## Представления элементов

### Цель работы

Познакомится с основными приемами визуального представления элементов приложения, управления их взаимным расположением, оформлением, работы с соответствующими атрибутами из XML и из Java кода.

### Методические указания

#### Виды представлений

В Android SDK существует множество способов группировки элементов, которые называются представлениями (layout). На данном занятии мы изучим основные виды представлений.

Одним из наиболее простых в понимании является AbsoluteLayout (AL). Он обеспечивает абсолютное позиционирование элементов на экране. Вы указываете координаты для левого верхнего угла компонента.

Рассмотрим произвольный пример:

<AbsoluteLayout

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

android:layout\_width="match\_parent"

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

<Button

android:id="@+id/button1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_x="42dp"

android:layout\_y="62dp"

android:text="Button">

</Button>

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_x="142dp"

android:layout\_y="131dp"

android:text="TextView">

</TextView>

<CheckBox

android:id="@+id/checkBox1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_x="55dp"

android:layout\_y="212dp"

android:text="CheckBox">

</CheckBox>

<RadioButton

android:id="@+id/radioButton1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_x="171dp"

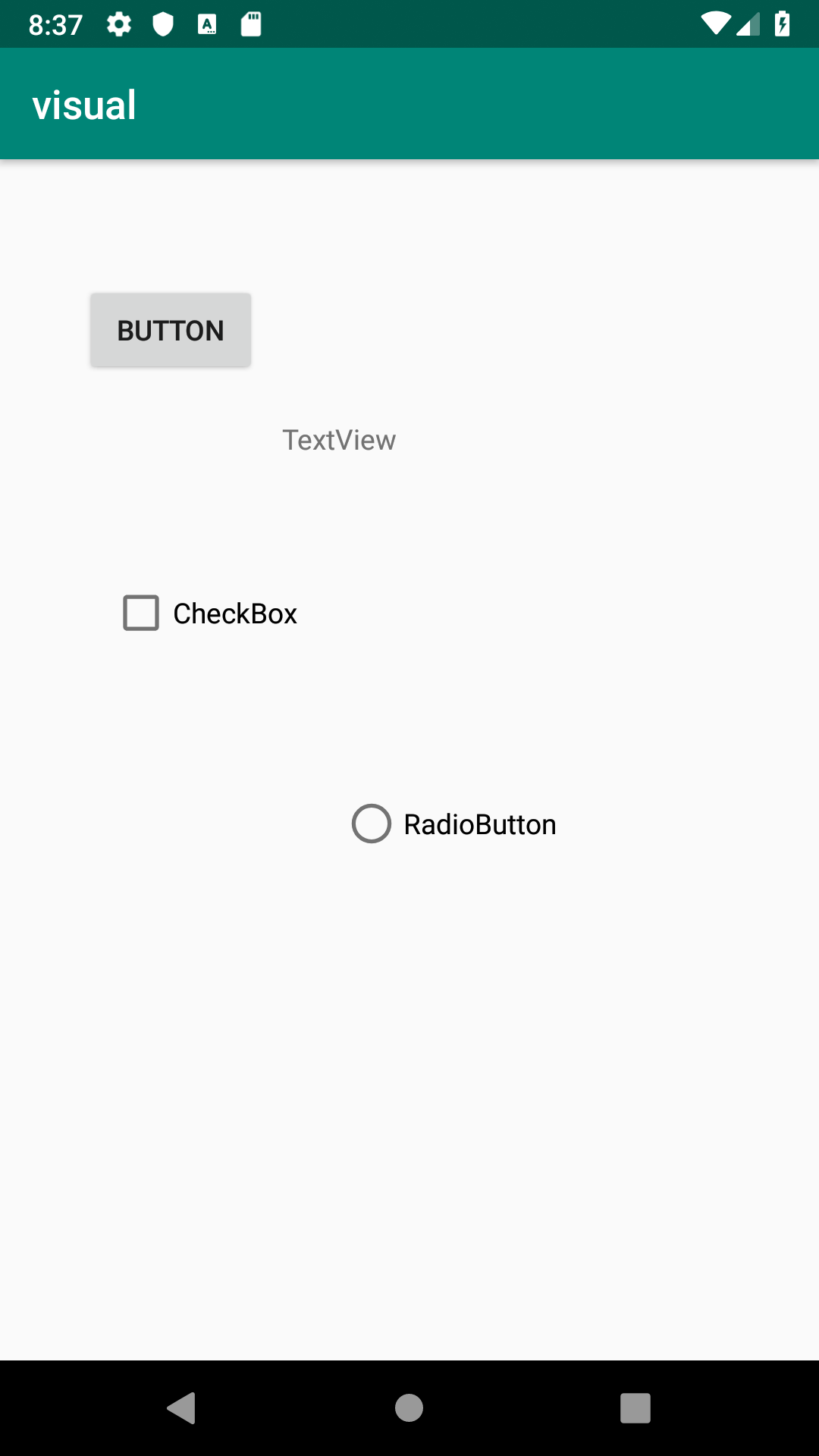
android:layout\_y="318dp"

android:text="RadioButton">

</RadioButton>

</AbsoluteLayout>

Проверим, как располагаются элементы на экране:



Обратите внимание, что этот элемент считается устаревшим и не рекомендуется к использованию в серьезных приложения.

Наверное, вторым по простоте является линейное представление. Оно размещает все элементы, содержащиеся в нем, последовательно. Создадим простое представление с тремя кнопками.

<LinearLayout

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button" />

<Button

android:id="@+id/button2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button" />

<Button

android:id="@+id/button3"

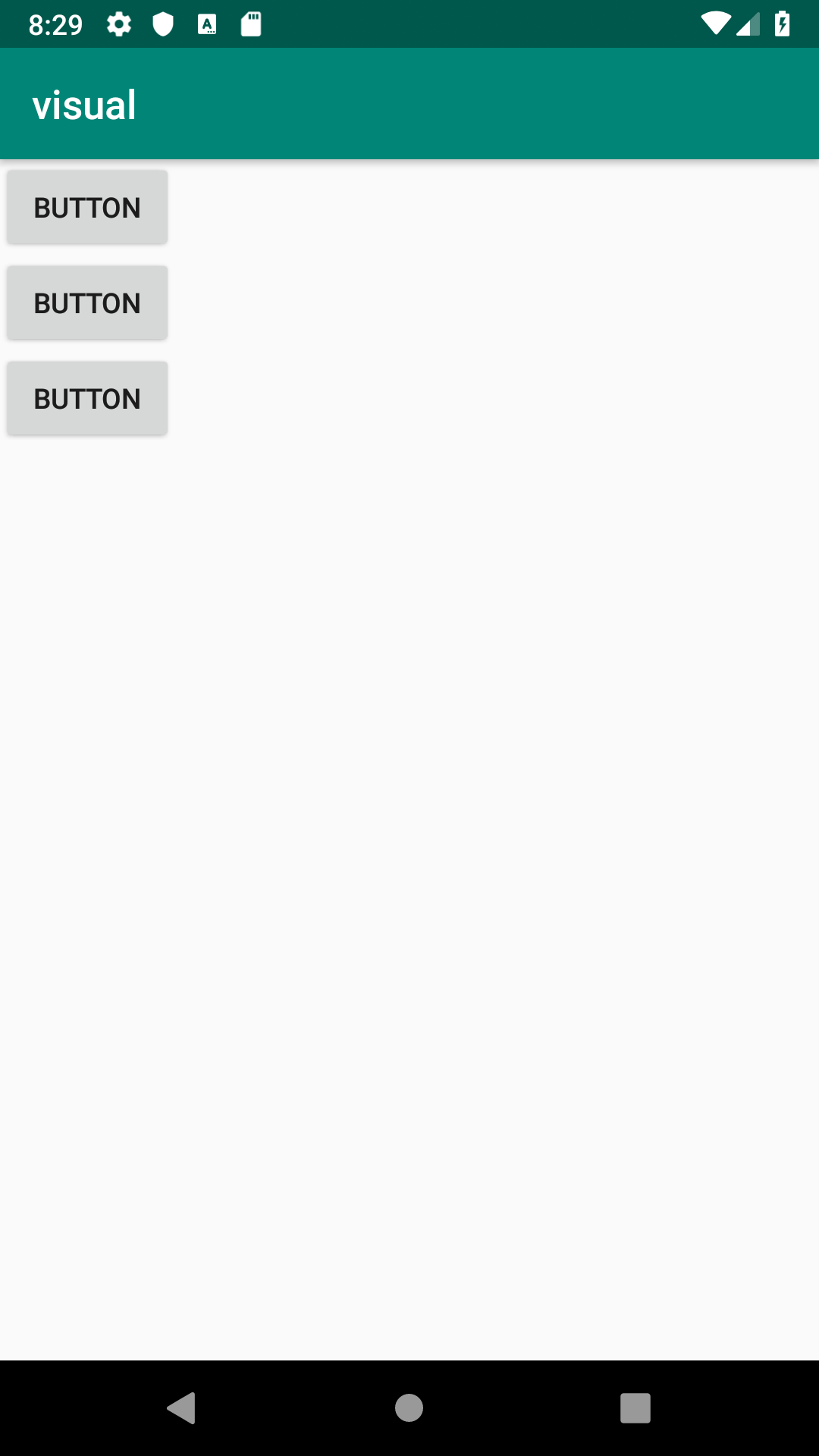
android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button" />

</LinearLayout>

Обратите внимание на атрибут orientation элемента LinearLayout. Он задает направление элементов. В данном случае, три кнопки будут расположены вертикально, то есть одна под другой:



Давайте заменим расположение элементов на горизонтальное, изменив значение соответствующего атрибута:

android:orientation="horizontal">

Мы увидим, что теперь элементы располагаются в ряд, горизонтально.



Помните, что вы можете вкладывать одно представление в другое, комбинируя в произвольном порядке типы упорядочения элементов и получая расположения произвольной сложности.

