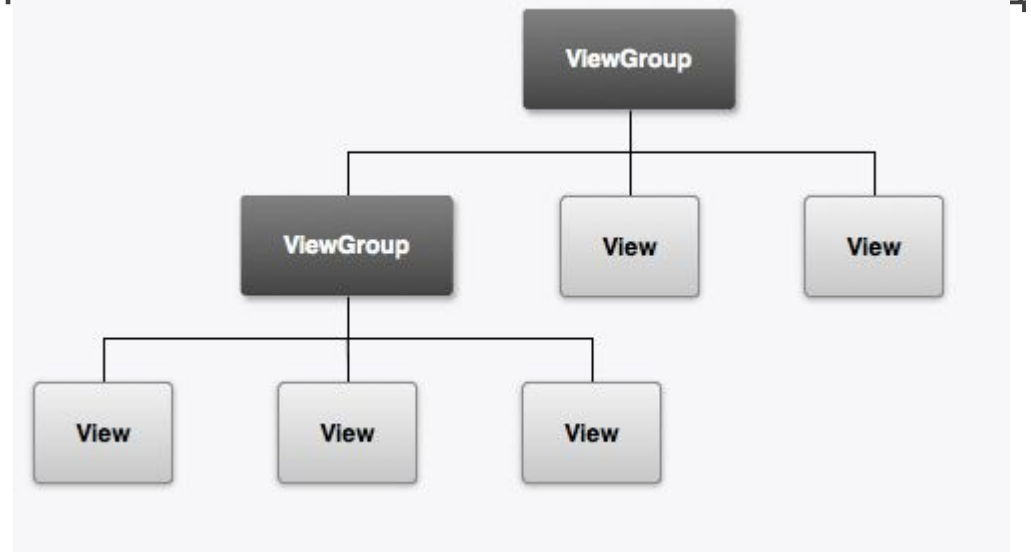
Основы создания интерфейса

ЛЕКЦИЯ

Графический интерфейс пользователя представляет собой иерархию объектов android.view.View и android.view.ViewGroup. Каждый объект ViewGroup представляет контейнер, который содержит и упорядочивает дочерние объекты View. В частности, к контейнерам относят такие элементы, как RelativeLayout, LinearLayout, GridLayout, ConstraintLayout и ряд других.

Простые объекты View представляют собой элементы управления и прочие виджеты, например, кнопки, текстовые поля и т[illegible]действует с программой:



Иерархия визуальных компонентов

Большинство визуальных элементов, наследующихся от класса View, такие как кнопки, текстовые поля и другие, располагаются в пакете android.widget

Стратегии определения интерфейса

Разметка определяет визуальную структуру пользовательского интерфейса. Установить разметку можно двумя способами:

1. Создать элементы управления программно в коде java (пример)
2. Объявить элементы интерфейса в XML*
3. Сочетание обоих способов - базовые элементы разметки определить в XML, а остальные добавлять во время выполнения*

Создание интерфейса в коде java

Определим в классе **MainActivity** простейший интерфейс:

```
package com.example.eugene.viewsapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends
AppCompatActivity {

    @Override
    protected void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        // создание TextView
        TextView textView = new TextView(this);
        // установка текста в TextView
        textView.setText("Hello Android!");
        // установка высоты текста
        textView.setTextSize(22);
        // установка визуального интерфейса для
activity
        setContentView(textView);
    }
}
```

Здесь весь интерфейс представлен элементом TextView, которое предназначено для вывода текста. С помощью методов, которые, как правило, начинаются на set, можно установить различные свойства TextView. Например, в данном случае метод setText() устанавливает текст в поле, а setTextSize() задает высоту шрифта.

Для установки элемента в качестве интерфейса приложения в коде Activity вызывается метод setContentView(), в который передается визуальный элемент.

