5.4. Переопределенный метод `onStart` класса `MainActivity`

```
61 // Вызывается после завершения выполнения onCreate
62 @Override
63 protected void onStart()
64 {
```

```

65     super.onStart();
66
67     if (preferencesChanged)
68     {
69         // после того, как была задана конфигурация по умолчанию,
70         // инициализировать QuizFragment и запустить викторину
71         QuizFragment quizFragment = (QuizFragment)
72             getFragmentManager().findFragmentById(R.id.quizFragment);
73         quizFragment.updateGuessRows(
74             PreferenceManager.getDefaultSharedPreferences(this));
75         quizFragment.updateRegions(
76             PreferenceManager.getDefaultSharedPreferences(this));
77         quizFragment.resetQuiz();
78         preferencesChanged = false;
79     }
80 } // Конец метода onStart
81

```

5.5.4. Переопределение метода onCreateOptionsMenu

Метод `onCreateOptionsMenu` (листинг 5.5) вызывается для инициализации стандартного меню активности. Система передает объект `Menu`, в котором должны отображаться команды. В нашем приложении меню должно отображаться только при запуске приложения в портретной ориентации. В строках 87–88 объект `WindowManager` используется для получения объекта `Display` с текущими значениями ширины и высоты экрана, изменяющимися в зависимости от ориентации устройства. Если ширина меньше высоты, значит устройство находится в портретной ориентации. В строке 89 создается объект `Point` для хранения текущей ширины и высоты, после чего в строке 90 для объекта `Display` вызывается метод `getRealSize`, который сохраняет ширину и высоту экрана в открытых переменных `x` и `y` объекта `Point`. Если ширина меньше высоты (строка 93), то строка 95 создает меню на базе `menu.xml` — ресурса меню по умолчанию, созданного средой разработки при создании проекта. Метод `getMenuInflater`, унаследованный от `Activity`, возвращает объект `MenuItemInflater`, для которого вызывается метод `inflate` с двумя аргументами — идентификатором ресурса меню, используемого для заполнения меню, и объектом `Menu`, в который будут помещены команды меню. Если метод `onCreateOptionsMenu` возвращает `true`, это означает, что меню должно отображаться на экране.

Листинг 5.5. Переопределенный метод onCreateOptionsMenu класса MainActivity

```

82     // Меню отображается на телефонах и планшетах в портретной ориентации
83     @Override
84     public boolean onCreateOptionsMenu(Menu menu)
85     {
86         // Получение объекта Display для текущего экрана
87         Display display = ((WindowManager)
88             getSystemService(WINDOW_SERVICE)).getDefaultDisplay();
89         Point screenSize = new Point(); // Для хранения размера экрана

```

```

90     display.getRealSize(screenSize); // Размер сохраняется в screenSize
91
92     // Меню отображается только в портретной ориентации
93     if (screenSize.x < screenSize.y) // x - ширина, y - высота
94     {
95         getMenuInflater().inflate(R.menu.main, menu); // Заполнение меню
96         return true;
97     }
98     else
99         return false;
100 } // Конец метода onCreateOptionsMenu
101

```

5.5.5. Переопределение метода onOptionsItemSelected

Метод `onOptionsItemSelected` (листинг 5.6) вызывается при выборе команды меню. В нашем приложении меню по умолчанию, сгенерированное средой разработки при создании проекта, состоит из единственной команды `Settings`; следовательно, если этот метод вызывается, пользователь выбрал команду `Settings`. Стока 106 создает явный интент для запуска `SettingsActivity`. Используемый конструктор `Intent` получает объект контекста, из которого будет запускаться активность, и класс, представляющий запускаемую активность (`SettingsActivity.class`). Затем этот интент передается унаследованному методу `startActivity` для запуска активности.

Листинг 5.6. Переопределенный метод onOptionsItemSelected класса MainActivity

```

102    // Запускает SettingsActivity при выполнении на телефоне
103    @Override
104    public boolean onOptionsItemSelected(MenuItem item)
105    {
106        Intent preferencesIntent = new Intent(this, SettingsActivity.class);
107        startActivity(preferencesIntent);
108        return super.onOptionsItemSelected(item);
109    }
110

```

5.5.6. Анонимный внутренний класс, реализующий интерфейс OnSharedPreferenceChangeListener

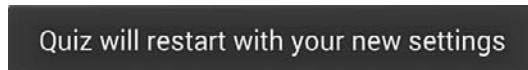
Объект `preferenceChangeListener` (листинг 5.7) является объектом анонимного внутреннего класса, реализующим интерфейс `OnSharedPreferenceChangeListener`. Этот объект был зарегистрирован в методе `onCreate` для прослушивания изменений в конфигурации `SharedPreferences` приложения. Когда происходит изменение, метод `onSharedPreferenceChanged` задает `preferencesChanged` значение `true` (строка 120), после чего получает ссылку на `QuizFragment` (строки 122–123) для сброса викторины

с новой конфигурацией. Если изменился параметр CHOICES, то в строках 127–128 вызываются методы updateGuessRows и resetQuiz класса QuizFragment.

Листинг 5.7. Анонимный внутренний класс, реализующий OnSharedPreferenceChangeListener

```
111 // Слушатель изменений в конфигурации SharedPreferences приложения
112 private OnSharedPreferenceChangeListener preferenceChangeListener =
113     new OnSharedPreferenceChangeListener()
114 {
115     // Вызывается при изменении конфигурации пользователем
116     @Override
117     public void onSharedPreferenceChanged(
118         SharedPreferences sharedPreferences, String key)
119     {
120         preferencesChanged = true; // Пользователь изменил конфигурацию
121
122         QuizFragment quizFragment = (QuizFragment)
123             getFragmentManager().findFragmentById(R.id.quizFragment);
124
125         if (key.equals(CHOICES)) // Изменилось количество вариантов
126         {
127             quizFragment.updateGuessRows(sharedPreferences);
128             quizFragment.resetQuiz();
129         }
130         else if (key.equals(REGIONS)) // Изменился набор регионов
131         {
132             Set<String> regions =
133                 sharedPreferences.getStringSet(REGIONS, null);
134
135             if (regions != null && regions.size() > 0)
136             {
137                 quizFragment.updateRegions(sharedPreferences);
138                 quizFragment.resetQuiz();
139             }
140             else // Должен быть выбран хотя бы один регион
141             {
142                 // (по умолчанию выбирается Северная Америка)
143                 SharedPreferences.Editor editor = sharedPreferences.edit();
144                 regions.add(
145                     getResources().getString(R.string.default_region));
146                 editor.putStringSet(REGIONS, regions);
147                 editor.commit();
148                 Toast.makeText(MainActivity.this,
149                     R.string.default_region_message,
150                     Toast.LENGTH_SHORT).show();
151             }
152         }
153
154         Toast.makeText(MainActivity.this,
155             R.string.restarting_quiz, Toast.LENGTH_SHORT).show();
156     } // Конец метода onSharedPreferenceChanged
157 }; // Конец анонимного внутреннего класса
158 } // Конец класса MainActivity
```

Если изменился параметр `REGIONS`, то строки 132–133 получают коллекцию `Set<String>` с включенными регионами. Метод `getStringSet` класса `SharedPreferences` возвращает `Set<String>` для заданного ключа. В викторину должен быть включен хотя бы один регион, поэтому если множество `Set<String>` не пусто, строки 137–138 вызывают методы `updateRegions` и `resetQuiz` класса `QuizFragment`. В противном случае в строках 142–146 в параметре `REGIONS` выбирается Северная Америка как регион по умолчанию, а в строках 147–149 выводится предупреждение о назначении региона по умолчанию. В аргументах метода `makeText` класса `Toast` передается контекст, в котором отображается `Toast`, выводимое сообщение и продолжительность вывода. Временное окно отображается на экране вызовом метода `show` класса `Toast`. Независимо от того, какой параметр изменился, в строках 153–154 выводится предупреждение о том, что викторина будет перезапущена с новыми параметрами. Временное окно `Toast`, появляющееся при изменении конфигурации приложения пользователем, показано на рис. 5.10.



Quiz will restart with your new settings

Рис. 5.10. Временное окно, появляющееся при изменении конфигурации

5.6. Класс QuizFragment

Класс `QuizFragment` (листинги 5.8–5.16) строит графический интерфейс игры и реализует ее логику.

5.6.1. Команда package и команды import

В листинге 5.8 приведены команды `package` и `import` класса `QuizFragment`. В строках 5–33 импортируются различные классы и интерфейсы Java и Android, используемые приложением. Мы выделили новые команды `import`; соответствующие классы и интерфейсы рассматриваются в разделе 5.3, а также там, где они упоминаются (в разделах 5.6.2–5.6.10).

Листинг 5.8. Команда package и команды import класса QuizFragment

```
1 // QuizFragment.java
2 // Логика игры Flag Quiz
3 package com.deitel.flagquiz;
4
5 import java.io.IOException;
6 import java.io.InputStream;
7 import java.security.SecureRandom;
8 import java.util.ArrayList;
9 import java.util.Collections;
10 import java.util.List;
```

```
11 import java.util.Set;
12
13 import android.app.AlertDialog;
14 import android.app.Dialog;
15 import android.app.DialogFragment;
16 import android.app.Fragment;
17 import android.content.DialogInterface;
18 import android.content.SharedPreferences;
19 import android.content.res.AssetManager;
20 import android.graphics.drawable.Drawable;
21 import android.os.Bundle;
22 import android.os.Handler;
23 import android.util.Log;
24 import android.view.LayoutInflater;
25 import android.view.View;
26 import android.view.View.OnClickListener;
27 import android.view.ViewGroup;
28 import android.view.animation.Animation;
29 import android.view.animation.AnimationUtils;
30 import android.widget.Button;
31 import android.widget.ImageView;
32 import android.widget.LinearLayout;
33 import android.widget.TextView;
34
```

5.6.2. Поля

В листинге 5.9 перечислены статические переменные и переменные экземпляров класса QuizFragment. Константа TAG (строка 38) используется при регистрации ошибок средствами класса Log (листинг 5.13), чтобы эти сообщения об ошибках отличались от других сообщений, записываемых в журнал устройства. Константа FLAGS_IN QUIZ (строка 40) представляет количество флагов в викторине.

Листинг 5.9. Поля класса QuizFragment

```
35 public class QuizFragment extends Fragment
36 {
37     // Страна, используемая при регистрации сообщений об ошибках
38     private static final String TAG = "FlagQuiz Activity";
39
40     private static final int FLAGS_IN QUIZ = 10;
41
42     private List<String> fileNameList; // Имена файлов с флагами
43     private List<String> quizCountriesList; // Страны в текущей игре
44     private Set<String> regionsSet; // Регионы в текущей игре
45     private String correctAnswer; // Правильная страна для текущего флага
46     private int totalGuesses; // Количество предположений
47     private int correctAnswers; // Количество правильных ответов
48     private int guessRows; // Количество строк с кнопками вариантов
49     private SecureRandom random; // Обеспечивает случайное распределение
50     private Handler handler; // Для задержки при загрузке следующего флага
51     private Animation shakeAnimation; // Анимация неправильного ответа
52
```

```
53     private TextView questionNumberTextView; // Для номера вопроса
54     private ImageView flagImageView; // Для вывода флага
55     private LinearLayout[] guessLinearLayouts; // Для кнопок с вариантами
56     private TextView answerTextView; // Для текста Correct! или Incorrect!
57
```

Переменная `fileNameList` (строка 42) содержит имена файлов с изображениями флагов для текущего набора выбранных регионов. Переменная `quizCountriesList` (строка 43) содержит имена файлов с флагами для стран, используемых в текущей игре. В переменной `regionsSet` (строка 44) хранится информация о наборе регионов.

Переменная `correctAnswer` (строка 45) содержит имя файла с флагом для текущего правильного ответа. В переменной `totalGuesses` (строка 46) хранится общее количество правильных и неправильных ответов игрока до настоящего момента. Переменная `correctAnswers` (строка 47) содержит количество правильных ответов; если пользователь завершит викторину, это значение будет равно `FLAGS_IN QUIZ`. В переменной `guessRows` (строка 48) хранится количество компонентов `LinearLayout` с тремя кнопками, используемыми для вывода вариантов ответов.

Переменная `random` (строка 49) используется для случайного выбора флагов, включаемых в викторину, и местонахождения кнопки, представляющей правильный ответ в компонентах `LinearLayout`. Если пользователь выбрал правильный ответ, но викторина еще не закончена, объект `Handler` (строка 50) используется для загрузки следующего флага после непродолжительной задержки.

Объект анимации `shakeAnimation` (строка 51) содержит динамически заполняемую анимацию «встряхивания», которая применяется к изображению флага в том случае, если пользователь дал неправильный ответ. Строки 53–56 содержат переменные, используемые для выполнения операций с различными компонентами GUI из программного кода.

5.6.3. Переопределение метода onCreateView

Метод `onCreateView` класса `QuizFragment` (листинг 5.10) заполняет GUI и инициализирует большинство переменных экземпляров `QuizFragment` — `guessRows` и `regionsSet` инициализируются, когда `MainActivity` вызывает методы `updateGuessRows` и `updateRegions` класса `QuizFragment`. После вызова метода `onCreateView` суперкласса (строка 63) мы заполняем графический интерфейс `QuizFragment` (строки 64–65), используя объект `LayoutInflater`, передаваемый методу `onCreateView` в аргументе. Метод `inflate` класса `LayoutInflater` получает три аргумента:

- ❑ идентификатор ресурса макета, определяющий заполняемый макет;
- ❑ объект `ViewGroup` (объект макета), в котором будет отображаться макет; передается во втором аргументе `onCreateView`;
- ❑ логический признак, который указывает, должен ли заполненный графический интерфейс быть присоединен к объекту `ViewGroup` из второго аргумента, — значение

`false` означает, что фрагмент был объявлен в макете родительской активности, а `true` — что фрагмент создается динамически и к нему необходимо присоединить графический интерфейс.

Метод `inflate` возвращает ссылку на объект `View`, содержащий заполненный графический интерфейс. Ссылка сохраняется в локальной переменной `view`, чтобы ее можно было вернуть из `onCreateView` после инициализации других переменных экземпляров `QuizFragment`.

Листинг 5.10. Переопределенный метод `onCreateView` класса `QuizFragment`

```
58 // Настраивает фрагмент QuizFragment при создании его представления
59 @Override
60 public View onCreateView(LayoutInflater inflater, ViewGroup container,
61     Bundle savedInstanceState)
62 {
63     super.onCreateView(inflater, container, savedInstanceState);
64     View view =
65         inflater.inflate(R.layout.fragment_quiz, container, false);
66
67     fileNameList = new ArrayList<String>();
68     quizCountriesList = new ArrayList<String>();
69     random = new SecureRandom();
70     handler = new Handler();
71
72     // Загрузка анимации неправильных ответов
73     shakeAnimation = AnimationUtils.loadAnimation(getActivity(),
74         R.anim.incorrect_shake);
75     shakeAnimation.setRepeatCount(3); // Анимация повторяется 3 раза
76
77     // Получение ссылок на компоненты GUI
78     questionNumberTextView =
79         (TextView) view.findViewById(R.id.questionNumberTextView);
80     flagImageView = (ImageView) view.findViewById(R.id.flagImageView);
81     guessLinearLayouts = new LinearLayout[3];
82     guessLinearLayouts[0] =
83         (LinearLayout) view.findViewById(R.id.row1LinearLayout);
84     guessLinearLayouts[1] =
85         (LinearLayout) view.findViewById(R.id.row2LinearLayout);
86     guessLinearLayouts[2] =
87         (LinearLayout) view.findViewById(R.id.row3LinearLayout);
88     answerTextView = (TextView) view.findViewById(R.id.answerTextView);
89
90     // Настройка слушателей для кнопок ответов
91     for (LinearLayout row : guessLinearLayouts)
92     {
93         for (int column = 0; column < row.getChildCount(); column++)
94         {
95             Button button = (Button) row.getChildAt(column);
96             button.setOnClickListener(guessButtonListener);
97         }
98     }
99
100    // Назначение текста questionNumberTextView
```

```
101     questionNumberTextView.setText(  
102         getResources().getString(R.string.question, 1, FLAGS_IN QUIZ));  
103     return view; // Возвращает представление фрагмента для отображения  
104 } // Конец метода onCreateView  
105
```

В строках 67–68 создаются объекты `ArrayList<String>`, в которых хранятся имена файлов с изображениями флагов выбранных регионов и названия стран для текущей серии флагов соответственно. В строке 69 создается объект `SecureRandom` для генерирования случайного набора флагов и кнопок с вариантами. В строке 70 создается объект-обработчик `Handler`, который обеспечивает двухсекундную задержку перед отображением следующего флага после того, как пользователь правильно выберет страну для текущего флага.

В строках 73–74 происходит динамическая загрузка анимации, применяемой к флагу при неправильном ответе. Статический метод `loadAnimation` класса `AnimationUtils` загружает анимацию из XML-файла, представленного константой `R.anim.incorrect_shake`. Первый аргумент обозначает объект `Context`, содержащий ресурсы, к которым применяется анимация, — унаследованный метод `getActivity` возвращает объект `Activity`, управляющий этим объектом `Fragment`. Класс `Activity` является непрямым субклассом `Context`. В строке 75 метод `setRepeatCount` класса `Animation` задает количество повторений анимации.

Строки 78–88 получают ссылки на различные компоненты GUI, с которыми будут выполняться операции на программном уровне. Строки 91–98 последовательно получают каждую кнопку из трех компонентов `guessLinearLayout` и регистрируют `guessButtonListener` (раздел 5.6.9) как слушателя `OnClickListener`.

Строки 101–102 заполняют `questionNumberTextView` текстом строки, возвращаемой статическим методом `format` класса `String`. В первом аргументе `format` передается строковый ресурс `R.string.question` — форматная строка с占олнителями для двух целочисленных значений (раздел 5.4.2). Унаследованный от `Fragment` метод `getResources` возвращает объект `Resources` (пакет `android.content.res`), который используется для загрузки ресурсов. Затем мы вызываем метод `getString` этого объекта для загрузки ресурса `R.string.question`, представляющего строку `Question %1$d of %2$d`.

Строка 103 возвращает графический интерфейс `QuizFragment`.

5.6.4. Метод updateGuessRows

Метод `updateGuessRows` (листинг 5.11) вызывается из объекта `MainActivity` при запуске приложения, а также каждый раз, когда пользователь изменяет количество вариантов ответа, отображаемых для каждого флага. Строки 110–111 используют аргумент `SharedPreferences` для получения строки, соответствующей ключу `MainActivity.CHICES`, — константы с именем, под которым `SettingsFragment` хранит количество отображаемых вариантов ответа. Страна 112 преобразует значение

параметра к типу `int` и делит его на три, чтобы вычислить значение `guessRows`, определяющее количество отображаемых компонентов `guessLinearLayout` (по одному на каждые три кнопки). Затем в строках 115–116 скрываются все компоненты `guessLinearLayout`, чтобы строки 119–120 могли отобразить правильное количество `guessLinearLayout` на основании значения `guessRows`.

Листинг 5.11. Метод updateGuessRows класса QuizFragment

```

106     // Обновление guessRows по данным из SharedPreferences
107     public void updateGuessRows(SharedPreferences sharedpreferences)
108     {
109         // Получить количество отображаемых вариантов
110         String choices =
111             sharedpreferences.getString(MainActivity.CHICES, null);
112         guessRows = Integer.parseInt(choices) / 3;
113
114         // Скрыть все компоненты LinearLayout с кнопками вариантов
115         for (LinearLayout layout : guessLineareLayouts)
116             layout.setVisibility(View.INVISIBLE);
117
118         // Вывести нужное количество компонентов LinearLayout
119         for (int row = 0; row < guessRows; row++)
120             guessLineareLayouts[row].setVisibility(View.VISIBLE);
121     }
122

```

5.6.5. Метод updateRegions

Метод `updateRegions` (листинг 5.12) вызывается из объекта `MainActivity` при запуске приложения, а также каждый раз, когда пользователь изменяет набор регионов. В строках 126–127 аргумент `SharedPreferences` используется для получения имен всех включенных регионов в виде коллекции `Set<String>`. Константа `MainActivity.REGIONS` содержит имя, под которым в объекте `SettingsFragment` хранится список включаемых регионов.

Листинг 5.12. Метод updateregions класса QuizFragment

```

123     // Обновление выбранных регионов по данным из SharedPreferences
124     public void updateRegions(SharedPreferences sharedpreferences)
125     {
126         regionsSet =
127             sharedpreferences.getStringSet(MainActivity.REGIONS, null);
128     }
129

```

5.6.6. Метод resetQuiz

Метод `resetQuiz` (листинг 5.13) настраивает и запускает викторину. Напомним, что изображения, используемые в игре, хранятся в папке `assets` приложения. Чтобы

обратиться к содержимому этой папки, метод получает объект `AssetManager` приложения (строка 134) вызовом метода `getAssets` родительской активности. Затем строка 135 очищает `fileNameList`, чтобы подготовиться к загрузке имен файлов изображений только для географических регионов, включенных в викторину. Цикл в строках 140–147 перебирает все включенные регионы. Для каждого региона вызывается метод `list` класса `AssetManager` (строка 143) для получения массива имен файлов с изображениями флагов, который сохраняется в строковом массиве `paths`. В строках 145–146 из имени файла удаляется расширение `.png`, а оставшиеся имена помещаются в `fileNameList`. Метод `list` класса `AssetManager` выдает исключения `IOException`, которые являются *проверяемыми* (поэтому исключение необходимо перехватить или объявить). Если исключение происходит из-за того, что приложение не может получить доступ к папке `assets`, в строках 149–152 исключение перехватывается и регистрируется для последующей отладки встроенными средствами Android. Статический метод `e` класса `Log` используется для регистрации сообщений об ошибках в журнале. Полный список методов `Log` доступен по адресу

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

Листинг 5.13. Метод resetQuiz класса QuizFragment

```
130 // Настройка и запуск следующей серии вопросов
131 public void resetQuiz()
132 {
133     // Использование AssetManager для получения имен файлов изображений
134     AssetManager assets = getActivity().getAssets();
135     fileNameList.clear(); // Пустой список имен файлов изображений
136
137     try
138     {
139         // Перебор всех регионов
140         for (String region : regionsSet)
141         {
142             // Получение списка всех файлов изображений для региона
143             String[] paths = assets.list(region);
144
145             for (String path : paths)
146                 fileNameList.add(path.replace(".png", ""));
147         }
148     }
149     catch (IOException exception)
150     {
151         Log.e(TAG, "Error loading image file names", exception);
152     }
153
154     correctAnswers = 0; // Сброс количества правильных ответов
155     totalGuesses = 0; // Сброс общего количества попыток
156     quizCountriesList.clear(); // Очистка предыдущего списка стран
157
158     int flagCounter = 1;
159     int numberOfflags = fileNameList.size();
160
161     // Добавление FLAGS_IN QUIZ случайных имен файлов в quizCountriesList
```

```

162     while (flagCounter <= FLAGS_IN QUIZ)
163     {
164         int randomIndex = random.nextInt(numberOfFlags);
165
166         // Получение случайного имени файла
167         String fileName = fileNameList.get(randomIndex);
168
169         // Если регион включен, но еще не был выбран
170         if (!quizCountriesList.contains(fileName))
171         {
172             quizCountriesList.add(fileName); // Добавить файл в список
173             ++flagCounter;
174         }
175     }
176
177     loadNextFlag(); // Запустить викторину загрузкой первого флага
178 } // Конец метода resetQuiz
179

```

Затем в строках 154–156 обнуляются счетчики правильных попыток, сделанных пользователем (`correctAnswers`), и общего количества попыток (`totalGuesses`), а также очищается список `quizCountriesList`.

В строках 162–175 в список `quizCountriesList` добавляются `FLAGS_IN QUIZ` (10) случайно выбранных имен файлов. Мы получаем общее количество флагов, после чего генерируем случайный индекс в диапазоне от 0 до количества флагов, уменьшенного на 1. Сгенерированный индекс используется для выбора одного имени файла из `fileNameList`. Если `quizCountriesList` еще не содержит это имя, оно добавляется в `quizCountriesList` с увеличением счетчика `flagCounter`. Процесс повторяется до тех пор, пока не будут выбраны `FLAGS_IN QUIZ` уникальных имен файлов. Затем в строке 177 вызывается метод `loadNextFlag` (листинг 5.14) для загрузки первого флага.

5.6.7. Метод `loadNextFlag`

Метод `loadNextFlag` (листинг 5.14) загружает и отображает следующий флаг и соответствующий набор кнопок с вариантами ответа. В списке `quizCountriesList` хранятся имена файлов в формате

регион-страна

без расширения `.png`. Если `regionName` или `countryName` содержат несколько слов, то слова разделяются символом подчеркивания (`_`).

Листинг 5.14. Метод `loadNextFlag` класса QuizFragment

```

180     // Загрузка следующего флага после правильного ответа
181     private void loadNextFlag()
182     {
183         // Получение имени файла следующего флага и удаление его из списка
184         String nextImage = quizCountriesList.remove(0);

```

```
185     correctAnswer = nextImage; // Обновление правильного ответа
186     answerTextView.setText(""); // Очистка answerTextView
187
188     // Отображение номера текущего вопроса
189     questionNumberTextView.setText(
190         getResources().getString(R.string.question,
191             (correctAnswers + 1), FLAGS_IN QUIZ));
192
193     // Извлечение региона из имени следующего изображения
194     String region = nextImage.substring(0, nextImage.indexOf('-'));
195
196     // Использование AssetManager для загрузки следующего изображения
197     AssetManager assets = getActivity().getAssets();
198
199     try
200     {
201         // Получение объекта InputStream для ресурса следующего флага
202         InputStream stream =
203             assets.open(region + «/» + nextImage + «.png»);
204
205         // Загрузка графики в виде объекта Drawable и вывод на flagImageView
206         Drawable flag = Drawable.createFromStream(stream, nextImage);
207         flagImageView.setImageDrawable(flag);
208     }
209     catch (IOException exception)
210     {
211         Log.e(TAG, "Error loading " + nextImage, exception);
212     }
213
214     Collections.shuffle(fileNameList); // Перестановка имен файлов
215
216     // Помещение правильного ответа в конец fileNameList
217     int correct = fileNameList.indexOf(correctAnswer);
218     fileNameList.add(fileNameList.remove(correct));
219
220     // Добавление 3, 6 или 9 кнопок в зависимости от значения guessRows
221     for (int row = 0; row < guessRows; row++)
222     {
223         // Размещение кнопок в currentTableRow
224         for (int column = 0;
225             column < guessLinearLayouts[row].getChildCount(); column++)
226         {
227             // Получение ссылки на Button
228             Button newGuessButton =
229                 (Button) guessLinearLayouts[row].getChildAt(column);
230             newGuessButton.setEnabled(true);
231
232             // Назначение названия страны текстом newGuessButton
233             String fileName = fileNameList.get((row * 3) + column);
234             newGuessButton.setText(getCountryName(fileName));
235         }
236     }
237
238     // Случайная замена одной кнопки правильным ответом
239     int row = random.nextInt(guessRows); // Выбор случайной строки кнопок
```

```
240     int column = random.nextInt(3); // Выбор случайного столбца
241     LinearLayout randomRow = guessLinearLayouts[row]; // Получение строки
242     String countryName = getCountryName(correctAnswer);
243     ((Button) randomRow.getChildAt(column)).setText(countryName);
244 } // Конец метода loadNextFlag
245
```

Строка 184 удаляет первое имя из `quizCountriesList` и сохраняет его в `nextImage`. Имя также сохраняется в `correctAnswer` для последующей проверки правильности ответа. Затем приложение очищает `answerTextView` и выводит следующий номер вопроса в поле `questionNumberTextView` (строки 189–191) с использованием отформатированного строкового ресурса `R.string.question`.

Строка 194 извлекает из `nextImage` регион, который должен использоваться как имя вложенной папки из `assets`, из которой будет загружаться изображение. Затем мы получаем объект `AssetManager` и используем его в команде `try` для открытия потока `InputStream` (пакет `java.io`), чтобы прочитать байты из файла с изображением флага. Поток передается в аргументе статического метода `createFromStream` класса `Drawable`, создающим объект `Drawable` (пакет `android.graphics.drawable`). Объект `Drawable` назначается визуальным источником данных компонента `flagImageView` вызовом метода `setImageDrawable`. Если происходит исключение, оно регистрируется для отладки (строка 211).

Затем строка 214 переставляет элементы `fileNameList` в случайном порядке, а строки 217–218 находят правильный ответ `correctAnswer` и перемещают его в конец `fileNameList` — позднее этот ответ будет случайно помещен на одну из кнопок с вариантами.

Цикл в строках 221–236 перебирает кнопки в компонентах `guessLinearLayout` для текущего количества панелей с кнопками `guessRows`. Для каждой кнопки:

- ❑ строки 228–229 получают ссылку на следующую кнопку;
- ❑ строка 230 разблокирует кнопку;
- ❑ строка 233 получает имя файла с флагом из `fileNameList`;
- ❑ строка 234 заполняет текст кнопки названием страны, возвращаемым методом `getCountryName` (раздел 5.6.8);
- ❑ строки 239–243 выбирают случайную строку (на основании текущего значения `guessRows`) и столбец кнопок, после чего задают текст соответствующей кнопки.

5.6.8. Метод `getCountryName`

Метод `getCountryName` (листинг 5.15) выделяет название страны из имени файла с изображением. Сначала выделяется подстрока, которая начинается с дефиса, отделяющего регион от названия страны. Затем вызывается метод `replace` класса `String` для замены подчеркиваний (`_`) пробелами.

Листинг 5.15. Метод getCountryName класса QuizFragment

```

246     // Метод выделяет из имени файла название страны
247     private String getCountryName(String name)
248     {
249         return name.substring(name.indexOf('-') + 1).replace('_', ' ');
250     }
251

```

5.6.9. Анонимный внутренний класс, реализующий интерфейс OnClickListener

В строках 91–98 (листинг 5.10) объект `guessButtonListener` (листинг 5.16) регистрируется как обработчик события для каждой кнопки с вариантом ответа. Переменная экземпляра `guessButtonListener` содержит ссылку на объект анонимного внутреннего класса, реализующий интерфейс `OnClickListener` для обработки событий кнопок. Метод получает нажатую кнопку в параметре `v`. Мы получаем текст кнопки (строка 259) и название страны (строка 260), после чего увеличиваем `totalGuesses`.

Если ответ правильный (строка 263), мы увеличиваем счетчик `correctAnswers`. Затем компонент `answerTextView` заполняется текстом с названием страны, а в качестве цвета шрифта выбирается цвет, представленный константой `R.color.correct_answer` (зеленый). После этого вызывается вспомогательный метод `disableButtons` (раздел 5.6.10), блокирующий все кнопки ответов.

Листинг 5.16. Анонимный внутренний класс, реализующий OnClickListener

```

252     // Вызывается при касании кнопки с ответом
253     private OnClickListener guessButtonListener = new OnClickListener()
254     {
255         @Override
256         public void onClick(View v)
257         {
258             Button guessButton = ((Button) v);
259             String guess = guessButton.getText().toString();
260             String answer = getCountryName(correctAnswer);
261             ++totalGuesses; // Увеличение количества попыток
262
263             if (guess.equals(answer)) // Если ответ правильный
264             {
265                 ++correctAnswers; // Увеличить счетчик правильных ответов
266
267                 // Правильный ответ выводится зеленым шрифтом
268                 answerTextView.setText(answer + "!");
269                 answerTextView.setTextColor(
270                     getResources().getColor(R.color.correct_answer));
271
272                 disableButtons(); // Все кнопки с ответами блокируются
273

```

```
274 // Если пользователь правильно угадал FLAGS_IN QUIZ флагов
275 if (correctAnswers == FLAGS_IN QUIZ)
276 {
277     // DialogFragment для вывода статистики и начала новой серии
278     DialogFragment quizResults =
279         new DialogFragment()
280     {
281         // Создание и возвращение объекта AlertDialog
282         @Override
283         public Dialog onCreateDialog(Bundle bundle)
284         {
285             AlertDialog.Builder builder =
286                 new AlertDialog.Builder(getActivity());
287             builder.setCancelable(false);
288
289             builder.setMessage(
290                 getResources().getString(R.string.results,
291                 totalGuesses, (1000 / (double) totalGuesses)));
292
293             // Кнопка "Reset Quiz"
294             builder.setPositiveButton(R.string.reset_quiz,
295                 new DialogInterface.OnClickListener()
296                 {
297                     public void onClick(DialogInterface dialog,
298                         int id)
299                     {
300                         resetQuiz();
301                     }
302                 } // Конец анонимного внутреннего класса
303             ); // Конец вызова setPositiveButton
304
305             return builder.create(); // Возвращает AlertDialog
306         } // Конец метода onCreateDialog
307     }; // Конец анонимного внутреннего класса DialogFragment
308
309     // Использование FragmentManager для вывода DialogFragment
310     quizResults.show(getFragmentManager(), "quiz results");
311 }
312 else // Ответ правильный, но вопросы еще остались
313 {
314     // Загрузить следующий флаг после односекундной задержки
315     handler.postDelayed(
316         new Runnable()
317         {
318             @Override
319             public void run()
320             {
321                 loadNextFlag();
322             }
323         }, 2000); // двухсекундная задержка (2000 миллисекунд)
324     }
325 }
326 else // Неправильный ответ
327 {
328     flagImageView.startAnimation(shakeAnimation); // Анимация
```

```

329             // Текст "Incorrect!" выводится красным шрифтом
330             answerTextView.setText(R.string.incorrect_answer);
331             answerTextView.setTextColor(
332                 getResources().getColor(R.color.incorrect_answer));
333             guessButton.setEnabled(false); // Неправильный ответ блокируется
334         }
335     }
336 }
337 }; // Конец guessButtonListener
338

```

Если значение `correctAnswers` равно `FLAGS_IN QUIZ` (строка 275), значит викторина завершена. В строках 278–307 создается новый анонимный внутренний класс, расширяющий класс `DialogFragment`, который будет использоваться для вывода результатов. Метод `onCreateDialog` класса `DialogFragment` использует `AlertDialog.Builder` для настройки и создания `AlertDialog`, после чего возвращает его. Когда пользователь касается кнопки `Reset Quiz` в диалоговом окне, вызывается метод `resetQuiz`, который запускает новую игру (строка 300). Чтобы отобразить фрагмент `DialogFragment`, строка 310 вызывает его метод `show`, передавая в аргументах объект `FragmentManager`, полученный при вызове `getFragmentManager` и `String`. Второй аргумент может использоваться с методом `getFragmentByTag` класса `FragmentManager` для получения ссылки на `DialogFragment` в будущем — в нашем приложении эта возможность не используется.

Если значение `correctAnswers` меньше `FLAGS_IN QUIZ`, то в строках 315–323 вызывается метод `postDelayed` объекта `handler`. Первый аргумент определяет анонимный внутренний класс, реализующий интерфейс `Runnable`, — он представляет задачу (`loadNextFlag`), которая должна быть выполнена через заданное количество миллисекунд. Второй аргумент определяет задержку в миллисекундах (2000). Если ответ неправилен, то строка 328 вызывает метод `startAnimation` компонента `flagImageView` для воспроизведения анимации `shakeAnimation`, загруженной в методе `onCreateView`. Также компонент `answerTextView` настраивается для вывода текста "Incorrect!" красным шрифтом (строки 331–333), а кнопка, соответствующая неправильному ответу, блокируется.

5.6.10. Метод disableButtons

Метод `disableButtons` (листинг 5.17) перебирает кнопки с вариантами ответов и блокирует их.

Листинг 5.17. Метод disableButtons класса QuizFragment

```

339     // Вспомогательный метод, блокирующий все кнопки ответов
340     private void disableButtons()
341     {
342         for (int row = 0; row < guessRows; row++)
343         {
344             LinearLayout guessRow = guessLinearLayouts[row];

```

```

345         for (int i = 0; i < guessRow.getChildCount(); i++)
346             guessRow.getChildAt(i).setEnabled(false);
347     }
348 }
349 } // Конец класса FlagQuiz

```

5.7. Класс SettingsFragment

Класс `SettingsFragment` (листинг 5.18) расширяет класс `PreferenceFragment`, представляющий средства для управления настройками приложения. Переопределенный метод `onCreate` (строки 11–16) вызывается при создании объекта `SettingsFragment` — либо активностью `SettingsActivity` при выполнении приложения в портретной ориентации, либо `MainActivity` при выполнении на планшете в альбомной ориентации. В строке 15 унаследованный от `PreferenceFragment` метод `addPreferencesFromResource` используется для построения графического интерфейса управления конфигурацией. В аргументе передается идентификатор ресурса файла `preferences.xml`, созданного в разделе 5.4.10.

Листинг 5.18. Субкласс `PreferenceFragment` для управления настройками приложения

```

1 // SettingsFragment.java
2 // Субкласс PreferenceFragment для управления настройками приложения
3 package com.deitel.flagquiz;
4
5 import android.os.Bundle;
6 import android.preference.PreferenceFragment;
7
8 public class SettingsFragment extends PreferenceFragment
9 {
10    // Создает GUI на основе файла preferences.xml из папки res/xml
11    @Override
12    public void onCreate(Bundle savedInstanceState)
13    {
14        super.onCreate(savedInstanceState);
15        addPreferencesFromResource(R.xml.preferences); // Загрузка из XML
16    }
17 } // Конец класса SettingsFragment

```

5.8. Класс SettingsActivity

Класс `SettingsActivity` (листинг 5.19) управляет `SettingsFragment` при запуске приложения в портретной ориентации. Чтобы создать этот класс, щелкните правой кнопкой мыши на пакете (`com.deitel.flagquiz`) и выберите команду `New > Class`, чтобы открыть диалоговое окно `New Java Class`. Введите в поле `Name` имя нового класса `SettingsActivity`, в поле `Superclass` — имя суперкласса `android.app.Activity` и нажмите `Finish`.

Переопределенный метод `onCreate` (строки 11–16) вызывает метод `setContentView` класса `Activity`, чтобы заполнить графический интерфейс, определяемый в файле `activity_settings.xml` (раздел 5.4.6), представленный ресурсом `R.layout.activity_settings`.

Листинг 5.19. Активность для отображения `SettingsFragment` на телефоне

```
1 // SettingsActivity.java
2 // Активность для отображения SettingsFragment на телефоне
3 package com.deitel.flagquiz;
4
5 import android.app.Activity;
6 import android.os.Bundle;
7
8 public class SettingsActivity extends Activity
9 {
10     // Использование FragmentManager для отображения SettingsFragment
11     @Override
12     protected void onCreate(Bundle savedInstanceState)
13     {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_settings);
16     }
17 } // Конец класса SettingsActivity
```

5.9. AndroidManifest.xml

Каждая активность в приложении должна быть объявлена в файле `AndroidManifest.xml` приложения; в противном случае Android не будет знать о том, что активность существует, и не сможет запустить ее. При создании приложения среда разработки включила объявление класса `MainActivity` в `AndroidManifest.xml`. Чтобы добавить объявление активности `SettingsActivity`, выполните следующие действия.

1. Откройте `AndroidManifest.xml` и щелкните на вкладке `Application` в нижней части окна редактора манифеста.
2. В разделе `Application Nodes` нажмите `Add...`, выберите в открывшемся диалоговом окне `Activity` и нажмите `OK`.
3. В разделе `Application Nodes` выделите узел новой активности, чтобы вывести ее атрибуты в разделе `Attributes for Activity`.
4. В поле `Name` введите текст `.SettingsActivity`. Точка (.) перед `SettingsActivity` является сокращенным обозначением имени пакета приложения (`com.deitel.flagquiz`).
5. В поле `Label` введите значение `@string/settings_activity` — этот строковый ресурс отображается на панели действий во время выполнения `SettingsActivity`.

За подробной информацией о файле манифеста обращайтесь по адресу
<http://developer.android.com/guide/topics/manifest/manifest-intro.html>

5.10. Резюме

В этой главе было создано приложение Flag Quiz для проверки знания пользователем флагов различных государств. Ключевой особенностью этой главы стало использование фрагментов для формирования частей графического интерфейса активности. Мы использовали две активности для отображения фрагментов QuizFragment и SettingsFragment при запуске приложения в портретной ориентации и одну активность для отображения обоих фрагментов при выполнении приложения на планшете в альбомной ориентации — таким образом обеспечивается более эффективное использование пространства экрана. Субкласс PreferenceFragment помогает автоматизировать процесс хранения и загрузки настроек приложения, а субкласс DialogFragment отображает диалоговое окно AlertDialog. Мы рассмотрели основные стадии жизненного цикла Fragment и показали, как при помощи FragmentManager получить ссылку на фрагмент для выполнения с ним операций в программе.

В портретной ориентации меню действий приложения использовалось для отображения активности SettingsActivity, содержащей SettingsFragment. Для запуска SettingsActivity использовался явный интент.

Мы показали, как при помощи объекта Android WindowManager получить объект Display, чтобы определить, выполняется ли приложение на планшете в альбомной ориентации. В этом случае меню не отображается, потому что содержимое SettingsFragment уже находится на экране.

Также в этой главе было показано, как управлять большим количеством графических ресурсов с использованием вложенных папок в папке assets приложения и как обращаться к этим ресурсам через AssetManager. Мы построили объект Drawable на основе байтов изображения, для этого мы прочитали их из InputStream, а затем отобразили Drawable в ImageView.

Мы рассмотрели другие папки из папки res приложения — menu для хранения файлов ресурсов меню, anim для хранения файлов ресурсов анимации и xml для хранения файлов с разметкой XML. Также вы узнали, как использовать квалификиаторы при создании папки для хранения макета, который должен использоваться только на больших устройствах в альбомной ориентации.

Временные окна Toast использовались для вывода сообщений о второстепенных ошибках или информации, которая должна на непродолжительное время появляться на экране. Чтобы следующий флаг выводился после короткой задержки, мы использовали объект Handler, выполняющий задачу Runnable по истечении заданного промежутка в миллисекундах. Вы узнали, что задача Runnable выполняется в программном потоке, создавшем Handler (UI-поток в этом приложении).

Мы определили анимацию в файле XML и применили ее к компоненту `ImageView` приложения, если пользователь выбрал неправильный ответ. Вы научились сохранять информацию об исключениях для последующей диагностики с использованием встроенного механизма Android и класса `Log`. Мы также использовали вспомогательные классы и интерфейсы из пакета `java.util`, включая `List`, `ArrayList`, `Collections` и `Set`.

В главе 6 мы создадим приложение Cannon Game. При его создании будет использована многопоточность и покадровая анимация, выполняться обработка жестов. Вы узнаете, как создать цикл с максимально быстрым обновлением экрана, чтобы анимации воспроизводились плавно, а с точки зрения пользователя приложение работало с постоянной скоростью независимо от скорости процессора конкретного устройства. Также будет рассмотрен простой механизм выявления коллизий.

6 Приложение Cannon Game

Прослушивание событий касания, покадровая анимация, графика, звук, программные потоки, SurfaceView и SurfaceHolder

В этой главе...

- Создание игры, несложной в реализации и интересной
- Создание субкласса `SurfaceView` и его использование для отображения игровой графики в отдельном программном потоке
- Создание графики с использованием `Paints` И `Canvas`
- Переопределение метода `onTouchEvent` класса `View` для обработки событий касания
- Простое обнаружение столкновений
- Добавление звука в приложение с использованием `SoundPool` И `AudioManager`
- Переопределение методов «жизненного цикла» `onPause` И `onDestroy` класса `Fragment`

6.1. Введение

В игре Cannon Game требуется разрушить состоящую из семи частей мишень в течение 10 секунд, отведенных на игру (рис. 6.1). Игра включает четыре визуальных компонента — *пушку*, управляемую пользователем, *пушечное ядро*, *мишень* и *блок*, защищающий мишень. Чтобы навести пушку на цель, коснитесь пальцем экрана. Пушка нацелится в точку, в которой вы коснулись экрана пальцем, и выстрелит по прямой в указанном направлении. В конце игры приложение отобразит диалоговое окно `AlertDialog`, с помощью которого можно узнать, выиграли вы или проиграли, а также количество сделанных выстрелов и время игры (рис. 6.2).

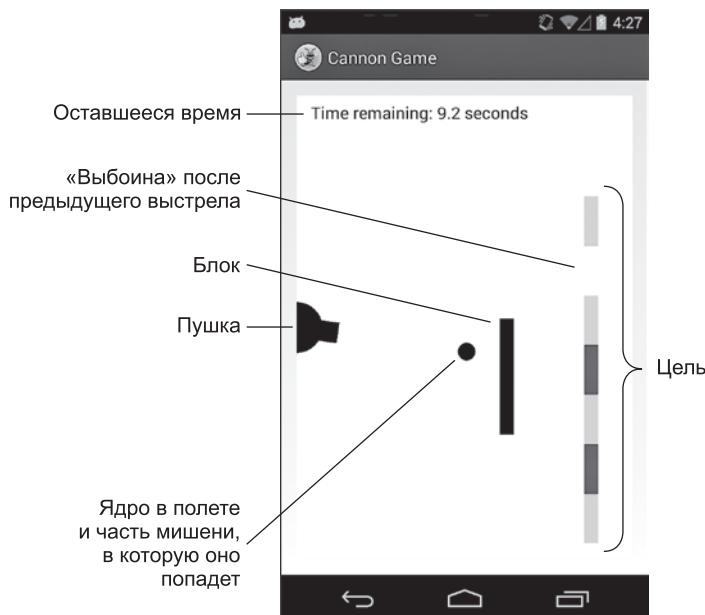


Рис. 6.1. Приложение Cannon Game

В начале игры каждому пользователю выделяются 10 секунд игрового времени. При каждом попадании в секцию мишени к лимиту времени добавляются три секунды, а при каждом попадании в блок вычитаются две секунды. Игра считается выигранной, если были разрушены все секции мишени до истечения лимита времени. Если значение таймера уменьшается до нуля до разрушения мишени, вы проиграете.

После выстрела из пушки приложение воспроизводит звук выстрела. Мишень состоит из семи секций. После попадания пушечного ядра в секцию мишени раздается звук разбивающегося стекла, а сама секция исчезает с экрана. Если ядро попадает в блок, раздается звук удара, а ядро отскакивает назад. При этом блок не разрушается. Мишень и блок перемещаются по вертикали с разной скоростью, а направление перемещения меняется после соприкосновения с верхней или нижней частью экрана.

