

Глава 8. Макеты	65
8.1. LayoutParams	65
8.2. Gravity и Gravity макета	65
8.3. Поведение прокрутки CoordinatorLayout	68
8.4. Процентные макеты	70
8.5. Вес вида	72
8.6. Создание LinearLayout программным способом	73
8.7. LinearLayout	75
8.8. RelativeLayout	75
8.9. FrameLayout	76
8.10. GridLayout	78
8.11. CoordinatorLayout	79
8.12. ConstraintLayout	80
Глава 9. ConstraintLayout	81
9.1. Добавление ConstraintLayout в проект	82
9.2. Цепочки (Chains)	82
Глава 10. TextInputLayout	83
10.1. Базовое использование	83
10.2. Переключатели видимости пароля	83
10.3. Добавление подсчета символов	84
10.4. Обработка ошибок	84
10.5. Настройка внешнего вида TextInputLayout	85
10.6. TextInputEditText	86
Глава 11. CoordinatorLayout и поведение	86
11.1. Создание простого поведения	86
11.2. Использование поведения SwipeDismissBehavior	87
11.3. Создание зависимостей между представлениями	88
Глава 12. TabLayout	88
12.1. Использование TabLayout без ViewPager	88
Глава 13. ViewPager	89
13.1. ViewPager с точечным индикатором	89
13.2. Базовое использование ViewPager с фрагментами	91
13.3. ViewPager с фрагментом PreferenceFragment	93
13.4. Добавление ViewPager	94
13.5. Настройка OnPageChangeListener	95
13.6. ViewPager с TabLayout	95
Глава 14. CardView	97
14.1. Начало работы с CardView	98
14.2. Добавление пульсирующей анимации	99
14.3. Настройка CardView	99
14.4. Анимирование цвета фона CardView с помощью TransitionDrawable	100
Глава 15. NavigationView	100
15.1. Как добавить NavigationView	100
15.2. Добавить подчеркивание в элементы меню	105
15.3. Добавление разделителей в меню	106
15.4. Добавление разделителя меню с использованием стандартного DividerItemDecoration	107

Глава 16. RecyclerView	108
16.1. Добавление RecyclerView	108
16.2. Более плавная загрузка элементов	109
16.3. RecyclerView с привязкой данных	110
16.4. Анимация изменения данных	112
16.5. Всплывающее меню с RecyclerView	116
16.6. Использование нескольких ViewHolders с ItemType	118
16.7. Фильтрация элементов внутри RecyclerView с помощью SearchView	119
16.8. Перетаскивание и пролистывание с помощью RecyclerView	120
16.9. Как показывать представление по умолчанию до загрузки элементов или когда данные недоступны	121
16.10. Добавление верхнего/нижнего колонтитула в RecyclerView	123
16.11. Бесконечная прокрутка в RecyclerView	126
16.12. Добавление разделительных линий в элементы RecyclerView	127
 Глава 17. Украшения RecyclerView	128
17.1. Добавление разделителя в RecyclerView	128
17.2. Отрисовывание сепаратора	130
17.3. Как добавить разделители с помощью DividerItemDecoration	131
17.4. Отступы для каждого элемента с помощью ItemDecoration	131
17.5. ItemOffsetDecoration для GridLayoutManager в RecyclerView	132
 Глава 18. Слушатели onClickListeners в RecyclerView	133
18.1. Пример на Kotlin и RxJava	133
18.2. Слушатель кликов RecyclerView	134
18.3. Другой способ реализации слушателя клика элемента	135
18.4. Новый пример	137
18.5. Простой пример OnLongClick и OnClickListener	138
18.6. Слушатели клика элементов	141
 Глава 19. RecyclerView и LayoutManagers	143
19.1. Добавление представления заголовка в RecyclerView с помощью менеджера GridLayout	143
19.2. GridLayoutManager с динамическим подсчетом диапазонов	144
19.3. Простой список с LinearLayoutManager	147
19.4. StaggeredGridLayoutManager	150
 Глава 20. Пагинация в RecyclerView	152
20.1. MainActivity.java	152
 Глава 21. ImageView	157
21.1. Установка оттенка	157
21.2. Установка альфа-канала	157
21.3. Установка типа масштабирования	157
21.4. Тип масштабирования Center	160
21.5. Тип масштабирования CenterCrop	161
21.6. Тип масштабирования CenterInside	161
21.7. Тип масштабирования FitStart и FitEnd	162
21.8. Тип масштабирования FitCenter	163
21.9. Установка ресурса изображения	164
21.10. Тип масштабирования FitXY	164
21.11. MLRoundedImageView.java	165
 Глава 22. VideoView	166
22.1. Воспроизведение видео с URL-адреса с помощью VideoView	166
22.2. Создание VideoView	167

Глава 23. Оптимизация VideoView	167
23.1. Оптимизация VideoView в ListView	168
 Глава 24. WebView	180
24.1. Поиск и устранение неисправностей WebView с помощью вывода консольных сообщений или удаленной отладки	180
24.2. Передача данных из JavaScript в Java (Android)	181
24.3. Связь между Java и JavaScript	182
24.4. Пример работы с набирателем номера	182
24.5. Открытие локального файла и создание динамического содержимого в WebView	183
24.6. Диалоговые окна оповещения JavaScript в WebView	183
 Глава 25. SearchView	184
25.1. Установка темы для SearchView	184
25.2. SearchView в панели инструментов с использованием Fragment	184
25.3. Appcompat SearchView с наблюдателем RxBindings	186
 Глава 26. BottomNavigationView	189
26.1. Базовая реализация	189
26.2. Настройка BottomNavigationView	190
26.3. Обработка состояний Enabled / Disabled	190
26.4. Разрешение более чем трех меню	191
 Глава 27. Отрисовка на объекте холста Canvas с использованием SurfaceView	192
27.1. SurfaceView с потоком отрисовки	192
 Глава 28. Создание пользовательских представлений	197
28.1. Основы создания пользовательских представлений	197
28.2. Добавление атрибутов к представлениям	199
28.3. Советы по повышению производительности CustomView	201
28.4. Создание составного представления	202
28.6. Реакция на события касания	205
 Глава 29. Получение расчетных размеров представления	206
29.1. Расчет начальных размеров вида в активности	206
 Глава 30. Добавление FuseView в проект Android	207
30.1. Приложение hikr, просто еще один android.view.View	207
 Глава 31. Поддержка экранов с различными разрешениями и размерами	213
31.1. Использование квалификаторов конфигурации	213
31.2. Преобразование dp и sp в пиксели	214
31.3. Размер текста и различные размеры экрана Android	215
 Глава 32. ViewFlipper	215
32.1. ViewFlipper со скольжением изображения	215
 Глава 33. Паттерны проектирования	216
33.1. Шаблон наблюдателя	216
33.2. Пример класса Singleton	217
 Глава 34. Активность	218
34.1. Режим запуска активности	218

34.2. Исключение активности из истории обратного стека	219
34.3. Жизненный цикл активности в Android	219
34.4. Завершение работы приложения с исключением из последних открытых	222
34.5. Представление пользовательского интерфейса с помощью setContentView	223
34.6. Навигация вверх для активностей	224
34.7. Очистка стека текущих активностей и запуск новой активности	226
 Глава 35. Распознавание активности	226
35.1. Google Play ActivityRecognitionAPI	226
35.2. Распознавание активности PathSense	228
 Глава 36. Раздельный экран и многоэкранные действия	230
 Глава 37. Материальный дизайн	231
37.1. Добавление панели инструментов	231
37.2. Кнопки, стилизованные под материальный дизайн	232
37.3. Добавление плавающей кнопки действия (FloatingActionButton, FAB)	233
37.4. RippleDrawable	234
37.5. Добавление TabLayout	238
37.6. Нижние листы в библиотеке поддержки проектирования	239
37.7. Применение темы AppCompat	242
37.8. Добавление всплывающего уведомления (Snackbar)	243
37.9. Добавление выдвижной панели навигации	244
37.10. Как использовать TextInputLayout	247
 Глава 38. Ресурсы	248
38.1. Определение цвета	248
38.2. Уровень прозрачности цвета (альфа)	249
38.3. Определение множественного числа строк	250
38.4. Определение строк	250
38.5. Определение размеров	251
38.6. Форматирование строк в файле strings.xml	252
38.7. Определение целочисленного массива	252
38.8. Определение списка состояний цвета	253
38.9. 9 патчей (9-patch)	253
38.10. Получение ресурсов без предупреждений «deprecated»	255
38.11. Работа с файлом strings.xml	256
38.12. Определение массива строк	257
38.13. Определение целых чисел	257
38.14. Определение ресурса меню и его использование внутри активности или фрагмента	258
 Глава 39. Библиотека привязки данных	259
39.1. Базовая привязка текстовых полей	259
39.2. Встроенное двустороннее связывание данных	260
39.3. Пользовательское событие с использованием лямбда-выражения	261
39.4. Значение по умолчанию в привязке данных	263
39.5. Привязка данных в диалоге	264
39.6. Привязка с помощью метода доступа	264
39.7. Передача виджета в качестве ссылки в BindingAdapter	264
39.8. Слушатель кликов с привязкой	265
39.9. Привязка данных в адаптере RecyclerView Adapter	266
39.10. Привязка данных во фрагменте	267

39.11. Привязывание данных с пользовательскими переменными типов int, boolean	268
39.12. Ссыпочные классы	269
Глава 40. SharedPreferences	270
40.1. Реализация экрана настроек с помощью SharedPreferences	270
40.2. Commit и Apply	272
40.3. Чтение и запись значений в SharedPreferences	272
40.4. Извлечение всех сохраненных записей из определенного файла SharedPreferences	273
40.5. Чтение и запись данных в SharedPreferences с помощью синглтон-класса	274
40.6. getPreferences(int) и getSharedPreferences(String, int)	278
40.7. Прослушивание изменений SharedPreferences	279
40.8. Сохранение, извлечение, удаление и очистка данных из SharedPreferences	279
40.9. Добавление фильтра для EditTextPreference	280
40.10. Поддерживаемые типы данных в SharedPreferences	281
40.11. Различные способы инстанцирования объекта SharedPreferences	281
40.12. Удаление ключей	281
Глава 41. Намерение (Intent)	282
41.1. Получение результата от другой активности	282
41.2. Передача данных между активностями	284
41.3. Открытие URL-адреса в браузере	286
41.4. Паттерн стартера	287
41.5. Очистка стека активности	288
41.6. Запуск активности	288
41.7. Отправка электронных писем	289
41.8. CustomTabsIntent для пользовательских вкладок Chrome	289
41.9. Намерение URI	290
41.10. Запуск звонка	290
41.11. Передача широковещательных сообщений другим компонентам	291
41.12. Передача пользовательского объекта между активностями	292
41.13. Передача широты и долготы на Google Maps	294
41.14. Передача различных данных в активность при помощи намерения	294
41.15. ShareIntent	296
41.16. Отображение выбора файла и чтение результата	296
41.17. Совместное использование нескольких файлов с помощью намерения	298
41.18. Запуск несвязанной службы с помощью намерения	298
41.19. Получение результата из активности во фрагмент	299
Глава 42. Фрагменты	300
42.1. Передача данных из активности во фрагмент с помощью объекта Bundle	300
42.2. Шаблон newInstance()	301
42.3. Навигация между фрагментами с помощью обратного стека и шаблона Static Fabric	302
42.4. Отправка событий обратно в активность с помощью интерфейса обратного вызова	303
42.5. Анимирование перехода между фрагментами	304
42.6. Связь между фрагментами	305
Глава 43. Кнопка	309
43.1. Использование одного и того же события щелчка для одного или нескольких представлений в XML	309
43.2. Определение внешнего слушателя	310
43.3. Встроенный слушатель	310
43.4. Настройка стиля кнопок	310

43.5. Пользовательский ClickListener для предотвращения многократных быстрых щелчков	314
43.6. Использование макета для определения действия щелчка	315
43.7. Прослушивание событий длинного щелчка	315
 Глава 44. Эмулятор	316
44.1. Создание скриншотов	316
44.2. Имитация вызова	317
44.3. Открытие диспетчера AVD	318
44.4. Устранение ошибок при запуске эмулятора	318
 Глава 45. Службы	318
45.1. Жизненный цикл службы	318
45.2. Определение процесса службы	319
45.3. Создание несвязанной службы	320
45.4. Запуск службы	322
45.5. Создание связанной службы с помощью Binder	322
45.6. Создание удаленной службы (через AIDL)	323
 Глава 46. Файл манифеста	325
46.1. Объявление компонентов	325
46.2. Объявление разрешений в файле манифеста	326
 Глава 47. Gradle для Android	326
47.1. Базовый файл build.gradle	326
47.2. Определение и использование полей конфигурации сборки	329
47.3. Централизация зависимостей с помощью файла dependencies.gradle	331
47.4. Подписание APK без раскрытия пароля хранилища ключей	332
47.5. Добавление зависимостей, специфичных для варианта продукта	334
47.6. Указание различных идентификаторов приложений для типов сборки и вариантов продукта	335
47.7. Версионирование сборок с помощью файла version.properties	335
47.8. Определение вариантов продукта	336
47.9. Изменение выходного имени apk и добавление имени версии	337
47.10. Добавление ресурсов, специфичных для варианта продукта	337
47.11. Почему в проекте Android Studio два файла build.gradle?	338
47.12. Структура каталогов для ресурсов, специфичных для варианта продукта	338
47.13. Включение Proguard с помощью gradle	339
47.14.忽略する	339
47.15. Включение поддержки экспериментальных плагинов NDK для Gradle и AndroidStudio	339
47.16. Отображение информации о подписи	341
47.17. Просмотр дерева зависимостей	342
47.18. Отключение сжатия изображений для уменьшения размера APK-файла	343
47.19. Автоматическое удаление несогласованных (unaligned) apk	343
47.20. Выполнение сценария оболочки из gradle	344
47.21. Как показать все задачи проекта gradle	344
47.22. Отладка ошибок в Gradle	346
47.23. Использование gradle.properties для централизованного определения номера версии или конфигурации сборки	347
47.24. Определение типов сборки	348
 Глава 48. Файловый ввод/вывод в Android	348
48.1. Получение рабочего каталога	349

48.2. Запись необработанного массива байтов	349
48.3. Сериализация объекта	349
48.4. Запись на внешний носитель (SD-карту)	349
48.5. Решение проблемы с невидимыми файлами МТР	349
48.6. Работа с большими файлами	350
	350
Глава 49. FileProvider	351
49.1. Общий доступ к файлу	351
Глава 50. Хранение файлов во внутреннем и внешнем хранилищах	353
50.1. Android: внутренние и внешние хранилища – уточнение терминологии	353
50.2. Использование внешнего хранилища	356
50.3. Использование внутреннего хранилища	358
50.4. Получение каталога устройств	358
50.5. Сохранение базы данных на SD-карте (резервное копирование базы данных)	360
Глава 51. Zip-архивация в Android	361
Глава 52. Распаковка Zip-файлов в Android	362
Глава 53. Камера и галерея	362
53.1. Фотографирование	362
53.2. Получение полноразмерной фотографии	365
53.3. Декодирование растрового изображения, правильно повернутого из URI	368
53.4. Установка разрешения камеры	370
53.5. Как запустить камеру или галерею и сохранить результат работы камеры	371
Глава 54. API Camera2	374
54.1. Предварительный просмотр основной камеры в TextureView	374
Глава 55. API Fingerprint	383
55.1. Как использовать Android Fingerprint API для сохранения паролей пользователей	383
55.2. Добавление сканера отпечатков пальцев в приложение	391
Глава 56. Bluetooth и Bluetooth Low Energy API	393
56.1. Разрешения	393
56.2. Проверка включения Bluetooth	393
56.3. Поиск ближайших устройств Bluetooth Low Energy	393
56.4. Сделать устройство обнаруживаемым	397
56.5. Подключение к устройству Bluetooth	397
56.6. Поиск ближайших Bluetooth-устройств	399
Глава 57. Разрешения во время выполнения	399
57.1. Множественные разрешения	399
57.2. Несколько разрешений на выполнение из одних и тех же групп разрешений	400
57.3. Использование PermissionUtil	402
57.4. Включение всего кода, связанного с разрешениями, в абстрактный базовый класс	403
57.5. Обеспечение прав доступа в широковещательных передачах и URI	405
Глава 58. API Places	406
58.1. Получение текущего местоположения с помощью API Places	406
58.2. Интеграция автозаполнения мест	408