Определение размеров

В ОС Android мы можем использовать различные типы измерений:

---

1. px: пиксели текущего экрана. Однако эта единица измерения не рекомендуется, так как реальное представление внешнего вида может изменяться в зависимости от устройства; каждое устройство имеет определенный набор пикселей на дюйм, поэтому количество пикселей на экране может также меняться
2. dp: (device-independent pixels) независимые от плотности экрана пиксели. Абстрактная единица измерения, основанная на физической плотности экрана с разрешением 160 dpi (точек на дюйм). В этом случае 1dp = 1px. Если размер экрана больше или меньше, чем 160dpi, количество пикселей, которые применяются для отрисовки 1dp соответственно увеличивается или уменьшается. Например, на экране с 240 dpi 1dp=1,5px, а на экране с 320dpi 1dp=2px. Общая формула для получения количества физических пикселей из dp: px = dp * (dpi / 160)
3. sp: (scale-independent pixels) независимые от масштабирования пиксели. Допускают настройку размеров, производимую пользователем. Рекомендуются для работы со шрифтами.
4. pt: 1/72 дюйма, базируются на физических размерах экрана
5. mm: миллиметры
6. in: дюймы

***Предпочтительными единицами для использования являются dp.

Ширина и высота элементов

Все визуальные элеметы, которые мы используем в приложении, как правило, упорядочиваются на экране с помощью контейнеров. В Android подобными контейнерами служат такие классы как RelativeLayout, LinearLayout, GridLayout, TableLayout, ConstraintLayout, FrameLayout. Все они по разному располагают элементы и управляют ими, но есть некоторые общие моменты при компоновке визуальных компонентов, которые мы сейчас рассмотрим.

Для организации элементов внутри контейнера используются параметры разметки. Для их задания в файле xml используются атрибуты, которые начинаются с префикса layout_. В частности, к таким параметрам относятся атрибуты layout_height и layout_width, которые используются для установки размеров и могут принимать одно из следующих значений:

1. точные размеры элемента, например 96 dp
2. значение **wrap_content**: элемент растягивается до тех границ, которые достаточны, чтобы вместить все его содержимое
3. значение **match_parent**: элемент заполняет всю область родительского контейнера

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/activity_main"

   android:layout_width="match_parent"
   android:layout_height="match_parent"
   >

   <TextView
      android:text="Hello Android 7"
      android:layout_height="wrap_content"
      android:layout_width="wrap_content"
      android:textSize="26sp"
      android:background="#e0e0e0" />
</RelativeLayout>
```

Контейнер самого верхнего уровня, в качестве которого в данном случае выступает RelativeLayout, для высоты и ширины имеет значение **match_parent**, то есть он будет заполнять всю область для activity - как правило, весь экран.
А элемент TextView растягивается до тех значений, которые достаточны для рамещения его текста.

Внутренние и внешние отступы

Параметры разметки позволяют задать отступы как от внешних границ элемента до границ контейнера, так и внутри самого элемента между его границами и содержимым.

Padding

Для установки внутренних отступов применяется атрибут android:padding. Он устанавливает отступы контента от всех четырех сторон контейнера. Можно устанавливать отступы только от одной стороны контейнера, применяя следующие атрибуты: android:paddingLeft,

android:paddingRight,

android:paddingTop

и android:paddingBottom.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/activity_main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="50dp">

   <TextView android:text="Hello Android 7"
      android:layout_height="match_parent"
      android:layout_width="match_parent"
      android:textSize="26sp"
      android:background="#e0e0e0"
      android:paddingTop="60dp"
      android:paddingLeft="40dp"/>

</RelativeLayout>
```