Как пример, давайте расположим шесть кнопок в два ряда по три кнопки. Для этого используем только LinearLayout (хотя, существуют более явные способы добиться такого расположения):

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

<LinearLayout

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button" />

<Button

android:id="@+id/button2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button" />

<Button

android:id="@+id/button3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button" />

</LinearLayout>

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<Button

android:id="@+id/button4"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="Button" />

<Button

android:id="@+id/button5"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="Button" />

<Button

android:id="@+id/button6"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

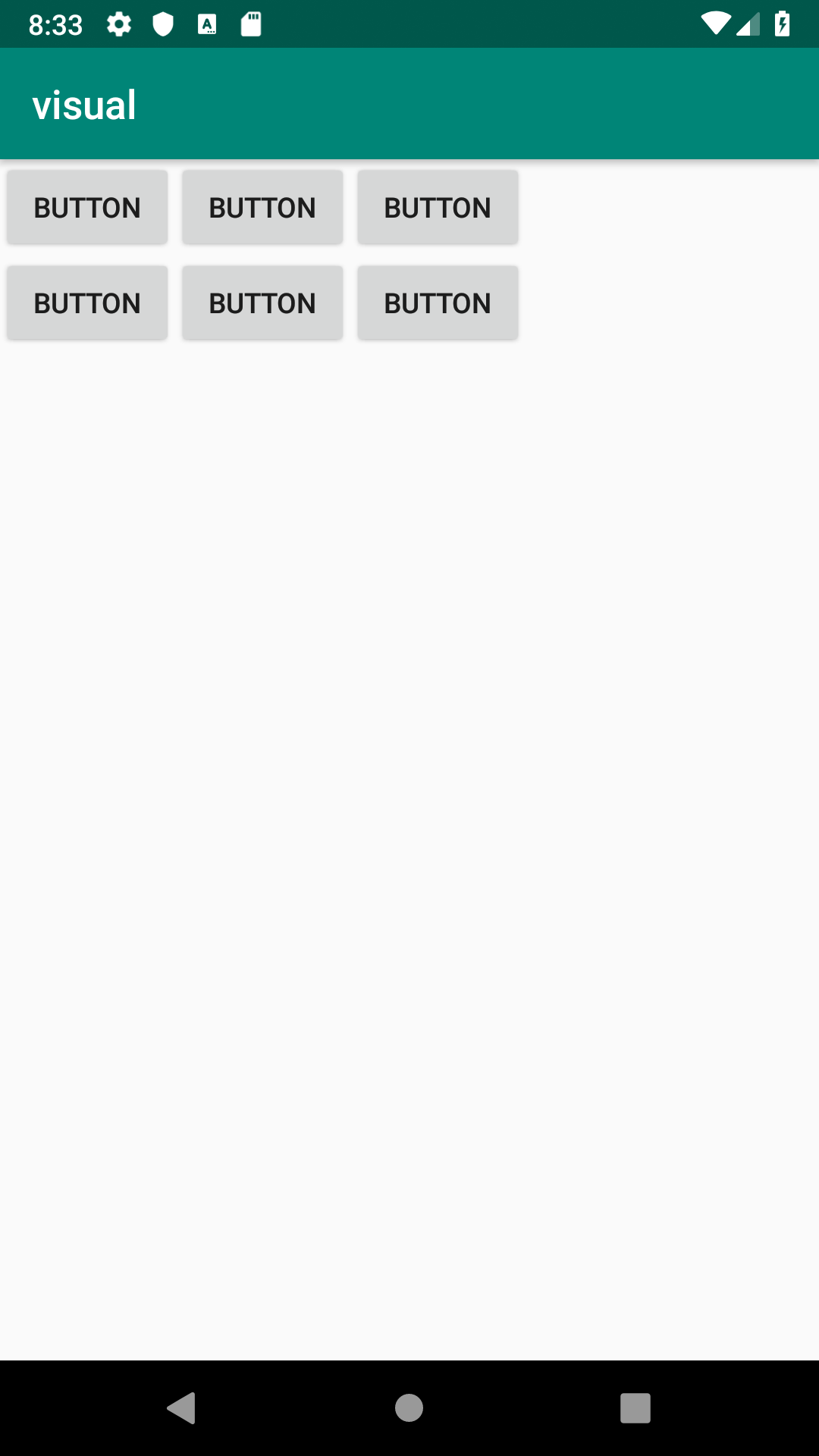
android:layout\_weight="1"

android:text="Button" />

</LinearLayout>

</LinearLayout>

Расположение элементов в этом приложении не должно нас удивлять:



Еще одним типом расположения является RelativeLayout (RL). В этом виде представления каждый элемент может быть расположен определенным образом относительно другого указанного элемента. Используются следующие виды отношений (подробнее в [документации](https://developer.android.com/guide/topics/ui/layout/relative)):

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

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

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

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

Это представление удобно использовать по смыслу, когда у вас в приложении вид и положение на экране нескольких элементов должны быть координированы между собой. Например, мы хотим создать поле ввода на всю ширину экрана, а под ним две кнопки так, чтобы они были выровнены справа к границе этого поля ввода. Для этого проще всего описать это расположения с помощью атрибутов RelativeLayout:

<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"

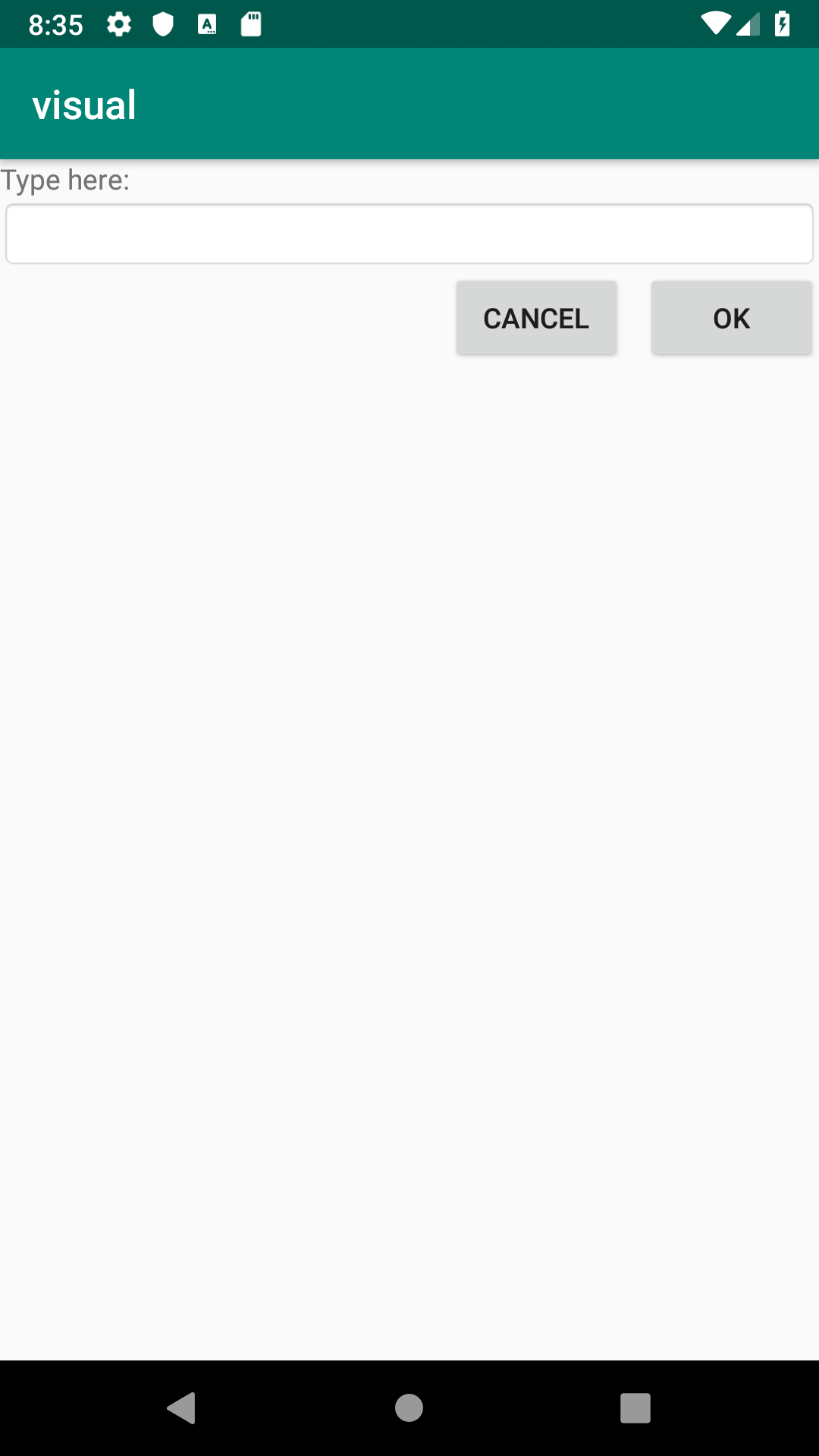
android:layout\_toLeftOf="@+id/ok"

android:text="Cancel">

</Button>

</RelativeLayout>

Проверим расположение элементов на реальном приложении:



Еще одним простым представлением является табличное (TableLayout). Оно состоит из строк и вложенных элементов и, в принципе, аналогично элементу <TABLE> в языке HTML.

