

# Layouts

Frame  
Layout

- This is designed to block out an area on the screen to display a single item.

Linear  
Layout

- A Layout that arranges its children in a single column or a single row.

Relative  
Layout

- This Layout is a view group that displays child views in relative positions.

Table  
Layout

- A layout that arranges its children into rows and columns

# android

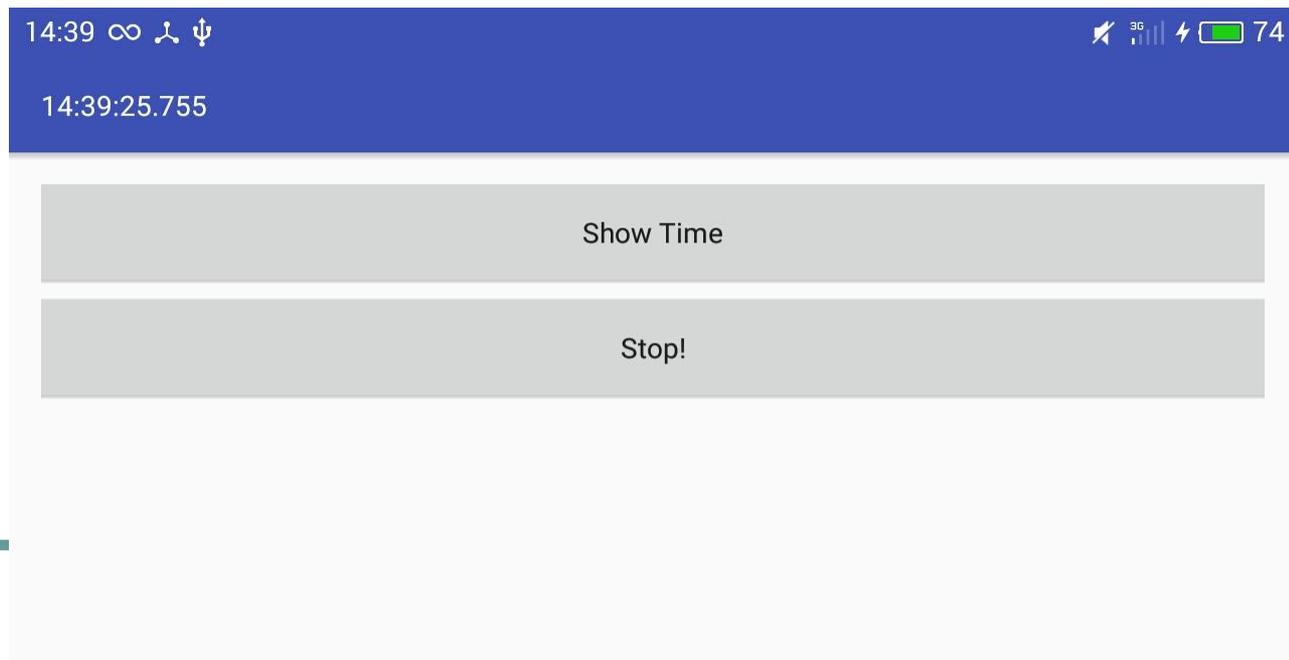


# План презентации

- Класс Handler
- Практика на работу с таймерами
- Основные типы макетов Android
- Relative Layout
- Linear Layout
- Grid Layout
- Frame Layout
- Обзор Constraint Layout
- Практика

# Который час?

- <https://git.io/viTcl> (XML)
- <https://git.io/viTc3> (Java)



# Handler

Класс **Handler** предоставляет удобный способ работы с дополнительными потоками, не нарушая работу основного UI-потока. Отлично подходит для неоднократного выполнения определённого набора действий с заданным интервалом, либо для отложенного выполнения кода через некоторое время.