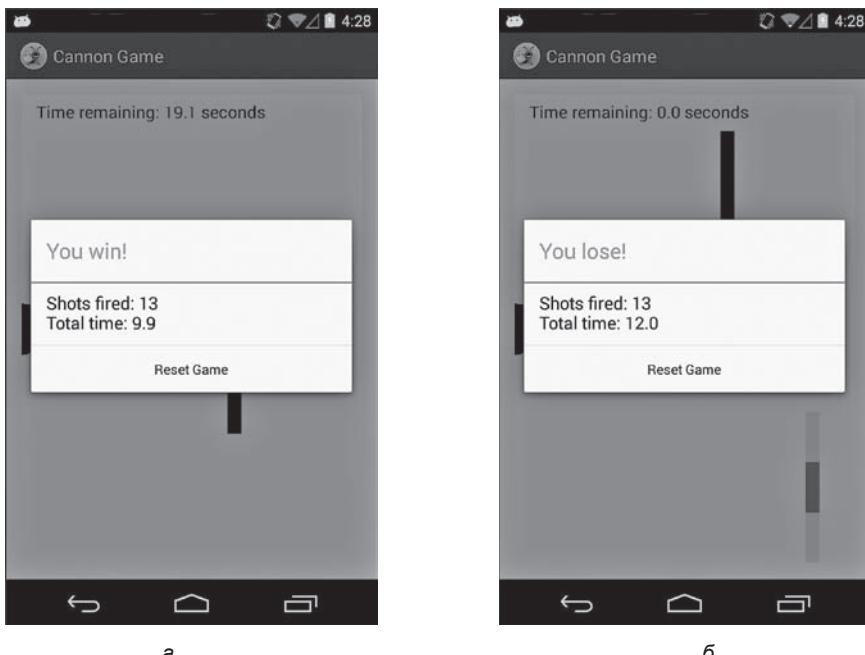


Рис. 6.2. Диалоговое окно AlertDialog приложения Cannon Game с сообщениями о выигрыше и проигрыше: *а* — окно AlertDialog после разрушения всех семи секций мишени; *б* — окно AlertDialog выводится в том случае, если время истекло до разрушения всех секций мишени

[Примечание: из-за проблем быстродействия эмулятора Android это приложение следует тестировать на физическом устройстве.]

6.2. Тестирование приложения Cannon Game

Открытие и выполнение приложения

Откройте Eclipse и импортируйте проект Cannon Game. Выполните следующие действия.

1. Откройте диалоговое окно Import командой **File > Import....**
2. Импортируйте проект приложения Cannon Game. В диалоговом окне Import раскройте узел **General** и выберите параметр **Existing Projects into Workspace**. Нажмите **Next >** для перехода к шагу **Import Projects**. Убедитесь в том, что в окне установлен переключатель **Select root directory**, и нажмите **Browse...** В диалоговом окне **Browse For Folder** выберите папку **CannonGame** в папке примеров книги и нажмите **OK**. Нажмите **Finish** для импорта проекта в среду Eclipse. Проект отобразится в окне **Package Explorer**, находящемся в левой части окна Eclipse.

- Запустите приложение Cannon Game. В среде Eclipse щелкните правой кнопкой мыши на проекте CannonGame в окне Package Explorer, затем в появившемся меню выберите команды Run As ▶ Android Application.

Игра

Коснитесь экрана, чтобы навести пушку и выстрелить. Вы сможете выстрелить только в том случае, если на экране не находится другое ядро. Если приложение выполняется на экране AVD, в качестве «пальца» будет использоваться мышь. Попытайтесь уничтожить мишень как можно быстрее. Игра кончается, если таймер дойдет до нуля или будут уничтожены все секции мишеней.

6.3. Обзор применяемых технологий

В этом разделе вашему вниманию представлены несколько новых технологий, использованных при создании приложения Cannon Game. Эти технологии будут рассмотрены в порядке их описания в главе.

6.3.1. Присоединение пользовательского представления к макету

Вы можете создать пользовательское представление, расширяя класс `View` или один из его субклассов, как это делается с классом `CannonView` (раздел 6.8), расширяющим `SurfaceView` (см. далее). Чтобы добавить пользовательский компонент в XML-файл макета, необходимо указать его полностью уточненное имя (с именем пакета и именем класса), так что пользовательский класс `View` должен существовать до того, как он будет добавлен в макет.

Процесс создания класса `CannonView` и добавления его в макет описан в разделе 6.4.3.

6.3.2. Использование папки ресурсов raw

Медиафайлы (например, звуки), используемые в приложении Cannon Game, находятся в папке ресурсов приложения `res/raw`. О том, как создать эту папку, рассказано в разделе 6.4.5; далее остается перетащить мышью звуковые файлы в эту папку.

6.3.3. Методы жизненного цикла активности и фрагмента

Когда фрагмент присоединяется к активности, как это было сделано в главе 5 (и будет сделано в этой главе), его жизненный цикл связывается с жизненным циклом родительской активности. Существуют шесть методов жизненного цикла

активности, у которых имеются соответствующие методы жизненного цикла фрагмента — `onCreate`, `onStart`, `onResume`, `onPause`, `onStop` и `onDestroy`. Когда система вызывает эти методы для активности, это приводит к вызову соответствующих методов всех присоединенных фрагментов активности (а возможно, и других методов жизненного цикла фрагментов).

В данном приложении используются методы жизненного цикла фрагмента `onPause` и `onDestroy`. Метод активности `onPause` вызывается при передаче фокуса другой активности, что приводит к приостановке активности, потерявшей фокус, и переводе ее в фоновый режим. Когда активность, управляющая фрагментами, приостанавливается, вызываются методы `onPause` всех ее фрагментов. В нашем приложении `CannonView` отображается в `CannonGameFragment` (раздел 6.7). Мы переопределяем `onPause` для приостановки игры в `CannonView`, чтобы игра не продолжалась, пока приложение недоступно для пользователя (чтобы не тратить заряд батареи). У многих методов жизненного цикла активностей имеются соответствующие методы в жизненном цикле фрагментов.

При завершении активности вызывается ее метод `onDestroy`, который в свою очередь вызывают методы `onDestroy` всех фрагментов, находящихся под управлением активности. В классе `CannonFragment` этот метод используется для освобождения звуковых ресурсов `CannonView`.

Другие методы жизненного цикла будут рассмотрены по мере необходимости. За дополнительной информацией о полном жизненном цикле активности обращайтесь по адресу

<http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle>

Подробности о полном жизненном цикле фрагментов доступны по адресу

<http://developer.android.com/guide/components/fragments.html#Lifecycle>

6.3.4. Переопределение метода `onTouchEvent` класса `View`

Пользователь взаимодействует с приложением, касаясь экрана устройства. *Касание* нацеливает пушку на определенную точку экрана и производит выстрел. Чтобы организовать обработку простых событий касания `CannonView`, переопределите метод `onTouchEvent` класса `View` (раздел 6.8.13), а затем воспользуйтесь константами класса `MotionEvent` (пакет `android.view`) для определения типа происходящих событий и их последующей обработки.

6.3.5. Добавление звука с помощью `SoundPool` и `AudioManager`

Для управления звуковыми эффектами приложения используется класс `SoundPool` (пакет `android.media`), предоставляющий средства для загрузки, воспроизведения

и выгрузки звуков. Для воспроизведения используется один из нескольких аудиопотоков Android, включающих потоки для оповещений, музыки, уведомлений, телефонных звонков, системных звуков и т. д. В документации Android рекомендуется использовать для воспроизведения звука в играх *музыкальный аудиопоток*. С помощью метода `setVolumeControlStream` класса `Activity` определяется возможность управления громкостью звука в игре кнопками регулировки громкости устройства. Метод получает константу из класса `AudioManager` (пакет `android.media`), которая предоставляет доступ к управлению громкостью динамика и телефонных звонков.

6.3.6. Покадровая анимация с помощью потоков, `SurfaceView` и `SurfaceHolder`

В этом приложении анимации реализуются вручную — путем обновления элементов игры, находящихся в отдельном потоке выполнения. Для этого используется субкласс `Thread` с методом `run`, который с помощью пользовательского класса `CannonView` обновляет позиции всех элементов игры, а затем обновляет сами элементы. Метод `run` управляет *покадровой анимацией*.

Как правило, любые обновления пользовательского интерфейса должны выполняться в потоке выполнения GUI. Но в Android важно свести к минимуму объем работы, выполняемой в потоке GUI, чтобы убедиться в том, что графический интерфейс реагировал на действия пользователя и не отображал диалоговые окна ANR (Application Not Responding). Зачастую в играх задействована сложная логика, которая должна работать в отдельных потоках выполнения, иногда эти потоки требуется отображать на экране. Для подобных случаев Android поддерживает класс `SurfaceView` — субкласс `View`, на котором можно «рисовать» из произвольного потока, а потом указать, что результаты нужно отобразить в GUI-потоке. Классом `SurfaceView` можно манипулировать посредством объекта класса `SurfaceHolder`, позволяющего получить объект `Canvas`, на котором выводится графика. Класс `SurfaceHolder` также поддерживает методы, обеспечивающие потоку *монопольный* доступ к объекту `Canvas` для рисования, поскольку рисовать на `SurfaceView` может лишь один поток одновременно. Каждый субкласс `SurfaceView` должен реализовать интерфейс `SurfaceHolder.Callback`, включающий методы, которые вызываются при *создании, изменении* (размеров или ориентации) либо *уничтожении* компонента `SurfaceView`.

6.3.7. Простое обнаружение столкновений

С помощью класса `CannonView` выполняется простое *обнаружение столкновений* пушечного ядра с каким-либо краем компонента `CannonView`, с блоком или с секцией мишени. Эти методики представлены в разделе 6.8. Многие фреймворки, предназначенные для разработки игр, предоставляют более сложные механизмы

обнаружения столкновений. Сейчас разработчикам доступны многочисленные библиотеки разработки игр с открытым кодом.

6.3.8. Рисование графики с помощью Paint и Canvas

Методы класса `Canvas` (пакет `android.graphics`) применяются для рисования текста, линий и окружностей. Метод `Canvas` рисует в области объекта `Bitmap` класса `View`. Каждый метод рисования класса `Canvas` использует объект класса `Paint` (пакет `android.graphics`) для определения характеристик рисования, включая цвет, толщину линии, размер шрифта и другие параметры. Эти средства предоставляет метод `drawGameElements`, описанный в разделе 6.8. За дополнительной информацией о графических возможностях объекта `Paint` обращайтесь на сайт developer.android.com/reference/android/graphics/Paint.html

6.4. Создание графического интерфейса приложения и файлов ресурсов

В этом разделе будут созданы файлы ресурсов приложения и файл разметки `main.xml`.

6.4.1. Создание проекта

Начнем с создания нового проекта Android под названием `CannonGame`. В диалоговом окне `New Android Project` введите следующие значения.

- Application name: Cannon Game
- Project Name: Cannon Game
- Package name: com.deitel.cannongame
- Minimum Required SDK: API18: Android 4.3
- Target SDK: API19: Android 4.4
- Compile With: API19: Android 4.4
- Theme: Holo Light with Dark Action Bar

На втором шаге мастера `New Android Project` (`New Android Application`) оставьте настройки по умолчанию и нажмите кнопку `Next >`. На шаге `Configure Launcher Icon` выберите значок приложения и нажмите кнопку `Next >`. На шаге `Create Activity` выберите шаблон `Blank Activity` и нажмите кнопку `Next >`. На шаге `Blank Activity` оставьте значения по умолчанию и нажмите `Finish`, чтобы создать проект. Переключитесь в редакторе на вкладку `activity_main.xml`. В макетном редакторе выберите в раскрывающемся списке тип экрана `Nexus 4` и удалите компонент `TextView` с текстом "Hello world!".

Настройка приложения для портретной ориентации

Игра спроектирована для выполнения в портретной ориентации. Выполните действия, описанные в разделе 3.6, чтобы назначить приложению портретную ориентацию экрана.

6.4.2. Файл strings.xml

Строковые ресурсы уже создавались в предыдущих главах, поэтому сейчас мы приведем только таблицу с именами ресурсов и соответствующими значениями (табл. 6.1). Сделайте двойной щелчок на файле strings.xml из папки res/values, чтобы запустить редактор для создания этих строковых ресурсов.

Таблица 6.1. Строковые ресурсы, используемые в приложении Cannon Game

Имя ресурса	Значение
results_format	Shots fired: %1\$d\nTotal time: %2\$.1f
reset_game	Reset Game
win	You win!
lose	You lose!
time_remaining_format	Time remaining: %.1f seconds

6.4.3. Файл fragment_game.xml

Макет fragment_game.xml для фрагмента CannonGameFragment содержит компонент FrameLayout для отображения CannonView. Компонент FrameLayout предназначен для отображения только одного представления — в данном случае CannonView. В этом разделе мы создадим макет CannonGameFragment и класс CannonView. Чтобы добавить макет fragment_game.xml, выполните следующие действия.

1. Откройте узел res/layout в окне Package Explorer.
2. Щелкните правой кнопкой мыши на папке values и выберите команду New ▶ Android XML File. На экране появляется диалоговое окно New Android XML File.
3. Введите в поле File значение fragment_game.xml и нажмите Finish, чтобы завершить создание файла.
4. В разделе Root Element выберите компонент FrameLayout и нажмите Finish.
5. Перетащите из раздела Advanced палитры компонент view (со строчной буквой v) в область построения интерфейса.
6. На экране появляется диалоговое окно Choose Custom View Class. В этом диалоговом окне нажмите Create New..., чтобы вызвать диалоговое окно New Java Class.

7. В поле **Name** введите значение **CannonView**. В поле **Superclass** замените имя суперкласса **android.view.View** на **android.view.SurfaceView**. Убедитесь в том, что флажок **Constructors from superclass** установлен, и нажмите **Finish**. Среда разработки создает и открывает файл **CannonView.java**. Мы будем использовать только конструктор с двумя аргументами, поэтому два других конструктора можно удалить. Сохраните и закройте файл **CannonView.java**.
8. В файле **fragment_game.xml** выделите компонент **view1** в окне **Outline**. В разделе **Layout Parameters** окна **Properties** задайте свойствам **Width** и **Height** значение **match_parent**.
9. В окне **Outline** щелкните правой кнопкой мыши на компоненте **view1**, выберите команду **Edit ID...**, переименуйте **view1** в **cannonView** и нажмите **OK**.
10. Сохраните файл **fragment_game.xml**.

6.4.4. Файл **activity_main.xml**

Макет **activity_main.xml** активности **MainActivity** нашего приложения должен содержать только фрагмент **CannonGameFragment**. Чтобы добавить этот фрагмент в макет, выполните следующие действия.

1. Откройте файл **activity_main.xml** в макетном редакторе.
2. В разделе **Layouts** палитры перетащите фрагмент в область построения интерфейса или на узел **RelativeLayout** в окне **Outline**.
3. На экране появляется диалоговое окно **Choose Fragment Class**. Нажмите **Create New...**, чтобы открыть диалоговое окно **New Java Class**.
4. Введите в поле **Name** значение **CannonGameFragment** и замените содержимое поля **Superclass** значением **android.app.Fragment**. Нажмите **Finish**, чтобы создать класс. Среда разработки открывает файл с кодом Java, который пока можно закрыть.
5. Сохраните файл **activity_main.xml**.

6.4.5. Добавление звуков в приложение

Как упоминалось ранее, звуковые файлы хранятся в папке **res/raw** приложения. В нашем приложении используются три звуковых файла — **blocker_hit.wav**, **target_hit.wav** и **cannon_fire.wav**, находящиеся во вложенной папке **sounds** из папки примеров. Чтобы добавить эти файлы в проект, выполните следующие действия.

1. Щелкните правой кнопкой мыши на папке **res** проекта и выберите команду **New ▶ Folder**.
2. Введите имя папки **raw** и нажмите **Finish**, чтобы создать папку.
3. Перетащите звуковые файлы в папку **res/raw**.

6.5. Класс Line

Приложение включает четыре класса:

- ❑ `Line` (листинг 6.1);
- ❑ `MainActivity` (субкласс `Activity`; раздел 6.6);
- ❑ `CannonGameFragment` (раздел 6.7);
- ❑ `CannonView` (раздел 6.8).

В этом разделе рассматривается класс `Line`, который представляет начальную и конечную точки отрезка (объекты `Point`). Объекты этого класса определяют блок и мишень. Чтобы добавить класс `Line` в проект, выполните следующие действия.

1. Раскройте узел проекта `src` в окне Package Explorer.
2. Щелкните правой кнопкой мыши на пакете (`com.deitel.cannongame`) и в контекстном меню выберите команду `New > Class` для отображения диалогового окна `New Java Class`.
3. В поле `Name` введите `Line` и нажмите `Finish`.
4. Введите код, представленный в листинге 6.4, в файл `Line.java`. Конструктор `Point` по умолчанию инициализирует открытые переменные `x` и `y` класса `Point` нулями.

Листинг 6.1. Класс `Line` определяет отрезок прямой по двум конечным точкам

```
1 // Line.java
2 // Класс Line представляет отрезок с двумя конечными точками.
3 package com.deitel.cannongame;
4
5 import android.graphics.Point;
6
7 public class Line
8 {
9     public Point start = new Point(); // Начальная точка --(0,0) по умолчанию
10    public Point end = new Point(); // Конечная точка --(0,0) по умолчанию
11 } // Конец класса Line
```

6.6. Класс MainActivity

Класс `MainActivity` (листинг 6.2) управляет фрагментом `CannonGameFragment` приложения Cannon Game. В этом приложении переопределяется только метод `onCreate` класса `Activity`, заполняющий графический интерфейс.

Листинг 6.2. Класс `MainActivity` управляет фрагментом `CannonGameFragment`

```
1 // MainActivity.java
2 // MainActivity управляет фрагментом CannonGameFragment
3 package com.deitel.cannongame;
```

```

4
5 import android.app.Activity;
6 import android.os.Bundle;
7
8 public class MainActivity extends Activity
9 {
10     // Вызывается при первом запуске приложения
11     @Override
12     public void onCreate(Bundle savedInstanceState)
13     {
14         super.onCreate(savedInstanceState); // Вызов версии onCreate суперкласса
15         setContentView(R.layout.activity_main); // Заполнение макета
16     }
17 } // Конец класса MainActivity

```

6.7. Класс CannonGameFragment

Класс `CannonGameFragment` (листинг 6.3) переопределяет четыре метода `Fragment`:

- ❑ `onCreateView` (строки 17–28) — как вы узнали в разделе 5.3.3, этот метод вызывается после метода `onCreate` класса `Fragment` для построения и возвращения компонента `View`, содержащего графический интерфейс фрагмента. В строках 22–23 происходит заполнение графического интерфейса. Стока 26 получает ссылку на компонент `CannonView` класса `CannonGameFragment`, чтобы вызывать его методы в программе;
- ❑ `onActivityCreated` (строки 31–38) — метод вызывается после создания управляющей активности фрагмента. В строке 37 вызывается метод `setVolumeControlStream` класса `Activity`, чтобы громкостью звука в игре можно было управлять кнопками устройства;
- ❑ `onPause` (строки 41–46) — при переводе `MainActivity` в фоновый режим (а следовательно, приостановке) выполняется метод `onPause` фрагмента `CannonGameFragment`. В строке 45 вызывается метод `stopGame` компонента `CannonView` (раздел 6.8.11) для остановки покадровой анимации;
- ❑ `onDestroy` (строки 49–54) — при уничтожении активности `MainActivity` ее метод `onDestroy` вызывает метод `onDestroy` фрагмента `CannonGameFragment`. В строке 46 вызывается метод `releaseResources` компонента `CannonView` (раздел 6.8.11) для освобождения звуковых ресурсов.

Листинг 6.3. Класс `CannonGameFragment` создает и управляет представлением `CannonView`

```

1 // CannonGameFragment.java
2 // Класс CannonGameFragment создает и управляет CannonView
3 package com.deitel.cannongame;
4
5 import android.app.Fragment;
6 import android.media.AudioManager;
7 import android.os.Bundle;

```

```
8 import android.view.LayoutInflater;
9 import android.view.View;
10 import android.view.ViewGroup;
11
12 public class CannonGameFragment extends Fragment
13 {
14     private CannonView cannonView; // Пользовательское представление для игры
15
16     // Вызывается при создании представления фрагмента
17     @Override
18     public View onCreateView(LayoutInflater inflater, ViewGroup container,
19         Bundle savedInstanceState)
20     {
21         super.onCreateView(inflater, container, savedInstanceState);
22         View view =
23             inflater.inflate(R.layout.fragment_game, container, false);
24
25         // Получение CannonView
26         cannonView = (CannonView) view.findViewById(R.id.cannonView);
27         return view;
28     }
29
30     // Настройка управления громкостью при создании активности
31     @Override
32     public void onActivityCreated(Bundle savedInstanceState)
33     {
34         super.onActivityCreated(savedInstanceState);
35
36         // Разрешить использование кнопок управления громкостью
37         getActivity().setVolumeControlStream(AudioManager.STREAM_MUSIC);
38     }
39
40     // При приостановке MainActivity игра завершается
41     @Override
42     public void onPause()
43     {
44         super.onPause();
45         cannonView.stopGame(); // Завершение игры
46     }
47
48     // При приостановке MainActivity освобождаются ресурсы
49     @Override
50     public void onDestroy()
51     {
52         super.onDestroy();
53         cannonView.releaseResources();
54     }
55 } // Конец класса CannonGameFragment
```

6.8. Класс CannonView

Класс `CannonView` (листинги 6.4–6.17), являющийся субклассом `View`, реализует логику приложения Cannon Game и выводит игровые объекты на экран.

6.8.1. Команда package и команды import

В листинге 6.4 представлены команды `package` и `import` класса `CannonView`. В разделе 6.3 рассматриваются важнейшие новые классы и интерфейсы, используемые классом `CannonView`. В листинге 6.4 они выделены жирным шрифтом.

Листинг 6.4. Команды package и import класса CannonView

```

1 // CannonView.java
2 // Отображение и управление игрой Cannon Game
3 package com.deitel.cannongame;
4
5 import android.app.Activity;
6 import android.app.AlertDialog;
7 import android.app.Dialog;
8 import android.app.DialogFragment;
9 import android.content.Context;
10 import android.content.DialogInterface;
11 import android.graphics.Canvas;
12 import android.graphics.Color;
13 import android.graphics.Paint;
14 import android.graphics.Point;
15 import android.media.AudioManager;
16 import android.media.SoundPool;
17 import android.os.Bundle;
18 import android.util.AttributeSet;
19 import android.util.Log;
20 import android.util.SparseIntArray;
21 import android.view.MotionEvent;
22 import android.view.SurfaceHolder;
23 import android.view.SurfaceView;
24

```

6.8.2. Переменные экземпляров и константы

В листинге 6.5 представлены многочисленные константы и переменные экземпляров класса `CannonView`. Многие из них не требуют дополнительных объяснений, но тем не менее будут рассмотрены по мере их применения в коде.

Листинг 6.5. Поля класса CannonView

```

25 public class CannonView extends SurfaceView
26     implements SurfaceHolder.Callback
27 {
28     private static final String TAG = "CannonView"; // Для регистрации ошибок
29
30     private CannonThread cannonThread; // Управление циклом игры
31     private Activity activity; // Для вывода окна завершения игры в потоке GUI
32     private boolean dialogIsDisplayed = false;
33
34     // Константы, используемые в игре
35     public static final int TARGET_PIECES = 7; // Количество секций мишени
36     public static final int MISS_PENALTY = 2; // Штраф при промахе (с.)

```

```
37     public static final int HIT_REWARD = 3; // Прибавка при попадании (с.)
38
39     // Переменные для цикла игры и сбора статистики
40     private boolean gameOver; // Игра закончена?
41     private double timeLeft; // Оставшееся время в секундах
42     private int shotsFired; // Количество выстрелов пользователя
43     private double totalElapsed Time; // Количество прошедших секунд
44
45     // Переменные для определения блока и мишени
46     private Line blocker; // Начальная и конечная точки блока
47     private int blockerDistance; // Расстояние от блока слева
48     private int blockerBeginning; // Вертикальное смещение верхнего края блока
49     private int blockerEnd; // Вертикальное смещение нижнего края блока
50     private int initialBlockerVelocity; // Исходная скорость блока
51     private float blockerVelocity; // Множитель скорости блока во время игры
52
53     private Line target; // Начальная и конечная точки мишени
54     private int targetDistance; // Расстояние до мишени слева
55     private int targetBeginning; // Расстояние до мишени сверху
56     private double pieceLength; // Длина секции мишени
57     private int targetEnd; // Вертикальное смещение до нижнего края мишени
58     private int initialTargetVelocity; // Начальный множитель скорости мишени
59     private float targetVelocity; // Множитель скорости мишени во время игры
60
61     private int lineWidth; // Ширина цели и блока
62     private boolean[] hitStates; // Поражены все секции мишени?
63     private int targetPiecesHit; // Количество пораженных секций (из 7)
64
65     // Переменные для пушки и ядра
66     private Point cannonball; // Левый верхний угол изображения ядра
67     private int cannonballVelocityX; // Горизонтальная скорость ядра
68     private int cannonballVelocityY; // Вертикальная скорость ядра
69     private boolean cannonballOnScreen; // Ядро находится на экране?
70     private int cannonballRadius; // Радиус ядра
71     private int cannonballSpeed; // Скорость ядра
72     private int cannonBaseRadius; // Радиус основания пушки
73     private int cannonLength; // Длина ствола пушки
74     private Point barrelEnd; // Конец ствола пушки
75     private int screenWidth;
76     private int screenHeight;
77
78     // Константы и переменные для управления звуком
79     private static final int TARGET_SOUND_ID = 0;
80     private static final int CANNON_SOUND_ID = 1;
81     private static final int BLOCKER_SOUND_ID = 2;
82     private SoundPool soundPool; // Для воспроизведения звуковых эффектов
83     private SparseIntArray soundMap; // Связывание идентификаторов с SoundPool
84
85     // Переменные Paint для рисования на экране
86     private Paint textPaint; // Для вывода текста
87     private Paint cannonballPaint; // Для рисования ядра
88     private Paint cannonPaint; // Для рисования пушки
89     private Paint blockerPaint; // Для рисования блока
90     private Paint targetPaint; // Для рисования мишени
91     private Paint backgroundPaint; // Для очистки области рисования
92
```

6.8.3. Конструктор

В листинге 6.6 приведен конструктор класса `CannonView`. При заполнении объекта `View` вызывается его конструктор, в аргументах которого передаются объекты `Context` и `AttributeSet`. Первый представляет активность, отображающую фрагмент `CannonGameFragment` с `CannonView`, а второй (пакет `android.util`) содержит значения атрибутов `CannonView`, заданные в XML-документе макета. Эти аргументы передаются конструктору суперкласса (строка 96), чтобы обеспечить правильную настройку пользовательского представления значениями стандартных атрибутов `View`, указанных в XML-документе макета. В строке 97 сохраняется ссылка на `MainActivity`, которая используется в конце игры для отображения окна `AlertDialog` из GUI-потока активности.

Листинг 6.6. Конструктор `CannonView`

```

93  // Открытый конструктор
94  public CannonView(Context context, AttributeSet attrs)
95  {
96      super(context, attrs); // Вызов конструктора суперкласса
97      activity = (Activity) context; // Сохранение ссылки на MainActivity
98
99      // Регистрация слушателя SurfaceHolder.Callback
100     getHolder().addCallback(this);
101
102     // Инициализация игровых объектов Line и Point
103     blocker = new Line(); // blocker создается как объект Line
104     target = new Line(); // target создается как объект Line
105     cannonball = new Point(); // cannonball создается как объект Point
106
107     // Инициализация массива логических значений hitStates
108     hitStates = new boolean[TARGET_PIECES];
109
110     // Инициализация SoundPool для звуковых эффектов приложения
111     soundPool = new SoundPool(1, AudioManager.STREAM_MUSIC, 0);
112
113     // Создание контейнера Map и предварительная загрузка звуков
114     soundMap = new SparseIntArray(3); // Создание массива SparseIntArray
115     soundMap.put(TARGET_SOUND_ID,
116                 soundPool.load(context, R.raw.target_hit, 1));
117     soundMap.put(CANNON_SOUND_ID,
118                 soundPool.load(context, R.raw.cannon_fire, 1));
119     soundMap.put(BLOCKER_SOUND_ID,
120                 soundPool.load(context, R.raw.blocker_hit, 1));
121
122     // Создание объектов Paint для вывода текста, рисования ядра,
123     // пушки, блока и мишени; эти параметры настраиваются
124     // в методе onSizeChanged
125     textPaint = new Paint();
126     cannonPaint = new Paint();
127     cannonballPaint = new Paint();
128     blockerPaint = new Paint();
129     targetPaint = new Paint();
130     backgroundPaint = new Paint();
131 } // Конец конструктора CannonView

```

Регистрация слушателя SurfaceHolder.Callback

В строке 100 `this` (то есть объект `CannonView`) регистрируется как объект, реализующий `SurfaceHolder.Callback` для получения вызовов методов, сообщающих о создании, обновлении и уничтожении `SurfaceView`. Унаследованный от `SurfaceView` метод `getHolder` возвращает объект `SurfaceHolder` для управления `SurfaceView`, а метод `addCallback` класса `SurfaceHolder` сохраняет объект, реализующий интерфейс `SurfaceHolder.Callback`.

Создание блока, мишени и ядра

В строках 103–105 блок (`blocker`) и мишень (`target`) создаются как объекты `Line`, а ядро (`cannonball`) — как объект `Point`. Затем мы создаем массив логических значений `hitStates` для отслеживания секций мишени, которые были поражены выстрелами (а следовательно, не должны прорисовываться на экране).

Настройка SoundPool и загрузка звуков

В строках 111–120 настраивается конфигурация звуков, используемых в приложении. Сначала создается объект `SoundPool`, применяемый для загрузки и воспроизведения звуковых эффектов, используемых в приложении. Первый аргумент конструктора представляет максимальное количество звуковых потоков, которые могут воспроизводиться одновременно. В игре одновременное воспроизведение звуков не используется, поэтому этот аргумент равен 1. Второй аргумент определяет аудиопоток, который будет использоваться для воспроизведения звуков. Существуют семь аудиопотоков, идентифицируемых константами класса `AudioManager`, но в документации для класса `SoundPool` рекомендуется для воспроизведения звука в играх использовать поток воспроизведения музыки (`AudioManager.STREAM_MUSIC`). Последний аргумент представляет качество звука, хотя в документации сказано, что в данное время это значение не используется (и по умолчанию следует передавать 0).

В строке 114 создается объект `SparseIntArray` (переменная `soundMap`), связывающий целочисленные ключи с целочисленными значениями. Класс `SparseIntArray` похож на `HashMap<Integer, Integer>`, но более эффективен для небольшого количества пар «ключ/значение». В нашем примере ключи звуков (см. листинг 6.5, строки 79–81) связываются с идентификаторами загруженных звуков, представленных возвращаемыми значениями метода `load` класса `SoundPool` (вызываемого в листинге 6.6, строки 116, 118 и 120). Идентификаторы могут использоваться для воспроизведения звука (и последующего возврата ресурсов системе). Метод `load` класса `SoundPool` получает три аргумента — контент (`Context`) приложения; идентификатор загружаемого звукового файла; а также *приоритет* звука. В соответствии с документацией, описывающей этот метод, последний аргумент в настоящее время не используется, и в нем следует передавать значение 1.

Создание объектов Paint, используемых для рисования игровых элементов

В строках 124–129 создаются объекты `Paint`, используемые для рисования объектов игры. Настройка этих объектов выполняется в методе `onSizeChanged` (раздел 6.8.4),

поскольку некоторые из параметров объекта `Paint` зависят от масштаба элементов игры, основанном на размере экрана устройства.

6.8.4. Переопределение метода `onSizeChanged` класса `View`

В коде из листинга 6.7 выполняется переопределение метода `onSizeChanged` класса `View`, который вызывается в случае изменения размеров класса `View`, например при первом добавлении класса `View` в иерархию классов `View` в процессе заполнения разметки. Это приложение всегда отображается в *портретном режиме*, поэтому метод `onSizeChanged` вызывается только в том случае, если метод `onCreate` активности заполняет графический интерфейс. Этот метод получает новые значения ширины и высоты объекта `View` (при первом вызове метода старые значения ширины и высоты равны 0). Выполняемые здесь вычисления масштабируют экранные элементы на основании ширины и высоты устройства в пикселях. Значения масштабных коэффициентов подбирались «методом проб и ошибок», чтобы игровые элементы хорошо смотрелись на экране. В строках 170–175 настраиваются объекты `Paint`, определяющие характеристики графических элементов игры. После выполнения вычислений в строке 177 вызывается метод `newGame` (см. листинг 6.8).

Листинг 6.7. Переопределенный метод `onSizeChanged`

```

132 // Вызывается surfaceChanged при изменении размеров SurfaceView -
133 // например, при первом добавлении в иерархии View
134 @Override
135 protected void onSizeChanged(int w, int h, int oldw, int oldh)
136 {
137     super.onSizeChanged(w, h, oldw, oldh);
138
139     screenWidth = w; // Сохранение ширины CannonView
140     screenHeight = h; // Сохранение высоты CannonView
141     cannonBaseRadius = h / 18; // Радиус основания пушки равен
142         // 1/18 высоты экрана
143     cannonLength = w / 8; // Длина пушки равна 1/8 ширины экрана
144
145     cannonballRadius = w / 36; // Радиус ядра равен 1/36 ширины экрана
146     cannonballSpeed = w * 3 / 2; // Множитель скорости ядра
147
148     lineWidth = w / 24; // Ширина мишени и блока равна 1/24 ширины экрана
149
150     // Настройка переменных экземпляра, связанных с блоком
151     blockerDistance = w * 5 / 8; // 5/8 ширины экрана от левого края
152     blockerBeginning = h / 8; // Вертикальное смещение верха -
153         // 1/8 высоты экрана
154     blockerEnd = h * 3 / 8; // Вертикальное смещение низа -
155         // 3/8 высоты экрана
156     initialBlockerVelocity = h / 2; // Начальный множитель скорости блока
157     blocker.start = new Point(blockerDistance, blockerBeginning);
158     blocker.end = new Point(blockerDistance, blockerEnd);
159
160     // Настройка переменных экземпляра, связанных с мишенью

```

```

158     targetDistance = w * 7 / 8; // Смещение мишени слева – 7/8 ширины экрана
159     targetBeginning = h / 8; // Вертикальное смещение верха –
160                     // 1/8 высоты экрана
160     targetEnd = h * 7 / 8; // Вертикальное смещение низа – 7/8 высоты экрана
161     pieceLength = (targetEnd - targetBeginning) / TARGET_PIECES;
162     initialTargetVelocity = -h / 4; // Начальный множитель скорости мишени
163     target.start = new Point(targetDistance, targetBeginning);
164     target.end = new Point(targetDistance, targetEnd);
165
166     // Ствол пушки изначально направлен горизонтально
167     barrelEnd = new Point(cannonLength, h / 2);
168
169     // Настройка объектов Paint для рисования элементов игры
170     textPaint.setTextSize(w / 20); // Размер текста равен 1/20 ширины экрана
171     textPaint.setAntiAlias(true); // Сглаживание текста
172     cannonPaint.setStrokeWidth(lineWidth * 1.5f); // Толщина линии пушки
173     blockerPaint.setStrokeWidth(lineWidth); // Толщина линии блока
174     targetPaint.setStrokeWidth(lineWidth); // Толщина линии мишени
175     backgroundPaint.setColor(Color.WHITE); // Цвет фона
176
177     newGame(); // Настройка и запуск новой игры
178 } // Конец метода onSizeChanged
179

```

6.8.5. Метод newGame

Метод newGame (листинг 6.8) сбрасывает состояние переменных, используемых для управления игрой. Если переменная gameOver содержит true (что происходит только после завершения первой игры), строка 203 сбрасывает gameOver, а строки 204–205 создают новый поток CannonThread и запускают в нем цикл, управляющий ходом игры. Дополнительная информация приведена в разделе 6.8.14.

Листинг 6.8. Метод newGame класса CannonView

```

180     // Сброс всех экранных элементов и запуск новой игры
181     public void newGame()
182     {
183         // Всем элементам hitStates присваивается false (восстановление мишени)
184         for (int i = 0; i < TARGET_PIECES; i++)
185             hitStates[i] = false;
186
187         targetPiecesHit = 0; // Все сегменты мишени целы
188         blockerVelocity = initialBlockerVelocity; // Начальная скорость
189         targetVelocity = initialTargetVelocity; // Начальная скорость
190         timeLeft = 10; // Таймер отсчитывает 10 секунд
191         cannonballOnScreen = false; // Ядро не находится на экране
192         shotsFired = 0; // Количество сделанных выстрелов
193         totalElapsed = 0.0; // Время игры обнуляется
194
195         // Начальная и конечная точки блока и мишени
196         blocker.start.set(blockerDistance, blockerBeginning);
197         blocker.end.set(blockerDistance, blockerEnd);
198         target.start.set(targetDistance, targetBeginning);

```

```

199     target.end.set(targetDistance, targetEnd);
200
201     if (gameOver) // Начать новую игру после завершения предыдущей
202     {
203         gameOver = false; // Игра не закончена
204         cannonThread = new CannonThread(getHolder()); // Создать поток
205         cannonThread.start(); // Запустить поток для цикла игры
206     } // Конец if
207 } // Конец метода newGame
208

```

6.8.6. Метод updatePositions

Метод `updatePositions` (листинг 6.9) вызывается методом `run` объекта `CannonThread` (раздел 6.8.14) для обновления позиций элементов на экране и простого обнаружения *столкновений*. Новые положения игровых элементов вычисляются по времени (в миллисекундах), прошедшем между предыдущим и текущим кадром анимации. Таким образом игра определяет расстояние, на которое перемещается каждый элемент, в соответствии с частотой обновления изображения на экране устройства. Эти вопросы будут разобраны более подробно при рассмотрении циклов игры в разделе 6.8.14.

Листинг 6.9. Метод updatePositions класса CannonView

```

209     // Многократно вызывается CannonThread для обновления элементов игры
210     private void updatePositions(double elapsedTimeMS)
211     {
212         double interval = elapsedTimeMS / 1000.0; // Преобразовать в секунды
213
214         if (cannonballOnScreen) // Если ядро находится на экране
215         {
216             // Обновление позиции ядра
217             cannonball.x += interval * cannonballVelocityX;
218             cannonball.y += interval * cannonballVelocityY;
219
220             // Проверка столкновения с блоком
221             if (cannonball.x + cannonballRadius > blockerDistance &&
222                 cannonball.x - cannonballRadius < blockerDistance &&
223                 cannonball.y + cannonballRadius > blocker.start.y &&
224                 cannonball.y - cannonballRadius < blocker.end.y)
225             {
226                 cannonballVelocityX *= -1; // Ядро летит в обратном направлении
227                 timeLeft -= MISS_PENALTY; // Пользователь теряет секунды
228
229                 // Воспроизведение звука столкновения с блоком
230                 soundPool.play(soundMap.get(BLOCKER_SOUND_ID), 1, 1, 1, 0, 1f);
231             }
232             // Проверка столкновений со стенами
233             else if (cannonball.x + cannonballRadius > screenWidth ||
234                     cannonball.x - cannonballRadius < 0)
235         {

```

```
236         cannonballOnScreen = false; // Ядро уходит с экрана
237     }
238     // Проверка столкновений с верхней и нижней стеной
239     else if (cannonball.y + cannonballRadius > screenHeight ||
240             cannonball.y - cannonballRadius < 0)
241     {
242         cannonballOnScreen = false; // Удаление ядра с экрана
243     }
244     // Проверка столкновения ядра с мишенью
245     else if (cannonball.x + cannonballRadius > targetDistance &&
246               cannonball.x - cannonballRadius < targetDistance &&
247               cannonball.y + cannonballRadius > target.start.y &&
248               cannonball.y - cannonballRadius < target.end.y)
249     {
250         // Определение номера секции мишени (0 - верхняя)
251         int section =
252             (int) ((cannonball.y - target.start.y) / pieceLength);
253
254         // Проверить, не было ли попаданий в эту секцию
255         if ((section >= 0 && section < TARGET_PIECES) &&
256             !hitStates[section])
257         {
258             hitStates[section] = true; // Есть попадание
259             cannonballOnScreen = false; // Убрать ядро
260             timeLeft += HIT_REWARD; // Прибавить секунды
261
262             // Воспроизведение звука попадания в мишень
263             soundPool.play(soundMap.get(TARGET_SOUND_ID), 1,
264                           1, 1, 0, 1f);
265
266             // Если поражены все секции
267             if (++targetPiecesHit == TARGET_PIECES)
268             {
269                 cannonThread.setRunning(false); // Завершить поток
270                 showGameOverDialog(R.string.win); // Сообщить о победе
271                 gameOver = true;
272             }
273         }
274     }
275 }
276
277 // Обновление позиции блока
278 double blockerUpdate = interval * blockerVelocity;
279 blocker.start.y += blockerUpdate;
280 blocker.end.y += blockerUpdate;
281
282 // Обновление позиции мишени
283 double targetUpdate = interval * targetVelocity;
284 target.start.y += targetUpdate;
285 target.end.y += targetUpdate;
286
287 // Если блок достиг верха или низа, поменять направление движения
288 if (blocker.start.y < 0 || blocker.end.y > screenHeight)
289     blockerVelocity *= -1;
290
```

```
291      // Если мишень достигла верха или низа, поменять направление движения
292      if (target.start.y < 0 || target.end.y > screenHeight)
293          targetVelocity *= -1;
294
295      timeLeft -= interval; // Уменьшить оставшееся время
296
297      // Если таймер достиг нуля
298      if (timeLeft <= 0.0)
299      {
300          timeLeft = 0.0;
301          gameOver = true; // Игра закончена
302          cannonThread.setRunning(false); // Завершить поток
303          showGameOverDialog(R.string.lose); // Сообщить о проигрыше
304      }
305  } // Конец метода updatePositions
306
```

Время от последнего кадра анимации

В строке 206 выполняется преобразование времени, прошедшего с момента последнего кадра анимации (из миллисекунд в секунды). Это значение используется для изменения позиций различных элементов игры.

Проверка столкновений с блоком

В строке 214 проверяется, отображается ли пушечное ядро на экране. Если ядро отображается, его обновленная позиция вычисляется путем добавления расстояния, которое должно быть пройдено с момента последнего события таймера. Расстояние вычисляется умножением скорости на прошедшее время (строки 217–218). В строках 221–224 проверяется, столкнулось ли ядро с блоком. В приложении используется простейшая проверка столкновений, основанная на прямоугольной рамке вокруг ядра. Для столкновения ядра с блоком должны выполняться следующие четыре условия.

- ❑ Сумма координаты x пушечного ядра и радиуса ядра должна быть больше, чем расстояние от блока до левой границы экрана (`b blockerDistance`), строка 221. Это означает, что ядро достигло блока, «пролетев» расстояние, отделяющее блок от левой границы экрана.
- ❑ Разность координаты x ядра и радиуса ядра должна быть меньше, чем расстояние от блока до левого края экрана (строка 222). Эта проверка гарантирует, что ядро еще не пролетело за блок.
- ❑ Часть ядра должна располагаться ниже, чем верхняя часть блока (строка 223).
- ❑ Часть ядра должна располагаться выше, чем нижняя часть блока (строка 224).

Если все эти условия выполняются, направление движения ядра меняется на противоположное (строка 226), пользователь теряет часть оставшегося времени (значение `timeLeft` уменьшается на `MISS_PENALTY`), после чего вызывается метод `play` класса `soundPool` для воспроизведения звука, сопровождающего попадание в блок. Для выборки звука в `SoundPool` в качестве ключа `soundMap` используется константа `BLOCKER_SOUND_ID`.

Проверка выхода ядра за пределы экрана

Изображение ядра исчезает с экрана после столкновения с любой из сторон экрана. Строки 233–237 проверяют, столкнулось ли ядро с левой либо с правой стенкой, и если это так — изображение ядра удаляется с экрана. В строках 233–235 ядро удаляется при столкновении с верхним или нижним краем экрана.

Проверка столкновения с мишенью

Затем выполняется проверка столкновения ядра с мишенью (строки 245–248). При этом проверяются условия, подобные условиям столкновения ядра с блоком. Если ядро попало в мишень, программа определяет секцию мишени, в которую попало ядро. В строках 251–252 секция мишени, в которую попало ядро, определяется делением расстояния между ядром и нижней частью мишени на длину секции мишени. Результат лежит в диапазоне от 0 (верхняя секция мишени) до 6 (нижняя секция мишени). Для проверки попаданий ядра в секцию мишени используется массив `hitStates` (строка 256). При попадании ядра в секцию мишени соответствующему элементу массива `hitStates` присваивается значение `true`, а изображение ядра удаляется с экрана. Затем значение переменной `HIT_REWARD` добавляется в счетчик `timeLeft`, увеличивая тем самым значение оставшегося времени, и воспроизводится звук попадания в мишень (`TARGET_SOUND_ID`). Увеличивается значение переменной `targetPiecesHit`, определяется, равно ли оно значению переменной `TARGET_PIECES` (строка 267). Если это условие выполнено, игра закончена; поток `CannonThread` завершается вызовом метода `setRunning` с аргументом `false`, после чего вызывается метод `showGameOverDialog` с идентификатором строкового ресурса, который определяет сообщение о выигрыше и присваивает `gameOver` значение `true`.

Обновление позиций блока и мишени

Теперь, после проверки всех возможных столкновений пушечного ядра, следует обновить значения позиций мишени и блока. В строках 278–280 изменяется значение позиции блока путем умножения значения `blockerVelocity` на время, прошедшее с момента последнего обновления; произведение прибавляется к текущим координатам `x` и `y`. В строках 283–285 выполняются те же операции по отношению к мишени. Если блок сталкивается с верхней или нижней стенками, направление его движения реверсируется путем умножения значения скорости на `-1` (строки 288–289). В строках 292–293 выполняются те же проверки и настройки для мишени, имеющей полную длину, включая секции мишени, в которые было попадание ядра.

Обновление оставшегося времени и проверка таймера

Значение переменной `timeLeft` уменьшается на время, которое прошло с момента воспроизведения предыдущего кадра анимации. Если значение переменной `timeLeft` становится равным нулю, игра завершается. Переменной `timeLeft` присваивается значение 0.0 (если этого не сделать, на экране может отобразиться отрицательное значение времени). Затем переменной `gameOver` присваивается значение `true`, поток `CannonThread` завершается вызовом `setRunning` с аргументом `false`, а также вызывается метод `showGameOverDialog` с идентификатором строкового ресурса, используемого для вывода сообщения о проигрыше.

6.8.7. Метод fireCannonball класса CannonView

Если пользователь касается экрана, обработчик событий для этого события (раздел 6.8.13) вызывает метод `fireCannonball` (листинг 6.10). Если ядро уже находится на экране, метод немедленно завершается. В противном случае (изображение ядра на экране отсутствует) выполняется выстрел из пушки. В строке 313 вызывается метод `alignCannon`, с помощью которого нацеливается пушка в точку касания, а также считывается значение угла наклона пушки. В строках 316–317 пушка «заряжается» (то есть ядро помещается в ствол), после чего в строках 320 и 323 вычисляются горизонтальная и вертикальная компоненты скорости пушечного ядра. Затем переменной `cannonballOnScreen` присваивается значение `true`, чтобы ядро прорисовывалось методом `drawGameElements` (см. листинг 6.12), и увеличивается переменная `shotsFired`. Метод завершается воспроизведением звука выстрела (`CANNON_SOUND_ID`).