58.3. Пример использования Place Picker	409
58.4. Настройка фильтров типов мест для PlaceAutocomplete	410
58.5. Добавление более одной активности Google Autocomplete	411
 Глава 59. Android NDK	 412
59.1. Вход в систему NDK	412
59.2. Создание нативных исполняемых файлов (native executables) для Android	413
59.3. Как очистить сборку	413
59.4. Как использовать makefile, отличающийся от Android.mk	413
 Глава 60. Тема DayNight	 413
 Глава 61. Библиотека Glide	 414
61.1. Загрузка изображения	414
61.2. Добавление Glide в проект	415
61.3. Преобразование круга в Glide (загрузка изображения в круговой ImageView)	416
61.4. Преобразования по умолчанию	417
61.5. Изображение с закругленными углами с помощью пользовательской Glide Target	417
61.6. Обработка заполнителей и ошибок	418
61.7. Предварительная загрузка изображений	418
61.8. Обработка ошибок при загрузке изображений Glide	419
61.9. Загрузка изображения в круговой ImageView без пользовательских преобразований	419
 Глава 62. Диалог	 420
62.1. Добавление Material Design AlertDialog в приложение с помощью Appcompat	420
62.2. Базовый диалог оповещения	421
62.3. ListView в диалоге оповещения	422
62.4. Пользовательский диалог оповещения с EditText	422
62.5. DatePickerDialog	424
62.6. DatePicker	424
62.7. Диалог оповещения	425
62.8. Диалог оповещения с многострочным заголовком	425
62.9. Выбор даты в диалоговом фрагменте	427
62.10. Полнэкранный пользовательский диалог без фона и заголовка	429
 Глава 63. Расширение возможностей диалоговых окон оповещения	 430
63.1. Диалог оповещения, содержащий ссылку, по которой можно перейти	430
 Глава 64. Анимированное диалоговое окно AlertDialog	 430
64.1. Код для анимированного диалога	431
 Глава 65. GreenDAO	 433
65.1. Вспомогательные методы для запросов SELECT, INSERT, DELETE, UPDATE	433
65.2. Создание сущности в GreenDAO 3.X с составным первичным ключом	436
65.3. Начало работы с GreenDAO v3.X	436
 Глава 66. Атрибуты инструментов	 438
 Глава 67. Форматирование строк	 439
67.1. Форматирование строкового ресурса	439
67.2. Форматирование типов данных в String и обратно	439
67.3. Форматирование временной метки в строку	439

Глава 68. SpannableString	439
68.1. Добавление стилей в TextView	439
68.2. Много строк, использование разных цветов	442
 Глава 69. Уведомления	
69.1. Уведомление в режиме Heads Up	442
69.2. Создание простого уведомления	442
69.3. Установка пользовательского уведомления – отображение полного содержания текста	444
69.4. Динамическое получение правильного пиксельного размера для большого значка	445
69.5. Текущее уведомление с кнопкой действия	445
69.6. Установка различных приоритетов в уведомлении	446
69.7. Установка пользовательского значка уведомления с помощью библиотеки Picasso	446
69.8. Планирование уведомлений	447
 Глава 70. AlarmManager	449
70.1. Как отменить сигнал Alarm	449
70.2. Создание точных Alarm-сигналов на всех версиях Android	449
70.3. На API23+ режим Doze вмешивается в работу AlarmManager	449
70.4. Как отложить запуск намерения	450
 Глава 71. Обработчик	450
71.1. HandlerThreads и взаимодействие между потоками	450
71.2. Использование обработчика для создания таймера (аналогично javax.swing.Timer)	451
71.3. Использование обработчика для выполнения кода через промежуток времени	452
71.4. Остановка выполнения обработчика	453
 Глава 72. BroadcastReceiver	453
72.1. Использование LocalBroadcastManager	454
72.2. Основы BroadcastReceiver	454
72.3. Дополнительная информация о приемниках вещания	455
72.4. Использование упорядоченных трансляций	455
72.5. Липкая трансляция (Sticky Broadcast)	456
72.6. Включение и отключение приемника вещания программным способом	456
72.7. Пример LocalBroadcastManager	456
72.8. Остановленное состояние (stopped state)	457
72.9. Передача данных между двумя активностями через пользовательский Broadcast receiver	458
72.10. BroadcastReceiver для обработки событий BOOT_COMPLETED	458
72.11. Bluetooth BroadcastReceiver	459
 Глава 73. Жизненный цикл пользовательского интерфейса	460
73.1. Сохранение данных при обрезке памяти	460
 Глава 74. HttpURLConnection	461
74.1. Создание HttpURLConnection	461
74.2. Отправка запроса HTTP GET	462
74.3. Чтение тела запроса HTTP GET	462
74.4. Отправка HTTP POST-запроса с параметрами	463
74.5. Многоцелевой класс HttpURLConnection для обработки всех типов HTTP-запросов	464

74.6. Использование HttpURLConnection для multipart- и multiform-данных	467
74.7. Загрузка (POST) файла с помощью HttpURLConnection	470
Глава 75. URL обратного вызова	471
75.1. Пример URL-адреса обратного вызова с использованием Instagram OAuth	471
Глава 76. Всплывающее уведомление (Snackbar)	472
76.1. Создание простого всплывающего уведомления	472
76.2. Пользовательское всплывающее уведомление	472
76.3. Пользовательское всплывающее уведомление (View не требуется)	473
76.4. Всплывающее уведомление с обратным вызовом	474
76.5. Snackbar и Toast: что лучше использовать?	474
76.6. Пользовательское всплывающее уведомление со значком Undo	475
Глава 77. Виджеты	475
77.1. Объявление в манифесте	475
77.2. Метаданные	476
77.3. Класс AppWidgetProvider	476
77.4. Создание и встраивание базового виджета	476
77.5. Объявление двух виджетов с разными макетами	478
Глава 78. Toast-уведомления	479
78.1. Создание пользовательского Toast-уведомления	479
78.2. Установка положения Toast-уведомления	480
78.3. Отображение Toast-уведомления	480
78.4. Отображение Toast-уведомления над экранной клавиатурой	481
78.5. Потокобезопасный способ отображения Toast-уведомления (для всего приложения)	481
78.6. Потокобезопасный способ отображения Toast-уведомления (для AsyncTask)	482
Глава 79. Создание синглтон-класса для тост-уведомления	482
Глава 80. Интерфейсы	484
80.1. Пользовательский слушатель	484
80.2. Базовый слушатель	486
Глава 81. Аниматоры	487
81.1. Анимация TransitionDrawable	487
81.2. Анимация с постепенным появлением и затуханием	488
81.3. ValueAnimator	489
81.4. Разворачивание и сворачивание анимации представления	489
81.5. ObjectAnimator	490
81.6. ViewPropertyAnimator	491
81.7. Анимация встряхивания ImageView	491
Глава 82. Расположение	492
82.1. API Fused Location	492
82.2. Получение адреса из местоположения с помощью геокодера	497
82.3. Запрос обновлений местоположения с помощью LocationManager	498
82.4. Запрос обновлений местоположения в отдельном потоке с использованием LocationManager	499
82.5. Получение обновлений местоположения в приемнике BroadcastReceiver	501

82.6. Регистрация GeoFence	501
Глава 83. Тема, стиль, атрибут	505
83.1. Определение основного, основного темного и акцентного цветов	505
83.2. Несколько тем в одном приложении	505
83.3. Цвет панели навигации	505
83.4. Использование пользовательской темы для каждой активности	507
83.5. Глобальное использование пользовательской темы	507
83.6. Цвет прокрутки	508
83.7. Цвет пульсации	508
83.8. Полупрозрачные панели навигации и состояния	508
83.9. Наследование тем	508
Глава 84. MediaPlayer	509
84.1. Базовое создение и воспроизведение	509
84.2. Медиаплеер с прогрессом буфера и позицией воспроизведения	509
84.3. Получение системных мелодий	512
84.4. Асинхронная подготовка	512
84.5. Импорт звука в Android Studio и его воспроизведение	513
84.6. Получение и настройка системной громкости	514
Глава 85. Звук и мультимедиа в Android	515
85.1. Как выбрать изображение и видео	515
85.2. Воспроизведение звука через SoundPool	516
Глава 86. MediaSession	517
86.1. Получение и обработка событий от кнопок	517
Глава 87. MediaStore	519
87.1. Выборка аудио/MP3-файлов из определенной папки устройства или выборка всех файлов	519
Глава 88. Multidex и предел метода Dex	522
88.1. Включение Multidex	522
88.2. Реализация Multidex путем расширения приложения	523
88.3. Реализация Multidex путем расширения MultiDexApplication	524
88.4. Реализация Multidex при использовании MultiDexApplication напрямую	524
88.5. Подсчет ссылок на методы при каждой сборке (плагин Dexcount Gradle)	524
Глава 89. Синхронизация данных с помощью Sync Adapter	525
89.1. Фиктивный адаптер синхронизации с провайдером-заглушкой	525
Глава 90. Режим Портера-Даффа	531
90.1. Создание цветового фильтра PorterDuff	531
90.2. Создание режима PorterDuff XferMode	531
90.3. Применение виньетки к растровому изображению с использованием PorterDuffXfermode	532
Глава 91. Меню	532
91.1. Меню опций с разделителями	532
91.2. Применение пользовательского шрифта к меню	533
91.3. Создание меню в активности	533

Глава 92. Picasso	536
92.1. Добавление библиотеки Picasso в проект	536
92.2. Создание круглых аватаров с помощью Picasso	536
92.3. Местозаполнители	538
92.4. Изменение размеров и поворот	538
92.5. Отключение кэша в Picasso	538
92.6. Использование Picasso в качестве ImageGetter для Html.fromHtml	539
92.7. Отмена запросов изображений с помощью Picasso	540
92.8. Загрузка изображения из внешнего хранилища	540
92.9. Загрузка изображения как Bitmap с помощью Picasso	541
Глава 93. RoboGuice	541
93.1. Простой пример	541
93.2. Установка для проектов Gradle	541
93.3. Аннотация @ContentView	542
93.4. Аннотация @InjectResource	542
93.5. Аннотация @InjectView	542
93.6. Введение в RoboGuice	542
Глава 94. Библиотека ACRA	544
94.1. ACRAHandler	544
94.2. Пример манифеста	545
94.3. Установка	545
Глава 95. Parcelable	545
95.1. Создание пользовательского Parcelable-объекта	545
95.2. Parcelable-объект, содержащий другой Parcelable-объект	547
95.3. Использование Enums с Parcelable-объектами	548
Глава 96. Пикеры даты и времени	549
96.1. Диалоговое окно выбора даты Date Picker	549
96.2. DatePicker с использованием материального оформления	549
Глава 97. Локализованные дата и время	551
97.1. Пользовательский локализованный формат даты с помощью DateUtils.formatDateTime()	551
97.2. Стандартное форматирование даты и времени	552
97.3. Полностью настраиваемые дата и время	552
Глава 98. Возможности для работы со временем	552
98.1. Проверка в течение периода	552
98.2. Преобразование формата даты в миллисекунды	553
98.3. Метод GetCurrentRealTime	554
Глава 99. Встроенные покупки	554
99.1. Расходуемые встроенные покупки	554
99.2. Использование сторонней библиотеки In-App v3	559
Глава 100. Плавающая кнопка действия	560
100.1. Как добавить плавающую кнопку действия в макет	560
100.2. Показ и скрытие плавающей кнопки действия при пролистывании	561
100.3. Показ и скрытие плавающей кнопки действия при прокрутке	563
100.4. Настройка поведения кнопки FAB	565

Глава 101. События касания	566
101.1. Как различать события касания дочерней и родительской групп представлений	566
Глава 102. Обработка событий касания и движения	566
102.1. Кнопки	568
102.2. Поверхности	568
102.3. Обработка множественного касания поверхности	569
	570
Глава 103. Обнаружение события встряхивания в Android	571
103.1. Детектор тряски	571
103.2. Использование системы обнаружения встряхиваний Seismic	572
Глава 104. GreenRobot EventBus	573
104.1. Передача простого события	573
104.2. События приема	574
104.3. Отправка событий	575
Глава 105. Вибрация	575
105.1. Начало работы с вибрацией	575
105.2. Вибрация с неопределенной продолжительностью	575
105.3. Шаблоны вибрации	576
105.4. Остановка вибрации	576
105.5. Однократная вибрация	576
Глава 106. Провайдеры контента	576
106.1. Реализация базового класса провайдера контента	576
Глава 107. Dagger 2	580
107.1. Настройка компонентов для внедрения зависимостей приложений и активностей	580
107.2. Пользовательские области действия	582
107.3. Использование @Subcomponent вместо @Component(dependencies={...})	582
107.4. Создание компонента из нескольких модулей	582
107.5. Как добавить Dagger 2 в build.gradle	584
107.6. Внедрение конструктора	584
Глава 108. Realm	585
108.1. Сортированные запросы	585
108.2. Использование Realm с RxJava	585
108.3. Основы использования	586
108.4. Список примитивов (RealmList<Integer/String/...>)	589
108.5. Асинхронные запросы	590
108.6. Добавление Realm в проект	590
108.7. Модели Realm	590
Глава 109. Версии Android	591
109.1. Проверка версии Android на устройстве во время выполнения программы	591
Глава 110. Подключение Wi-Fi	592
110.1. Подключение с WEP-шифрованием	592
110.2. Подключение с шифрованием WPA2	593
110.3. Сканирование точек доступа	593

Глава 111. SensorManager	595
111.1. Определение, является ли устройство статичным, с помощью акселерометра	595
111.2. Получение событий от сенсорных датчиков	596
111.3. Преобразование значений датчиков в мировую систему координат	597
Глава 112. Индикатор ProgressBar	598
112.1. Индикатор ProgressBar в материальном оформлении	598
112.2. Тонирование ProgressBar	599
112.3. Настраиваемый ProgressBar	599
112.4. Создание пользовательского диалогового окна выполнения	602
112.5. Неопределенный ProgressBar	603
112.6. Определенный ProgressBar	604
Глава 113. Пользовательские шрифты	606
113.1. Пользовательский шрифт в тексте объекта Canvas (холста)	606
113.2. Новые возможности для работы со шрифтами	606
113.3. Пользовательский шрифт для всей активности	607
113.4. Использование пользовательского шрифта в приложении	607
113.5. Инициализация шрифта	607
113.6. Использование пользовательского шрифта в TextView	607
113.7. Применение шрифта к TextView с помощью xml (без Java-кода)	607
113.8. Эффективная загрузка шрифтов	609
Глава 114. Получение имен системных шрифтов и их использование	609
114.1. Получение имен системных шрифтов	609
114.2. Применение системного шрифта к TextView	610
Глава 115. Преобразование текста в речь (TTS)	610
115.1. Основы преобразования текста в речь	610
115.2. Реализация TTS в API-интерфейсах	612
Глава 116. Спиннер	614
116.1. Пример спиннера	614
116.2. Добавление спиннера в активность	616
Глава 117. Шифрование и расшифровка данных	617
117.1. Безопасное AES-шифрование данных с использованием пароля	617
Глава 118. OkHttp	618
118.1. Пример базового использования	618
118.2. Настройка OkHttp	619
118.3. Перехватчик протоколирования	619
118.4. Синхронный вызов Get	619
118.5. Асинхронный вызов Get	620
118.6. Параметры формы отправки сообщения	620
118.7. Отправка многосоставного запроса	620
118.8. Перезапись ответов	621
Глава 119. Обработка глубоких ссылок	621
119.1. Получение параметров запроса	622
119.2. Простая глубокая ссылка	622
119.3. Несколько путей в одном домене	622

119.4. Несколько доменов и несколько путей	623
119.5. http и https для одного и того же домена	623
119.6. Использование pathPrefix	624
Глава 120. Создание отчетов о сбоях	624
120.1. Fabric – Crashlytics	624
120.2. Перехват сбоев с помощью Sherlock	624
120.3. Принудительный краш-тест при помощи Fabric	628
120.4. Отчетность о сбоях с помощью ACRA	629
Глава 121. Проверка подключения к Интернету	630
121.1. Проверка наличия у устройства подключения к Интернету	630
121.2. Проверка качества сигнала в сети	631
Благодарности	634

Глава 1. Начало работы с Android

Версия	Уровень API	Код версии	Дата выпуска
1.0	1	BASE	2008-09-23
1.1	2	BASE_1_1	2009-02-09
1.5	3	CUPCAKE	2009-04-27
1.6	4	DONUT	2009-09-15
2.0	5	ECLAIR	2009-10-26
2.0.1	6	ECLAIR_0_1	2009-12-03
2.1.x	7	ECLAIR_MR1	2010-01-12
2.2.x	8	FROYO	2010-05-20
2.3	9	GINGERBREAD	2010-12-06
2.3.3	10	GINGERBREAD_MR1	2011-02-09
3.0.x	11	HONEYCOMB	2011-02-22
3.1.x	12	HONEYCOMB_MR1	2011-05-10
3.2.x	13	HONEYCOMB_MR2	2011-07-15
4.0	14	ICE_CREAM SANDWICH	2011-10-18
4.0.3	15	ICE_CREAM SANDWICH_MR1	2011-12-16
4.1	16	JELLY_BEAN	2012-07-09
4.2	17	JELLY_BEAN_MR1	2012-11-13
4.3	18	JELLY_BEAN_MR2	2013-07-24
4.4	19	KITKAT	2013-10-31
4.4W	20	KITKAT_WATCH	2014-06-25
5.0	21	LOLLIPOP	2014-11-12
5.1	22	LOLLIPOP_MR1	2015-03-09
6.0	23	M (Marshmallow)	2015-10-05
7.0	24	N (Nougat)	2016-08-22
7.1	25	N_MR1 (Nougat MR1)	2016-10-04
8.0	26	O (Developer Preview 4)	2017-07-24
8.1	27	OREO (Oatmeal Cookie)	2017-12-05
9	28	PIE (Pistachio Ice Cream)	2018-08-06
10	29	Quince Tart	2019-09-03
11	30	Red Velvet Cake	2020-09-08
12	31	Snow Cone	2021-10-04
12L	32	Snow Cone v2	2022-03-07
13	33	Tiramisu	2022-08-15
14	34	Upside Down Cake	2023-10-04

1.1. Создание нового проекта

Настройка Android Studio

Начните с установки Android Studio, а затем откройте ее. Теперь вы готовы создать свое первое приложение для Android!

