

Рис. 1.

При разработке, каждому Activity сопоставляется одноименный java-класс (наследник класса android.app.Activity). При запуске приложения, когда система должна показать Activity и в дальнейшем работать с ним, она будет вызывать методы этого класса. И от того, что мы в этих методах накодим, зависит поведение Activity.

При создании модуля мы указывали, что надо создать Activity с именем MainActivity

Давайте посмотрим этот класс.

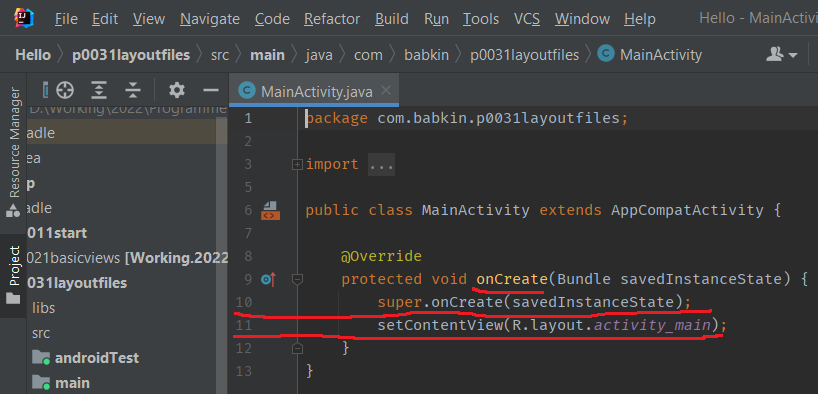


Рис. 2.

Смотрим java-код. Нас интересует метод onCreate – он вызывается, когда приложение создает и отображает Activity (на остальные методы пока не обращаем внимания). Посмотрим код реализации onCreate.

|  |  |
| --- | --- |
| 10 | super.onCreate(savedInstanceState); |

это вызов метода родительского класса, выполняющий необходимые процедуры, его мы не трогаем.

Нас сейчас очень интересует следующая строка:

|  |  |
| --- | --- |
| 11 | setContentView(R.layout.activity\_main); |

Метод [setContentView(int)](http://developer.android.com/reference/android/app/Activity.html#setContentView(int)) – устанавливает содержимое Activity из layout-файла. Но в качестве аргумента мы указываем не путь к layout-файлу (res/layout/activity\_main.xml), а константу, которая является ID файла. Эта константа генерируется автоматически в файле R.java, который мы пока трогать не будем. В этом классе будут храниться сгенерированные ID для всех ресурсов проекта (из папки res/\*), чтобы мы могли к ним обращаться. Имена этих ID-констант совпадают с именами файлов ресурсов (без расширений).

Файл res/layout/activity\_main.xml был создан средой разработки вместе с Activity. Его название запрашивалось на том же экране, где и название Activity.

Попробуем отобразить содержимое другого файла. Создадим еще один layout-файл, например myscreen.xml. Для этого выделим папку res/layout в нашем модуле и нажмем на ней правую кнопку мыши. В появившемся меню выбираем New > Layout resource file. Для любителей горячих клавиш есть более удобный путь: при выделенной папке res/layout нажать ALT+Insert, и там уже Enter на пункте Layout resource file.

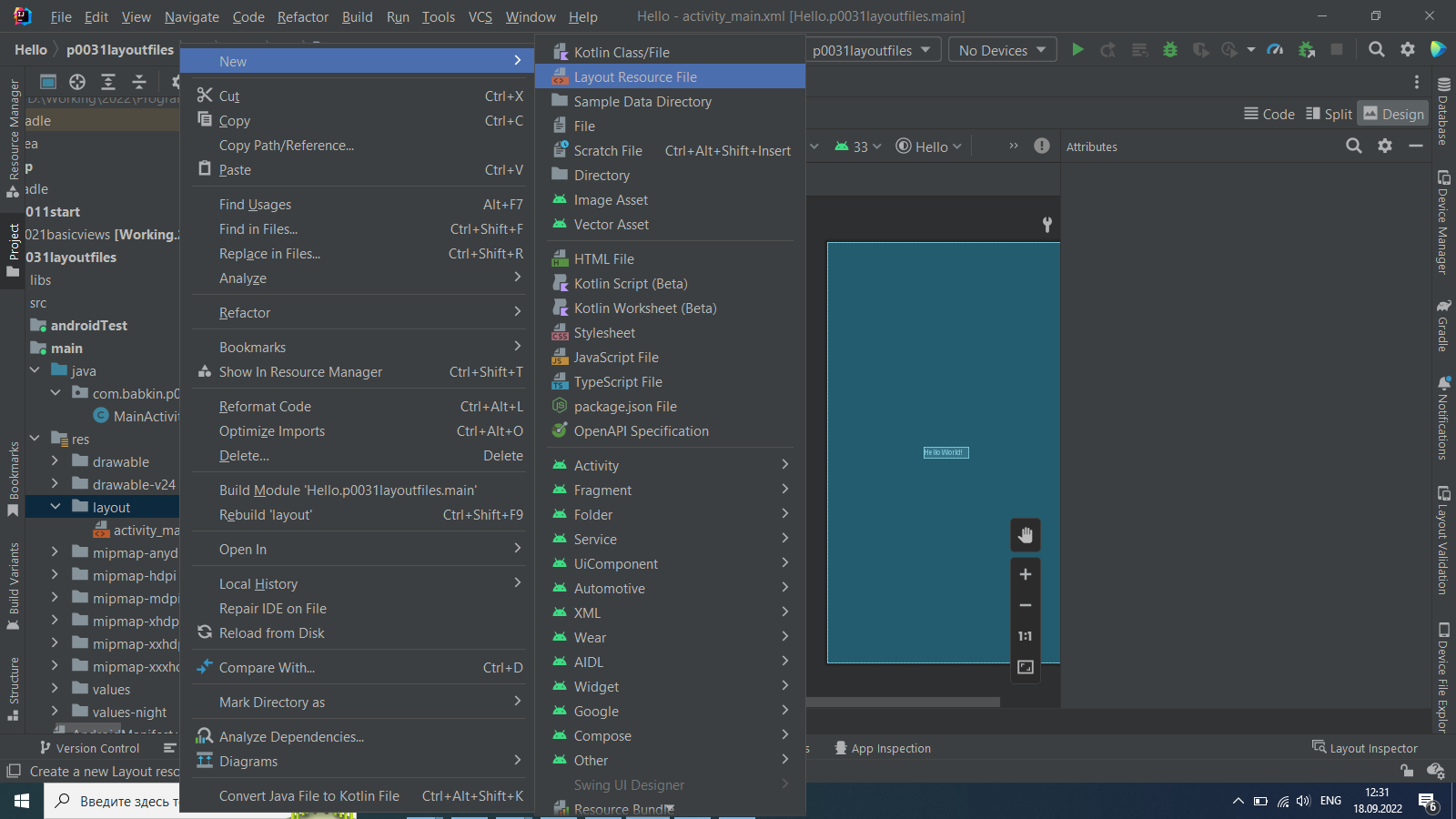


Рис. 3.

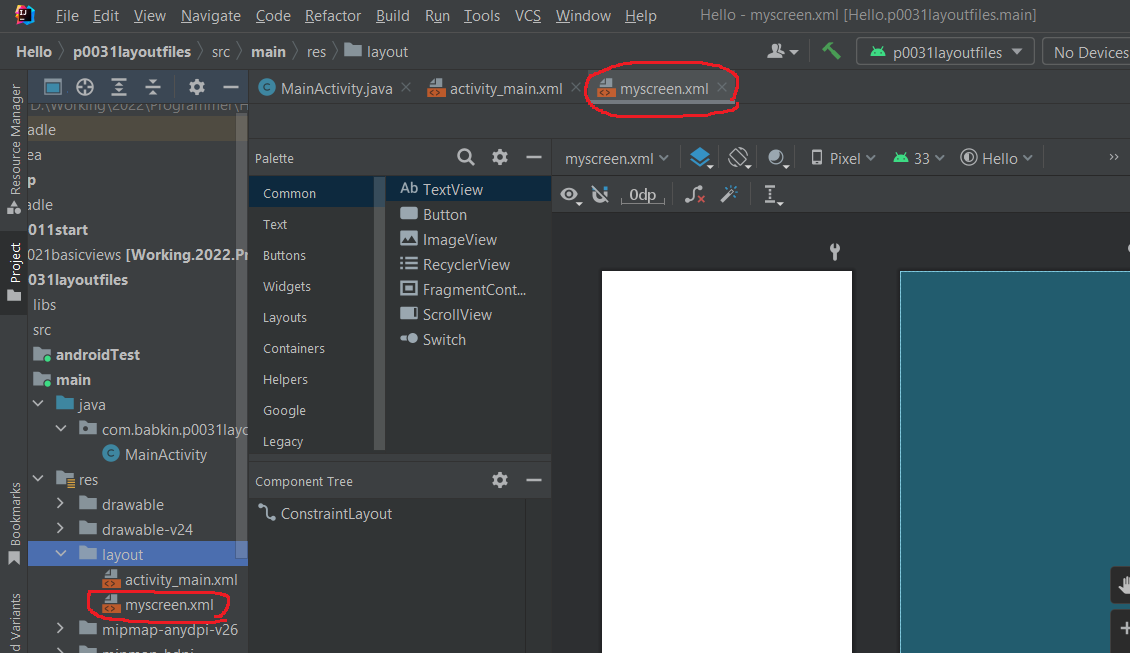


Рис. 4.

Добавим на экран элемент Plain TextView из списка слева и через Properties изменим его текст на: «new layout file myscreen for activity».