Давайте рассмотрим простой пример:

<TableLayout android:layout\_height="wrap\_content"

xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="wrap\_content">

<TableRow

android:layout\_width="match\_parent"

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

<Button

android:id="@+id/button7"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button" />

<Button

android:id="@+id/button8"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button" />

</TableRow>

<TableRow

android:layout\_width="match\_parent"

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

<Button

android:id="@+id/button9"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

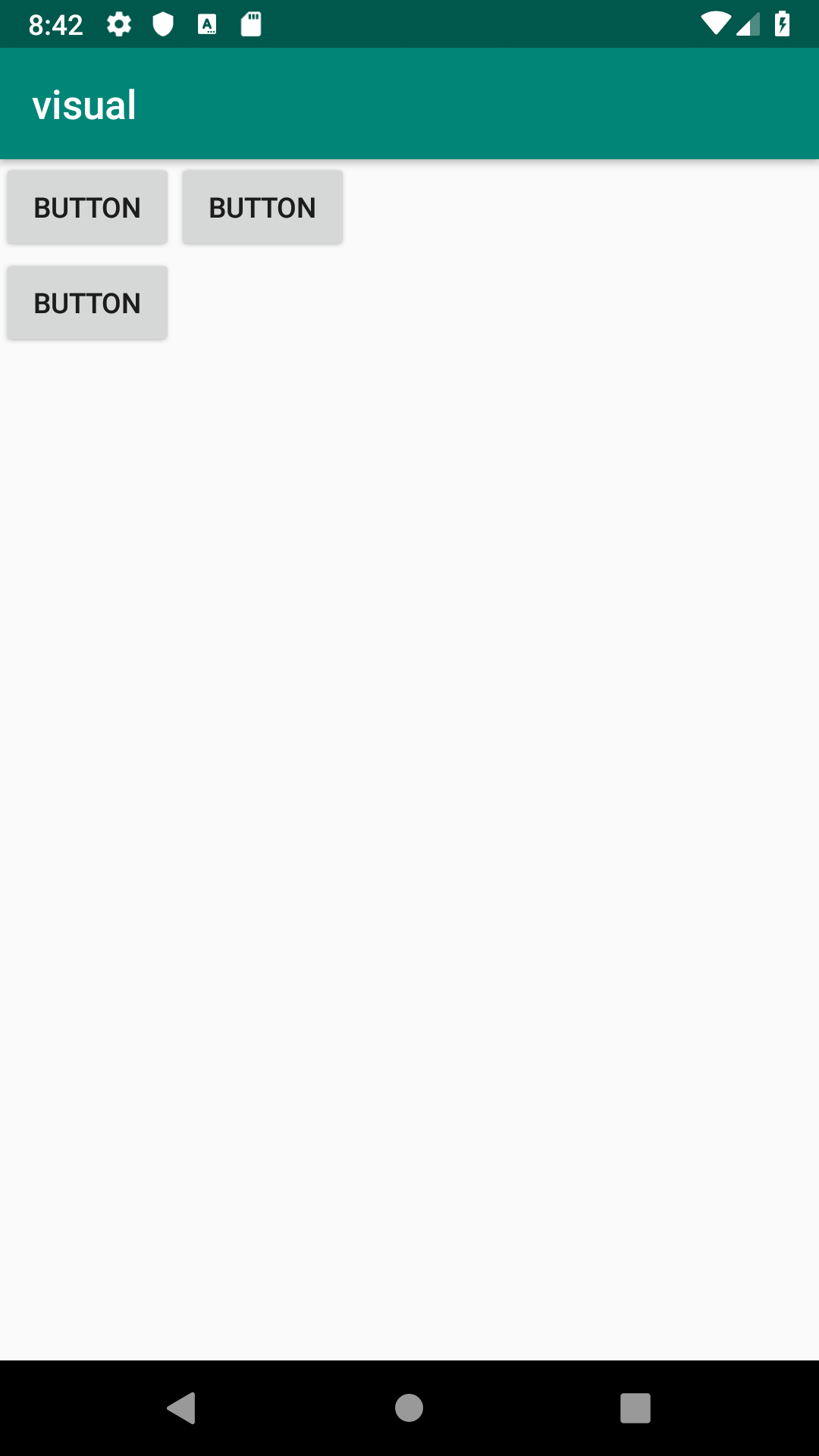
android:text="Button" />

</TableRow>

</TableLayout>

Здесь мы видим таблицу, состоящую из двух строк. В первой из них два элемента, во второй - один.

Проверим расположение элементов в приложении:



#### Визуальная настройка элементов

Android SDK предоставляет широкий набор возможностей для настройки визуальных параметров отображения элементов. Большинство из них настраиваются изменением значений соответствующих атрибутов элементов XML-элемента.

Про ширину (layout\_width) и высоту (layout\_height) мы уже немного знаем. Мы можем указывать для них абсолютные значения, а можем использовать константы. Разберем подробнее эти возможности.

Абсолютные значения:

Используются следующие единицы измерения (ЕИ):

dp - Density-independent Pixels. Абстрактная единица, позволяющая приложениям выглядеть одинаково на различных экранах и разрешениях;

sp - Scale-independent Pixels. То же, что и dp, только используется для размеров шрифта в элементах;

pt - 1/72 дюйма, определяется по физическому размеру экрана. Это единица из типографии;

px – пиксель, не рекомендуется использовать т. к. на разных экранах приложение будет выглядеть по-разному;

mm – миллиметр, определяется по физическому размеру экрана;

in – дюйм, определяется по физическому размеру экрана;

Константы

match\_parent (fill\_parent) – означает, что элемент займет всю доступную ему в родительском элементе ширину/высоту;

wrap\_content – ширина/высота элемента будет определяться его содержимым;

Для примера рассмотрим простое линейное представление, состоящее из двух кнопок:

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

android:layout\_width="match\_parent"

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

<Button

android:id="@+id/button10"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="B1" />

<Button

android:id="@+id/button11"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="B1" />

</LinearLayout>

Данное представление будет выглядеть так:



Обратите внимание, что кнопки занимают столько места по ширине, сколько требует их содержимое (в данном случае, текст).

Давайте изменим значение атрибута:

android:layout\_width="match\_parent

Обратите внимание, что теперь кнопки расползлись на всю ширину экрана, причем они равны по ширине.



Теперь воспользуемся атрибутом layout\_weight, который позволяет создавать неравномерные по размеру элементы. Присвоим значение 1 первой кнопке:

android:layout\_weight="1"

А второй - значение 4:

android:layout\_weight="4"

Проверим, что теперь вторая кнопка в 4 раза шире первой:



Еще одним часто применяемым атрибутом является layout\_margin, который позволяет настроить отступы от элемента со всех сторон.

Давайте рассмотрим элементы, расположенные равномерно в таблице 3 на 3:

<TableLayout

android:layout\_height="wrap\_content"

android:layout\_width="match\_parent"

android:id="@+id/tableLayout1">

<TableRow

android:id="@+id/tableRow1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<Button

android:text="Button"

android:id="@+id/button1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

</Button>

<Button

android:text="Button"

android:id="@+id/button2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

</Button>

<Button

android:text="Button"

android:id="@+id/button3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

</Button>

</TableRow>

<TableRow

android:id="@+id/tableRow2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<Button

android:text="Button"

android:id="@+id/button4"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

</Button>

<Button

android:text="Margin"

android:id="@+id/button5"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

</Button>

<Button

android:text="Button"

android:id="@+id/button6"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

</Button>

</TableRow>

<TableRow

android:id="@+id/tableRow3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<Button

android:text="Button"

android:id="@+id/button7"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

</Button>

<Button

android:text="Button"

android:id="@+id/button8"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

</Button>

<Button

android:text="Button"

android:id="@+id/button9"

android:layout\_width="wrap\_content"

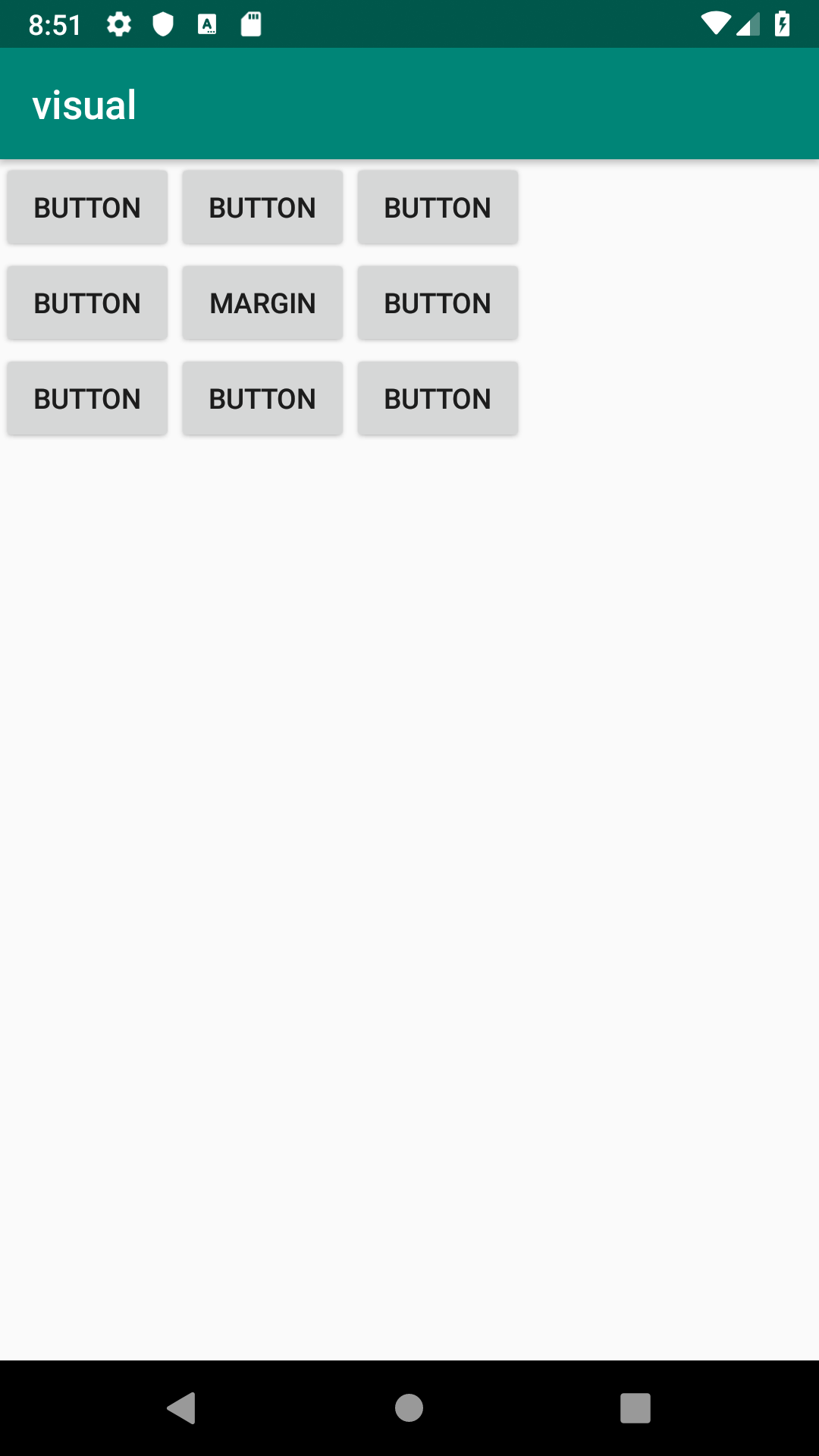
android:layout\_height="wrap\_content">

</Button>

</TableRow>

</TableLayout>

При запуске мы увидим красивую и ровную таблицу:

