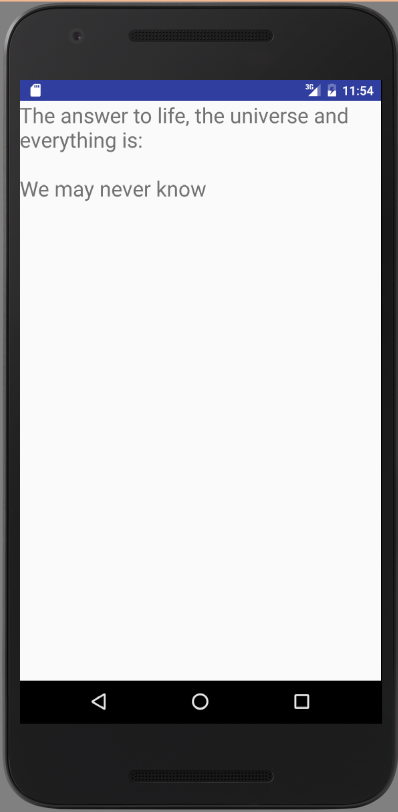
**Инструменты отладки в Android Studio**

Android Studio поставляется с отладчиком, который обеспечивает разработчиков мощными инструментами для анализа внутреннего состояния работающего приложения. И это может быть полезно, когда у вас есть ошибка в приложении, и вы не уверены точно, чем она вызвана. Поэтому полезно использовать отладчик Android Studio для анализа приложения.

Рассмотрим простое Android приложение TheAnswer. Это приложение предназначено для показа ответа на жизнь, вселенную и так далее.

Приложение, которое должно говорить нам, что все 42. Однако, вместо того, чтобы отображать 42, это приложение говорит «we may never know».



Так что очевидно, здесь где-то есть ошибка.

Давайте, используем отладчик для отслеживания этой ошибки. Итак, откроем Android Studio и откроем приложение TheAnswer с исходным кодом:

**package** com.example.vladimir.anserw\_002;  
**import** android.app.Activity;  
**import** android.os.Bundle;  
**import** android.widget.TextView;  
  
**public class** MainActivity **extends** Activity {  
  
 **public static final int**[] ***answers*** = { 42, -10, 0, 100, 1000 };  
 **public static final int *answer*** = 42;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
  
 setContentView(R.layout.***activity\_main***);  
  
 TextView answerView = (TextView) findViewById(R.id.***answer\_view***);  
  
 **int** val = findAnswer();  
 String output = (val == ***answer***) ? **"42"** : **"We may never know"**;  
 answerView  
 .setText(**"The answer to life, the universe and everything**

**is:\n\n"**+ output);  
 }  
  
 **private int** findAnswer() {  
 **for** (**int** val : ***answers***) {  
 **if** (val != ***answer***)  
 **return** val;  
 }  
 **return** -1;  
 }  
}

Теперь, правда, видно, что это довольно простая программа. И ошибка достаточно очевидна, после того как вы посмотрите на код. Но наша цель показать работу отладчика.

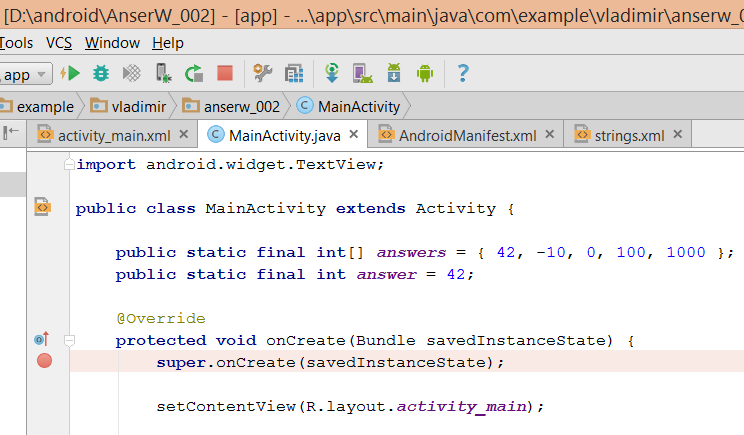
Теперь вы можете видеть, что существует массив *answers*, с помощью которого программа будет искать ответ 42.

В коде есть метод onCreate, который является входной точкой выполнения класса.