# Что почитать про Handler

- <https://developer.android.com/reference/android/os/Handler.html>
- <http://findevelop.blogspot.com/2014/01/handler-android.html>
- <http://startandroid.ru/ru/uroki/vse-uroki-spiskom/143-urok-80-handler-nemnogo-teorii-nagljadnyj-primjer-ispolzovanija.html> (урок 80-84)
- Альтернативы: классы `Timer`, `AsyncTask`  
<https://habrahabr.ru/post/136942/>

# Методы Handler

- **post** – добавляет в очередь на исполнение код объекта, реализующего интерфейс **Runnable**. Код, прописанный в методе **run**, практически без промедления начинает исполняться в параллельном потоке
- **postDelayed** – код метода **run** запускается не сразу, а через указанное количество миллисекунд
- **postAtTime** – код метода **run** запускается в указанное время
- **removeCallbacks(this)** – остановка таймера

# Практика на таймеры

Задание №1:

Кнопка начинает плавно  
двигаться вправо.

# Задание №2

- Приложение показывает в заголовке, сколько миллисекунд прошло после его запуска

# Задание №3

- Приложение показывает, сколько времени осталось до НГ/ДР/окончания курса Android (обновление каждую секунду)

# Задание №4

- Фонарик начинает моргать (полсекунды работает – полсекунды выключен). Если нет фонарика – пусть моргает кнопка.

# Домашнее задание

- Реализовать эффект бегущей строки в заголовке (как в новостях)
- Цвет фона приложения плавно меняется от чёрного к красному, от красного к жёлтому, от жёлтого – к зелёному
- Пользователю даётся 20 секунд, чтобы совершить максимально возможное количество кликов по кнопке. По истечении времени показать тост, который сообщает набранное количество кликов, и максимальный рекорд по итогам всех попыток
- Пользователь пишет сообщение в текстовом поле, и нажимает на кнопку. Фонарик начинает азбукой Морзе транслировать это сообщение

# Зачем нужны макеты

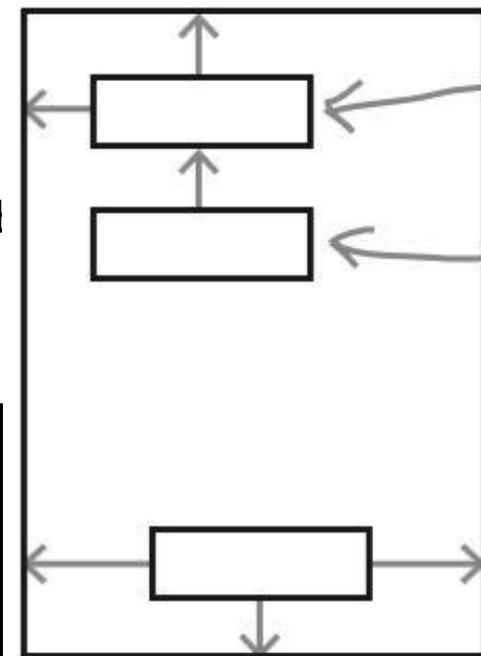
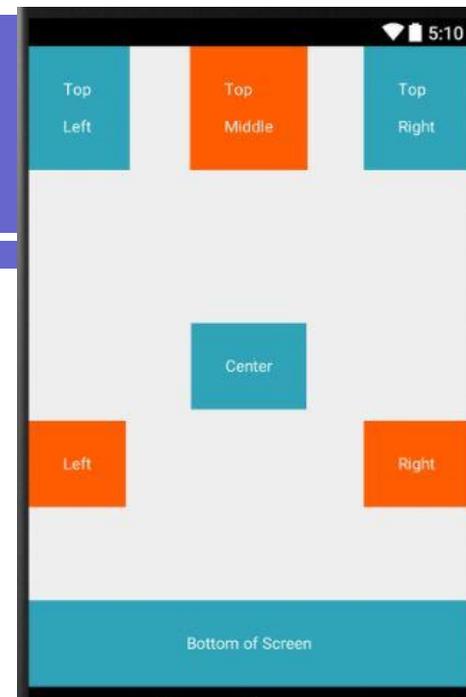
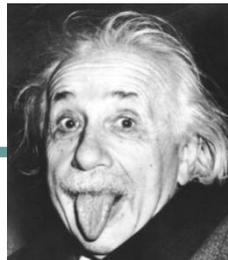
Макет определяет внешний вид экрана, и для описания макета обычно используется формат разметки XML. Макеты содержат компоненты графического интерфейса - кнопки, текстовые поля и тд. Пользователь взаимодействует с этими компонентами, чтобы приложение выполняло нужные операции. До этих пор в примерах применялся только **Relative Layout**, но существуют и другие типы макетов.