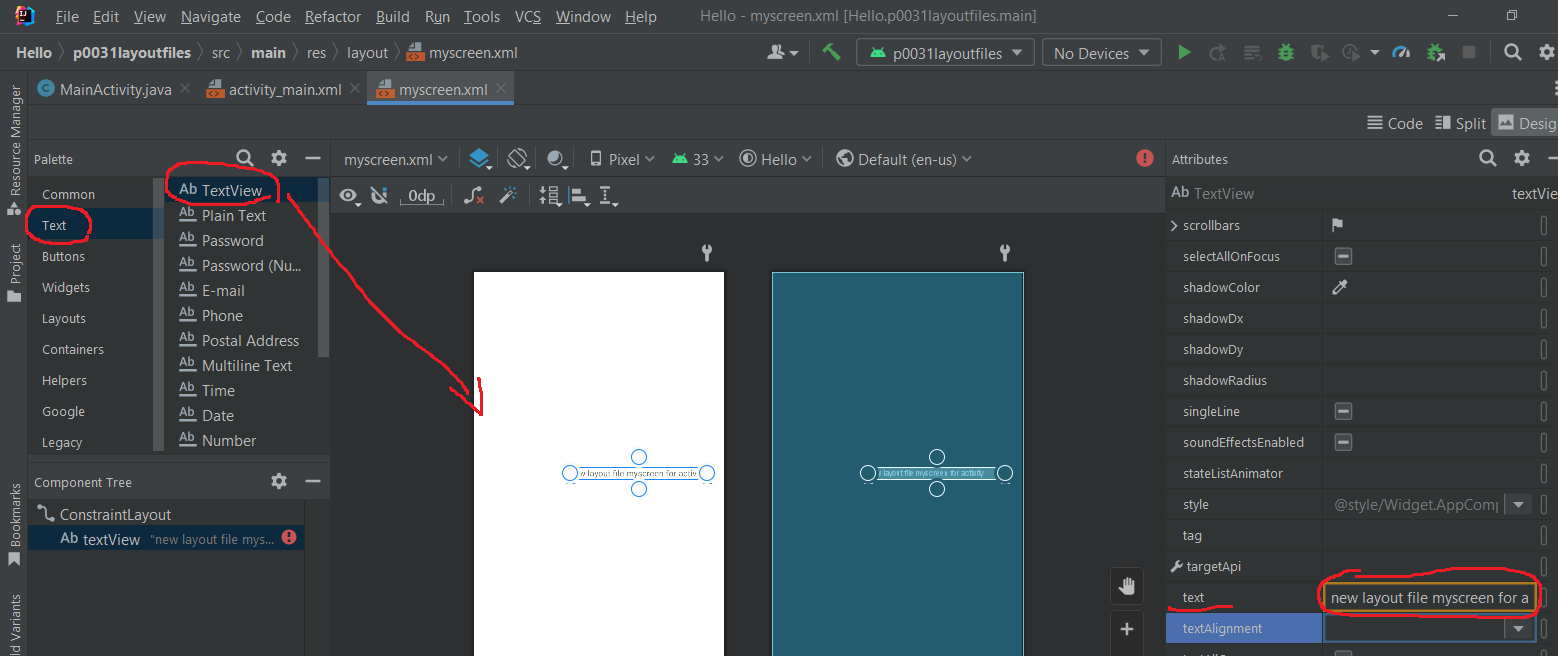


Рис. 5.

При создании нового layout-файла myscreen, среда добавила в R.java новую константу для этого файла - R.layout.myscreen. И мы теперь в коде сможем через эту константу указать на этот новый layout-файл.

Настроим так, чтобы Activity использовало новый файл myscreen.xml, а не activity\_main.xml, который был изначально. Откроем MainActivity.java и поменяем аргумент метода setContentView. Замените «R.layout.activity\_main», на «R.layout.myscreen» (ID нового layout-файла). Должно получиться так:

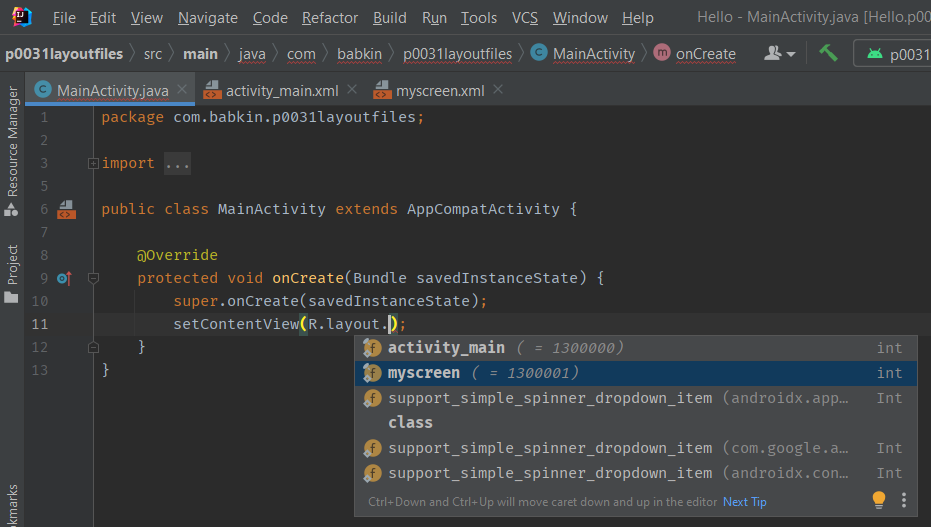


Рис. 6.



Рис. 7.

Видим, что теперь оно отображает содержимое из myscreen.xml, т.к. мы явно ему это указали в методе setContentView, который выполняется при создании (onCreate) Activity

Открыв в layout файл myscreen, вы видите его визуальное представление. Т.е. некий предпросмотр, как это будет выглядеть на экране. Откройте вкладку Code

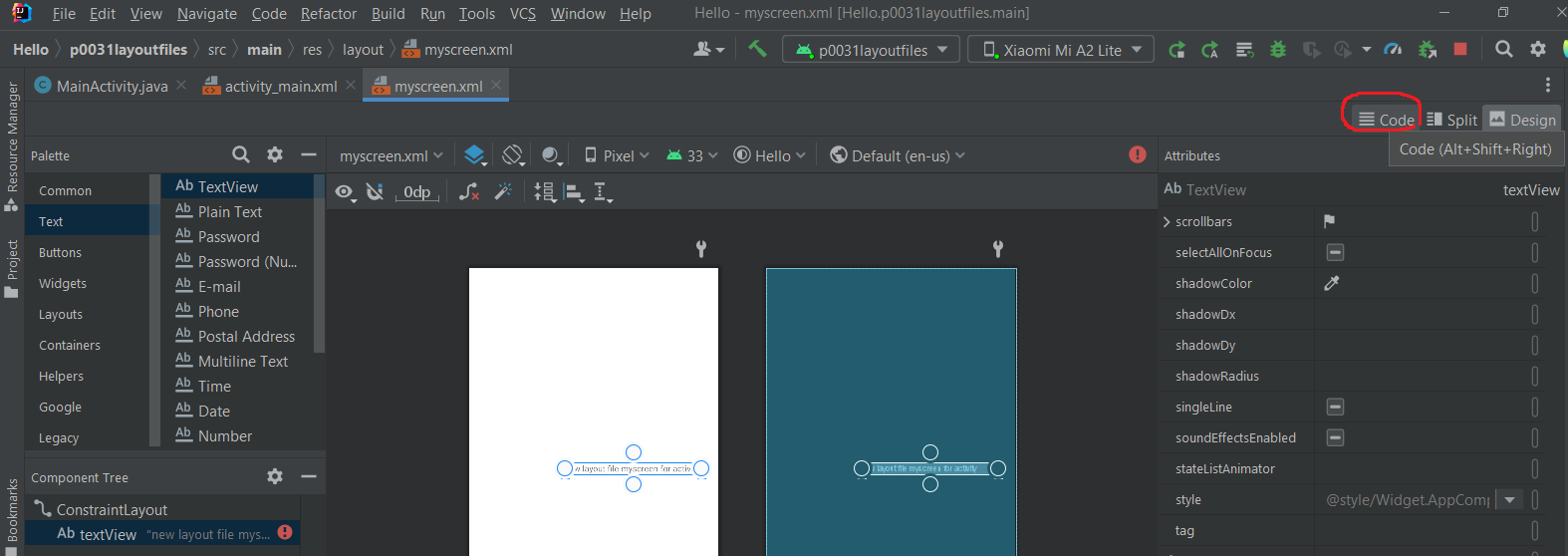


Рис. 8.

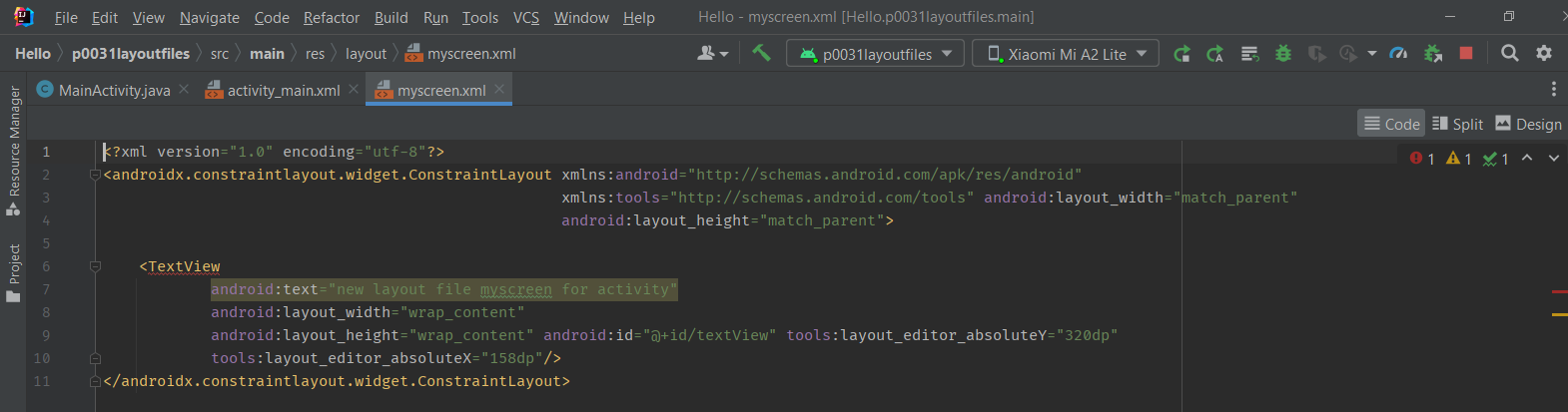


Рис. 9.