Листинг 6.10. Метод fireCannonball класса CannonView

```

307     // Выстрел из пушки
308     public void fireCannonball(MotionEvent event)
309     {
310         if (cannonballOnScreen) // Если ядро уже находится на экране
311             return; // Не делать ничего
312
313         double angle = alignCannon(event); // Определить угол наклона ствола
314
315         // Поместить ядро в ствол пушки
316         cannonball.x = cannonballRadius; // Координата x совмещается с пушкой
317         cannonball.y = screenHeight / 2; // Центровка ядра по вертикали
318
319         // Получить составляющую x общей скорости
320         cannonballVelocityX = (int) (cannonballSpeed * Math.sin(angle));
321
322         // Получить составляющую y общей скорости
323         cannonballVelocityY = (int) (-cannonballSpeed * Math.cos(angle));
324         cannonballOnScreen = true; // Ядро находится на экране
325         ++shotsFired; // Увеличить shotsFired
326
327         // Воспроизведение звука выстрела
328         soundPool.play(soundMap.get(CANNON_SOUND_ID), 1, 1, 1, 0, 1f);
329     } // Конец метода fireCannonball
330

```

6.8.8. Метод alignCannon

Метод `alignCannon` (листинг 6.11) наводит пушку на точку касания. В строке 335 программа получает координаты `x` и `y` точки касания из аргумента `MotionEvent`, после чего вычисляется вертикальное расстояние точки касания от центра экрана. Если это расстояние отлично от нуля, вычисляется угол наклона ствола пушки относительно горизонта (строка 345). Если касание было сделано в нижней части

экрана, то угол увеличивается на величину `Math.PI` (строка 349). Затем переменные `cannonLength` и `angle` используются для определения координат x и y конца ствола пушки. Результат используется для рисования линии от центра основания пушки у левого края экрана до конечной точки ствола пушки.

Листинг 6.11. Метод alignCannon класса CannonView

```

331 // Изменение угла наклона пушки в ответ на касание
332 public double alignCannon(MotionEvent event)
333 {
334     // Получение точки касания в этом представлении
335     Point touchPoint = new Point((int) event.getX(), (int) event.getY());
336
337     // Вычисление расстояния от точки касания до центра экрана
338     // по оси  $y$ 
339     double centerMinusY = (screenHeight / 2 - touchPoint.y);
340
341     double angle = 0; // Переменная angle инициализируется значением 0
342
343     // Вычисление угла наклона ствола относительно горизонтальной оси
344     if (centerMinusY != 0) // Предотвращение деления на 0
345         angle = Math.atan((double) touchPoint.x / centerMinusY);
346
347     // Если касание было сделано в нижней половине экрана
348     if (touchPoint.y > screenHeight / 2)
349         angle += Math.PI; // Внесение поправки
350
351     // Вычисление конечной точки ствола
352     barrelEnd.x = (int) (cannonLength * Math.sin(angle));
353     barrelEnd.y =
354         (int) (-cannonLength * Math.cos(angle) + screenHeight / 2);
355
356     return angle; // Вернуть вычисленный угол
357 } // Конец метода alignCannon
358

```

6.8.9. Метод drawGameElements

Метод `drawGameElements` (листинг 6.12) рисует пушку, пушечное ядро, блок и мишень с помощью класса `SurfaceView`. При этом используется компонент `Canvas`, полученный потоком `CannonThread` (раздел 6.8.14) из реализации `SurfaceHolder` объекта `SurfaceView`.

Листинг 6.12. Метод drawGameElements класса CannonView

```

359 // Рисование элементов игры
360 public void drawGameElements(Canvas canvas)
361 {
362     // Очистка фона
363     canvas.drawRect(0, 0, canvas.getWidth(), canvas.getHeight(),
364                     backgroundPaint);

```

```

365
366    // Вывод оставшегося времени
367    canvas.drawText(getResources().getString(
368        R.string.time_remaining_format, timeLeft), 30, 50, textPaint);
369
370    // Если ядро находится на экране, нарисовать его
371    if (cannonballOnScreen)
372        canvas.drawCircle(cannonball.x, cannonball.y, cannonballRadius,
373            cannonballPaint);
374
375    // Нарисовать ствол пушки
376    canvas.drawLine(0, screenHeight / 2, barrelEnd.x, barrelEnd.y,
377        cannonPaint);
378
379    // Нарисовать основание пушки
380    canvas.drawCircle(0, (int) screenHeight / 2,
381        (int) cannonBaseRadius, cannonPaint);
382
383    // Нарисовать блок
384    canvas.drawLine(blocker.start.x, blocker.start.y, blocker.end.x,
385        blocker.end.y, blockerPaint);
386
387    Point currentPoint = new Point(); // Начало текущей секции мишени
388
389    // currentPoint инициализируется начальной точкой мишени
390    currentPoint.x = target.start.x;
391    currentPoint.y = target.start.y;
392
393    // Рисование мишени
394    for (int i = 0; i < TARGET_PIECES; i++)
395    {
396        // Если секция цела, нарисовать ее
397        if (!hitStates[i])
398        {
399            // Чередование цветов
400            if (i % 2 != 0)
401                targetPaint.setColor(Color.BLUE);
402            else
403                targetPaint.setColor(Color.YELLOW);
404
405            canvas.drawLine(currentPoint.x, currentPoint.y, target.end.x,
406                (int) (currentPoint.y + pieceLength), targetPaint);
407        }
408
409        // Перемещение currentPoint в начало следующей секции
410        currentPoint.y += pieceLength;
411    }
412 } // Конец метода drawGameElements
413

```

Очистка объекта Canvas методом drawRect

Сначала вызывается метод `drawRect` класса `Canvas` (строки 363–364), который стирает содержимое объекта `Canvas`, чтобы все элементы игры отображались в новых положениях. Этот метод получает в качестве аргументов координаты верхнего

левого угла прямоугольника, ширину и высоту прямоугольника и объект `Paint`, определяющий характеристики рисования (`backgroundPaint` выбирает белый цвет рисования).

Вывод оставшегося времени методом `drawText` класса `Canvas`

Затем вызывается метод `drawText` класса `Canvas` (строки 367–368) для отображения оставшегося времени игры. В качестве аргументов передаются выводимая строка, координаты `x` и `y` ее вывода и компонент `textPaint` (сконфигурирован в строках 170–171), описывающий вывод текста (размер шрифта, цвет и другие атрибуты текста).

Рисование ядра методом `drawCircle` класса `Canvas`

Если пушечное ядро находится на экране, в строках 372–373 используется метод `drawCircle` класса `Canvas` для рисования ядра в его текущей позиции. Первые два аргумента представляют координаты центра окружности, а третий аргумент — радиус окружности. В последнем аргументе передается объект `Paint`, определяющий характеристики рисования окружности.

Рисование ствола, блока и мишени методом `drawLine` класса `Canvas`

С помощью метода `drawLine` класса `Canvas` отображается ствол пушки (строки 376–377), блок (строки 384–385) и секции мишени (строки 405–406). Этот метод получает пять параметров — первые четыре параметра представляют координаты начала и конца линии, а последний содержит объект `Paint`, определяющий характеристики линии (например, толщину).

Рисование основания пушки методом `drawCircle` класса `Canvas`

В строках 380–381 метод `drawCircle` класса `Canvas` используется для рисования полукруглого основания пушки. При этом рисуется окружность, центр которой находится на левом краю экрана. Таким образом половина окружности скрывается за левой границей `SurfaceView`.

Рисование секций мишени методом `drawLine` класса `Canvas`

В строках 390–411 рисуются секции мишени. Программа перебирает секции мишени, назначая каждой из них нужный цвет (синий для нечетных секций, желтый для четных). На экране отображаются только те секции мишени, в которые еще не попало ядро.

6.8.10. Метод `showGameOverDialog`

При завершении игры метод `showGameOverDialog` (листинг 6.13) отображает фрагмент `DialogFragment` (см. раздел 5.6.9) с диалоговым окном `AlertDialog`, в котором отображается сообщение о выигрыше/проигрыше, количество произведенных выстрелов и общее время игры. Вызов метода `setPositiveButton` (строки 433–444) создает кнопку сброса для начала новой игры.

Листинг 6.13. Метод showGameOverDialog класса CannonView

```

414 // Отображение окна AlertDialog при завершении игры
415 private void showGameOverDialog(final int messageId)
416 {
417     // Объект DialogFragment для вывода статистики и начала новой игры
418     final DialogFragment gameResult =
419         new DialogFragment()
420     {
421         // Метод создает объект AlertDialog и возвращает его
422         @Override
423         public Dialog onCreateDialog(Bundle bundle)
424         {
425             // Создание диалогового окна с выводом строки messageId
426             AlertDialog.Builder builder =
427                 new AlertDialog.Builder(getActivity());
428             builder.setTitle(getResources().getString(messageId));
429
430             // Вывод количества выстрелов и затраченного времени
431             builder.setMessage(getResources().getString(
432                 R.string.results_format, shotsFired, totalElapsedTime));
433             builder.setPositiveButton(R.string.reset_game,
434                 new DialogInterface.OnClickListener()
435             {
436                 // Вызывается при нажатии кнопки "Reset Game"
437                 @Override
438                 public void onClick(DialogInterface dialog, int which)
439                 {
440                     dialogIsDisplayed = false;
441                     newGame(); // Создание и начало новой партии
442                 }
443             } // Конец анонимного внутреннего класса
444         ); // Конец вызова setPositiveButton
445
446         return builder.create(); // Вернуть AlertDialog
447     } // Конец метода onCreateDialog
448 }; // Конец анонимного внутреннего класса DialogFragment
449
450 // В UI-потоке FragmentManager используется для вывода DialogFragment
451 activity.runOnUiThread (
452     new Runnable() {
453         public void run()
454         {
455             dialogIsDisplayed = true;
456             gameResult.setCancelable(false); // Модальное окно
457             gameResult.show(activity.getFragmentManager(), "results");
458         }
459     } // Конец Runnable
460 ); // Конец вызова runOnUiThread
461 } // Конец метода showGameOverDialog
462

```

Метод `onClick` слушателя кнопок сигнализирует о том, что диалоговое окно больше не отображается, и вызывает метод `newGame` для настройки и запуска новой игры. Диалоговое окно должно отображаться в потоке GUI, поэтому в строках 451–460

вызывается метод `runOnUiThread` класса `Activity` для определения объекта `Runnable`, который должен быть выполнен в потоке GUI как можно раньше. В аргументе передается объект анонимного внутреннего класса, реализующий `Runnable`.

6.8.11. Методы `stopGame` и `releaseResources`

Методы `onPause` и `onDestroy` (листинг 6.14) класса `CannonGame` (класс `Activity`) вызывают методы `stopGame` и `releaseResources` класса `CannonView` (листинг 7.19) соответственно. Метод `stopGame` (строки 464–468) вызывается из главного класса активности для остановки игры при вызове метода `onPause` класса `Activity` (для упрощения в этом примере не сохраняются сведения о состоянии игры). Метод `releaseResources` (строки 471–475) вызывает метод `release` класса `SoundPool`, с помощью которого выполняется освобождение ресурсов, связанных с `SoundPool`.

Листинг 6.14. Методы `stopGame` и `releaseResources` класса `CannonView`

```
463 // Остановка игры; вызывается методом onPause класса CannonGameFragment
464 public void stopGame()
465 {
466     if (cannonThread != null)
467         cannonThread.setRunning(false); // Приказываем потоку завершиться
468 }
469
470 // Освобождение ресурсов; вызывается методом onDestroy класса CannonGame
471 public void releaseResources()
472 {
473     soundPool.release(); // Освободить все ресурсы, используемые SoundPool
474     soundPool = null;
475 }
476
```

6.8.12. Реализация методов `SurfaceHolder.Callback`

В листинге 6.15 приведена реализация методов `surfaceChanged`, `surfaceCreated` и `surfaceDestroyed` интерфейса `SurfaceHolder.Callback`. Тело метода `surfaceChanged` в этом приложении остается пустым, потому что приложение всегда отображается в портретной ориентации. Этот метод вызывается при изменении размеров или ориентации `SurfaceView` и обычно используется для перерисовки графики для новой конфигурации. Метод `surfaceCreated` (строки 485–494) вызывается при создании `SurfaceView` — например, при первой загрузке приложения или при возвращении его из фонового режима. В нашем приложении `surfaceCreated` используется для создания и запуска потока `CannonThread`, инициирующего цикл игры. Метод `surfaceDestroyed` (строки 497–515) вызывается при уничтожении `SurfaceView` — например, при завершении приложения. Мы используем его для обеспечения корректного завершения `CannonThread`. Сначала в строке 502 метод `setRunning` класса `CannonThread` вызывается с аргументом `false`, который показывает,

что поток должен остановиться, после чего строки 504–515 ожидают завершения потока. Тем самым предотвращаются возможные попытки рисования на SurfaceView после завершения surfaceDestroyed.

Листинг 6.15. Реализация методов SurfaceHolder.Callback

```

477 // Вызывается при изменении размеров поверхности
478 @Override
479 public void surfaceChanged(SurfaceHolder holder, int format,
480     int width, int height)
481 {
482 }
483
484 // Вызывается при создании поверхности
485 @Override
486 public void surfaceCreated(SurfaceHolder holder)
487 {
488     if (!dialogIsDisplayed)
489     {
490         cannonThread = new CannonThread(holder); // Создание потока
491         cannonThread.setRunning(true); // Запуск игры
492         cannonThread.start(); // Запуск потока цикла игры
493     }
494 }
495
496 // Вызывается при уничтожении поверхности
497 @Override
498 public void surfaceDestroyed(SurfaceHolder holder)
499 {
500     // Обеспечить корректное завершение потока
501     boolean retry = true;
502     cannonThread.setRunning(false); // Завершение cannonThread
503
504     while (retry)
505     {
506         try
507         {
508             cannonThread.join(); // Ожидание завершения cannonThread
509             retry = false;
510         }
511         catch (InterruptedException e)
512         {
513             Log.e(TAG, "Thread interrupted", e);
514         }
515     }
516 } // Конец метода surfaceDestroyed
517

```

6.8.13. Переопределение метода onTouchEvent

В этом примере переопределяется метод onTouchEvent класса View (листинг 6.16) для получения информации о касании экрана пользователем. Параметр MotionEvent содержит информацию о произошедшем событии. В строке 523 метод getAction

класса `MotionEvent` используется для определения типа события касания. Затем строки 526–527 определяют, коснулся ли пользователь экрана (`MotionEvent.ACTION_DOWN`) или провел пальцем по экрану (`MotionEvent.ACTION_MOVE`). В обоих случаях строка 529 вызывает метод `fireCannonball` представления `cannonView` для наведения пушки на точку касания и выполнения выстрела. Затем строка 532 возвращает `true` — признак того, что событие касания было обработано.

Листинг 6.16. Переопределение метода onTouchEvent класса View

```

518    // Вызывается при касании экрана в этой активности
519    @Override
520    public boolean onTouchEvent(MotionEvent e)
521    {
522        // Получение типа действия, инициировавшего событие
523        int action = e.getAction();
524
525        // Пользователь коснулся экрана или провел по нему пальцем
526        if (action == MotionEvent.ACTION_DOWN ||
527            action == MotionEvent.ACTION_MOVE)
528        {
529            fireCannonball(e); // Выстрел к точке касания
530        }
531
532        return true;
533    } // Конец метода onTouchEvent
534

```

6.8.14. Поток CannonThread: использование потока для создания цикла игры

В листинге 6.17 определяется субкласс класса `Thread`, обновляющий состояние игры. Поток хранит ссылку на метод `SurfaceHolder` класса `SurfaceView` (строка 538) и логическое значение, указывающее, выполняется ли поток. Метод класса `run` (строки 556–587) управляет покадровыми анимациями (этот процесс называется *циклом игры*). Каждое обновление элементов игры на экране выполняется с учетом количества миллисекунд, прошедших с момента последнего обновления. В строке 559 считывается текущее системное время в миллисекундах, прошедшее с момента начала выполнения потока. В строках 561–586 цикл выполняется до тех пор, пока значение `threadIsRunning` не станет равным `false`.

Листинг 6.17. Реализация Runnable, обновляющая игру через каждые TIME_INTERVAL миллисекунд

```

535    // Субкласс Thread, управляющий циклом игры
536    private class CannonThread extends Thread
537    {
538        private SurfaceHolder surfaceHolder; // Для операций с Canvas
539        private boolean threadIsRunning = true; // Выполняется по умолчанию
540

```

```

541      // Инициализация SurfaceHolder
542      public CannonThread(SurfaceHolder holder)
543      {
544          surfaceHolder = holder;
545          setName("CannonThread");
546      }
547
548      // Изменение состояния выполнения
549      public void setRunning(boolean running)
550      {
551          threadIsRunning = running;
552      }
553
554      // Управление циклом игры
555      @Override
556      public void run()
557      {
558          Canvas canvas = null; // Используется для рисования
559          long previousFrameTime = System.currentTimeMillis();
560
561          while (threadIsRunning)
562          {
563              try
564              {
565                  // Захват Canvas для монопольного рисования из этого потока
566                  canvas = surfaceHolder.lockCanvas(null);
567
568                  // Блокировка surfaceHolder для рисования
569                  synchronized(surfaceHolder)
570                  {
571                      long currentTime = System.currentTimeMillis();
572                      double elapsedTimeMS = currentTime - previousFrameTime;
573                      totalElapsedTime += elapsedTimeMS / 1000.0;
574                      updatePositions(elapsedTimeMS); // Обновление состояния игры
575                      drawGameElements(canvas); // Рисование
576                      previousFrameTime = currentTime; // Обновление предыдущего
577                                         // времени
578                  }
579              }
580              finally
581              {
582                  // Вывод содержимого Canvas на CannonView
583                  // и разрешение использования Canvas другими потоками
584                  if (canvas != null)
585                      surfaceHolder.unlockCanvasAndPost(canvas);
586              }
587          } // Конец while
588      } // Конец метода run
589  } // Конец вложенного класса CannonThread
590 } // Конец класса CannonView

```

Сначала мы получаем объект `Canvas`, используемый для рисования на `SurfaceView`, вызовом метода `lockCanvas` класса `SurfaceHolder` (строка 566). На `SurfaceView` в любой момент времени должен рисовать только один поток, поэтому сначала нужно

заблокировать `SurfaceHolder`. Затем программа получает текущее время (в миллисекундах), вычисляется прошедшее время и прибавляет его счетчику времени — это значение применяется для вывода оставшегося времени игры. В строке 574 вызывается метод `updatePositions`, перемещающий все элементы игры; в аргументе этого метода передается время выполнения приложения в миллисекундах. Таким образом обеспечивается постоянная скорость игры *независимо от используемого устройства*. Если интервал времени между соседними кадрами увеличивается (на медленном устройстве), элементы игры будут перемещаться на большее расстояние при отображении очередного кадра анимации. Если же интервал времени между кадрами меньше (на быстром устройстве), то элементы игры будут перемещаться на меньшее расстояние при отображении очередного кадра. И наконец, в строке 575 рисуются элементы игры с использованием объекта `Canvas` класса `SurfaceView`, а в строке 576 значение `currentTime` заменяет предыдущее значение `previousFrameTime` для подготовки вычисления времени между текущим и *следующим* кадром анимации.

6.9. Резюме

В этой главе вы создали приложение Cannon Game, в котором игрок должен разрушить состоящую из семи секций мишень до истечения 10-секундного интервала. Пользователь наводит на цель и стреляет из пушки, касаясь экрана. Чтобы рисование выполнялось в отдельном потоке, мы создали отдельное представление расширением класса `SurfaceView`. Вы узнали о том, что имена классов пользовательских компонентов должны быть полностью заданы в элементе разметки XML, представляющем компонент. Мы рассмотрели дополнительные методы «жизненного цикла» класса `Activity`: метод `onPause` вызывается из текущей активности в случае, если другая деятельность становится активной, а метод `onDestroy` — при завершении активности системой. Переопределение метода `onTouchEvent` класса `Activity` было использовано для обработки касаний. Мы добавили звуковые эффекты в папку приложения `res/raw` и управляли ими с помощью класса `SoundPool`, а также использовали системную службу `AudioManager` для получения текущего значения громкости звучания музыки и использования этого значения для установки громкости воспроизведения звука.

Это приложение реализует анимации вручную, путем обновления элементов игры на `SurfaceView` из отдельного потока выполнения. Мы расширили класс `Thread` и создали метод `run`, который выводит графику с использованием методов класса `Canvas`. Мы использовали объект `SurfaceHolder` класса `SurfaceView` для получения объекта `Canvas`, а также научились создавать циклы, управляющие игрой на основе информации о времени, которое прошло между кадрами анимации. В результате игра выполняется с одной и той же скоростью на различных устройствах.

В главе 7 будет представлено приложение Doodlz, использующее графические возможности Android для превращения экрана устройства в *виртуальный холст*. Также вы узнаете о *режиме погружения*, появившемся в Android 4.4, и возможностях печати.

7

Приложение Doodlz

Двумерная графика, объекты Canvas и Bitmap, акселерометр, объекты SensorManager, события многоточечных касаний, MediaStore, печать, режим погружения

В этой главе...

- Обнаружение событий, связанных с касанием экрана, перемещением пальца вдоль экрана и отведения пальца от экрана
- Обработка многоточечных касаний экрана и возможность рисования несколькими пальцами одновременно
- Использование объекта SensorManager и акселерометра для обнаружения событий перемещения
- Применение объектов Paint для определения цвета и ширины линии
- Использование объектов Path для хранения данных линий, и объектов Canvas для рисования линий на Bitmap
- Создание меню и отображение команд на панели действий
- Режим погружения Android 4.4 с возможностью рисования на всем экране
- Использование инфраструктуры печати Android 4.4 и класса PrintHelper из Android Support Library для вывода изображений на печать

7.1. Введение

Приложение Doodlz превращает экран вашего устройства в *виртуальный холст* (рис. 7.1). В этом приложении используется *режим погружения* Android 4.4, позволяющий рисовать на всем экране, — *панель действий* и *системные панели* устройства скрываются и появляются при касании экрана.

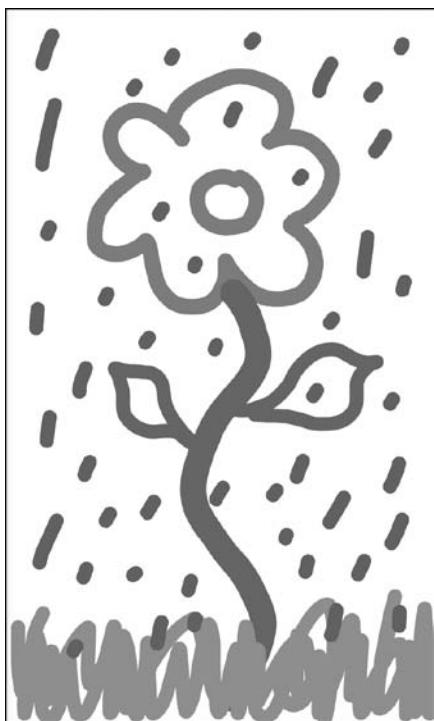


Рис. 7.1. Приложение Doodlz с завершенным рисунком

Меню приложения позволяет настроить цвет и толщину линии. В диалоговом окне Choose Color (рис. 7.2, а) отображаются компоненты SeekBar (ползунки) для настройки интенсивности альфа-канала (прозрачности), красного, зеленого и синего цветов, позволяя выбрать цвет в формате ARGB (см. раздел 1.9). По мере перемещения ползунка каждого компонента SeekBar в образце цвета под компонентами SeekBar отображается текущий цвет. В диалоговом окне Choose Line Width, показанном на рис. 7.2, б, отображается компонент SeekBar для настройки толщины рисуемой линии. Другие команды меню (рис. 7.3) позволяют включить режим стирания (Eraser), очистить экран (Clear), сохранить текущий рисунок в галерее устройства (Save) или на устройствах Android 4.4 — вывести текущее изображение на печать. В зависимости от размера экрана устройства некоторые пункты меню отображаются прямо на панели действий, а остальные скрываются в меню. Вы можете в любой момент

очистить экран, встряхнув устройство. Поскольку тестовый запуск приложения уже был описан в разделе 1.9, здесь мы его не приводим. Хотя приложение работает и на виртуальных устройствах AVD, на физическом устройстве оно работает более плавно. [Примечание: из-за ошибки в приложении Gallery, существовавшей на момент написания книги, на некоторых устройствах для нормального сохранения изображений из приложения Doodlz необходимо сначала сделать фотографию.]

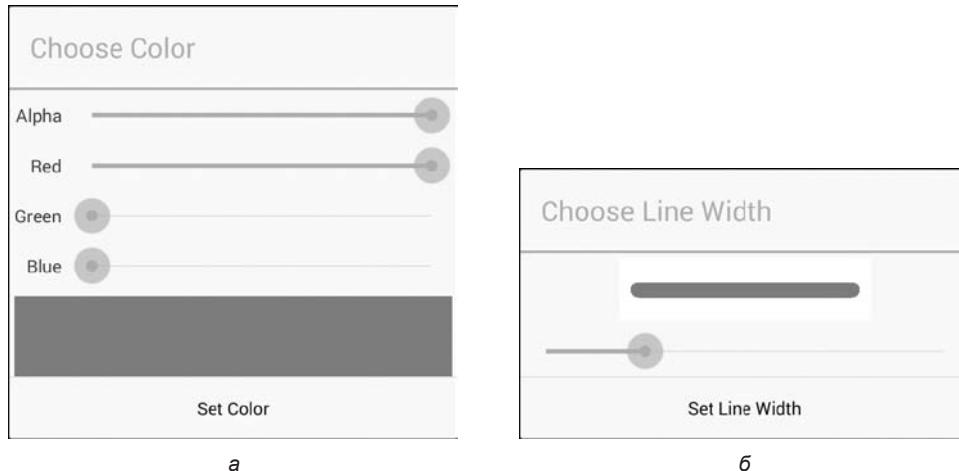


Рис. 7.2. Диалоговые окна выбора цвета и толщины линии в приложении Doodlz:
а — диалоговое окно выбора цвета; б — диалоговое окно выбора толщины линии

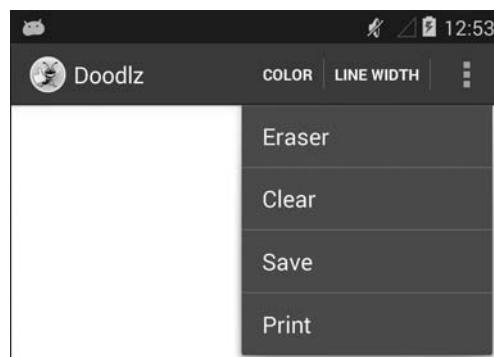


Рис. 7.3. Дополнительные команды меню Doodlz
на телефоне с системой Android 4.4

7.2. Обзор применяемых технологий

В этом разделе рассматриваются новые технологии, задействованные при создании приложения Doodlz.

7.2.1. Использование SensorManager для прослушивания событий акселерометра

Для удаления текущего рисунка, созданного с помощью приложения, достаточно встрихнуть устройство. Многие устройства Android снабжены *акселерометрами*, с помощью которых приложения отслеживают перемещения устройств. Также в настоящее время Android поддерживает датчики гравитации, гироскопы, датчики освещения, линейного ускорения, магнитного поля, ориентации, давления, приближения, вектора поворота и температуры. Перечень констант `Sensor`, представляющих эти типы датчиков, приведен по адресу

<http://developer.android.com/reference/android/hardware/Sensor.html>

Обработка событий акселерометра и датчиков рассматривается в разделе 7.5. За информацией о других датчиках Android обращайтесь по адресу

http://developer.android.com/guide/topics/sensors/sensors_overview.html

7.2.2. Пользовательские реализации DialogFragment

В предыдущих приложениях для вывода или получения данных от пользователя (в виде нажатий кнопок) применялись окна `AlertDialog` в фрагментах `DialogFragment`. Объекты `AlertDialog`, использовавшиеся ранее, создавались с помощью анонимных внутренних классов, которые расширяли `DialogFragment` и позволяли выводить только текст и кнопки. Объекты `AlertDialog` также могут содержать пользовательские представления. В этом приложении будут определены следующие три субкласса `DialogFragment`.

- ❑ `ColorDialogFragment` (раздел 7.7) отображает окно `AlertDialog` с пользовательским представлением, содержащим компоненты GUI для предварительного просмотра и выбора нового цвета линии в формате ARGB.
- ❑ `LineWidthDialogFragment` (раздел 7.8) отображает окно `AlertDialog` с пользовательским представлением, содержащим компоненты GUI для предварительного просмотра и выбора толщины линии.
- ❑ `EraseImageDialogFragment` (раздел 7.9) отображает стандартное окно `AlertDialog` для подтверждения стирания всего рисунка.

Пользовательские представления фрагментов `ColorDialogFragment` и `EraseImageDialogFragment` будут заполняться из ресурсного файла макета. В каждом из трех субклассов `DialogFragment` также будут переопределяться следующие методы жизненного цикла `Fragment`.

- ❑ `onAttach` — первый метод жизненного цикла фрагмента, вызываемый при присоединении фрагмента к родительской активности.

- `onDetach` — последний метод жизненного цикла фрагмента, вызываемый перед отсоединением фрагмента от родительской активности.

Предотвращение одновременного появления нескольких диалоговых окон

Обработчик события встрихивания может попытаться вывести диалоговое окно подтверждения стирания в тот момент, когда на экране уже находится другое диалоговое окно. Чтобы этого не произошло, мы воспользуемся методами `onAttach` и `onDetach` для задания логической переменной, которая сообщает, когда диалоговое окно находится на экране. Если переменная истинна, приложение запрещает обработчику события встрихивания отображать диалоговое окно.

7.2.3. Рисование с использованием Canvas и Bitmap

Приложение рисует линии на объектах `Bitmap` (пакет `android.graphics`). Разработчик может связать с `Bitmap` объект `Canvas`, а затем использовать `Canvas` для рисования на `Bitmap`; изображение появится на экране (см. раздел 7.6). Объект `Bitmap` также можно сохранить в файле — мы воспользуемся этой возможностью для сохранения рисунков в галерее изображений устройства командой `Save`.

7.2.4. Обработка событий многоточечных касаний и хранение данных линий в объектах Path

Чтобы создать рисунок, пользователь проводит одним или несколькими пальцами по экрану. Приложение сохраняет информацию от каждого пальца в объекте `Path` (пакет `android.graphics`), представляющем геометрический контур из набора отрезков и кривых. Обработка событий касания осуществляется переопределением метода `onTouchEvent` класса `View` (раздел 7.6). Этот метод получает объект `MotionEvent` (пакет `android.view`) с информацией о типе произошедшего события касания и идентификатором пальца, сгенерировавшего событие. По идентификатору можно различать пальцы и добавлять информацию в соответствующие объекты `Path`. По типу события касания мы определяем, какую операцию выполнил пользователь — коснулся экрана, провел по экрану или отнял палец от экрана.

7.2.5. Режим погружения Android 4.4

В Android 4.4 появился новый полноэкранный *режим погружения* (раздел 7.6), в котором приложение использует весь экран, но пользователь может при необходимости получить доступ к системным панелям. В нашем приложении этот режим используется при запуске на устройствах с Android 4.4 и выше.

7.2.6. GestureDetector и SimpleOnGestureListener

Для сокрытия/отображения системных панелей устройства и панели действий приложения в этом приложении используется класс `GestureDetector` (пакет `android.view`). `GestureDetector` позволяет приложению реагировать на такие пользовательские действия, как одиночные и двойные касания, долгие нажатия и смахивание, реализуя методы интерфейсов `GestureDetector.OnGestureListener` и `GestureDetector.OnDoubleTapListener`. Класс `GestureDetector.SimpleOnGestureListener` представляет собой простой класс-адаптер, реализующий все методы двух интерфейсов, чтобы разработчик мог расширить этот класс и переопределить только нужные методы этих интерфейсов. В разделе 7.6 объект `GestureDetector` будет инициализирован экземпляром `SimpleOnGestureListener`, который будет обрабатывать события одиночного касания для сокрытия или отображения системных панелей и панели действий.

7.2.7. Сохранение рисунка в галерее устройства

Команда меню `Save` сохраняет изображение в *галерее* устройства — стандартном месте для хранения изображений и фотографий. С помощью объекта `ContentResolver` (пакет `android.content`) приложение считывает данные с устройства, а также сохраняет их на этом же устройстве. В приложении объект `ContentResolver` (раздел 7.6) и метод `insertImage` класса `MediaStore.Images.Media` используются для сохранения изображения в галерее. Объект `MediaStore` управляет графическими, звуковыми и видеофайлами, хранящимися на устройстве.

7.2.8. Поддержка печати в Android 4.4 и класс PrintHelper из Android Support Library

В Android 4.4 появилась поддержка печати. В нашем приложении класс `PrintHelper` (раздел 7.6) используется для вывода текущего изображения на печать. Класс `PrintHelper` предоставляет пользовательский интерфейс для выбора принтера, содержит метод для проверки поддержки печати заданным устройством, а также метод для вывода `Bitmap` на печать. Класс `PrintHelper` входит в Android Support Library — группу библиотек, часто используемых для реализации новых возможностей в более старых версиях Android. Библиотеки также включают дополнительные вспомогательные средства (такие, как класс `PrintHelper`) для конкретных версий Android.

7.3. Создание графического интерфейса и файлов ресурсов приложения

В этом разделе мы создадим файлы ресурсов, файлы макетов и классы приложения `Doodlz`.

7.3.1. Создание проекта

Начните с создания нового проекта Android под названием Doodlz. В диалоговом окне New Android Project укажите следующие значения, затем нажмите кнопку Finish.

- Application name: Doodlz
- Project Name: Doodlz
- Package Name: com.deitel.doodlz
- Minimum Required SDK: API18: Android 4.3
- Target SDK: API19: Android 4.4
- Compile With: API19: Android 4.4
- Theme: Holo Light with Dark Action Bar

На втором шаге мастера New Android Project (New Android Application) оставьте настройки по умолчанию и нажмите кнопку Next >. На шаге Configure Launcher Icon выберите значок приложения и нажмите кнопку Next >. На шаге Create Activity выберите шаблон Blank Activity, нажмите кнопку Next >. На шаге Blank Activity оставьте значения по умолчанию и нажмите Finish, чтобы завершить создание проекта. В макетном редакторе выберите в раскрывающемся списке тип экрана Nexus 4 и удалите компонент TextView с текстом “Hello world!”.

Новый проект автоматически настраивается для использования текущей версии Android Support Library. Если вы обновляете существующий проект, добавьте последнюю версию Android Support Library в проект. За дополнительной информацией обращайтесь по адресам

<http://developer.android.com/tools/support-library/index.html>

<http://developer.android.com/tools/support-library/setup.html>

7.3.2. Файл strings.xml

Строчные ресурсы уже создавались в предыдущих главах, поэтому сейчас мы приведем только таблицу с именами ресурсов и соответствующими значениями (табл. 7.1). Сделайте двойной щелчок на файле strings.xml из папки res/values, чтобы запустить редактор для создания этих строчных ресурсов.

Таблица 7.1. Строчные ресурсы, используемые в приложении Doodlz

Имя ресурса	Значение
app_name	Doodlz
button_erase	Erase Image
button_cancel	Cancel

Таблица 7.1 (окончание)

Имя ресурса	Значение
button_set_color	Set Color
button_set_line_width	Set Line Width
line_imageview_description	This displays the line thickness
label_alpha	Alpha
label_red	Red
label_green	Green
label_blue	Blue
menuitem_clear	Clear
menuitem_color	Color
menuitem_eraser	Eraser
menuitem_line_width	Line Width
menuitem_save	Save
menuitem_print	Print
message_erase	Erase the drawing?
message_error_saving	There was an error saving the image
message_saved	Your painting has been saved to the Gallery
message_error_printing	Your device does not support printing
title_color_dialog	Choose Color
title_line_width_dialog	Choose Line Width

7.3.3. Файл dimens.xml

В табл. 7.2 приведены имена ресурсов метрик и значениями, добавленными в файл dimens.xml. Откройте файл dimens.xml из папки res/values, чтобы запустить редактор для создания этих ресурсов. Ресурс line_imageview_height определяет высоту компонента ImageView с образцом толщины линии во фрагменте LineWidthDialogFragment, а ресурс color_view_height определяет высоту компонента View с образцом цвета в ColorDialogFragment.

Таблица 7.2. Метрические ресурсы, используемые в приложении Doodlz

Имя ресурса	Значение
line_imageview_height	50dp
color_view_height	80dp

7.3.4. Меню DoodleFragment

В главе 5 для отображения команды меню `Settings` в приложении `Flag Quiz` использовалось стандартное меню, сгенерированное средой разработки. В этом приложении меню по умолчанию не используется, поэтому файл `main.xml` можно удалить из папки `res/menu` проекта. Мы определим собственное меню `DoodleFragment`.

Меню для разных версий Android

Мы создадим две версии меню `DoodleFragment` — для устройств на базе Android 4.3 и более ранних версий и для устройств с Android 4.4 и выше. Печать поддерживается только в Android 4.4 и выше, поэтому команда `Print` будет включена в меню только этих устройств. Один ресурс меню будет определен в папке `res/menu`, а другой — в папке `res/menu-v19` (19 — версия Android API, соответствующая Android 4.4). Android выберет ресурс меню в папке `res/menu-v19` при запуске приложения на устройствах Android 4.4 и выше. Чтобы создать папку `res/menu-v19`, щелкните правой кнопкой мыши на папке `res`, выберите команду `New > Folder`, введите имя папки `menu-v19` и нажмите `Finish`.

Меню для Android 4.3 и более ранних версий

Чтобы создать ресурс меню для Android 4.3 и более ранних версий, выполните следующие действия.

1. Щелкните правой кнопкой мыши на папке `res/menu` и выберите команду `New > Android XML File`.
2. В открывшемся диалоговом окне введите имя файла `doodle_fragment_menu.xml` и нажмите `Finish`. Среда разработки открывает файл в редакторе ресурсов меню.
3. Нажмите `Add...`, перейдите на вкладку `Layout` в открывшемся диалоговом окне, выберите вариант `Item` и нажмите `OK`. Среда разработки выделяет новый пункт меню и отображает ее атрибуты справа.
4. Задайте свойству `Id` значение `@+id/color`, свойству `Title` — значение `@string/menuitem_color`, а свойству `Show as action` — значение `ifRoom`. Значение `ifRoom` указывает, что команды меню должны отображаться на панели действий при наличии свободного пространства; в противном случае команда меню должна открываться из меню в правой части панели действий. Другие значения `Show as action` можно найти по адресу

<http://developer.android.com/guide/topics/resources/menu-resource.html>

5. Повторите шаги 3 и 4 для пунктов `lineWidth`, `eraser`, `clear` и `save` (табл. 7.3). Обратите внимание: когда вы щелкаете на кнопке `Add...` для каждой дополнительной команды меню, в открывшемся диалоговом окне необходимо установить переключатель `Create a new element at the top level in Menu`.
6. Сохраните и закройте файл `doodle_fragment_menu.xml`.

Таблица 7.3. Дополнительные команды меню DoodleFragment

Идентификатор	Значение
@+id/lineWidth	@string/menuitem_line_width
@+id/eraser	@string/menuitem_eraser
@+id/clear	@string/menuitem_clear
@+id/save	@string/menuitem_save

Меню для Android 4.4 и выше

Чтобы создать ресурс меню для устройств с Android 4.4 и выше, выполните следующие действия.

1. Скопируйте файл `doodle_fragment_menu.xml` из папки `res/menu`, вставьте его в папку `res/menu-v19` и откройте файл.
2. Щелкните на кнопке `Add...`, установите переключатель `Create a new element at the top level in Menu`, выберите `Item` и нажмите `OK`.
3. Задайте свойству `Id` нового элемента значение `@+id/print`, свойству `Title` — значение `@string/menuitem_print`, а свойству `Show as action` — значение `ifRoom`.

7.3.5. Макет `activity_main.xml` для `MainActivity`

Макет `activity_main.xml` для активности `MainActivity` этого приложения содержит только фрагмент `DoodleFragment`. Чтобы добавить этот фрагмент в макет, выполните следующие действия.

1. Откройте файл `activity_main.xml` в макетном редакторе.
2. Перетащите компонент `Fragment` из раздела `Layouts` палитры в область построения интерфейса или на узел `RelativeLayout` в окне `Outline`.
3. На экране появляется диалоговое окно `Choose Fragment Class`. Нажмите `Create New...`, чтобы открыть диалоговое окно `New Java Class`.
4. Введите в поле `Name` значение `DoodleFragment` и замените содержимое поля `Superclass` значением `android.app.Fragment`. Нажмите `Finish`, чтобы создать класс. Среда разработки открывает файл с кодом Java, который пока можно закрыть.
5. Замените идентификатор (`Id`) нового фрагмента значением `@+id/doodleFragment` и сохраните макет.

7.3.6. Файл fragment_doodle.xml для фрагмента DoodleFragment

Файл `fragment_doodle.xml` для фрагмента `DoodleFragment` содержит компонент `FrameLayout`, который отображает представление `DoodleView`. В этом разделе мы создадим макет `DoodleFragment` и класс `DoodleView`. Чтобы добавить макет `fragment_doodle.xml`, выполните следующие действия.

1. Откройте узел `res/layout` в окне `Package Explorer`.
2. Щелкните правой кнопкой мыши на папке `res/layout` и выберите команду `New > Android XML File`. На экране появляется диалоговое окно `New Android XML File`.
3. В поле `File` введите имя файла `fragment_doodle.xml`.
4. В разделе `Root Element` выберите компонент `FrameLayout` и нажмите `Finish`.
5. Перетащите из раздела `Advanced` палитры компонент `view` (со строчной буквой `v`) в область построения интерфейса.
6. На экране появляется диалоговое окно `Choose Custom View Class`. Нажмите `Create New...`, чтобы открыть диалоговое окно `New Java Class`.
7. В поле `Name` введите значение `DoodleView`. Убедитесь в том, что флажок `Constructors from superclass` установлен, и нажмите `Finish`. Среда разработки создает и открывает файл `DoodleView.java`. Мы будем использовать только конструктор с двумя аргументами, поэтому два других конструктора можно удалить. Сохраните и закройте файл `DoodleView.java`.
8. В файле `fragment_doodle.xml` выделите компонент `view1` в окне `Outline`. В разделе `Layout Parameters` окна `Properties` задайте свойствам `Width` и `Height` значение `match_parent`.
9. В окне `Outline` щелкните правой кнопкой мыши на компоненте `view1`, выберите команду `Edit ID...`, переименуйте `view1` в `doodleView` и нажмите `OK`.
10. Сохраните и закройте файл `fragment_doodle.xml`.

7.3.7. Макет fragment_color.xml для фрагмента ColorDialogFragment

Файл `fragment_color.xml` для фрагмента `ColorDialogFragment` содержит компонент `GridLayout` с графическим интерфейсом выбора и предварительного просмотра образца цвета. В этом разделе мы создадим макет `ColorDialogFragment` и класс `ColorDialogFragment`. Чтобы создать макет `fragment_color.xml`, выполните следующие действия.

1. Откройте узел `res/layout` в окне Package Explorer.
2. Щелкните правой кнопкой мыши на папке `res/layout` и выберите команду `New > Android XML File`. На экране появится диалоговое окно `New Android XML File`.
3. В поле `File` введите имя файла `fragment_color.xml`.
4. В разделе `Root Element` выберите компонент `GridLayout` и нажмите `Finish`.
5. В окне `Outline` выделите компонент `GridLayout` и замените значение его свойства `Id` на `@+id/colorDialogGridLayout`.
6. Используя палитру макетного редактора, перетащите компоненты `TextView`, `SeekBar` и `View` на узел `colorDialogGridLayout` в окне `Outline`. Перетаскивайте компоненты в порядке их следования на рис. 7.4; задайте идентификаторы элементов в соответствии с иллюстрацией.

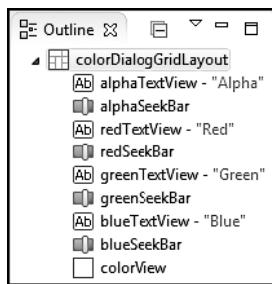


Рис. 7.4. Окно Outline для `fragment_color.xml`

7. После завершения шага 6 задайте свойства компонентов GUI в соответствии с табл. 7.4. Сохраните и закройте файл `fragment_color.xml`.

Таблица 7.4. Значения свойств компонентов GUI в файле `fragment_color.xml`

Компонент	Свойство	Значение
colorDialogGridLayout	Column Count	2
	Orientation	vertical
	Use Default Margins	true
alphaTextView	Layout Parameters – Column	0
	– Gravity	right center_vertical
	– Row	0
	Other Properties – Text	@string/label_alpha
alphaSeekBar	Layout Parameters – Column	1
	– Gravity	fill_horizontal
	– Row	0
	Other Properties – Max	255

Компонент	Свойство	Значение
redTextView	Layout Parameters – Column – Gravity – Row Other Properties – Text	0 right center_vertical 1 @string/label_red
redSeekBar	Layout Parameters – Column – Gravity – Row Other Properties – Max	1 fill_horizontal 1 255
greenTextView	Layout Parameters – Column – Gravity – Row Other Properties – Text	0 right center_vertical 2 @string/label_red
greenSeekBar	Layout Parameters – Column – Gravity – Row Other Properties – Max	1 fill_horizontal 2 255
blueTextView	Layout Parameters – Column – Gravity – Row Other Properties – Text	0 right center_vertical 3 @string/label_red
blueSeekBar	Layout Parameters – Column – Gravity – Row Other Properties – Max	1 fill_horizontal 3 255
colorView	Layout Parameters – Height – Column – Column Span – Gravity	@dimen/color_view_height 0 2 fill_horizontal

Добавление класса ColorDialogFragment в проект

Чтобы добавить класс `ColorDialogFragment` в проект, выполните следующие действия.

1. Щелкните правой кнопкой мыши на пакете `com.deitel.doodlz` в папке `src` и выберите команду `New > Class`. На экране появляется диалоговое окно `New Java Class`.
2. Введите в поле `Name` значение `ColorDialogFragment`.
3. В поле `Superclass` замените имя суперкласса на `android.app.DialogFragment`.
4. Нажмите `Finish`, чтобы создать класс.

7.3.8. Макет `fragment_line_width.xml` для фрагмента `LineWidthDialogFragment`

Файл `fragment_line_width.xml` для фрагмента `LineWidthDialogFragment` содержит компонент `GridLayout` с графическим интерфейсом выбора и предварительного просмотра толщины линии. В этом разделе мы создадим макет `LineWidthDialogFragment` и класс `LineWidthDialogFragment`. Чтобы создать макет `fragment_line_width.xml`, выполните следующие действия.

1. Откройте узел `res/layout` в окне `Package Explorer`.
2. Щелкните правой кнопкой мыши на папке `layout` и выберите команду `New > Android XML File`. На экране появится диалоговое окно `New Android XML File`.
3. В поле `File` введите имя файла `fragment_line_width.xml`.
4. В разделе `Root Element` выберите компонент `GridLayout` и нажмите `Finish`.
5. В окне `Outline` выделите компонент `GridLayout` и замените значение его свойства `Id` на `@+id/lineWidthDialogGridLayout`.
6. Используя палитру макетного редактора, перетащите компоненты `ImageView` и `SeekBar` на узел `lineWidthDialogGridLayout` в окне `Outline`. Окно должно выглядеть так, как показано на рис. 7.5. Задайте идентификаторы элементов в соответствии с иллюстрацией.

