## Установка Android Studio. “Hello, World”

### Цель работы

Получить навыки установки и первоначальной настройки среды разработки, получить базовое представление о структуре приложения и порядке его запуска.

### Задания для выполнения

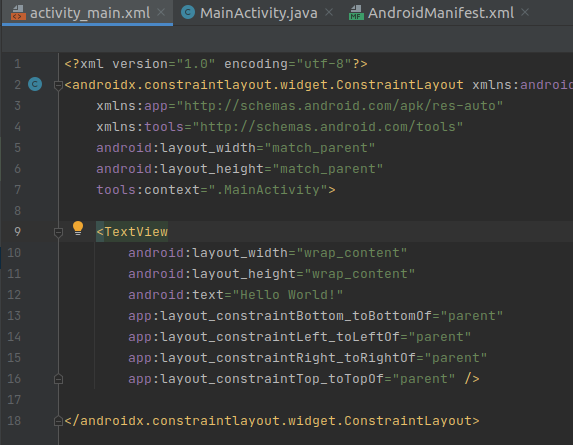
1. Установите на компьютер среды разработки JDK и Android Studio.
2. Запустите Android Studio.
3. Создайте новый пустой проект на языке программирования Java.
4. Познакомьтесь со структурой проекта и с содержанием главных файлов:
   1. MainActivity.java
   2. activity\_main.xml
   3. AndroidManifest.xml
   4. strings.xml

Модуль имеет три подпапки:

* **manifests**: хранит файл манифеста **AndroidManifest.xml**, который описывает конфигурацию приложения и определяет каждый из компонентов данного приложения.
* **java**: хранит файлы кода на языке java, которые структурированы по отдельным пакетам. Так, в папке com.example.helloapp (название которого было указано на этапе создания проекта) имеется по умолчанию файл **MainActivity.java** с кодом на языке Java, который представляет класс MainActivity, запускаемый по умолчанию при старте приложения
* **res**: содержит используемые в приложении ресурсы. Все ресурсы разбиты на подпапки.
  + папка **drawable** предназначена для хранения изображений, используемых в приложении
  + папка **layout** предназначена для хранения файлов, определяющих графический интерфейс. По умолчанию здесь есть файл **activity\_main.xml**, который определяет интерфейс для класса MainActivity в виде xml
  + папки **mipmap** содержат файлы изображений, которые предназначены для создания иконки приложения при различных разрешениях экрана.
  + папка **values** хранит различные xml-файлы, содержащие коллекции ресурсов - различных данных, которые применяются в приложении. По умолчанию здесь есть два файла и одна папка:
    - файл **colors.xml** хранит описание цветов, используемых в приложении
    - файл **strings.xml** содержит строковые ресурсы, используемые в приложении
    - папки **themes** хранит две темы приложения - для светлую (дневную) и темную (ночную)

1. Найдите в справочном материале и кратко опишите своими словами назначение и содержание каждого из этих файлов.

* MainActivity.java - Выполнение приложения Android по умолчанию начинается с класса MainActivity, который по умолчанию открыт в Android Studio. Класс MainActivity по сути представляет обычный класс java, в начале которого идет определение пакета данного класса. Далее идет импорт классов из других пакетов, функциональность которых используется в MainActivity. Затем идет собственно определение класса. По умолчанию MainActivity наследуется от класса AppCompatActivity, который выше подключен с помощью директивы импорта. Класс AppCompatActivity по сути представляет отдельный экран (страницу) приложения или его визуальный интерфейс. И MainActivity наследует весь этот функционал.   
  По умолчанию MainActivity содержит только один метод onCreate(), в котором фактически и создается весь интерфейс приложения. В метод setContentView() передается ресурс разметки графического интерфейса. Именно здесь и решается, какой именно визуальный интерфейс будет иметь MainActivity. Но что в данном случае представляет ресурс R.layout.activity\_main? Это файл activity\_main.xml из папки res/layout (в принципе можно заметить, что название ресурса соответствует названию файла), который также по умолчанию открыт в Android Studio
* activity\_main.xml



activity\_main.xml - это обычный текстовый файл с разметкой xml. Для переключения к коду нажмем на кнопку Code над графическим представлением. (Дополнительно с помощью кнопки Split можно переключиться на комбинированное представление код + графический дизайнер).

ConstraintLayout позволяет расположить вложенные элементы в определенных местах экрана. Вначале элемента ConstraintLayout идет определение пространств имен XML.

Каждое пространство имен задается следующим образом: xmlns:префикс="название\_ресурса".