Примечание: данное руководство основано на Android Studio 2.2, но в других версиях процесс в основном такой же.

Настройка базовой конфигурации проекта

Начать новый проект можно двумя способами.

В окне приветствия нажмите Start a New Android Studio Project. Если проект уже открыт, перейдите в меню File → New Project.

Далее необходимо описать свою заявку, заполнив несколько полей.

1. Application Name (Имя приложения) – это имя будет показано пользователю.

Пример: Hello World.

Вы всегда можете изменить его позже в файле `AndroidManifest.xml`.

2. Company Domain (Домен компании) – квалифициатор имени пакета вашего проекта.

Пример: `stackoverflow.com`.

3. Package Name (Имя пакета) (или `applicationId`) – полное имя пакета проекта.

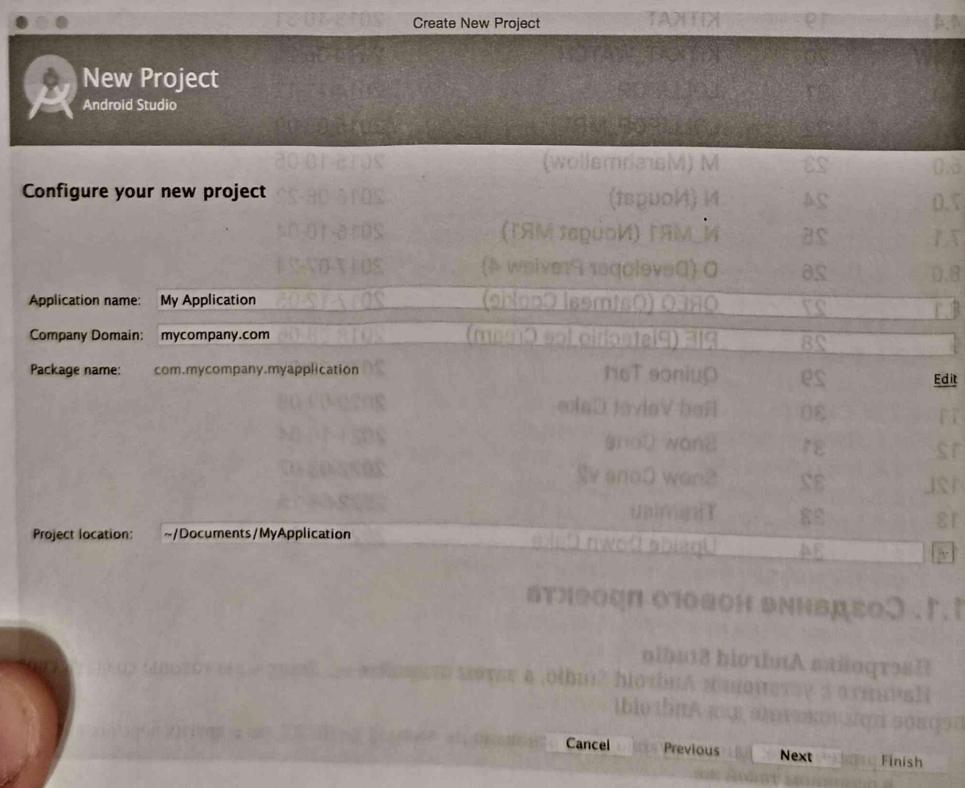
Он должен соответствовать *обратной нотации доменных имен (Reverse Domain Name Notation, она же Reverse DNS)*:

Top Level Domain. Company Domain. [Company Segment.] Application Name.

(Домен верхнего уровня. Домен компании. [Сегмент компании.] Название приложения.)

Пример: `com.stackoverflow.android.helloworld` или `com.stackoverflow.helloworld`. Вы всегда можете изменить свой `applicationId`, переопределив его в файле `gradle`.

Не используйте стандартный префикс “`com.example`”, если вы собираетесь отправлять свое приложение в Google Play Store. Имя пакета будет являться **уникальным идентификатором приложения** в Google Play.



4. Project Location (Расположение проекта) – каталог, в котором будет храниться ваш проект.

Выбор форм-факторов и уровня API

В следующем окне можно выбрать форм-факторы, поддерживаемые приложением: телефон, планшет, телевизор, носимые устройства и Google Glass. Выбранные форм-факторы становятся модулями приложения в проекте. Для каждого форм-фактора можно также выбрать уровень API для этого приложения. Чтобы получить дополнительную информацию, нажмите кнопку **Help me choose**.

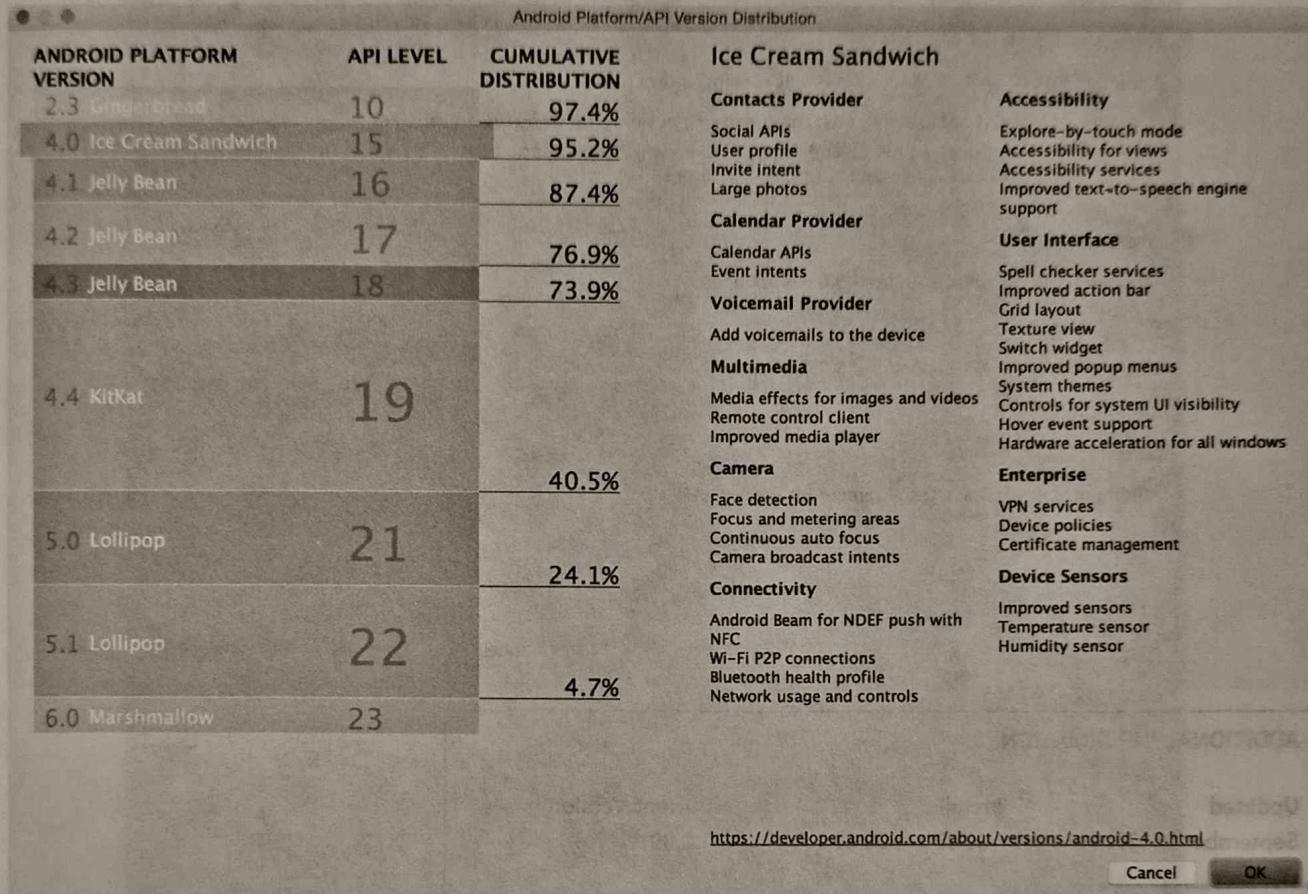


Диаграмма распределения текущих версий Android, отображаемая при нажатии кнопки *Help me choose*.

В окне “Распределение платформ Android” показано распределение мобильных устройств, работающих под управлением каждой версии Android, как показано на рисунке выше. Щелкните на уровне API, чтобы увидеть список функций, появившихся в соответствующей версии Android. Это поможет вам выбрать минимальный уровень API, содержащий все необходимые функции для вашего приложения, чтобы охватить как можно больше устройств. Затем нажмите кнопку **OK**.

Теперь выберите, какие платформы и версии Android SDK будет поддерживать приложение.

Пока выберите только *Phone* и *Tablet*.

Minimum SDK – это нижняя граница для вашего приложения. Это один из сигналов, используемых магазином Google Play Store для определения устройств, на которые может быть установлено приложение. Например, приложение Stack Exchange поддерживает Android 4.1+.

Android Studio подскажет (приблизительно), какой процент устройств будет поддерживаться при заданном минимальном SDK.

Более низкие уровни API ориентированы на большее количество устройств, но имеют меньшее количество доступных функций.

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

Phone and Tablet

Minimum SDK API 15: Android 4.0.3 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available.

By targeting API 15 and later, your app will run on approximately 97.4% of the devices that are active on the Google Play Store.

[Help me choose](#)

Wear

Minimum SDK API 21: Android 5.0 (Lollipop)

TV

Minimum SDK API 21: Android 5.0 (Lollipop)

Android Auto

Glass

Minimum SDK Glass Development Kit Preview (API 19)

Cancel Previous Next Finish

ADDITIONAL INFORMATION

Updated September 28, 2016	Installs 100,000 - 500,000	Current Version 1.0.89
Requires Android 4.1 and up	Content Rating Rated for 12+ Parental Guidance Recommended Learn more	Interactive Elements Users Interact
Permissions View details	Report Flag as inappropriate	Offered By Stack Exchange

При выборе **минимального SDK** следует обратить внимание на Dashboards stats (статистику информационных панелей), которая позволяет получить информацию о версиях устройств, посетивших Google Play Store в глобальном масштабе за последнюю неделю.

Добавление активности

Теперь мы собираемся выбрать активность по умолчанию для нашего приложения. В **Android Activity** (активность) – это отдельный экран, который будет представлен пользователю. Приложение может содержать несколько активностей и перемещаться между ними. Для данного примера выберем **Empty Activity** и нажмем кнопку **Next**.

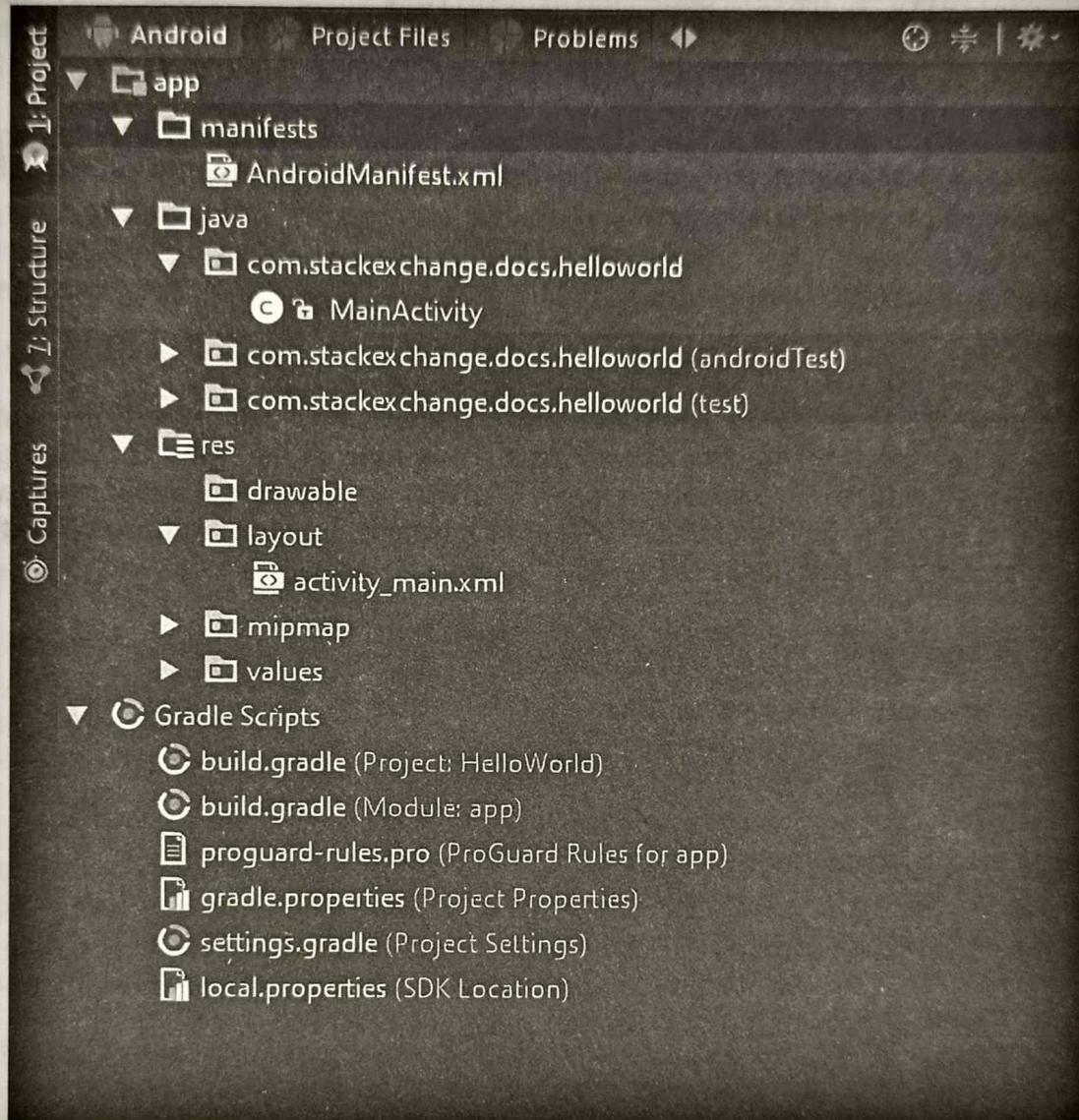
Здесь при желании можно изменить название активности и макета. Хорошей практикой является использование суффикса **Activity** в качестве имени активности и префикса **activity_** в качестве имени макета. Если мы оставим их по умолчанию, то **Android Studio** сре-

нерирует для нас активность под названием `MainActivity`, а файл макета – `activity_main`. Теперь нажмите кнопку `Finish`.

Android Studio создаст и настроит наш проект, что может занять некоторое время в зависимости от системы.

Инспектирование проекта

Чтобы понять, как работает Android, давайте посмотрим на некоторые файлы, которые были созданы для нас. На левой панели Android Studio мы видим *структур* нашего *Android-приложения*.



Для начала откроем файл `AndroidManifest.xml`, дважды щелкнув на нем мышью. Файл манифеста Android описывает некоторую базовую информацию о приложении Android. Он содержит объявление наших активностей, а также некоторые более сложные компоненты.