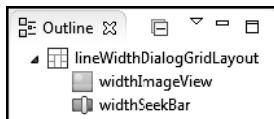


Рис. 7.5. Окно Outline
для `fragment_line_width.xml`

7. После завершения шага 6 задайте свойства GUI в соответствии с табл. 7.5. Сохраните и закройте файл `fragment_line_width.xml`.

Таблица 7.5. Значения свойств компонентов GUI в файле fragment_line_width.xml

Компонент	Свойство	Значение
lineWidthDialogGridLayout	Column Count Orientation Use Default Margins	1 vertical true
widthImageView	Layout Parameters – Height – Gravity Other Properties – Content Description	@dimen/line_imageview_height fill_horizontal @string/line_imageview_description
widthSeekBar	Layout Parameters – Gravity Other Properties – Max	fill_horizontal 50

Добавление класса LineWidthDialogFragment в проект

Чтобы добавить класс `LineWidthDialogFragment` в проект, выполните следующие действия.

- Щелкните правой кнопкой мыши на пакете `com.deitel.doodlz` в папке `src` и выберите команду `New ▶ Class`. На экране появляется диалоговое окно `New Java Class`.
- Ведите в поле `Name` значение `LineWidthDialogFragment`.
- В поле `Superclass` замените имя суперкласса на `android.app.DialogFragment`.
- Нажмите `Finish`, чтобы создать класс.

7.3.9. Добавление класса EraseImageDialogFragment

Для класса `EraseImageDialogFragment` ресурс макета не нужен, поскольку он просто выводит окно `AlertDialog` с текстом. Чтобы добавить в проект класс `EraseImageDialogFragment`, выполните следующие действия.

- Щелкните правой кнопкой мыши на пакете `com.deitel.doodlz` в папке `src` и выберите команду `New ▶ Class`. На экране появляется диалоговое окно `New Java Class`.
- Ведите в поле `Name` значение `EraseImageDialogFragment`.
- В поле `Superclass` замените имя суперкласса на `android.app.DialogFragment`.
- Нажмите `Finish`, чтобы создать класс.

7.4. Класс MainActivity

Приложение состоит из шести классов.

- ❑ **MainActivity** (листинг 7.1) – родительская активность для фрагментов этого приложения.
- ❑ **DoodleFragment** (раздел 7.5) – управляет **DoodleView** и обработкой событий акселерометра.
- ❑ **DoodleView** (раздел 7.6) – предоставляет функции рисования, сохранения и печати.
- ❑ **ColorDialogFragment** (раздел 7.7) – субкласс **DialogFragment**, отображаемый командой меню COLOR для выбора цвета.
- ❑ **LineWidthDialogFragment** (раздел 7.8) – субкласс **DialogFragment**, отображаемый командой меню LINE WIDTH для выбора толщины линии.
- ❑ **EraseImageDialogFragment** (раздел 7.9) – субкласс **DialogFragment**, отображаемый командой меню CLEAR или встряхиванием устройства для стирания текущего рисунка.

Метод `onCreate` класса **MainActivity** (листинг 7.1) заполняет графический интерфейс (строка 16), после чего использует средства, описанные в разделе 5.2.2, для определения размера устройства и назначения ориентации **MainActivity**. Если приложение выполняется на очень большом экране (строка 24), выбирается альбомная ориентация (строки 25–26); в противном случае назначается портретная ориентация (строки 28–29).

Листинг 7.1. Класс MainActivity

```
1 // MainActivity.java
2 // Подготовка макета MainActivity
3 package com.deitel.doodlz;
4
5 import android.app.Activity;
6 import android.content.pm.ActivityInfo;
7 import android.content.res.Configuration;
8 import android.os.Bundle;
9
10 public class MainActivity extends Activity
11 {
12     @Override
13     protected void onCreate(Bundle savedInstanceState)
14     {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_main);
17
18         // Определение размера экрана
19         int screenSize =
20             getResources().getConfiguration().screenLayout &
```

```

21     Configuration.SCREENLAYOUT_SIZE_MASK;
22     // Альбомная ориентация для очень больших планшетов,
23     // портретная во всех остальных случаях.
24     if (screenSize == Configuration.SCREENLAYOUT_SIZE_XLARGE)
25         setRequestedOrientation(
26             ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
27     else
28         setRequestedOrientation(
29             ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
30     }
31 } // Конец класса MainActivity

```

7.5. Класс DoodleFragment

Класс `DoodleFragment` (листинги 7.2–7.9) отображает представление `DoodleView` (раздел 7.6), управляет командами на панели действий и в меню, а также обработкой событий для функции стирания рисунка встряхиванием устройства.

Команда package, команды import и поля

В разделе 7.2 рассматриваются новые классы и интерфейсы, используемые классом `DoodleFragment`. Эти классы и интерфейсы выделены в листинге 7.2. Переменная `doodleView` класса `DoodleView` (строка 22) представляет область рисования. Вещественные переменные, объявляемые в строках 23–25, используются для пересчета изменений в ускорении устройства для выявления события встряхивания (чтобы спросить, действительно ли пользователь хочет стереть рисунок). Константа в строке 29 используется для того, чтобы малые перемещения не интерпретировались как встряхивание, — мы подобрали эту константу методом проб и ошибок на нескольких разных устройствах. В строке 26 определяется логическая переменная, по умолчанию содержащая `false`; она будет использоваться для обозначения присутствия диалогового окна на экране. Таким образом предотвращается одновременное появление нескольких диалоговых окон — например, если пользователь случайно встряхнет устройство в то время, когда на экране находится диалоговое окно `Choose Color`, диалоговое окно стирания рисунка отображаться *не должно*.

Листинг 7.2. Класс DoodleFragment: команда package, команды import и поля

```

1 // DoodleFragment.java
2 // Фрагмент, в котором отображается DoodleView
3 package com.deitel.doodlz;
4
5 import android.app.Fragment;
6 import android.content.Context;
7 import android.graphics.Color;
8 import android.hardware.Sensor;
9 import android.hardware.SensorEvent;
10 import android.hardware.SensorEventListener;
11 import android.hardware.SensorManager;
12 import android.os.Bundle;

```

```
13 import android.view.LayoutInflater;
14 import android.view.Menu;
15 import android.view.MenuInflater;
16 import android.view.MenuItem;
17 import android.view.View;
18 import android.view.ViewGroup;
19
20 public class DoodleFragment extends Fragment
21 {
22     private DoodleView doodleView; // Обработка событий касания и рисование
23     private float acceleration;
24     private float currentAcceleration;
25     private float lastAcceleration;
26     private boolean dialogOnScreen = false;
27
28     // Используется для обнаружения встяхивания устройства
29     private static final int ACCELERATION_THRESHOLD = 100000;
30 }
```

Переопределение метода onCreateView

Метод `onCreateView` (листинг 7.3) заполняет графический интерфейс `DoodleFragment` и инициализирует переменные экземпляров. Фрагмент, как и активность, может размещать элементы на панели действий приложения и в меню. Для этого он должен вызвать свой метод `setHasOptionsMenu` с аргументом `true`. Если родительская активность также размещает свои команды в меню, то элементы активности и фрагмента будут размещаться на панели действий и в меню (в зависимости от их настроек).

Листинг 7.3. Переопределение метода onCreateView фрагмента

```
31     // Вызывается при создании представления фрагмента
32     @Override
33     public View onCreateView(LayoutInflater inflater, ViewGroup container,
34         Bundle savedInstanceState)
35     {
36         super.onCreateView(inflater, container, savedInstanceState);
37         View view =
38             inflater.inflate(R.layout.fragment_doodle, container, false);
39
40         setHasOptionsMenu(true); // у фрагмента имеются команды меню
41
42         // Получение ссылки на DoodleView
43         doodleView = (DoodleView) view.findViewById(R.id.doodleView);
44
45         // Инициализация параметров ускорения
46         acceleration = 0.00f;
47         currentAcceleration = SensorManager.GRAVITY_EARTH;
48         lastAcceleration = SensorManager.GRAVITY_EARTH;
49         return view;
50     }
51 }
```

Строка 43 получает ссылку на `DoodleView`, после чего в строках 46–48 инициализируются переменные экземпляров, которые используются при вычислении изменений ускорения и помогают определить, встряхнул ли пользователь устройство. Изначально переменным `currentAcceleration` и `lastAcceleration` присваивается константа `GRAVITY_EARTH` класса `SensorManager`, представляющая ускорение, обусловленное земным притяжением. `SensorManager` также предоставляет константы для других планет Солнечной системы и ряд других занимательных значений; с ними можно ознакомиться по адресу

<http://developer.android.com/reference/android/hardware/SensorManager.html>

Методы `onStart` и `enableAccelerometerListening`

Прослушивание показаний акселерометра должно быть включено только в то время, когда `DoodleFragment` находится на экране. По этой причине мы переопределяем метод жизненного цикла фрагмента `onStart` (листинг 7.4, строки 53–58), который вызывает метод `enableAccelerometerListening` (строки 61–72) для включения прослушивания событий акселерометра. Объект `SensorManager` используется для регистрации слушателей событий акселерометра.

Листинг 7.4. Методы `onStart` и `enableAccelerometerListening`

```
52 // Включение прослушивания событий датчика
53 @Override
54 public void onStart()
55 {
56     super.onStart();
57     enableAccelerometerListening(); // Прослушивание событий встряхивания
58 }
59
60 // Включение прослушивания событий акселерометра
61 public void enableAccelerometerListening()
62 {
63     // Получение объекта SensorManager
64     SensorManager sensorManager =
65         (SensorManager) getActivity().getSystemService(
66             Context.SENSOR_SERVICE);
67
68     // Регистрация для прослушивания событий акселерометра
69     sensorManager.registerListener(sensorEventListener,
70         sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
71         SensorManagerSENSOR_DELAY_NORMAL);
72 }
73 }
```

Метод `enableAccelerometerListening` сначала использует метод `getSystemService` активности для получения системного объекта `SensorManager`, через который приложение взаимодействует с датчиками устройства. Затем в строках 69–71 приложение регистрируется для получения событий акселерометра, для чего используется метод `registerListener` объекта `SensorManager`, получающий три аргумента:

- ❑ Объект `SensorEventListener`, реагирующий на события (определяется в листинге 7.6).
- ❑ Объект `Sensor`, представляющий тип данных, которые приложение желает получать от датчиков, — для его получения вызывается метод `getDefautSensor` объекта `SensorManager` с передачей константы типа датчика (`Sensor.TYPE_ACCELEROMETER` в нашем приложении).
- ❑ Частота получения событий датчика приложением. Константа `SENSOR_DELAY_NORMAL` определяет частоту событий по умолчанию — большая частота позволит получать более точные данные, но она также требует больших затрат процессорного времени и заряда батареи.

Методы `onPause` и `disableAccelerometerListening`

Чтобы отключить прослушивание событий акселерометра на то время, когда `DoodleFragment` не находится на экране, мы переопределяем метод `onPause` жизненного цикла фрагмента (листинг 7.5; строки 75–80), вызывая в нем метод `disableAccelerometerListening` (строки 83–93). Метод `disableAccelerometerListening` вызывает метод `unregisterListener` класса `SensorManager` для прекращения прослушивания событий акселерометра.

Листинг 7.5. Методы `enableAccelerometerListening` и `disableAccelerometerListening`

```
74 // Прекращение прослушивания событий датчиков
75 @Override
76 public void onPause()
77 {
78     super.onPause();
79     disableAccelerometerListening(); // Прекращение прослушивания
80 } // событий встрихивания
81
82 // Отключение прослушивания событий акселерометра
83 public void disableAccelerometerListening()
84 {
85     // Получение объекта SensorManager
86     SensorManager sensorManager =
87         (SensorManager) getActivity().getSystemService(
88             Context.SENSOR_SERVICE);
89
90     // Отказ от прослушивания событий акселерометра
91     sensorManager.unregisterListener(sensorEventListener,
92         sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER));
93 }
94 }
```

Анонимный внутренний класс, реализующий интерфейс `SensorEventListener` для обработки событий акселерометра

В листинге 7.6 переопределяется метод `onSensorChanged` класса `SensorEventListener` (строки 100–125), обрабатывающий события акселерометра. При перемещении устройства пользователем этот метод определяет, следует ли рассматривать данное

перемещение как встряхивание. Если проверка дает положительный результат, в строке 123 вызывается метод `confirmErase` (листинг 7.7), который отображает фрагмент `EraseImageDialogFragment` (раздел 7.9) для подтверждения удаления рисунка. Интерфейс `SensorEventListener` также включает метод `onAccuracyChanged` (строки 128–131). Так как в нашем приложении этот метод не используется, его тело остается пустым.

Листинг 7.6. Анонимный внутренний класс, реализующий интерфейс `SensorEventListener` для обработки событий акселерометра

```
95 // Обработчик событий акселерометра
96 private SensorEventListener sensorEventListener =
97     new SensorEventListener()
98 {
99     // Проверка встряхивания по показаниям акселерометра
100    @Override
101    public void onSensorChanged(SensorEvent event)
102    {
103        // На экране не должно быть других диалоговых окон
104        if (!dialogOnScreen)
105        {
106            // Получить значения x, у и z для SensorEvent
107            float x = event.values[0];
108            float y = event.values[1];
109            float z = event.values[2];
110
111            // Сохранить предыдущие данные ускорения
112            lastAcceleration = currentAcceleration;
113
114            // Вычислить текущее ускорение
115            currentAcceleration = x * x + y * y + z * z;
116
117            // Вычислить изменение ускорения
118            acceleration = currentAcceleration *
119                (currentAcceleration - lastAcceleration);
120
121            // Если изменение превышает заданный порог
122            if (acceleration > ACCELERATION_THRESHOLD)
123                confirmErase();
124        }
125    } // Конец метода onSensorChanged
126
127    // Обязательный метод интерфейса SensorEventListener
128    @Override
129    public void onAccuracyChanged(Sensor sensor, int accuracy)
130    {
131    }
132}; // Конец анонимного внутреннего класса
133
```

Пользователь может случайно встряхнуть устройство даже в тех случаях, если диалоговые окна отображаются на экране. По этой причине метод `onSensorChanged` сначала проверяет, отображается ли диалоговое окно (строка 104). Эта проверка

гарантирует отсутствие на экране других диалоговых окон, что особенно важно в случае, если события датчика происходят в другом потоке выполнения. Без такой проверки окно подтверждения стирания изображения могло бы появиться одновременно с отображением на экране другого диалогового окна.

Параметр `SensorEvent` содержит информацию о произошедшем изменении состояния датчика. Предназначенный для хранения информации о событиях акселерометра массив значений этого параметра включает три элемента, которые представляют ускорение (в м/с²) в направлениях x (влево/вправо), y (вверх/вниз) и z (вперед/назад). Описание и диаграмму системы координат, используемой API `SensorEvent`, можно найти на сайте:

developer.android.com/reference/android/hardware/SensorEvent.html

По ссылке вы также познакомитесь с интерпретацией значений x, y и z `SensorEvent` для различных датчиков.

Компоненты ускорения сохраняются в строках 107–109. Очень важно быстро обрабатывать события датчика или копировать данные событий (как сделано в нашем приложении), потому что массив значений датчика заново используется для каждого события. Последнее значение `currentAcceleration` сохраняется в строке 112. В строке 115 суммируются квадраты компонентов ускорения x, y и z, и сумма сохраняется в переменной `currentAcceleration`. По значениям `currentAcceleration` и `lastAcceleration` вычисляется значение `acceleration`, которое сравнивается с константой `ACCELERATION_THRESHOLD`. Если значение превышает порог, это означает, что пользователь перемещает устройство достаточно быстро и перемещение может интерпретироваться как встрихивание. В этом случае вызывается метод `confirmErase`.

Метод `confirmErase`

Метод `confirmErase` (листинг 7.7) просто создает объект `EraseImageDialogFragment` (раздел 7.9) и использует метод `show` класса `DialogFragment` для его отображения.

Листинг 7.7. Метод `confirmErase` отображает фрагмент `EraseImageDialogFragment`

```
134 // Подтверждение стирания рисунка
135 private void confirmErase()
136 {
137     EraseImageDialogFragment fragment = new EraseImageDialogFragment();
138     fragment.show(getFragmentManager(), "erase dialog");
139 }
140
```

Переопределение методов `onCreateOptionsMenu` и `onOptionsItemSelected`

В листинге 7.8 переопределяется метод `onCreateOptionsMenu` класса `Fragment` (строки 142–147), добавляющий команды в аргумент `Menu` с использованием аргумента `MenuInflater`. Когда пользователь выбирает команду меню, метод `onOptionsItemSelected` фрагмента (строки 150–180) обрабатывает его действие.

Листинг 7.8. Переопределение методов onCreateOptionsMenu и onOptionsItemSelected фрагмента

```

141 // Отображение команд меню фрагмента
142 @Override
143 public void onCreateOptionsMenu(Menu menu, MenuInflater inflater)
144 {
145     super.onCreateOptionsMenu(menu, inflater);
146     inflater.inflate(R.menu.doodle_fragment_menu, menu);
147 }
148
149 // Обработка выбора команд меню
150 @Override
151 public boolean onOptionsItemSelected(MenuItem item)
152 {
153     // Выбор в зависимости от идентификатора MenuItem
154     switch (item.getItemId())
155     {
156         case R.id.color:
157             ColorDialogFragment colorDialog = new ColorDialogFragment();
158             colorDialog.show(getFragmentManager(), "color dialog");
159             return true; // Событие меню обработано
160         case R.id.lineWidth:
161             LineWidthDialogFragment widthdialog =
162                 new LineWidthDialogFragment();
163             widthdialog.show(getFragmentManager(), "line width dialog");
164             return true; // Событие меню обработано
165         case R.id.eraser:
166             doodleView.setDrawingColor(Color.WHITE); // Белый цвет линии
167             return true; // Событие меню обработано
168         case R.id.clear:
169             confirmErase(); // Получить подтверждение перед стиранием
170             return true; // Событие меню обработано
171         case R.id.save:
172             doodleView.saveImage(); // Сохранить текущее изображение
173             return true; // Событие меню обработано
174         case R.id.print:
175             doodleView.printImage(); // Напечатать текущее изображение
176             return true; // Событие меню обработано
177     } // Конец switch
178
179     return super.onOptionsItemSelected(item); // Вызов метода суперкласса
180 } // Конец метода onOptionsItemSelected
181

```

Метод `getMenuItem` класса `MenuItem` (строка 154) используется для получения идентификатора ресурса выбранной команды меню, после чего приложение выполняет различные действия в зависимости от выбора пользователя. Эти действия перечислены ниже.

- `R.id.color` — в строках 157–158 создается и отображается фрагмент `ColorDialogFragment` (раздел 7.7), с помощью которого пользователь выбирает новый цвет линии рисунка.

- ❑ R.id.lineWidth — в строках 161–163 создается и отображается фрагмент `LineWidthDialogFragment` (раздел 7.8), с помощью которого пользователь выбирает новую толщину линии.
- ❑ R.id.eraser — в строке 166 для `doodleView` выбирается белый цвет рисования, при котором движения пальцев стирают изображение.
- ❑ R.id.clear — в строке 169 вызывается метод `confirmErase` (листинг 7.7), который отображает фрагмент `EraseImageDialogFragment` (раздел 7.9) и предлагает пользователю подтвердить стирание изображения.
- ❑ R.id.save — в строке 172 для представления `doodleView` вызывается метод `saveImage`, сохраняющий текущий рисунок в галерее устройства.
- ❑ R.id.print — в строке 175 для представления `doodleView` вызывается метод `printImage`, позволяющий сохранить изображение в формате PDF или вывести его на печать.

Методы `getDoodleView` и `setDialogOnScreen`

Методы `getDoodleView` и `setDialogOnScreen` (листинг 7.9) вызываются методами субклассов `DialogFragment`, используемыми в нашем приложении. Метод `getDoodleView` возвращает ссылку на объект `DoodleView` текущего фрагмента, чтобы реализация `DialogFragment` могла назначить цвет и толщину линии или стереть изображение. Метод `setDialogOnScreen` вызывается методами жизненного цикла фрагмента субклассов `DialogFragment` приложения; он устанавливает флаг присутствия диалогового окна на экране.

Листинг 7.9. Методы `getDoodleView` и `setDialogOnScreen`

```
182 // Возвращает объект DoodleView
183 public DoodleView getDoodleView()
184 {
185     return doodleView;
186 }
187
188 // Устанавливает флаг отображения диалогового окна
189 public void setDialogOnScreen(boolean visible)
190 {
191     dialogOnScreen = visible;
192 }
193 }
```

7.6. Класс `DoodleView`

Класс `DoodleView` (листинги 7.10–7.23) обрабатывает касания пользователя и рисует соответствующие линии.

Команда package и команды import

В листинге 7.10 приведены команды `package` и `import`, а также поля класса `DoodleView`. Новые классы и интерфейсы выделены жирным шрифтом. Многие из них описаны в разделе 7.2, а остальные будут рассматриваться по мере их использования в классе `DoodleView`.

Листинг 7.10. Команды package и import класса DoodleView

```

1 // DoodleView.java
2 // Главное представление приложения Doodlz.
3 package com.deitel.doodlz;
4
5 import java.util.HashMap;
6 import java.util.Map;
7
8 import android.content.Context;
9 import android.graphics.Bitmap;
10 import android.graphics.Canvas;
11 import android.graphics.Color;
12 import android.graphics.Paint;
13 import android.graphics.Path;
14 import android.graphics.Point;
15 import android.os.Build;
16 import android.provider.MediaStore;
17 import android.support.v4.print.PrintHelper;
18 import android.util.AttributeSet;
19 import android.view.GestureDetector;
20 import android.view.GestureDetector.SimpleOnGestureListener;
21 import android.view.Gravity;
22 import android.view.MotionEvent;
23 import android.view.View;
24 import android.widget.Toast;
25

```

Статические переменные и переменные экземпляров DoodleView

Переменные класса `DoodleView` (листинг 7.11, строки 30–43) используются для управления данными набора линий, нарисованных пользователем, и для прорисовки этих линий. В строке 38 создается объект `pathMap`, связывающий идентификатор каждого пальца (называемый *указателем*) с объектом `Path` для рисуемых линий. В строках 39–40 создается объект `previousPointMap`, в котором хранится последняя точка каждого пальца, — с перемещением пальца проводится линия от текущей точки к предыдущей. Другие поля будут рассматриваться по мере их использования в классе `DoodleView`.

Листинг 7.11. Статические переменные и переменные экземпляров класса DoodleView

```

26 // Основной экран, на котором рисует пользователь
27 public class DoodleView extends View
28 {

```

```

29 // Смещение, необходимое для продолжения рисования
30 private static final float TOUCH_TOLERANCE = 10;
31
32 private Bitmap bitmap; // Область рисования для вывода или сохранения
33 private Canvas bitmapCanvas; // Используется для рисования на Bitmap
34 private final Paint paintScreen; // Используется для вывода Bitmap на экран
35 private final Paint paintLine; // используется для рисования линий на Bitmap
36
37 // Данные нарисованных контуров Path и содержащихся в них точек
38 private final Map<Integer, Path> pathMap = new HashMap<Integer, Path>();
39 private final Map<Integer, Point> previousPointMap =
40     new HashMap<Integer, Point>();
41
42 // Для сокрытия/отображения системных панелей
43 private GestureDetector singleTapDetector;
44

```

Конструктор doodleview

Конструктор (листинг 7.12) инициализирует некоторые переменные экземпляров класса — две коллекции Map инициализируются в их объявлении в листинге 7.11. Стока 49 создает объект paintScreen класса Paint, который будет использоваться для рисования на экране, а в строке 52 создается объект paintLine класса Paint, определяющий настройки линии, которую в настоящий момент рисует пользователь. Параметры paintLine задаются в строках 53–57. Метод setAntiAlias класса Paint вызывается со значением true для включения режима сглаживания границ линий. Наконец, вызов метода setStyle класса Paint назначает объекту Paint стиль Paint.Style.STROKE. Выбирая стиль STROKE, FILL или FILL_AND_STROKE, можно выбрать режим рисования линий, заполненных фигур без контура и заполненных фигур с контуром соответственно. По умолчанию, используется значение Paint.Style.FILL. Толщина линии задается методом setStrokeWidth объекта Paint. При этом приложению назначается принятая по умолчанию толщина линии в 5 пикселов. Мы также используем метод setStrokeCap класса Paint для вывода линий с закругленными концами (Paint.Cap.ROUND). В строках 60–61 создается объект GestureDetector, который использует объект singleTapListener для проверки событий одиночных касаний.

Листинг 7.12. Конструктор DoodleView

```

45 // Конструктор DoodleView инициализирует объект DoodleView
46 public DoodleView(Context context, AttributeSet attrs)
47 {
48     super(context, attrs); // Конструктору View передается контекст
49     paintScreen = new Paint(); // Используется для вывода на экран
50
51     // Исходные параметры рисуемых линий
52     paintLine = new Paint();
53     paintLine.setAntiAlias(true); // Сглаживание краев
54     paintLine.setColor(Color.BLACK); // По умолчанию черный цвет
55     paintLine.setStyle(Paint.Style.STROKE); // Сплошная линия
56     paintLine.setStrokeWidth(5); // Толщина линии по умолчанию
57     paintLine.setStrokeCap(Paint.Cap.ROUND); // Закругленные концы

```

```

58      // GestureDetector для одиночных касаний
59      singleTapDetector =
60          new GestureDetector(getContext(), singleTapListener);
61
62      }
63

```

Переопределенный метод onSizeChanged

Размер DoodleView определяется только после того, как представление будет заполнено и добавлено в иерархию представлений `MainActivity`; следовательно, определить размер объекта `Bitmap`, используемого при рисовании, в методе `onCreate` не удастся. По этой причине мы переопределяем метод `onSizeChanged` класса `View` (листинг 7.13), который вызывается при изменении размера `DoodleView` — например, при добавлении в иерархию представлений активности или при повороте устройства. В нашем приложении метод `onSizeChanged` вызывается только при добавлении в иерархию представлений активности `Doodlz`, потому что приложение всегда выполняется в портретном режиме на телефонах и малых планшетах и в альбомном режиме на больших планшетах.

Листинг 7.13. Переопределенный метод onSizeChanged

```

64      // Метод onSizeChanged создает Bitmap и Canvas после появления приложения
65      @Override
66      public void onSizeChanged(int w, int h, int oldW, int oldH)
67      {
68          bitmap = Bitmap.createBitmap(getWidth(), getHeight(),
69              Bitmap.Config.ARGB_8888);
70          bitmapCanvas = new Canvas(bitmap);
71          bitmap.eraseColor(Color.WHITE); // Bitmap стирается белым цветом
72      }
73

```

Статический метод `createBitmap` класса `Bitmap` создает объект `Bitmap` с заданной шириной и высотой — в качестве размеров `Bitmap` используются ширина и высота `DoodleView`. В последнем аргументе `createBitmap` передается *кодировка Bitmap*, определяющая, в каком формате хранится каждый пиксель `Bitmap`. Константа `Bitmap.Config.ARGB_8888` означает, что цвет каждого пикселя хранится в четырех байтах (по одному байту для альфа-канала, красной, зеленой и синей составляющих). Затем мы создаем новый объект `Canvas`, который используется для рисования прямо на `Bitmap`. Наконец, метод `eraseColor` класса `Bitmap` заполняет `Bitmap` белыми пикселями (по умолчанию `Bitmap` использует черный цвет фона).

Методы clear, setDrawingColor, getDrawingColor, setLineWidth и getLineWidth

В листинге 7.14 определяются методы `clear` (строки 75–81), `setDrawingColor` (строки 84–87), `getDrawingColor` (строки 90–93), `setLineWidth` (строки 96–99) и `getLineWidth` (строки 102–105) класса `DoodleView`, вызываемые из `DoodleFragment`. Метод `clear`, используемый в `EraseImageDialogFragment`, очищает коллекции `pathMap` и `previousPointMap`, стирает `Bitmap`, заполняя все пиксели белым цветом, а затем

вызывает унаследованный от View метод `invalidate`, чтобы сообщить о необходимости перерисовки View. Затем система автоматически определяет, когда должен быть вызван метод `onDraw` класса View. Метод `setDrawingColor` изменяет текущий цвет линий, назначая цвет объекта `paintLine`. Метод `setColor` класса Paint получает значение `int`, представляющее новый цвет в формате ARGB. Метод `getDrawingColor` возвращает текущий цвет, который используется в `ColorDialogFragment`. Метод `setLineWidth` задает толщину линии `paintLine` в пикселях. Метод `getLineWidth` возвращает текущую толщину линии, которая используется в `LineWidthDialogFragment`.

Листинг 7.14. Методы `clear`, `setDrawingColor`, `getDrawingColor`, `setLineWidth` и `getLineWidth` класса `DoodleView`

```
74 // Стирание рисунка
75 public void clear()
76 {
77     pathMap.clear(); // Удалить все контуры
78     previousPointMap.clear(); // Удалить все предыдущие точки
79     bitmap.eraseColor(Color.WHITE); // Очистка изображения
80     invalidate(); // Перерисовать изображение
81 }
82
83 // Назначение цвета рисуемой линии
84 public void setDrawingColor(int color)
85 {
86     paintLine.setColor(color);
87 }
88
89 // Получение цвета рисуемой линии
90 public int getDrawingColor()
91 {
92     return paintLine.getColor();
93 }
94
95 // Назначение толщины рисуемой линии
96 public void setLineWidth(int width)
97 {
98     paintLine.setStrokeWidth(width);
99 }
100
101 // Получение толщины рисуемой линии
102 public int getLineWidth()
103 {
104     return (int) paintLine.getStrokeWidth();
105 }
106
```

Переопределенный метод `onDraw`

Когда представлению потребуется перерисовка, вызывается его метод `onDraw`. В листинге 7.15 метод `onDraw` переопределяется для отображения объекта `Bitmap` (содержащего изображение) на `DoodleView`, для чего вызывается метод `drawBitmap` аргумента `Canvas`. В первом аргументе передается объект `Bitmap`, в двух других — координаты

х и у левого верхнего угла `Bitmap` в представлении, и в последнем — объект `Paint` с характеристиками графического вывода. Далее цикл в строках 115–116 перебирает и отображает объекты `Path` текущего рисунка. Для каждого целочисленного ключа в `pathMap` соответствующий объект `Path` передается методу `drawPath` объекта `Canvas` для прорисовки с использованием объекта `paintLine`, определяющего толщину и цвет линии.

Листинг 7.15. Переопределенный метод `onDraw` класса `View`

```

107    // Вызывается при каждой перерисовке представления
108    @Override
109    protected void onDraw(Canvas canvas)
110    {
111        // Вывод фона
112        canvas.drawBitmap(bitmap, 0, 0, paintScreen);
113
114        // Для каждого контура в прорисовке
115        for (Integer key : pathMap.keySet())
116            canvas.drawPath(pathMap.get(key), paintLine); // Нарисовать линию
117    }
118

```

Методы `hideSystemBars` и `showSystemBars` класса `DoodleView`

В этом приложении используется новый *режим погружения*, появившийся в Android 4.4, чтобы пользователь мог рисовать на всем экране. Когда пользователь касается экрана, слушатель `SimplyOnGestureListener` класса `GestureDetector` (листинг 7.17) определяет, отображаются ли системные панели и панель действий. Если панели отображаются, вызывается метод `hideSystemBars` (листинг 7.16, строки 120–130); в противном случае вызывается метод `showSystemBars` (листинг 7.16, строки 133–140). В нашем приложении режим погружения активизируется только для Android 4.4. Итак, оба метода сначала проверяют, что версия Android на устройстве (`Build.VERSION.SDK_INT`) больше либо равна константе Android 4.4 (API уровня 19) — `Build.VERSION_CODES.KITKAT`. В этом случае оба метода используют метод `setSystemUiVisibility` класса `View` для настройки системных панелей и панели действий. Чтобы скрыть системные панели и панель действий с переходом интерфейса в режим погружения, мы передаем `setSystemUiVisibility` константы, объединенные поразрядным оператором ИЛИ (`|`), в строках 124–129. Чтобы отобразить системные панели и панель действий, мы передаем `setSystemUiVisibility` константы в строках 137–139. Эти комбинации констант `View` гарантируют, что представление `DoodleView` не будет изменять размеры при каждом отображении и скрытии системных панелей и панели действий. Вместо этого панели накладываются на `DoodleView` — то есть часть `DoodleView` временно скрывается, когда системные панели и панель действий находятся на экране. Константа `View.SYSTEM_UI_FLAG_IMMERSIVE` появилась в Android 4.4. За дополнительной информацией о режиме погружения обращайтесь по адресу

<http://developer.android.com/training/system-ui/immersive.html>

Листинг 7.16. Методы hideSystemBars и showSystemBars класса DoodleView

```

119     // Скрыть системные панели и панель действий
120     public void hideSystemBars()
121     {
122         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT)
123             setSystemUiVisibility(
124                 View.SYSTEM_UI_FLAG_LAYOUT_STABLE |
125                 View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION |
126                 View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN |
127                 View.SYSTEM_UI_FLAG_HIDE_NAVIGATION |
128                 View.SYSTEM_UI_FLAG_FULLSCREEN |
129                 View.SYSTEM_UI_FLAG_IMMERSIVE);
130     }
131
132     // Отобразить системные панели и панель действий
133     public void showSystemBars()
134     {
135         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT)
136             setSystemUiVisibility(
137                 View.SYSTEM_UI_FLAG_LAYOUT_STABLE |
138                 View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION |
139                 View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
140     }
141

```

Анонимный внутренний класс, реализующий интерфейс SimpleOnGestureListener

В листинге 7.17 создается реализация `SimpleOnGestureListener` с именем `singleTapListener`, которая регистрируется в строках 60–61 (листинг 7.4) в объекте `GestureDetector`. Напомним, что `SimpleOnGestureListener` — класс-адаптер, реализующий интерфейсы `OnGestureListener` и `OnDoubleTapListener`. Методы просто возвращают `false` — признак того, что события не были обработаны. Переопределяется только метод `onSingleTap` (строки 146–155), который вызывается при касании экрана. Мы проверяем, отображаются ли системные панели и панель приложения (строки 149–150), вызывая метод `getSystemUiVisibility` класса `View` и объединяя его результат с константой `View.SYSTEM_UI_FLAG_HIDE_NAVIGATION`. Если результат равен 0, то системные панели и панель приложения отображаются на экране, поэтому мы вызываем метод `hideSystemBars`; в противном случае вызывается метод `showSystemBars`. Возвращаемое значение `true` указывает, что событие одиночного касания было обработано.

Листинг 7.17. Анонимный внутренний класс, реализующий интерфейс SimpleOnGestureListener

```

142     // create SimpleOnGestureListener for single tap events
143     private SimpleOnGestureListener singleTapListener =
144         new SimpleOnGestureListener()
145     {
146         @Override

```

```

147     public boolean onSingleTapUp(MotionEvent e)
148     {
149         if ((getSystemUiVisibility() &
150             View.SYSTEM_UI_FLAG_HIDE_NAVIGATION) == 0)
151             hideSystemBars();
152         else
153             showSystemBars();
154         return true;
155     }
156 };
157

```

Переопределение метода onTouchEvent класса View

Метод onTouchEvent (листинг 7.18) вызывается при получении представлением события касания. Android поддерживает *многоточечные* касания, то есть прикосновение к экрану сразу несколькими пальцами. В любой момент пользователь может приложить к экрану другие пальцы или отвести их от экрана. По этой причине каждому пальцу присваивается уникальный *идентификатор*, который идентифицирует его в разных событиях касания. Мы используем его для идентификации объектов Path, представляющих каждую линию, рисуемую в настоящий момент. Объекты Path хранятся в коллекции pathMap.

Листинг 7.18. Переопределенный метод onTouchEvent класса View

```

158 // Обработка события касания
159 @Override
160 public boolean onTouchEvent(MotionEvent event)
161 {
162     // Получение типа события и идентификатор пальца,
163     // если событие касания произошло в версии KitKat и выше
164     if (singleTapDetector.onTouchEvent(event))
165         return true;
166
167     int action = event.getActionMasked(); // Тип события
168     int actionIndex = event.getActionIndex(); // Указатель (палец)
169     // Определить, что происходит: начало касания,
170     // конец касания, перемещение
171     if (action == MotionEvent.ACTION_DOWN ||
172         action == MotionEvent.ACTION_POINTER_DOWN)
173     {
174         touchStarted( event.getX(actionIndex), event.getY(actionIndex),
175                     event.getPointerId(actionIndex));
176     }
177     else if (action == MotionEvent.ACTION_UP ||
178             action == MotionEvent.ACTION_POINTER_UP)
179     {
180         touchEnded(event.getPointerId(actionIndex));
181     }
182     else
183     {
184         touchMoved(event);
185     }
186 }

```

```
185      }
186
187      invalidate(); // Перерисовка
188      return true;
189  } // Конец метода onTouchEvent
190
```

Когда происходит событие касания, строка 164 вызывает метод `onTouchEvent` объекта `singleTapDetector (GestureDetector)` для определения того, было ли событие касанием для отображения или сокрытия системных панелей и панели приложения. В этом случае метод немедленно возвращает управление.

Метод `getActionMasked` (строка 167) класса `MotionEvent` возвращает значение `int` — признак типа события `MotionEvent`, который может использоваться в сочетании с константами класса `MotionEvent` для определения того, как должно обрабатываться каждое событие. Метод `getActionIndex` (строка 168) класса `MotionEvent` возвращает целочисленный индекс пальца, сгенерировавшего событие. Этот индекс не является уникальным идентификатором пальца — он всего лишь определяет, в какой позиции объекта `MotionEvent` хранится информация от этого пальца. Чтобы получить уникальный идентификатор пальца, сохраняющийся между событиями `MotionEvent` до того, как пользователь отведет палец от экрана, мы воспользуемся методом `getPointerID` (строки 175 и 180) класса `MotionEvent`; этот метод получает индекс пальца в аргументе.

Если обнаружено действие `MotionEvent.ACTION_DOWN` или `MotionEvent.ACTION_POINTER_DOWN` (строки 171–172), значит пользователь коснулся экрана новым пальцем. Первый палец, коснувшийся экрана, генерирует событие `MotionEvent.ACTION_DOWN`, а все остальные пальцы генерируют события `MotionEvent.ACTION_POINTER_DOWN`. В таких случаях вызывается метод `touchStarted` (листинг 7.19) для хранения исходных координат касания. Если обнаружено действие `MotionEvent.ACTION_UP` или `MotionEvent.ACTION_POINTER_UP`, значит пользователь отвел палец от экрана, поэтому вызывается метод `touchEnded` (листинг 7.21) для рисования завершенного контура `Path` на растровом изображении. Для остальных событий касания вызывается метод `touchMoved` (листинг 7.20) для рисования линий. После обработки строки 187 вызывает унаследованный от `View` метод `invalidate` для перерисовки экрана, а строка 188 возвращает `true` — признак того, что событие было обработано.

Метод `touchStarted` класса `DoodleView`

Метод `touchStarted` (листинг 7.19) вызывается при первом соприкосновении пальца с экраном. В аргументах передаются координаты касания и идентификатор пальца. Если объект `Path` для заданного идентификатора уже существует (строка 198), мы вызываем метод `reset` объекта `Path` для стирания всех существующих точек, чтобы заново использовать `Path` для нового контура. В противном случае создается новый объект `Path`, он добавляется в `pathMap`, а новый объект `Point` добавляется в `previousPointMap`. В строках 213–215 вызывается метод `moveTo` объекта `Path` для задания начальных координат `Path` и задания значений `x` и `y` нового объекта `Point`.

Листинг 7.19. Метод touchStarted класса DoodleView

```

191 // Вызывается при касании экрана
192 private void touchStarted(float x, float y, int lineID)
193 {
194     Path path; // Для хранения контура с заданным идентификатором
195     Point point; // Для хранения последней точки в контуре
196
197     // Если для lineID уже существует объект Path
198     if (pathMap.containsKey(lineID))
199     {
200         path = pathMap.get(lineID); // Получение Path
201         path.reset(); // Очистка Path с началом нового касания
202         point = previousPointMap.get(lineID); // Последняя точка Path
203     }
204     else
205     {
206         path = new Path();
207         pathMap.put(lineID, path); // Добавление Path в Map
208         point = new Point(); // Создание нового объекта Point
209         previousPointMap.put(lineID, point); // Добавление Point в Map
210     }
211
212     // Переход к координатам касания
213     path.moveTo(x, y);
214     point.x = (int) x;
215     point.y = (int) y;
216 } // Конец метода touchStarted
217

```

Метод touchMoved класса DoodleView

Метод `touchMoved` (листинг 7.20) вызывается при проведении одним или несколькими пальцами по экрану. Системный объект `MotionEvent`, передаваемый из `onTouchEvent`, содержит информацию о нескольких перемещениях на экране, если они происходят одновременно. Метод `getPointerCount` класса `MotionEvent` (строка 222) возвращает количество касаний, описываемых объектом `MotionEvent`. Для каждого касания идентификатор пальца сохраняется в `pointerID` (строка 225), а соответствующий индекс из `MotionEvent` (строка 226) сохраняется в `pointerIndex`. Затем проверяется наличие соответствующего объекта `Path` в коллекции `pathMap` (строка 229). Если объект присутствует в коллекции, методы `getX` и `getY` класса `MotionEvent` используются для получения последних координат события перетаскивания для заданного значения `pointerIndex`. Мы получаем объект `Path` и последний объект `Point` для `pointerID` из соответствующих коллекций `HashMap`, после чего вычисляем различия между последней и текущей точкой — объект `Path` должен обновиться только в том случае, если величина перемещения превысила константу `TOUCH_TOLERANCE`. Мы делаем это, потому что многие устройства обладают высокой чувствительностью и могут генерировать события `MotionEvent` для малых перемещений, даже когда пользователь пытается держать палец неподвижно. Если пользователь переместил палец на расстояние, превышающее `TOUCH_TOLERANCE`, мы используем метод `quadTo` класса `Path` (строки 248–249) для добавления геометрической кривой (а конкретно

квадратичной кривой Безье) от предыдущей точки к новой, после чего обновляем данные последней точки для этого пальца.

Листинг 7.20. Метод touchMoved класса DoodleView

```

218     // Вызывается при перемещении пальца по экрану
219     private void touchMoved(MotionEvent event)
220     {
221         // Для каждого указателя (пальца) в объекте MotionEvent
222         for (int i = 0; i < event.getPointerCount(); i++)
223         {
224             // Получить идентификатор и индекс указателя
225             int pointerID = event.getPointerId(i);
226             int pointerIndex = event.findPointerIndex(pointerID);
227
228             // Если существует объект Path, связанный с указателем
229             if (pathMap.containsKey(pointerID))
230             {
231                 // Получить новые координаты для указателя
232                 float newX = event.getX(pointerIndex);
233                 float newY = event.getY(pointerIndex);
234
235                 // Получить объект Path и предыдущий объект Point,
236                 // связанный с указателем
237                 Path path = pathMap.get(pointerID);
238                 Point point = previousPointMap.get(pointerID);
239
240                 // Вычислить величину смещения от последнего обновления
241                 float deltaX = Math.abs(newX - point.x);
242                 float deltaY = Math.abs(newY - point.y);
243
244                 // Если расстояние достаточно велико
245                 if (deltaX >= TOUCH_TOLERANCE || deltaY >= TOUCH_TOLERANCE)
246                 {
247                     // Расширение контура до новой точки
248                     path.quadTo(point.x, point.y, (newX + point.x) / 2,
249                               (newY + point.y) / 2);
250
251                     // Сохранение новых координат
252                     point.x = (int) newX;
253                     point.y = (int) newY;
254                 }
255             }
256         }
257     } // Конец метода touchMoved
258

```

Метод touchEnded класса DoodleView

Метод touchEnded (листинг 7.21) вызывается, когда пользователь отводит палец от экрана. В аргументе метода передается идентификатор пальца (`lineID`), касание которого только что закончилось. Стока 262 получает соответствующий объект `Path`. В строке 263 вызывается `drawPath` объекта `bitmapCanvas` для рисования `Path`

на объекте `Bitmap` с именем `bitmap` перед очисткой `Path`. Сброс `Path` не приводит к стиранию соответствующей нарисованной линии с экрана, потому что эти линии уже были нарисованы на объекте `bitmap`, отображаемом на экране. Линии, рисуемые пользователем в настоящее время, рисуются поверх `bitmap`.

Листинг 7.21. Метод touchEnded класса DoodleView

```

259     // Вызывается при завершении касания
260     private void touchEnded(int lineID)
261     {
262         Path path = pathMap.get(lineID); // Получение объекта Path
263         bitmapCanvas.drawPath(path, paintLine); // Рисование на bitmapCanvas
264         path.reset(); // Сброс объекта Path
265     }
266

```

Метод saveImage класса DoodleView

Метод `saveImage` (листинг 7.22) сохраняет текущее изображение, создавая файл в галерее устройства. Стока 271 создает имя файла для изображения, после чего в строках 274–276 изображение сохраняется, для чего вызывается метод `insertImage` класса `MediaStore.Images.Media`. Метод получает четыре аргумента:

- ❑ объект `ContentResolver`, используемый методом для определения места хранения изображения на устройстве;
- ❑ объект `Bitmap` с сохраняемым рисунком;
- ❑ имя изображения;
- ❑ описание изображения.

Метод `insertImage` возвращает строку, описывающую местонахождение изображения на устройстве, или `null`, если сохранить изображение не удалось. Строки 278–295 проверяют, было ли изображение сохранено, и отображают соответствующее времменное окно `Toast`.

Листинг 7.22. Метод saveImage класса DoodleView

```

267     // Сохранение текущего изображения в галерее
268     public void saveImage()
269     {
270         // Имя состоит из префикса "Doodlz" и текущего времени
271         String name = "Doodlz" + System.currentTimeMillis() + ".jpg";
272
273         // Сохранение изображения в галерее устройства
274         String location = MediaStore.Images.Media.insertImage(
275             getApplicationContext().getContentResolver(), bitmap, name,
276             "Doodlz Drawing");
277
278         if (location != null) // Изображение было сохранено
279         {
280             // Вывод сообщения об успешном сохранении

```

```

281     Toast message = Toast.makeText(getApplicationContext(),
282         R.string.message_saved, Toast.LENGTH_SHORT);
283     message.setGravity(Gravity.CENTER, message.getXOffset() / 2,
284         message.getYOffset() / 2);
285     message.show();
286 }
287 else
288 {
289     // Вывод сообщения об ошибке сохранения
290     Toast message = Toast.makeText(getApplicationContext(),
291         R.string.message_error_saving, Toast.LENGTH_SHORT);
292     message.setGravity(Gravity.CENTER, message.getXOffset() / 2,
293         message.getYOffset() / 2);
294     message.show();
295 }
296 } // Конец метода saveImage
297

```

Метод printImage класса DoodleView

На устройствах с Android 4.4 и выше метод `printImage` (листинг 7.23) использует класс `PrintHelper` из Android Support Library для печати текущего рисунка. Стока 301 сначала убеждается в том, что поддержка печати доступна на устройстве. Если проверка дает положительный результат, строка 304 создает объект `PrintHelper`. Затем строка 307 задает *режим масштабирования* устройства — значение `PrintHelper.SCALE_MODE_FIT` сообщает, что размеры изображения должны быть подогнаны под размеры печатаемой области листа. Также существует режим `PrintHelper.SCALE_MODE_FILL`, при котором изображение заполняет лист (возможно, с отсечением части изображения). Наконец, строка 308 вызывает метод `printBitmap` класса `PrintHelper`, передавая в аргументах имя задания печати (используемое принтером для идентификации) и объект `Bitmap` с выводимым изображением. На экране появляется диалоговое окно печати Android, в котором пользователь может выбрать между сохранением изображения в формате PDF на устройстве и печатью на принтере.