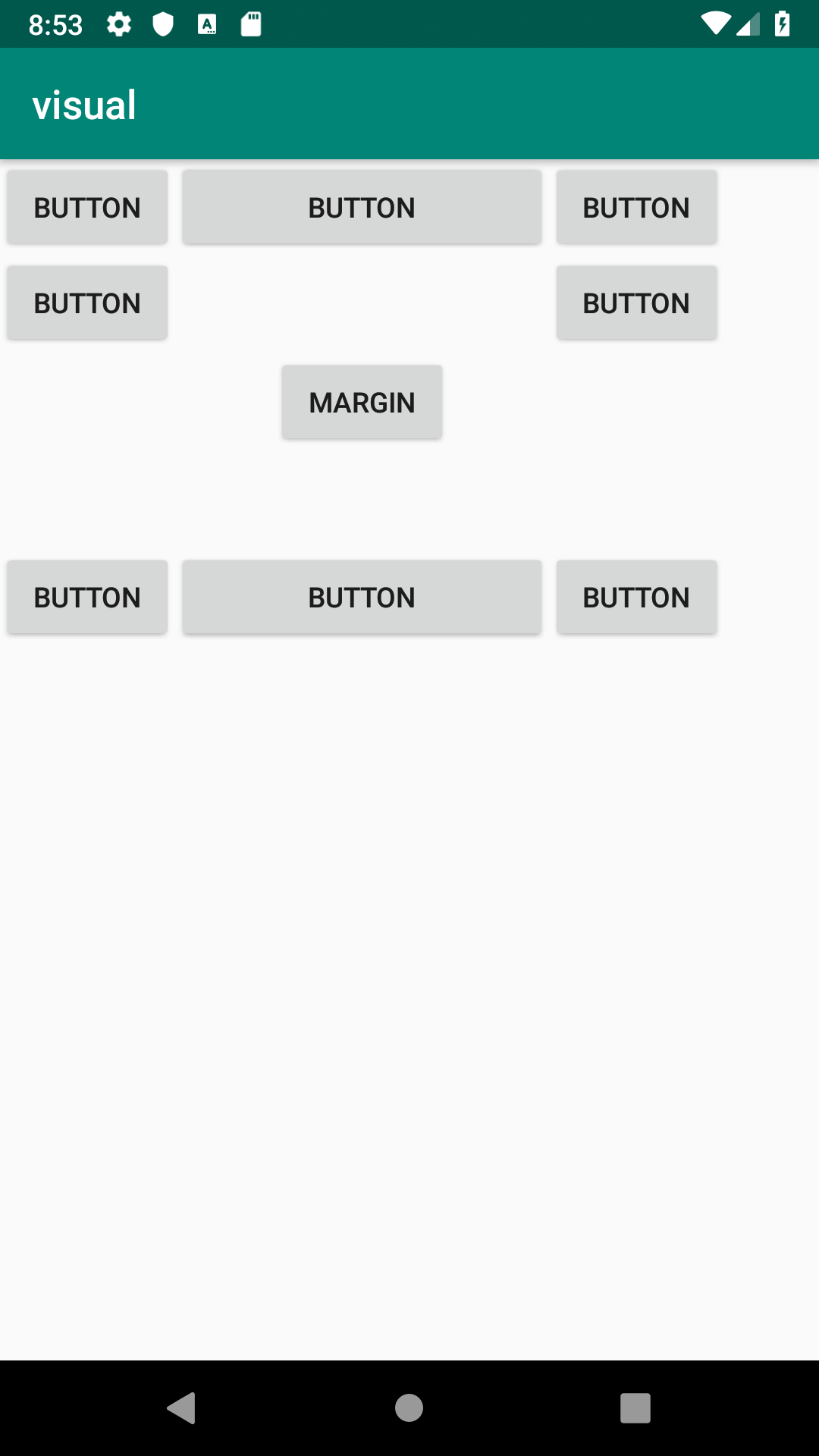


В такой таблице все отступы будут особенно видны.

Найдем среднюю кнопку (вторая во второй строке) и зададим ей отступ со всех сторон в 50 пикселей:

android:layout\_margin="50dp">

Увидим, как изменилась наша таблица:

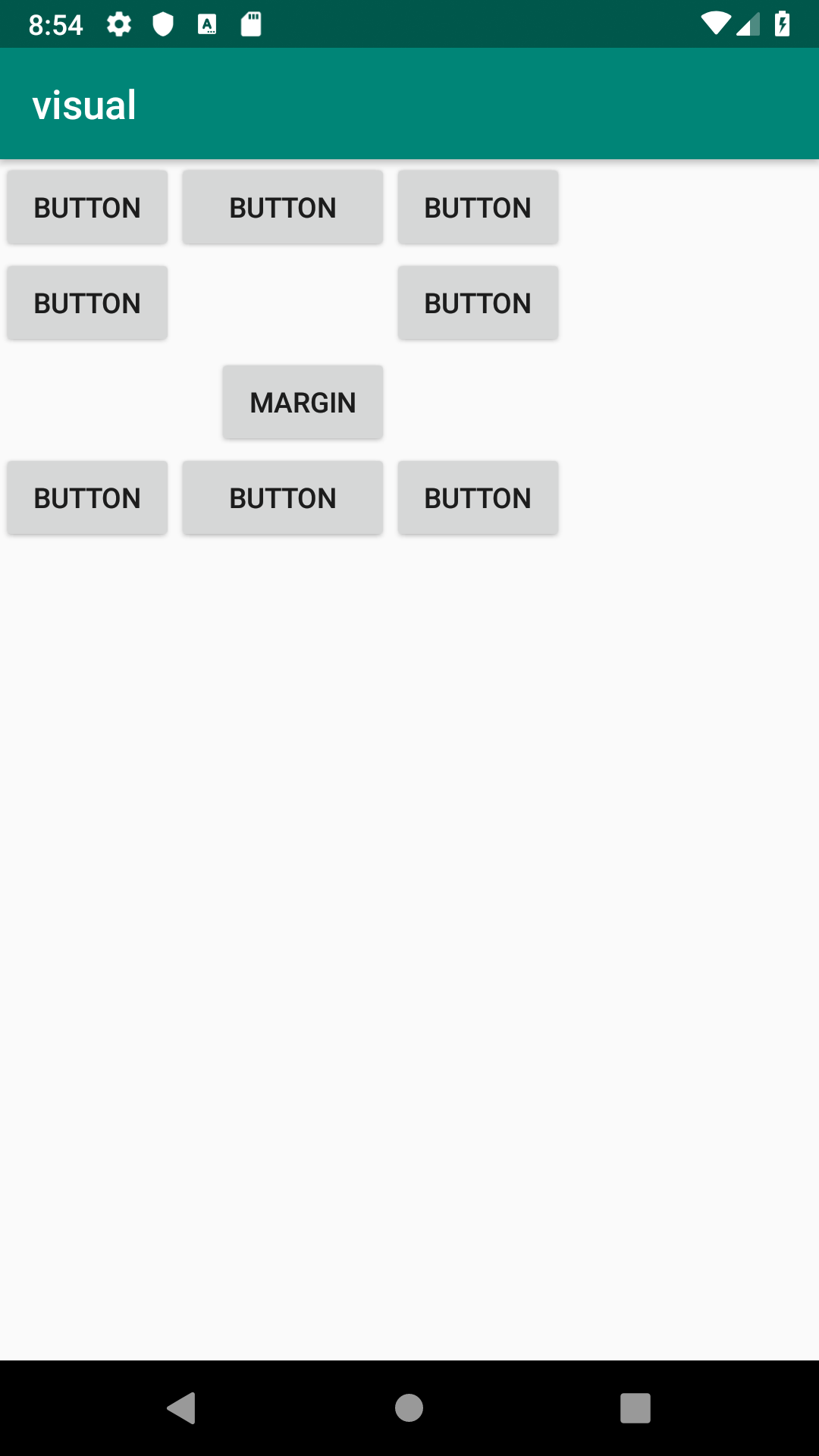


Так же как и в HTML, можно задавать отступы от элемента с каждой из четырех сторон независимо, например так:

android:layout\_marginTop="50dp"

android:layout\_marginLeft="20dp">

Проверим изменения:



Параметр layout\_gravity аналогичен выравниванию из Word или Excel. Удобнее всего продемонстрировать его с использованием FrameLayout. Все помещаемые в него элементы он по умолчанию помещает в левый верхний угол и никак их не выстраивает. Нам это очень подходит для демонстрации настроек выравнивания.