Если приложению требуется доступ к функции, защищенной разрешением, оно должно объявить о необходимости такого разрешения с помощью элемента `<uses-permission>` в манифесте. Затем, когда приложение устанавливается на устройство, программа установки определяет, нужно ли предоставлять запрошенное разрешение, проверяя авторитеты, подписавшие сертификаты приложения, и в некоторых случаях спрашивая пользователя. Приложение также может защищать свои собственные компоненты (действия, службы, приемники вещания и поставщики контента) с помощью разрешений. Оно может использовать любое из разрешений, определенных Android (перечисленных в `android.Manifest.permission`) или объявленных другими приложениями. Или же может определить свои собственные.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.stackoverflow.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name" android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

Далее откроем файл `activity_main.xml`, расположенный в `app/src/main/res/layout/`. Этот файл содержит декларации для визуальных компонентов нашей `MainActivity`. Вы увидите визуальный дизайнер. Он позволяет перетаскивать элементы на выбранный макет.

Также можно переключиться на конструктор xml-макетов, нажав кнопку "Text" в нижней части Android Studio, как показано здесь:



```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"

```

```
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.stackexchange.docs.helloworld.MainActivity">

<TextView
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:text="Hello World!" />
</RelativeLayout>
```

Внутри этого макета появится виджет `TextView` со свойством `android:text`, установленным в значение "Hello World!". Это блок текста, который будет показан пользователю при запуске приложения. Подробнее о *макетах и атрибутах* можно прочитать в соответствующих главах.

Далее рассмотрим `MainActivity`. Вот Java-код, сгенерированный для `MainActivity`:

```
public class MainActivity extends AppCompatActivity {

    // Метод onCreate вызывается при запуске активности
    // Здесь мы настраиваем наш макет
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // setContentView устанавливает макет активности на указанный XML-макет
        // В нашем случае мы используем макет activity_main
        setContentView(R.layout.activity_main);
    }
}
```

Как определено в нашем манифесте Android, `MainActivity` будет запускаться по умолчанию при запуске пользователем приложения `HelloWorld`. Откройте файл с именем `build.gradle`, расположенный в `app/`. Android Studio использует систему сборки **Gradle** для компиляции и сборки Android-приложений и библиотек.

```
apply plugin: 'com.android.application'

android {
    signingConfigs {
        applicationName {
            keyAlias 'applicationName'
            keyPassword 'password'
            storeFile file('../key/applicationName.jks')
            storePassword 'anotherPassword'
        }
    }
    compileSdkVersion 26
    buildToolsVersion "26.0.0"

    defaultConfig {
        applicationId "com.stackexchange.docs.helloworld"
        minSdkVersion 16
        targetSdkVersion 26
        versionCode 1
    }
}
```

```

versionName "1.0"
signingConfig signingConfigs.applicationName
}
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-
        rules.pro'
    }
}
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:26.0.0'
}

```

Этот файл содержит информацию о сборке и версии вашего приложения, а также может использоваться для добавления зависимостей от внешних библиотек. Пока не будем вносить никаких изменений.

Рекомендуется всегда выбирать последнюю доступную версию зависимостей.

compileSdkVersion

`compileSdkVersion` – это способ указать *Gradle*, с какой версией Android SDK компилировать приложение. Использование нового Android SDK является обязательным условием для использования любого из новых API, добавленных на этом уровне.

Следует подчеркнуть, что изменение версии `compileSdkVersion` не приводит к изменению поведения во время выполнения. Хотя при изменении версии `compileSdkVersion` могут появиться новые предупреждения или ошибки компилятора, версия `compileSdkVersion` не включается в APK: она используется только во время компиляции.

Поэтому настоятельно рекомендуется всегда компилировать с использованием последней версии SDK. Вы получите все преимущества новых проверок компиляции существующего кода, сможете избежать устаревших API и будете готовы к использованию новых API.

minSdkVersion

Если `compileSdkVersion` устанавливает самые новые доступные API, то `minSdkVersion` – это *нижняя граница* для вашего приложения. Значение `minSdkVersion` является одним из сигналов, используемых Google Play Store для определения того, на какое из пользовательских устройств может быть установлено приложение.

Вы получите предупреждение об использовании API, превышающих вашу `minSdkVersion`, что поможет вам избежать проблем, связанных с попыткой вызова API, которого не существует. Проверка версии системы во время выполнения – распространенный прием при использовании API только на новых версиях платформ.

targetSdkVersion

`targetSdkVersion` – это основной способ, с помощью которого Android обеспечивает совместимость на будущее, не применяя изменения поведения до тех пор, пока не будет обновлена версия `targetSdkVersion`. Это позволяет использовать новые API до того, как будут внесены изменения в поведение.

Обновление до последней версии SDK должно быть приоритетным для каждого приложения. Это не означает, что вы должны использовать каждую новую появившуюся возможность или слепо обновлять версию `targetSdkVersion` без тестирования.

`targetSDKVersion` – это версия Android, которая является верхней границей для доступных инструментов. Если `targetSDKVersion` меньше 23, то приложению не нужно запраши-

вать разрешения во время выполнения для экземпляра, даже если приложение выполняется на API 23+. TargetSDKVersion не препятствует запуску приложения на версиях Android выше выбранной.

Запуск приложения

Теперь запустим наше приложение *HelloWorld*. Вы можете запустить виртуальное устройство Android (которое можно настроить с помощью AVD Manager в Android Studio, как описано в примере ниже) или подключить собственное устройство Android через USB-кабель.

Настройка устройства Android

Чтобы запустить приложение из Android Studio на устройстве Android, необходимо включить USB Debugging (отладку по USB) в окне Developer Options (параметры разработчика) в настройках устройства.

Settings > Developer options > USB debugging

Если Developer Options не отображаются в настройках, перейдите в раздел About Phone и нажмите Build Number (номер сборки) семь раз. Это позволит отобразить Developer Options в настройках.

Settings > About phone > Build number

Также может потребоваться изменение конфигурации build.gradle для сборки на версии, которая есть на вашем устройстве.

Запуск из Android Studio

Нажмите зеленую кнопку Run на панели инструментов в верхней части Android Studio. В появившемся окне выберите устройство, на котором будет запущено приложение (при необходимости запустите виртуальное устройство Android или см. главу «Настройка виртуального устройства AVD (Android Virtual Device)»), и нажмите кнопку OK.



Появится всплывающее окно для разрешения отладки по USB. Нажмите OK, чтобы согласиться.

Теперь приложение будет установлено и запущено на вашем Android-устройстве или в эмуляторе.

Расположение файла APK

При подготовке приложения к выпуску выполняются конфигурирование, сборка и тестирование релизной (выпускной) версии приложения. Задачи конфигурирования просты и включают в себя базовую очистку и модификацию кода, которые помогают оптимизировать приложение. Процесс сборки аналогичен процессу отладочной сборки и может быть выполнен с помощью средств JDK и Android SDK. Задачи тестирования служат финальной проверкой, гарантирующей, что приложение работает в реальных условиях так, как ожидалось. После завершения подготовки приложения к выпуску вы получаете подписанный APK-файл, который можно распространять непосредственно среди пользователей или через торговую площадку приложений, например Google Play.

Android Studio

Поскольку в приведенных выше примерах используется Gradle, местоположение сгенерированного APK-файла: <Ваше местоположение проекта>/app/build/outputs/apk/app-debug.apk

IntelliJ

Если вы являетесь пользователем IntelliJ до перехода на Studio и импортируете свой проект IntelliJ напрямую, то ничего не изменилось. Расположение выходных данных будет таким же:

`out/production/...`

Примечание: эта функция будет устаревать примерно с версии 1.0.

Eclipse

Если вы собираетесь импортировать проект Android Eclipse напрямую, не делайте этого! Как только в проекте появятся зависимости (jars или Library Projects), это не сработает, и проект не будет правильно настроен. Если у вас нет зависимостей, то apk будет находиться в том же месте, где вы найдете его в Eclipse:

`bin/...`

1.2. Настройка Android Studio

Android Studio – это среда разработки для Android, официально поддерживаемая и рекомендуемая компанией Google. Android Studio поставляется в комплекте с Android SDK Manager, который представляет собой инструмент для загрузки компонентов Android SDK, необходимых для начала разработки приложений.

Установка инструментов Android Studio и Android SDK:

1. Загрузите и установите Android Studio.
2. Загрузите последние версии SDK Tools и инструменты для платформы SDK, открыв Android Studio, а затем следуя инструкциям Android SDK Tool Updates. Необходимо установить последние доступные стабильные пакеты.

Если вам необходимо работать со старыми проектами, которые были созданы с использованием старых версий SDK, то вам может потребоваться загрузить эти версии.

Начиная с версии Android Studio 2.2 копия последней версии OpenJDK поставляется в комплекте с установкой и является рекомендуемым JDK (Java Development Kit) для всех проектов Android Studio. Таким образом, отпадает необходимость в установке пакета JDK от Oracle. Чтобы использовать поставляемый SDK, выполните следующие действия.

1. Откройте проект в Android Studio и выберите в строке меню **Файл > Структура проекта**.
2. На странице **SDK Location** и в разделе **JDK location** установите **Use embedded JDK**.
3. Нажмите кнопку **OK**.

Настройка Android Studio

Android Studio предоставляет доступ к двум конфигурационным файлам через меню Help:

- `studio.vmoptions`: настраивает параметры виртуальной машины Java (JVM) Studio, такие как размер кучи (heap size) и размер кэша. Обратите внимание, что на Linux-устройствах этот файл может иметь имя `studio64.vmoptions`, в зависимости от версии Android Studio.
- `idea.properties`: настройка свойств Android Studio, таких как путь к папке `plugins` или максимальный поддерживаемый размер файла.

Изменение или добавление темы

Вы можете делать это по своему усмотрению. Используйте **File->Settings->Editor->Colors & Fonts->** (Файл->Настройки->Редактор->Цвета и шрифты->) и выберите тему. Также можно загрузить новые темы с сайта <http://color-themes.com/>. После загрузки .jar.zip файла перейдите в **File -> Import Settings...** и выберите загруженный файл.

Компиляция приложений

Создайте новый или откройте существующий проект в Android Studio и нажмите зеленую кнопку на верхней панели инструментов для его запуска. Если она серого цвета, то необходимо подождать немного, чтобы Android Studio правильно проиндексировала некоторые файлы, ход выполнения которых можно увидеть в нижней строке состояния.

Если вы хотите создать проект из оболочки, убедитесь, что у вас есть файл local.properties, который создается Android Studio автоматически. Если необходимо создать проект без Android Studio, то в строке, начинающейся с `sdk.dir=`, должен быть указан путь к установке SDK.

Откройте оболочку и перейдите в каталог проекта. Введите `./gradlew aR` и нажмите клавишу Enter. aR – это сокращение для `assembleRelease`, который загрузит все зависимости и соберет приложение. Итоговый APK-файл будет находиться в каталоге `ProjectName/ModuleName/build/outputs/apk` и будет называться `ModuleName-release.apk`.

1.3. Программирование под Android без использования IDE

Приведем минималистичный пример Hello World, в котором используются только самые базовые инструменты Android.

Требования и допущения

- Oracle JDK 1.7 или более поздней версии
- Android SDK Tools (только инструменты командной строки)

В данном примере предполагается использование ОС Linux. Возможно, вам придется скорректировать синтаксис для своей платформы.

Настройка Android SDK

После распаковки релиза SDK:

1. Установите дополнительные пакеты с помощью менеджера SDK. Не используйте `android update sdk --no-ui`, как указано в поставляемом в комплекте Readme.txt; он загружает около 30 Гбайт ненужных файлов. Вместо этого используйте интерактивный SDK-менеджер `android sdk`, чтобы получить рекомендуемый минимум пакетов.

2. Добавьте в PATH следующие каталоги JDK и SDK. Это необязательно, но в приведенных ниже инструкциях это предполагается.

- `JDK/bin`
- `SDK/platform-tools`
- `SDK/tools`
- `SDK/build-tools/LATEST (как установлено в шаге 1)`

3. Создайте виртуальное устройство Android. Используйте интерактивный менеджер AVD (`android avd`). Возможно, придется немного повозиться и поискать советы – инструкции на сайте не всегда помогают. (Вы также можете использовать свое собственное устройство).

4. Запустите устройство:

```
emulator -avd DEVICE
```

5. Если экран устройства окажется заблокированным, проведите пальцем по экрану, чтобы разблокировать его. Оставьте его разблокированным, пока вы работаете с кодом.

Работа с кодом приложения

- Перейдите в пустой рабочий каталог.
- Создайте исходный файл:

```
mkdir --parents src/dom/domain
touch src/dom/domain/SayingHello.java
```

Содержание:

```
package dom.domain;
import android.widget.TextView;
public final class SayingHello extends android.app.Activity {
    protected @Override void onCreate( final android.os.Bundle activityState ) {
        super.onCreate( activityState );
        final TextView textV = new TextView( SayingHello.this );
        textV.setText("Hello world" );
        setContentView( textV );
    }
}
```

- Добавьте манифест:

```
touch AndroidManifest.xml
```

Содержание:

```
<?xml version='1.0'?>
<manifest xmlns:a='http://schemas.android.com/apk/res/android' package='dom.domain'
a:versionCode='0' a:versionName='0'>
    <application a:label='Saying hello'>
        <activity a:name='dom.domain.SayingHello'>
            <intent-filter>
                <category a:name='android.intent.category.LAUNCHER' />
                <action a:name='android.intent.action.MAIN' />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

- Создайте подкаталог для объявленных ресурсов:

```
mkdir res
```

Оставьте его пока пустым.

Сборка

- Сгенерируйте исходный код для объявления ресурсов. Подставьте правильный путь к вашей SDK и установленный API для сборки (например, "android-23"):

```
aapt package -f \\
-I SDK/platforms/android-API/android.jar \\
-J src -m \\
-M AndroidManifest.xml -S res -v
```

Объявление ресурсов (описанное ниже) фактически является необязательным. В то же время приведенный выше вызов ничего не дает, если res/ по-прежнему пуст.

2. Скомпилируйте исходный код в байт-код Java (.java → .class):

```
javac \\ 
 -bootclasspath SDK/platforms/android-API/android.jar \\ 
 -classpath src -source 1.7 -target 1.7 \\ 
 src/dom/domain/*.java
```

3. Переведите байт-код с языка Java на Android (.class → .dex):

Сначала используйте Jill (.class → .jayce):

```
java -jar SDK/build-tools/LATEST/jill.jar \\ 
 --output classes.jayce src
```

Затем Jack (.jayce → .dex):

```
java -jar SDK/build-tools/LATEST/jack.jar \\ 
 --import classes.jayce --output-dex .
```

Раньше байткод Android назывался “исполняемый код Dalvik” (Dalvik executable code), отсюда термин “dex”.

При желании можно заменить шаги 11 и 12 одним вызовом Jack; он может компилироваться непосредственно из исходного кода Java (.java → .dex). Однако компиляция с помощью javac имеет свои преимущества. Это более известный, лучше документированный и более широко применяемый инструмент.

3. Упакуйте файлы ресурсов, включая манифест:

```
aapt package -f \\ 
 -F app.apkPart \\ 
 -I SDK/platforms/android-API/android.jar \\ 
 -M AndroidManifest.xml -S res -v
```

В результате получается неполный APK-файл (Android application package, пакет приложения для Android).

4. Создайте полный APK с помощью инструмента ApkBuilder:

```
java -classpath SDK/tools/lib/sdklib.jar \\ 
 com.android.sdklib.build.ApkBuilderMain \\ 
 app.apkUnalign \\ 
 -d -f classes.dex -v -z app.apkPart
```

Он предупреждает: “THIS TOOL IS DEPRECATED. See --help for more information (Этот инструмент устарел. Дополнительную информацию см. в разделе --help”). Если при ошибке --help выдается сообщение ArrayIndexOutOfBoundsException, то не передавайте никаких аргументов:

```
java -classpath SDK/tools/lib/sdklib.jar \\ 
 com.android.sdklib.build.ApkBuilderMain
```

Здесь объясняется, что CLI (ApkBuilderMain) устарел в пользу прямого вызова Java API (ApkBuilder).

5. Оптимизируйте выравнивание данных в APK:

```
zipalign -f -v 4 app.apkUnalign app.apk
```

Установка и запуск

0. Установите приложение на устройство Android:

```
adb install -r app.apk
```

1. Запустите приложение:

```
adb shell am start -n dom.domain/.SayingHello
```

Оно должно запуститься и выполнить приветствие.

Вот и все, что нужно для реализации подобной программы при использовании основных средств Android.

Объявление ресурса

Этот раздел не является обязательным для изучения. Для простого приложения "hello world" объявление ресурсов не требуется. Если для вашего приложения они также не требуются, то можно несколько упростить сборку, пропустив шаг 10 и удалив ссылку на каталог res/ из шага 13.

В противном случае вот краткий пример того, как объявить ресурс и как на него ссылаться.

0. Добавить файл ресурсов:

```
mkdir res/values  
touch res/values/values.xml
```

Содержание:

```
<?xml version='1.0'?>  
<resources>  
    <string name='appLabel'>Saying hello</string>  
</resources>
```

1. Сделайте ссылку на ресурс из XML-манифеста. Это декларативный стиль ссылок:

```
<!-- <application a:label='Saying hello'> -->  
    <application a:label='@string/appLabel'>
```

2. Сделайте ссылку на тот же ресурс из исходного кода Java. Это императивная ссылка:

```
// v.setText("Hello world");  
v.setText("This app is called "  
+ getResources().getString( R.string.appLabel ));
```

3. Протестируйте указанные изменения, перестроив, переустановив и запустив приложение заново.

Приложение должно перезапуститься и выдать сообщение: "This app is called Saying hello".

Деинсталляция приложения

```
adb uninstall dom.domain
```

1.4. Основы создания приложений

Приложения для Android пишутся на языке Java. Средствами Android SDK код, данные и файлы ресурсов компилируются в APK (Android package, пакет Android). Как правило, один APK-файл содержит все содержимое приложения.

Каждое приложение работает на собственной виртуальной машине (virtual machine, VM), что позволяет изолировать его от других приложений. Система Android работает по принципу наименьших привилегий. Каждое приложение имеет доступ только к тем компонентам, которые необходимы ему для выполнения работы, и не более того. Однако существуют способы обмена данными с другими приложениями, например, обмен идентификатором пользователя Linux между приложениями, или приложения могут запрашивать разрешение на доступ к данным устройства, таким как SD-карта, контакты и т. д.

Компоненты приложений

Компоненты приложений – это строительные блоки приложения для Android. Каждый компонент играет определенную роль в приложении Android, служит определенной цели и имеет различные жизненные циклы (порядок создания и уничтожения компонента). Вот четыре типа компонентов приложения.

1. **Activities (Активности):** Активность представляет собой отдельный экран с пользовательским интерфейсом (UI). В приложении для Android может быть несколько активностей. (Например, в приложении для работы с электронной почтой одна активность может использоваться для просмотра списка всех писем, другая – для отображения содержимого каждого письма, третья – для создания нового письма.) Все действия в приложении работают вместе, создавая пользовательский опыт (User eXperience, UX).

2. **Services (Службы):** Служба работает в фоновом режиме для выполнения длительных операций или для выполнения работы для удаленных процессов. Служба не предоставляет никакого пользовательского интерфейса, она работает только в фоновом режиме при участии пользователя. (Например, служба может проигрывать музыку в фоновом режиме, пока пользователь находится в другом приложении, или загружать данные из Интернета, не блокируя взаимодействие пользователя с Android-устройством).

3. **Content Providers (Провайдеры контента или поставщики содержимого):** Провайдер контента управляет общими данными приложения. Существует четыре способа хранения данных в приложении: они могут быть записаны в файл и сохранены в файловой системе, вставлены или обновлены в базе данных SQLite, опубликованы в Интернете или сохранены в любом другом постоянном месте хранения, к которому приложение может получить доступ. Через провайдеров контента другие приложения могут запрашивать или даже изменять эти данные. (Например, система Android предоставляет контент-провайдер, который управляет контактной информацией пользователя, так что любое приложение, имеющее разрешение, может запрашивать контакты). Провайдеры контента также могут использоваться для сохранения данных, которые являются конфиденциальными для приложения, для обеспечения большей целостности данных.

4. **Broadcast receivers (Приемники широковещательной передачи или широковещательные приемники):** Такой приемник реагирует на общесистемные передачи объявлений (например, сообщение о выключении экрана, разряде батареи и т. д.) или от приложений (например, чтобы сообщить другим приложениям, что некоторые данные были загружены на устройство и доступны для использования). Широковещательные приемники не имеют пользовательского интерфейса, но они могут отображать уведомления в строке состояния, чтобы предупредить пользователя. Обычно широковещательные приемники используются в качестве шлюза к другим компонентам приложения, состоящим в основном из активностей и служб.

Уникальность системы Android заключается в том, что любое приложение может запустить компонент другого приложения (например, если вы хотите позвонить, отправить SMS, открыть веб-страницу или просмотреть фотографию, есть приложение, которое уже делает это, и ваше приложение может использовать его, вместо того чтобы разрабатывать новую активность для той же задачи).

Когда система запускает компонент, она запускает процесс для этого приложения (если он еще не запущен, то есть в системе Android в каждый момент времени может работать только один процесс переднего плана для каждого приложения) и инстанцирует классы, необходимые для этого компонента. Таким образом, компонент запускается в процессе того приложения, которому он принадлежит. Поэтому, в отличие от приложений на других системах, приложения Android не имеют единственной точки входа (отсутствует метод `main()`).

Поскольку система запускает каждое приложение в отдельном процессе, одно приложение не может напрямую активировать компоненты другого, однако система Android может это сделать. Таким образом, для запуска компонента другого приложения необходимо отправить системе сообщение о намерении запустить этот компонент, после чего система запустит его.

Контекст

Экземпляры класса `android.content.Context` обеспечивают связь с системой Android, на которой выполняется приложение. Экземпляр `Context` необходим для получения доступа к ресурсам проекта и глобальной информации о среде приложения.

Приведем простой для восприятия пример: предположим, вы находитесь в гостинице и хотите поесть. Вы звоните в службу обслуживания номеров и просите принести вам что-нибудь или убрать за вами. Теперь представьте, что отель – это приложение для Android, вы – это активность, а сотрудник службы обслуживания номеров – это ваш контекст, который предоставляет вам доступ к ресурсам отеля, таким как обслуживание номеров, продукты питания и т. д.