Листинг 7.23. Метод printImage класса DoodleView

```

298     // Печать текущего изображения
299     public void printImage()
300     {
301         if (PrintHelper.systemSupportsPrint())
302         {
303             // Использование класса PrintHelper для печати
304             PrintHelper printHelper = new PrintHelper(getApplicationContext());
305
306             // Изображение масштабируется и выводится на печать
307             printHelper.setScaleMode(PrintHelper.SCALE_MODE_FIT);
308             printHelper.printBitmap("Doodlz Image", bitmap);
309         }
310     else
311     {
312         // Вывод сообщения о том, что система не поддерживает печать
313         Toast message = Toast.makeText(getApplicationContext(),

```

```

314         R.string.message_error_printing, Toast.LENGTH_SHORT);
315     message.setGravity(Gravity.CENTER, message.getXOffset() / 2,
316         message.getYOffset() / 2);
317     message.show();
318   }
319 }
320 } // Конец класса DoodleView

```

7.7. Класс ColorDialogFragment

Класс `ColorDialogFragment` (листинги 7.24–7.28) расширяет `DialogFragment` для создания окна `AlertDialog`, в котором определяется цвет линии. Переменные экземпляра класса (строки 19–24) используются для обращения к элементам графического интерфейса, предназначенным для выбора нового цвета, отображения его образца и сохранения цвета в виде 32-разрядного значения `int`, представляющего составляющие цвета в формате ARGB.

Листинг 7.24. Команда `package`, команды `import` и переменные экземпляров класса `ColorDialogFragment`

```

1 // ColorDialogFragment.java
2 // Используется для выбора цвета линии в DoodleView
3 package com.deitel.doodlz;
4
5 import android.app.Activity;
6 import android.app.AlertDialog;
7 import android.app.Dialog;
8 import android.app.DialogFragment;
9 import android.content.DialogInterface;
10 import android.graphics.Color;
11 import android.os.Bundle;
12 import android.view.View;
13 import android.widget.SeekBar;
14 import android.widget.SeekBar.OnSeekBarChangeListener;
15
16 // Класс диалогового окна выбора цвета
17 public class ColorDialogFragment extends DialogFragment
18 {
19     private SeekBar alphaSeekBar;
20     private SeekBar redSeekBar;
21     private SeekBar greenSeekBar;
22     private SeekBar blueSeekBar;
23     private View colorView;
24     private int color;
25

```

Переопределенный метод `onCreateDialog` класса `DialogFragment`

Метод `onCreateDialog` (листинг 7.25) заполняет пользовательское представление (строки 32–34), определенное в файле `fragment_color.xml` с графическим интерфейсом

выбора цвета, после чего связывает представление с окном `AlertDialog`, вызывая метод `setView` объекта `AlertDialog.Builder` (строка 35). Строки 42–50 получают ссылки на компоненты `SeekBar` и `colorView` диалогового окна. Затем строки 53–56 регистрируют объект `colorChangedListener` (листинг 7.28) слушателем событий компонентов `SeekBar`.

Листинг 7.25. Переопределенный метод `onCreateDialog` класса `DialogFragment`

```
26     // Создание и возвращение объекта AlertDialog
27     @Override
28     public Dialog onCreateDialog(Bundle bundle)
29     {
30         AlertDialog.Builder builder =
31             new AlertDialog.Builder(getActivity());
32         View colorDialogView =
33             getActivity().getLayoutInflater().inflate(
34                 R.layout.fragment_color, null);
35         builder.setView(colorDialogView); // Добавление GUI в диалоговое окно
36
37         // Назначение сообщения AlertDialog
38         builder.setTitle(R.string.title_color_dialog);
39         builder.setCancelable(true);
40
41         // Получение значений SeekBar и назначение слушателей onChange
42         alphaSeekBar = (SeekBar) colorDialogView.findViewById(
43             R.id.alphaSeekBar);
44         redSeekBar = (SeekBar) colorDialogView.findViewById(
45             R.id.redSeekBar);
46         greenSeekBar = (SeekBar) colorDialogView.findViewById(
47             R.id.greenSeekBar);
48         blueSeekBar = (SeekBar) colorDialogView.findViewById(
49             R.id.blueSeekBar);
50         colorView = colorDialogView.findViewById(R.id.colorView);
51
52         // Регистрация слушателей событий SeekBar
53         alphaSeekBar.setOnSeekBarChangeListener(colorChangedListener);
54         redSeekBar.setOnSeekBarChangeListener(colorChangedListener);
55         greenSeekBar.setOnSeekBarChangeListener(colorChangedListener);
56         blueSeekBar.setOnSeekBarChangeListener(colorChangedListener);
57
58         // Использование текущего цвета линии для инициализации SeekBar
59         final DoodleView doodleView = getDoodleFragment().getDoodleView();
60         color = doodleView.getDrawingColor();
61         alphaSeekBar.setProgress(Color.alpha(color));
62         redSeekBar.setProgress(Color.red(color));
63         greenSeekBar.setProgress(Color.green(color));
64         blueSeekBar.setProgress(Color.blue(color));
65
66         // Добавление кнопки назначения цвета
67         builder.setPositiveButton(R.string.button_set_color,
68             new DialogInterface.OnClickListener()
69             {
70                 public void onClick(DialogInterface dialog, int id)
71                 {
```

```

72             doodleView.setDrawingColor(color);
73         }
74     }
75 ); // Конец вызова setPositiveButton
76
77 return builder.create(); // Возвращение диалогового окна
78 } // Конец метода onCreateDialog
79

```

В строке 59 вызывается метод `getDoodleFragment` (листинг 7.26) для получения ссылки на фрагмент `DoodleFragment`, после чего вызывается метод `getDoodleView` класса `DoodleFragment` для получения объекта `DoodleView`. Строки 60–64 получают текущий цвет рисования `DoodleView`, который используется для инициализации компонентов `SeekBar`. Статические методы `alpha`, `red`, `green` и `blue` класса `Color` извлекают значения составляющих ARGB, а метод `setProgress` класса `SeekBar` устанавливает ползунки в нужных позициях. В строках 67–75 позитивная кнопка `AlertDialog` применяет новый цвет рисования `DoodleView`. Стока 77 возвращает объект `AlertDialog` для отображения на экране.

Метод `getDoodleFragment`

Метод `getDoodleFragment` (листинг 7.26) использует объект `FragmentManager` для получения ссылки на `DoodleFragment`.

Листинг 7.26. Метод `getDoodleFragment`

```

80 // Получение ссылки на DoodleFragment
81 private DoodleFragment getDoodleFragment()
82 {
83     return (DoodleFragment) getFragmentManager().findFragmentById(
84         R.id.doodleFragment);
85 }
86

```

Переопределенные методы `onAttach` и `onDetach` жизненного цикла фрагмента

При добавлении `ColorDialogFragment` в родительскую активность вызывается метод `onAttach` (листинг 7.27, строки 88–96). Стока 92 получает ссылку на `DoodleFragment`. Если эта ссылка отлична от `null`, то строка 95 вызывает метод `setDialogOnScreen` класса `DoodleFragment` для установки флага отображения диалогового окна `Choose Color`. При удалении `ColorDialogFragment` из родительской активности вызывается метод `onDetach` (строки 99–107). В строке 106 вызывается метод `setDialogOnScreen` класса `DoodleFragment`, указывающий, что диалоговое окно `Choose Color` перестало отображаться на экране.

Листинг 7.27. Переопределенные методы `onAttach` и `onDetach` жизненного цикла фрагмента

```

87 // Сообщает DoodleFragment, что диалоговое окно находится на экране
88 @Override

```

```

89  public void onAttach(Activity activity)
90  {
91      super.onAttach(activity);
92      DoodleFragment fragment = getDoodleFragment();
93
94      if (fragment != null)
95          fragment.setDialogOnScreen(true);
96  }
97
98 // Сообщает DoodleFragment, что диалоговое окно не отображается
99 @Override
100 public void onDetach()
101 {
102     super.onDetach();
103     DoodleFragment fragment = getDoodleFragment();
104
105     if (fragment != null)
106         fragment.setDialogOnScreen(false);
107 }
108

```

Анонимный внутренний класс, реализующий интерфейс OnSeekBarChangeListener для обработки событий компонентов SeekBar

В листинге 7.28 определяется анонимный внутренний класс, реализующий интерфейс `OnSeekBarChangeListener` для обработки событий, возникающих при изменении компонентов `SeekBar` в диалоговом окне `Choose Color`. Он регистрируется как обработчик события `SeekBar` в листинге 7.25 (строки 53–56). Метод `onProgressChanged` (строки 115–123) вызывается при изменении позиции ползунка `SeekBar`. Если пользователь переместил ползунок `SeekBar` (строка 118), новый цвет сохраняется в строках 119–121. Статический метод `argb` класса `Color` объединяет значения `SeekBar` в `Color` и возвращает соответствующий цвет в формате `int`. Затем метод `setBackgroundColor` класса `View` используется для обновления `colorView` цветом, соответствующим текущему состоянию `SeekBar`.

Листинг 7.28. Анонимный внутренний класс, реализующий интерфейс OnSeekBarChangeListener для обработки событий компонентов SeekBar

```

109 // OnSeekBarChangeListener для компонентов SeekBar в диалоговом окне
110 private OnSeekBarChangeListener colorChangedListener =
111     new OnSeekBarChangeListener()
112 {
113     // Отображение обновленного цвета
114     @Override
115     public void onProgressChanged(SeekBar seekBar, int progress,
116         boolean fromUser)
117     {
118         if (fromUser) // SeekBar изменено пользователем (не программой)
119             color = Color.argb(alphaSeekBar.getProgress(),
120                 redSeekBar.getProgress(), greenSeekBar.getProgress(),
121                 blueSeekBar.getProgress());

```

```
122         colorView.setBackgroundColor(color);
123     }
124
125     @Override
126     public void onStartTrackingTouch(SeekBar seekBar) // Обязательный метод
127     {
128     }
129
130     @Override
131     public void onStopTrackingTouch(SeekBar seekBar) // Обязательный метод
132     {
133     }
134 }; // Конец colorChanged
135 } // Конец класса ColorDialogFragment
```

7.8. Класс LineWidthDialogFragment

Класс `LineWidthDialogFragment` (листинг 7.29) расширяет `DialogFragment` для создания окна `AlertDialog`, в котором определяется толщина линии. Этот класс аналогичен классу `ColorDialogFragment`, поэтому здесь рассматриваются только важнейшие различия. Единственная переменная класса — компонент `ImageView` (строка 22), в которой рисуется образец линии с текущей толщиной.

Листинг 7.29. Класс LineWidthDialogFragment

```
1 // LineWidthDialogFragment.java
2 // Используется для выбора толщины линии в DoodleView
3 package com.deitel.doodlz;
4
5 import android.app.Activity;
6 import android.app.AlertDialog;
7 import android.app.Dialog;
8 import android.app.DialogFragment;
9 import android.content.DialogInterface;
10 import android.graphics.Bitmap;
11 import android.graphics.Canvas;
12 import android.graphics.Paint;
13 import android.os.Bundle;
14 import android.view.View;
15 import android.widget.ImageView;
16 import android.widget.SeekBar;
17 import android.widget.SeekBar.OnSeekBarChangeListener;
18
19 // Класс диалогового окна выбора цвета
20 public class LineWidthDialogFragment extends DialogFragment
21 {
22     private ImageView widthImageView;
23
24     // Создает и возвращает объект AlertDialog
25     @Override
26     public Dialog onCreateDialog(Bundle bundle)
```

```
27  {
28      AlertDialog.Builder builder =
29          new AlertDialog.Builder(getActivity());
30      View lineWidthDialogView = getActivity().getLayoutInflater().inflate(
31          R.layout.fragment_line_width, null);
32      builder.setView(lineWidthDialogView); // Добавление GUI
33
34      // Назначение сообщения the AlertDialog
35      builder.setTitle(R.string.title_line_width_dialog);
36      builder.setCancelable(true);
37
38      // Получение объекта ImageView
39      widthImageView = (ImageView) lineWidthDialogView.findViewById(
40          R.id.widthImageView);
41
42      // Настройка widthSeekBar
43      final DoodleView doodleView = getDoodleFragment().getDoodleView();
44      final SeekBar widthSeekBar = (SeekBar)
45          lineWidthDialogView.findViewById(R.id.widthSeekBar);
46      widthSeekBar.setOnSeekBarChangeListener(lineWidthChanged);
47      widthSeekBar.setProgress(doodleView.getLineWidth());
48
49      // Добавление кнопки назначения толщины линии
50      builder.setPositiveButton(R.string.button_set_line_width,
51          new DialogInterface.OnClickListener()
52          {
53              public void onClick(DialogInterface dialog, int id)
54              {
55                  doodleView.setLineWidth(widthSeekBar.getProgress());
56              }
57          }
58      ); // Конец вызова setPositiveButton
59
60      return builder.create(); // Возвращает диалоговое окно
61  } // Конец метода onCreateDialog
62
63  // Получение ссылки на DoodleFragment
64  private DoodleFragment getDoodleFragment()
65  {
66      return (DoodleFragment) getSupportFragmentManager().findFragmentById(
67          R.id.doodleFragment);
68  }
69
70  // Сообщает DoodleFragment, что диалоговое окно находится на экране
71  @Override
72  public void onAttach(Activity activity)
73  {
74      super.onAttach(activity);
75      DoodleFragment fragment = getDoodleFragment();
76
77      if (fragment != null)
78          fragment.setDialogOnScreen(true);
79  }
80
81  // Сообщает DoodleFragment, что окно не отображается
```

```

82     @Override
83     public void onDetach()
84     {
85         super.onDetach();
86         DoodleFragment fragment = getDoodleFragment();
87
88         if (fragment != null)
89             fragment.setDialogOnScreen(false);
90     }
91
92     // OnSeekBarChangeListener для SeekBar в диалоговом окне толщины линии
93     private OnSeekBarChangeListener lineWidthChanged =
94         new OnSeekBarChangeListener()
95     {
96         Bitmap bitmap = Bitmap.createBitmap(
97             400, 100, Bitmap.Config.ARGB_8888);
98         Canvas canvas = new Canvas(bitmap); // Связывается с Canvas
99
100        @Override
101        public void onProgressChanged(SeekBar seekBar, int progress,
102            boolean fromUser)
103        {
104            // Настройка объекта Paint для текущего значения SeekBar
105            Paint p = new Paint();
106            p.setColor(
107                getDoodleFragment().getDoodleView().getDrawingColor());
108            p.setStrokeCap(Paint.Cap.ROUND);
109            p.setStrokeWidth(progress);
110
111            // Стирание объекта Bitmap и перерисовка линии
112            bitmap.eraseColor(
113                getResources().getColor(android.R.color.transparent));
114            canvas.drawLine(30, 50, 370, 50, p);
115            widthImageView.setImageBitmap(bitmap);
116        }
117
118        @Override
119        public void onStartTrackingTouch(SeekBar seekBar)
120            // Обязательный метод
121        {
122
123            @Override
124            public void onStopTrackingTouch(SeekBar seekBar)
125                // Обязательный метод
126            {
127            }; // Конец lineWidthChanged
128     }

```

Метод onCreateDialog

Метод `onCreateDialog` (строки 25–61) заполняет пользовательское представление (строки 30–31), определенное в файле `fragment_line_width.xml` с графическим интерфейсом выбора толщины линии, после чего связывает представление с окном

`AlertDialog`, вызывая метод `setView` объекта `AlertDialog.Builder` (строка 32). Строки 39–40 получают ссылку на компонент `ImageView`, в котором будет выводиться образец линии. Затем строки 43–47 получают ссылку на компонент `widthSeekBar`, регистрируют `lineWidthChanged` (строки 93–127) как слушателя `SeekBar` и устанавливают текущую толщину линии как текущее значение `SeekBar`. Строки 50–58 определяют позитивную кнопку диалогового окна таким образом, чтобы при нажатии кнопки `Set Line Width` вызывался метод `setLineWidth` класса `DoodleView`. Стока 60 возвращает объект `AlertDialog` для отображения на экране.

Анонимный внутренний класс, реализующий интерфейс `OnSeekBarChangeListener` для обработки событий `widthSeekBar`

В строках 93–127 определяется слушатель `OnSeekBarChangeListener` с именем `lineWidthChanged`, реагирующий на события при изменении компонента `SeekBar` в диалоговом окне `Choose Line Width`. В строках 96–97 создается объект `Bitmap`, на котором будет выводиться образец, представляющий выбранный толщину линии. В строке 98 создается объект `Canvas` для рисования на `Bitmap`. Метод `onProgressChanged` (строки 100–116) рисует образец линии с текущим цветом и толщиной, определяемой значением `SeekBar`. Сначала в строках 105–109 настраивается объект `Paint` для рисования образца линии. Метод `setStrokeCap` класса `Paint` (строка 108) определяет внешний вид концов линии — в нашем случае используются закругленные концы (`Paint.Cap.ROUND`). В строках 112–113 фон объекта `bitmap` заполняется заранее определенным в Android цветом `android.R.color.transparent` методом `eraseColor` класса `Bitmap`. Объект `canvas` используется для рисования образца линии. Наконец, строка 115 выводит объект `bitmap` на `widthImageView`, передавая его методу `setImageBitmap` компонента `ImageView`.

7.9. Класс `EraseImageDialogFragment`

Класс `EraseImageDialogFragment` (листинг 7.30) расширяет `DialogFragment` для создания окна `AlertDialog`, в котором пользователь подтверждает стирание всего изображения. Этот класс аналогичен классам `ColorDialogFragment` и `LineWidthDialogFragment`, поэтому здесь рассматривается только метод `onCreateDialog` (строки 16–41). Метод создает окно `AlertDialog` с кнопками стирания изображения и отмены. В строках 27–35 кнопка стирания `Erase Image` настраивается как *позитивная* — когда пользователь касается ее, строка 32 в слушателе кнопки вызывает метод `clear` класса `DoodleView` для стирания изображения. Стока 38 настраивает кнопку отмены `Cancel` как *негативную* — если пользователь коснется этой кнопки, диалоговое окно будет закрыто. Стока 40 возвращает объект `AlertDialog`.

Листинг 7.30. Класс `EraseImageDialogFragment`

```
1 // EraseImageDialogFragment.java
2 // Фрагмент для стирания изображения
```

```
3 package com.deitel.doodlz;
4
5 import android.app.Activity;
6 import android.app.AlertDialog;
7 import android.app.Dialog;
8 import android.app.DialogFragment;
9 import android.content.DialogInterface;
10 import android.os.Bundle;
11
12 // Класс диалогового окна
13 public class EraseImageDialogFragment extends DialogFragment
14 {
15     // Создание и возвращение объекта AlertDialog
16     @Override
17     public Dialog onCreateDialog(Bundle bundle)
18     {
19         AlertDialog.Builder builder =
20             new AlertDialog.Builder(getActivity());
21
22         // Назначение сообщения AlertDialog
23         builder.setMessage(R.string.message_erase);
24         builder.setCancelable(false);
25
26         // Добавление кнопки стирания
27         builder.setPositiveButton(R.string.button_erase,
28             new DialogInterface.OnClickListener()
29             {
30                 public void onClick(DialogInterface dialog, int id)
31                 {
32                     getDoodleFragment().getDoodleView().clear(); // Очистка
33                 }
34             });
35     } // Конец вызова setPositiveButton
36
37     // Добавление кнопки отмены
38     builder.setNegativeButton(R.string.button_cancel, null);
39
40     return builder.create(); // Возвращает диалоговое окно
41 } // Конец метода onCreateDialog
42
43 // Получение ссылки на DoodleFragment
44 private DoodleFragment getDoodleFragment()
45 {
46     return (DoodleFragment) getFragmentManager().findFragmentById(
47         R.id.doodleFragment);
48 }
49
50 // Сообщает DoodleFragment, что диалоговое окно находится на экране
51 @Override
52 public void onAttach(Activity activity)
53 {
54     super.onAttach(activity);
55     DoodleFragment fragment = getDoodleFragment();
56 }
```

```
57     if (fragment != null)
58         fragment.setDialogOnScreen(true);
59     }
60
61     // Сообщает DoodleFragment, что окно не отображается
62     @Override
63     public void onDetach()
64     {
65         super.onDetach();
66         DoodleFragment fragment = getDoodleFragment();
67
68         if (fragment != null)
69             fragment.setDialogOnScreen(false);
70     }
71 } // Конец класса EraseImageDialogFragment
```

7.10. Резюме

В этой главе мы создали приложение Doodlz, которое позволяет пользователю рисовать на экране одним или несколькими пальцами. В приложении реализована функция стирания посредством встрихивания устройства: класс `Android SensorManager` был использован для регистрации слушателя `SensorEventListener`, реагирующего на события акселерометра (также вы узнали о том, что Android поддерживает много других видов датчиков).

Мы создали субклассы `DialogFragment`, отображающие пользовательские представления в окнах `AlertDialog`, и переопределили методы `onAttach` и `onDetach` жизненного цикла фрагмента, которые вызываются при присоединении или отсоединении фрагментов от родительской активности.

Вы узнали, как связать объект `Canvas` с объектом `Bitmap`, чтобы затем использовать `Canvas` для рисования на `Bitmap`. Был рассмотрен принцип обработки многоточечных касаний, чтобы пользователь мог рисовать несколькими пальцами одновременно. Информация по каждому пальцу хранится в объекте `Path`. Обработка событий касания основана на переопределении метода `onTouchEvent` класса `View`, получающего объект `MotionEvent` с типом события и идентификатором пальца, сгенерированного событием. По идентификаторам мы различали пальцы, чтобы добавить информацию в соответствующие объекты `Path`.

В новом полноэкранном режиме погружения, появившемся в Android 4.4, приложение может распоряжаться всем экранным пространством, но при этом пользователь может при необходимости вызвать системные панели и панель действий. Чтобы переключиться в режим погружения, мы воспользовались объектом `GestureDetector` для проверки одиночных касаний.

Объект `ContentResolver` и метод `MediaStore.Images.Media.insertImage` использовались для сохранения изображений в галерее устройства. Наконец, при помощи новой инфраструктуры печати Android 4.4 пользователь может напечатать свой

рисунок. Для вывода объектов `Bitmap` использовался класс `PrintHelper` из Android Support Library. Класс `PrintHelper` отображает интерфейс для выбора принтера или сохранения изображения в документе PDF.

В главе 8 будет построено приложение Address Book, работающее с базой данных и предоставляющее простой и быстрый доступ к списку контактов, с возможностью добавления, удаления и редактирования контактов. Вы научитесь динамически переключать фрагменты в графическом интерфейсе; кроме того, в этом приложении снова будут определены раздельные макеты, оптимизирующие использование экранного пространства на телефонах и планшетах.

8

Приложение Address Book

ListFragment и FragmentTransaction, стек возврата, многопоточность и AsyncTask, CursorAdapter, SQLite и стили GUI

В этой главе...

- Использование ListFragment для отображения и управления списками ListView
- Использование транзакций и стека возврата для динамического присоединения и отсоединения фрагментов
- Создание и открытие баз данных SQLite с помощью объекта SQLiteOpenHelper, вставка, удаление и выборка данных в базах данных SQLite с использованием объекта SQLiteDatabase
- Использование класса SimpleCursorAdapter для связывания результатов запроса базы данных с элементами списка ListView
- Работа с результатами запроса базы данных с помощью класса Cursor
- Использование многопоточности и AsyncTask для выполнения операций с базой данных за пределами потока GUI и обеспечения быстрого отклика приложения
- Определение стилей с часто используемыми значениями и атрибутами GUI, применяемыми к разным компонентам графического интерфейса

8.1. Введение

Приложение Address Book (рис. 8.1) обеспечивает удобный доступ к информации контактов, хранящейся в базе данных SQLite на устройстве. Пользователь прокручивает список контактов, отсортированный в алфавитном порядке. Чтобы просмотреть подробные сведения о контакте, пользователь касается его имени.

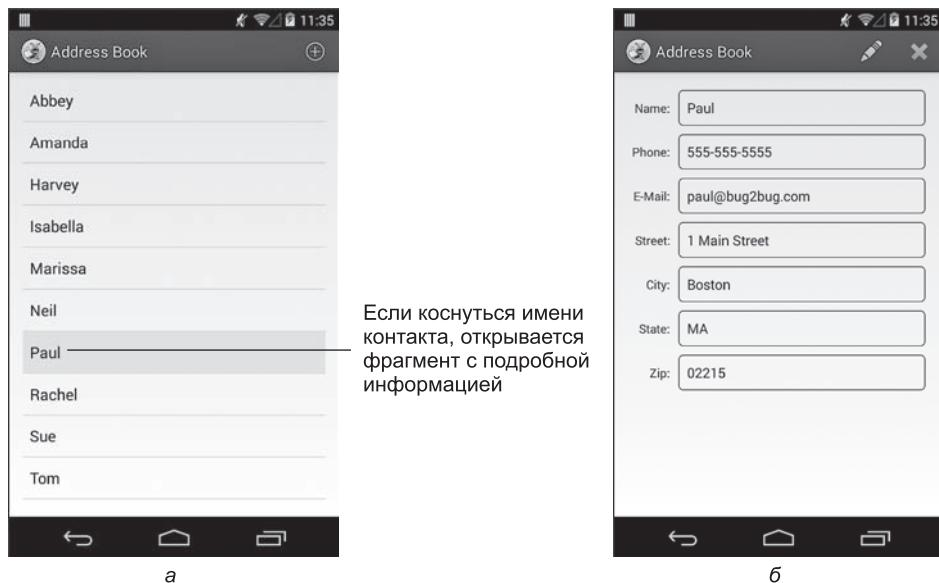


Рис. 8.1. Список контактов и подробная информация о выбранном контакте:
а — список контактов с выделенным контактом; б — подробная информация о выделенном контакте

Если нажать кнопку редактирования (✎) во время просмотра подробных сведений, открывается фрагмент с заранее заполненными компонентами EditText для редактирования данных контакта (рис. 8.2). Кнопка удаления (☒) открывает фрагмент с предложением подтвердить удаление контакта (рис. 8.3).

Если во время просмотра списка контактов нажать кнопку добавления (+), открывается фрагмент с компонентами EditText для ввода данных нового контакта (рис. 8.4). Во время редактирования существующего или добавления нового контакта кнопка Save Contact сохраняет введенные данные. На рис. 8.5 показано приложение на планшете в альбомной ориентации. На планшетах список контактов всегда отображается в левой части приложения.

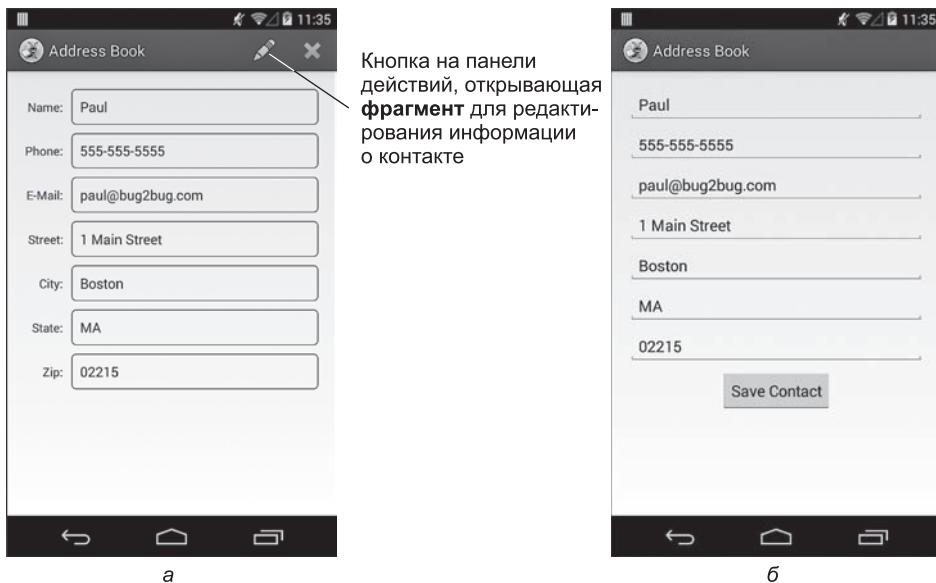


Рис. 8.2. Редактирование данных контакта: а — кнопка редактирования открывает фрагмент для изменения текущего контакта; б — фрагмент для редактирования контакта

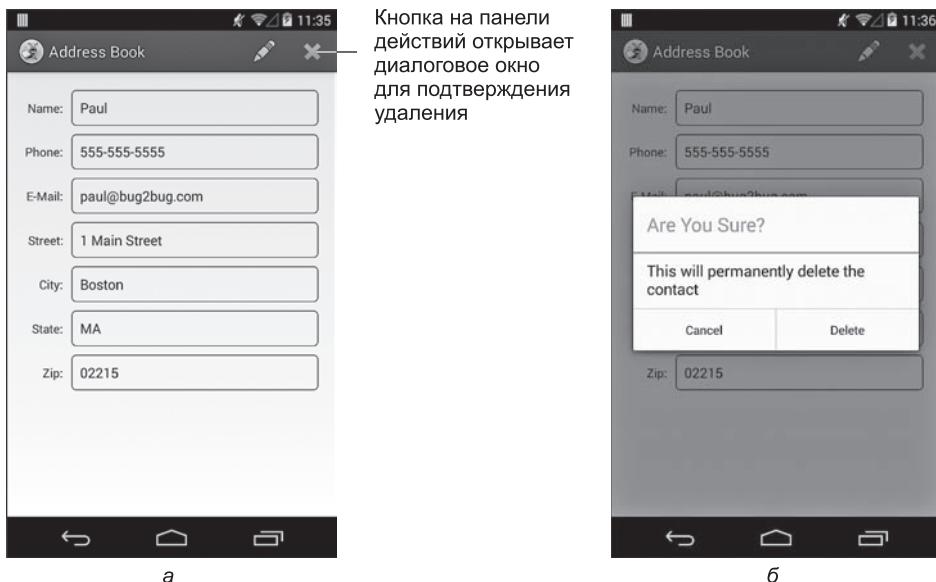


Рис. 8.3. Удаление контакта из базы данных: а — кнопка удаления открывает фрагмент для удаления текущего контакта; б — диалоговое окно для подтверждения удаления

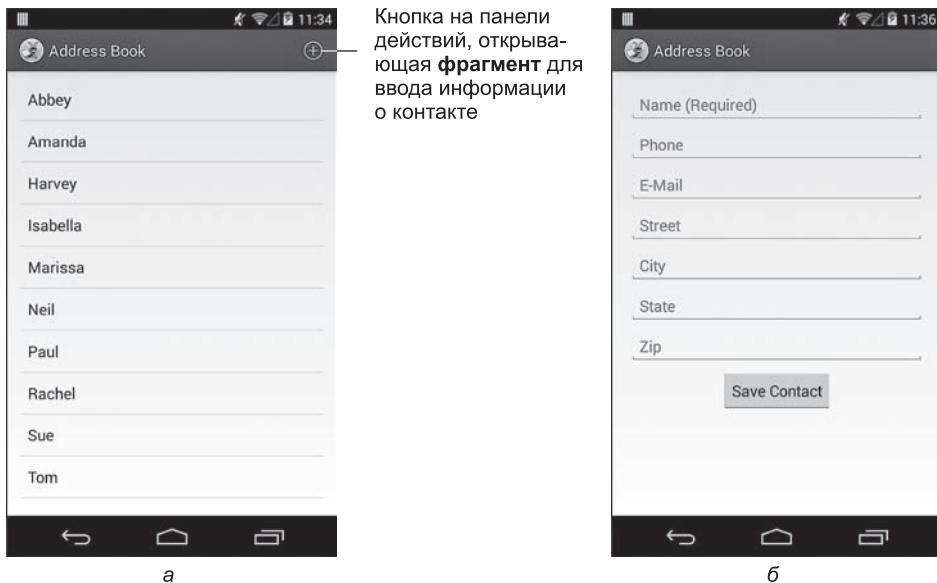


Рис. 8.4. Добавление контакта в базу данных:
а — кнопка добавления открывает фрагмент для создания нового контакта;
б — фрагмент для добавления контакта

8.2. Тестирование приложения Address Book

Открытие и выполнение приложения

Откройте среду Eclipse и импортируйте проект Address Book. Выполните следующие действия.

1. Выполните команду **File > Import...** для открытия диалогового окна Import.
2. В диалоговом окне Import раскройте узел **General** и выберите параметр **Existing Projects into Workspace**. Нажмите **Next>** для перехода к шагу Import Projects. Убедитесь в том, что в окне установлен переключатель **Select root directory**, и нажмите **Browse....** В диалоговом окне **Browse For Folder** выберите папку **AddressBook** в папке примеров книги и нажмите **OK**. Щелкните на кнопке **Finish** для импорта проекта в среду Eclipse. Проект отобразится в окне **Package Explorer**, находящемся в левой части окна Eclipse.
3. Запустите приложение Address Book. В среде Eclipse щелкните правой кнопкой мыши на проекте **AddressBook** в окне **Package Explorer**, затем в появившемся меню выберите команды **Run As > Android Application**.

Добавление контакта

После первого запуска приложения список контактов пуст, а в центре экрана выводится текст No Contacts. Коснитесь кнопки на панели действий. Открывается экран для добавления новой записи в список контактов. После ввода информации о контакте коснитесь кнопки Save Contact для сохранения контакта в базе данных и возврата на главный экран приложения. Если вы передумали добавлять контакт в базу данных, коснитесь кнопки Back устройства для возврата к главному экрану. При желании добавьте дополнительные контакты. На планшетах после добавления новых контактов справа от списка будет отображаться подробная информация (рис. 8.5).

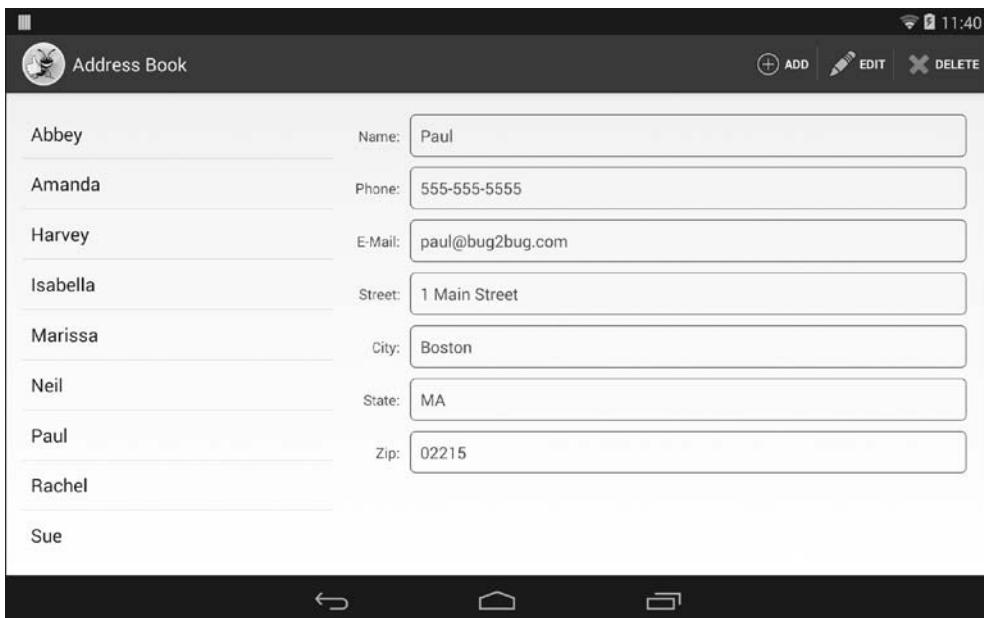


Рис. 8.5. Приложение Address Book в альбомной ориентации на планшете: в альбомной ориентации на телефонах и планшетах кнопки на панели действий отображаются с текстовыми подписями

Просмотр контакта

Для просмотра дополнительных сведений о контакте, добавленном в список, коснитесь его имени. На планшетах подробная информация выводится справа от списка.

Изменение контакта

Во время просмотра сведений о контакте коснитесь кнопки на панели действий. Появится экран с компонентами EditTexts, уже заполненными данными контакта. Внесите необходимые изменения и коснитесь кнопки Save Contact для сохранения обновленной информации в базе данных и возврата на главный экран приложения.

На планшете после редактирования подробная обновленная информация выводится справа от списка.

Удаление контакта

В процессе просмотра подробных сведений о контакте коснитесь кнопки  на панели действий. Открывается диалоговое окно для подтверждения удаления. Если подтвердить удаление, контакт будет удален из базы данных, а в приложении откроется обновленный список контактов.

8.3. Обзор применяемых технологий

В этом разделе вашему вниманию будут представлены новые технологии, применяемые в процессе создания приложения Address Book (в порядке их использования в главе).

8.3.1. Отображение фрагментов с использованием FragmentTransaction

В предыдущих приложениях, в которых использовались фрагменты, мы определяли фрагменты в макетах активностей или, для `DialogFragment`, создавали их вызовом метода `show`. Приложение показывает, как организовать размещение фрагментов в разных активностях. На устройствах с размером экрана, соответствующим экрану телефона, фрагменты отображаются по одному, тогда как на планшетах всегда отображается фрагмент со списком контактов, а фрагменты просмотра, добавления и редактирования контактов отображаются по мере необходимости в правой части экрана. Для динамического отображения фрагментов используются объекты `FragmentManager` и `FragmentTransaction`. Кроме того, мы используем *стек возврата* фрагментов Android — структуру данных, хранящую фрагменты в порядке LIFO (Last-In-First-Out) — для автоматической поддержки кнопки `Back` на системной панели Android и для того, чтобы приложение могло удалять фрагменты в порядке, обратном порядку их добавления.

8.3.2. Передача данных между фрагментом и управляющей активностью

Обмен данными между фрагментами и управляющей активностью (или фрагментами другой активности) рекомендуется осуществлять через управляющую активность — тем самым расширяются возможности повторного использования фрагментов, которые не обращаются друг к другу напрямую. Обычно каждый фрагмент определяет интерфейс методов обратного вызова, реализуемый в управляющей активности. Мы используем этот прием для того, чтобы активность `MainActivity` приложения получала оповещения о выборе контакта, касании кнопок

на панелях действий (, или) , завершении редактирования существующего или добавления нового контакта.

8.3.3. Метод onSaveInstanceState

Метод `onSaveInstanceState` вызывается системой при изменении конфигурации устройства во время выполнения — например, при повороте устройства или выдвижении клавиатуры на устройстве с аппаратной клавиатурой. Этот метод может использоваться для сохранения информации состояния, которое вам хотелось бы восстановить при вызове метода `onCreate` приложения как части изменения конфигурации. Когда приложение просто переводится в фоновый режим (например, для того, чтобы пользователь мог ответить на телефонный звонок, или при запуске другого приложения), компоненты GUI будут автоматически сохранять свое содержимое при переводе приложения в фоновый режим (при условии, что система не уничтожит приложение). Метод `onSaveInstanceState` используется в листинге 8.35.

8.3.4. Определение и применение стилей к компонентам GUI

Пары «атрибут—значение» для часто используемых компонентов GUI можно определить в виде ресурсов стилей XML (раздел 8.4.4). Затем эти стили применяются ко всем компонентам, которые разделяют соответствующие значения (раздел 8.4.7), для чего используется атрибут `style`. Все последующие изменения, внесенные в стиль, будут автоматически применены ко всем компонентам GUI, использующим этот стиль. Мы используем стили для оформления компонентов `TextView`, в которых отображается информация контакта.

8.3.5. Определение фона для компонентов `TextView`

По умолчанию, у компонентов `TextView` отсутствует граница. Чтобы задать границу, укажите значение `Drawable` для атрибута `android:background` компонента `TextView`. В качестве значения `Drawable` может использоваться изображение, хотя для этого приложения можно определить новый тип `Drawable`, используя XML-представление фигуры (раздел 8.4.5). Файл ресурсов для такого объекта `Drawable` определяется в одной или нескольких папках `drawable` приложения — в нашем случае файл `textview_border.xml` определяется в папке `drawable-mdpi`.

8.3.6. Расширение класса `ListFragment` для создания фрагмента, содержащего `ListView`

Если основной задачей фрагмента является отображение прокручиваемого списка элементов, можно расширить класс `ListFragment` (пакет `android.app`, раздел 8.6) — это

делается почти так же, как при расширении `ListActivity`, как было сделано в главе 4. Класс `ListFragment` использует `ListView` в качестве макета по умолчанию. В этом приложении вместо `ArrayAdapter` для отображения результатов запроса к базе данных в `ListView` используется класс `CursorAdapter` (пакет `android.widget`).

8.3.7. Работа с базой данных SQLite

Информация о контактах приложения хранится в базе данных SQLite. Система управления базами данных SQLite (www.sqlite.org) — одна из самых популярных СУБД во всем мире. Классы фрагментов, используемые в приложении, взаимодействуют с SQLite с помощью вспомогательного класса `DatabaseConnector` (раздел 8.9). В этом классе можно использовать вложенный подкласс класса `SQLiteOpenHelper` (пакет `android.database.sqlite`), который упрощает создание базы данных и позволяет воспользоваться объектом `SQLiteDatabase` (пакет `android.database.sqlite`) для работы с содержимым базы данных. Запросы к базе данных выполняются на языке SQL (Structured Query Language), а для управления результатами запроса к базе данных используется класс `Cursor` (пакет `android.database`).

8.3.8. Выполнение операций с базами данных за пределами потока GUI с использованием `AsyncTask`

Долгие операции (и операции, блокирующие выполнение приложения до своего завершения — например, обращения к файлам и базам данных) должны выполняться вне потока GUI. Это помогает приложению быстрее реагировать на действия пользователя и предотвращает появление диалоговых окон ANR (Activity Not Responding). Если нужно получить доступ к результатам операций с базами данных в потоке GUI, используется субкласс `AsyncTask` (пакет `android.os`) для выполнения операций в одном потоке и получения результатов выполнения в потоке GUI. Класс `AsyncTask` берет на себя все подробности создания потоков и выполнения операций с ними, как и передачу результатов из `AsyncTask` в поток GUI.

8.4. Создание графического интерфейса пользователя и файлов ресурсов

В этом разделе будут созданы дополнительные файлы с исходным кодом Java, файлы ресурсов и файлы разметки GUI.

8.4.1. Создание проекта

Начните с создания нового проекта Android. В диалоговом окне `New Android Project` укажите следующие значения, затем нажмите кнопку `Finish`.

- Application Name: Address Book
- Project Name: AddressBook
- Package Name: com.deitel.addressbook
- Minimum Required SDK: API18: Android 4.3
- Target SDK: API19: Android 4.4
- Compile With: API19: Android 4.4
- Theme: Holo Light with Dark Action Bar

На втором шаге диалогового окна New Android Project (New Android Application) оставьте значения по умолчанию и нажмите кнопку **Next >**. На шаге Configure Launcher Icon выберите значок приложения и нажмите кнопку **Next >**. На шаге Create Activity выберите шаблон **Blank Activity** и нажмите кнопку **Next >**. На шаге Blank Activity оставьте значения по умолчанию и нажмите **Finish**, чтобы создать проект. В макетном редакторе выберите в раскрывающемся списке тип экрана **Nexus 4** и удалите компонент **TextView** с текстом «Hello world!».

8.4.2. Создание классов приложения

Приложение состоит из пяти классов.

- Класс **MainActivity** (раздел 8.5) управляет фрагментами приложения и координирует взаимодействия между ними.
- Класс **ContactListFragment** (раздел 8.6) является субклассом **ListFragment**; он выводит имена контактов и предоставляет команду меню для добавления нового контакта.
- Класс **AddEditFragment** (раздел 8.7) является субклассом **Fragment**; он предоставляет графический интерфейс для добавления новых и редактирования существующих контактов.
- Класс **DetailsFragment** (раздел 8.8) является субклассом **Fragment**; он выводит данные одного контакта и предоставляет команды меню для редактирования и удаления этих данных.
- Класс **DatabaseConnector** (раздел 8.9) является субклассом **Object**; он управляет взаимодействиями приложения с базой данных **SQLite**.

Класс **MainActivity** создается средой разработки при создании проекта. Как и в предыдущих проектах, другие классы необходимо добавить в пакет **com.deitel.addressbook** проекта в папке **src**. Для каждого класса щелкните правой кнопкой мыши на пакете и выберите команду **New ▶ Class**, после чего укажите имя класса и суперкласса.

8.4.3. Файл strings.xml

В табл. 8.1 приведены имена строковых ресурсов приложения и соответствующие значения. Сделайте двойной щелчок на файле strings.xml в папке res/values, чтобы вызвать редактор для создания строковых ресурсов.

Таблица 8.1. Строковые ресурсы, используемые в Address Book

Имя ресурса	Значение
no_contacts	No Contacts
menuitem_add	Add
menuitem_edit	Edit
menuitem_delete	Delete
button_save_contact	Save Contact
hint_name	Name (Required)
hint_email	E-Mail
hint_phone	Phone
hint_street	Street
hint_city	City
hint_state	State
hint_zip	Zip
label_name	Name:
label_email	E-Mail:
label_phone	Phone:
label_street	Street:
label_city	City:
label_state	State:
label_zip	Zip:
confirm_title	Are You Sure?
confirm_message	This will permanently delete the contact
ok	OK
error_message	You must enter a contact name
button_cancel	Cancel
button_delete	Delete

8.4.4. Файл styles.xml

В этом разделе определяются стили компонентов `TextView` фрагмента `DetailsFragment`, в которых отображается информация о контактах (раздел 8.4.7). Ресурсы стилей, как и другие ресурсы, размещаются в папке `res/values`. При создании проекта среда разработки создает файл `styles.xml` с заранее определенными стилями. Для каждого созданного вами стиля указывается имя, используемое для применения этого стиля к компонентам GUI, и один или несколько элементов, определяющих значения применяемых свойств. Чтобы создать новые стили, выполните следующие действия.

1. Откройте файл `styles.xml` из папки `res/values`. Убедитесь в том, что в нижней части окна редактора выбрана вкладка `Resources`.
2. Щелкните на кнопке `Add...`, выберите вариант `Style/Theme` и нажмите `OK`, чтобы создать новый стиль.
3. Введите в поле `Name` имя стиля `ContactLabelTextview` и сохраните файл.
4. Убедитесь в том, что в окне выделен стиль `ContactLabelTextview`. Щелкните на кнопке `Add...` и нажмите `OK`, чтобы добавить новый элемент в стиль. Задайте атрибуты `Name` и `Value` нового элемента и сохраните файл. Повторите этот шаг для каждого имени и значения в табл. 8.2.