Продемонстрируем пять элементов, каждый со своей настройкой:

<FrameLayout

android:id="@+id/frameLayout1"

android:layout\_width="match\_parent"

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

<Button

android:id="@+id/button1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="top|left"

android:text="gravity = top left">

</Button>

<Button

android:id="@+id/button2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="top|right"

android:text="gravity = top right">

</Button>

<Button

android:id="@+id/button3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="bottom|left"

android:text="gravity = bottom left">

</Button>

<Button

android:id="@+id/button4"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="bottom|right"

android:text="gravity = bottom right">

</Button>

<Button

android:id="@+id/button5"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

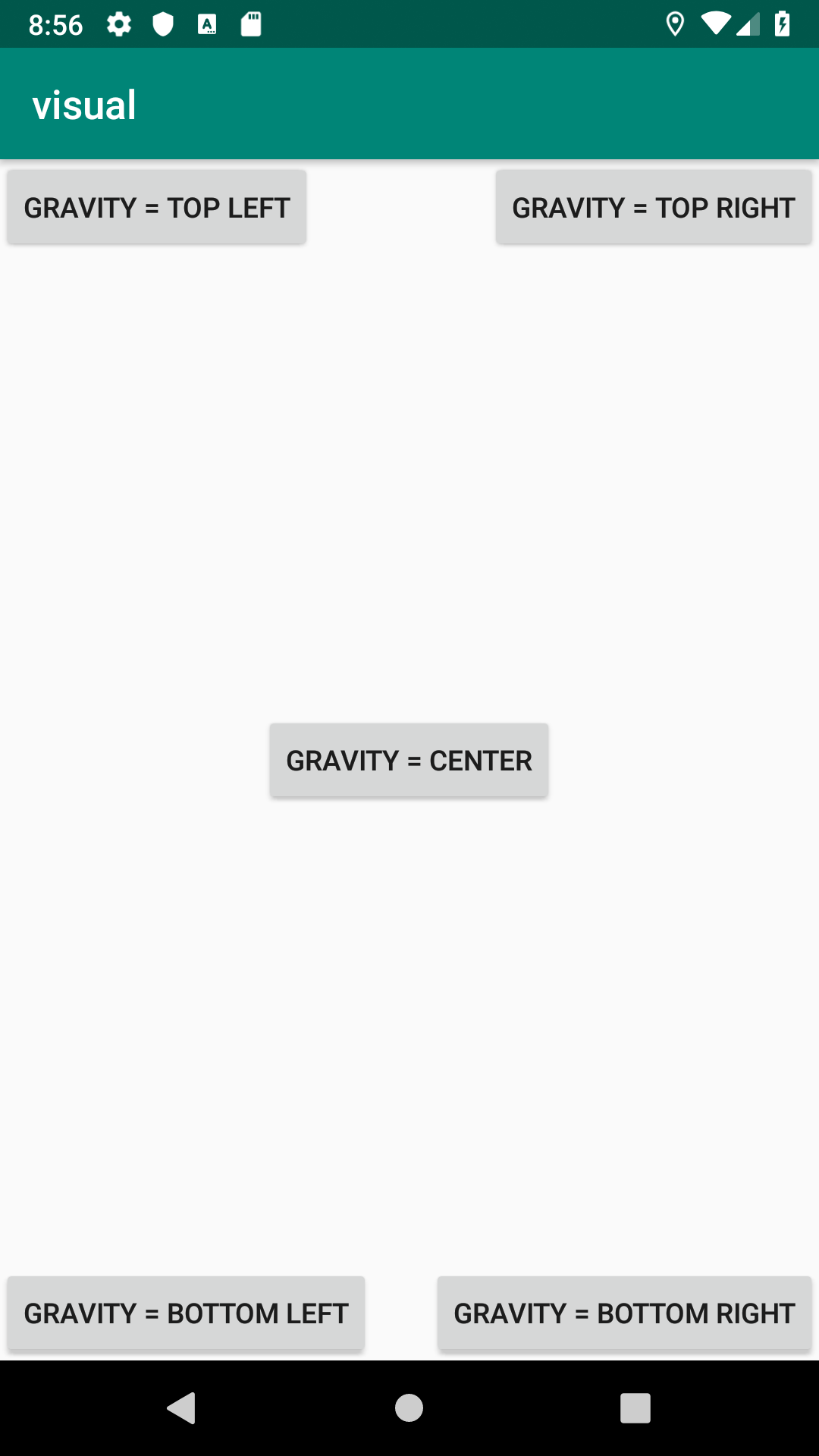
android:layout\_gravity="center"

android:text="gravity = center">

</Button>

</FrameLayout>

Запустим приложение и проверим расположение элементов:



#### Использование ресурсов

Значения атрибутов элементов приложения, как мы знаем, могут быть ссылками на ресурсы. Раньше мы использовали это только для выделения строк. Но этот же механизм можно использовать для создания аналога стилей в приложении. Давайте рассмотрим, как можно воспользоваться файлами ресурсов для изолированного хранения значений атрибутов оформления элементов.

Заодно рассмотрим еще несколько атрибутов оформления.

Создадим простое окно, состоящее из двух одинаковых частей: верхней и нижней. В каждой расположен надпись и кнопка

<LinearLayout

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

android:layout\_height="match\_parent"

android:layout\_width="match\_parent"

android:orientation="vertical">

<LinearLayout

android:layout\_width="match\_parent"

android:id="@+id/llTop"

android:orientation="vertical"

android:layout\_weight="1"

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

<TextView

android:text="TextView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:id="@+id/tvTop"

android:layout\_marginTop="30dp">

</TextView>

<Button

android:text="Button"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:id="@+id/btnTop"

android:layout\_width="wrap\_content">

</Button>

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:id="@+id/llBottom"

android:orientation="vertical"

android:layout\_weight="1"

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

<TextView

android:text="TextView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:id="@+id/tvBottom"

android:layout\_marginTop="30dp">

</TextView>

<Button

android:text="Button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

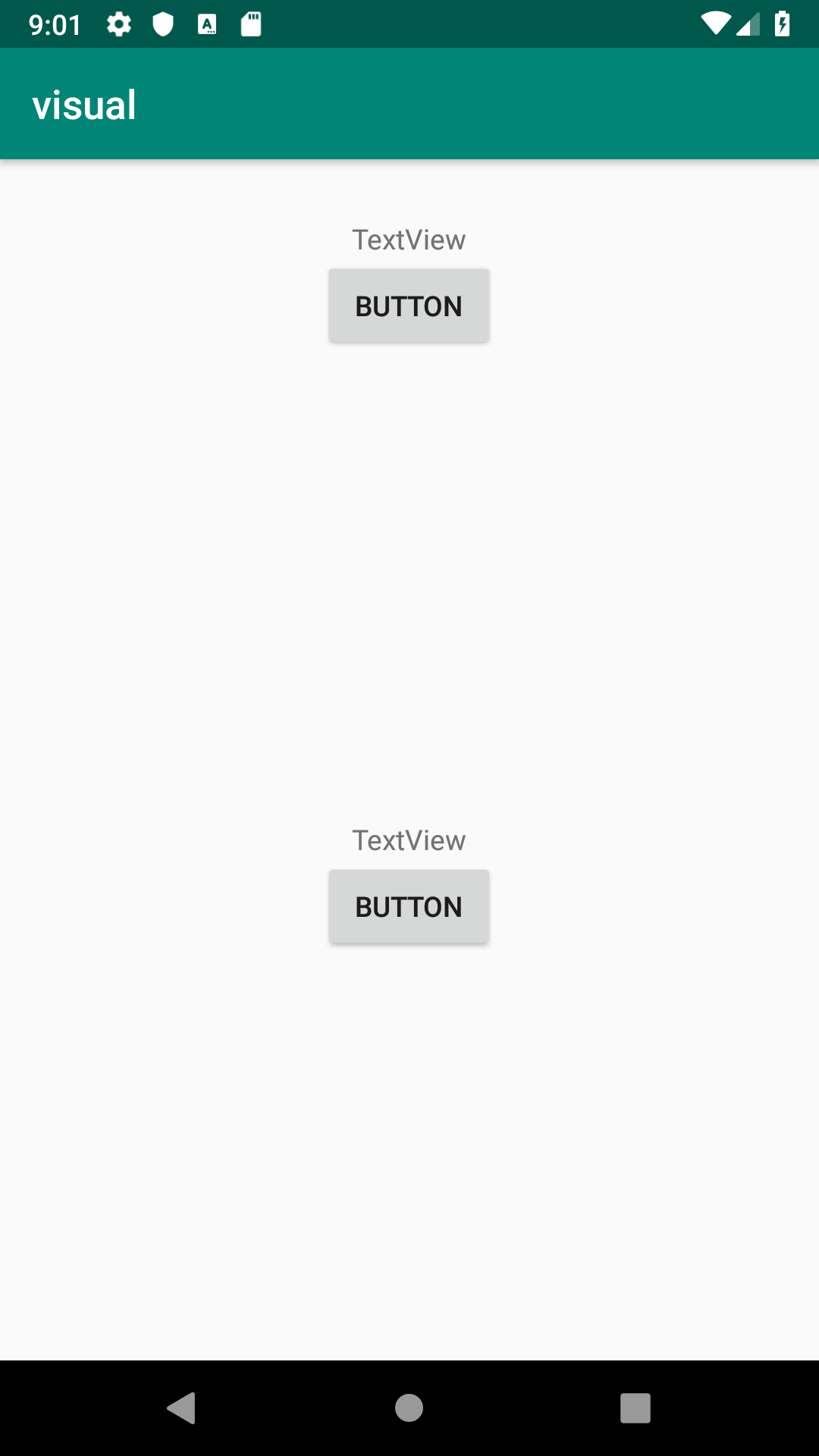
android:id="@+id/btnBottom">

</Button>

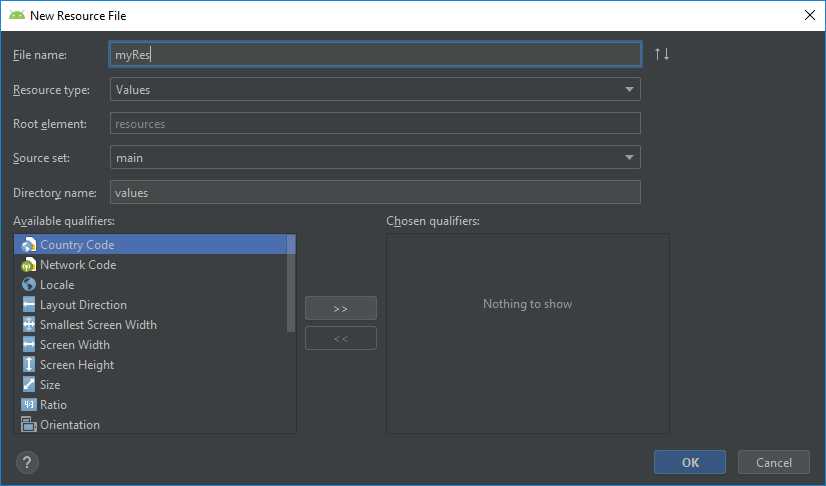
</LinearLayout>

</LinearLayout>

По умолчанию, приложение выглядит так:



Теперь добавим оформления. Для того, чтобы все ресурсы, используемые для оформления этого приложения хранились вместе, создадим новый файл ресурсов и назовем его, скажем, MyRes:



Добавим в этот файл несколько необходимых строк и цветов:

<resources>

<string name="tvTopText">Верхний текст</string>

<string name="btnTopText">Верхняя кнопка</string>

<string name="tvBottomText">Нижний текст</string>

<string name="btnBottomText">Нижняя кнопка</string>

<color name="llTopColor">#336699</color>

<color name="llBottomColor">#339966</color>

</resources>

Теперь мы можем делать ссылки на наши новые ресурсы. Зададим, например, фоновый цвет верхнего LinearLayout:

android:background="@color/llTopColor"

android:orientation="vertical">

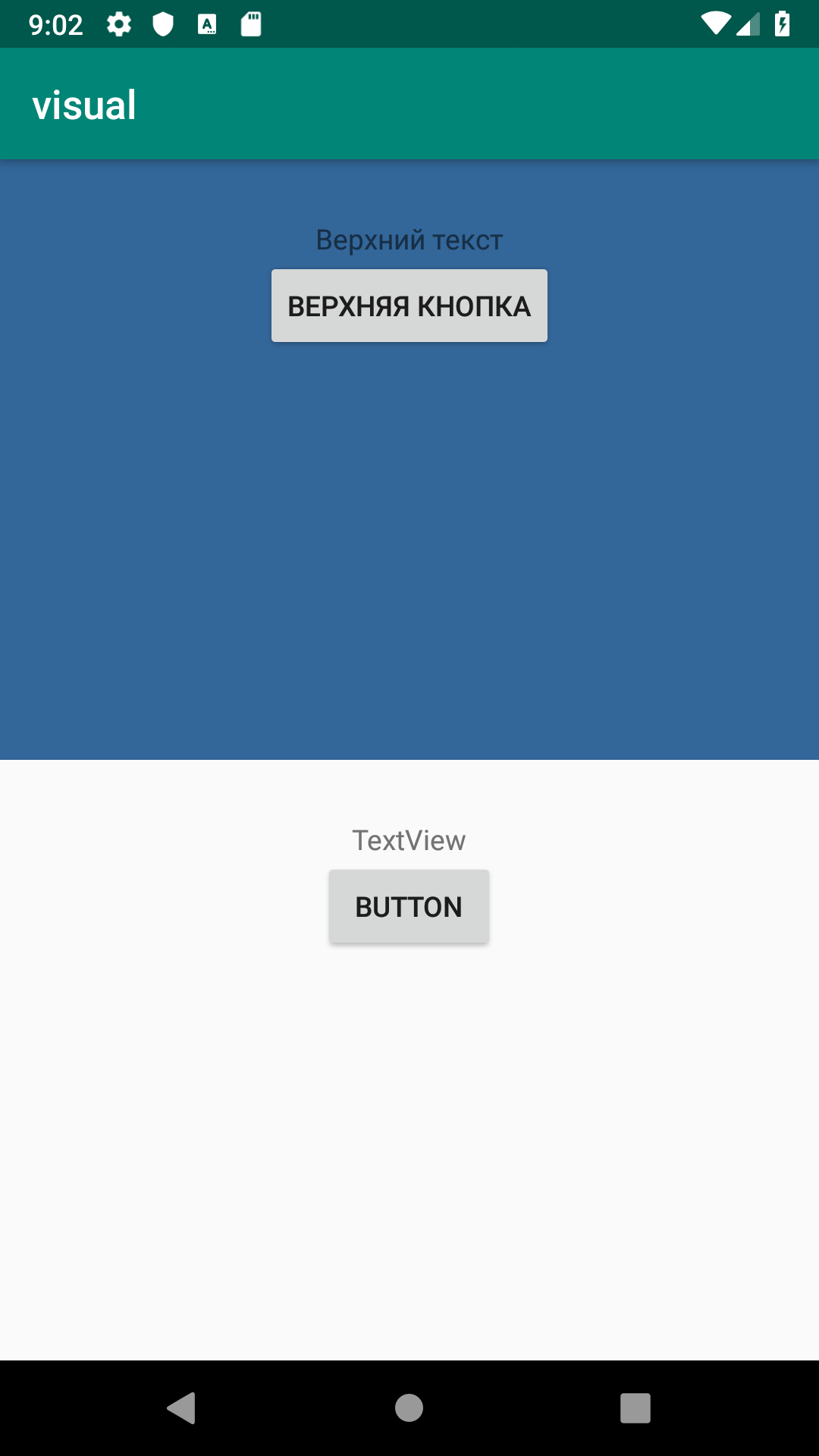
текст верхней надписи:

android:text="@string/tvTopText"></TextView>

и текст на верхней кнопке:

android:text="@string/btnTopText"></Button>

Посмотрим, что получилось:



Для оформления нижней части приложения потренируемся обращаться к ресурсам из Java-кода. Для этого как всегда сначала создадим необходимые переменные:

LinearLayout llBottom = (LinearLayout) findViewById(R.id.*llBottom*);

TextView tvBottom = (TextView) findViewById(R.id.*tvBottom*);

Button btnBottom = (Button) findViewById(R.id.*btnBottom*);

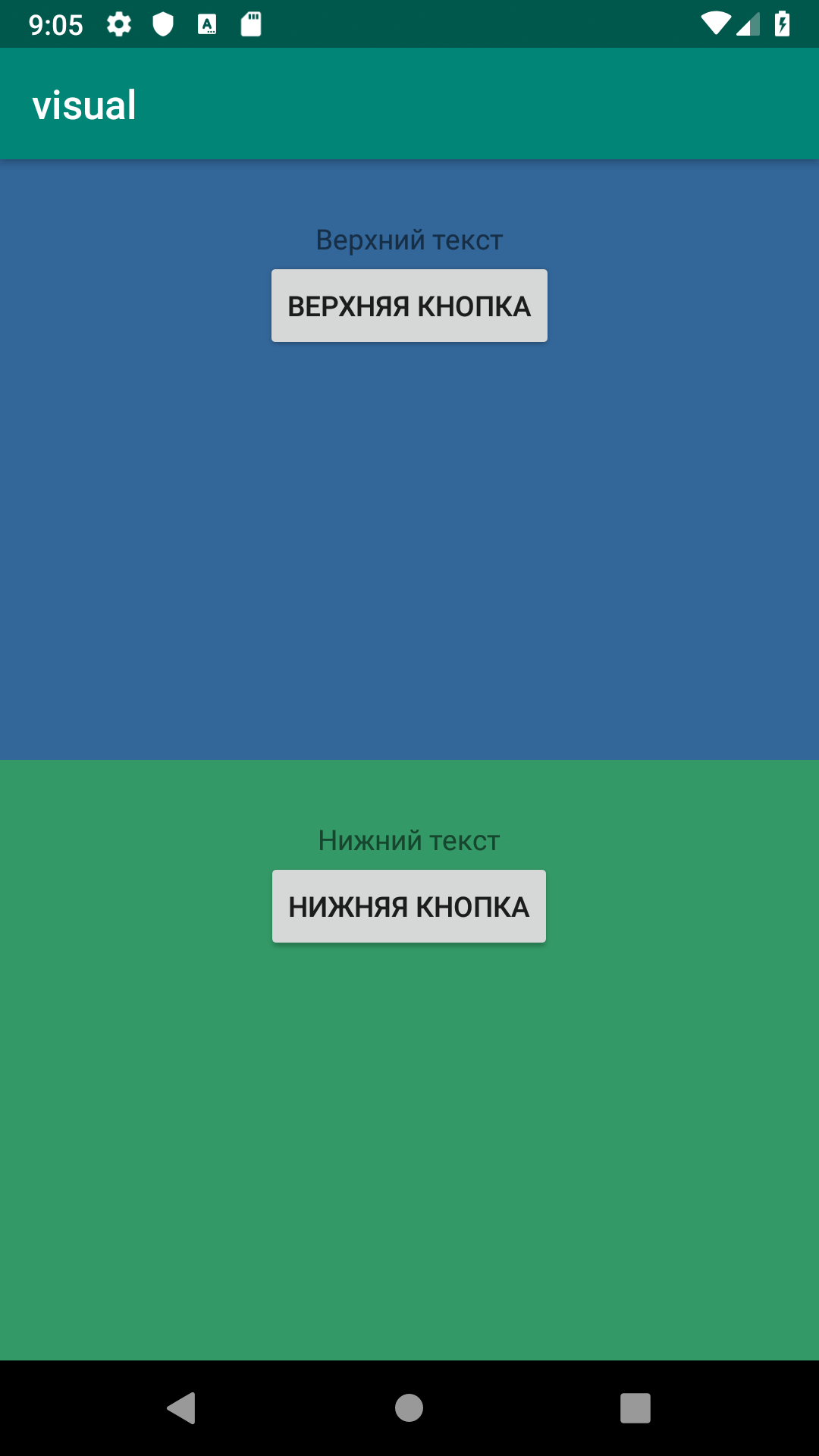
Теперь установим значения атрибутов оформления через соответствующие методы элемента (подробнее вы можете прочитать в документации к нужному элементу):

llBottom.setBackgroundResource(R.color.*llBottomColor*);

tvBottom.setText(R.string.*tvBottomText*);

btnBottom.setText(R.string.*btnBottomText*);

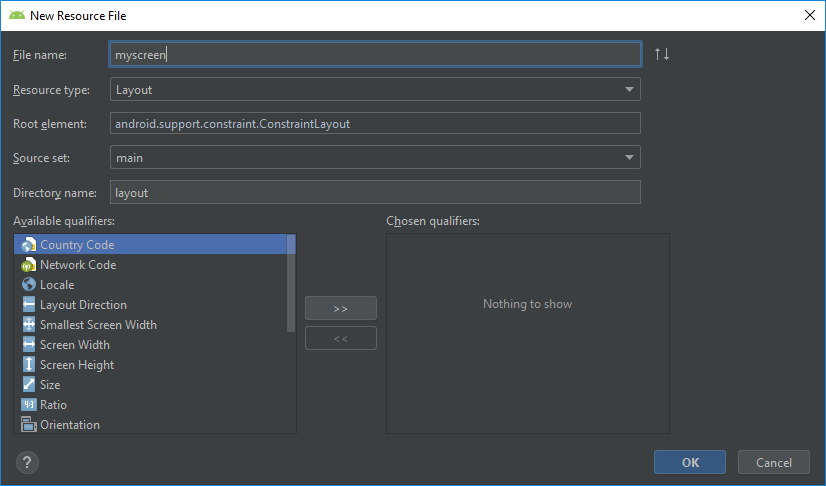
Опять, посмотрим, что у нас получилось:



#### Альтернативные ресурсы

Сейчас мы потренируемся создавать альтернативные ресурсы для разных условий работы нашего приложения. Начнем с самого значимого ресурса - расположений (layout). Нам придется создавать новые расположения и порой бывает нужно связать другое расположение с контекстом приложения.