Внутри метода onCreate есть вызов метода findAnswer(), который предназначен для поиска в массиве answers числа 42. Программа сохраняет возвращенное методом findAnswer() значение и затем использует его для вычисления строки output. А потом, наконец, программа отображает строку output.

Итак, для поиска ошибки используем отладчик.

И первое, что сделаем, это установим точку остановки, которая будет останавливать программу, когда она будет доходить до метода onCreate.

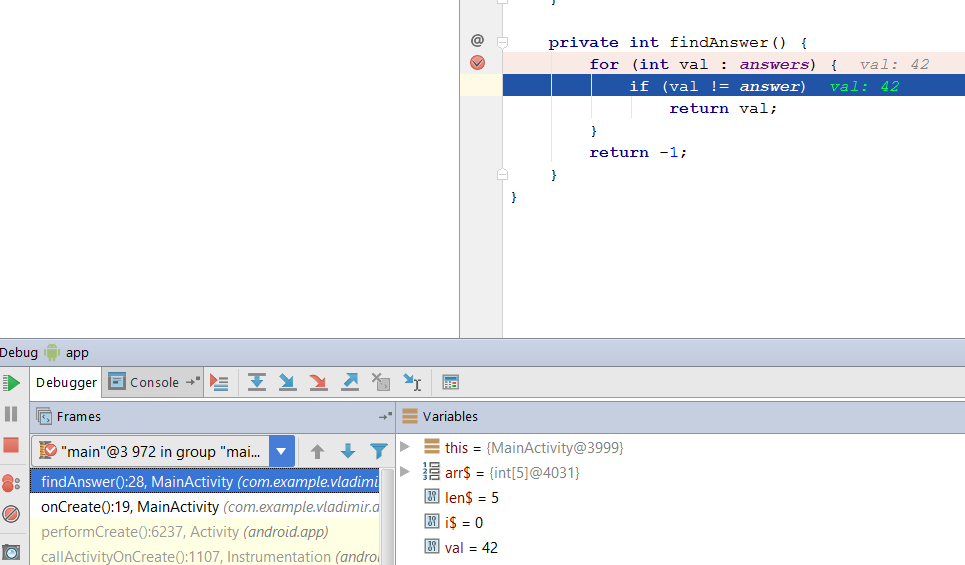


Хотя значение должно быть 42.

Таким образом, проблема в методе findAnswer().

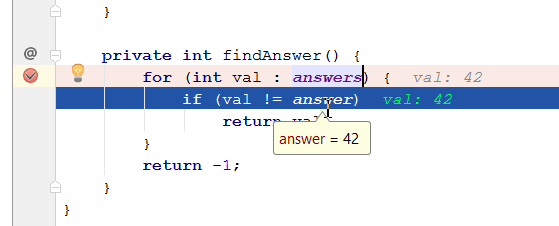
Так что давайте более внимательно изучим метод findAnswer().

А для этого, я поставлю точку остановки на методе findAnswer() и перезапущу приложение.

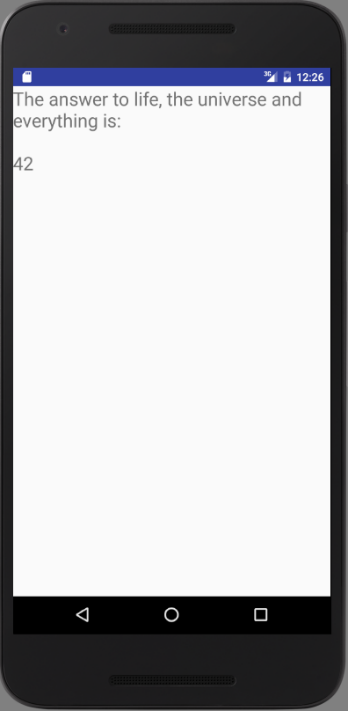


Теперь, когда приложение остановится на onCreate, я нажму кнопку Step Over, и приложение продолжит выполнение до точки остановки findAnswer().

Я могу пройти вперед в этом методе и проверить значения переменных, чтобы определить, что значение 42 действительно есть в массиве.



И, метод видит значение, но считает, что это неправильный ответ.



Теперь мы увидели, что проблема в том, что здесь неправильно поставлена проверка значения:

вместо “==” стоит “!=”.