Мы видим достаточно читабельное xml-описание всех View нашего layout-файла. Названия xml-элементов - это классы View-элементов, xml-атрибуты - это параметры View-элементов, т.е. все те параметры, что мы меняем через вкладку Properties. Также вы можете вносить изменения прямо сюда и изменения будут отображаться во вкладке Design. Например, изменим текст у TextView. Вместо «new layout file myscreen for activity», напишем текст «Какой-либо текст»

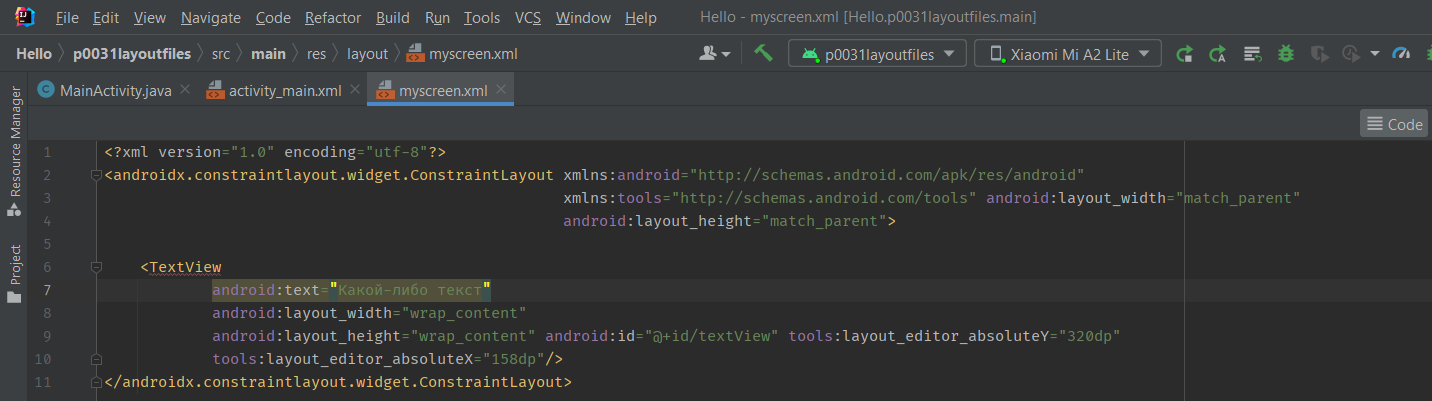


Рис. 10.

Сохраняем. Открываем Design и наблюдаем изменения.

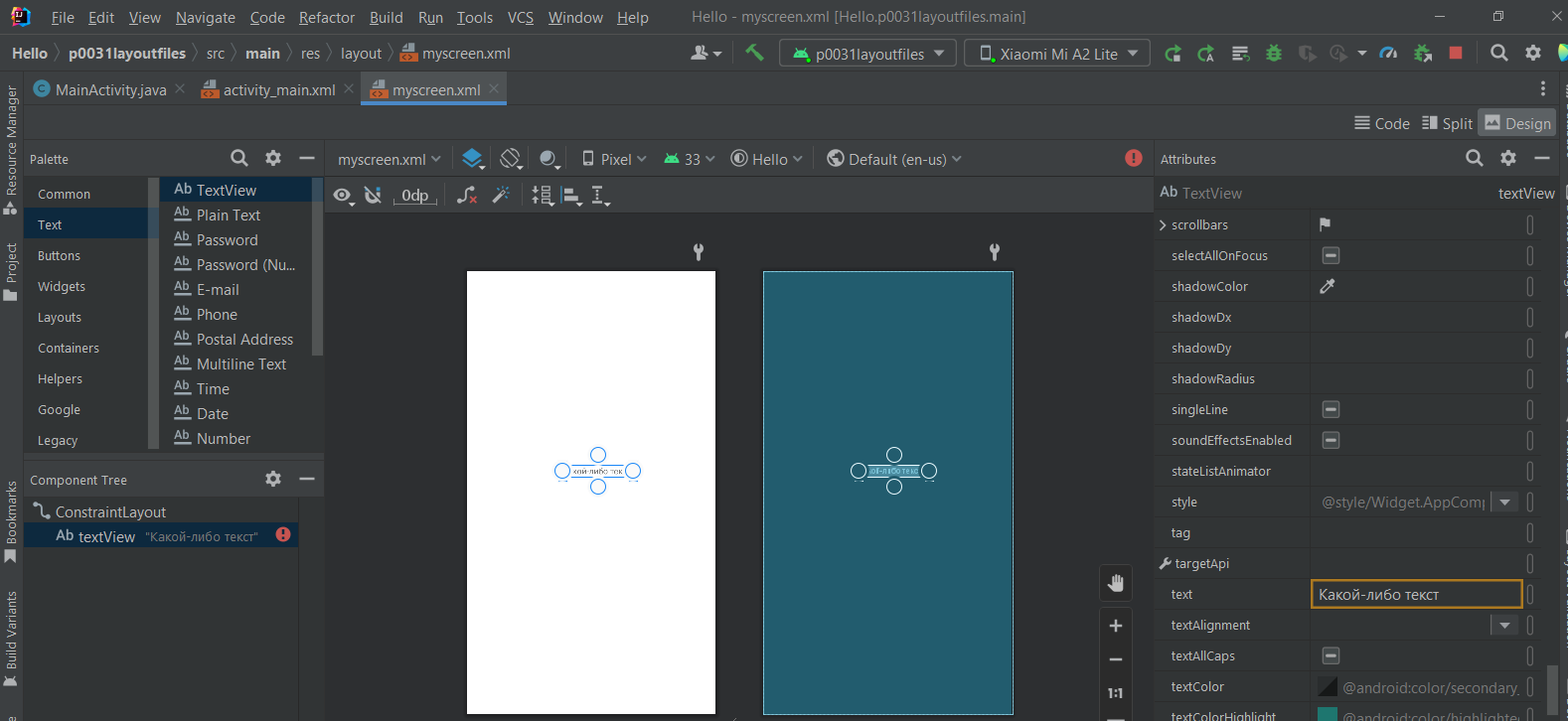


Рис. 11.

По умолчанию мы настраиваем layout-файл под вертикальную ориентацию экрана. Но что будет если мы повернем смартфон и включится горизонтальная ориентация? Давайте смотреть.

Изменим myscreen.xml. Добавим вертикальный ряд кнопок и изменим надпись.

xml-код (вы можете скопировать его и заменить им содержимое вашего layout файла myscreen во вкладке Text):

<LinearLayout

    xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    android:orientation="vertical">

    <TextView

        android:id="@+id/textView1"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:text="Вертикальная ориентация экрана">

    </TextView>

    <LinearLayout

        android:id="@+id/linearLayout1"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:orientation="vertical">

        <Button

            android:id="@+id/button1"

            android:layout\_width="100dp"

            android:layout\_height="100dp"

            android:text="Button1">

        </Button>

        <Button

            android:id="@+id/button2"

            android:layout\_width="100dp"

            android:layout\_height="100dp"

            android:text="Button2">

        </Button>

        <Button

            android:id="@+id/button3"

            android:layout\_width="100dp"

            android:layout\_height="100dp"

            android:text="Button3">

        </Button>

        <Button

            android:id="@+id/button4"

            android:layout\_width="100dp"

            android:layout\_height="100dp"

            android:text="Button4">

        </Button>

    </LinearLayout>

</LinearLayout>

Обратите внимание - я добавил вертикальный LinearLayout и поместил в него 4 кнопки. Подробнее обсудим это позднее.

Запустим приложение.

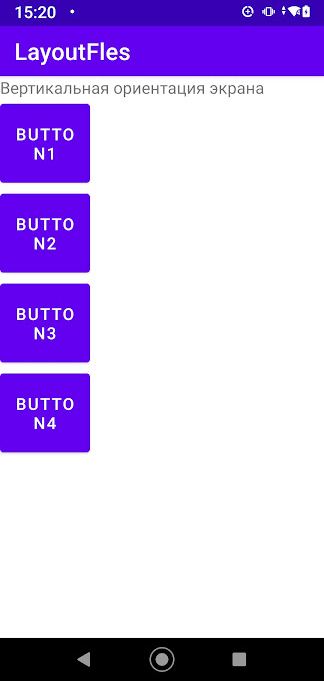


Рис. 12.

В вертикальной ориентации все ок. А в горизонтальной ориентации не все хорошо. Ориентация сменилась на горизонтальную и наши кнопки уже не влезают в экран.