Для начала работы создадим новый файл ресурса типа layout и назовем его myscreen:



Теперь перейдем в код главной активности и заменим строчку, привязывающую расположение к активности:

setContentView(R.layout.myscreen);

После этого, наше новое расположение будет использоваться для активности по умолчанию. Теперь мы можем заполнить наше приложение. Создадим простое расположение с текстом и несколькими кнопками, расположенными вертикально:

<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>

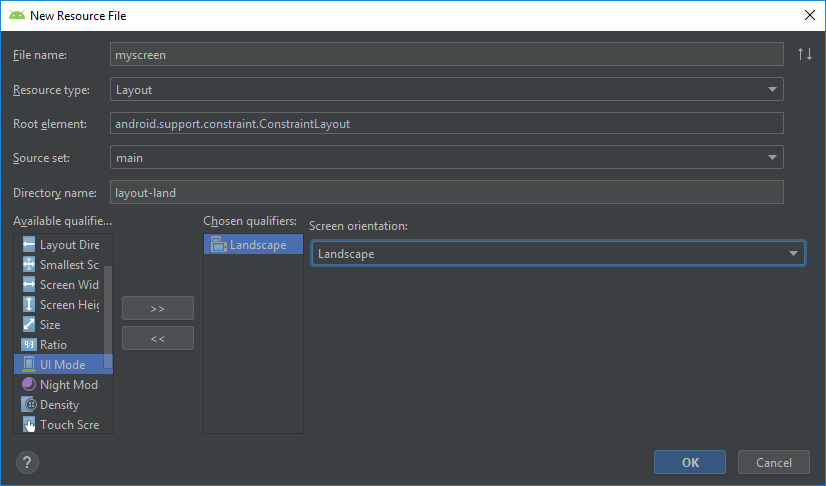
</LinearLayout>

Запустим приложение и увидим, что все в порядке. Но заодно давайте изменим ориентацию экрана на виртуальном устройстве и посмотрим, как наше приложение адаптировалось к изменениям:

|  |  |
| --- | --- |
|  |  |

Выбранное нами расположение элементов хорошо подходит к вертикальной ориентации экрана, но в горизонтальной выглядит очень некрасиво. Для того, чтобы приложение выглядело хорошо на всех экранах часто необходимо, чтобы оно использовало разные представления в разных ситуациях.

Поэтому сейчас мы создадим новое представление также, как и чуть ранее. Назовем его точно так же, как и наше основное, myscreen, это важно. нижней части экрана мы видим область выбора квалификаторов для разных ситуаций. Сейчас нам нужно выбрать orientation - landscape.



В созданном файле запишем очень похожее расположение элементов с той разницей, что используем горизонтальное выравнивание элементов:

<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="Горизонтальная ориентация экрана">

</TextView>

<LinearLayout

android:id="@+id/linearLayout1"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<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">

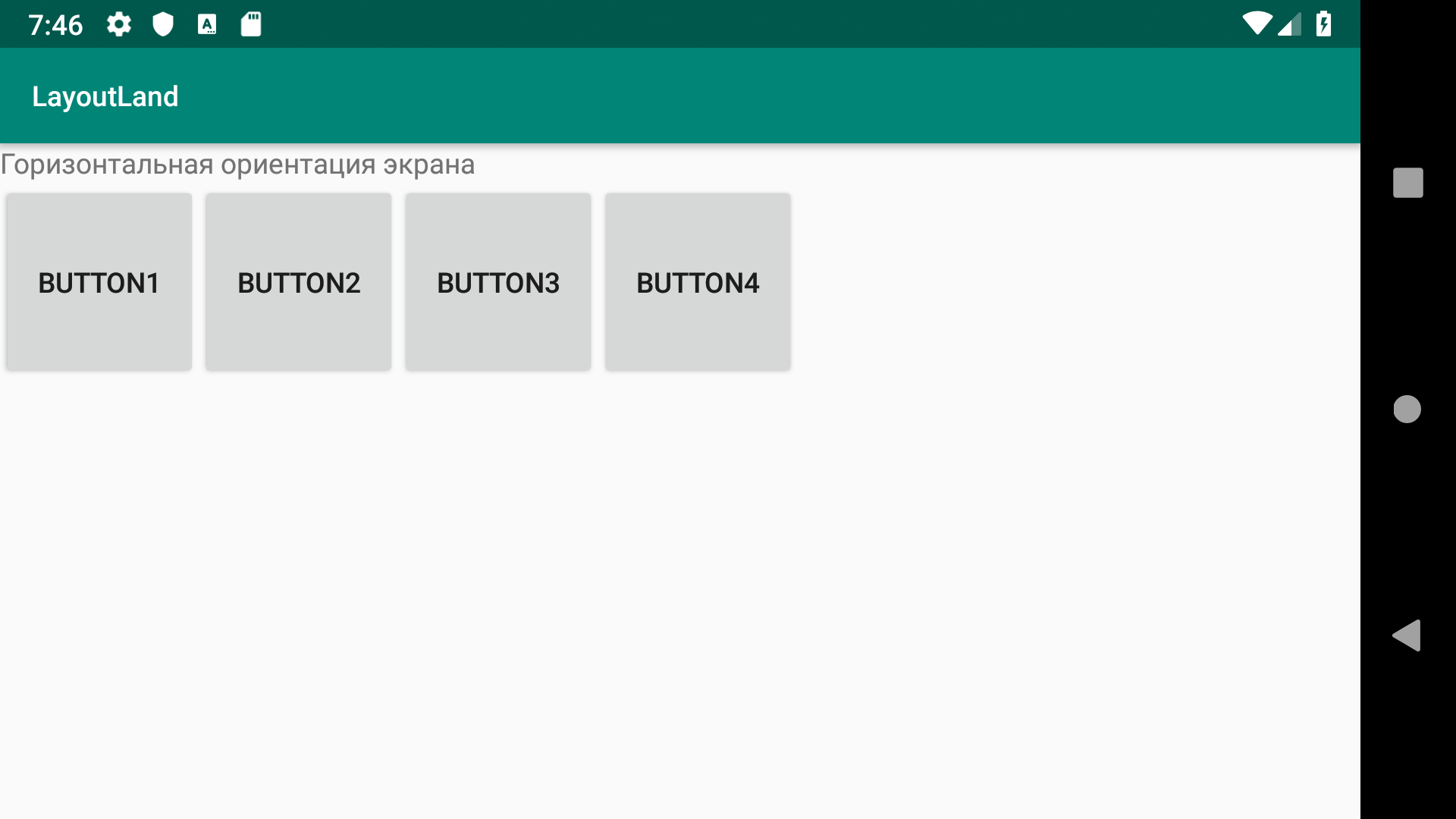
</Button>

</LinearLayout>

</LinearLayout>

Обратите внимание, на изменившуюся структуру проекта. Теперь наши два файла myscreen стали альтернативными ресурсами. Они используется по умолчанию, а второй - в горизонтальной ориентации экрана.

Запустим наше приложение и проверим горизонтальное расположение элементов:



По такому же принципу можно создавать ресурсы любого типа: строки, расположения, картинки и для разных ситуаций: разные локали, размеры экрана, версия андроида, наличие клавиатуры и так далее.

### Контрольные вопросы

1. С помощью какого класса предоставляется доступ к ресурсам из кода Java?

Все идентификаторы resource ID определены в классе R проекта, который автоматически генерирует утилита aapt.

1. Какие основные квалификаторы ресурсов вы знаете?

Orientation – ориентация экрана

Version – версия андроида

Size – размер приложения

Locale – язык и регион

Если папка без квалификатора, то ресурсы в ней используются по умолчанию.

1. Какие основные XML-атрибуты используются для задания расположения виджета на экране?

Android:layout\_gravity – выравнивание элемента по левому верхнему краю/левому нижнему краю/правому нижнему краю/правому/верхнему краю/центру

AndroidLlayout\_margin – отступы от элемента со всех сторон

AndroidLlayout\_marginBottom – отступ от низа

AndroidLlayout\_marginEnd – отступ от конца

AndroidLlayout\_marginLeft – отступ слева

AndroidLlayout\_marginRight – отступ справа

AndroidLlayout\_marginStart – отступ от начала

AndroidLlayout\_marginTop – отступ сверху

Для RelativeLayout:

Android:layout\_above – сверху от заданного элемента

Android:layout\_below – снизу от заданного элемента

Android:layout\_toRightOf – правее заданного элемента

Android:layout\_toLeftOf – левее заданного элемента

Android:layout\_alignLeft – выровнять по левому краю заданного элемента

Android:layout\_alignRight – выровнять по правому краю заданного элемента

Android:layout\_alignTop – выровнять по верхнему краю заданного элемента

Android:layout\_alignBottom – выровнять по нижнему краю заданногo элемента

Android:layout\_alignParentLeft – выровнять по левому краю родителя

Android:layout\_alignParentRight – выровнять по правому краю родителя

Android:layout\_alignParentTop – выровнять по верхнему краю родителя

Android:layout\_alignParentBottom – выровнять по нижнему краю родителя

Android:layout\_centerVerical – выровнять по вертикали

Android:layout\_cenrerHorizontal – выровнять по горизонтали

Android:layout\_centerParent – выровнять по родителю

Для AbsoluteLayout:

Android:layout\_x – задаем x координату левого верхнего угла

Android:layout\_y – задаем y координату левого верхнего угла

1. Какие основные XML-атрибуты используются для задания отображения виджета на экране?