# Типы макетов Android

- **Relative Layout**
- **Linear Layout**
- **Grid Layout**
- Frame Layout
- Table Layout
- **Absolute Layout (deprecated!)**

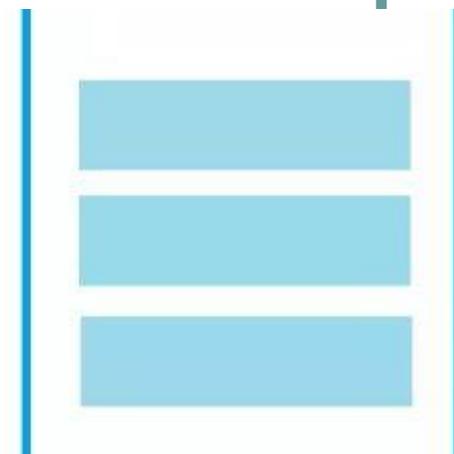
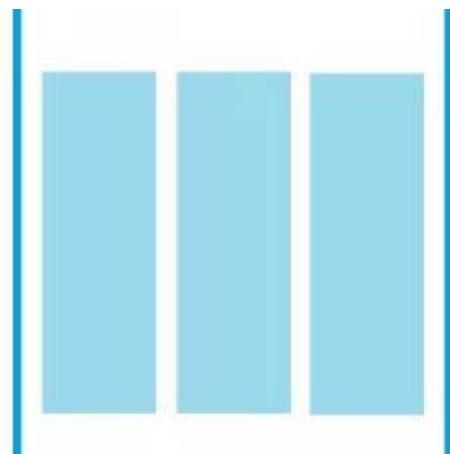
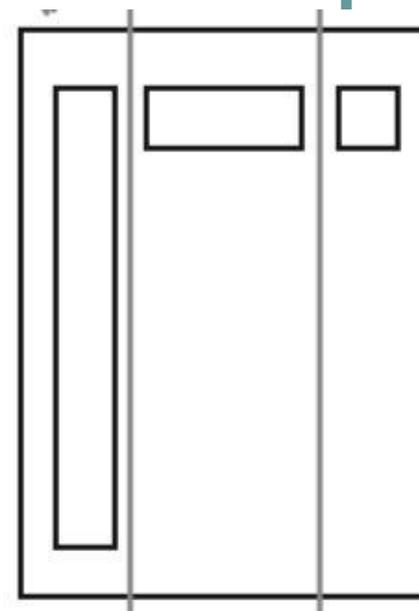
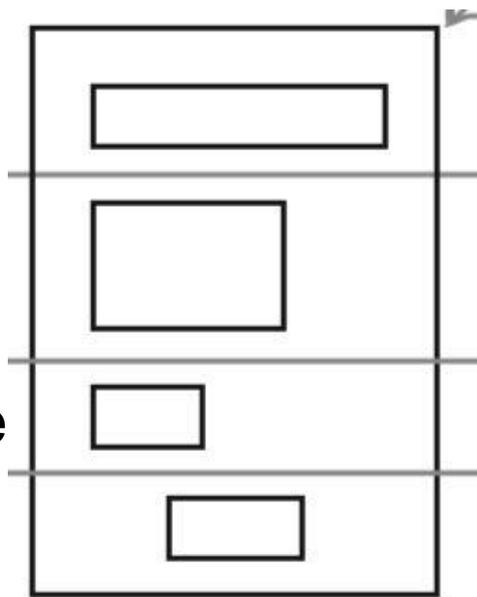
# Relative Layout

В относительном макете входящие в него вьюшки размещаются в относительных позициях. Позиция каждой вьюшки определяется **относительно других вьюшек** в макете или **относительно родительского макета**. Например, текстовое поле можно разместить относительно верхнего края родительского макета, раскрывающийся список разместит под этим текстовым полем, а кнопку - относительно нижнего края родительского макета.