Название ресурса (или URI - Uniform Resource Indicator) - "http://schemas.android.com/apk/res/android". И этот ресурс сопоставляется с префиксом android (xmlns:android).

Зачем эти пространства имен нужны? Каждый ресурс или URI определяет некоторую функциональность, которая используется в приложении, например, предоставляют теги и атрибуты, которые необходимые для построения приложения.

* xmlns:android="http://schemas.android.com/apk/res/android": содержит основные атрибуты, которые предоставляются платформой Android, применяются в элементах управления и определяют их визуальные свойства (например, размер, позиционирование)
* xmlns:app="http://schemas.android.com/apk/res-auto": содержит атрибуты, которые определены в рамках приложения
* xmlns:tools="http://schemas.android.com/tools": применяется для работы с режиме дизайнера в Android Studio

И чтобы упростить работу с этими ресурсами, применяются префиксы.

**android:layout\_width** определяет ширину контейнера. Этот атрибут (layout\_width) расположен в ресурсе "http://schemas.android.com/apk/res/android". И поскольку этот ресурс сопоставляется с префиксом android, то для обращения к атрибуту перед ним через двоеточие указывается префикс данного ресурса.

Значением атрибута android:layout\_weight является **"match\_parent"**. Это значит, что элемент (ConstraintLayout) будет растягиваться по всей ширине контейнера (экрана устройства).

Атрибут **android:layout\_height="match\_parent"** определяет высоту контейнера и также определен в "http://schemas.android.com/apk/res/android". Значение "match\_parent" указывает, что ConstraintLayout будет растягивается по всей длине контейнера (экрана устройства).

Атрибут **tools:context** определяет, какой класс activity (экрана приложения) связан с текущим определением интерфейса. В данном случае это класс MainActivity. Это позволяет использовать в Android Studio различные возможности в режиме дизайнера, которые зависят от класса activity.

Таким образом, при запуске приложения сначала запускается класс MainActivity, который в качестве графического интерфейса устанавливает разметку из файла activity\_main.xml. И поскольку в этой разметке прописан элемент TextView, который представляет некоторый текст, то мы и увидим его текст на экране смартфона.

* AndroidManifest.xml

Каждое приложение содержит файл манифеста AndroidManifest.xml. Данный файл определяет важную информацию о приложении - название, версию, иконки, какие разрешения приложение использует, регистрирует все используемые классы activity, сервисы и т.д. Данный файл можно найти в проекте в папке manifests:

* strings.xml

Ресурсы строк - один из важных компонентов приложения. Мы используем их при выведении названия приложения, различного текста, например, текста кнопок и т.д.

XML-файлы, представляющие собой ресурсы строк, находятся в проекте в папке **res/values**. По умолчанию ресурсы строк находятся в файле **strings.xml**

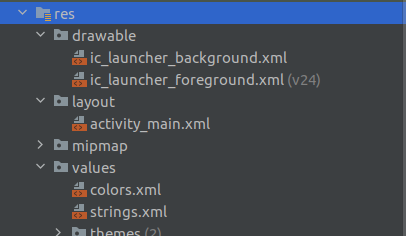
В самом простом виде этот файл определяет один ресурс "app\_name", который устанавливает название приложения, которое мы видим в заголовке приложения на экране устройства. Но естественно мы можем определить любые строковые ресурсы. Каждый отдельный ресурс определяется с помощью элемента **string**, а его атрибут name содержит название ресурса.

Затем в приложении в файлах кода мы можем ссылаться на эти ресурсы.

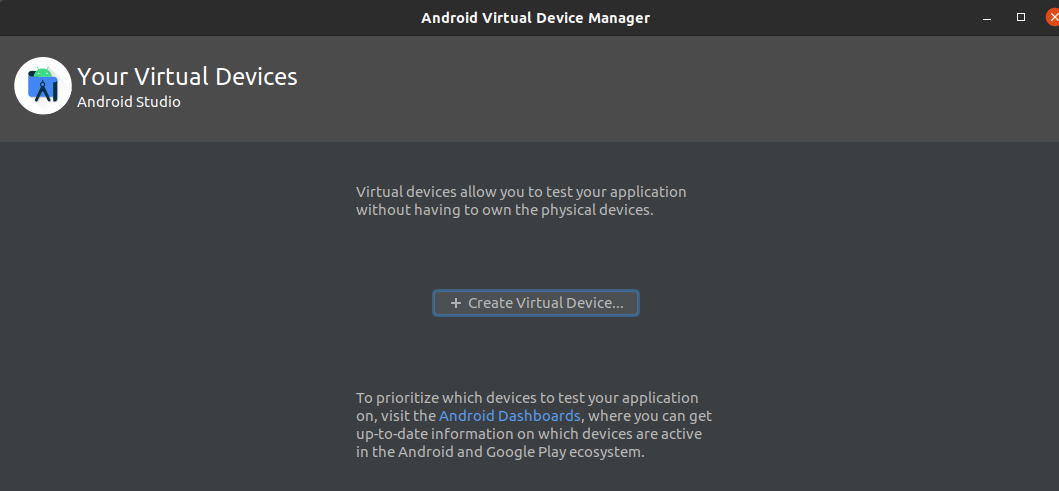
1. Познакомьтесь с присутствующими в проекте ресурсами.