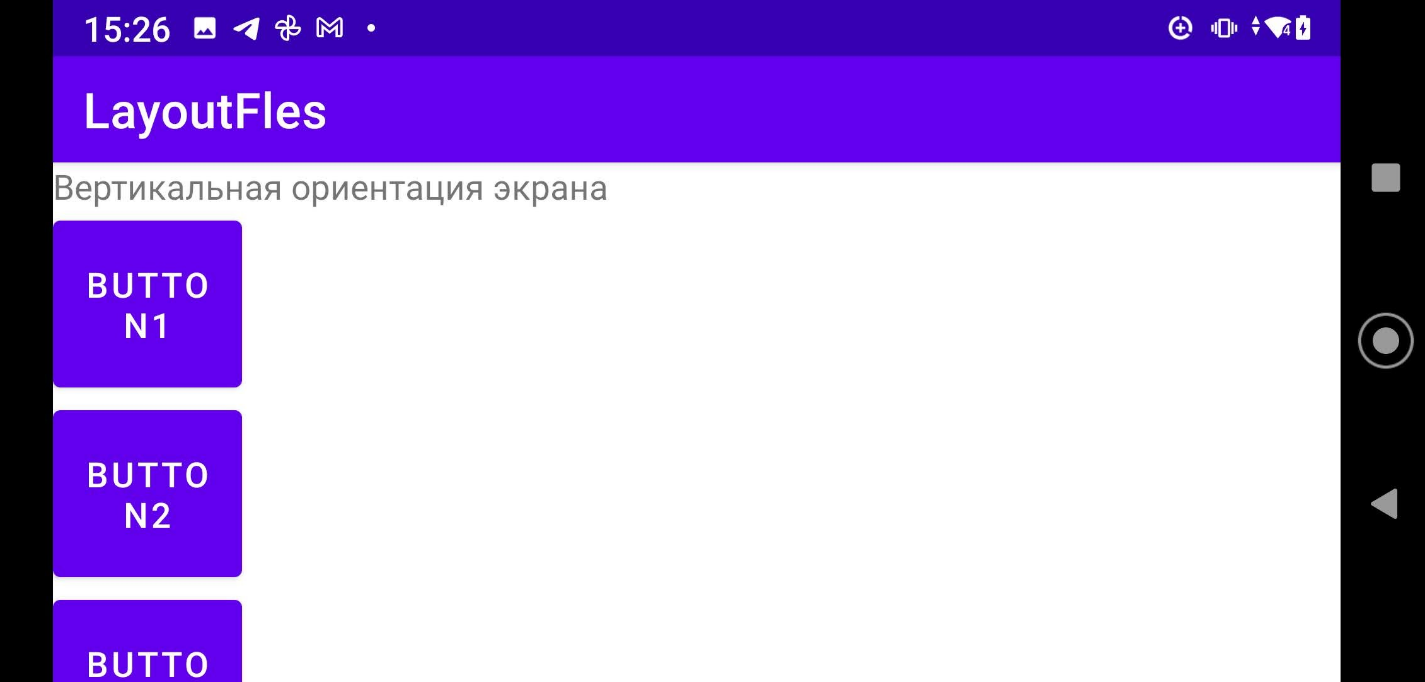


Рис. 13.

Т.е. нам необходим еще один layout-файл, который был бы заточен под горизонтальную ориентацию и в нашем случае вывел бы кнопки горизонтально.

Но как дать знать Activity, что она в вертикальной ориентации должна использовать один layout-файл, а в горизонтальной – другой? Об этом за нас уже подумали создатели Андроид. У нас есть возможность создать layout-файл, который будет использоваться приложением, когда устройство находится в горизонтальной ориентации. Создаем папку с именем layout-land, в которой создаем второй файл с именем myscreen

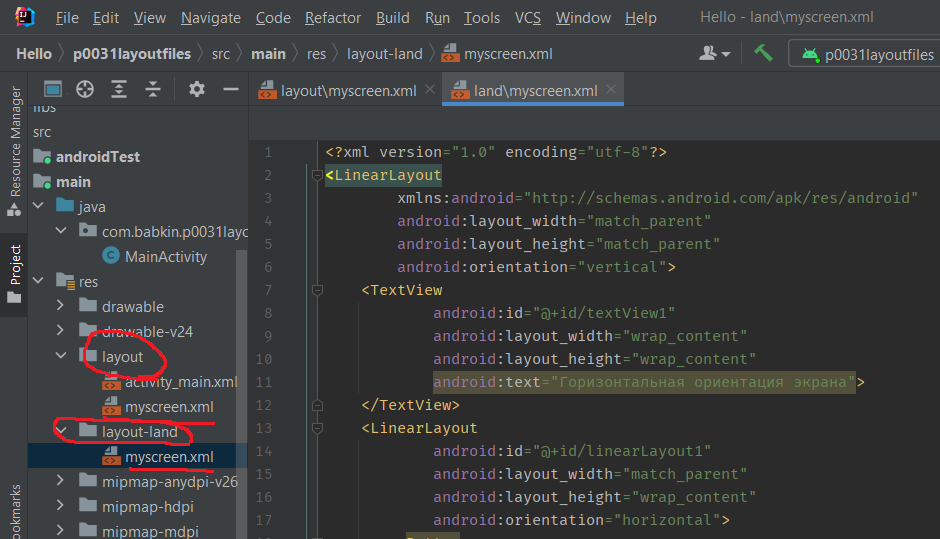


Рис. 14.

Во втором файле добавляем спецификатор, который даст приложению понять, что layout-файл надо юзать в горизонтальной ориентации. Нам надо указать, что нас интересует горизонтальная ориентация: Landscape. Выберите это значение из выпадающего списка.

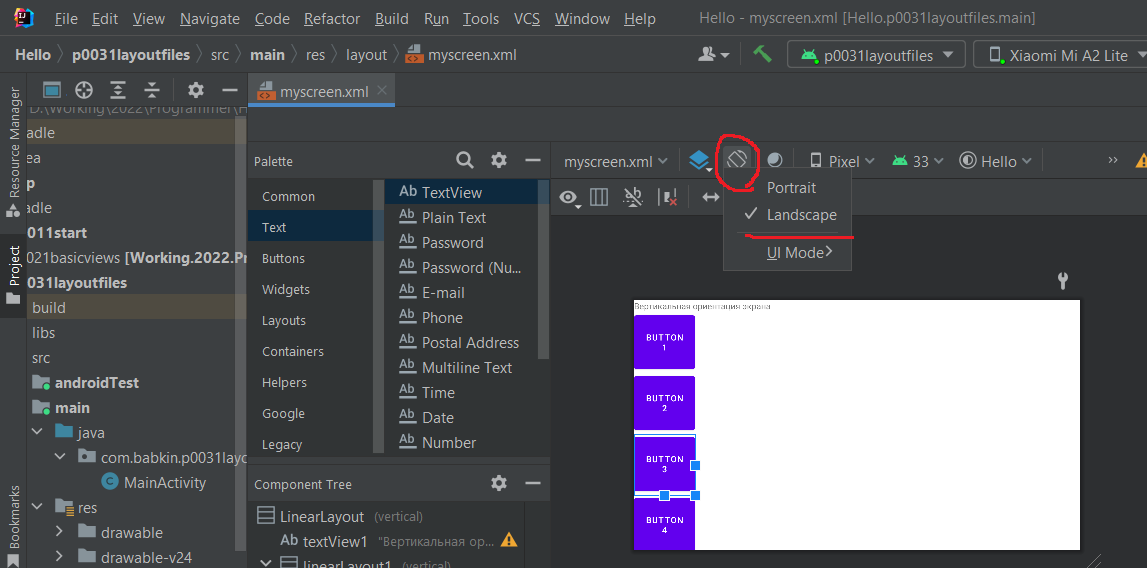


Рис. 15.

Теперь мы рассмотрим основные виды Layout:

LinearLayout – отображает View-элементы в виде одной строки (если он Horizontal) или одного столбца (если он Vertical). TableLayout – отображает элементы в виде таблицы, по строкам и столбцам.

RelativeLayout – для каждого элемента настраивается его положение относительно других элементов.

AbsoluteLayout – для каждого элемента указывается явная позиция на экране в системе координат (x,y)

Рассмотрим эти виды:

LinearLayout (LL)

Этот вид ViewGroup по умолчанию предлагается при создании новых layout-файлов. Он действительно удобен и достаточно гибок, чтобы создавать экраны различной сложности. LL имеет свойство Orientation, которое определяет, как будут расположены дочерние элементы – горизонтальной или вертикальной линией.

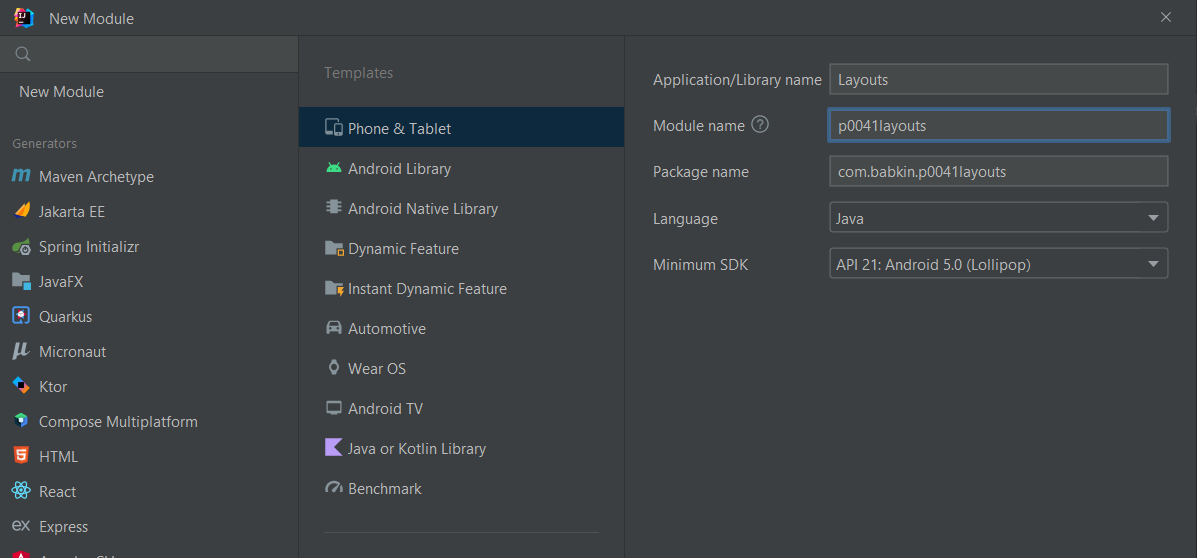


Рис. 16.