Другой пример: вы находитесь в ресторане, сидите за столиком, и за каждым столиком закреплен офицант; когда вы хотите заказать еду, вы обращаетесь к офицанту с просьбой принести ее. Офицант делает заказ, и блюда подаются на ваш стол. В данном примере ресторан – это приложение Android, столики или клиенты – компоненты приложения, блюда – ресурсы приложения, а офицант – это контекст, обеспечивающий доступ к ресурсам, например к блюдам.

Для активации любого из перечисленных компонентов требуется экземпляр контекста. Не только вышеперечисленные, но и практически все системные ресурсы – создание пользовательского интерфейса с помощью представлений (об этом позже), создание экземпляров системных служб, запуск новых активностей или служб – требуют контекста.

Более подробно эта тема раскрыта на ресурсе <http://apoorgvparmar.com/2016/11/15/android-app-development-intro/>.

1.5. Настройка виртуального устройства AVD (Android Virtual Device)

AVD позволяет моделировать реальные устройства и тестировать приложения без реального устройства. В документации для разработчиков Android (<https://developer.android.com/studio/run/managing-avds.html>) говорится следующее:

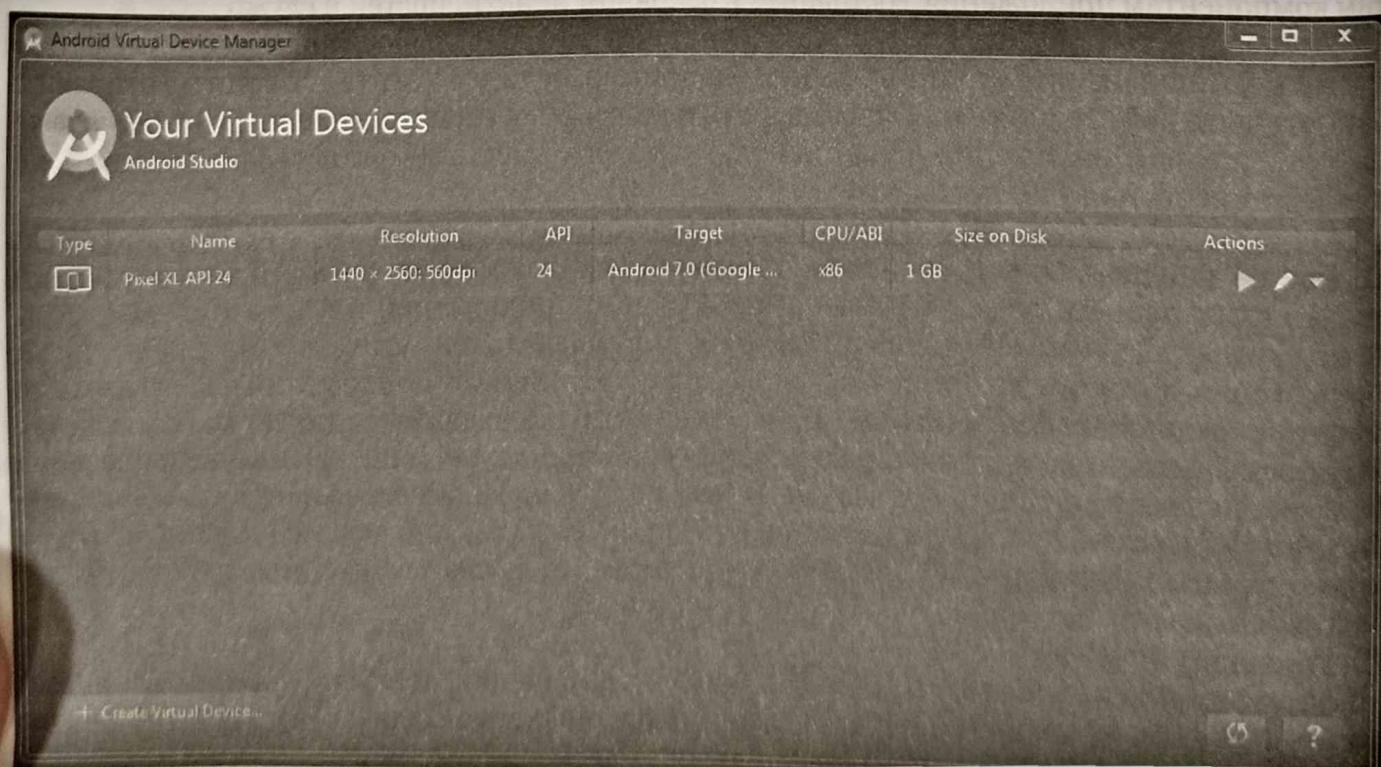
Определение виртуального устройства Android (AVD) позволяет определить характеристики телефона, планшета, устройства Android Wear или Android TV, которые необходимо смоделировать в эмуляторе Android. Менеджер AVD Manager позволяет легко создавать и управлять AVD.

Чтобы настроить AVD, выполните следующие действия.

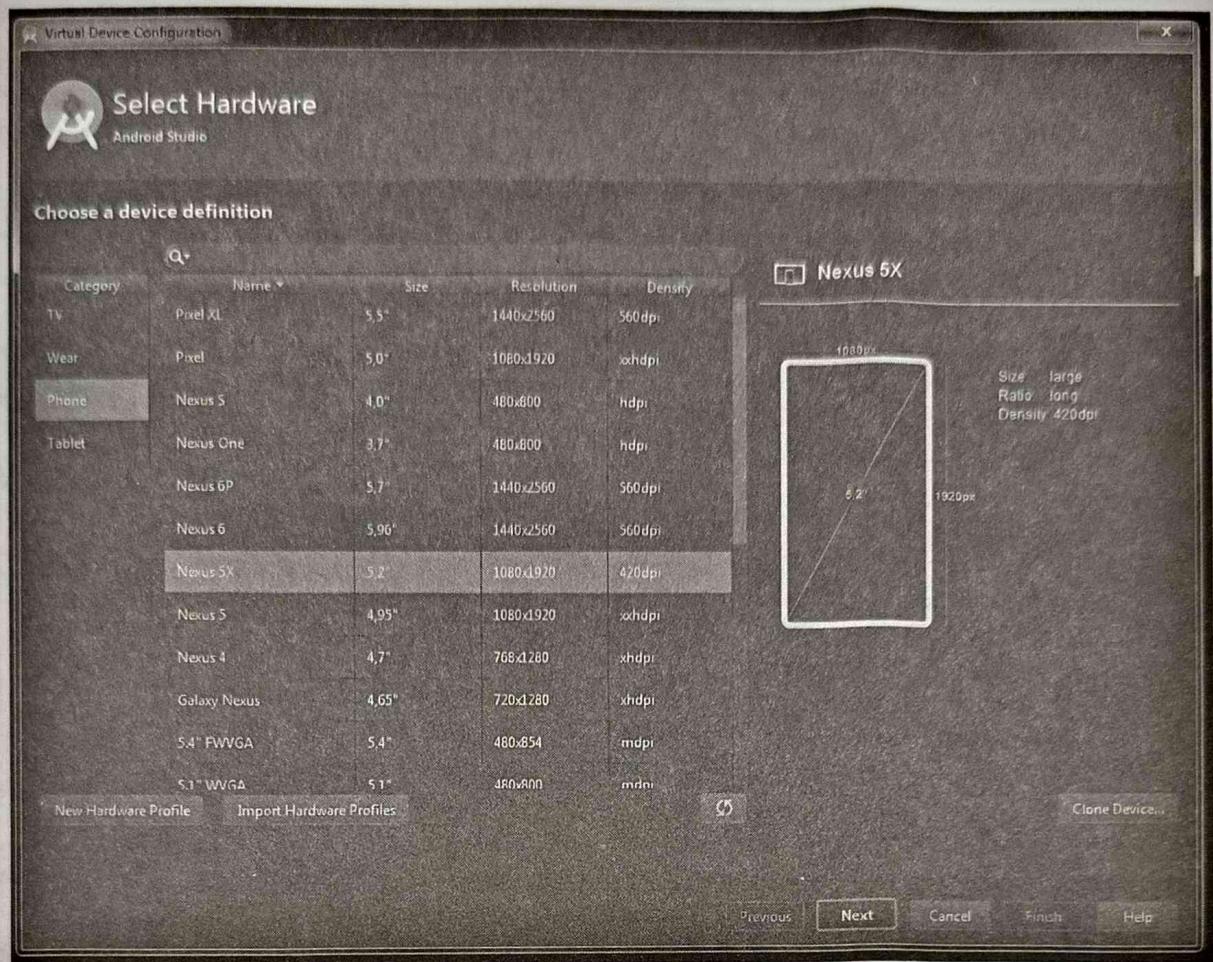
1. Щелкните на этой кнопке, чтобы вызвать менеджер AVD Manager:



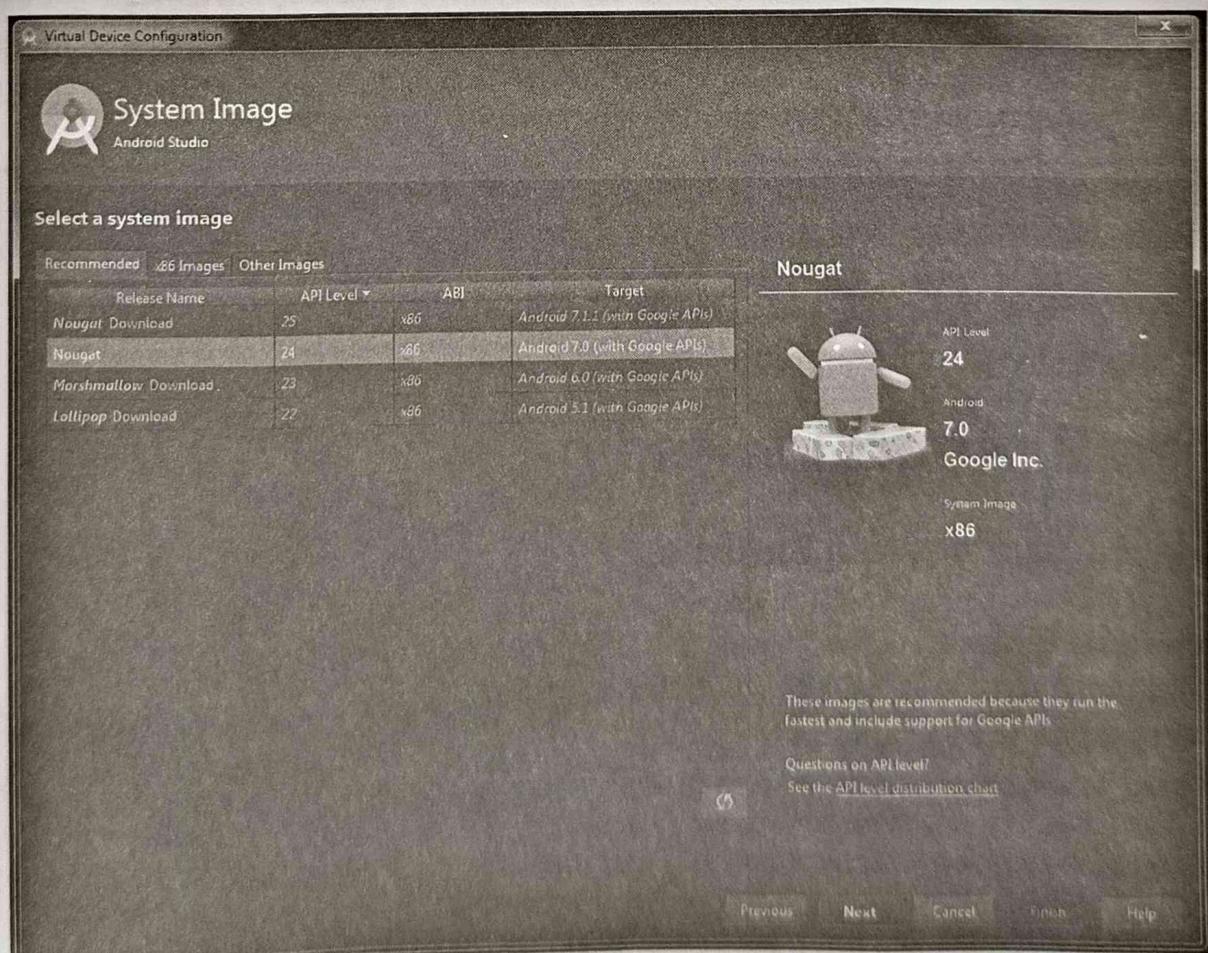
2. Должно появиться диалоговое окно следующего вида:



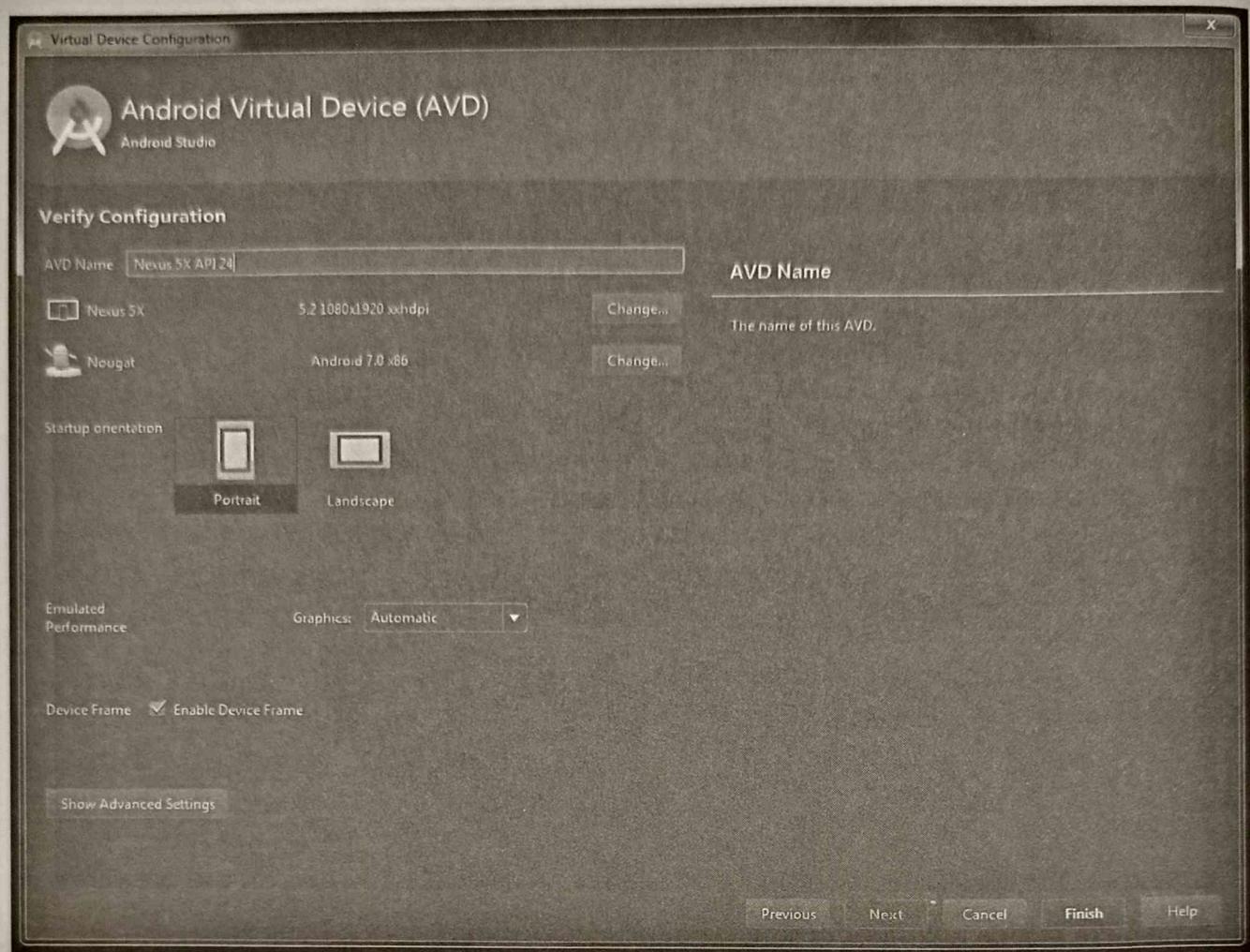
3. Теперь нажмите кнопку + Create Virtual Device... . В результате появится диалог конфигурации виртуального устройства:



4. Выберите любое нужное устройство и нажмите кнопку Next (Далее):



5. Здесь необходимо выбрать версию Android для эмулятора. Возможно, сначала потребуется загрузить ее, нажав кнопку Download. После выбора версии нажмите кнопку Next.



Здесь введите имя эмулятора, его начальную ориентацию, а также укажите, нужно ли отображать рамку вокруг него. После того как все это выбрано, нажмите кнопку Finish.

Теперь у вас есть новый AVD, готовый к запуску на нем приложений.

Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
□ Nexus 5X API 24	Nexus 5X API 24	1080 × 1920; 420dpi	24	Android 7.0 (Google ...)	x86	650 MB	▶ ⚙

Глава 2. Android Studio

2.1. Настройка Android Studio

Системные требования

- Microsoft® Windows® 8/7/Vista/2003 (32 или 64 бит).
- Mac® OS X® 10.8.5 или выше
- Среда рабочего стола GNOME или KDE

Установка

Windows

1. Скачайте и установите JDK (Java Development Kit) с сайта <http://www.oracle.com>.

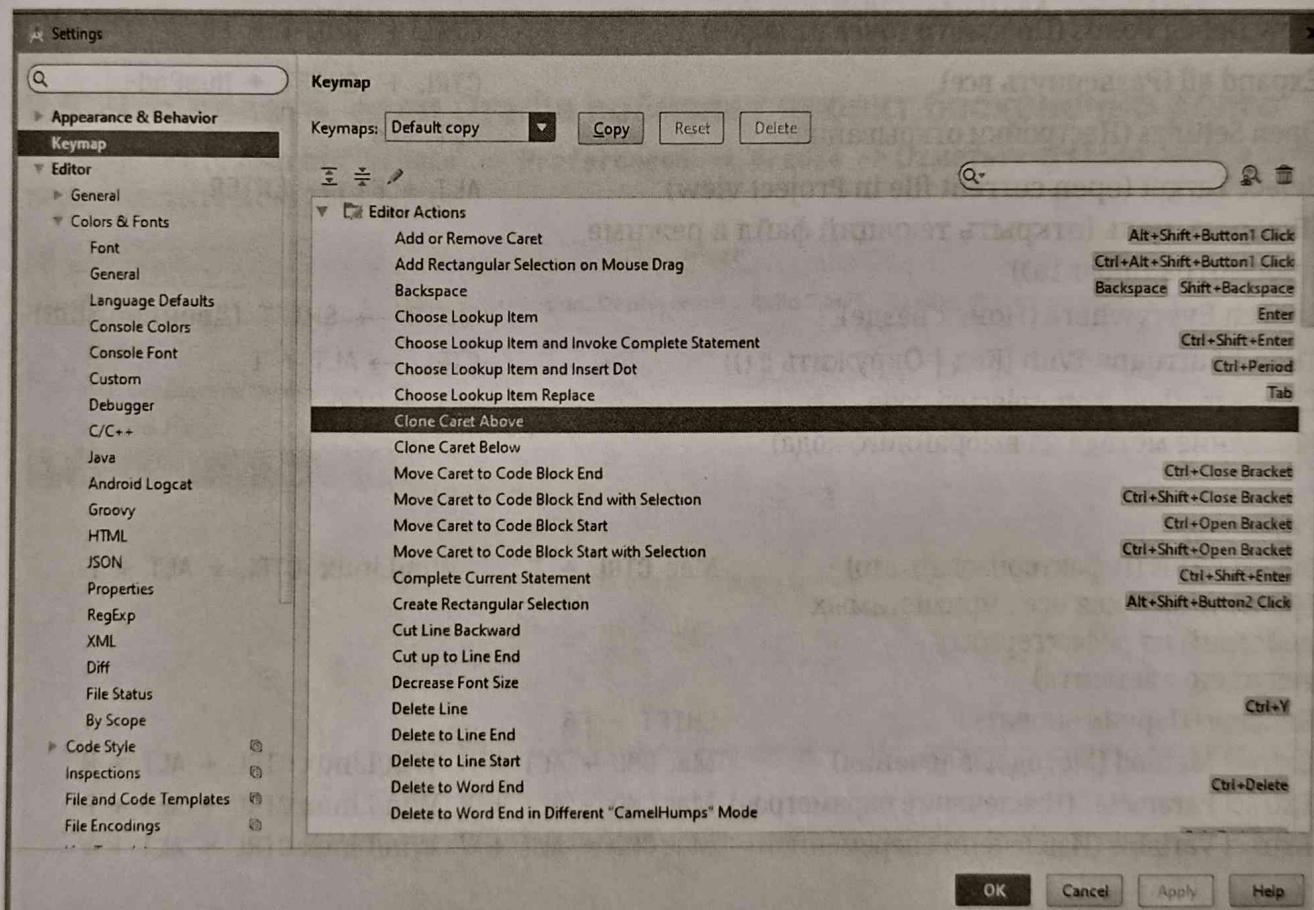
2. Скачайте Android Studio с <https://developer.android.com/studio/index.html>.
3. Запустите файл `Android Studio.exe`, затем укажите путь к JDK и загрузите последнюю версию SDK.

Linux

1. Скачайте и установите JDK (Java Development Kit) с сайта <http://www.oracle.com>.
2. Скачайте Android Studio с <https://developer.android.com/studio/index.html>.
3. Распакуйте zip-файл.
4. Откройте терминал, перейдите в распакованную папку, перейдите в папку `bin` (например `cd android-studio/bin`).
5. Запустите `./studio.sh`.

2.2. Просмотр и добавление ярлыков в Android Studio

Если перейти в меню `Settings >> Кеумар`, то появится окно, в котором будут показаны все действия редактора с их названиями и ярлыками. Некоторые действия редактора не имеют ярлыков. Поэтому щелкните на них правой кнопкой мыши и добавьте новый ярлык. См. рисунок ниже:



2.3. Полезные сочетания клавиш в Android Studio

Ниже приведены наиболее распространенные и полезные сочетания клавиш. Они основаны на стандартной карте сочетаний клавиш IntelliJ. Переключение на другие распространенные карты IDE осуществляется через меню `File -> Settings -> Кеумар -> <Выберите Eclipse/Visual Studio или другое из выпадающего списка>`.

Действие	Клавиатурное сокращение
Format code (Форматировать код)	CTRL + ALT + L
Add unimplemented methods (Добавить нереализованные методы)	CTRL + I
Show logcat (Показать logcat)	ALT + 6
Build (Создать)	CTRL + F9
Build and Run (Создать и запустить)	CTRL + F10
Find (Найти)	CTRL + F
Find in project (Найти в проекте)	CTRL + SHIFT + F
Find and replace (Поиск и замена)	CTRL + R
Find and replace in project (Поиск и замена в проекте)	CTRL + SHIFT + R
Override methods (Методы переопределения)	CTRL + O
Show project (Показать проект)	ALT + 1
Hide project - logcat (Скрыть проект - logcat)	SHIFT + ESC
Collapse all (Свернуть все)	CTRL + SHIFT + NumPad+
View Debug Points (Просмотр точек отладки)	CTRL + SHIFT + F8
Expand all (Развернуть все)	CTRL + SHIFT + NumPad-
Open Settings (Настройки открывания)	ALT + s
Select Target (open current file in Project view) (Выбрать цель (открыть текущий файл в режиме просмотра проекта))	ALT + F1 → ENTER
Search Everywhere (Поиск везде)	SHIFT → SHIFT (Двойной Shift)
Code Surround With (Код Окружить с ())	CTRL → ALT + T
Create method from selected code (Создание метода из выбранного кода)	ALT + CTRL

Рефакторинг:

Refactor This (Рефакторизовать это) (меню/пикер для всех применимых действий по рефакторингу текущего элемента)	Mac CTRL + T	Win/Linux CTRL + ALT + T
Rename (Переименовать)	SHIFT + F6	
Extract Method (Метод извлечения)	Mac CMD + ALT + M	Win/Linux CTRL + ALT + M
Extract Parameter (Извлечение параметров)	Mac CMD + ALT + P	Win/Linux CTRL + ALT + P
Extract Variable (Извлечение переменной)	Mac CMD + ALT + V	Win/Linux CTRL + ALT + V

2.4. Совет по улучшению производительности Android Studio**Разрешите работу offline:**

1. Нажмите File → Settings. Найдите слово “gradle” и щелкните в рабочем поле offline.
2. Перейдите в раздел Compiler (в том же диалоговом окне настроек, чуть ниже Gradle) и добавьте в текстовое поле Command-line Options строку --offline.

Повысьте производительность Gradle

Добавьте следующие две строки кода в файл gradle.properties.