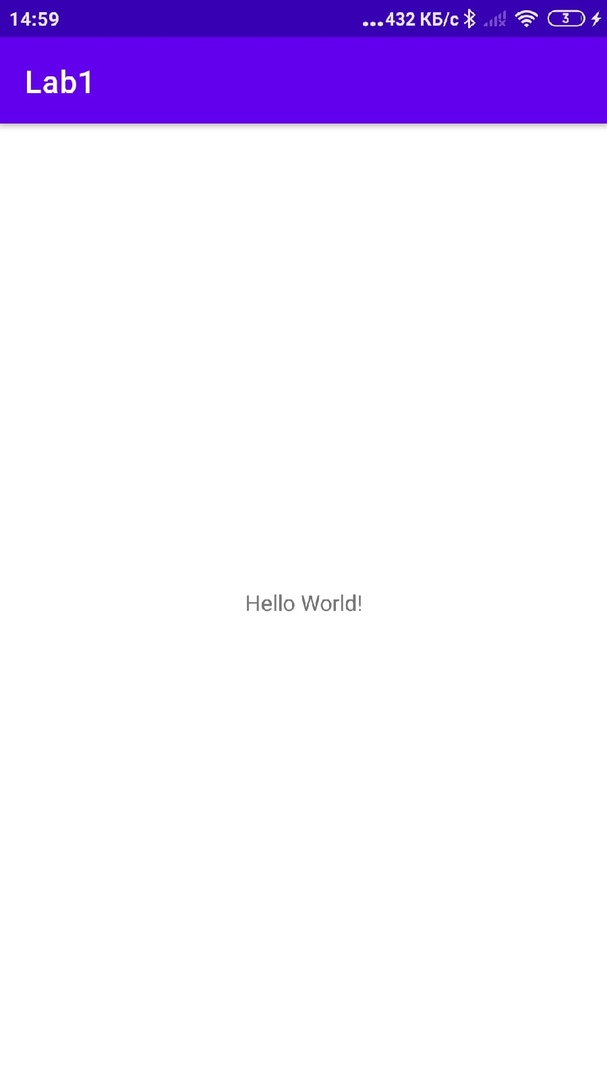
Ресурс в приложении Android представляет собой файл, например, файл разметки интерфейса или некоторое значение, например, простую строку. То есть ресурсы представляют собой и файлы разметки, и отдельные строки, и звуковые файлы, файлы изображений и т.д. Все ресурсы находятся в проекте в каталоге **res**. Для различных типов ресурсов, определенных в проекте, в каталоге **res** создаются подкаталоги. Поддерживаемые подкаталоги:

* **animator/**: xml-файлы, определяющие анимацию свойств
* **anim/**: xml-файлы, определяющие tween-анимацию
* **color/**: xml-файлы, определяющие список цветов
* **drawable/**: Графические файлы (*.png*, *.jpg*, *.gif*)
* **mipmap/**: Графические файлы, используемые для иконок приложения под различные разрешения экранов
* **layout/**: xml-файлы, определяющие пользовательский интерфейс приложения
* **menu/**: xml-файлы, определяющие меню приложения
* **raw/**: различные файлы, которые сохраняются в исходном виде
* **values/**: xml-файлы, которые содержат различные используемые в приложении значения, например, ресурсы строк
* **xml/**: Произвольные xml-файлы
* **font/**: файлы с определениями шрифтом и расширениями .ttf, .otf или .ttc, либо файлы XML, который содержат элемент <font-family>