Откроем layout-файл **main.xml**, и поместите в него следующий код:

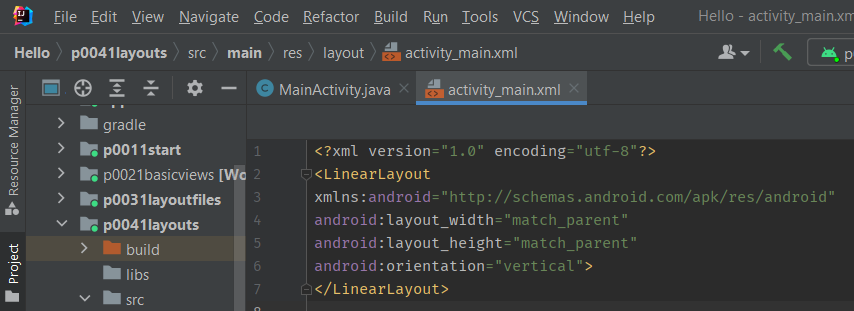


Рис. 17.

Теперь корневой элемент у нас LinearLayout с вертикальной ориентацией. Перетащите слева в корневой LinearLayout три кнопки. Они выстроились вертикально. Укажем для них ширину wrap\_content

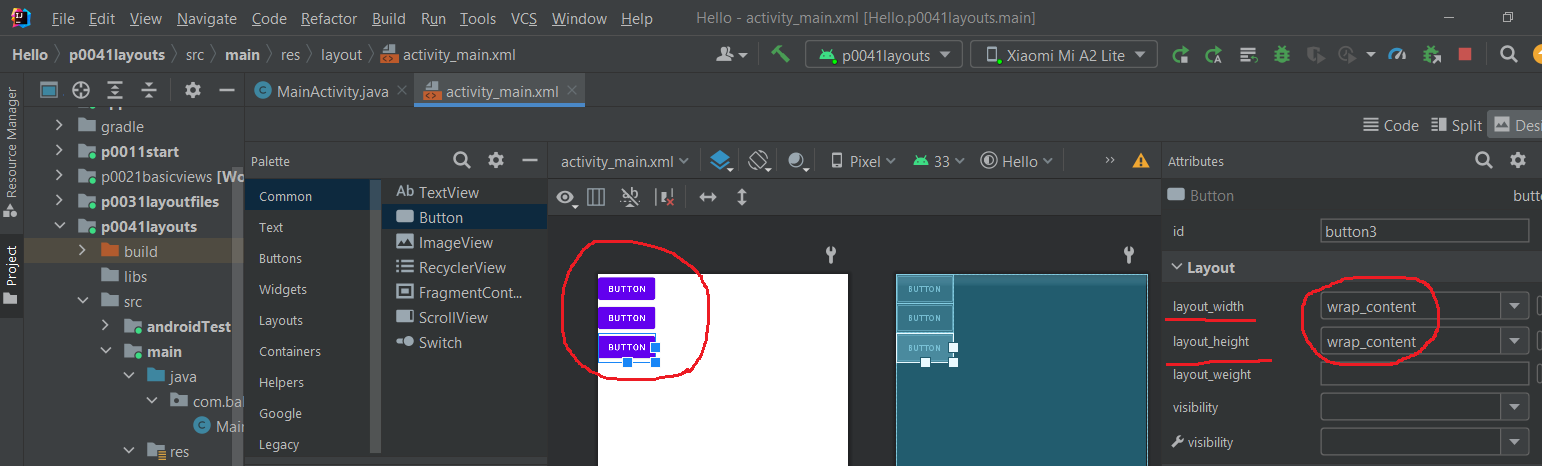


Рис. 18.

Теперь в Properties меняем для LL свойство **Orientation**на **horizontal** и сохраняем (CTRL+SHIFT+S) – кнопки выстроились горизонтально.

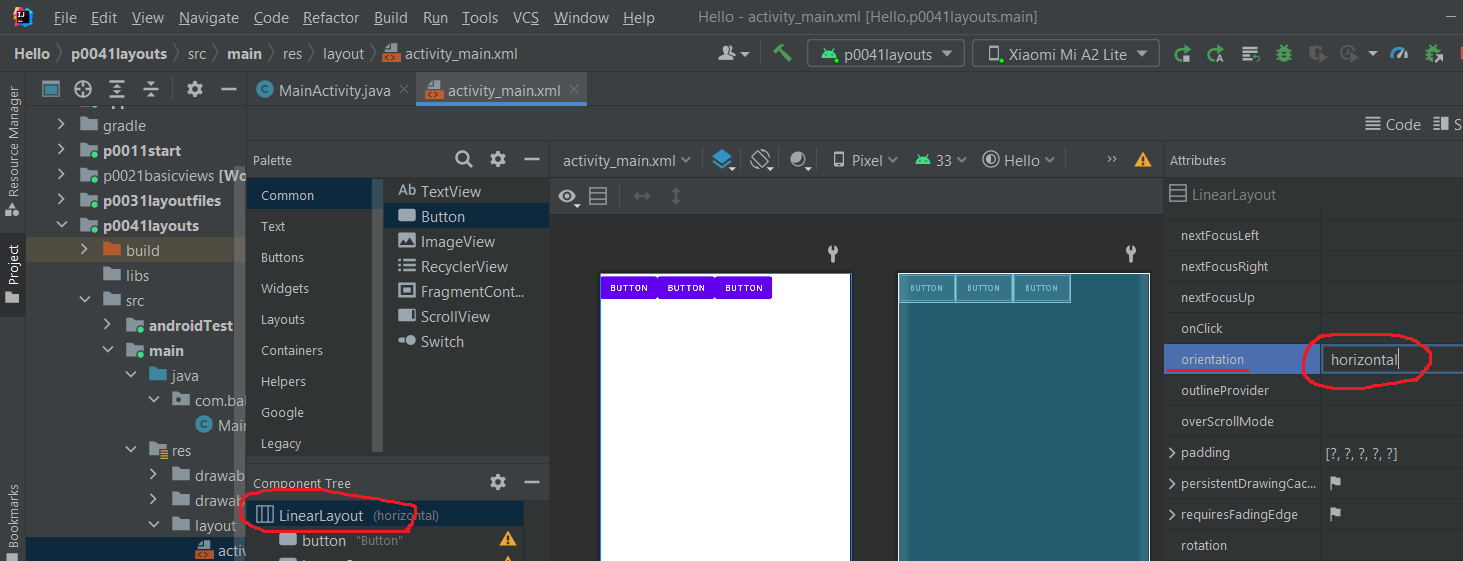


Рис. 19.

GroupView можно вкладывать друг в друга. Вложим в один LL два других. Удалите в main.xml все элементы (три кнопки) кроме корневого LL. Ориентацию корневого LL укажем вертикальную и добавим в него два новых горизонтальных LL. В списке элементов слева они находятся в разделе Layouts. Напоминаю, что вы можете перетаскивать элементы из списка не только на экран, но и на конкретный элемент на вкладке Outline.

В каждый горизонтальный LL добавим по три кнопки. Получилось два горизонтальных ряда кнопок. Убедитесь, что у горизонтальных LinearLayout высота (height) установлена в wrap\_content.

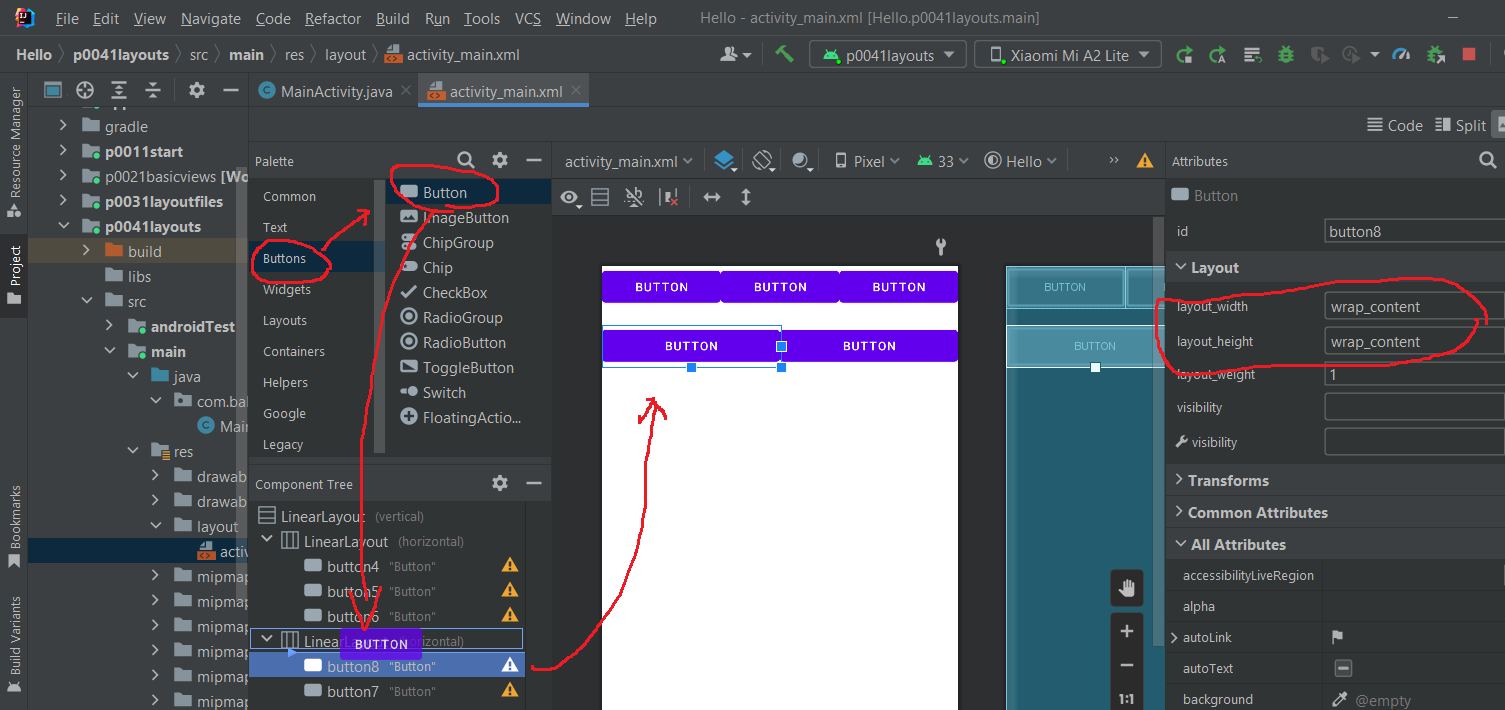


Рис. 20.