Margin

Для установки внешних остступов используется атрибут layout_margin. Данный атрибут имеет модификации, которые позволяют задать отступ только от одной стороны: android:layout_marginBottom, android:layout_marginTop, android:layout_marginLeft и android:layout_marginRight (отступы соответственно от нижней, верхней, левой и правой границ)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/activity_main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"

   android:paddingTop="50dp">

   <TextView android:text="Hello Android 7"
      android:layout_height="match_parent"
      android:layout_width="match_parent"

      android:layout_marginTop="50dp"
      android:layout_marginBottom="60dp"
      android:layout_marginLeft="60dp"
      android:layout_marginRight="60dp"

      android:textSize="26sp"
      android:background="#e0e0e0"/>

</RelativeLayout>
```

Программная установка отступов

Для программной установки внутренних отступов у элементы вызыватся метод **setPadding(left, top, right, bottom)**, в который передаются четыре значения для каждой из сторон.

Для установки внешних отступов необходимо реализовать объект **LayoutParams** для того контейнера, который применяется. И затем вызвать у этого объекта LayoutParams метод **setMargins(left, top, right, bottom)**:

```
RelativeLayout.LayoutParams layoutParams = new RelativeLayout.LayoutParams
        (ViewGroup.LayoutParams.MATCH_PARENT, 200);
// установка внешних отступов
layoutParams.setMargins(30,40,50,60);
// устанавливаем размеры
textView1.setLayoutParams(layoutParams);
// установка внутренних отступов
textView1.setPadding(30,30,30,30);
// добавляем TextView в RelativeLayout
relativeLayout.addView(textView1);
setContentView(relativeLayout);
```

LinearLayout

Контейнер LinearLayout представляет объект ViewGroup, который упорядочивает все дочерние элементы в одном направлении: по горизонтали или по вертикали. Все элемены расположены один за другим. Направление разметки указывается с помощью атрибута android:orientation.

Если, например, ориентация разметки вертикальная (android:orientation="vertical"), то все элементы располагаются в столбик - по одному элементу на каждой строке. Если ориентация горизонтальная (android:orientation="horizontal"), то элементы располагаются в одну строку. Например, расположим элементы в горизонтальный ряд

```
<?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="horizontal" >

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello"
      android:textSize="26sp" />
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Android"
      android:textSize="26sp" />
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Nougat"
      android:textSize="26sp" />
```

RelativeLayout RelativeLayout представляет объект ViewGroup, который располагает дочерние элементы **относительно позиции других дочерних элементов разметки или относительно области самой разметки RelativeLayout. Используя относительное позиционирование, мы можем установить элемент по правому краю или в центре или иным способом, который предоставляет данный контейнер.** Для установки элемента в файле xml мы можем применять следующие атрибуты:

---

android:layout_above: располагает элемент над элементом с указанным Id

android:layout_below: располагает элемент под элементом с указанным Id

android:layout_toLeftOf: располагается слева от элемента с указанным Id

android:layout_toRightOf: располагается справа от элемента с указанным Id

android:layout_alignBottom: выравнивает элемент по нижней границе другого элемента с указанным Id

android:layout_alignLeft: выравнивает элемент по левой границе другого элемента с указанным Id

android:layout_alignRight: выравнивает элемент по правой границе другого элемента с указанным Id

Прокомментируйте код

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView android:text="Left Top"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:textSize="26sp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" />

    <TextView android:text="Right Top"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:textSize="26sp"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true" />

    <TextView android:text="Left Bottom"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:textSize="26sp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true" />

    <TextView android:text="Right Bottom"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:textSize="26sp"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true" />
</RelativeLayout>
```