Android:layout\_width – задание ширины элемента

Android:layout\_height – задание высоты элемента

Android:layout\_weight – задание веса элемента

### Дополнительные задания

1. Изучите все виды квалификаторов в окне создания нового ресурса. Изучите документацию и узнайте, для чего они служат.

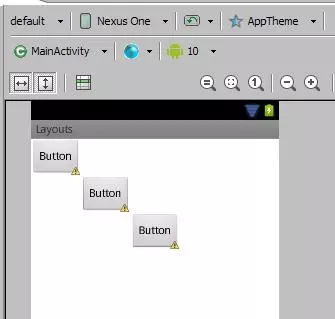
Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

1. Изучите основные виды расположений: LinearLayout, RelativeLayout, AbsoluteLayout, TableLayout. C помощью каждого из них постройте расположение элементов, как показано на рисунке:

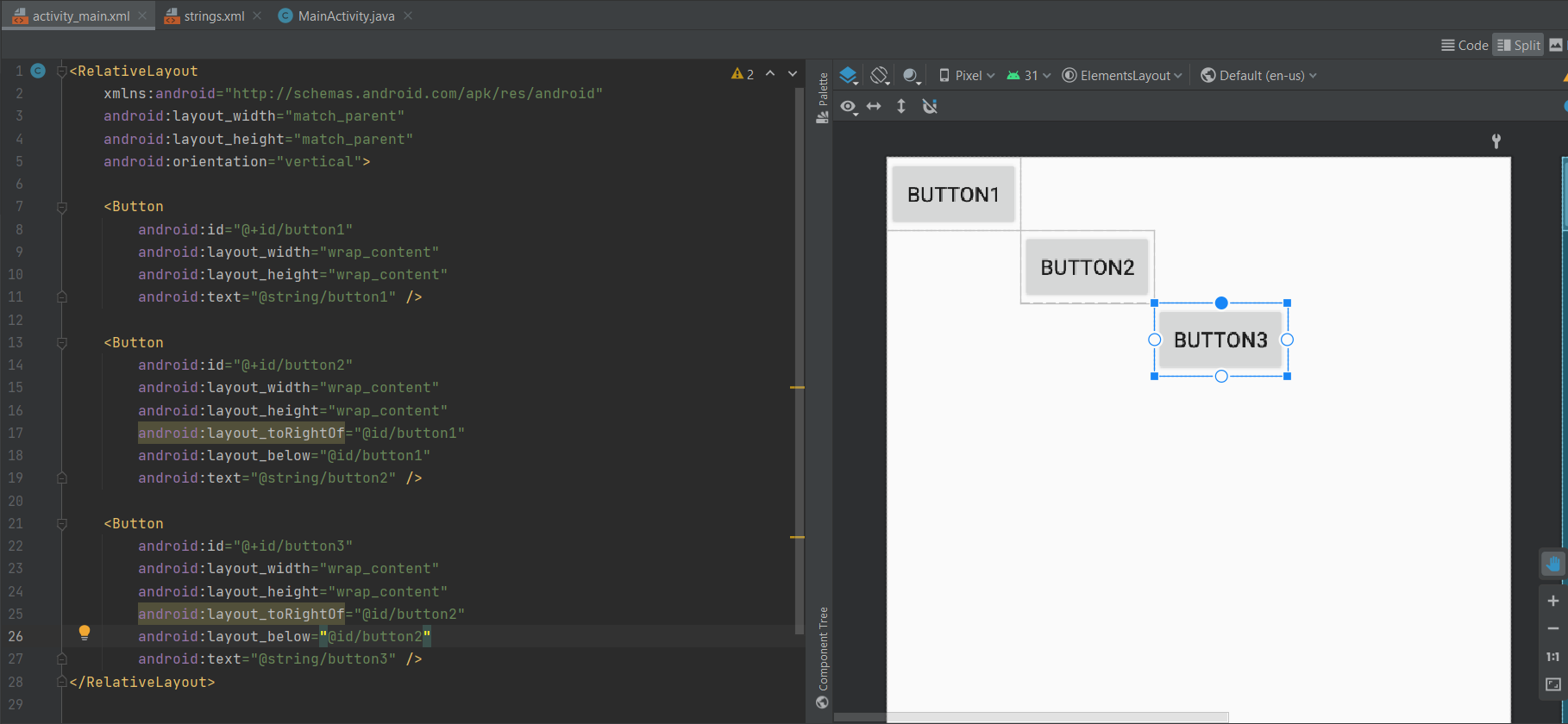


LinearLayout:

Изображение выглядит как текст

Автоматически созданное описание

RelativeLayout:



AbsoluteLayout:

Изображение выглядит как текст

Автоматически созданное описание

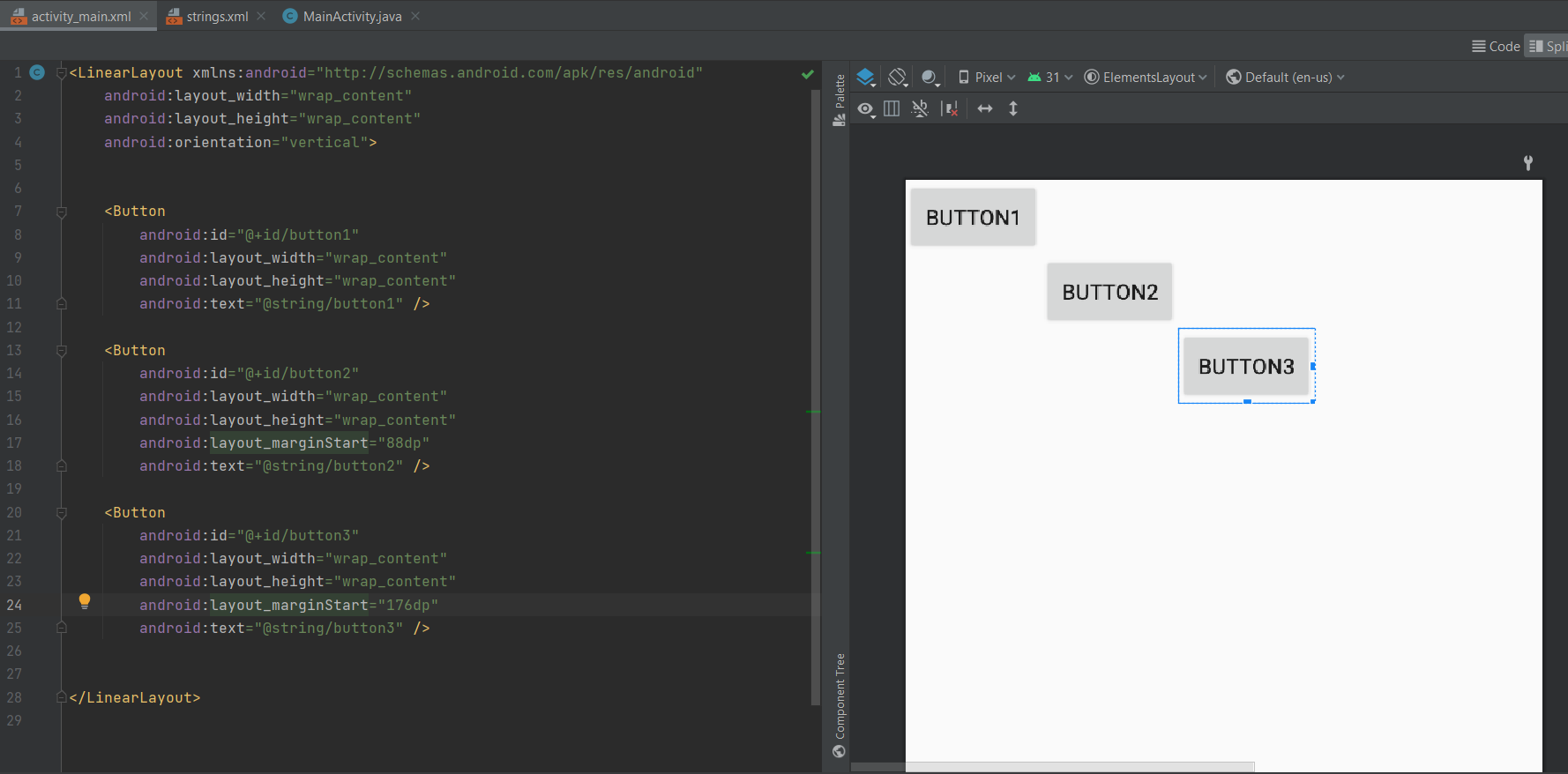
TableLayout:

Изображение выглядит как текст

Автоматически созданное описание

1. Проведите рефакторинг написанного приложения так, чтобы студия не показывала ни одного предупреждения.

LinearLayout:



RelativeLayout:

Изображение выглядит как текст, снимок экрана, монитор

Автоматически созданное описание

AbsoluteLayout:

Изображение выглядит как текст

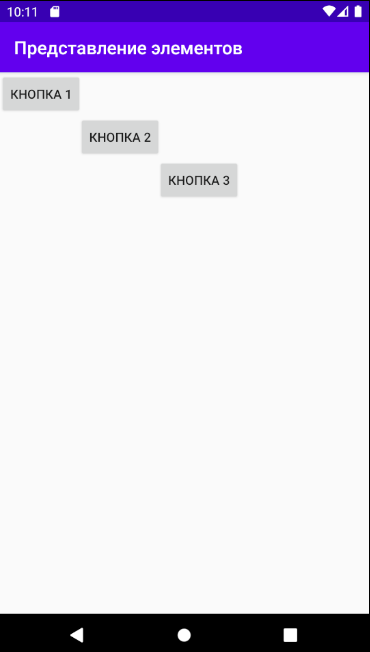
Автоматически созданное описание

TableLayout:

Изображение выглядит как текст

Автоматически созданное описание

1. (\*) Переведите созданные на предыдущих занятиях приложения на английский язык (на русский, если были на английском).



Изображение выглядит как текст

Автоматически созданное описание