Рис. 21.

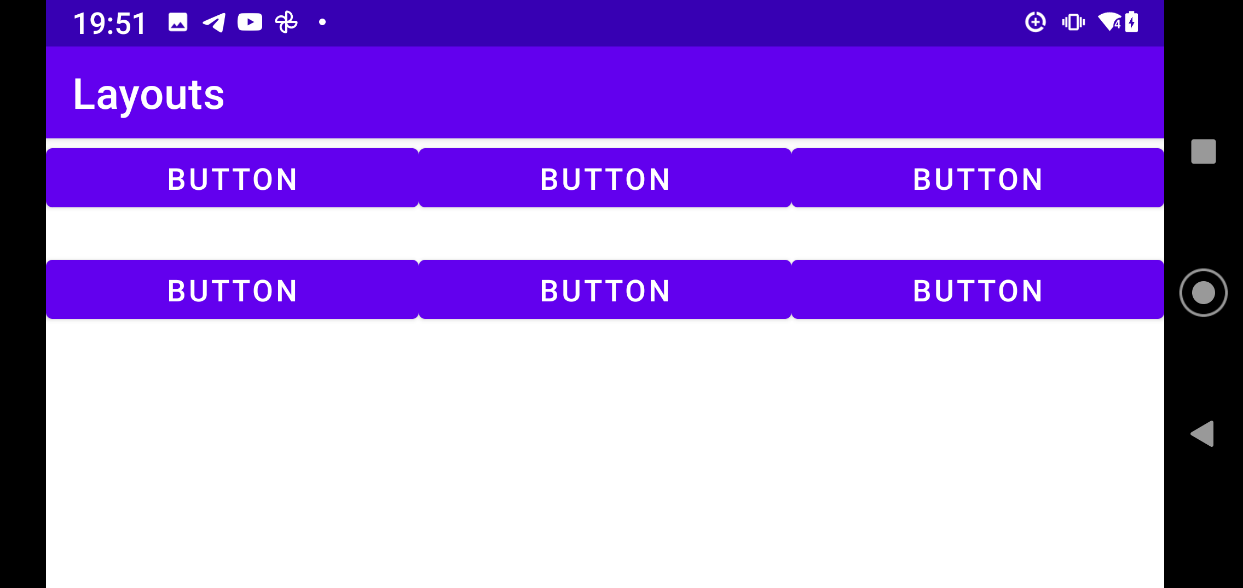


Рис. 22.

TableLayout (TL)

TL состоит из строк TableRow (TR). Каждая TR в свою очередь содержит View-элементы, формирующие столбцы. Т.е. кол-во View в TR - это кол-во столбцов. Но кол-во столбцов в таблице должно быть равным для всех строк. Поэтому, если в разных TR разное кол-во View-элементов (столбцов), то общее кол-во определяется по TR с максимальным кол-вом. Рассмотрим на примере.

Создадим layout-файл tlayout.xml. с корневым элементом TableLayout

Добавим в корневой TableLayout три TableRow-строки (из раздела Layouts слева) и в каждую строку добавим по две кнопки. Результат: наша таблица имеет три строки и два столбца.

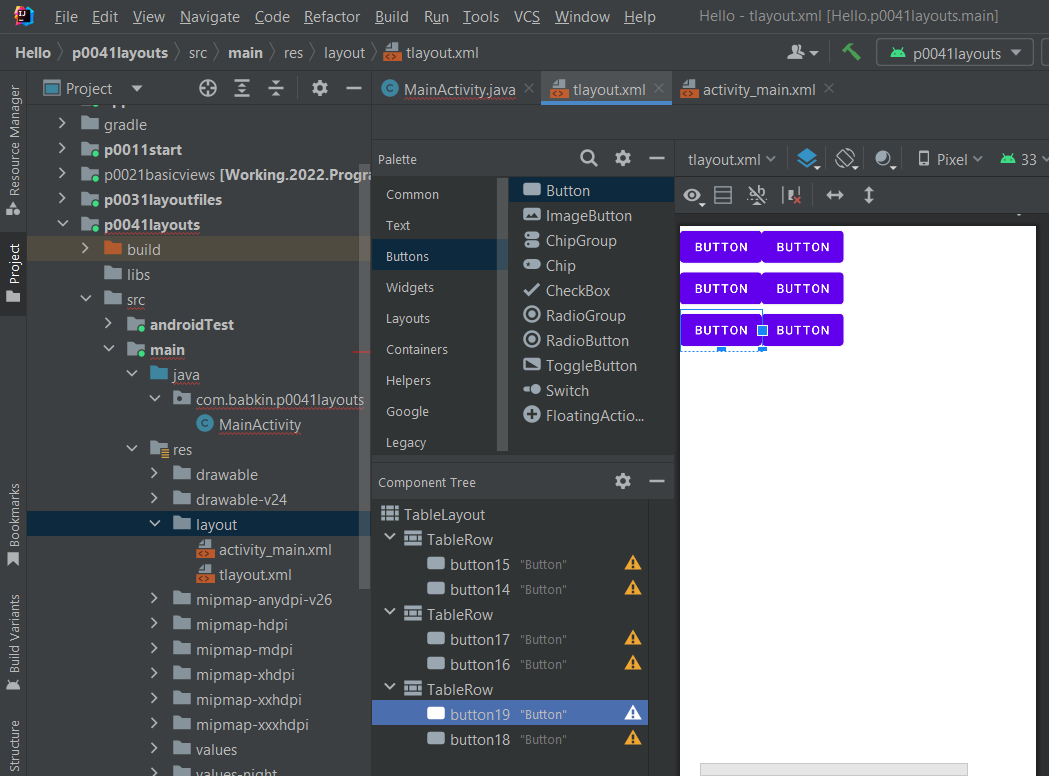


Рис. 23.

Добавим в первую строку еще пару кнопок. Кол-во столбцов для всех строк теперь равно 4, т.к. оно определяется по строке с максимальным количеством элементов, т.е. по первой строке. Для второй и третьей строки третий и четвертый столбцы просто ничем не заполнены.

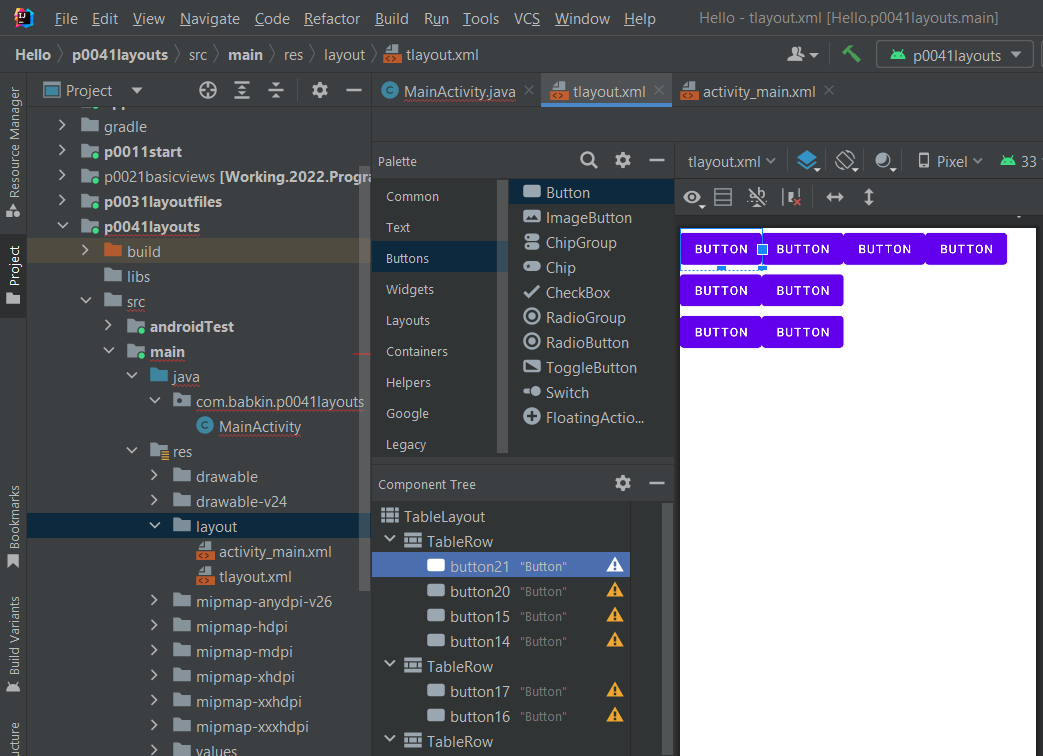


Рис. 24.