```
org.gradle.daemon=true
org.gradle.parallel=true
```

Увеличьте значения параметров `-Xmx` и `-Xms` в файле `studio.vmoptions`

```
-Xms1024m
-Xmx4096m
-XX:MaxPermSize=1024m
-XX:ReservedCodeCacheSize=256m
-XX:+UseCompressedOops
```

Windows

`%USERPROFILE%.{FOLDER_NAME}\studio.exe.vmoptions` и/или
`%USERPROFILE%.{FOLDER_NAME}\studio64.exe.vmoptions`

Mac

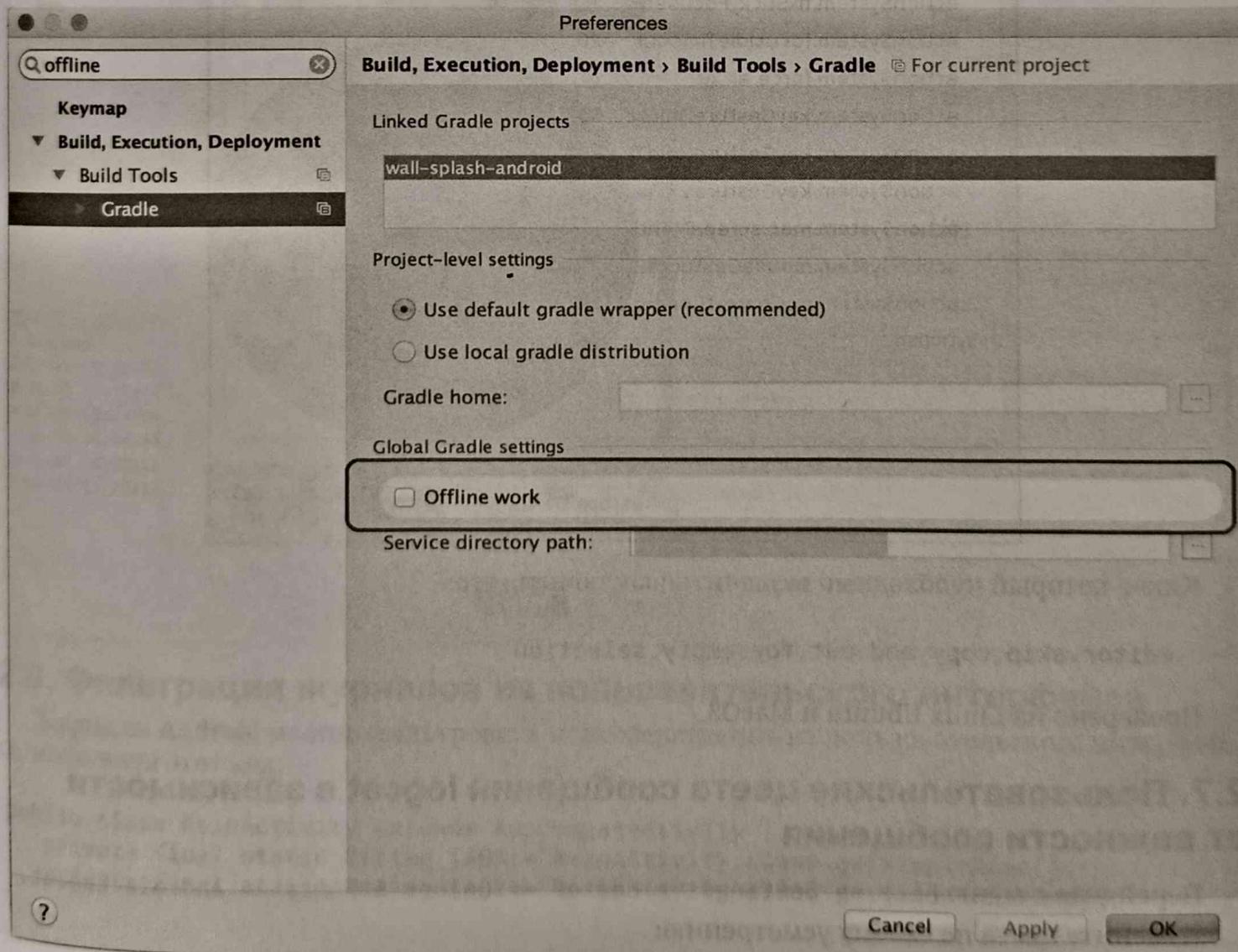
`~/Library/Preferences/{FOLDER_NAME}/studio.vmoptions`

Linux

`~.{FOLDER_NAME}/studio.vmoptions` и/или `~.{FOLDER_NAME}/studio64.vmoptions`

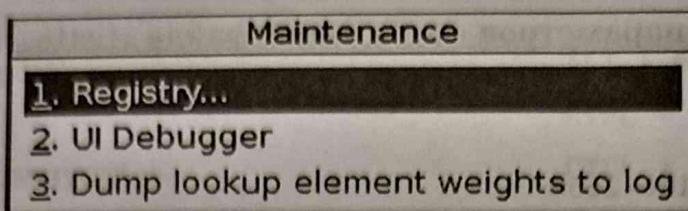
2.5. Что делать, если Gradle собирает проект бесконечно долго

Выполните `Android Studio -> Preferences -> Gradle ->` Отметьте `Offline work`, а затем перезапустите Android-студию.

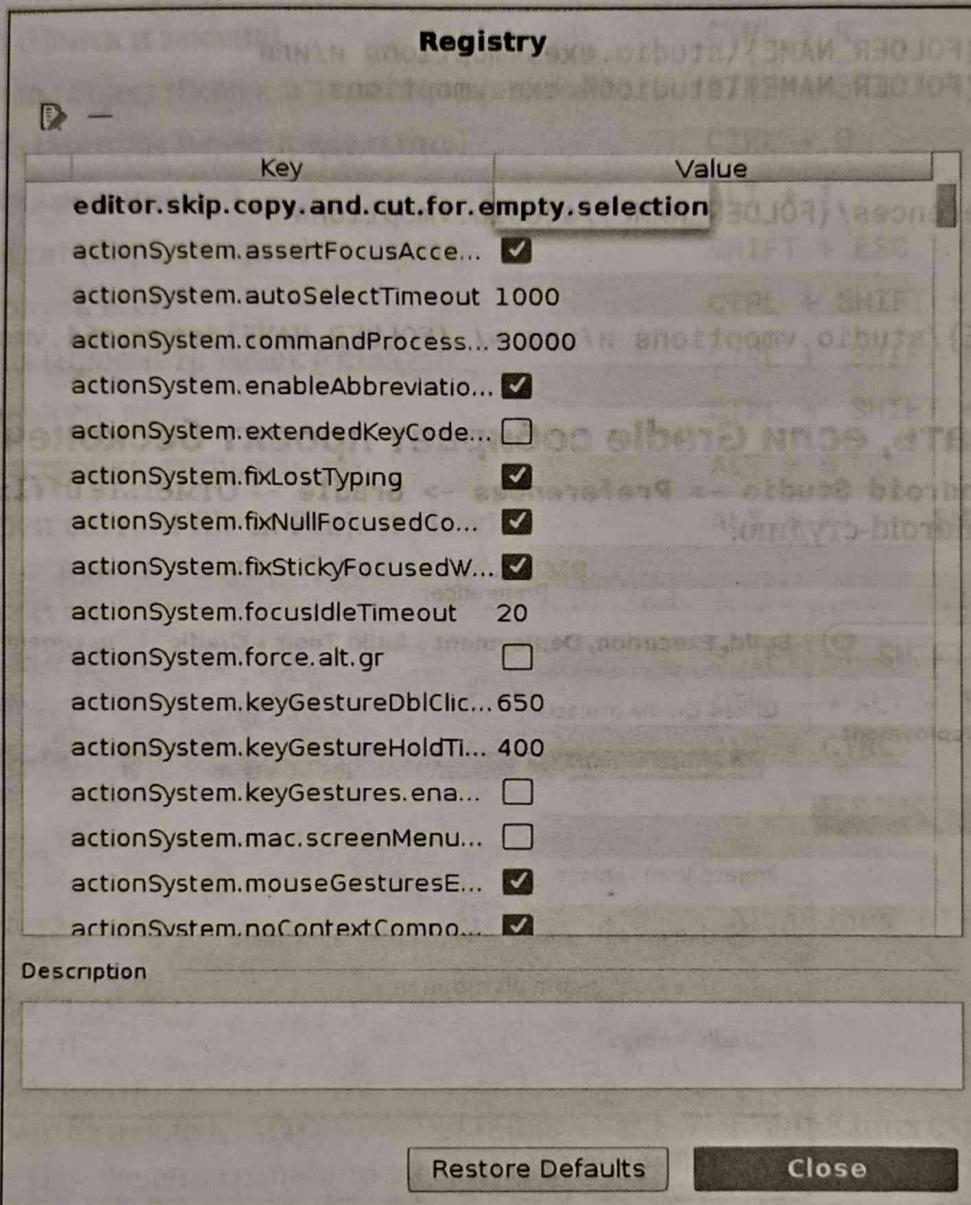


2.6. Включение/выключение копирования пустых строк

После нажатия `ctrl + alt + shift + /` (`cmd + alt + shift + /` в MacOS) должен появиться следующий диалог:



Щелкнув на пункте Registry, вы увидите следующее:



Ключ, который необходимо включить/выключить, это:

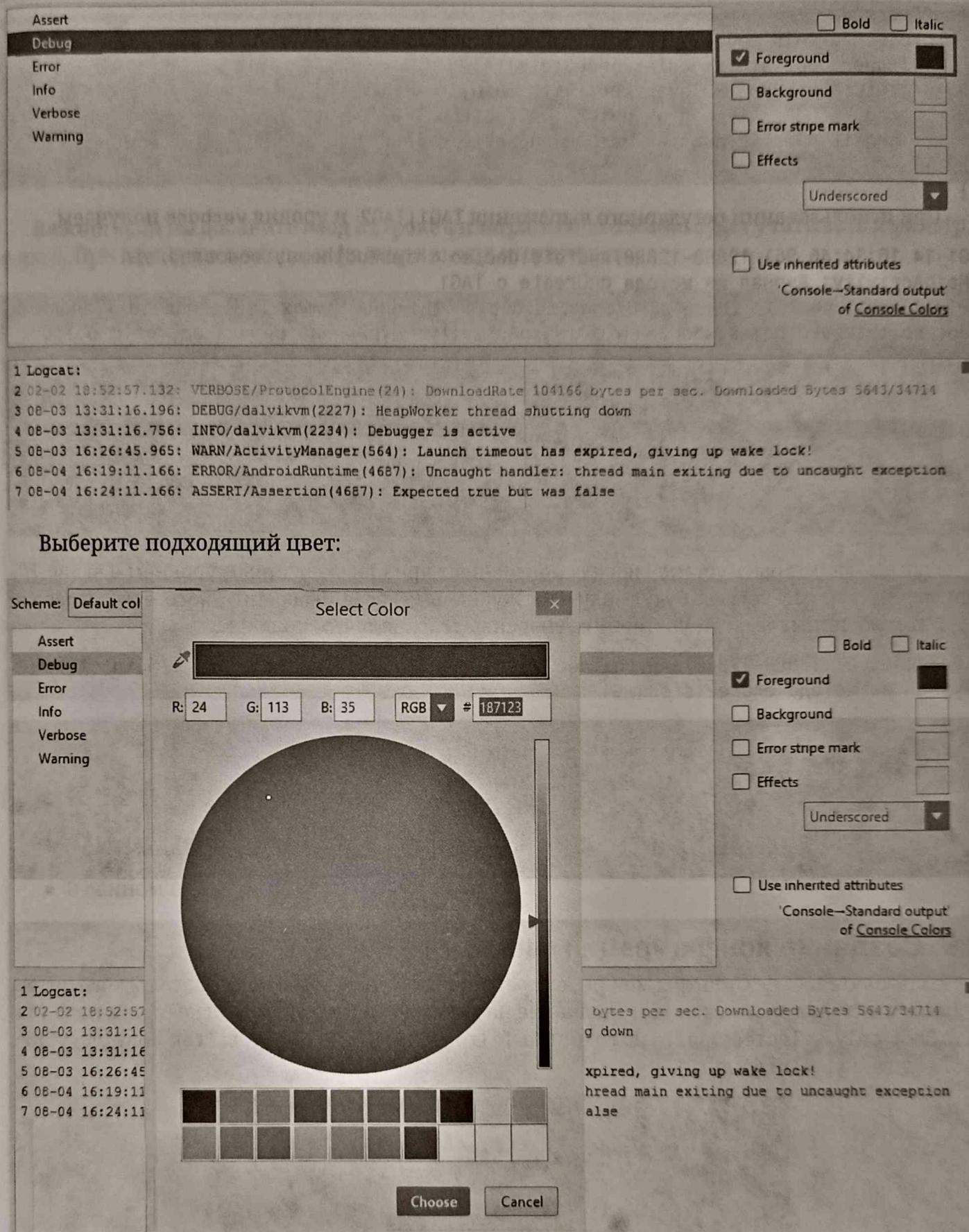
`editor.skip.copy.and.cut.for.empty.selection`

Проверено на Linux Ubuntu и MacOS.

2.7. Пользовательские цвета сообщений logcat в зависимости от важности сообщения

Перейдите в меню `File -> Settings -> Editor -> Colors & Fonts -> Android Logcat`.

Измените цвета по своему усмотрению:



2.8. Фильтрация журналов из пользовательского интерфейса

Журналы Android можно фильтровать непосредственно из пользовательского интерфейса, используя этот код:

```

public class MainActivity extends AppCompatActivity {
    private final static String TAG1 = MainActivity.class.getSimpleName();
    private final static String TAG2 = MainActivity.class.getCanonicalName();

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.e(TAG1, "Журнал из метода onCreate с TAG1");
    Log.i(TAG2, "Журнал из метода onCreate с TAG2");
}
}

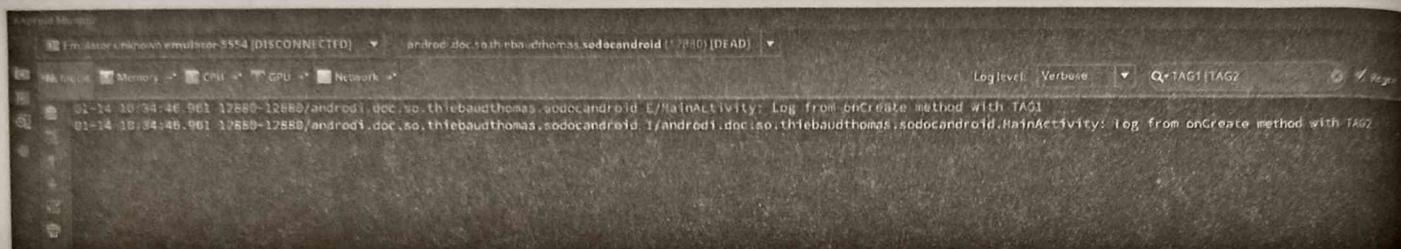
```

При использовании регулярного выражения TAG1 | TAG2 и уровня verbose получаем:

```

01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid E/
MainActivity: Журнал из метода onCreate с TAG1
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid I/android.
doc.so.thiebaudthomas.sodocandroid.MainActivity: Журнал из метода onCreate с TAG2

```



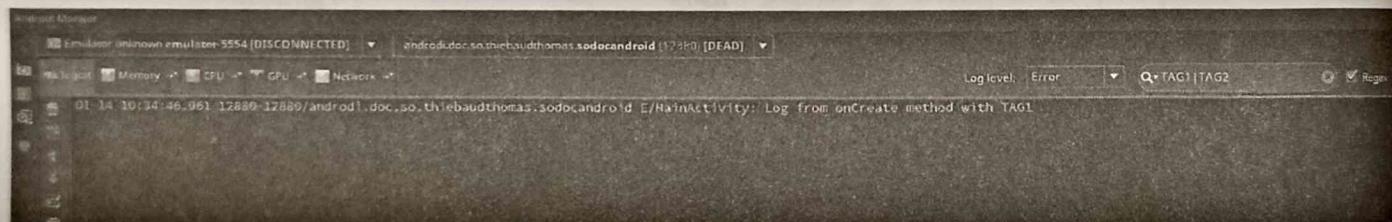
Уровень может быть установлен для получения журналов с заданным уровнем и выше. Например, при уровне verbose будут получены журналы verbose, debug, info, warn, error и assert.

В том же примере, если установить уровень error, получим только:

```

10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid E/MainActivity:
Журнал из метода onCreate с TAG1