Gravity и layout_gravity

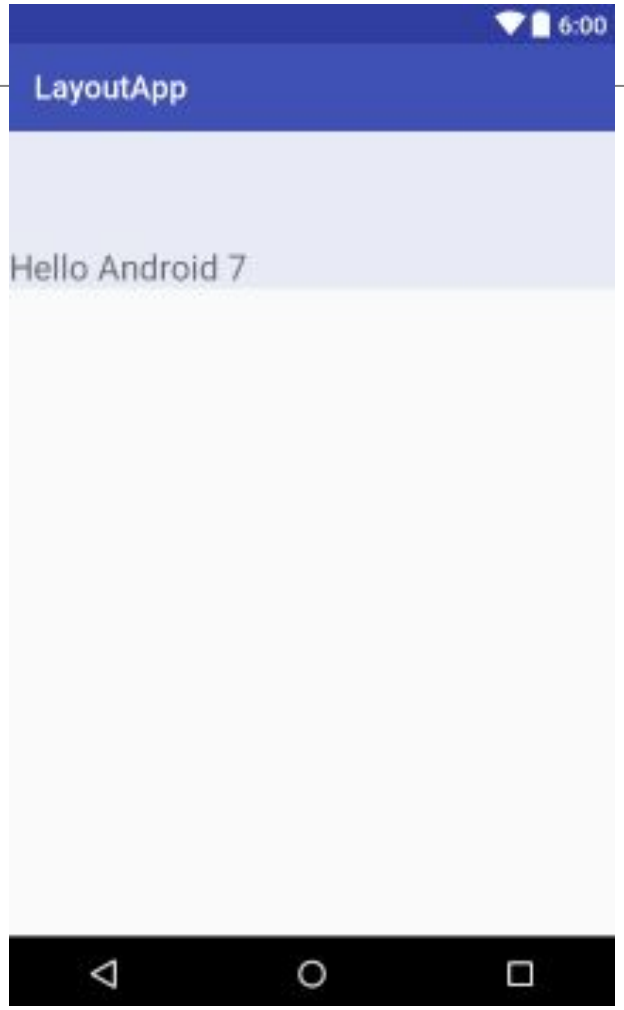
Атрибут gravity задает позиционирование содержимого внутри объекта. Он может принимать следующие значения:

top: элементы размещаются вверху
bottom: элементы размещаются внизу
left: элементы размещаются в левой стороне
right: элементы размещаются в правой стороне контейнера
center_vertical: выравнивает элементы по центру по вертикали
center_horizontal: выравнивает элементы по центру по горизонтали
center: элементы размещаются по центру
fill_vertical: элемент растягивается по вертикали
fill_horizontal: элемент растягивается по горизонтали
fill: элемент заполняет все пространство контейнера
clip_vertical: обрезает верхнюю и нижнюю границу элементов
clip_horizontal: обрезает правую и левую границу элементов
start: элемент позиционируется в начале (в верхнем левом углу) контейнера
end: элемент позиционируется в конце контейнера(в верхнем правом углу)

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

      <TextView
         android:gravity="bottom"