Так что исправим ошибку и начнем все сначала, и увидим теперь правильное поведение.

Теперь, просто чтобы убедиться, давайте вернемся и запустим приложение.

Я думаю, отладчик хорошо помогает, когда вы имеете достаточно хорошее представление о том, где ваша проблема, так что вы можете сосредоточиться на нескольких строках кода. Но во многих других ситуациях такого рода низко-уровневый, по строковый, анализ кода менее полезен.

Вместо этого проще использовать более общие инструменты мониторинга системы, которые позволяют контролировать поведение системы, как она работает в течение долгого времени.

***Android предоставляет набор таких инструментов мониторинга в своем сервисе Dalvik Debug Monitor Service или DDMS.***

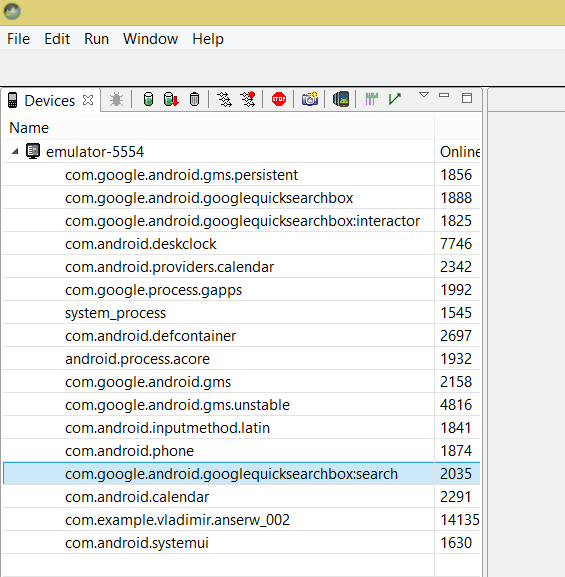
Рассмотрим четыре инструмента из этого набора:

1. File Explorer - для просмотра файловой системы.
2. Logcat - для журналирования и отображения событий среды выполнения.
3. Traceview - для отображения журналов выполнения приложения.
4. Hierarchyview - для анализа GUI-макетов.

Давайте посмотрим на эти инструменты и посмотрим, как получить доступ к ним из Android Studio.

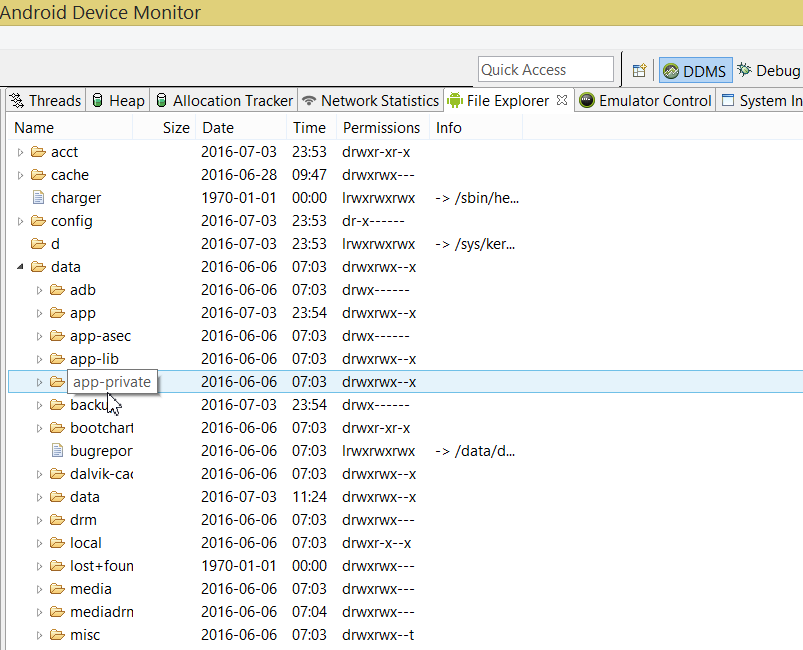
Во-первых, давайте откроем перспективу DDMS, которая предоставляет ряд панелей и кнопок, которые обеспечивают доступ к различным инструментам DDMS (Tools->Android->.

Мы увидим панель устройств, которая показывает устройства, подключенные в данный момент.



А также показывает список процессов, работающих на этих устройствах.