Во вторую строку добавим TextView и Button, и текст в добавленном TextView сделаем пустым. В третьей строке сделаем то же самое. Мы видим, что эти элементы легли в третий и четвертый столбец. И т.к. TextView у нас без текста и на экране не виден, кажется что третий столбец во второй и третьей строке пустой.

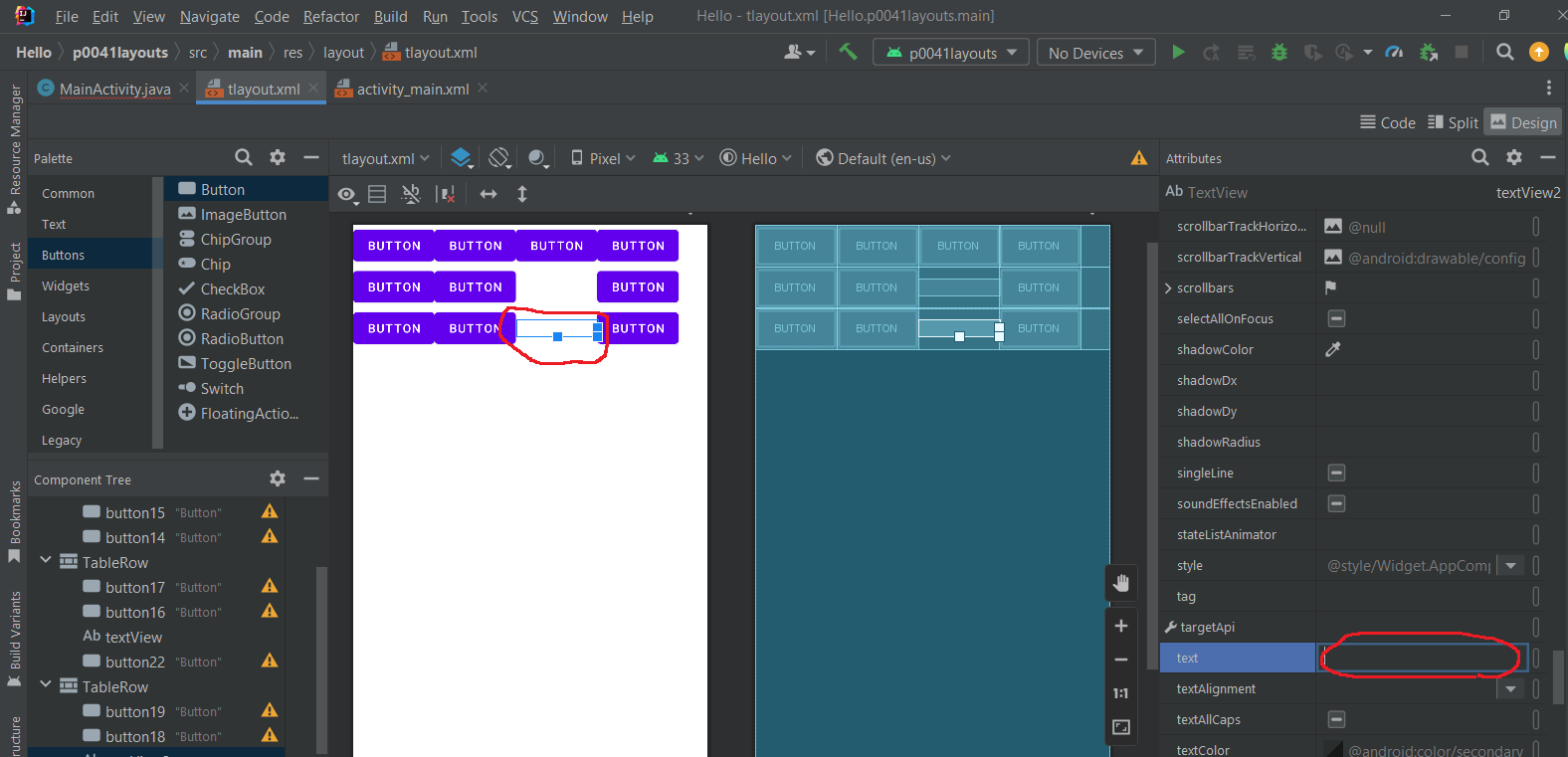


Рис. 25.

Ширина столбца определяется по самому широкому элементу из этого столбца. Введем текст в один из TextView и видим, что он расширил столбец.

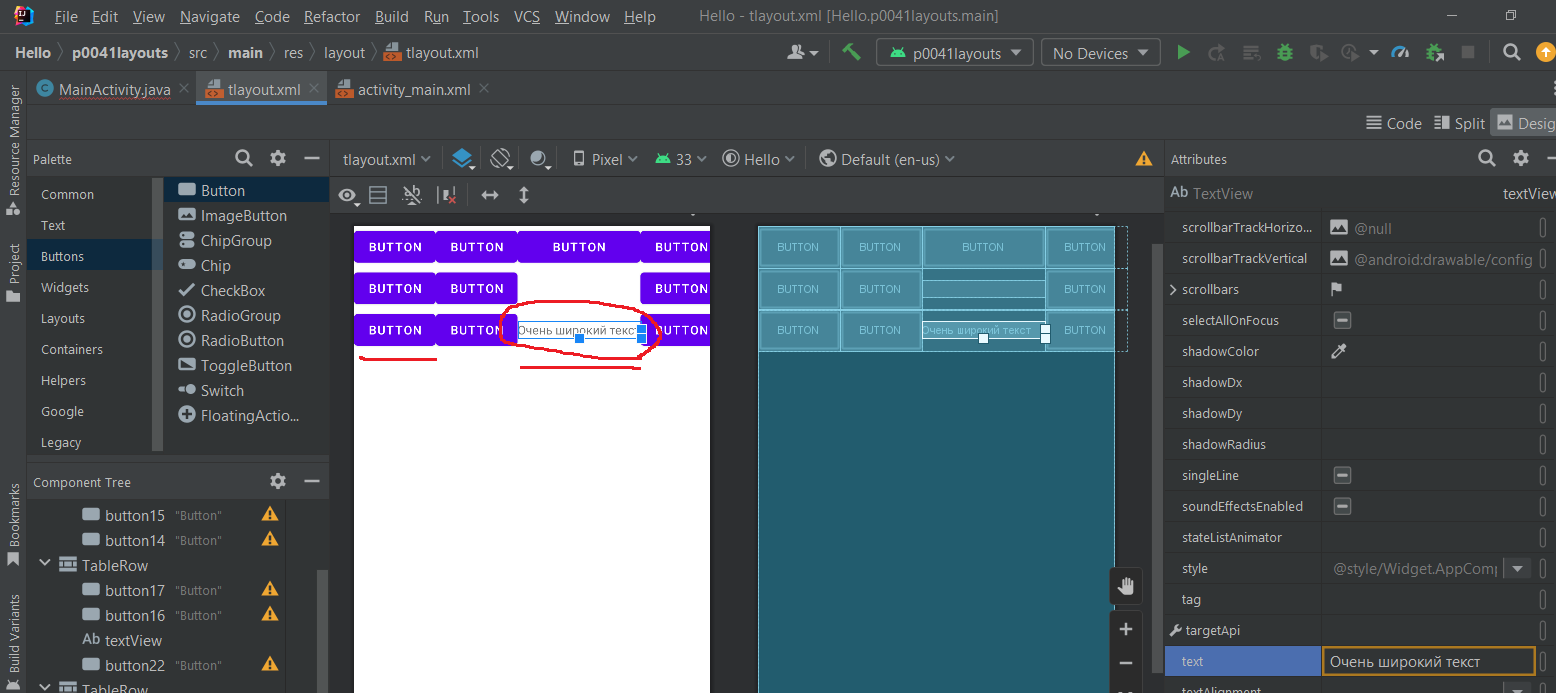


Рис. 26.

Попробуйте сами сделать так же в качестве упражнения.