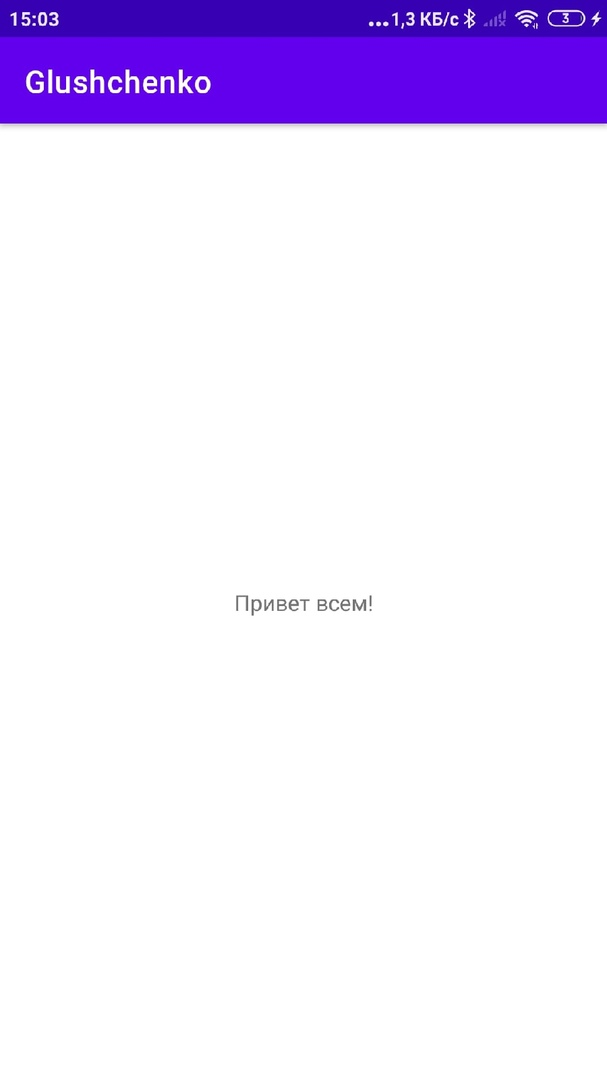


1. Запустите этот проект на виртуальном устройстве Android. Если виртуальных устройств не существует, создайте.





1. Модифицируйте проект таким образом, чтобы в названии приложения выводилась Ваша фамилия, а в строке на главном экране - любая другая строка по Вашему желанию.



1. Запустите данное приложение на нескольких виртуальных устройствах с разными версиями API и разными размерами и разрешениями экрана.

### Методические указания

Скачать JDK можно с сайта Oracle (http://www.oracle.com)

Чтобы скачать JDK нужно сначала принять условия лицензионного соглашения, а затем выбрать нужную версию.

После скачивания запустите setup-файл и установите JDK.

Скачать среду можно с сайта для разработчиков Android (http://developer.android.com/sdk/index.html).

### Контрольные вопросы

1. Что такое Activity и зачем оно нужно?

Каждый отдельный экран или страница в приложении описывается таким понятием как activity. Если мы запустим приложение на устройстве, то на экране мы по сути увидим определенную activity, которая предсталяет данный интерфейс.

Activity - то единичная сфокусированная вещь, которую может делать пользователь. Почти все действия взаимодействуют с пользователем, поэтому класс Activity позаботится о создании для вас окна, в котором вы можете разместить свой пользовательский интерфейс [setContentView(View)](https://developer.android.com/reference/android/app/Activity" \l "setContentView(android.view.View)). Хотя действия часто представляются пользователю как полноэкранные окна, их также можно использовать и другими способами: как плавающие окна (через тему с [R.attr.windowIsFloating](https://developer.android.com/reference/android/R.attr" \l "windowIsFloating)set), в многооконном режиме или встроенные в другие окна. Почти все подклассы Activity реализуют два метода:

* [onCreate(Bundle)](https://developer.android.com/reference/android/app/Activity" \l "onCreate(android.os.Bundle))здесь вы инициализируете свою деятельность. Что наиболее важно, здесь вы обычно вызываете setContentView(int) ресурс макета, определяющий ваш пользовательский интерфейс, и используете его findViewByld(int) для извлечения виджетов в этом пользовательском интерфейсе, с которыми вам нужно взаимодействовать программно.
* [onPause()](https://developer.android.com/reference/android/app/Activity" \l "onPause())Здесь вы имеете дело с пользователем, который приостанавливает активное взаимодействие с действием. Любые изменения, сделанные пользователем, на этом этапе должны быть зафиксированы (обычно для [ContentProvider](https://developer.android.com/reference/android/content/ContentProvider)хранения данных). В этом состоянии действие по-прежнему отображается на экране.