Первый инструмент, который мы будем рассматривать - это File Explorer.

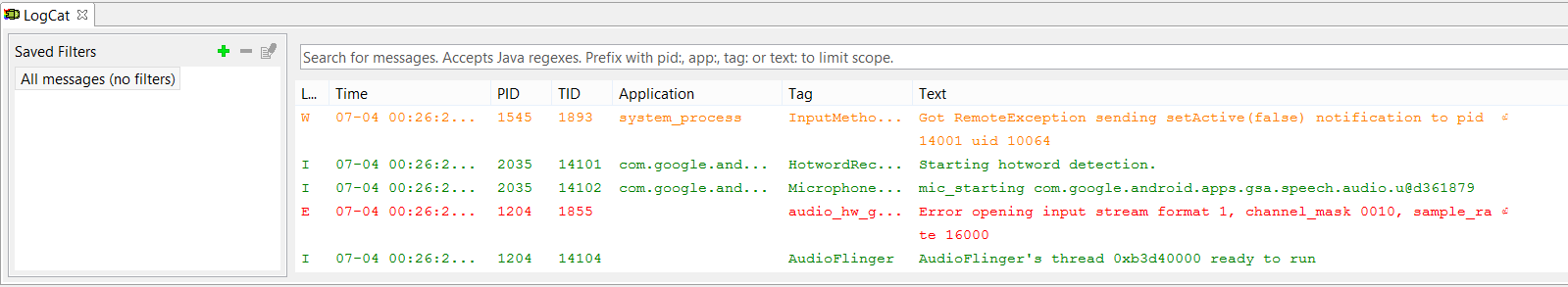


Например, здесь мы видим, что это устройство имеет каталог /data/data, в котором приложения могут хранить свои постоянные данные.

Данное представление показывает файлы в файловой системе.

Это устройство также имеет внешнюю карту памяти, на которой расположены каталоги для хранения музыки и фильмов и других вещей.

Другая панель показывает представление Logcat.



Android регистрирует множество событий по умолчанию. То есть, по мере того как программное обеспечение работает, оно выдает информация о вещах, которые случаются на устройстве, и эта информация сохраняется и она может быть отображена в представлении Logcat.

Android также предоставляет методы, которые позволяют разработчикам журналировать собственную информацию.

Еще один инструмент DDMS - это Traceview.

Traceview позволяет отслеживать методы, которые приложение вызывает.

Давайте сделаем это для приложения TheAnswer.

И чтобы упростить демонстрацию, поставим точку остановки на методе onCreate.

А теперь начнем отладку приложения.