Таблица 8.2. Атрибуты стиля `ContactLabelTextview`

Name	Value
<code>android:layout_width</code>	<code>wrap_content</code>
<code>android:layout_height</code>	<code>wrap_content</code>
<code>android:layout_gravity</code>	<code>right center_vertical</code>

5. Повторите шаги 2 и 3 для создания стиля с именем `ContactTextview` — когда вы щелкаете на кнопке `Add...`, должен быть установлен режим `Create a new element at the top level in Resources`. Повторите шаг 4 для каждого имени и значения в табл. 8.3. Завершив ввод данных, сохраните и закройте файл `styles.xml`.

Таблица 8.3. Атрибуты стиля `ContactTextview`

Name	Value
<code>android:layout_width</code>	<code>wrap_content</code>
<code>android:layout_height</code>	<code>wrap_content</code>
<code>android:layout_gravity</code>	<code>fill_horizontal</code>
<code>android:textSize</code>	<code>16sp</code>
<code>android:background</code>	<code>@drawable/textview_border</code>

8.4.5. Файл `textview_border.xml`

Стиль `ContactTextView`, созданный в предыдущем разделе, определяет внешний вид компонентов `TextView`, используемых для отображения подробной информации о контактах. Мы задали объект `Drawable` (графическое изображение) с именем `@drawable/textview_border` как значение атрибута `android:background` attribute компонента `TextView`. В этом разделе мы определим этот объект `Drawable` в папке `res/drawable-mdpi` приложения. Если объект `Drawable` определяется только в одной из папок `drawable` проекта, Android будет использовать его для всех размеров устройств и разрешения. Чтобы определить объект `Drawable`, выполните следующие действия.

1. Щелкните правой кнопкой мыши на папке `res/drawable-mdpi` и выберите команду `New > Android XML File`.
2. Задайте свойству `File` значение `textview_border.xml`, выберите корневым элементом `shape` и нажмите `Finish`.
3. На момент написания книги в среде разработки не было специального редактора для создания объектов `Drawable`, поэтому введите разметку XML из листинга 8.1 в файл.

Листинг 8.1. XML-представление объекта `Drawable`, используемого для создания рамок компонентов `TextView`

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <shape xmlns:android="http://schemas.android.com/apk/res/android"
3   android:shape="rectangle" >
4     <corners android:radius="5dp"/>
5     <stroke android:width="1dp" android:color="#555"/>
6     <padding android:top="10dp" android:left="10dp" android:bottom="10dp"
7       android:right="10dp"/>
8 </shape>
```

Атрибут `android:shape` элемента `shape` (строка 3) может иметь значение "rectangle" (использованное в этом примере), "oval", "line" или "ring". Элемент `corners` (строка 4) определяет радиус закруглений углов прямоугольного контура. Элемент `stroke` (строка 5) определяет толщину и цвет контура прямоугольника. Элемент `padding` (строки 6–7) определяет интервалы вокруг содержимого элемента, к которому применяется объект `Drawable`. Отступы со всех четырех сторон должны задаваться отдельно. Полную информацию об определении фигур можно просмотреть по адресу

<http://developer.android.com/guide/topics/resources/drawable-resource.html#Shape>

8.4.6. Файл макета `MainActivity`: `activity_main.xml`

для `MainActivity` будут определены два макета — для устройств с размером экрана, соответствующим размеру экрана телефона (папка `res/layout`), и для

планшетных устройств в папке `res/layout-large`. Папку `layout-large` необходимо добавить в проект.

Макет для телефона: `activity_main.xml` в `res/layout`

Откройте файл `activity_main.xml` в папке `res/layout`. Щелкните правой кнопкой мыши на узле `RelativeLayout` в окне `Outline`, выберите команду `Change Layout...` и переключитесь на макет `FrameLayout`. Задайте идентификатору `FrameLayout` значение `@+id/rightPaneContainer`. Этот компонент `FrameLayout` будет использоваться для отображения фрагментов приложения на телефонах.

Макет для планшета: `activity_main.xml` в `res/layout-large`

Создайте файл `activity_main.xml` в папке `res/layout-large`. Этот макет будет состоять из горизонтального компонента `LinearLayout`, содержащего фрагмент `ContactListFragment` и пустой компонент `FrameLayout`. Добавьте в макет `ContactListFragment`, затем добавьте `FrameLayout` так, как описано в разделе 5.4.9. Задайте следующие значения свойств.

- ❑ `LinearLayout`: задайте свойству `Weight Sum` значение 3 — это нужно для распределения горизонтального пространства между `ContactListFragment` и `FrameLayout`.
- ❑ `Фрагмент`: задайте свойству `Id` значение `@+id/contactListFragment`, свойству `Width` — значение 0, свойству `Height` — значение `match_parent`, свойству `Weight` — значение 1 и свойству `Right` — значение `@dimen/activity_horizontal_margin`.
- ❑ `FrameLayout`: задайте свойству `Id` значение `@+id/rightPaneContainer`, свойству `Width` — значение 0, свойству `Height` — значение `match_parent` и свойству `Weight` — значение 2.

Задавая свойству `Weight Sum` компонента `LinearLayout` значение 3, а свойствам `Weight` компонентов `ContactListFragment` и `FrameLayout` — значения 1 и 2 соответственно, мы указываем, что компонент `ContactListFragment` должен занимать 1/3 ширины `LinearLayout`, а компонент `FrameLayout` — оставшиеся 2/3.

8.4.7. Файл макета `DetailsFragment`: `fragment_details.xml`

Когда пользователь касается контакта в `MainActivity`, приложение отображает фрагмент `DetailsFragment` (рис. 8.6). Макет этого фрагмента (`fragment_details.xml`) состоит из компонента `ScrollView` с вертикальной панелью `GridLayout`, состоящей из двух столбцов `TextView`. Компонент `ScrollView` может содержать другие представления с возможностью прокрутки информации, не помещающейся на экране. Мы используем `ScrollView`, чтобы пользователь мог прокрутить информацию о контакте, если на устройстве не хватает места для отображения всех компонентов `TextView` на рис. 8.6. Чтобы создать файл `fragment_details.xml`, выполните действия из раздела 5.4.8, но выберите корневым элементом (`Root Element`) компонент `ScrollView`. После создания файла задайте свойству `Id` компонента `ScrollView` значение `@+id/detailsScrollView` и добавьте на `ScrollView` компонент `GridLayout`.

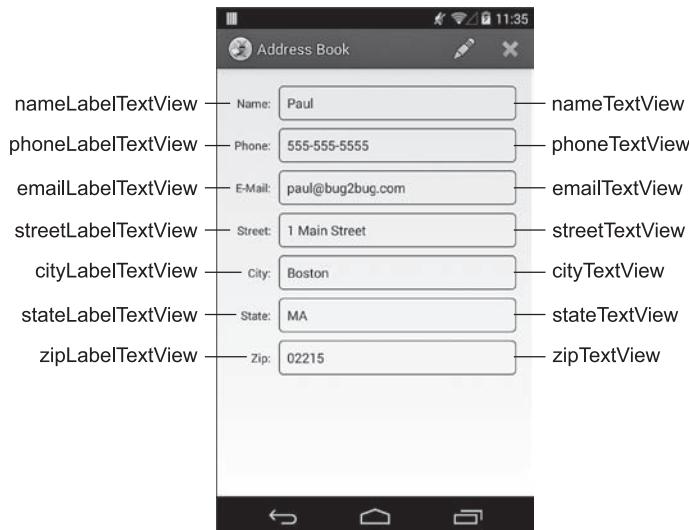


Рис. 8.6. Компоненты GUI DetailsFragment

Настройка компонента GridLayout

Для компонента `GridLayout` задайте свойству `Width` значение `match_parent`, свойству `Height` — значение `wrap_content`, свойству `Column Count` — значение 2 и свойству `Use Default Margins` — значение `true`. Значение `Height` позволяет родительскому компоненту `ScrollView` определить фактическую высоту `GridLayout` и решить, нужно ли предоставить средства прокрутки.

Добавьте компонент `TextView` на панель `GridLayout`, как показано на рис. 8.6.

Настройки компонентов `TextView` в левом столбце

Задайте свойство `Id` каждого компонента `TextView` в левом столбце в соответствии с рис. 8.6. Кроме того, задайте следующие свойства.

- `Row` — значение от 0 до 6 (в зависимости от строки).
- `Column` — значение 0.
- `Text` — соответствующий строковый ресурс из файла `strings.xml`.
- `Style` (из категории `View`) — значение `@style/ContactLabelTextview` (для определения стилевых ресурсов используется синтаксис `@style/имяСтиля`).

Настройки компонентов `TextView` в правом столбце

Задайте свойство `Id` каждого компонента `TextView` в правом столбце в соответствии с рис. 8.6. Кроме того, задайте следующие свойства.

- `Row` — значение от 0 до 6 (в зависимости от строки).
- `Column` — значение 1.
- `Style` (из категории `View`) — значение `@style/ContactTextview`.

8.4.8. Макет AddEditFragment: fragment_add_edit.xml

Когда пользователь касается кнопок или на панели действий, активность `MainActivity` отображает фрагмент `AddEditFragment` (рис. 8.7) с макетом (`fragment_add_edit.xml`), использующим компонент `ScrollView` с одностолбцовой вертикальной панелью `GridLayout`. Обязательно задайте свойству `Id` компонента `ScrollView` значение `@+id/addEditScrollView`. Если на экране отображается фрагмент `AddEditFragment` для добавления нового контакта, в полях `EditText` вместо информации будут отображаться подсказки (см. рис. 8.4). В противном случае в них отображаются данные контакта, переданного фрагменту `AddEditFragment` активностью `MainActivity`. Для каждого компонента `EditText` задаются свойства `Input Type` и `IME Options`. Для устройств, отображающих виртуальную клавиатуру, свойство `Input Type` определяет тип клавиатуры, которая должна отображаться при касании соответствующего компонента `EditText`. Это позволяет приспособить клавиатуру для конкретного типа данных, которые должны вводиться в текстовом поле. Свойство `IME Options` используется для отображения кнопки `Next` на виртуальной клавиатуре для компонентов `nameEditText`, `emailEditText`, `phoneEditText`, `streetEditText`, `cityEditText` и `stateEditText`. Если один из этих компонентов имеет фокус ввода, эта кнопка передает фокус следующему компоненту `EditText`. Если фокус принадлежит полю `zipEditText`, виртуальную клавиатуру можно закрыть кнопкой `Done`.

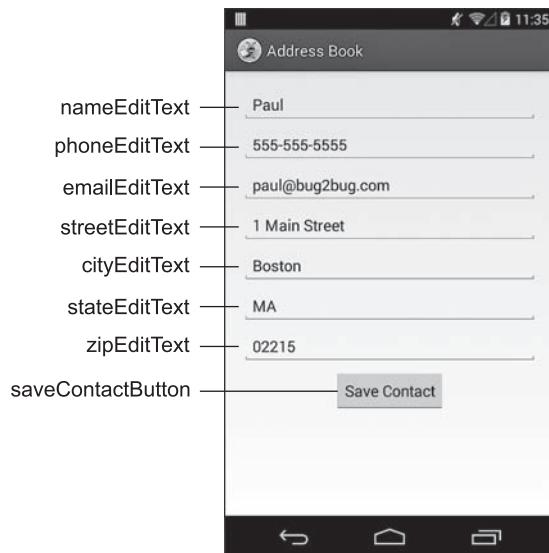


Рис. 8.7. Компоненты GUI `AddEditFragment` со значениями свойств `id`. Корневым компонентом GUI является компонент `ScrollView`, содержащий вертикальную панель `GridLayout`

Свойства GridLayout

Для компонента `GridLayout` задайте свойству `Width` значение `match_parent`, свойству `Height` — значение `wrap_content`, свойству `Column Count` — значение `1` и свойству `Use Default Margins` — значение `true`. Добавьте компоненты, показанные на рис. 8.7.

Свойства EditText

Для каждого компонента `EditText` задайте свойство `Id` в соответствии с рис. 8.7, а также следующие свойства.

- `Width` — значение `match_parent`.
- `Height` — значение `wrap_content`.
- `Hint` — соответствующий строковый ресурс из файла `strings.xml`.
- `IME Options` — значение `actionNext` для всех компонентов, кроме `zipEditText` (для этого компонента используется значение `actionDone`).
- `Style` (из категории `View`) — значение `@style/ContactLabelTextView` (для определения стилевых ресурсов используется синтаксис `@style/имяСтиля`).

Задайте свойство `Input Type` компонентов `EditText` для отображения соответствующих клавиатур.

- `nameEditText: textPersonName | textCapWords` — для ввода имен, каждое слово начинается с буквы верхнего регистра.
- `phoneEditText: phone` — для ввода телефонных номеров.
- `emailEditText: textEmailAddress` — для ввода адресов электронной почты.
- `streetEditText: textPostalAddress | textCapWords` — для ввода адресов, каждое слово начинается с буквы верхнего регистра.
- `cityEditText: textPostalAddress | textCapWords`.
- `stateEditText: textPostalAddress | textCapCharacters` — гарантирует, что сокращенные названия штатов отображаются в верхнем регистре.
- `zipEditText: number` — для ввода чисел.

8.4.9. Определение меню фрагментов

Теперь мы воспользуемся приемами, описанными в разделе 7.3.4, для создания двух файлов с ресурсами меню в папке приложения `res/menu`:

- `fragment_contact_list_menu.xml` определяет команду меню для добавления контакта.
- `fragment_details_menu.xml` определяет команды меню для редактирования и удаления контакта.

Когда `ContactListFragment` и `DetailsFragment` отображаются на планшете одновременно, видны все команды меню.

В табл. 8.4–8.5 приведены настройки команд меню в двух файлах ресурсов. Значения `Order in category` определяют порядок отображения команд меню на панели действий. Свойству `Icon` каждой команды меню задается стандартный значок Android. Полный список стандартных значков содержится в папке `platforms` Android SDK (в папке `data/res/drawable-hdpi` для каждой версии платформы). Чтобы использовать ссылки на эти значки в меню или макетах, снабдите их префиксом `@android:drawable/имя_значка`.

Таблица 8.4. Команда меню для `fragment_contact_list_menu.xml`

Name	Value
Id	<code>@+id/action_add</code>
Order in category	0
Title	<code>@string/menuitem_add</code>
Icon	<code>@android:drawable/ic_menu_add</code>
Show as action	<code>ifRoom withText</code>

Таблица 8.5. Команда меню для `fragment_details_menu.xml`

Name	Value
Команда Edit	
Id	<code>@+id/action_edit</code>
Order in category	1
Title	<code>@string/menuitem_edit</code>
Icon	<code>@android:drawable/ic_menu_edit</code>
Show as action	<code>ifRoom withText</code>
Команда Delete	
Id	<code>@+id/action_delete</code>
Order in category	2
Title	<code>@string/menuitem_delete</code>
Icon	<code>@android:drawable/ic_delete</code>
Show as action	<code>ifRoom withText</code>

8.5. Класс MainActivity

Класс `MainActivity` (листинги 8.2–8.11) управляет фрагментами приложения и координирует взаимодействия между ними. На телефонах `MainActivity` в любой момент времени отображает только один фрагмент начиная с `ContactListFragment`. На планшетах `MainActivity` всегда отображает `ContactListFragment` слева и в зависимости от контекста — либо `DetailsFragment`, либо `AddEditFragment` в правых 2/3 экрана.

Команда package, команды import и поля класса MainActivity

Класс `MainActivity` (листинг 8.2) использует класс `FragmentTransaction` (импортируемый в строке 6) для добавления и удаления фрагментов приложения. `MainActivity` реализует следующие три интерфейса.

- `ContactListFragment.ContactListFragmentListener` содержит методы обратного вызова, при помощи которых `ContactListFragment` сообщает `MainActivity`, что пользователь выбрал контакт в списке или добавил новый контакт.
- `DetailsFragment.DetailsFragmentListener` содержит методы обратного вызова, при помощи которых `DetailsFragment` сообщает `MainActivity`, что пользователь удаляет контакт или хочет отредактировать существующий контакт.
- `AddEditFragment.AddEditFragmentListener` содержит методы обратного вызова, при помощи которых `AddEditFragment` сообщает `MainActivity`, что пользователь завершил добавление нового контакта или редактирование существующего контакта.

Константа `ROW_ID` (строка 15) используется как ключ в паре «ключ—значение», передаваемой между активностью `MainActivity` и ее фрагментами. Переменная экземпляра `contactListFragment` (строка 17) приказывает `ContactListFragment` обновить список контактов после добавления или удаления контакта.

Листинг 8.2. Команда package, команды import и поля класса MainActivity

```

1 // MainActivity.java
2 // Управление фрагментами приложения Address Book
3 package com.deitel.addressbook;
4
5 import android.app.Activity;
6 import android.app.FragmentTransaction;
7 import android.os.Bundle;
8
9 public class MainActivity extends Activity
10    implements ContactListFragment.ContactListFragmentListener,
11           DetailsFragment.DetailsFragmentListener,
12           AddEditFragment.AddEditFragmentListener
13 {
14    // Ключи идентификатора строки в объекте Bundle, передаваемом фрагменту
15    public static final String ROW_ID = "row_id";
16
17    ContactListFragment contactListFragment; // Вывод списка контактов
18

```

Переопределенный метод onCreate класса MainActivity

Метод `onCreate` (листинг 8.3) заполняет графический интерфейс `MainActivity` и, если приложения выполняется на телефоне, — отображает `ContactListFragment`. Как будет показано в разделе 8.6, фрагмент можно настроить для сохранения изменения конфигурации (например, при повороте устройства). Если активность восстанавливается после завершения или создается повторно после изменения конфигурации, значение `savedInstanceState` будет отлично от `null`. В этом случае метод просто возвращает управление (строка 28), потому что `ContactListFragment` уже существует — на телефоне он будет сохранен, а на планшете он является частью макета `MainActivity`, заполняемого в строке 24.

Листинг 8.3. Переопределенный метод onCreate класса MainActivity

```
19 // Отображает ContactListFragment при первой загрузке MainActivity
20 @Override
21 protected void onCreate(Bundle savedInstanceState)
22 {
23     super.onCreate(savedInstanceState);
24     setContentView(R.layout.activity_main);
25     // Если активность восстанавливается, просто вернуть управление;
26     // заново создавать GUI не нужно.
27     if (savedInstanceState != null)
28         return;
29
30     // Проверить, содержит ли макет fragmentContainer (макет для телефона);
31     // ContactListFragment отображается всегда.
32     if (findViewById(R.id.fragmentContainer) != null)
33     {
34         // Создание ContactListFragment
35         contactListFragment = new ContactListFragment();
36
37         // Добавление фрагмента в FrameLayout
38         FragmentTransaction transaction =
39             getFragmentManager().beginTransaction();
40         transaction.add(R.id.fragmentContainer, contactListFragment);
41         transaction.commit(); // Приводит к отображению ContactListFragment
42     }
43 }
44 }
```

Если `R.id.fragmentContainer` существует в макете `MainActivity` (строка 32), значит приложение выполняется на телефоне. В этом случае строка 35 создает `ContactListFragment`, после чего в строках 38–41 объект `FragmentTransaction` используется для добавления `ContactListFragment` в пользовательский интерфейс. В строках 38–39 вызывается метод `beginTransaction` объекта `FragmentManager` для получения объекта `FragmentTransaction`. Затем в строке 40 метод `add` класса `FragmentTransaction` указывает, что при завершении `FragmentTransaction` фрагмент `ContactListFragment` должен быть присоединен к представлению с идентификатором, передаваемым в первом аргументе. Наконец, строка 41 использует метод `commit`

класса `FragmentTransaction` для завершения транзакции и отображения фрагмента `ContactListFragment`.

Переопределенный метод `onResume` класса `MainActivity`

Метод `onResume` (листинг 8.4) проверяет `contactListFragment` на `null` — если проверка дает положительный результат, значит приложение выполняется на планшете, поэтому в строках 55–57 объект `FragmentManager` используется для получения ссылки на существующий фрагмент `ContactListFragment` в макете `MainActivity`.

Листинг 8.4. Переопределенный метод `onResume` класса `MainActivity`

```

45    // Вызывается при продолжении выполнения MainActivity
46    @Override
47    protected void onResume()
48    {
49        super.onResume();
50
51        // Если значение contactListFragment равно null, значит, активность
52        // работает на планшете; получить ссылку от FragmentManager
53        if (contactListFragment == null)
54        {
55            contactListFragment =
56                (ContactListFragment) getFragmentManager().findFragmentById(
57                    R.id.contactListFragment);
58        }
59    }
60

```

Метод `onContactSelected` класса `MainActivity`

Метод `onContactSelected` (листинг 8.5) из интерфейса `ContactListFragment.ContactListFragmentListener` вызывается объектом `ContactListFragment` для оповещения `MainActivity` о том, что пользователь выбрал контакт для отображения. Если приложение работает на телефоне (строка 65), то в строке 66 вызывается метод `displayContact` (листинг 8.6), который заменяет `ContactListFragment` в объекте `fragmentContainer` (см. определение в разделе 8.4.6) фрагментом `DetailsFragment`, содержащим информацию о контакте. На планшетах в строке 69 вызывается метод `popBackStack` класса `FragmentManager` для извлечения верхнего фрагмента из стека возврата, после чего строка 70 вызывает метод `displayContact`, заменяющий содержимое `rightPaneContainer` (см. раздел 8.4.6) фрагментом `DetailsFragment` с подробной информацией о контакте.

Листинг 8.5. Метод `onContactSelected` класса `MainActivity`

```

61    // Отображение DetailsFragment для выбранного контакта
62    @Override
63    public void onContactSelected(long rowID)
64    {
65        if (findViewById(R.id.fragmentContainer) != null) // Телефон
66            displayContact(rowID, R.id.fragmentContainer);

```

```

67     else // Планшет
68     {
69         getSupportFragmentManager().popBackStack(); // Снять со стека возврата
70         displayContact(rowID, R.id.rightPaneContainer);
71     }
72 }
73

```

Метод displayContact класса MainActivity

Метод `displayContact` (листинг 8.6) создает фрагмент `DetailsFragment` с информацией выбранного контакта и использует объект `FragmentTransaction` для присоединения его к графическому интерфейсу. Чтобы передать аргументы фрагменту, разместите их в объекте `Bundle`, содержащем пары «ключ—значение», — мы используем эту возможность для передачи идентификатора `rowID` контакта, чтобы фрагмент `DetailsFragment` знал, какой контакт следует загрузить из базы данных. Объект `Bundle` создается в строке 80. Стока 81 вызывает его метод `putLong` для сохранения пары «ключ—значение» с ключом `ROW_ID (String)` и значением `rowID (long)`. Стока 82 передает объект `Bundle` методу `setArguments` фрагмента — далее фрагмент может извлечь информацию из `Bundle` (см. раздел 8.8). В строках 85–86 мы получаем объект `FragmentTransaction`, после чего строка 87 вызывает метод `replace` класса `FragmentTransaction`; тем самым мы указываем, что при завершении `FragmentTransaction` фрагмент `DetailsFragment` должен заменить содержимое представления с идентификатором, переданным в первом аргументе. Стока 88 вызывает метод `addToBackStack` класса `FragmentTransaction` для включения `DetailsFragment` в стек возврата. Это делается для того, чтобы при нажатии кнопки `Back` фрагмент извлекался из стека возврата, а активность `MainActivity` могла извлечь фрагмент из стека на программном уровне.

Листинг 8.6. Метод displayContact класса MainActivity

```

74     // Отображение информации о контакте
75     private void displayContact(long rowID, int viewID)
76     {
77         DetailsFragment detailsFragment = new DetailsFragment();
78
79         // Передача rowID в аргументе DetailsFragment
80         Bundle arguments = new Bundle();
81         arguments.putLong(ROW_ID, rowID);
82         detailsFragment.setArguments(arguments);
83
84         // Использование FragmentTransaction для отображения DetailsFragment
85         FragmentTransaction transaction =
86             getSupportFragmentManager().beginTransaction();
87         transaction.replace(viewID, detailsFragment);
88         transaction.addToBackStack(null);
89         transaction.commit(); // Приводит к отображению DetailsFragment
90     }
91

```

Метод onAddContact класса MainActivity

Метод `onAddContact` (листинг 8.7) из интерфейса `ContactListFragment.ContactListFragmentListener` вызывается объектом `ContactListFragment` для оповещения `MainActivity` о том, что пользователь выбрал команду добавления нового контакта. Если макет содержит `fragmentContainer`, то в строке 97 вызывается метод `displayAddEditFragment` (листинг 8.8), отображающий `AddEditFragment` в `fragmentContainer`. В противном случае в строке 99 вызывается метод `displayAddEditFragment` для отображения фрагмента в `rightPaneContainer`. Во втором аргументе передается объект `Bundle`. Передача `null` означает, что добавляется новый контакт.

Листинг 8.7. Метод onAddContact класса MainActivity

```

92     // Отображение фрагмента AddEditFragment для добавления контакта
93     @Override
94     public void onAddContact()
95     {
96         if (findViewById(R.id.fragmentContainer) != null) // Телефон
97             displayAddEditFragment(R.id.fragmentContainer, null);
98         else // Планшет
99             displayAddEditFragment(R.id.rightPaneContainer, null);
100    }
101

```

Метод displayAddEditFragment класса MainActivity

Метод `displayAddEditFragment` (листинг 8.8) получает идентификатор ресурса представления, который сообщает, куда следует прикрепить `AddEditFragment`, а также объект `Bundle` с парами «ключ—значение». Если второй аргумент равен `null`, добавляется новый контакт; в противном случае объект `Bundle` содержит данные, которые должны отображаться в `AddEditFragment` для редактирования. В строке 105 создается объект `AddEditFragment`. Если аргумент `Bundle` отличен от `null`, он используется в строке 108 для назначения аргументов `Fragment`. Затем в строках 111–115 создается объект `FragmentTransaction`, содержимое представления заменяется ресурсом с заданным идентификатором, фрагмент добавляется в стек возврата, после чего транзакция закрепляется.

Листинг 8.8. Метод displayAddEditFragment класса MainActivity

```

102     // Отображение фрагмента для изменения или добавления контакта
103     private void displayAddEditFragment(int viewID, Bundle arguments)
104     {
105         AddEditFragment addEditFragment = new AddEditFragment();
106
107         if (arguments != null) // Редактирование существующего контакта
108             addEditFragment.setArguments(arguments);
109
110         // Использование FragmentTransaction для отображения AddEditFragment
111         FragmentTransaction transaction =
112             getFragmentManager().beginTransaction();
113         transaction.replace(viewID, addEditFragment);

```

```

114     transaction.addToBackStack(null);
115     transaction.commit(); // Приводит к отображению AddEditFragment
116 }
117

```

Метод onContactDeleted класса MainActivity

Метод `onContactDeleted` (листинг 8.9) из интерфейса `DetailsFragment.DetailsFragmentListener` вызывается объектом `DetailsFragment` для оповещения `MainActivity` об удалении контакта. В этом случае в строке 122 фрагмент `DetailsFragment` извлекается из стека возврата. Если приложение выполняется на планшете, то строка 125 вызывает метод `updateContactList` объекта `contactListFragment` для перезагрузки контактов.

Листинг 8.9. Метод onContactDeleted класса MainActivity

```

118 // Возврат к списку контактов после удаления
119 @Override
120 public void onContactDeleted()
121 {
122     getFragmentManager().popBackStack(); // Извлекает верхний элемент
123                                         // из стека
124     if (findViewById(R.id.fragmentContainer) == null) // Планшет
125         contactListFragment.updateContactList();
126 }
127

```

Метод onEditContact класса MainActivity

Метод `onEditContact` (листинг 8.10) из интерфейса `DetailsFragment.DetailsFragmentListener` вызывается объектом `DetailsFragment` для оповещения `MainActivity` о том, что пользователь коснулся команды меню для редактирования контакта. `DetailsFragment` передает объект `Bundle` с данными контакта, чтобы их можно было вывести в полях `AddEditFragment` для редактирования. Если макет содержит `fragmentContainer`, то в строке 133 вызывается метод `displayAddEditFragment`, отображающий `AddEditFragment` в `fragmentContainer`. В противном случае в строке 135 вызывается метод `displayAddEditFragment` для отображения `AddEditFragment` в `rightPaneContainer`.

Листинг 8.10. Метод onEditContact класса MainActivity

```

128 // Отображение AddEditFragment для изменения существующего контакта
129 @Override
130 public void onEditContact(Bundle arguments)
131 {
132     if (findViewById(R.id.fragmentContainer) != null) // Телефон
133         displayAddEditFragment(R.id.fragmentContainer, arguments);
134     else // Планшет
135         displayAddEditFragment(R.id.rightPaneContainer, arguments);
136 }
137

```

Метод onAddEditCompleted класса MainActivity

Метод `onAddEditCompleted` (листинг 8.11) из интерфейса `AddEditFragment.AddEditFragmentListener` вызывается объектом `AddEditFragment` для оповещения `MainActivity` о том, что пользователь сохраняет новый контакт или изменения в существующем контакте. Стока 142 извлекает `AddEditFragment` из стека возврата. Если приложение выполняется на планшете (строка 144), то строка 146 снова выполняет извлечение из стека для удаления фрагмента `DetailsFragment` (если он есть). Затем строка 147 обновляет список контактов в `ContactListFragment`, а в строке 150 информация нового или обновленного контакта отображается в `rightPaneContainer`.

Листинг 8.11. Метод onAddEditCompleted класса MainActivity

```

138     // Обновление GUI после сохранения нового или измененного контакта
139     @Override
140     public void onAddEditCompleted(long rowID)
141     {
142         getSupportFragmentManager().popBackStack(); // Извлечение из стека
143
144         if (findViewById(R.id.fragmentContainer) == null) // Планшет
145         {
146             getSupportFragmentManager().popBackStack(); // Извлечение из стека
147             contactListFragment.updateContactList(); // Обновление списка
148
149             // На планшете вывести добавленный или измененный контакт
150             displayContact(rowID, R.id.rightPaneContainer);
151         }
152     }
153 }
```

8.6. Класс ContactListFragment

Класс `ContactListFragment` (листинги 8.12–8.21) расширяет `ListFragment` для вывода списка контактов в компоненте `ListView`, а также предоставляет команду меню для добавления нового контакта.

Команда package и команды import класса ContactListFragment

В листинге 8.12 приведены команда `package` и команды `import` класса `ContactListFragment`, с выделением команд импорта новых классов и интерфейсов.

Листинг 8.12. Команды package и import класса ContactListFragment

```

1 // ContactListFragment.java
2 // Вывод списка имен контактов
3 package com.deitel.addressbook;
4
5 import android.app.Activity;
6 import android.app.ListFragment;
7 import android.database.Cursor;
8 import android.os.AsyncTask;
```

```

9 import android.os.Bundle;
10 import android.view.Menu;
11 import android.view.MenuInflater;
12 import android.view.MenuItem;
13 import android.view.View;
14 import android.widget.AdapterView;
15 import android.widget.AdapterView.OnItemClickListener;
16 import android.widget.CursorAdapter;
17 import android.widget.ListView;
18 import android.widget.SimpleCursorAdapter;
19

```

Интерфейс ContactListFragmentListener и переменные экземпляров ContactListFragment

В листинге 8.13 начинается объявление класса `ContactListFragment`. В строках 23–30 объявляется вложенный интерфейс `ContactListFragmentListener` с методами обратного вызова, которые реализуются `MainActivity` для оповещения о выборе пользователем контакта (строка 26) и о том, что пользователь коснулся команды меню, добавляющей новый контакт (строка 29). В строке 32 объявляется переменная экземпляра `listener`, которая будет ссылаться на объект (`MainActivity`), реализующий интерфейс. Переменная экземпляра `contactListView` (строка 34) содержит ссылку на компонент `ContactListFragment`, встроенный в `ListView`; по этой ссылке мы сможем взаимодействовать с ним. Переменная экземпляра `contactAdapter` содержит ссылку на объект `CursorAdapter`, заполняющий компонент `ListView` приложения.

Листинг 8.13. Интерфейс ContactListFragmentListener и переменные экземпляров ContactListFragment

```

20 public class ContactListFragment extends ListFragment
21 {
22     // Методы обратного вызова, реализованные MainActivity
23     public interface ContactListFragmentListener
24     {
25         // Вызывается при выборе контакта пользователем
26         public void onContactSelected(long rowID);
27
28         // Вызывается при добавлении контакта
29         public void onAddContact();
30     }
31
32     private ContactListFragmentListener listener;
33
34     private ListView contactListView; // Компонент ListView для ListActivity
35     private CursorAdapter contactAdapter; // Адаптер ListView
36

```

Переопределенные методы `onAttach` и `onDetach` класса `ContactListFragment`

Класс `ContactListFragment` переопределяет методы `onAttach` и `onDetach` жизненного цикла фрагмента (листинг 8.14) для назначения переменной экземпляра `listener`. В нашем приложении `listener` присваивается ссылка на управляющую активность

(строка 42) при присоединении ContactListFragment и null (строка 50) при отсоединении ContactListFragment.

Листинг 8.14. Переопределенные методы onAttach и onDetach класса ContactListFragment

```

37 // Назначение ContactListFragmentListener при присоединении фрагмента
38 @Override
39 public void onAttach(Activity activity)
40 {
41     super.onAttach(activity);
42     listener = (ContactListFragmentListener) activity;
43 }
44
45 // Удаление ContactListFragmentListener при отсоединении фрагмента
46 @Override
47 public void onDetach()
48 {
49     super.onDetach();
50     listener = null;
51 }
52

```

Переопределенный метод onViewCreated класса ContactListFragment

Так как класс ListFragment уже содержит ListView, нам не нужно заполнять графический интерфейс так, как это делалось во фрагментах предыдущего приложения. Однако у класса ContactListFragment существуют операции, которые должны выполняться после заполнения макета по умолчанию. По этой причине ContactListFragment переопределяет метод onViewCreated жизненного цикла фрагмента (листинг 8.15), который вызывается после onCreateView.

Листинг 8.15. Переопределенный метод onViewCreated класса ContactListFragment

```

53 // Вызывается после создания представления
54 @Override
55 public void onViewCreated(View view, Bundle savedInstanceState)
56 {
57     super.onViewCreated(view, savedInstanceState);
58     setRetainInstance(true); // Сохранение между изменениями конфигурации
59     setHasOptionsMenu(true); // У фрагмента есть команды меню
60
61     // Текст, отображаемый при отсутствии контактов
62     setEmptyText(getResources().getString(R.string.no_contacts));
63
64     // Получение ссылки на ListView и настройка ListView
65     contactListView = getListView();
66     contactListView.setOnItemClickListener(viewContactListener);
67     contactListView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
68
69     // Имя контакта связывается с TextView в макете ListView
70     String[] from = new String[] { "name" };

```

```
71     int[] to = new int[] { android.R.id.text1 };
72     contactAdapter = new SimpleCursorAdapter(getActivity(),
73         android.R.layout.simple_list_item_1, null, from, to, 0);
74     setListAdapter(contactAdapter); // Адаптер, поставляющий данные
75 }
76
```

В строке 58 вызывается метод `setRetainInstance` класса `Fragment` с аргументом `true`; это означает, что экземпляр `ContactListFragment` должен сохраняться при изменениях конфигурации (вместо его повторного создания при повторном создании управляющей активности) — например, при повороте устройства пользователем. Стока 59 сообщает, что у `ContactListFragment` имеются команды меню, которые должны отображаться на панели действий активности (или в раскрывающемся меню). Метод `setEmptyText` класса `ListFragment` определяет текст ("No Contacts"), отображаемый при отсутствии данных в адаптере `ListView`.

В строке 65 унаследованный от `ListActivity` метод `getListView` используется для получения ссылки на встроенный компонент `ListView`. В строке 66 слушателем `OnItemClickListener` для `ListView` назначается `viewContactListener` (листинг 8.16), реагирующий на касание контакта в `ListView`. В строке 67 метод `setSelectionMode` класса `ListView` вызывается для обозначения того, что в любой момент времени может быть выбран только один пункт списка.

Настройка адаптера `CursorAdapter`, связывающего информацию из базы данных с `ListView`

Чтобы вывести результаты выборки в `ListView`, мы создаем новый объект `CursorAdapter` (строки 70–73), который предоставляет доступ к данным `Cursor` так, чтобы они могли использоваться компонентом `ListView`. `SimpleCursorAdapter` — субкласс `CursorAdapter`, упрощающий связывание столбцов `Cursor` с компонентами `TextView` и `ImageView`, определенными в макетах XML. Чтобы создать объект `SimpleCursorAdapter`, мы сначала определяем массивы с именами столбцов, связываемых с компонентами GUI, и идентификаторами ресурсов компонентов, в которых будут отображаться данные из этих столбцов. В строке 70 создается массив `String`, который указывает, что отображаться будет только столбец "name", а в строке 71 создается параллельный массив `int` с идентификаторами ресурсов соответствующих компонентов. В главе 4 было показано, как создавать макеты ресурсов для элементов `ListView`. В этом приложении используется предопределенный ресурс макета с именем `android.R.layout.simple_list_item_1` — макет, содержащий один компонент `TextView` с идентификатором `android.R.id.text1`. В строках 72–73 создается объект `SimpleCursorAdapter`. Его конструктор получает следующие параметры.

- Контекст, в котором выполняется `ListView` (например, `MainActivity`).
- Идентификатор ресурса макета, используемый для отображения каждого элемента `ListView`.
- Объект `Cursor`, предоставляющий доступ к данным, — в этом аргументе передается `null`, потому что объект `Cursor` будет задан позднее.

- ❑ Массив строк с именами отображаемых столбцов.
- ❑ Массив `int` с соответствующими идентификаторами ресурсов GUI.
- ❑ В последнем аргументе обычно передается 0.

В строке 74 унаследованный от `ListActivity` метод `setListAdapter` используется для связывания `ListView` с `CursorAdapter`, чтобы компонент `ListView` мог вывести данные.

viewContactListener для обработки событий выбора элементов ListView

Слушатель `viewContactListener` (листинг 8.16) оповещает `MainActivity` о том, что пользователь коснулся контакта, чтобы вывести подробную информацию. В строке 84 аргумент `id` (идентификатор строки выбранного контакта) передается методу `onContactSelected` слушателя (листинг 8.5).

Листинг 8.16. Слушатель viewContactListener для обработки событий выбора элементов ListView

```

77    // Обработка касания имени контакта в ListView
78    OnItemClickListener viewContactListener = new OnItemClickListener()
79    {
80        @Override
81        public void onItemClick(AdapterView<?> parent, View view,
82            int position, long id)
83        {
84            listener.onContactSelected(id); // Выбранный элемент передается
                                              // MainActivity
85        }
86    }; // Конец viewContactListener
87

```

Переопределенный метод onResume класса ContactListFragment

Метод `onResume` жизненного цикла фрагмента (листинг 8.17) создает и выполняет задачу `AsyncTask` (строка 93) типа `GetContactsTask` (листинг 8.18), которая получает полный список контактов из базы данных и назначает курсор объекта `contactAdapter` для заполнения компонента `ListView` фрагмента `ContactListFragment`. Метод `execute` класса `AsyncTask` выполняет задачу в отдельном программном потоке.

Аргумент метода `execute` в данном случае указывает, что задача не получает аргументов, — этот метод может получать переменное количество аргументов, которые в свою очередь передаются в аргументах метода `doInBackground` задачи. При каждом выполнении строки 93 создается новый объект `GetContactsTask` — это необходимо, потому что каждая задача `AsyncTask` может выполняться только один раз.

Листинг 8.17. Переопределенный метод onResume класса ContactListFragment

```

88    // При возобновлении фрагмента задача GetContactsTask загружает контакты
89    @Override
90    public void onResume()
91    {

```

```
92     super.onResume();  
93     new GetContactsTask().execute((Object[]) null);  
94 }  
95
```

Класс GetContactsTask

Вложенный класс `GetContactsTask` (листинг 8.18) расширяет класс `AsyncTask`. Этот класс определяет взаимодействия с объектом `DatabaseConnector` (раздел 8.9) для получения всех контактов и возвращения результатов потоку GUI активности для отображения в `ListView`. Обобщенный тип `AsyncTask` получает три параметра-типа.

- ❑ Тип списка параметров переменной длины для метода `doInBackground` класса `AsyncTask` (строки 103–108) — при вызове `execute` метод `doInBackground` выполняет задачу в отдельном потоке. Мы указываем `Object` в качестве параметра-типа и передаем `null` в аргументе метода `execute` класса `AsyncTask`, потому что `GetContactsTask` не требует дополнительных данных для выполнения своей задачи.
- ❑ Тип списка параметров переменной длины для метода `onProgressUpdate` класса `AsyncTask` — этот метод выполняется в потоке GUI и используется для получения обновлений от продолжительной задачи. В нашем примере эта возможность не используется, поэтому мы передаем тип `Object` и игнорируем этот параметр-тип.
- ❑ Тип результата задачи, передаваемый методу `onPostExecute` класса `AsyncTask` (строки 111–116), — этот метод выполняется в потоке GUI и позволяет `ContactListFragment` использовать результаты `AsyncTask`.

Важнейшее преимущество класса `AsyncTask` заключается в том, что он берет на себя все технические подробности создания программных потоков и выполнения методов в соответствующих потоках, чтобы вам не приходилось выполнять низкоуровневые операции.

Листинг 8.18. Субкласс GetContactsTask класса AsyncTask

```
96 // Выполнение запроса к базе данных вне потока GUI  
97 private class GetContactsTask extends AsyncTask<Object, Object, Cursor>  
98 {  
99     DatabaseConnector databaseConnector =  
100     new DatabaseConnector(getActivity());  
101  
102     // Открыть базу данных и вернуть курсор (Cursor) для всех контактов  
103     @Override  
104     protected Cursor doInBackground(Object... params)  
105     {  
106         databaseConnector.open();  
107         return databaseConnector.getAllContacts();  
108     }  
109  
110     // Использовать курсор, полученный от метода doInBackground  
111     @Override
```

```

112     protected void onPostExecute(Cursor result)
113     {
114         contactAdapter.changeCursor(result); // Назначение курсора
115                                     // для адаптера
116     }
117 } // Конец класса GetContactsTask
118

```

В строках 99–100 создается новый объект вспомогательного класса `DatabaseConnector`, при этом конструктору класса в качестве аргумента передается контекст (управляющая активность фрагмента `ContactListFragment`). Метод `doInBackground` использует объект `databaseConnector` для открытия подключения к базе данных и выборки всех контактов. Объект `Cursor`, возвращаемый `getAllContacts`, передается методу `onPostExecute`, который получает объект `Cursor` с результатами и передает его методу `changeCursor` объекта `contactAdapter`. Это позволяет заполнить компонент `ListView` класса `ContactListFragment` именами контактов.

Переопределенный метод `onStop` класса `ContactListFragment`

Метод `onStop` жизненного цикла фрагмента (листинг 8.19) вызывается после метода `onPause`, когда фрагмент становится невидимым для пользователя. В данном случае объект `Cursor`, позволяющий заполнить `ListView`, не нужен, поэтому в строке 123 вызывается метод `getCursor` класса `CursorAdapter` вызывается для получения текущего курсора от `contactAdapter`. Стока 124 вызывает метод `changeCursor` класса `CursorAdapter` с аргументом `null` для удаления `Cursor` из `CursorAdapter`. Затем строка 127 вызывает метод `close` класса `Cursor` для освобождения ресурсов, используемых `Cursor`.

Листинг 8.19. Переопределенный метод `onStop` класса `ContactListFragment`

```

119 // При остановке фрагмента закрыть курсор и удалить из contactAdapter
120 @Override
121 public void onStop()
122 {
123     Cursor cursor = contactAdapter.getCursor(); // Получение текущего
124                                     // курсора
125     contactAdapter.changeCursor(null); // Адаптер не имеет курсора
126
127     if (cursor != null)
128         cursor.close(); // Освобождение ресурсов курсора
129
130     super.onStop();
131 }

```

Переопределенные методы `onCreateOptionsMenu` и `onOptionsItemSelected` класса `ContactListFragment`

Метод `onCreateOptionsMenu` (листинг 8.20, строки 133–138) использует аргумент `MenuInflater` для создания меню на основе файла `fragment_contact_list_menu.xml`,

содержащего определение команды добавления (⊕). Если пользователь касается этой команды, метод `onOptionsItemSelected` (строки 141–152) вызывает метод `onAddContact`, чтобы оповестить `MainActivity` о добавлении нового контакта. Затем `MainActivity` отображает `AddEditFragment` (раздел 8.7).

Листинг 8.20. Переопределенные методы `onCreateOptionsMenu` и `onOptionsItemSelected` класса `ContactListFragment`

```

132 // Отображение команд меню фрагмента
133 @Override
134 public void onCreateOptionsMenu(Menu menu, MenuInflater inflater)
135 {
136     super.onCreateOptionsMenu(menu, inflater);
137     inflater.inflate(R.menu.fragment_contact_list_menu, menu);
138 }
139
140 // Обработка выбора команды из меню
141 @Override
142 public boolean onOptionsItemSelected(MenuItem item)
143 {
144     switch (item.getItemId())
145     {
146         case R.id.action_add:
147             listener.onAddContact();
148             return true;
149     }
150
151     return super.onOptionsItemSelected(item); // Вызов метода суперкласса
152 }
153

```

Метод `updateContactList` класса `ContactListFragment`

Метод `updateContactList` (листинг 8.21) создает и выполняет задачу `GetContactsTask` для обновления списка контактов.

Листинг 8.21. Метод `updateContactList` класса `ContactListFragment`

```

154 // Обновление набора данных
155 public void updateContactList()
156 {
157     new GetContactsTask().execute((Object[]) null);
158 }
159 // Конец класса ContactListFragment

```

8.7. Класс `AddEditFragment`

Класс `AddEditFragment` (листинги 8.22–8.28) предоставляет интерфейс для добавления новых или редактирования существующих контактов.

Команда package и команды import класса AddEditFragment

В листинге 8.22 приведены команды package и import для класса AddEditFragment. Никакие новые классы в этом фрагменте не используются.

Листинг 8.22. Команда package, команды import и поля класса AddEditFragment

```

1 // AddEditFragment.java
2 // Добавление нового или редактирование существующего контакта
3 package com.deitel.addressbook;
4
5 import android.app.Activity;
6 import android.app.AlertDialog;
7 import android.app.Dialog;
8 import android.app.DialogFragment;
9 import android.app.Fragment;
10 import android.content.Context;
11 import android.os.AsyncTask;
12 import android.os.Bundle;
13 import android.view.LayoutInflater;
14 import android.view.View;
15 import android.view.View.OnClickListener;
16 import android.view.ViewGroup;
17 import android.view.inputmethod.InputMethodManager;
18 import android.widget.Button;
19 import android.widget.EditText;
20
21 public class AddEditFragment extends Fragment
22 {

```

Интерфейс AddEditFragmentListener

В листинге 8.23 объявляется вложенный интерфейс AddEditFragmentListener, содержащий метод обратного вызова onAddEditCompleted, реализуемый MainActivity для оповещения о сохранении пользователем нового или измененного существующего контакта.

Листинг 8.23. Интерфейс AddEditFragmentListener