основная единица графического интерфейса (аналог окна или экранной формы)

* Видимая часть приложения (экран, окно, форма), отвечает за отображение графического интерфейса пользователя

Например:

* Приложение для работы   
  с будильником
* Приложение может иметь несколько активностей
* Активности приложения не зависят друг от друга

1. Что такое View и зачем оно нужно?

Богатый и расширяемый набор представлений

Для начала вам следует создать [собственный класс](http://developer.alexanderklimov.ru/android/customview.php) на основе **View/ViewGroup** или его предков. Затем, когда вы выберите элемент **<view>**, то в диалоговом окне можете увидеть созданный класс или выбрать что-то другое.

View – компонент, который пользователь может видеть и взаимодействовать с ним. В то время как ViewGroup является невидимым контейнером, который определяет структуру макета.

Объекты View обычно называют «виджетами». Они могут быть представлены одним из классов-наследников, таких как Button, TextView, ImageView. Объекты ViewGroup обычно называют «контейнерами». Они определяют, как именно будут располагаться элементы внутри экрана. Подробнее с ViewGroup мы познакомимся в следующих уроках.

1. Что такое Layout и зачем оно нужно?

компоновка элементов пользовательского интерфейса

Позволяет привязывать компонент к границам экрана или к другим компонентам. позволяет размещать View-компоненты в виде одной строки (horizontal) или одного столбца (vertical). настраивает положение каждого компонента относительно других.

Пользовательский интерфейс представлен layout файлом (рус. Макетом), который создаётся, используя язык разметки XML. Макет определяет структуру пользовательского интерфейса экрана. Все элементы макета построены с использованием иерархии объектов View и ViewGroup.

1. Почему все строки в Layout указываются косвенно из файла с ресурсами?

Потому что это ссылки на хранимые ресурсы. По этим ссылкам можно обращаться как в коде на странице activity\_main.xml, так и в графическом режиме (там они будут автоматически применяться).

1. Что такое версия API?

Это правда, что в Android между SDK и API существует отношение 1:1, и часто эти два термина используются как синонимы, но важно понимать, что это не одно и то же.  
  
Правильнее говорить, что для каждой версии Android есть SDK и эквивалентный API, а также уровень этого API.

## API

Расшифровывается как *Application Programming Interface* (программный интерфейс приложения). Это просто интерфейс, уровень абстракции, который обеспечивает связь между двумя разными «частями» программного обеспечения. Он работает как договор между поставщиком (например, библиотекой) и потребителем (например, приложением).  
  
Это набор формальных определений, таких как классы, методы, функции, модули, константы, которые могут использоваться другими разработчиками для написания своего кода. При этом API не включает в себя реализацию.

Уровень API — это целочисленное значение, однозначно идентифицирующее версию API фреймворка, предлагаемую платформой Android.  
  
Обычно обновления API фреймворка платформы разрабатываются таким образом, чтобы новая версия API оставалась совместимой с более ранними версиями, поэтому большинство изменений в новом API являются аддитивными, а старые части API становятся устаревшими, но не удаляются.

### Дополнительные задания

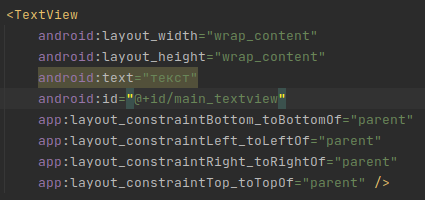
1. Замените RelativeLayout в главном окне приложения на LinearLayout с вертикальным расположением элементов.

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Привет всем!"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

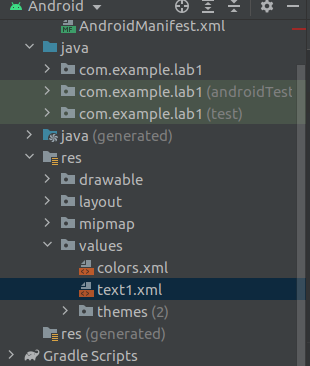


1. Присвойте строке, отображаемой в TextView идентификатор следующим атрибутом:

android:id=”@+id/main\_textview”



1. Измените название ресурса строки на “text1”



1. Модифицируйте строку динамически из кода Java, добавив в обработчик события onCreate следующий код:

// 1. Access the TextView defined in layout XML

// and then set its text

mainTextView = (TextView) findViewById(R.id.main\_textview);

mainTextView.setText(“Set in Java!”);