         android:layout_width="match_parent"
         android:layout_height="200px"
         android:text="Hello Android 7"
         android:background="#e8eaf6"/>
</RelativeLayout>
```



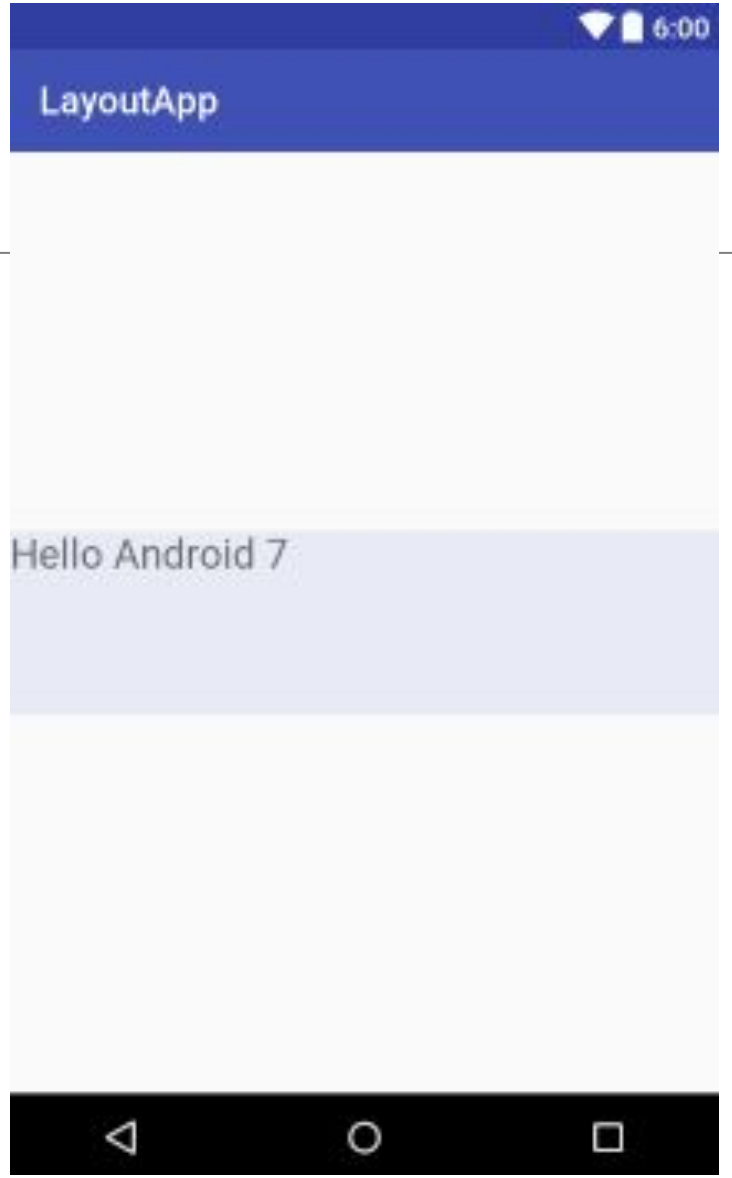
Layout_gravity

В отличие от gravity атрибут layout_gravity устанавливает позиционирование в контейнере. Он принимает те же значения, только позиционирование идет относительно внешнего контейнера:

top: выравнивает элемент по верхней границе контейнера
bottom: выравнивает элемент по нижней границе контейнера
left: выравнивает элемент по левой границе контейнера
right: выравнивает элемент по правой границе контейнера
center_vertical: выравнивает элемент по центру по вертикали
center_horizontal: выравнивает элемент по центру по горизонтали
center: элемент позиционируется в центре
fill_vertical: элемент растягивается по вертикали
fill_horizontal: элемент растягивается по горизонтали
fill: элемент заполняет все пространство контейнера
clip_vertical: обрезает верхнюю и нижнюю границу элемента
clip_horizontal: обрезает правую и левую границу элемента
start: элемент позиционируется в начале (в верхнем левом углу) контейнера
end: элемент позиционируется в конце контейнера (в верхнем правом углу)

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="horizontal">

      <TextView
         android:layout_gravity="center"
         android:layout_width="match_parent"
         android:layout_height="200px"
         android:textSize="22sp"
         android:text="Hello Android 7"
         android:background="#e8eaf6"/>
</LinearLayout>
```



***При этом надо учитывать, что layout_gravity применяется только к классу LinearLayout или к его классам-наследникам, например, FrameLayout. В RelativeLayout этот атрибут не окажет никакого влияния.

Также внутри LinearLayout стоит учитывать ориентацию контейнера. Например, при вертикальной ориентации все элементы будут представлять вертикальный стек, идущий сверху вниз. Поэтому значения, которые относятся к позиционированию элемента по вертикали (например, top или bottom) никак не будут влиять на элемент. Также при горизонтальнйо ориентации LinearLayout не окажут никакого влияния значения, которые позиционируют элемент по горизонтали, например, left и right.