```

23 // Метод обратного вызова, реализуемый MainActivity
24 public interface AddEditFragmentListener
25 {
26     // Вызывается после завершения редактирования
27     public void onAddEditCompleted(long rowID);
28 }
29

```

Переменные экземпляров AddEditFragment

В листинге 8.24 перечислены переменные экземпляров класса.

- ❑ Переменная *listener* содержит ссылку на слушателя AddEditFragmentListener, который оповещается о том, что пользователь нажал кнопку Save Contact.

- ❑ Переменная `rowID` представляет текущий контакт, если фрагмент отображался для редактирования существующего контакта.
- ❑ Переменная `contactInfoBundle` равна `null` при добавлении нового контакта или содержит ссылку на `Bundle` с данными контакта при редактировании существующего контакта.
- ❑ Переменные экземпляров в строках 36–42 содержат ссылки на компоненты `EditText` фрагмента.

Листинг 8.24. Переменные экземпляров `AddEditFragmentListener`

```
30     private AddEditFragmentListener listener;
31
32     private long rowID; // Идентификатор строки контакта в базе данных
33     private Bundle contactInfoBundle; // Аргументы для редактирования контакта
34
35     // Компоненты EditText для информации контакта
36     private EditText nameEditText;
37     private EditText phoneEditText;
38     private EditText emailEditText;
39     private EditText streetEditText;
40     private EditText cityEditText;
41     private EditText stateEditText;
42     private EditText zipEditText;
43
```

Переопределенные методы `onAttach` и `onDetach` класса `AddEditFragment`

Класс `AddEditFragment` переопределяет методы `onAttach` и `onDetach` жизненного цикла фрагмента (листинг 8.25) таким образом, чтобы переменной экземпляра `listener` присваивалась ссылка на управляющую активность (строка 49) при присоединении `AddEditFragment` или `null` (строка 57) при отсоединении `AddEditFragment`.

Листинг 8.25. Переопределенные методы `onAttach` и `onDetach` класса `AddEditFragment`

```
44     // Назначение AddEditFragmentListener при присоединении фрагмента
45     @Override
46     public void onAttach(Activity activity)
47     {
48         super.onAttach(activity);
49         listener = (AddEditFragmentListener) activity;
50     }
51
52     // Удаление AddEditFragmentListener при отсоединении фрагмента
53     @Override
54     public void onDetach()
55     {
56         super.onDetach();
57         listener = null;
58     }
59
```

Переопределенный метод onCreateView класса AddEditFragment

Строки 70–78 метода onCreateView (листинг 8.26) заполняют графический интерфейс и получают компоненты EditText фрагмента. Затем метод getArguments фрагмента используется для получения объекта Bundle с аргументами (если они есть). При запуске AddEditFragment из MainActivity объект Bundle не передается, потому что пользователь добавляет информацию нового контакта. В этом случае метод getArguments вернет null. Если он вернет Bundle (строка 82), значит выполнение AddEditFragment было инициировано из DetailsFragment, а пользователь выбрал команду редактирования существующего контакта. В строках 84–91 аргументы из Bundle читаются вызовом методов getLong (строка 84) и getString, а данные String отображаются в компонентах EditText для редактирования. В строках 95–97 регистрируется слушатель (листинг 8.27) для кнопки Save Contact.

Листинг 8.26. Переопределенный метод onCreateView класса AddEditFragment

```

60    // Вызывается при необходимости создания представления фрагмента
61    @Override
62    public View onCreateView(LayoutInflater inflater, ViewGroup container,
63        Bundle savedInstanceState)
64    {
65        super.onCreateView(inflater, container, savedInstanceState);
66        setRetainInstance(true); // Сохраняется между изменениями конфигурации
67        setHasOptionsMenu(true); // У фрагмента есть команды меню
68
69        // Заполнение GUI и получение ссылок на компоненты EditText
70        View view =
71            inflater.inflate(R.layout.fragment_add_edit, container, false);
72        nameEditText = (EditText) view.findViewById(R.id.nameEditText);
73        phoneEditText = (EditText) view.findViewById(R.id.phoneEditText);
74        emailEditText = (EditText) view.findViewById(R.id.emailEditText);
75        streetEditText = (EditText) view.findViewById(R.id.streetEditText);
76        cityEditText = (EditText) view.findViewById(R.id.cityEditText);
77        stateEditText = (EditText) view.findViewById(R.id.stateEditText);
78        zipEditText = (EditText) view.findViewById(R.id.zipEditText);
79
80        contactInfoBundle = getArguments(); // null при создании нового контакта
81
82        if (contactInfoBundle != null)
83        {
84            rowID = contactInfoBundle.getLong(MainActivity.ROW_ID);
85            nameEditText.setText(contactInfoBundle.getString("name"));
86            phoneEditText.setText(contactInfoBundle.getString("phone"));
87            emailEditText.setText(contactInfoBundle.getString("email"));
88            streetEditText.setText(contactInfoBundle.getString("street"));
89            cityEditText.setText(contactInfoBundle.getString("city"));
90            stateEditText.setText(contactInfoBundle.getString("state"));
91            zipEditText.setText(contactInfoBundle.getString("zip"));
92        }
93
94        // Назначение слушателя событий кнопки Save Contact
95        Button saveContactButton =
96            (Button) view.findViewById(R.id.saveContactButton);

```

```
97     saveContactButton.setOnClickListener(saveContactButtonClicked);
98     return view;
99 }
100
```

Слушатель OnClickListener для обработки событий кнопки Save Contact

Когда пользователь касается кнопки Save Contact, выполняется слушатель `saveContactButtonClicked` (листинг 8.27). Чтобы сохранить данные контакта, пользователь должен ввести как минимум его имя. Метод `onClick` проверяет, что длина имени превышает 0 символов (строка 107), и, если это так — создает и выполняет `AsyncTask` (для выполнения операции сохранения). Метод `doInBackground` (строки 113–118) вызывает `saveContact` (листинг 8.28) для сохранения контакта в базе данных. Метод `onPostExecute` (строки 120–131) скрывает клавиатуру (строки 124–128) и оповещает `MainActivity` о том, что контакт был сохранен (строка 130). Если `nameEditText` не содержит данных, то строки 139–153 выводят `DialogFragment` с сообщением о том, что для сохранения контакта необходимо ввести имя.

Листинг 8.27. Метод onClickListener для обработки событий кнопки Save Contact

```
101 // Обработка события, генерируемого при сохранении контакта
102 OnClickListener saveContactButtonClicked = new OnClickListener()
103 {
104     @Override
105     public void onClick(View v)
106     {
107         if (nameEditText.getText().toString().trim().length() != 0)
108         {
109             // AsyncTask сохраняет контакт и оповещает слушателя
110             AsyncTask<Object, Object, Object> saveContactTask =
111                 new AsyncTask<Object, Object, Object>()
112             {
113                 @Override
114                 protected Object doInBackground(Object... params)
115                 {
116                     saveContact(); // Сохранение контакта в базе данных
117                     return null;
118                 }
119
120                 @Override
121                 protected void onPostExecute(Object result)
122                 {
123                     // Скрывается виртуальная клавиатура
124                     InputMethodManager imm = (InputMethodManager)
125                         getActivity().getSystemService(
126                             Context.INPUT_METHOD_SERVICE);
127                     imm.hideSoftInputFromWindow(
128                         getView().getWindowToken(), 0);
129
130                     listener.onAddEditCompleted(rowID);
131                 }
132             }; // Конец AsyncTask
133 }
```

```

134          // Контакт сохраняется в базе данных в отдельном потоке
135          saveContactTask.execute((Object[]) null);
136      }
137      else // Имя не задано, вывести сообщение об ошибке
138      {
139          DialogFragment errorSaving =
140              new DialogFragment()
141              {
142                  @Override
143                  public Dialog onCreateDialog(Bundle savedInstanceState)
144                  {
145                      AlertDialog.Builder builder =
146                          new AlertDialog.Builder(getActivity());
147                      builder.setMessage(R.string.error_message);
148                      builder.setPositiveButton(R.string.ok, null);
149                      return builder.create();
150                  }
151              };
152
153          errorSaving.show(getFragmentManager(), "error saving contact");
154      }
155  } // Конец метода onClick
156 }; // Конец OnClickListener
157

```

Метод saveContact класса AddEditFragment

Метод `saveContact` (листинг 8.28) сохраняет информацию, содержащуюся в компонентах `EditText` текущего фрагмента. Сначала в строках 162–163 создается объект `DatabaseConnector`, после чего мы проверяем, содержит ли `contactInfoBundle` значение `null`. Если это так, значит пользователь создает новый контакт; в строках 168–175 извлекаются текстовые данные из `EditText`, которые передаются методу `insertContact` объекта `DatabaseConnector` для создания нового контакта. Если вместо `null` используется объект `Bundle`, значит обновляется существующий контакт. В этом случае данные из `EditText` передаются методу `updateContact` объекта `DatabaseConnector`, а идентификатор `rowID` указывает, какая запись должна быть обновлена. Методы `insertContact` и `updateContact` класса `DatabaseConnector` открывают и закрывают базу данных.

Листинг 8.28. Метод saveContact класса AddEditFragment

```

158  // Сохранение информации контакта в базе данных
159  private void saveContact()
160  {
161      // Получение DatabaseConnector для работы с базой данных SQLite
162      DatabaseConnector databaseConnector =
163          new DatabaseConnector(getActivity());
164
165      if (contactInfoBundle == null)
166      {
167          // Вставка контакта в базу данных
168          rowID = databaseConnector.insertContact(

```

```
169         nameEditText.getText().toString(),
170         phoneEditText.getText().toString(),
171         emailEditText.getText().toString(),
172         streetEditText.getText().toString(),
173         cityEditText.getText().toString(),
174         stateEditText.getText().toString(),
175         zipEditText.getText().toString());
176     }
177     else
178     {
179         databaseConnector.updateContact(rowID,
180             nameEditText.getText().toString(),
181             phoneEditText.getText().toString(),
182             emailEditText.getText().toString(),
183             streetEditText.getText().toString(),
184             cityEditText.getText().toString(),
185             stateEditText.getText().toString(),
186             zipEditText.getText().toString());
187     }
188 } // Конец метода saveContact
189 } // Конец класса AddEditFragment
```

8.8. Класс DetailsFragment

Класс **DetailsFragment** (листинги 8.29–8.38) выводит информацию одного контакта и предоставляет команды меню, при помощи которых пользователь может изменить или удалить данные контакта.

Команда package и команды import класса DetailsFragment

В листинге 8.29 приведены команда **package**, команды **import** и начало объявления класса **DetailsFragment**. Никакие новые классы и интерфейсы в этом классе не используются.

Листинг 8.29. Команда package и команды import класса DetailsFragment

```
1 // DetailsFragment.java
2 // Вывод подробной информации о контакте
3 package com.deitel.addressbook;
4
5 import android.app.Activity;
6 import android.app.AlertDialog;
7 import android.app.Dialog;
8 import android.app.DialogFragment;
9 import android.app.Fragment;
10 import android.content.DialogInterface;
11 import android.database.Cursor;
12 import android.os.AsyncTask;
13 import android.os.Bundle;
14 import android.view.LayoutInflater;
15 import android.view.Menu;
16 import android.view.MenuInflater;
17 import android.view.MenuItem;
```

```

18 import android.view.View;
19 import android.view.ViewGroup;
20 import android.widget.TextView;
21
22 public class DetailsFragment extends Fragment
23 {

```

Интерфейс DetailsFragmentListener

В листинге 8.30 объявляется вложенный интерфейс `DetailsFragmentListener`, содержащий методы обратного вызова, реализуемые `MainActivity` для оповещения об удалении контакта (строка 28) и касании команды меню для редактирования контакта (строка 31).

Листинг 8.30. Интерфейс DetailsFragmentListener

```

24     // Методы обратного вызова, реализуемые MainActivity
25     public interface DetailsFragmentListener
26     {
27         // Вызывается при удалении контакта
28         public void onContactDeleted();
29         // Вызывается для передачи объекта Bundle
30         // с данными редактируемого контакта.
31         public void onEditContact(Bundle arguments);
32     }
33

```

Переменные экземпляров DetailsFragment

В листинге 8.31 приведены переменные экземпляров класса. В строке 34 объявляется переменная `listener`, содержащая ссылку на объект (`MainActivity`), реализующий интерфейс `DetailsFragmentListener`. Переменная `rowID` представляет уникальный идентификатор записи контакта в базе данных. Переменные экземпляров `TextView` (строки 37–43) используются для отображения данных контакта на экране.

Листинг 8.31. Переменные экземпляров DetailsFragment

```

34     private DetailsFragmentListener listener;
35
36     private long rowID = -1;           // rowID выделенного контакта
37     private TextView nameTextView;    // Имя
38     private TextView phoneTextView;   // Телефон
39     private TextView emailTextView;  // Адрес электронной почты
40     private TextView streetTextView; // Улица
41     private TextView cityTextView;   // Город
42     private TextView stateTextView; // Штат
43     private TextView zipTextView;   // Почтовый индекс
44

```

Переопределенные методы `onAttach` и `onDetach` класса DetailsFragment

Класс `DetailsFragment` переопределяет методы `onAttach` и `onDetach` жизненного цикла фрагмента (листинг 8.32) таким образом, чтобы переменной экземпляра

listener присваивалось нужное значение при присоединении или отсоединении DetailsFragment.

Листинг 8.32. Переопределенные методы onAttach и onDetach класса DetailsFragment

```
45 // Назначение DetailsFragmentListener при присоединении фрагмента
46 @Override
47 public void onAttach(Activity activity)
48 {
49     super.onAttach(activity);
50     listener = (DetailsFragmentListener) activity;
51 }
52
53 // Удаление DetailsFragmentListener при отсоединении фрагмента
54 @Override
55 public void onDetach()
56 {
57     super.onDetach();
58     listener = null;
59 }
60
```

Переопределенный метод onCreateView класса DetailsFragment

Метод onCreateView (листинг 8.33) получает идентификатор rowID выбранного контакта (строки 70–79). Если фрагмент восстанавливается, значение rowID загружается из набора savedInstanceState; в противном случае мы получаем его идентификатор объекта Bundle с аргументами. В строках 82–93 приложение заполняет графический интерфейс и инициализирует ссылки на компоненты TextView.

Листинг 8.33. Переопределенный метод onCreateView класса DetailsFragment

```
61 // Вызывается при необходимости создания представления фрагмента
62 @Override
63 public View onCreateView(LayoutInflater inflater, ViewGroup container,
64     Bundle savedInstanceState)
65 {
66     super.onCreateView(inflater, container, savedInstanceState);
67     setRetainInstance(true); // Сохраняется между изменениями конфигурации
68
69     // Если фрагмент восстанавливается, получить сохраненное значение rowID
70     if (savedInstanceState != null)
71         rowID = savedInstanceState.getLong(MainActivity.ROW_ID);
72     else
73     {
74         // Получение набора аргументов и извлечение rowID контакта
75         Bundle arguments = getArguments();
76
77         if (arguments != null)
78             rowID = arguments.getLong(MainActivity.ROW_ID);
79     }
80
81     // Заполнение макета DetailsFragment
```

```

82     View view =
83         inflater.inflate(R.layout.fragment_details, container, false);
84     setHasOptionsMenu(true); // У фрагмента есть команды меню
85
86     // Получение ссылок на EditText
87     nameTextView = (EditText) view.findViewById(R.id.nameTextView);
88     phoneTextView = (EditText) view.findViewById(R.id.phoneTextView);
89     emailTextView = (EditText) view.findViewById(R.id.emailTextView);
90     streetTextView = (EditText) view.findViewById(R.id.streetTextView);
91     cityTextView = (EditText) view.findViewById(R.id.cityTextView);
92     stateTextView = (EditText) view.findViewById(R.id.stateTextView);
93     zipTextView = (EditText) view.findViewById(R.id.zipTextView);
94     return view;
95 }
96

```

Переопределенный метод onResume класса DetailsFragment

Метод onResume жизненного цикла фрагмента (листинг 8.34) создает и выполняет задачу AsyncTask (строка 102) типа LoadContactTask (листинг 8.37), которая получает заданный контакт из базы данных и отображает его данные. В аргументе метода execute передается идентификатор rowID загружаемого контакта. При каждом выполнении строки 102 он создает новый объект LoadContactTask — напомним, что это необходимо, потому что каждая задача AsyncTask может выполняться только один раз.

Листинг 8.34. Переопределенный метод onResume класса DetailsFragment

```

97     // Вызывается при продолжении выполнения DetailsFragment
98     @Override
99     public void onResume()
100    {
101        super.onResume();
102        new LoadContactTask().execute(rowID); // Загрузка контакта по rowID
103    }
104

```

Переопределенный метод onSaveInstanceState класса DetailsFragment

Метод onSaveInstanceState фрагмента (листинг 8.35) сохраняет rowID выбранного контакта при изменении конфигурации устройства в ходе выполнения приложения — например, когда пользователь поворачивает устройство или выдвигает клавиатуру на устройстве с физической клавиатурой. Состояние компонентов GUI сохраняется автоматически, но все остальные элементы, которые должны восстанавливаться при изменениях конфигурации, следует хранить в объекте Bundle, получаемом onSaveInstanceState.

Листинг 8.35. Переопределенный метод onSaveInstanceState класса DetailsFragment

```

105    // Сохранение rowID текущего контакта
106    @Override
107    public void onSaveInstanceState(Bundle outState)

```

```

108     {
109         super.onSaveInstanceState(outState);
110         outState.putLong(MainActivity.ROW_ID, rowID);
111     }
112

```

Переопределенные методы onCreateOptionsMenu и onOptionsItemSelected класса DetailsFragment

Меню DetailsFragment предоставляет команды для редактирования текущего контакта и его удаления. Метод onCreateOptionsMenu (листинг 8.36, строки 114–119) заполняет меню по ресурсному файлу fragment_details_menu.xml. Метод onOptionsItemSelected (строки 122–146) использует идентификатор ресурса выбранного элемента MenuItem для определения того, какой контакт был выбран. Если пользователь выбрал команду меню с идентификатором R.id.action_edit, строки 129–137 создают объект Bundle с данными контакта, после чего строка 138 передает Bundle слушателю DetailsFragmentListener для использования в AddEditFragment. Если пользователь выбрал команду меню с идентификатором R.id.action_delete, то строка 141 вызывает метод deleteContact (листинг 8.38).

Листинг 8.36. Переопределенные методы onCreateOptionsMenu и onOptionsItemSelected класса DetailsFragment

```

113     // Отображение команд меню фрагмента
114     @Override
115     public void onCreateOptionsMenu(Menu menu, MenuInflater inflater)
116     {
117         super.onCreateOptionsMenu(menu, inflater);
118         inflater.inflate(R.menu.fragment_details_menu, menu);
119     }
120
121     // Обработка выбора команд меню
122     @Override
123     public boolean onOptionsItemSelected(MenuItem item)
124     {
125         switch (item.getItemId())
126         {
127             case R.id.action_edit:
128                 // Создание объекта Bundle с данными редактируемого контакта
129                 Bundle arguments = new Bundle();
130                 arguments.putLong(MainActivity.ROW_ID, rowID);
131                 arguments.putCharSequence("name", nameTextView.getText());
132                 arguments.putCharSequence("phone", phoneTextView.getText());
133                 arguments.putCharSequence("email", emailTextView.getText());
134                 arguments.putCharSequence("street", streetTextView.getText());
135                 arguments.putCharSequence("city", cityTextView.getText());
136                 arguments.putCharSequence("state", stateTextView.getText());
137                 arguments.putCharSequence("zip", zipTextView.getText());
138                 listener.onEditContact(arguments); // Объект Bundle передается
139                                         // слушателю
140             return true;
141             case R.id.action_delete:

```

```

141         deleteContact();
142         return true;
143     }
144
145     return super.onOptionsItemSelected(item);
146 }
147

```

Класс LoadContactTask

Вложенный класс `LoadContactTask` (листинг 8.37) расширяет класс `AsyncTask`. Этот класс определяет взаимодействия с объектом `DatabaseConnector` (раздел 8.9) для получения информации одного контакта. В данном случае передаются три параметра-типа.

- ❑ `Long` для списка аргументов переменной длины, передаваемого методу `doInBackground` класса `AsyncTask`. Передается идентификатор `rowID`, необходимый для выборки одного контакта.
- ❑ `Object` для списка аргументов переменной длины, передаваемого методу `onProgressUpdate` класса `AsyncTask`. В нашем примере он не используется.
- ❑ `Cursor` для типа результата задачи, который передается методу `onPostExecute` класса `AsyncTask`.

Листинг 8.37. Класс LoadContactTask

```

148 // Выполнение запроса к базе данных вне потока GUI
149 private class LoadContactTask extends AsyncTask<Long, Object, Cursor>
150 {
151     DatabaseConnector databaseConnector =
152         new DatabaseConnector(getActivity());
153
154     // Открыть базу данных и вернуть курсор для данных указанного контакта
155     @Override
156     protected Cursor doInBackground(Long... params)
157     {
158         databaseConnector.open();
159         return databaseConnector.getOneContact(params[0]);
160     }
161
162     // Использовать курсор, полученный от метода doInBackground
163     @Override
164     protected void onPostExecute(Cursor result)
165     {
166         super.onPostExecute(result);
167         result.moveToFirst(); // Перемещение в начало
168
169         // Получение индекса столбца для каждого элемента данных
170         int nameIndex = result.getColumnIndex("name");
171         int phoneIndex = result.getColumnIndex("phone");
172         int emailIndex = result.getColumnIndex("email");
173         int streetIndex = result.getColumnIndex("street");
174         int cityIndex = result.getColumnIndex("city");

```

```
175     int stateIndex = result.getColumnIndex("state");
176     int zipIndex = result.getColumnIndex("zip");
177
178     // Заполнение компонентов TextView с загруженными данными
179     nameTextView.setText(result.getString(nameIndex));
180     phoneTextView.setText(result.getString(phoneIndex));
181     emailTextView.setText(result.getString(emailIndex));
182     streetTextView.setText(result.getString(streetIndex));
183     cityTextView.setText(result.getString(cityIndex));
184     stateTextView.setText(result.getString(stateIndex));
185     zipTextView.setText(result.getString(zipIndex));
186
187     result.close(); // Закрытие курсора
188     databaseConnector.close(); // Закрытие подключения к базе данных
189 } // Конец метода onPostExecute
190 } // Конец класса LoadContactTask
191
```

В строках 151–152 создается новый объект класса `DatabaseConnector` (раздел 8.9). Метод `doInBackground` (строки 155–160) открывает подключение к базе данных и вызывает метод `getOneContact` класса `DatabaseConnector`, который обращается с запросом к базе данных для получения контакта с заданным значением `rowID`, переданным в единственном аргументе метода `execute` задачи `AsyncTask`. В `doInBackground` значение `rowID` сохраняется в `params[0]`.

Объект курсора `Cursor` передается методу `onPostExecute` (строки 163–189). Курсор позиционируется перед первой строкой результирующего набора. В данном случае результирующий набор содержит всего одну запись, а метод `moveToFirst` класса `Cursor` (строка 167) может использоваться для перемещения курсора к первой строке набора. [Примечание: перед получением данных от курсора рекомендуется убедиться в том, что метод `moveToFirst` вернул `true`; впрочем, в нашем приложении курсор всегда содержит хотя бы одну строку данных.]

Метод `getColumnIndex` класса `Cursor` (строки 170–176) используется для получения индексов столбцов таблицы `contacts` базы данных. (В нашем приложении используются фиксированные имена столбцов, но их также можно было реализовать в виде строковых констант, как было сделано с `ROW_ID` в классе `MainActivity` из листинга 8.2.) Метод возвращает `-1`, если столбец отсутствует в результатах запроса. Класс `Cursor` также предоставляет метод `getColumnIndexOrThrow`, если вы предпочитаете обрабатывать исключение при отсутствии столбца с заданным именем. В строках 179–185 метод `getString` класса `Cursor` используется для чтения значений `String` из столбцов `Cursor`, с их последующим отображением в соответствующих компонентах `TextView`. Строки 187–188 закрывают `Cursor` и подключение к базе данных — они более не нужны. Рекомендуется освобождать неиспользуемые ресурсы (такие, как подключения к базам данных), чтобы они могли использоваться другими активностями.

Метод `deleteContact` и фрагмент `confirmDelete`

Метод `deleteContact` (листинг 8.38, строки 193–197) отображает фрагмент `DialogFragment` (строки 200–252) с предложением подтвердить удаление текущего

контакта. Если удаление подтверждается, `DialogFragment` использует задачу `AsyncTask` для удаления контакта из базы данных. Если же пользователь нажмет кнопку `Delete`, в строках 222–223 создается новый объект `DatabaseConnector`. В строках 226–241 создается задача `AsyncTask`, которая при выполнении (строка 244) передает значение `Long`, представляющее идентификатор записи контакта, методу `doInBackground`, который удаляет контакт. Стока 232 вызывает метод `deleteContact` класса `DatabaseConnector` для выполнения непосредственного удаления. Когда метод `doInBackground` завершает выполнение, в строке 239 вызывается метод `onContactDeleted` слушателя, чтобы активность `MainActivity` могла удалить `DetailsFragment` с экрана.

Листинг 8.38. Метод `deleteContact` и фрагмент `DialogFragment`

```

192    // Удаление контакта
193    private void deleteContact()
194    {
195        // FragmentManager используется для отображения фрагмента confirmDelete
196        confirmDelete.show(getFragmentManager(), "confirm delete");
197    }
198
199    // DialogFragment для подтверждения удаления контакта
200    private DialogFragment confirmDelete =
201        new DialogFragment()
202    {
203        // Создание и возвращение AlertDialog
204        @Override
205        public Dialog onCreateDialog(Bundle bundle)
206        {
207            // Создание нового объекта Builder для AlertDialog
208            AlertDialog.Builder builder =
209                new AlertDialog.Builder(getActivity());
210
211            builder.setTitle(R.string.confirm_title);
212            builder.setMessage(R.string.confirm_message);
213
214            // Кнопка OK просто закрывает диалоговое окно
215            builder.setPositiveButton(R.string.button_delete,
216                new DialogInterface.OnClickListener()
217                {
218                    @Override
219                    public void onClick(
220                        DialogInterface dialog, int button)
221                    {
222                        final DatabaseConnector databaseConnector =
223                            new DatabaseConnector(getActivity());
224
225                        // AsyncTask удаляет контакт и оповещает слушателя
226                        AsyncTask<Long, Object, Object> deleteTask =
227                            new AsyncTask<Long, Object, Object>()
228                            {
229                                @Override
230                                protected Object doInBackground(Long... params)
231                                {
232                                    databaseConnector.deleteContact(params[0]);

```

```

233                     return null;
234                 }
235
236             @Override
237             protected void onPostExecute(Object result)
238             {
239                 listener.onContactDeleted();
240             }
241         }; // Конец new AsyncTask
242
243         // Выполнение AsyncTask для удаления контакта по rowID
244         deleteTask.execute(new Long[] { rowID });
245     } // Конец метода onClick
246 } // Конец анонимного внутреннего класса
247 ); // Конец вызова метода setPositiveButton
248
249         builder.setNegativeButton(R.string.button_cancel, null);
250         return builder.create(); // Возвращает AlertDialog
251     }
252 }; // Конец анонимного внутреннего класса DialogFragment
253 } // Конец класса DetailsFragment

```

8.9. Вспомогательный класс DatabaseConnector

Вспомогательный класс `DatabaseConnector` (листинги 8.39–8.46) управляет взаимодействием приложения с SQLite для создания и работы с базой данных `UserContacts`, содержащей единственную таблицу с именем `contacts`.

Команда `package`, команды `import` и поля

В листинге 8.39 приведены команда `package`, команды `import` и поля класса `DatabaseConnector`. В листинге выделены команды `import` для новых классов и интерфейсов, рассматриваемых в разделе 8.3. Строковая константа `DATABASE_NAME` (строка 16) задает имя создаваемой или открываемой базы данных. Имена баз данных должны быть уникальными в пределах одного приложения, но их уникальность между приложениями не обязательна. Объект `SQLiteDatabase` (строка 18) предоставляет доступ к базе данных SQLite для чтения/записи. `DatabaseOpenHelper` (строка 19) — закрытый вложенный класс, расширяющий абстрактный класс `SQLiteOpenHelper`; такие классы используются для управления созданием, открытием и обновлением баз данных (возможно, с модификацией структуры базы). Класс `SQLiteOpenHelper` более подробно рассматривается в листинге 8.46.

Листинг 8.39. Команда `package`, команды `import` и переменные экземпляров класса `DatabaseConnector`

```

1 // DatabaseConnector.java
2 // Класс упрощает подключение к базе данных и создание
// базы данных UserContacts.
3 package com.deitel.addressbook;

```

```
4 import android.content.ContentValues;
5 import android.content.Context;
6 import android.database.Cursor;
7 import android.database.SQLException;
8 import android.database.sqlite.SQLiteDatabase;
9 import android.database.sqlite.SQLiteOpenHelper;
10 import android.database.sqlite.SQLiteDatabase.CursorFactory;
11 import android.database.sqlite.SQLiteDatabase.CursorFactory;
12
13 public class DatabaseConnector
14 {
15     // Имя базы данных
16     private static final String DATABASE_NAME = "UserContacts";
17
18     private SQLiteDatabase database; // Для взаимодействия с базой данных
19     private DatabaseOpenHelper databaseOpenHelper; // Создание базы данных
20 }
```

Конструктор и методы open и close класса DatabaseConnector

Конструктор `DatabaseConnection` (листинг 8.40, строки 22–27) создает новый объект класса `DatabaseOpenHelper` (листинг 8.46), который будет использоваться для открытия или создания базы данных. Конструктор `DatabaseOpenHelper` более подробно рассматривается в листинге 8.46. Метод `open` (строки 30–34) пытается установить подключение к базе данных и, если попытка завершается неудачей, выдает исключение `SQLException`. Метод `getWritableDatabase` (строка 33), унаследованный от `SQLiteOpenHelper`, возвращает объект `SQLiteDatabase`. Если база данных не была создана ранее, этот метод создает ее; в противном случае метод открывает ее. После успешного открытия база данных будет кэшироваться операционной системой для улучшения производительности последующих операций с базой данных. Метод `close` (строки 37–41) закрывает подключение к базе данных вызовом унаследованного от `SQLiteOpenHelper` метода `close`.

Листинг 8.40. Конструктор и методы open и close класса DatabaseConnector

```
21     // Открытый конструктор DatabaseConnector
22     public DatabaseConnector(Context context)
23     {
24         // Создание нового объекта DatabaseOpenHelper
25         databaseOpenHelper =
26             new DatabaseOpenHelper(context, DATABASE_NAME, null, 1);
27     }
28
29     // Открытие подключения к базе данных
30     public void open() throws SQLException
31     {
32         // Создание или открытие базы данных для чтения/записи
33         database = databaseOpenHelper.getWritableDatabase();
34     }
35
36     // Закрытие подключения к базе данных
37     public void close()
```

```

38     {
39         if (database != null)
40             database.close(); // Закрытие подключения к базе данных
41     }
42

```

Метод insertContact класса DatabaseConnector

Метод `insertContact` (листинг 8.41) вставляет в базу данных новую запись с заданной информацией о контакте. Сначала каждый элемент данных контакта помещается в новый объект `ContentValues` (строки 47–54), который поддерживает ассоциативный массив пар «ключ—значение» — именами столбцов базы данных являются ключи. Строки 56–58 открывают базу данных, вставляют новый контакт и закрывают базу данных. Метод `insert` класса `SQLiteDatabase` (строка 57) вставляет значения из заданного объекта `ContentValues` в таблицу, заданную первым аргументом, — таблицу `"contacts"` в данном случае. Второй параметр метода, который в этом приложении не используется, называется `nullColumnHack`; его необходимость объясняется тем, что `SQLite` не поддерживает вставку в таблицу совершенно пустых строк — это было бы эквивалентно передаче `insert` пустого объекта `ContentValues`. Вместо того, чтобы запрещать передачу методу пустого объекта `ContentValues`, мы используем параметр `nullColumnHack` для обозначения столбца, допускающего значения `NULL`.

Листинг 8.41. Метод insertContact класса DatabaseConnector

```

43     // Вставка нового контакта в базу данных
44     public long insertContact(String name, String phone, String email,
45         String street, String city, String state, String zip)
46     {
47         ContentValues newContact = new ContentValues();
48         newContact.put("name", name);
49         newContact.put("phone", phone);
50         newContact.put("email", email);
51         newContact.put("street", street);
52         newContact.put("city", city);
53         newContact.put("state", state);
54         newContact.put("zip", zip);
55
56         open(); // Открытие базы данных
57         long rowID = database.insert("contacts", null, newContact);
58         close(); // Закрытие базы данных
59         return rowID;
60     } // Конец метода insertContact
61

```

Метод updateContact класса DatabaseConnector

Метод `updateContact` (листинг 8.42) аналогичен методу `insertContact`, за исключением того, что он вызывает метод `update` класса `SQLiteDatabase` (строка 76) для обновления существующего контакта. Третий аргумент метода `update` представляет условие SQL `WHERE` (без ключевого слова `WHERE`), определяющего обновляемые записи. В данном случае конкретный контакт определяется идентификатором записи.

Листинг 8.42. Метод updateContact класса DatabaseConnector

```

62 // Обновление существующего контакта в базе данных
63 public void updateContact(long id, String name, String phone,
64     String email, String street, String city, String state, String zip)
65 {
66     ContentValues editContact = new ContentValues();
67     editContact.put("name", name);
68     editContact.put("phone", phone);
69     editContact.put("email", email);
70     editContact.put("street", street);
71     editContact.put("city", city);
72     editContact.put("state", state);
73     editContact.put("zip", zip);
74
75     open(); // Открытие базы данных
76     database.update("contacts", editContact, "_id=" + id, null);
77     close(); // Закрытие базы данных
78 }
79

```

Метод getAllContacts

Метод `getAllContacts` (листинг 8.43) использует метод `query` класса `SQLiteDatabase` (строки 83–84) для получения курсора, предоставляющего доступ к идентификаторам и именам всех контактов в базе. Аргументы метода:

- ❑ имя таблицы, к которой обращен запрос;
- ❑ массив `String` с именами возвращаемых столбцов (`_id` и `name` в данном случае) — с `null` возвращаются все столбцы таблицы, что обычно делать не рекомендуется (для экономии памяти, процессорного времени и заряда батареи следует получать только реально необходимые данные);
- ❑ условие SQL `WHERE` (без ключевого слова `WHERE`) или `null` для получения всех записей;
- ❑ массив `String` с аргументами, которые должны подставляться в условие `WHERE` на место заполнителей `?`, или `null`, если аргументы в условии `WHERE` не используются;
- ❑ условие SQL `GROUP BY` (без ключевых слов `GROUP BY`) или `null`, если группировка результатов не используется;
- ❑ условие SQL `HAVING` (без ключевого слова `HAVING`) для определения групп из условия `GROUP BY`, включаемых в результат, — если условие `GROUP BY` содержит `null`, этот аргумент также должен содержать `null`;
- ❑ условие SQL `ORDER BY` (без ключевых слов `ORDER BY`) для определения порядка результатов или `null`, если упорядочение не используется.

Объект `Cursor`, возвращаемый методом `query`, содержит все строки таблицы, соответствующие аргументам методов (так называемый *результатирующий набор*). Курсор устанавливается перед первой строкой результирующего набора — для перемещения

курсора по результирующему набору для обработки данных используются различные методы класса Cursor.

Листинг 8.43. Метод getAllContacts класса DatabaseConnector

```
80    // Получение курсора со всеми именами контактов в базе данных
81    public Cursor getAllContacts()
82    {
83        return database.query("contacts", new String[] {"_id", "name"},
84            null, null, null, null, "name");
85    }
86
```

Метод getOneContact

Метод getOneContact (листинг 8.44) также использует метод query класса SQLiteDatabase для обращения к базе данных с запросом. На этот раз из базы данных извлекаются все столбцы данных для контакта с заданным идентификатором.

Листинг 8.44. Метод getOneContact класса DatabaseConnector

```
87    // Получение курсора с информацией о заданном контакте
88    public Cursor getOneContact(long id)
89    {
90        return database.query(
91            "contacts", null, "_id=" + id, null, null, null);
92    }
93
```

Метод deleteContact

Метод deleteContact (листинг 8.45) использует метод delete класса SQLiteDatabase (строка 98) для удаления контакта из базы данных. В данном случае из базы данных извлекаются все столбцы для контакта с заданным идентификатором. В трех аргументах передается таблица, из которой удаляется запись, условие WHERE (без ключевого слова WHERE) и при наличии аргументов у условия WHERE — строковый массив значений, подставляемых в условие WHERE (null в нашем случае).

Листинг 8.45. Метод deleteContact класса DatabaseConnector

```
94    // Удаление контакта, заданного строкой имени
95    public void deleteContact(long id)
96    {
97        open(); // Открытие базы данных
98        database.delete("contacts", "_id=" + id, null);
99        close(); // Закрытие базы данных
100    }
101
```

Закрытый вложенный класс DatabaseOpenHelper

Закрытый вложенный класс DatabaseOpenHelper (листинг 8.46) расширяет абстрактный класс SQLiteOpenHelper, который помогает приложениям создавать базы данных

и отслеживать изменения версий. Конструктор (строки 105–109) просто вызывает конструктор суперкласса, которому передаются четыре аргумента.

- ❑ Контекст, в котором создается или открывается база данных.
- ❑ Имя базы данных — может быть равно `null` при использовании базы данных в памяти.
- ❑ Используемый объект `CursorFactory` — `null` означает, что вы хотите использовать объект `CursorFactory`, предоставляемый `SQLite` по умолчанию (подходит для большинства приложений).
- ❑ Номер версии базы данных (начиная с 1).

Вы должны переопределить абстрактные методы `onCreate` и `onUpgrade` этого класса. Если база данных еще не существует, метод `onCreate` класса `DatabaseOpenHelper` будет вызван для ее создания. Если указать более новый номер версии, чем у базы данных, хранящейся на устройстве, будет вызван метод `onUpgrade` класса `DatabaseOpenHelper` для обновления базы данных до новой версии (например, с добавлением таблиц или добавлением столбцов в существующую таблицу).

Листинг 8.46. Класс DatabaseOpenHelper