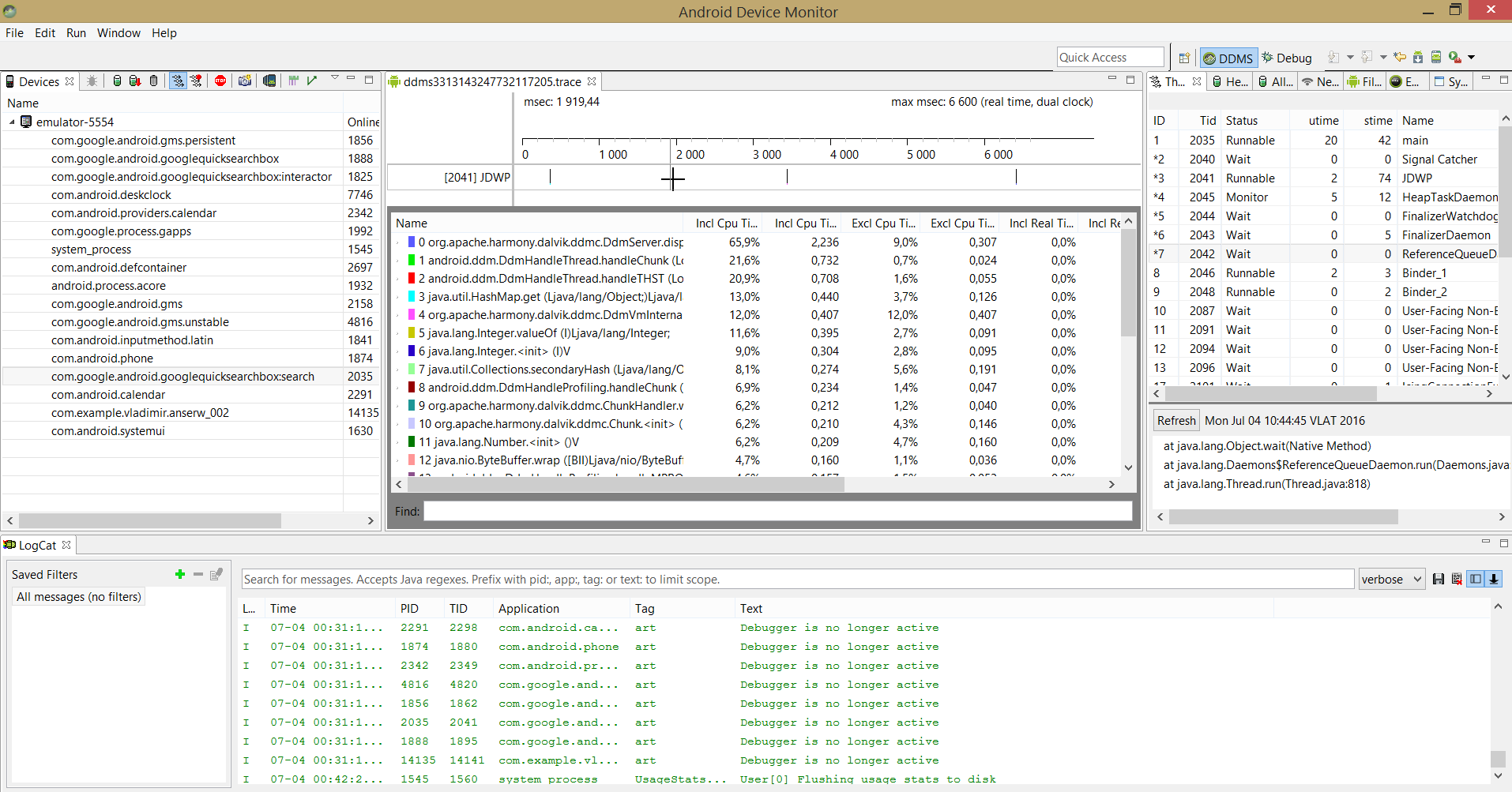
Теперь, как вы можете видеть, приложение остановилось в точке остановки. А теперь, начнем профилирование.

Откроем DDMS View и нажмем на кнопке с надписью Start Profiling.

А теперь вернемся обратно в режим отладки и возобновим выполнение приложения.

Позволим программе отработать некоторое время и вернемся в представление DDMS и нажмем ту же кнопку, на этот раз, чтобы остановить профилирование.

В этот момент Traceview будет обрабатывать информацию трассировки и покажет ее Дисплей Trace view показывает две панели.



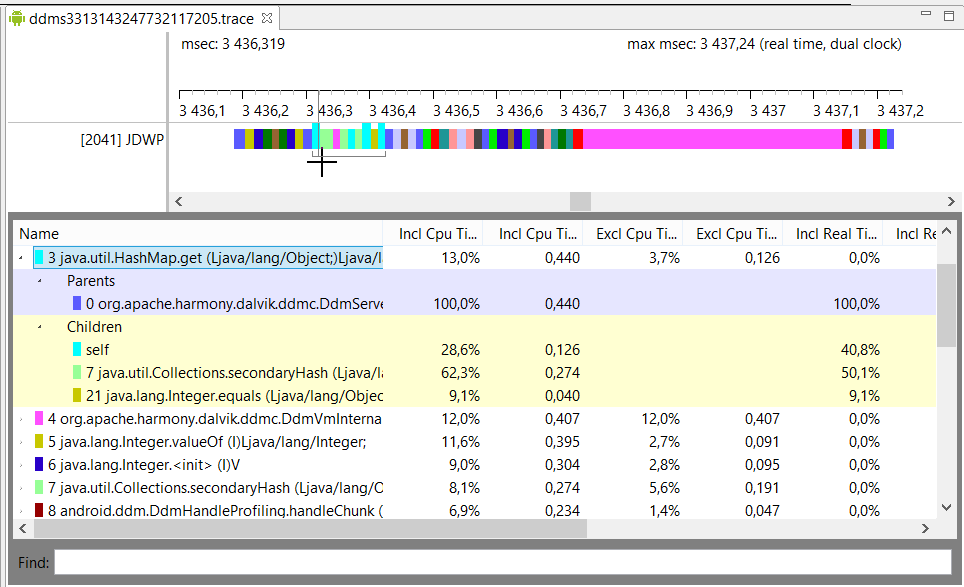
Одна из них (красная обводка) является Timeline представлением, показывающим каждый метод с последовательностью вызова и продолжительностью.