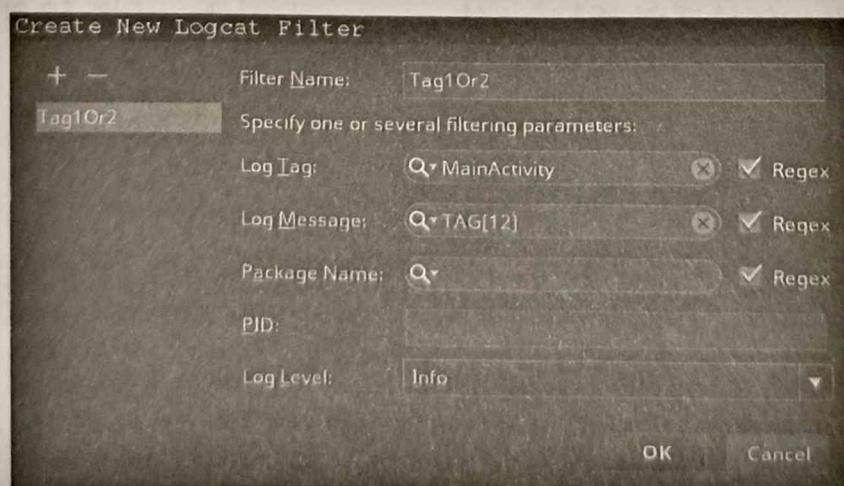
```



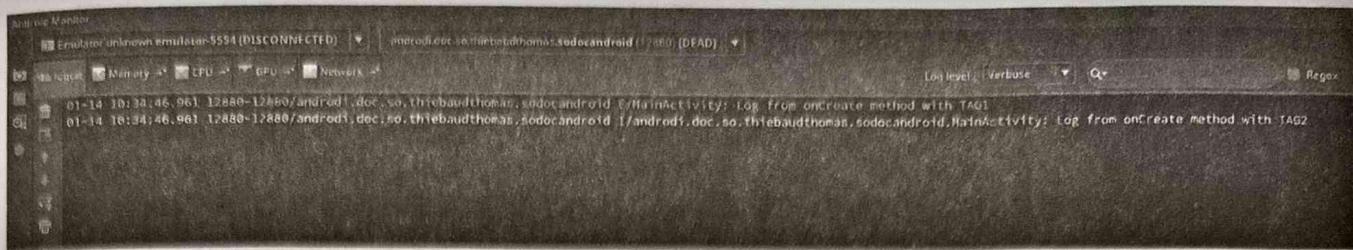
2.9. Создание конфигурации фильтров

Пользовательские фильтры можно настраивать и сохранять из пользовательского интерфейса. На вкладке AndroidMonitor щелкните на правом выпадающем списке (в нем надо указать Show only selected application или No filters) и выберите Edit filter configuration.

Ведите нужный фильтр:



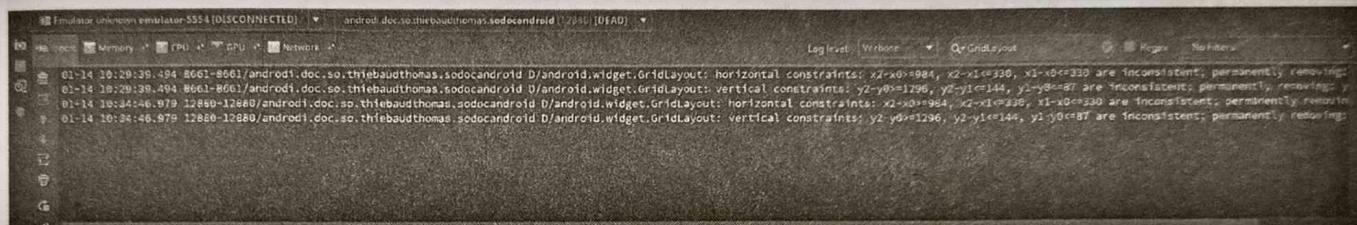
И используйте его (можно выбрать из того же выпадающего списка):



Важно: если вы добавите вход в строке фильтра, Android Studio будет учитывать и фильтр, и вход. При наличии и входа, и фильтра выход отсутствует:



В отсутствие фильтра на выходе получаем:



2.10. Создание папки ресурсов (assets)

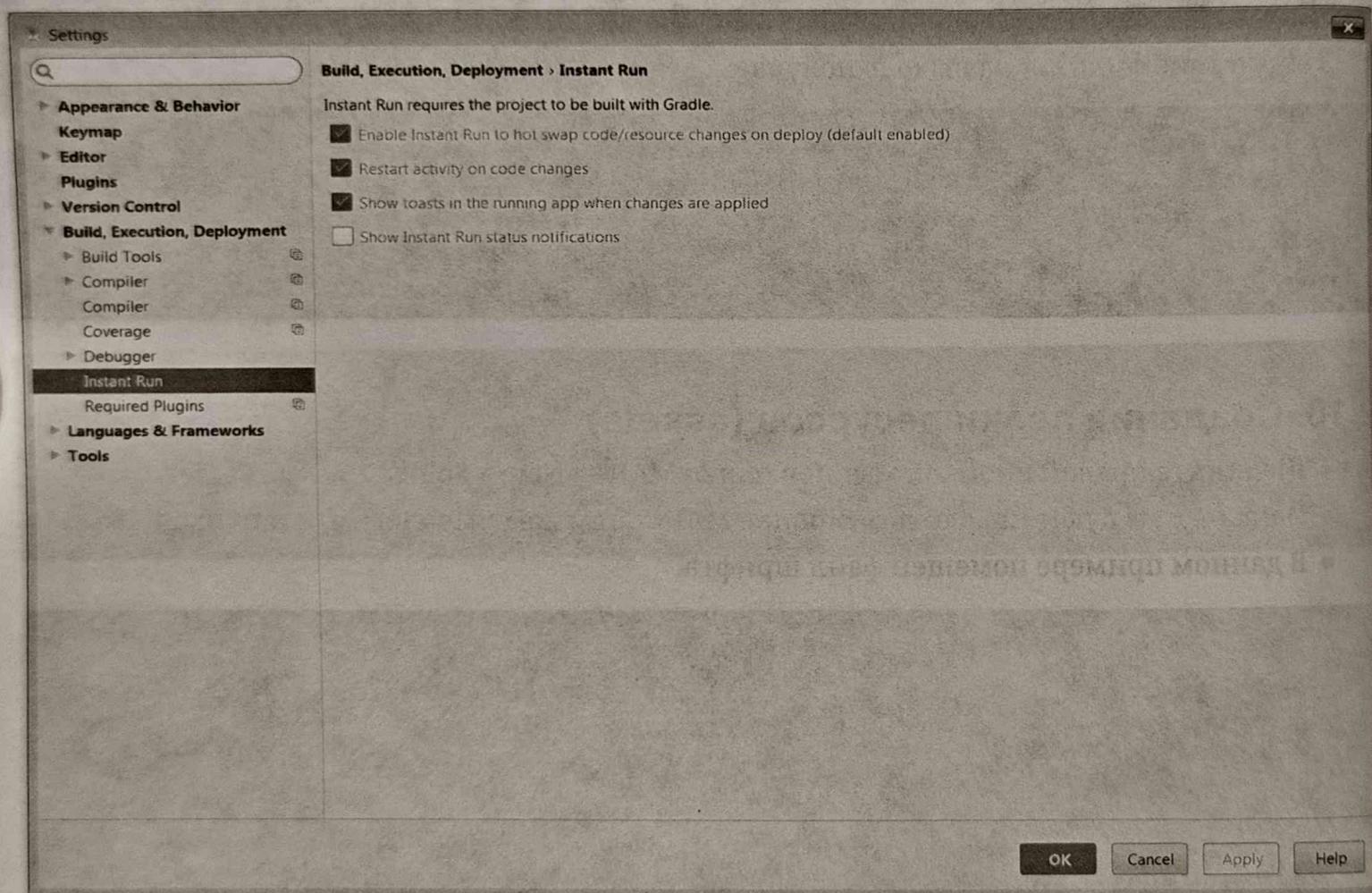
- Щелкните правой кнопкой мыши в папке MAIN > New > Folder > Assets Folder.
- Папка Assets будет находиться в папке MAIN с тем же символом, что и папка RES.
- В данном примере помещен файл шрифта.



Глава 3. Мгновенный запуск в Android Studio

3.1. Включение и отключение функции мгновенного запуска Instant Run

1. Откройте диалог “Settings” или “Preferences”:
 - В операционной системе Windows или Linux выберите в главном меню пункт **File > Settings**.
 - В Mac OSX в главном меню выберите **Android Studio > Preferences**.
2. Перейдите в раздел **Build, Execution, Deployment > Compiler**.
3. В текстовом поле рядом с пунктом **Command-line Options** введите параметры командной строки.
4. Нажмите кнопку **OK** для сохранения и выхода.



Верхняя опция – Instant run (Мгновенный запуск). Установите или снимите этот флагок. Документация по мгновенному запуску: <https://developer.android.com/studio/run/index.html#instant-run>.

3.2. Типы замен кода в мгновенном запуске

Существует три типа замены кода (code swaps), которые мгновенный запуск позволяет использовать для ускорения отладки и запуска приложений из вашего кода в Android Studio.

- Горячая замена (Hot Swap)
- Теплая замена (Warm Swap)
- Холодная замена (Cold Swap)

Когда срабатывает каждая из этих замен?

Горячая замена срабатывает при изменении реализации существующего метода.

Теплая замена срабатывает при изменении или удалении существующего ресурса (всего, что находится в папке res).

Холодная замена срабатывает всякий раз, когда в коде вашего приложения происходят структурные изменения, например:

1. Добавление, удаление или замена:

- аннотации
- поля экземпляра
- статического поля
- сигнатуры статического метода
- сигнатуры метода экземпляра

2. Изменение, от какого родительского класса наследует текущий класс.

3. Изменение списка реализованных интерфейсов.

4. Изменение статического инициализатора класса.

5. Упорядочивание элементов макета, использующих динамические идентификаторы ресурсов.

Что происходит при замене кода?

Изменения при горячей замене видны мгновенно – как только происходит очередной вызов метода, реализация которого изменилась.

Теплая замена перезапускает текущую работу.

Холодная замена перезапускает все приложение (без переустановки).

3.3. Неподдерживаемые изменения кода при использовании мгновенного запуска

Есть несколько изменений, при которых мгновенный запуск не справится со своей задачей и потребуется полная сборка и переустановка приложения, как это было до появления Instant Run.

1. Изменение манифеста приложения.
2. Изменение ресурсов, на которые ссылается манифест приложения.
3. Изменение элемента пользовательского интерфейса виджета Android (требуются Clean и Rerun (очистка и повторный запуск)).

Документация по мгновенному запуску: <https://developer.android.com/studio/run/index.html#instant-run>.

Глава 4. TextView

В этой главе рассматривается все, что связано с настройкой TextView в Android SDK.

4.1. Расширяемое TextView

В Android для выделения определенного участка текста другим цветом, стилем, размером и/или событием щелчка (нажатия) в одном виджете TextView можно использовать расширяемое (Spannable) TextView.

Допустим, мы определили TextView следующим образом:

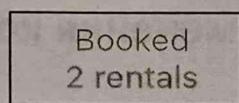
```
TextView textview=findViewById(R.id.textview);
```

Затем к нему можно применить различное выделение, как показано ниже:

- **Spannable color** – для того чтобы задать другой цвет некоторой части текста, можно использовать ForegroundColorSpan, как показано в следующем примере:

```
Spannable spannable = new SpannableString(firstWord+lastWord);
spannable.setSpan(new ForegroundColorSpan(firstWordColor), 0, firstWord.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
spannable.setSpan(new ForegroundColorSpan(lastWordColor), firstWord.length(),
firstWord.length()+lastWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
textview.setText( spannable );
```

Вывод, созданный приведенным выше кодом:



- **Spannable font** (расширяемый шрифт) – для того чтобы установить различный размер шрифта для некоторой части текста, можно использовать RelativeSizeSpan, как показано в следующем примере:

```
Spannable spannable = new SpannableString(firstWord+lastWord);
spannable.setSpan(new RelativeSizeSpan(1.1f),0, firstWord.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE); // установить размер
spannable.setSpan(new RelativeSizeSpan(0.8f), firstWord.length(), firstWord.
length() + lastWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE); // установка
размера
textview.setText( spannable );
```

Вывод, созданный приведенным выше кодом:



- **Spannable typeface** (расширяемая гарнитура шрифта) – для того чтобы установить другую гарнитуру шрифта для некоторого фрагмента текста, можно использовать пользовательский TypefaceSpan, как показано в следующем примере:

```
Spannable spannable = new SpannableString(firstWord+lastWord);
spannable.setSpan( new CustomTypefaceSpan("SFUIText-Bold.otf",fontBold), 0,
firstWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
```

4.2. Зачеркнутое TextView

```
spannable.setSpan( new CustomTypefaceSpan("SFUIText-Regular.otf", fontRegular),
firstWord.length(), firstWord.length() + lastWord.length(), Spannable.SPAN_
EXCLUSIVE_EXCLUSIVE);
text.setText( spannable );
```

Однако для того, чтобы приведенный выше код работал, класс `CustomTypefaceSpan` должен быть производным от класса `TypefaceSpan`. Это можно сделать следующим образом:

```
public class CustomTypefaceSpan extends TypefaceSpan {
    private final Typeface newType;

    public CustomTypefaceSpan(String family, Typeface type) {
        super(family);
        newType = type;
    }

    @Override
    public void updateDrawState(TextPaint ds) {
        applyCustomTypeFace(ds, newType);
    }

    @Override
    public void updateMeasureState(TextPaint paint) {
        applyCustomTypeFace(paint, newType);
    }

    private static void applyCustomTypeFace(Paint paint, Typeface tf) {
        int oldStyle;
        Typeface old = paint.getTypeface();
        if (old == null) {
            oldStyle = 0;
        } else {
            oldStyle = old.getStyle();
        }
        int fake = oldStyle & ~tf.getStyle();
        if ((fake & Typeface.BOLD) != 0) {
            paint.setFakeBoldText(true);
        }
        if ((fake & Typeface.ITALIC) != 0) {
            paint.setTextSkewX(-0.25f);
        }
        paint.setTypeface(tf);
    }
}
```

4.2. Зачеркнутое TextView

Зачеркивание всего текста

```
String sampleText = "Это проверка";
textView.setPaintFlags(tv.getPaintFlags() | Paint.STRIKE_THRU_TEXT_FLAG); textView.
setText(sampleText);
```

Зачеркивание только отдельных частей текста

```
String sampleText = "Это проверка";
SpannableStringBuilder spanBuilder = new SpannableStringBuilder(sampleText);
```

```

StrikethroughSpan strikethroughSpan = new StrikethroughSpan();
spanBuilder.setSpan(
    strikethroughSpan, // Добавляемый интервал
    0, // Старт
    4, // Конец интервала (включая значение)
    Spanned.SPAN_EXCLUSIVE_EXCLUSIVE // Изменения текста не будут отражаться
);
textView.setText(spanBuilder);

```

4.3. TextView с изображением

Android позволяет программистам размещать изображения по всем четырем углам TextView. Например, если вы создаете поле с TextView и в то же время хотите показать, что поле можно редактировать, то разработчики обычно размещают рядом с ним значок редактирования. Android предоставляет нам интересную возможность под названием **compound drawable** для TextView:

```

<TextView
    android:id="@+id/title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:drawablePadding="4dp"
    android:drawableRight="@drawable/edit"
    android:text="Hello world"
    android:textSize="18dp" />

```

Установить drawable-ресурс на любую сторону TextView можно следующим образом:

```

    android:drawableLeft="@drawable/edit"
    android:drawableRight="@drawable/edit"
    android:drawableTop="@drawable/edit"
    android:drawableBottom="@drawable/edit"

```

Установка drawable-ресурса также может быть выполнена программно следующим образом:

```

yourTextView.setCompoundDrawables(leftDrawable, rightDrawable, topDrawable,
bottomDrawable);

```

Установка любого из параметров, передаваемых в `setCompoundDrawables()`, в значение `null` приведет к удалению пиктограммы с соответствующей стороны TextView.

4.4. Выравнивание RelativeSizeSpan по верхнему краю

Для того чтобы сделать RelativeSizeSpan выровненным по верхнему краю, можно создать собственный класс, производный от класса SuperscriptSpan. В следующем примере производный класс назван TopAlignSuperscriptSpan:

activity_main.xml:

```

<TextView
    android:id="@+id/txtView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="26sp" />

```

MainActivity.java:

```

TextView txtView = (TextView) findViewById(R.id.txtView);

```

```
SpannableString spannableString = new SpannableString("RM123.456");
spannableString.setSpan( new TopAlignSuperscriptSpan( (float)0.35 ), 0, 2, Spanned.
SPAN_EXCLUSIVE_EXCLUSIVE );
txtView.setText(spannableString);
```

TopAlignSuperscriptSpan.java:

```
private class TopAlignSuperscriptSpan extends SuperscriptSpan {
    //разделить надстрочный индекс на данное число
    protected int fontScale = 2;

    // значение сдвига, от 0 до 1,0
    protected float shiftPercentage = 0;
    //не смещается
    TopAlignSuperscriptSpan() {}

    //устанавливает процент сдвига
    TopAlignSuperscriptSpan( float shiftPercentage ) {
        if( shiftPercentage > 0.0 && shiftPercentage < 1.0 )
            this.shiftPercentage = shiftPercentage;
    }

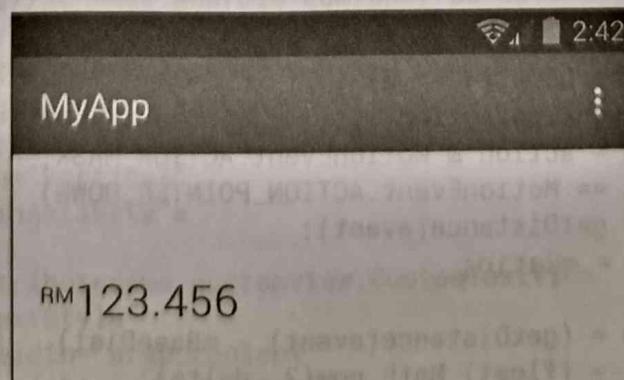
    @Override
    public void updateDrawState( TextPaint tp ) {
        //Оригинальный подъем шрифта
        float ascent = tp.ascent();

        //уменьшить шрифт
        tp.setTextSize( tp.getTextSize() / fontScale );

        //получение нового подъема шрифта
        float newAscent = tp.getFontMetrics().ascent;

        //перемещаем базовую линию в верхнюю часть старого шрифта, затем уменьшаем
        //размер нового шрифта
        //корректировка ошибок с помощью процента сдвига
        tp.baselineShift += ( ascent - ascent * shiftPercentage )
            - (newAscent - newAscent * shiftPercentage );
    }

    @Override
    public void updateMeasureState( TextPaint tp ) {
        updateDrawState( tp );
    }
}
```

Скриншот:

4.5. Pinchzoom на TextView

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/mytv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="Это мой пример текста для демонстрации pinch zoom, вы можете
        увеличивать и уменьшать масштаб с помощью pinch zoom, спасибо" />

</RelativeLayout>
```

MainActivity.java:

```
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.TextView;

public class MyTextViewPinchZoomClass extends Activity implements OnTouchListener {

    final static float STEP = 200;
    TextView mytv;
    float mRatio = 1.0f;
    int mBaseDist;
    float mBaseRatio;
    float fontsize = 13;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mytv = (TextView) findViewById(R.id.mytv);
        mytv.setTextSize(mRatio + 13);
    }

    public boolean onTouchEvent(MotionEvent event) {
        if (event.getPointerCount() == 2) {
            int action = event.getAction();
            int pureaction = action & MotionEvent.ACTION_MASK;
            if (pureaction == MotionEvent.ACTION_POINTER_DOWN) {
                mBaseDist = getDistance(event);
                mBaseRatio = mRatio;
            } else {
                float delta = (getDistance(event) - mBaseDist) / STEP;
                float multi = (float) Math.pow(2, delta);
                mytv.setTextSize(mBaseRatio * multi);
            }
        }
    }

    private float getDistance(MotionEvent event) {
        float x = event.getX(0) - event.getX(1);
        float y = event.getY(0) - event.getY(1);
        return (float) Math.sqrt(x * x + y * y);
    }
}
```