TableLayout

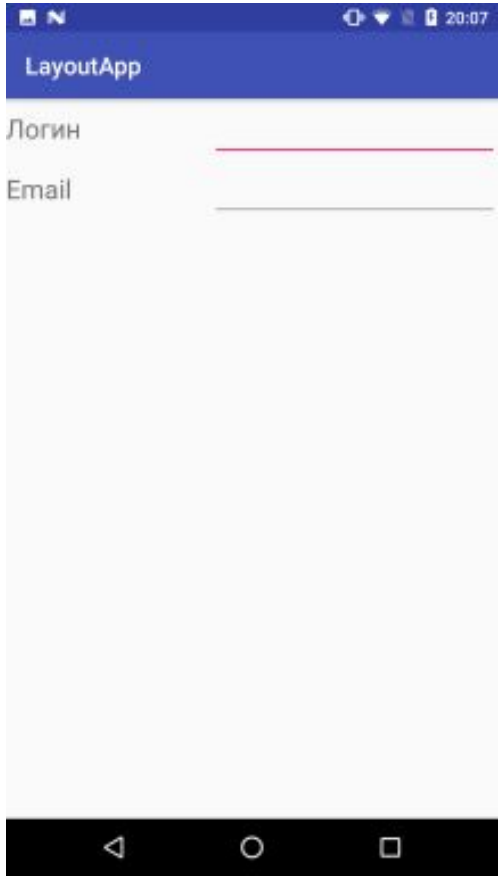
Контейнер **TableLayout** структурирует элементы управления по столбцам и строкам. Определим в файле **activity_main.xml** элемент TableLayout, который будет включать две строки и два столбца:

```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TableRow>
        <TextView
            android:layout_weight="0.5"
            android:text="Логин"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <EditText
            android:layout_weight="1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"  />
    </TableRow>

    <TableRow>
        <TextView
            android:layout_weight="0.5"
            android:text="Email"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <EditText
            android:layout_weight="1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </TableRow>
</TableLayout>
```



***FrameLayout

Контейнер FrameLayout предназначен для вывода на экран одного помещенного в него визуального элемента. Если же мы поместим несколько элементов, то они будут накладываться друг на друга.

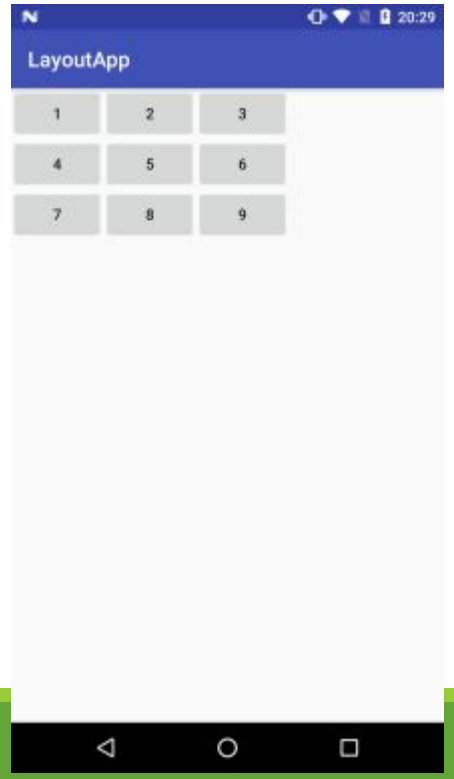
***GridLayout представляет еще один контейнер, который позволяет создавать табличные представления. GridLayout состоит из коллекции строк, каждая из которых состоит из отдельных ячеек

```
<GridLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="3"
    android:columnCount="3">

    <Button android:text="1" />
    <Button android:text="2" />
    <Button android:text="3" />
    <Button android:text="4" />
    <Button android:text="5" />
    <Button android:text="6" />
    <Button android:text="7" />

    <Button android:text="8" />

    <Button android:text="9" />
</GridLayout>
```



- **android:layout_column**: номер столбца (отсчет идет от нуля)
- **android:layout_row**: номер строки
- **android:layout_columnSpan**: количество столбцов, на которые растягивается элемент
- **android:layout_rowSpan**: количество строк, на которые растягивается элемент

Задание

1. Изучить самостоятельно информацию про ConstraintLayout, ScrollView, Вложенные layout.

2. Выполнить примеры по образцу из главы «Основы создания интерфейса»

3. Сделать заготовку интерфейса калькулятора. Создать кнопки, выполнить выравнивание кнопок.

4. Сделать общий отчёт о выполненной работе. В отчёт поместить скрины интерфейсов приложений и скрины кода.