Другая панель (зеленая обводка) - это представление Profile, которое показывает статистическую и другую информацию о каждом вызываемом методе.

Давайте посмотрим на нашего старого знакомого findAnswer().

Когда я нажимаю на findAnswer() в представлении Profile, можно увидеть, где он был вызван в Timeline представлении.

И теперь я могу увеличить в Timeline (выделение участка Timeline при нажатой левой клавише мыши), чтобы увидеть больше информации и окружающий контекст.



И если я хочу быть более точным в моем профилировании, Android также позволяет мне вставить утверждения непосредственно в программу.

Это позволит сделать целевой трассировку.

Однако, чтобы сделать это, необходимо установить некоторые разрешения для вашего приложения.

И мы не говорили о разрешениях пока, но мы будем говорить о них в следующем уроке.

Последний инструмент, который мы будем сегодня рассматривать - это представление UI Hierarchy.

Этот инструмент позволяет анализировать пользовательский интерфейс приложения и понять, как он организован.

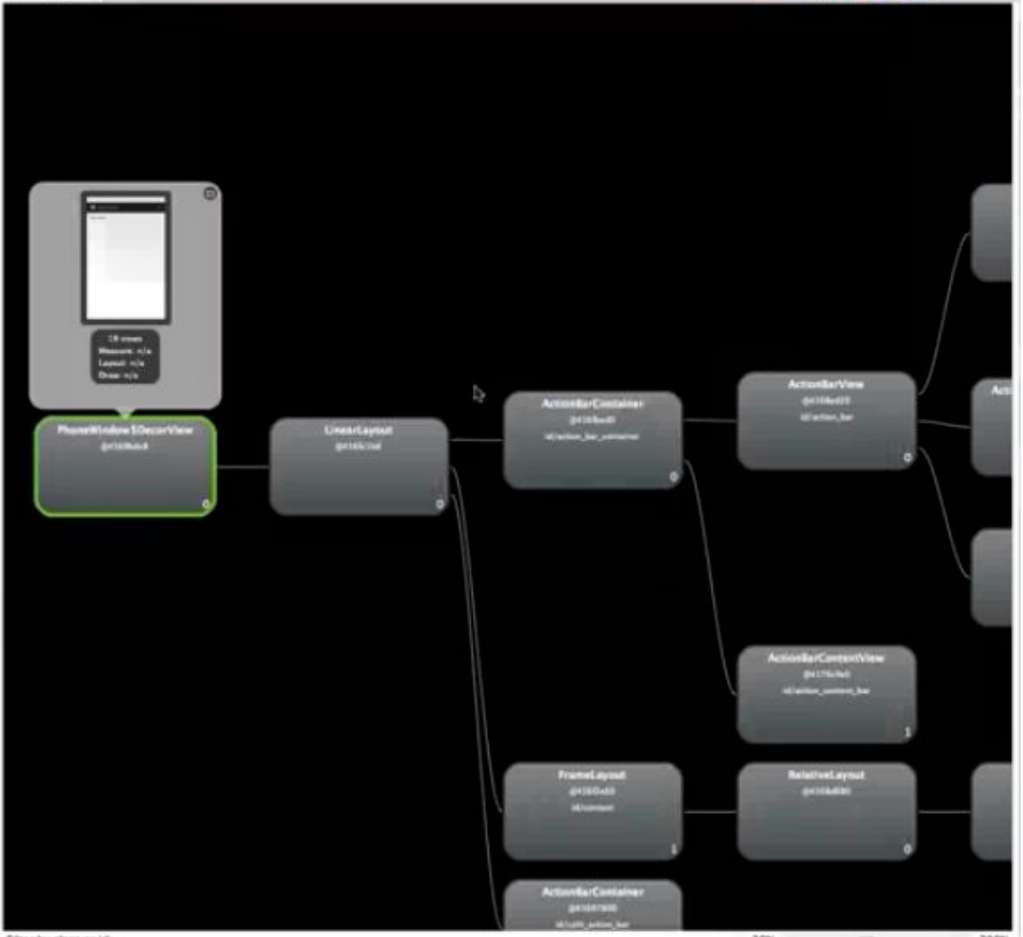
Например, давайте посмотрим еще раз на Hello Android приложение.