```
102 private class DatabaseOpenHelper extends SQLiteOpenHelper
103 {
104     // Конструктор
105     public DatabaseOpenHelper(Context context, String name,
106         CursorFactory factory, int version)
107     {
108         super(context, name, factory, version);
109     }
110
111     // Создание таблицы contacts при создании базы данных
112     @Override
113     public void onCreate(SQLiteDatabase db)
114     {
115         // Запрос для создания таблицы с именем contacts
116         String createQuery = "CREATE TABLE contacts" +
117             "(_id integer primary key autoincrement," +
118             "name TEXT, phone TEXT, email TEXT, " +
119             "street TEXT, city TEXT, state TEXT, zip TEXT);";
120
121         db.execSQL(createQuery); // Выполнение запроса для создания базы данных
122     }
123
124     @Override
125     public void onUpgrade(SQLiteDatabase db, int oldVersion,
126         int newVersion)
127     {
128     }
129 } // Конец класса DatabaseOpenHelper
130 } // Конец класса DatabaseConnector
```

Метод `onCreate` (строки 112–122) задает таблицу, создаваемую командой SQL `CREATE TABLE`, которая определяется в формате `String` (строки 116–119). В этом случае таблица `contacts` содержит целочисленное поле первичного ключа (`_id`), значения которого увеличиваются автоматически, и текстовые поля для всех остальных столбцов. В строке 121 метод `execSQL` класса `SQLiteDatabase` используется для выполнения команды `CREATE TABLE`. Так как обновлять базу данных не нужно, метод `onUpgrade` просто переопределяется с пустым телом. Класс `SQLiteOpenHelper` также предоставляет метод `onDowngrade`, который используется для перехода к более старой версии базы данных (если версия хранимой базы выше, чем версия, указанная при вызове конструктора класса `SQLiteOpenHelper`). Эта возможность позволяет вернуться к предыдущей версии базы данных с меньшим количеством столбцов в таблице или таблиц в базе данных — например, для исправления ошибки в приложении.

У всех методов `SQLiteDatabase`, используемых в классе `DatabaseConnector`, существуют аналогичные методы, которые выполняют те же операции, но выдают исключения в случае неудачи вместо простого возвращения `-1` (например, `insertOrThrow` вместо `insert`). Так как эти методы взаимозаменяемы, разработчик может сам решить, как ему лучше организовать обработку ошибок чтения и записи.

8.10. Резюме

В этой главе мы создали приложение Address Book, с помощью которого пользователи могут добавлять, просматривать, изменять либо удалять информацию о контактах, которая хранится в базе данных SQLite. Мы определили пары «атрибут—значение» компонентов GUI в виде XML-ресурсов `style`, а затем применили стили ко всем компонентам, которые совместно используют эти значения, с помощью атрибута компонента `style`. Чтобы добавить границы к компонентам `TextView`, мы задали объект `Drawable` в качестве значения атрибута `android:background` компонента `TextView`, а также создали пользовательский объект `Drawable` с помощью XML-представления формы. Стандартные значки Android были использованы для улучшения внешнего вида команд меню приложения.

Вы узнали, что для фрагментов, предназначенных для отображения списка с возможностью прокрутки, можно расширить класс `ListFragment` для создания фрагмента, в макете которого по умолчанию присутствует компонент `ListView`. Эта возможность была использована для отображения контактов, хранящихся в базе данных приложения. Для связывания данных с приложением был использован объект `CursorAdapter`, который выдает результаты запроса к базе данных.

В активности этого приложения объекты `FragmentTransaction` использовались для добавления и динамической замены фрагментов в GUI. Стек возврата был использован для поддержки кнопки `Back`, возвращающей приложение к предыдущему фрагменту, и для программного возврата к предыдущим фрагментам из активности приложения.

Мы показали, как организовать передачу данных между фрагментами и управляющей активностью (или другими фрагментами) через интерфейсы методов обратного вызова, реализуемые управляющей активностью. Для передачи аргументов фрагментам использовались объекты `Bundle`.

Мы использовали субкласс `SQLiteOpenHelper` для упрощения создания базы данных и получения объекта `SQLiteDatabase`, с помощью которого выполняются операции с содержимым базы данных. Класс `Cursor` использовался для обработки результатов запроса. Субклассы `AsyncTask` применялись для выполнения задач базы данных за пределами потока GUI и возвращения результатов в поток GUI, благодаря чему можно использовать преимущества потоков Android без прямого создания и управления потоками.

В главе 9 рассматривается коммерческая сторона программирования Android-приложений. Вы узнаете, как подготовить приложение для отправки в Google Play, включая подготовку значков; как протестировать приложения на устройствах и опубликовать их в Google Play. Мы рассмотрим характеристики выдающихся приложений и рекомендации по дизайну приложений, которые следует соблюдать. В этой главе приводятся рекомендации по выбору цены и маркетингу приложений. Также будут упомянуты преимущества предоставления бесплатной версии для увеличения сбыта других продуктов — например, версии приложения с расширенной функциональностью или привилегированного контента. Вы научитесь использовать Google Play для получения данных о продажах, проведения платежей и других целей.

9

Google Play и коммерческие аспекты разработки

В этой главе...

- Подготовка приложений для публикации
- Выбор цены, преимущества платных и бесплатных приложений
- Заработка на рекламе в приложениях
- Продажа виртуальных товаров с использованием внутренних платежей
- Регистрация в Google Play
- Создание учетной записи Google Wallet
- Отправка приложений в Google Play
- Запуск Play Store из приложения
- Другие магазины приложений Android
- Другие популярные мобильные платформы, на которые можно портировать приложения для расширения круга пользователей
- Продвижение приложений на рынок

9.1. Введение

В главах 2–8 мы разработали несколько работоспособных приложений Android. После того как вы освоите разработку и тестирование ваших собственных приложений (как в эмуляторе, так и на устройствах Android), следующим шагом должна стать отправка в Google Play (и/или другие магазины приложений) для распространения их по всему миру. В этой главе вы узнаете, как зарегистрироваться в Google Play и создать учетную запись Google Wallet, чтобы вы могли продавать свои приложения. Вы узнаете, как подготовить приложения к публикации и как загрузить их в Google Play. В отдельных случаях мы будем давать ссылки на документацию Android вместо того, чтобы описывать конкретные действия, потому что они с большой вероятностью изменятся. Мы также расскажем о других магазинах приложений, в которых вы можете распространять свои приложения Android. Также будет рассмотрен вопрос о преимуществах платного и бесплатного распространения и перечислены такие механизмы извлечения прибыли, как реклама в приложениях и продажа виртуальных товаров. Мы представим некоторые ресурсы для продвижения приложений на рынок и упомянем о других платформах, на которые можно портировать приложения Android для расширения круга пользователей.

9.2. Подготовка приложений к публикации

В разделе «Preparing for Release» руководства Dev Guide (<http://developer.android.com/tools/publishing/preparing.html>) перечислены ключевые моменты, которые следует учесть перед публикацией приложения в Google Play. В их числе:

- ❑ Тестирование приложения на устройствах Android.
- ❑ Включение в приложение лицензионного соглашения End User License Agreement (этот шаг не обязательен).
- ❑ Добавление значка и метки в манифест приложения.
- ❑ Контроль версий приложения (например, 1.0, 1.1, 2.0, 2.3, 3.0).
- ❑ Получение криптографического ключа для включения цифровой подписи в приложение.
- ❑ Компиляция приложения.

Также перед публикацией приложения стоит ознакомиться с контрольным списком Launch Checklist (<http://developer.android.com/distribute/googleplay/publish/preparing.html>) и Tablet App Quality Checklist (<http://developer.android.com/distribute/googleplay/quality/tablet.html>).

9.2.1. Тестирование приложения

Прежде чем отправлять свое приложение в Google Play, тщательно протестируйте его на разных устройствах. Даже если приложение идеально работает на эмуляторе, при запуске его на конкретных устройствах Android могут возникнуть проблемы. Google Play Developer Console теперь предоставляет поддержку альфа- и бета-тестирования приложений в группах через Google+. За дополнительной информацией обращайтесь по адресу

<https://play.google.com/apps/publish/>

9.2.2. Лицензионное соглашение

У вас имеется возможность включить в приложение лицензионное соглашение EULA (End User License Agreement). Это соглашение определяет условия, на которых вы предоставляете лицензию на свое приложение пользователю. Обычно в нем указываются правила пользования, ограничения на повторное распространение и декомпиляцию, ответственность за продукт, информация о соблюдении соответствующего законодательства и т. д. Возможно, вам стоит проконсультироваться у адвоката при подготовке EULA для вашего приложения. Пример EULA можно просмотреть по адресу

<http://www.rocketlawyer.com/document/end-user-license-agreement.rl>

9.2.3. Значки и метки

Спроектируйте значок для своего приложения и предоставьте текстовую метку (имя), которое будет отображаться в Google Play и на устройстве пользователя. В качестве значка можно использовать логотип компании, графику из приложения или специально созданное изображение. Android Asset Studio предоставляет инструмент для создания значков приложений по адресу

<http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html>

Создайте версию значка для всех вариантов плотности экрана:

- xx-high (XXHDPI)**: 144×144 пикселя;
- x-high (XHDPI)**: 96×96 пикселов;
- high (HDPI)**: 72×72 пикселя;
- medium (MDPI)**: 48×48 пикселов.

Вам также понадобится значок высокого разрешения, который будет использоваться в Google Play. Он должен иметь следующие параметры:

- 512×512 пикселов;

- 32-разрядный формат PNG;
- Максимальный размер 1 Мбайт.

Значок определяет первое впечатление потенциального пользователя, поэтому очень важно создать качественные значки. Рассмотрите возможность привлечения опытного дизайнера, который поможет вам создать привлекательный, профессиональный значок. В табл. 9.1 перечислены некоторые фирмы, предлагающие бесплатные значки профессионального уровня и коммерческие услуги по разработке значков по заказу клиентов. Когда значок и метка будут готовы, включите их в файл `AndroidManifest.xml`, задав атрибуты `android:icon` и `android:label` элемента `application`.

Таблица 9.1. Фирмы, предоставляющие услуги по дизайну значков

Компания	URL-адрес	Услуги
glyphlab	http://www.glyphlab.com/icon_design/	Разработка значков по заказу, галерея значков для бесплатной загрузки
Androidicons	http://www.androidicons.com	Разработка значков по заказу, продажа набора из 200 значков по фиксированной цене, галерея значков для бесплатной загрузки
Iconiza	http://www.iconiza.com	Разработка значков по фиксированной цене, продажа готовых значков
Aha-Soft	http://www.aha-soft.com/icon-design.htm	Разработка значков по фиксированной цене
Rosetta®	http://icondesign.rosetta.com/	Разработка значков по заказу
Elance®	http://www.elance.com	Поиск дизайнеров, занимающихся разработкой значков

9.2.4. Контроль версии приложения

Разработчик должен включить в приложение *имя версии* (которое видят пользователи) и *код версии* (целый номер, используемый Google Play) и продумать стратегию нумерации обновлений. Например, первой версии приложения можно присвоить имя 1.0, незначительным обновлениям — имена 1.1 и 1.2, а следующему основному обновлению — 2.0. Код версии представляет собой целое число, которое обычно начинается с 1 и увеличивается на 1 для каждой новой версии приложения. Дополнительную информацию можно найти в разделе «Versioning Your Applications» по адресу

<http://developer.android.com/tools/publishing/versioning.html>

9.2.5. Лицензирование для управления доступом к платным приложениям

Сервис лицензирования Google Play позволяет создавать лицензионные политики, управляющие доступом к платным приложениям. Например, лицензионная политика позволяет ограничить количество устройств, на которых может быть установлено приложение. За дополнительной информацией о сервисе лицензирования обращайтесь по адресу

<http://developer.android.com/google/play/licensing/index.html>

9.2.6. Маскировка кода

Приложения, отправляемые в Google Play, следует «маскировать», чтобы помешать декомпиляции кода и обеспечить дополнительную защиту приложений. Бесплатная утилита ProGuard (которая запускается при построении окончательных версий) сокращает размер файла .apk (файл приложения Android, содержащий пакет для установки), оптимизирует и маскирует код «посредством удаления неиспользуемого кода и замены имен классов, полей и методов семантически несодержательными именами»¹. За информацией о настройке и использовании программы ProGuard обращайтесь по адресу

<http://developer.android.com/tools/help/proguard.html>

Дополнительная информация о защите приложений от пиратства посредством маскировки кода доступна по адресу

<http://www.techrepublic.com/blog/app-builder/protect-your-android-apps-with-obfuscation/1724>

9.2.7. Получение закрытого ключа для цифровой подписи

Прежде чем отправлять свое приложение на устройство, в Google Play или другой магазин, вы должны снабдить файл .apk цифровой подписью, которая будет идентифицировать вас как автора приложения. Цифровой сертификат включает имя разработчика или название компании, контактную информацию и т. д. Вы можете создать цифровую подпись самостоятельно с использованием закрытого ключа (например, надежного пароля, используемого для шифрования сертификата); покупать сертификат у сторонней сертифицирующей организации не обязательно (хотя такой вариант тоже возможен). Eclipse автоматически снабжает приложение цифровой подписью при его выполнении на эмуляторе или на устройстве в целях отладки. Этот цифровой сертификат не может использоваться в Google

¹ <http://developer.android.com/tools/help/proguard.html#enabling>

Play, и его срок действия истекает через 365 дней после создания. За подробными инструкциями о создании цифровых подписей обращайтесь к разделу «*Siging Your Applications*» по адресу

<http://developer.android.com/tools/publishing/app-signing.html>

9.2.8. Снимки экрана

Сделайте не менее двух снимков экрана вашего приложения, которые будут включены в его описание в Google Play (табл. 9.2), — всего можно отправить до восьми снимков для смартфона, 7-дюймового планшета и 10-дюймового планшета. Снимки дают предварительное представление о приложении, так как пользователь не сможет протестировать приложение перед загрузкой (хотя и может вернуть приложение с возвратом денег в течение 15 минут после покупки и загрузки). Постарайтесь выбрать привлекательные снимки, которые демонстрируют функциональность приложения.

Таблица 9.2. Параметры снимков экрана

Параметр	Описание
Размер	Минимальный размер — 320 пикселов, максимальный размер — 3840 пикселов (максимальный размер не должен превышать минимальный более чем вдвое)
Формат	24-разрядный формат PNG или JPEG без альфа-эффектов (прозрачности)
Изображение	Обрезка по краю без полей и границ

Программа Dalvik Debug Monitor Service (DDMS) устанавливается с плагином ADT для Eclipse и упрощает отладку приложений, работающих на физических устройствах; она также поможет с созданием снимков экрана на устройствах. Выполните следующие действия.

1. Запустите приложение на устройстве (см. завершающую часть раздела 1.9).
2. В среде Eclipse выполните команды **Window > Open Perspective > DDMS**, чтобы получить доступ к инструментам DDMS.

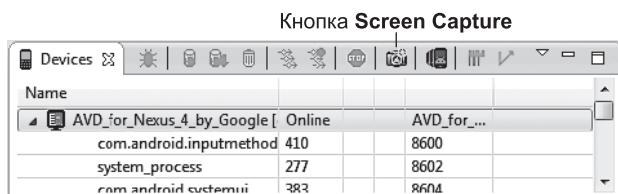


Рис. 9.1. Окно Devices в перспективе DDMS

3. В окне Devices выберите устройство, для которого будут делаться экранные снимки (рис. 9.1).
4. Нажмите Screen Capture. На экране появляется окно Device Screen Capture (рис. 9.2).

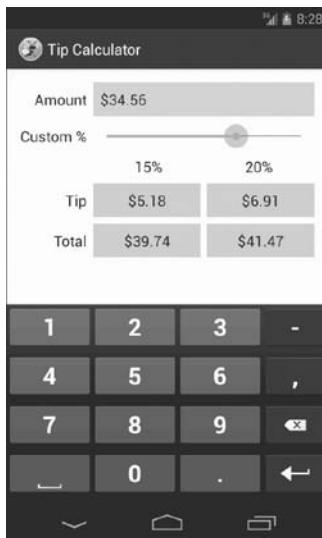


Рис. 9.2. Окно Device Screen Capture с экранным снимком приложения Tip Calculator (глава 3)

5. Убедитесь в том, что результат вас устраивает, и нажмите Save для сохранения полученного изображения.
6. Если вы захотите изменить содержимое экрана перед сохранением изображения, внесите изменения на устройстве и нажмите кнопку Refresh в окне Device Screen Capture window.

9.2.9. Информационный видеоролик

При отправке приложения в Google Play можно включить ULR короткого информационного видеоролика, размещенного на сайте YouTube. В табл. 9.3 приведены примеры. На некоторых видеороликах показаны люди, которые держат устройство и работают с приложением. В других роликах используются снимки экрана. В табл. 9.4 перечислены некоторые инструменты и сервисы для создания видеороликов и работы с ними (как бесплатные, так и коммерческие).

Чтобы отправить свой видеоролик, создайте или используйте существующую учетную запись YouTube. Нажмите Upload в правом верхнем углу страницы. Выберите

вариант Select files to upload, чтобы выбрать файл на своем компьютере, или просто перетащите видеоролик на страницу.

Таблица 9.3. Примеры информационных роликов для приложений в Google Play

Приложение	URL
Temple Run®: Oz	http://www.youtube.com/watch?v=QM9sT1ydtj0
GT Racing: Motor Academy	http://www.youtube.com/watch?v=2Z9OPICdgoA
Beach Buggy Blitz™	http://www.youtube.com/watch?v=YqDczawTsYw
Real Estate and Homes by Trulia®	http://www.youtube.com/watch?v=rLn697AszGs
Zappos.com®	http://www.youtube.com/watch?v=U-oNyK9kl_Q
Megopolis International	http://www.youtube.com/watch?v=JrqeEJ1xzCY

Таблица 9.4. Инструменты и сервисы для создания информационных видеороликов

Приложение	URL
Animoto	http://animoto.com
Apptamin	http://www.apptamin.com
Movie Maker for Microsoft Windows	http://windows.microsoft.com/en-us/windows-live/movie-maker
CamStudio™	http://camstudio.org
Jing	http://www.techsmith.com/jing.html
Camtasia Studio®	http://www.techsmith.com/camtasia.html
TurboDemo™	http://www.turbodemo.com/eng/index.php

9.3. Цена приложения: платное или бесплатное?

Разработчик сам определяет цену на приложения, распространяемые через Google Play. Очень часто разработчики предлагают бесплатные приложения, которые рекламируют платные версии этих же приложений с расширенным набором функций и возможностей. При этом они получают прибыль в результате продажи продуктов и услуг, версий тех же приложений с расширенной функциональностью, продажи дополнительного контента или от рекламы, встроенной в приложение. В табл. 9.5 перечислены способы монетизации приложений.

Таблица 9.5. Способы монетизации приложений

Способы монетизации приложений
Продажа приложения в Google Play
Продажа в других магазинах приложений Android
Продажа платных обновлений для приложения
Продажа виртуальных товаров (см. раздел 9.5)
Использование сервиса мобильной рекламы, встраиваемой в приложения (см. раздел 9.4)
Прямая продажа заказчикам рекламного места в приложении
Реклама версий приложения, обладающих расширенным набором функций

9.3.1. Платные приложения

Средняя цена приложения сильно зависит от категории. Например, по данным аналитического сайта AppBrain (<http://www.appbrain.com>), средняя цена игры-головоломки составляет \$1,54, а средняя цена бизнес-приложения — \$6.47.¹ И хотя на первый взгляд эти цены кажутся небольшими, следует учитывать то, что успешные приложения могут продаваться десятками, сотнями тысяч или даже миллионами копий.

Прежде чем установить цену на приложение, ознакомьтесь с ценами конкурентов. Сколько стоят продаваемые ими приложения? Предоставляют ли эти приложения аналогичный набор функций? Будут ли ваши приложения более «продвинутыми»? Предлагает ли ваше приложение те же возможности, что и приложения конкурентов, но по более низкой цене? Хотите ли вы вернуть потраченные на разработку приложения средства и получить прибыль?

Если ваша бизнес-стратегия изменится, со временем платные приложения даже могут перейти на модель бесплатного распространения. Тем не менее в настоящее время такой переход невозможен.

Финансовые операции при продаже платных приложений на Google Play выполняются с помощью учетной записи Google Wallet (<http://google.com/wallet>), хотя пользователи некоторых мобильных операторов (AT&T, Sprint и T-Mobile) могут воспользоваться биллинговыми системами этих операторов для получения средств от клиентов, загрузивших платные приложения. Начисление на счет Google Wallet осуществляется ежемесячно.² Разработчик несет ответственность за оплату налогов с прибыли, полученной с помощью Google Play.

¹ <http://www.appbrain.com/stats/android-market-app-categories>

² http://support.google.com/googleplay/android-developer/answer/137997?hl=en&ref_topic=15867

9.3.2. Бесплатные приложения

В настоящее время приблизительно 80% приложений для Android распространяется бесплатно, и они составляют подавляющее большинство загрузок.¹ Учитывая склонность пользователей к загрузке бесплатных приложений, предлагайте «упрощенную» версию приложения, которую пользователи смогут загрузить и опробовать бесплатно. Например, если вы написали игру, предложите версию с несколькими начальными уровнями. После прохождения бесплатных уровней на экране появится сообщение, в котором пользователю предлагается приобрести в Google Play коммерческую версию приложения с полным набором уровней. Также пользователю могут предлагаться дополнительные уровни приложения прямо из приложения (см. раздел 9.5). Согласно результатам исследований, которые проводились компанией AdMob, обновление «упрощенной» версии приложения — главный фактор приобретения платных приложений пользователями.²

Многие компании используют бесплатные приложения для продвижения своего товарного знака, а также продвижения других продуктов и услуг (табл. 9.6).

Таблица 9.6. Компании, использующие бесплатные приложения Android для продвижения своих брендов

Бесплатное приложение	Функциональность
Amazon® Mobile	Просмотр и приобретение товаров на сайте Amazon с мобильного устройства
Bank of America	Поиск банкоматов и отделений банка в вашем районе, проверка баланса и платежей
Best Buy®	Поиск и приобретение товаров на сайте Best Buy
CNN	Последние мировые новости, экстренные сообщения и видео в реальном времени
Epicurious Recipe	Тысячи кулинарных рецептов из журналов, издаваемых компанией Condé Nast, в том числе журналов Gourmet и Bon Appetit
ESPN® ScoreCenter	Настройка информационных панелей для отслеживания успехов ваших любимых спортивных команд (любителей и профессионалов)
NFL Mobile	Последние новости и обновления NFL, просмотр спортивных программ, отчеты NFL и другая информация

¹ <http://www.gartner.com/newsroom/id/2592315>

² <http://metrics.admob.com/wp-content/uploads/2009/08/AdMob-Mobile-Metrics-July-09.pdf>

Таблица 9.6 (окончание)

Бесплатное приложение	Функциональность
UPS® Mobile	Отслеживание почтовых отправлений, поиск утерянных посылок, примерная оценка стоимости почтовых отправлений и другой подобной информации
NYTimes	Чтение статей из журнала New York Times (бесплатно или за отдельную плату)
Pocket Agent™	Приложение от State Farm Insurance, с помощью которого можно связаться с агентом, заявить требование, найти местные отделения, получить информацию о банках State Farm и инвестиционных фондах и т. д.
Progressive® Insurance	Подача заявки и отправка фотографий с места происшествия, поиск локального агента, получение сведений о страховке при покупке нового автомобиля и ряда других сведений
USA Today®	Чтение статей из USA Today и получение сведений о результатах последних спортивных соревнований
Wells Fargo® Mobile	Мобильный поиск находящихся поблизости банкоматов и отделений банка, проверка баланса, проведение платежей и оплата счетов
Women's Health Workouts Lite	Различные физические упражнения из одного из популярных журналов для женщин

9.4. Монетизация приложений с помощью встроенной рекламы

Многие разработчики предлагают бесплатные приложения, доходы от которых обеспечиваются встроенной в приложение рекламой, — часто эта реклама имеет вид баннеров наподобие тех, что используются на веб-сайтах. Мобильные рекламные сети, такие как AdMob (<http://www.admob.com/>) и Google AdSense for Mobile (http://www.google.com/mobileads/publisher_home.html), собирают рекламные предложения от рекламодателей и отображают их в форме рекламы, актуальной для конкретного приложения (см. раздел 9.13). Прибыль от рекламы начисляется на основе количества просмотров. В первой сотне наиболее популярных бесплатных приложений прибыль от просмотра встроенной рекламы составляет от нескольких сотен до нескольких тысяч долларов в день. Конечно, встроенная реклама не будет столь доходной для большинства приложений, поэтому если вы хотите вернуть средства, потраченные на разработку приложения, и получить прибыль, продавайте приложение за деньги.

9.5. Внутренняя продажа виртуальных товаров

Реализованный в Google Play сервис внутренних продаж (<http://developer.android.com/google/play/billing/index.html>) позволяет продавать *виртуальные товары* (цифровой контент) с помощью приложений или устройств с Android 2.3 и выше (табл. 9.7). Согласно данным Google, приложения, использующие внутренние продажи, обеспечивают уровень прибыли, превышающий прибыль от продажи автономных платных приложений. Из 24 приложений, приносящих наибольшую прибыль на Google Play, 23 используют внутренние продажи.¹ Механизм внутренних продаж работает только в приложениях, приобретенных через Google Play; если же приложение продается через другие виртуальные магазины, этот сервис недоступен. Чтобы воспользоваться механизмом внутренних продаж, потребуется учетная запись, дающая право публикации в Google Play (см. раздел 9.6), и учетная запись продавца Google Wallet (см. раздел 9.7). Компания Google возвращает разработчику 70% выручки за покупки, сделанные в приложениях.

Продажа виртуальных товаров может принести большую прибыль (*из расчета на одного пользователя*), чем реклама². Среди приложений, особенно успешных в области продажи виртуальных товаров, стоит упомянуть Angry Birds, DragonVale, Zynga Poker, Bejeweled Blitz, NYTimes и Candy Crush Saga. Виртуальные товары особенно популярны в мобильных играх.

Таблица 9.7. Виртуальные товары

Подписки на журналы	Локализованные руководства	Аватары
Виртуальные предметы	Игровые уровни	Игровые сцены
Дополнительные функции	Рингтоны	Пиктограммы
Электронные открытки	Электронные подарки	Виртуальные деньги
Обои	Изображения	Виртуальные домашние животные
Аудиозаписи	Видеозаписи	Электронные книги

Чтобы включить поддержку внутренних покупок в приложение, обращайтесь к руководству по адресу

http://developer.android.com/google/play/billing/billing_integrate.html

¹ <http://android-developers.blogspot.com/2012/05/in-app-subscriptions-in-google-play.html>

² http://www.businessinsider.com/its-morning-in-venture-capital-2012-5?utm_source=readme&utm_medium=rightrail&utm_term=&utm_content=6&utm_campaign=recirc

За дополнительной информацией о механизме встроенной покупки (подписка, примеры приложений, рекомендации из области безопасности, тестирование и т. д.) обращайтесь по адресу

http://developer.android.com/google/play/billing/billing_overview.html

Вы также можете пройти бесплатный обучающий курс Selling In-app Products по адресу

<http://developer.android.com/training/in-app-billing/index.html>

Внутренние покупки в приложениях, проданных через альтернативные магазины

Если вы намереваетесь продавать приложения в других магазинах (см. раздел 9.11), обратитесь к независимым провайдерам мобильных платежей. У вас появится возможность встроить в приложения механизм покупки, реализованный с помощью библиотек соответствующих API (табл. 9.8), – использовать механизм внутренних покупок Google Play не удастся. Начните с создания дополнительной заблокированной функциональности (например, игровых уровней или аватаров). Если пользователь захочет что-либо приобрести, инструмент приобретения, встроенный в приложение, обрабатывает финансовую операцию и передает сообщение приложению, которое верифицирует платеж. Тогда приложение выполняет разблокировку дополнительной функциональности. Провайдеры мобильных платежей забирают себе от 25 до 45 % от стоимости приложения.

Таблица 9.8. Провайдеры мобильных платежей, используемых при покупках с помощью встроенной в приложения системы биллинга

Провайдер	URL	Описание
PayPal Mobile Payments Library	http://developer.paypal.com/webapps/developer/docs/classic/mobile/gs_MPL/	Пользователь щелкает на кнопке Pay with PayPal, входит в свою учетную запись PayPal, а затем щелкает на кнопке Pay
Amazon In-App Purchasing	http://developer.amazon.com/sdk/in-app-purchasing.html	Механизм внутренней покупки для приложений, продаваемых через магазин Amazon App Store for Android
Zong	http://www.zong.com/android	Отображает кнопку Buy для платежей, выполняемых одним щелчком мыши. Платежи включаются в счет телефонных услуг
Samsung In-App Purchase	http://developer.samsung.com/android/tools-sdks/In-App-Purchase-Library	Механизм внутренней покупки для приложений, спроектированных специально для устройств Samsung
Boku	http://www.boku.com	Пользователь щелкает на кнопке Pay by Mobile, вводит номер мобильного телефона, а затем завершает операцию, отвечая на текстовое сообщение, полученное на мобильный телефон

9.6. Регистрация в Google Play

Чтобы иметь возможность публиковать приложения в магазине Google Play, следует создать учетную запись на веб-странице <http://play.google.com/apps/publish>.

За регистрацию взимается однократный взнос \$25. В отличие от других мобильных платформ, Google Play *не требует одобрения загружаемых приложений*. Впрочем, разработчик должен соблюдать требования, изложенные в документе Google Play Developer Program Policies. Если приложение нарушает принципы этой политики, оно может быть удалено в любой момент. Повторяющиеся или серьезные нарушения положений этого документа могут привести к принудительному удалению учетной записи (табл. 9.9).

Таблица 9.9. Некоторые нарушения положений документа Google Play Content Policy for Developers (<http://play.google.com/about/developer-content-policy.html#showlanguages>)

Нарушение прав владельцев интеллектуальной собственности (товарные знаки, патенты и ко-пираты)	Нарушение неприкосновенности частной жизни пользователя
Нарушение закона	Препятствия в деятельности других сторон
Намеренный ввод пользователя в заблуждение относительно назначения приложения	Повреждение личных данных или устройства пользователя
Намеренные действия, направленные на рост расходов пользователя мобильной сети передачи данных или провайдера беспроводной сети	Азартные игры
Фальсификация или обман	Пропаганда ненависти и насилия
Поддержка контента порнографического или не-пристойного содержания либо любого другого контента, недопустимого для просмотра лицами до 18 лет	Реклама в виджетах и оповещениях системного уровня

9.7. Создание учетной записи Google Wallet

Для продажи приложений через магазин Google Play вам понадобится учетная запись сервиса Google Wallet, доступного разработчикам Google Play в 32 странах (табл. 9.10).¹ Google Wallet используется для проведения платежей в сетевых операциях. После регистрации и входа в Google Play (<http://play.google.com/apps/publish/>) щелкните на ссылке Financial Reports, а затем на ссылке Set up a merchant account. При создании учетной записи необходимо:

- ❑ предоставить закрытую информацию, по которой Google сможет связаться с вами;

¹ http://support.google.com/googleplay/android-developer/answer/150324?hl=en&ref_topic=15867

- предоставить контактную информацию для службы поддержки, по которой с вами смогут связаться пользователи;
- предоставить финансовую информацию, чтобы компания Google могла выполнить проверку кредитоспособности;
- согласиться с условиями обслуживания, которые описывают особенности обслуживания, допустимые и запрещенные действия, плату за обслуживание, сроки платежей и другие условия.

Таблица 9.10. Страны, в которых поддерживаются учетные записи Google Wallet

Австралия	Дания	Нидерланды	Тайвань
Австрия	Израиль	Новая Зеландия	Финляндия
Аргентина	Индия	Норвегия	Франция
Бельгия	Ирландия	Польша	Чешская Республика
Бразилия	Испания	Португалия	Швейцария
Великобритания	Италия	Россия	Швеция
Германия	Канада	Сингапур	Южная Корея
Гонконг	Мексика	США	Япония

Google Wallet обеспечивает обработку платежей и защиту от недобросовестных покупок. На продажи через Google Play не распространяются стандартные платежи за обработку.¹ Google выплачивает 70% от цены приложений. После создания учетной записи Google Wallet вы сможете использовать ее для других операций, помимо продажи приложений (например, для покупок в магазинах, принимающих участие в программе).

9.8. Отправка приложений в Google Play

Как только будут подготовлены все файлы и вы будете готовы к загрузке приложения, выполните действия, описанные в контрольном списке по адресу <http://developer.android.com/distribute/googleplay/publish/preparing.html>.

Подключитесь к Google Play по адресу <http://play.google.com/apps/publish> (см. раздел 9.6) и нажмите Publish an Android App on Google Play, чтобы начать процесс загрузки. Вам будет предложено отправить следующие компоненты.

- Пакетный файл приложения (.apk), содержащий файлы с кодом, ресурсы и файл манифеста.

¹ <http://checkout.google.com/termsOfService?type=SELLER>

- ❑ Не менее двух экранных снимков приложения, которые будут включены в описание в Google Play. Вы можете включить снимки для телефонов на базе Android, а также 7- и 10-дюймовых планшетов.
- ❑ Значок приложения в высоком разрешении (512×512 пикселов), который будет размещен в Google Play.
- ❑ Рекламная графика для Google Play, которая может использоваться компанией Google, если она решит продвигать ваше приложение (не обязательно). Такие изображения должны иметь размеры 180 пикселов в ширину и 120 пикселов в высоту, в 24-разрядном формате PNG или JPEG без эффектов прозрачности. Также изображение должно быть обрезано по краю (то есть доходить до края без дополнительных полей).
- ❑ Видеоролик, размещаемый в Google Play (не обязательно). Можно приложить URL-адрес видеоролика приложения (например, ссылку на видеоролик в YouTube с демонстрацией работы приложения).

Также вам будет предложено сообщить дополнительную информацию о приложении:

1. **Язык.** По умолчанию, описание приложения отображается на английском языке. Если нужно выводить сведения о приложении на дополнительных языках, выберите их из предоставленного списка (табл. 9.11).
2. **Заголовок** приложения, отображаемый на Google Play (не более 30 символов). Этот заголовок *не обязан* быть уникальным для каждого приложения Android.

Таблица 9.11. Языки описаний приложений в Google Play

Амхарский	Английский (Великобритания)	Английский (США)	Арабский	Африкаанс
Белорусский	Венгерский	Вьетнамский	Голландский	Греческий
Датский	Зулусский	Иврит	Индонезийский	Испанский (Испания)
Испанский (Латинская Америка)	Испанский (США)	Итальянский	Каталанский	Китайский (упрощенный)
Китайский (традиционный)	Корейский	Латвийский	Литовский	Малайский
Немецкий	Норвежский	Персидский	Польский	Португальский (Бразилия)
Португальский (Португалия)	Ретороманский	Румынский	Русский	Сербский
Словацкий	Словенский	Суахили	Тайский	Турецкий
Украинский	Филиппинский	Финский	Французский	Хинди
Хорватский	Чешский	Шведский	Эстонский	Японский

3. **Описание** приложения и его свойств (максимум 4000 символов). В последнем разделе описания рекомендуется обосновать обязательность каждой привилегии и продемонстрировать, каким образом она используется.
4. **Последние изменения.** Сводка изменений в последней версии приложения (максимум 500 символов).
5. **Промотекст.** Рекламный текст, используемый для маркетинга приложения (максимум 80 символов).
6. **Тип приложения.** Выберите тип: приложение (Application) или игра (Games).
7. **Категория.** Выберите категорию (см. табл. 1.9), которая соответствует вашей игре или другому приложению.
8. **Цена.** По умолчанию выбирается бесплатное распространение (Free). Чтобы продавать приложение, вам придется создать учетную запись Google Wallet.
9. **Рейтинг.** Выберите ограничения по возрасту пользователей. За дополнительной информацией обращайтесь к разделу «Rating your application content for Google Play» по адресу <http://support.google.com/googleplay/android-developer/answer/188189>.
10. **Географические ограничения.** По умолчанию, приложение будет доступно во всех странах, поддерживаемых Google Play в настоящее время и в будущем. При желании вы можете выбрать конкретные страны, в которых приложение должно быть доступно.
11. **Сайт.** В описание вашего приложения в Google Play будет включена ссылка Visit Developer's Website. Предоставьте прямую ссылку на страницу сайта, на которой пользователи, заинтересовавшиеся вашим приложением, смогут найти дополнительную информацию, рекламный текст, краткое описание, дополнительные снимки экранов, инструкции и т. д.
12. **Электронная почта.** В Google Play также будет включен ваш адрес электронной почты, чтобы покупатели могли обратиться к вам с вопросами, сообщениями об ошибках и т. д.
13. **Телефон.** Это поле лучше оставлять пустым, если только вы не организуете службу поддержки по телефону. Также номер службы поддержки можно разместить на сайте.

9.9. Запуск Play Store из приложения

Для повышения продаж приложений предоставьте пользователю возможность запускать приложение Play Store (Google Play) непосредственно из приложения (обычно с помощью кнопки, отображаемой на экране приложения). В результате пользователь получит возможность загружать другие опубликованные вами приложения либо приобрести приложение с более широким набором функций, чем

у загруженной «упрощенной» версии. Можно также предоставить пользователю возможность загрузки последних обновлений приложения через Play Store.

Существует два способа запуска приложения Play Store. Во-первых, можно вывести результаты поиска в Google Play приложений по таким критериям, как имя разработчика, имя пакета или строка символов. Например, если нужно стимулировать пользователей загружать другие ваши приложения, опубликованные на Google Play, включите соответствующую кнопку, отображаемую на экране приложения. Как только пользователь нажмет эту кнопку, запустится приложение Play Store, которое инициирует поиск приложений, включающих ваше имя или название компании. Во-вторых, отобразить страницу с информацией о соответствующем приложении на экране приложения Play Store.

За информацией о том, как запустить Play Store из вашего приложения, обращайтесь к разделу «Linking Your Products» по адресу

<http://developer.android.com/distribute/googleplay/promote/linking.html>

9.10. Управление приложениями в Google Play

С помощью консоли разработчика Google Play Developer Console можно управлять учетной записью и приложениями, проверять пользовательский рейтинг вашего приложения (от 0 до 5 звездочек), реагировать на комментарии пользователей, отслеживать количество общих и активных установок приложения (разность между количеством установок и удалений приложения). Можно также ознакомиться со статистикой установок и загрузок копий приложения для различных версий Android, устройств и прочей информацией. В отчетах об ошибках содержится информация о сбоях и «зависаниях», полученная от пользователей. Если вы разработали обновление приложения, то с минимальными усилиями можете опубликовать новую версию. Можно также удалить приложение из Play Store, но если пользователи заблаговременно загрузили его, оно останется на пользовательских устройствах. Пользователи, которые удалили приложение, смогут установить его повторно даже после удаления (приложение остается на серверах Google до тех пор, пока не будет удалено за нарушение правил использования, изложенных в документе Terms of Service).

9.11. Другие магазины приложений Android

Помимо Google Play, приложения можно распространять с помощью других магазинов приложений (табл. 9.12) или даже с помощью вашего собственного веб-сайта при участии таких служб, как AndroidLicenser (*<http://www.androidlicenser.com>*). За дополнительной информацией о публикации приложений на сторонних сайтах обращайтесь по адресу

http://developer.android.com/tools/publishing/publishing_overview.html

Таблица 9.12. Другие магазины приложений Android

Магазин	URL
Amazon Appstore	http://developer.amazon.com/welcome.html
Opera Mobile Store	http://apps.opera.com/en_us/index.php
Moborobo	http://www.moborobo.com
Appitalism®	http://www.appitalism.com/index.html
Samsung Apps	http://apps.samsung.com/mars/main/getMain.as
GetJar	http://www.getjar.com
SlideMe	http://www.slideme.org
Handango	http://www.handango.com
Mplayit™	http://www.mplayit.com
AndroidPit	http://www.androidpit.com

9.12. Другие популярные платформы мобильных приложений

По данным ABI Research, в 2013 году пользователи загрузят 56 миллиардов приложений для смартфонов и 14 миллиардов приложений для планшетов.¹ Портривание приложений Android на другие мобильные платформы, прежде всего iOS (устройства iPhone, iPad и iPod Touch), позволяет расширить круг потенциальных пользователей (табл. 9.13). Приложения Android могут разрабатываться на компьютерах с системой Windows, Linux и Mac с Java — одним из самых распространенных языков программирования. Однако приложения iOS должны разрабатываться на компьютерах Macintosh, которые стоят дороже, а для разработки применяется язык программирования Objective-C, который знаком небольшому числу разработчиков. Компания Google разработала утилиту с открытым кодом J2ObjC, которая помогает преобразовать код приложений Java на язык Objective-C для приложений iOS. За дополнительной информацией обращайтесь по адресу <http://code.google.com/p/j2objc/>.

Таблица 9.13. Популярные платформы мобильных приложений (<http://www.abiresearch.com/press/android-will-account-for-58-of-smartphone-app-down>)

Платформа	URL	Мировая доля рынка загрузок приложений
Android	http://developer.android.com	58% для смартфонов 17% для планшетов

¹ <http://www.abiresearch.com/press/android-will-account-for-58-of-smartphone-app-down>.

Платформа	URL	Мировая доля рынка загрузок приложений
iOS (Apple)	http://developer.apple.com/ios	33% для смартфонов 75% для планшетов
Windows Phone 8	http://developer.windowsphone.com	4% для смартфонов 2% для планшетов
BlackBerry (RIM)	http://developer.blackberry.com	3% для смартфонов
Amazon Kindle	http://developer.amazon.com	4% для планшетов

9.13. Маркетинг приложения

После публикации приложения представьте его аудитории будущих пользователей.¹ И в этом вам поможет *вирусный маркетинг* с помощью сайтов социальных сетей, таких как Facebook, Twitter и YouTube. Эти сайты имеют огромную аудиторию. По данным Pew Research Center, 72% взрослых пользователей Интернета используют социальные сети, а 67% из них представлены на Facebook.² В табл. 9.14 представлены наиболее популярные сайты социальных сетей. Зачастую в качестве недорогих и эффективных средств маркетинга используются рассылки по электронной почте и электронные бюллетени новостей.

Таблица 9.14. Популярные сайты социальных сетей

Название	URL	Описание
Facebook	http://www.facebook.com	Социальная сеть
Twitter	http://www.twitter.com	Микроблоги, социальная сеть
Google+	http://plus.google.com	Социальная сеть
Groupon	http://www.groupon.com	Выгодные предложения
Foursquare	http://www.foursquare.com	Публикация отметок геопозиционирования
Pinterest	http://www.pinterest.com	Публикация фотографий
YouTube	http://www.youtube.com	Видеоролики
LinkedIn	http://www.linkedin.com	Социальные сети для бизнеса
Flickr	http://www.flickr.com	Публикация фотографий

¹ Тема маркетинга приложений Android рассматривается в книге «Android Apps Marketing: Secrets to Selling Your Android App» Джейфри Хьюза (Jeffrey Hughes).

² <http://pewinternet.org/Commentary/2012/March/Pew-Internet-Social-Networking-full-detail.aspx>

Facebook

Facebook, один из самых популярных сайтов социальных сетей, имеет более 1 миллиарда активных пользователей¹ и свыше 150 миллиардов дружеских связей². Это превосходный ресурс *вирусного маркетинга* (путем распространения слухов). Начните с создания официальной страницы Facebook для вашего приложения или компании. Используйте эту страницу для публикации информации о приложении, новостей, обновлений, обзоров, рекомендаций, снимков экрана, рекордных счетов в играх, мнений пользователей и ссылок Google Play для загрузки приложения. Например, мы публикуем новости и обновления о своих книгах на странице Facebook по адресу <http://www.facebook.com/DeitelFan>.

Теперь, когда вы наслышаны о популярности Фейсбука, приступайте к распространению слухов. Попросите ваших коллег и друзей «лайкнуть» вашу страницу на Фейсбуке и порекомендовать своим друзьям и знакомым сделать то же самое. По мере того как пользователи будут посещать вашу страницу, заметки о впечатлениях будут появляться на страницах новостей друзей этих пользователей, способствуя росту армии поклонников вашего приложения.

Twitter

Twitter – это сайт микроблогов и социальных сетей, количество активных зарегистрированных пользователей которого превышает 554 миллиона.³ Пользователи публикуют «твиты» – сообщения длиной до 140 символов. Twitter предоставляет доступ к твитам всем последователям (на время написания книги одна известная рок-звезда имела более 40 млн последователей). Многие пользователи используют Твиттер для получения информации о новостях, поэтому можно рассыпать твиты, содержащие сведения о приложении (в том числе анонсы новых версий, советы, факты, комментарии для пользователей и другую информацию). Также посоветуйте друзьям и коллегам разослать твиты, содержащие сведения о приложении. Воспользуйтесь *хеш-тегом* (#) для создания ссылки на сообщение. Например, в процессе рассылки твитов, содержащих информацию о нашей книге, в нашей подписке на новости Twitter @deitel мы используем хеш-тег #AndroidFP2. Другие пользователи могут воспользоваться этим хеш-тегом, чтобы добавить комментарии о книге. Это облегчит поиск твитов для сообщений, связанных с книгой.

Вирусные видеоролики

Еще один способ распространения сведений о приложении – вирусные видеоролики, размещенные на сайтах публикации видеороликов в Интернете (YouTube, Bing Videos, Yahoo! Video), сайтах социальных сетей (например, Facebook, Twitter, Google+) или путем рассылки сообщений по электронной почте. Если вы создадите

¹ <http://investor.fb.com/releasedetail.cfm?ReleaseID=761090>

² <http://expandedramblings.com/index.php/by-the-numbers-17-amazing-facebook-stats/>

³ <http://www.statisticbrain.com/twitter-statistics/>

эффектный видеоролик — юмористический или даже возмутительный, — это будет способствовать быстрому росту популярности вашего приложения среди пользователей социальных сетей.

Рассылки по электронной почте

Для продвижения приложения можно воспользоваться рассылками новостей по электронной почте. Добавьте ссылки на Google Play, с помощью которых пользователи смогут загрузить приложение. Также добавьте ссылки на страницы в социальных сетях, на которых можно получить новейшую информацию о приложении.

Обзоры приложений

Свяжитесь с влиятельными блоггерами и сайтами, на которых публикуются обзоры приложений (табл. 9.15), и сообщите сведения о своем приложении. Предоставьте им промокод для бесплатной загрузки приложения (см. раздел 9.3). Учтите, что подобные сайты получают большое количество запросов, поэтому будьте кратки и предельно информативны. Многие обозреватели приложений публикуют видеообзоры приложений на YouTube и других подобных сайтах (табл. 9.16).

Таблица 9.15. Сайты, публикующие обзоры приложений Android

Название сайта	URL
Android Tapp™	http://www.androidtapp.com
Applicious™	http://www.androidapps.com
AppBrain	http://www.appbrain.com
AndroidZoom	http://www.androidzoom.com
Appstorm	http://android.appstorm.net
Best Android Apps Review	http://www.bestandroidappsreview.com
Android App Review Source	http://www.androidappreviewsource.com
Androinica	http://www.androinica.com
AndroidLib	http://www.androlib.com
Android and Me	http://www.androidandme.com
AndroidGuys	http://www.androidguys.com/category/reviews
Android Police	http://www.androidpolice.com
AndroidPIT	http://www.androidpit.com
Phandroid	http://www.phandroid.com

Таблица 9.16. Сайты с видеообзорами приложений Android

Название сайта	URL
Daily App Show	http://dailyappshow.com
Crazy Mike's Apps	http://crazymikesapps.com
Applicious™	http://www.appvee.com/?device_filter=android
Life of Android™	http://www.lifeofandroid.com/video/
Android Video Review	http://www.androidvideoreview.net/

Связи с интернет-общественностью

В индустрии связей с общественностью применяются средства массовой информации, предназначенные для облегчения передачи сообщений от компаний конечным пользователям. С появлением феномена, известного под названием Web 2.0, специалисты по связям с общественностью включают блоги, подкасты, RSS-подписки и социальные сети в свои PR-компании. В табл. 9.17 перечислены некоторые платные и бесплатные интернет-ресурсы, посвященные связям с общественностью, включая файлы распространения пресс-релизов, службы создания пресс-релизов и другие ресурсы.

Таблица 9.17. PR-ресурсы в Интернете

Ресурс	URL	Описание
Бесплатные		
PRWeb®	http://www.prweb.com	Сервис публикации пресс-релизов (бесплатно или на платной основе)
ClickPress™	http://www.clickpress.com	Отправьте на этот сайт свои новости для одобрения (бесплатно или на платной основе). Одобренные новости будут доступны на сайте ClickPress, а также на сайтах новостей. За плату ClickPress будет распространять пресс-релизы по глобальным каналам финансовых новостей
PRLog	http://www.prlog.org/pub/	Бесплатная отправка и распространение пресс-релизов
i-Newswire	http://www.i-newswire.com	Бесплатная отправка и распространение пресс-релизов
openPR®	http://www.openpr.com	Бесплатная публикация пресс-релизов
Платные		
PR Leap	http://www.prleap.com	Платная служба рассылки пресс-релизов в Интернете

Ресурс	URL	Описание
Marketwire	http://www.marketwire.com	Платная служба рассылки пресс-релизов, которая позволяет выбирать целевую аудиторию по различным признакам: географический, отраслевой и т. д.
Mobility PR	http://www.mobilitypr.com	Службы, поддерживающие связи с общественностью компаний, работающих в индустрии мобильных устройств
Press Release Writing	http://www.press-release-writing.com	Службы рассылки пресс-релизов, которые предлагают ряд дополнительных услуг: чтение, проверка на наличие ошибок и редактирование пресс-релизов и т. д. Ознакомьтесь с рекомендациями по написанию эффективных пресс-релизов

Мобильная реклама

Приобретение рекламы (например, в других приложениях, в Интернете, в газетах и журналах либо на радио и телевидении) — это еще один способ маркетинга приложений. Мобильные рекламные сети (табл. 9.18) специализируются на рекламе мобильных приложений, разработанных для платформы Android (и других платформ). Многие из мобильных рекламных сетей могут выбирать целевую аудиторию по местоположению, провайдеру, устройству (например, Android, iPhone, BlackBerry и другие устройства) и другим категориям. Имейте в виду, что прибыль от большинства приложений невелика, поэтому не тратьте много денег на рекламу.

Таблица 9.18. Мобильные рекламные сети

Мобильная рекламная сеть	URL
AdMob (Google)	http://www.admob.com/
Medialets	http://www.medialets.com
Tapjoy®	http://www.tapjoy.com
Nexage™	http://www.nexage.com
Jumptap®	http://www.jumptap.com
Smaato®	http://www.smaato.com
mMedia™	http://mmedia.com
InMobi™	http://www.inmobi.com
Flurry™	http://www.flurry.com

Рекламные сети можно также использовать для монетизации бесплатных приложений путем включения в их состав рекламы (баннеров, видеороликов и т. д.).

По данным отчета Opera State of Mobile Advertising¹, средний показатель eCPM (effective cost per 1000 impressions, эффективная стоимость за тысячу показов) для рекламы в приложениях Android равен \$0,88 (хотя среднее значение может зависеть от сети, устройства и т. д.). Большая часть платежей за рекламу на платформе Android основана на рейтинге «кликальности» (CTR, clickthrough rate), а не на количестве генерируемых показов. По данным отчета Jumptap, средний показатель CTR для встроенной рекламы в мобильных приложениях составляет 0,65%², хотя и это значение зависит от приложения, устройства, специализации рекламной сети и т. д. Если у вашего приложения много пользователей и показатели CTR для рекламы в нем высоки, доходы от рекламы могут быть довольно значительными. Кроме того, рекламная сеть может поставлять более высокооплачиваемую рекламу, увеличивая вашу прибыль.

9.14. Резюме

В этой главе был рассмотрен процесс регистрации на Google Play и настройки учетной записи Google Wallet, необходимой для продажи приложений. Мы показали, как подготовить приложение к публикации на Google Play, включая тестирование на эмуляторе и устройствах Android, создание значков и меток, а также редактирование файла `AndroidManifest.xml`. Вы узнали, как загрузить приложение на Google Play, и познакомились с альтернативными магазинами, в которых можно продавать приложения. Вашему вниманию были предложены рекомендации по формированию цен на приложения и списки ресурсов с описаниями монетизации приложений с помощью встроенной в них рекламы и продаж виртуальных товаров с помощью приложений. Также были описаны ресурсы, которые могут использоваться для маркетинга приложений, распространяемых через Google Play.

¹ <http://www.insidemobileapps.com/2012/12/14/ios-leads-the-pack-in-ecpm-traffic-and-revenue-on-operas-mobile-ad-platform-ipad-average-ecpm-of-4-42/>

² <http://paidcontent.org/2012/01/05/419-jumptap-android-the-most-popular-but-ios-still-more-interactive-for-ads/>

П. Дейтел, Х. Дейтел, Э. Дейтел

Android для разработчиков

Перевел с английского Е. Матвеев

Заведующий редакцией

П. Щеголев

Ведущий редактор

Ю. Сергиенко

Художник

С. Заматевская

Корректоры

С. Беляева, М. Рошаль

Верстка

Л. Соловьева

ООО «Питер Пресс», 192102, Санкт-Петербург, ул. Андреевская (д. Волкова), 3, литер А, пом. 7Н.

Налоговая льгота — общероссийский классификатор продукции ОК 034-2014, 58.11.12.000 — Книги печатные
профессиональные, технические и научные.

Подписано в печать 06.03.15. Формат 70x100/16. Усл. п. л. 30,960. Тираж 1200. Заказ 0000.

Отпечатано в ОАО «Первая Образцовая типография» Филиал «Чеховский Печатный Двор»
142300, Московская область, г. Чехов, ул. Полиграфистов, 1.

Сайт: www.chpk.ru. E-mail: marketing@chpk.ru
факс: 8(496) 726-54-10, телефон: (495) 988-63-87



ПРЕДСТАВИТЕЛЬСТВА ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР»
предлагают профессиональную и популярную литературу по различным
направлениям: история и публицистика, экономика и финансы, менеджмент
и маркетинг, компьютерные технологии, медицина и психология.

РОССИЯ

Санкт-Петербург: м. «Выборгская», Б. Сампсониевский пр., д. 29а
тел./факс: (812) 703-73-73, 703-73-72; e-mail: sales@piter.com

Москва: м. «Электрозводская», Семеновская наб., д. 2/1, стр. 1
тел./факс: (495) 234-38-15; e-mail: sales@msk.piter.com

Воронеж: тел.: 8 951 861-72-70; e-mail: voronej@piter.com

Екатеринбург: ул. Бебеля, д. 11а
тел./факс: (343) 378-98-41, 378-98-42; e-mail: office@ekat.piter.com

Нижний Новгород: тел.: 8 960 187-85-50; e-mail: nnovgorod@piter.com

Новосибирск: Комбинатский пер., д. 3
тел./факс: (383) 279-73-92; e-mail: sib@nsk.piter.com

Ростов-на-Дону: ул. Ульяновская, д. 26
тел./факс: (863) 269-91-22, 269-91-30; e-mail: piter-ug@rostov.piter.com

Самара: ул. Молодогвардейская, д. 33а, офис 223
тел./факс: (846) 277-89-79, 229-68-09; e-mail: samara@piter.com

УКРАИНА

Киев: Московский пр., д. 6, корп. 1, офис 33
тел./факс: (044) 490-35-69, 490-35-68; e-mail: office@kiev.piter.com

Харьков: ул. Сузdalские ряды, д. 12, офис 10
тел./факс: (057) 7584145, +38 067 545-55-64; e-mail: piter@kharkov.piter.com

БЕЛАРУСЬ

Минск: ул. Розы Люксембург, д. 163
тел./факс: (517) 208-80-01, 208-81-25; e-mail: minsk@piter.com

✎ Издательский дом «Питер» приглашает к сотрудничеству зарубежных торговых партнеров или посредников, имеющих выход на зарубежный рынок
тел./факс: (812) 703-73-73; e-mail: spb@piter.com

✎ Издательский дом «Питер» приглашает к сотрудничеству авторов
тел./факс издательства: (812) 703-73-72, (495) 974-34-50

✎ Заказ книг для вузов и библиотек
тел./факс: (812) 703-73-73, доб. 6250; e-mail: uchebnik@piter.com

✎ Заказ книг по почте: на сайте www.piter.com; по тел.: (812) 703-73-74, доб. 6225