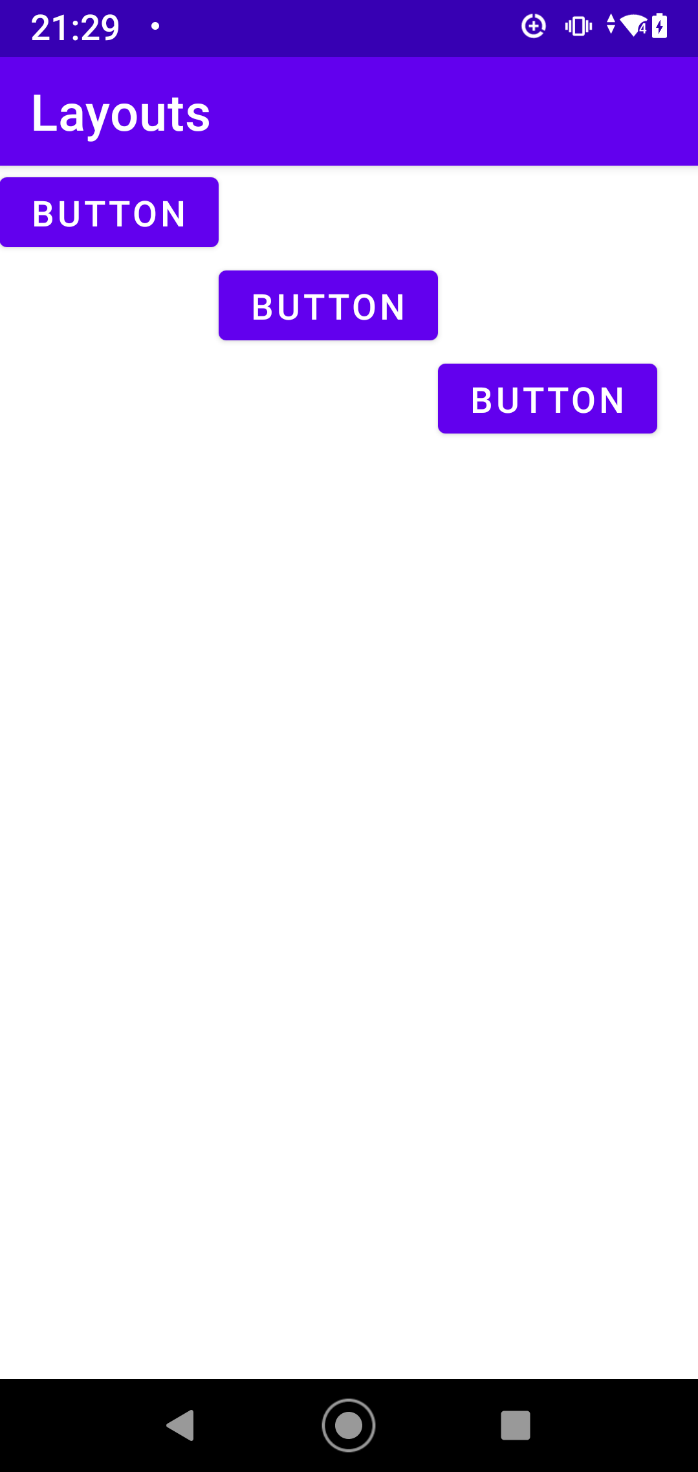


Рис. 27.

TL может содержать не только TR, но и обычные View. Добавьте, например, Button прямо в TL, а не в TR и увидите, что она растянулась на ширину всей таблицы.

RelativeLayout (RL)

В этом виде Layout каждый View-элемент может быть расположен определенным образом относительно указанного View-элемента.

Виды отношений:

1) слева, справа, сверху, снизу указанного элемента (layout\_toLeftOf, layout\_toRightOf, layout\_above, layout\_below)

2) выравненным по левому, правому, верхнему, нижнему краю указанного элемента (layout\_alignLeft, layout\_alignRight, layout\_alignTop, layout\_alignBottom)

3) выравненным по левому, правому, верхнему, нижнему краю родителя (layout\_alignParentLeft, layout\_alignParentRight, layout\_alignParentTop, layout\_alignParentBottom)

4) выравненным по центру вертикально, по центру горизонтально, по центру вертикально и горизонтально относительно родителя (layout\_centerVertical, layout\_centerHorizontal, layout\_centerInParent)

Подробно можно почитать в [хелпе](http://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html" \t "_blank).

 Создадим **rlayout.xml** и скопируем туда такой xml-код:

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout

    xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent">

    <TextView

        android:id="@+id/label"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:text="Type here:">

    </TextView>

    <EditText

        android:id="@+id/entry"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:layout\_below="@+id/label"

        android:background="@android:drawable/editbox\_background">

    </EditText>

    <Button

        android:id="@+id/ok"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:layout\_alignParentRight="true"

        android:layout\_below="@+id/entry"

        android:layout\_marginLeft="10dip"

        android:text="OK">

    </Button>

    <Button

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:layout\_alignTop="@+id/ok"

        android:layout\_toLeftOf="@+id/ok"

        android:text="Cancel">

    </Button>

</RelativeLayout>

Здесь у нас корневой элемент - RelativeLayout.

Получился такой экран:



Рис. 28.

Нам интересен xml-код. Сразу кратко опишу незнакомые атрибуты и их значения:

android:layout\_width="match\_parent"  
android:layout\_height="wrap\_content"  
android:id="@+id/entry"

- слово **android** в названии каждого атрибута – это [namespace](http://www.w3schools.com/xml/xml_namespaces.asp" \t "_blank), я его буду опускать при объяснениях.  
**- id** – это ID элемента,  
**- layout\_width** (ширина элемента) и **layout\_height** (высота элемента) могут задаваться в абсолютных значениях, а могут быть следующими: **fill\_parent** (максимально возможная ширина или высота в пределах родителя) и **wrap\_content** (ширина или высота определяется по содержимому элемента). В [хелпе](http://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html" \t "_blank) указывается, что есть еще **match\_parent**. Это тоже самое, что и fill\_parent. По каким-то причинам, разработчики системы решили, что название match\_parent удобнее, и от fill\_parent постепенно будут отказываться. А пока его оставили для совместимости. Так что запомните, что **match\_parent = fill\_parent** и в дальнейшем будем стараться использовать **match\_parent**. Позже мы еще остановимся на этом и разберем подробнее.

Сейчас вернемся к нашим элементам. В примере мы видим TextView, EditText и два Button – OK и Cancel. Давайте подробно разберем интересующие нас атрибуты.

TextView  
android:id="@+id/label" - ID  
android:layout\_width="match\_parent" - занимает всю доступную ему ширину (хоть это и не видно на экране);  
android:layout\_height="wrap\_content" – высота по содержимому;

EditText  
android:id="@+id/entry" - ID  
android:layout\_width="match\_parent" – вся доступная ему ширина  
android:layout\_height="wrap\_content" – высота по содержимому  
android:layout\_below="@+id/label" - расположен **ниже**TextView (ссылка по ID)

Button\_OK  
android:id="@+id/ok" – ID  
android:layout\_width="wrap\_content" – ширина по содержимому  
android:layout\_height="wrap\_content" – высота по содержимому  
android:layout\_below="@+id/entry" – расположен ниже EditText  
android:layout\_alignParentRight="true" - **выравнен по правому краю родителя**  
android:layout\_marginLeft="10dip" – имеет отступ слева (чтобы Button\_Cancel был не впритык)

Button\_Cancel  
android:layout\_width="wrap\_content" – ширина по содержимому  
android:layout\_height="wrap\_content" – высота по содержимому  
android:layout\_toLeftOf="@+id/ok" - расположен **слева**от Button\_OK  
android:layout\_alignTop="@+id/ok" - **выравнен по верхнему краю** Button\_OK

Вы можете подобавлять элементы и поэкспериментировать с их размещением.

Обратите внимание, что у View-элемента может не быть ID (android:id). Например, для TextView он обычно не нужен, т.к. они чаще всего статичны и мы к ним почти не обращаемся при работе приложения. Другое дело EditText – мы работаем с содержимым текстового поля, и Button – нам надо обрабатывать нажатия и соответственно знать, какая именно кнопка нажата. В будущем мы увидим еще одну необходимость задания ID для View-элемента.

AbsoluteLayout (AL)

Обеспечивает абсолютное позиционирование элементов на экране. Вы указываете координаты для левого верхнего угла компонента.

Создадим **alayout.xml** с корневым AbsoluteLayout

Теперь попробуйте перетаскиванием подобавлять различные элементы на экран. Они не выстраиваются, как при LinearLayout или TableLayout, а ложатся там, куда вы их перетащили. Т.е. это абсолютное позиционирование.

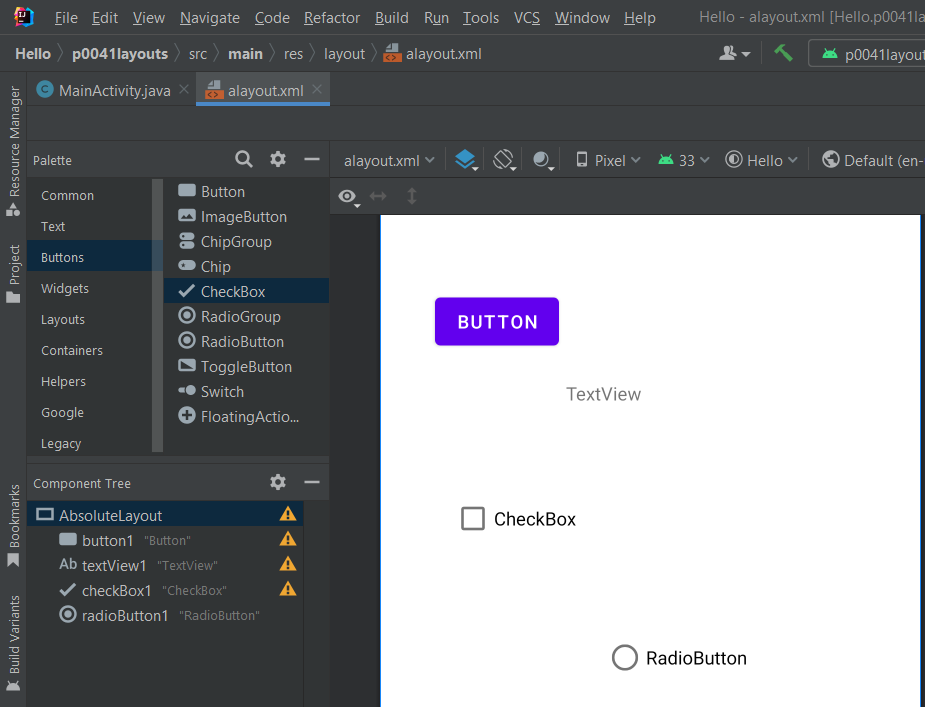


Рис. 29.

Поначалу кажется, что это наиболее удобный и интуитивно понятный способ расположения элементов на экране - они сразу располагаются там где надо. Но это только в случае, когда вы разрабатываете для экрана с конкретным разрешением. Если открыть такое приложение на другом экране, все элементы сместятся и получится не так, как вы планировали. Поэтому этот Layout не рекомендуется использовать. И его совместимость с будущими версиями Android не гарантируется.

Есть еще [много видов](http://developer.android.com/guide/topics/ui/layout-objects.html#viewgroupsummary) ViewGroup, и мы постепенно будем их осваивать. А пока нам хватит этих.