Во-первых, давайте запустим HelloAndroid приложение.

В Android Studio откроем меню Window и выберем Open Perspective. Затем выберем Hierarchy View.

Теперь, как только Hierarchy View запустится, в панеле Window дважды щелкнем узел, соответствующий HelloAndroid приложению.

Двойной щелчок на данном узле даст графическое представление GUI-интерфейса приложения, который, как и большинство пользовательских интерфейсов организован в иерархию.



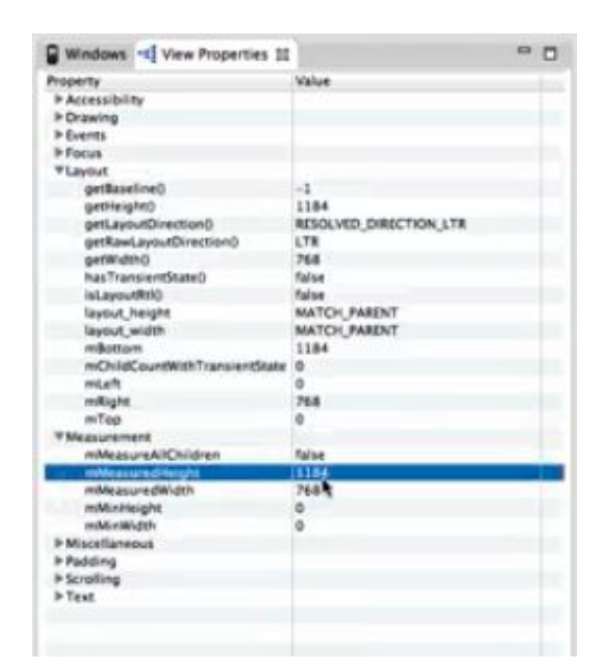
Например, есть окно, которое содержит весь GUI-интерфейс.

Он содержит два дочерних компонента, строку заголовка и большую прямоугольную область приложения.

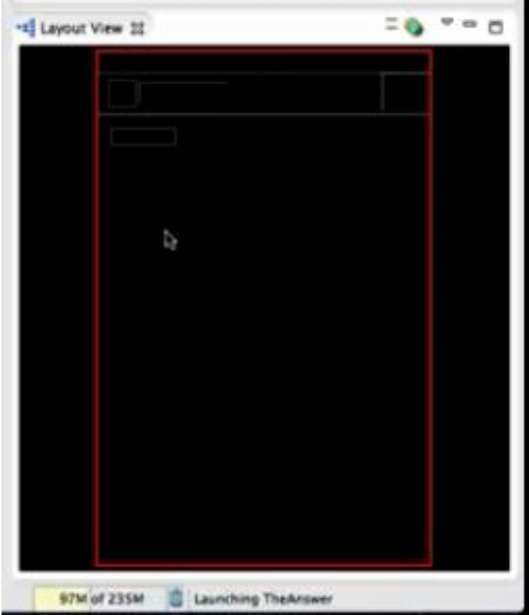
Каждый из этих компонентов, в свою очередь, содержит дополнительные элементы.

И если нажать на этом первом окне, можно увидеть подробную информацию о нем.

Например, его параметры компоновки, которые показывают, например, размеры окна в пикселях.



Область, занимаемая каждым элементом, показана в макете, в правом нижнем углу.



И та же информация также доступна для заголовка и других компонентов.

Вопросы к лекциям 1-2

Платформа Android содержит несколько компонентов. Какой из компонентов не является частью платформы Android?

1. Документация.
2. SDK.
3. Анализатор сетевого протокола.
4. Стек программного обеспечения.

Ответ: 3.

Какие функции Linux ядра платформы Android можно найти и в других Linux ядрах?

* 1. Архитектура разрешений.
  2. File & Network 10.
  3. Киллер нехватки памяти.
  4. Драйвера устройств.

Ответ: 3.

Какие функции Linux ядра платформы Android специфичны в отличие от других Linux ядрах?

* + 1. Драйвера устройств.
    2. Механизм Binder IPC.
    3. Управление питанием.
    4. File 10.

Ответ: 2 и 3.

Android приложения могут выполняться стандартной Java виртуальной машиной?

* + - 1. Да.
      2. Нет.

Ответ: 2.

Пакеты java.\* платформы Android аналогичны, но не полностью, таким же пакетам платформы Java SE. Какие из следующих классов есть в платформе Android?

* + - * 1. java.rmi.server.RemoteObject
        2. j ava.util. concurrent.RunnableFuture

Ответ: 2.

Одна вещь, которую диспетчер пакетов делает, чтобы позволить приложению использовать другое приложение. Например, несколько Android приложений могут использовать приложение Галерея, чтобы пользователи могли просматривать и выбирать фотографии. Можете ли вы найти хотя бы один пример, когда одно приложение использует другое приложение?

Менеджер активности управляет жизненным циклом приложения. С какими из следующих функций менеджер активности может быть связан?

Запуск и остановка активности.

Управление навигацией между активностями.

Управление панелью уведомлений.

Ответ: 1 и 2.

Какое из следующих утверждений подчеркивает преимущество использования Content Provider для управления данными приложений?

Обработка данных будет быстрее с Content Provider.

Данные, введенные в одном приложении, могут использоваться другим приложением.

Платформа Android не обеспечивает другого способа управления данными. Ответ: 2.

Область уведомлений или строка состояния является областью, которая контролируется Andraid-системой и к которой приложения имеют доступ через NotificationManager. Какие из следующих причин наиболее вероятны для выбора именно такого способа использования области уведомления?

Для лучшей производительности приложений.

Обеспечивается способ информирования пользователя о событиях вне UI- интерфейса активности.3. Обеспечивается область, которая контролируется Android-системой и которая почти всегда видна пользователю.

Ответ: 2 и 3.

Даже если вы напишете лучший плеер, чем стандартным плеер Android, это не будет иметь значения, потому что вам будет нужно разрешение от Android, чтобы заменить их плеер вашим.

Да.

Нет.

Ответ: 2.

Вы должны убедиться, что ваша среда разработки имеет последнюю версию Java, потому что иначе вы не сможете иметь доступ к последним улучшениям Java.

Да.

Нет.

Ответ 2.

Какие из следующих функций, как правило, ограничены в портативных устройствах относительно традиционных настольных устройств? Выберите все подходящие варианты:

1. Мощность.
2. Размер экрана.
3. Память.
4. Скорость CPU

Ответ 1,2,3,4.

Как называется метод в классе MainActivity, который устанавливает и отображает экран пользовательского интерфейса приложения?

* 1. main()
  2. onStart()
  3. onCreate()
  4. run()

Ответ 3.

Перейти на сайт Android разработчика и посмотреть на следующей веб-странице: AVDS и диспетчер AVD (<http://developer.android.com/tools/devices/managing-> avds.html). Перейдите в раздел Hardware Options. Там вы увидите список аппаратных особенностей, которые AVD поддерживает. Отключена ли по умолчанию поддержка камеры?

* + 1. Да.
    2. Нет.

Ответ 1.

Какие из следующих преимущества использования эмулятора по сравнению с использованием фактического устройства? Выберите все подходящие варианты.

* + - 1. Низкая стоимость.
      2. Предоставляет разработчикам опции, которых нет на реальном устройстве.
      3. Можно настраивать оборудование.
      4. Быстрее чем другие устройства.

Ответ 1,3.

Какое из следующих утверждений описывают недостатки использования эмулятора по сравнению с использованием реального устройства? Выберите все подходящие варианты.

* + - * 1. Меньший размер экрана.
        2. Эмулятор не работает на Linux.
        3. Не полностью может заменить реальное устройство.
        4. Эмулятор медленнее.

Ответ 3,4.

Logcat предоставляет окно консоли, которое отображает сообщения журнала, инициированные программами, запущенными на вашем устройстве.

Да.

Нет.Ответ 1

Traceview предоставляет статистику покрытия строк (то есть, какие строки выполняются, а какие нет) для работающего приложения Android.

Да.

Нет.

Ответ 2.

Android пользовательские интерфейсы организованы в виде дерева элементов пользовательского интерфейса.

Да.

Нет.

Ответ 1.