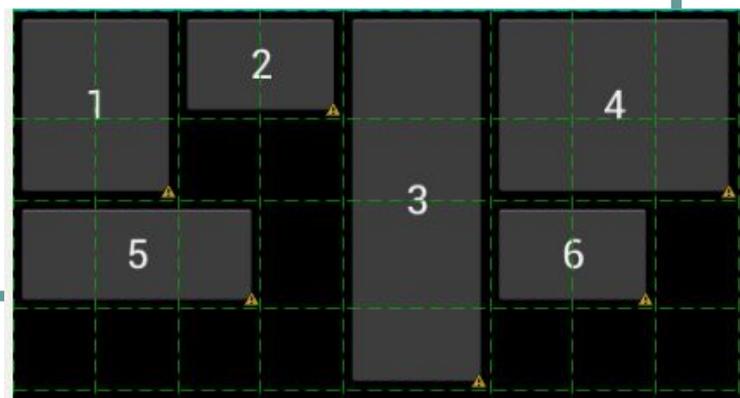
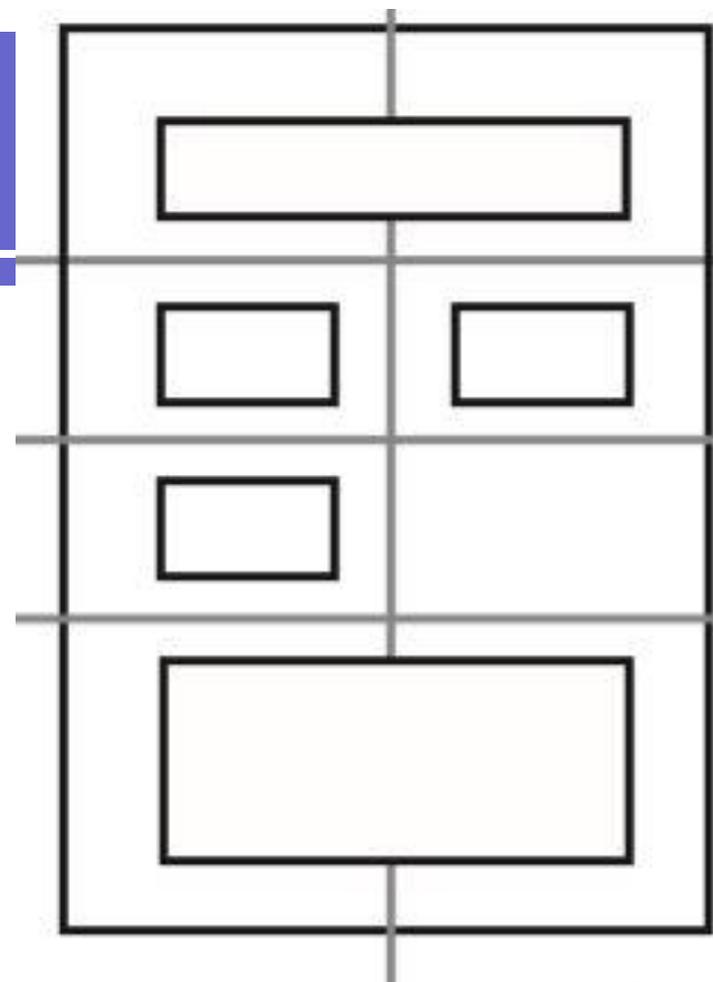
# Linear Layout

В линейном макете виджеты **размещаются рядом друг с другом по вертикали или по горизонтали**. Если используется вертикальное размещение, виджеты отображаются в один столбец. В варианте с горизонтальным размещением виджеты выводятся в одну строку.



# Grid Layout

В табличном макете **экран делится на строки и столбцы**, на пересечении которых находятся ячейки. Можно указать, сколько столбцов должно входить в макет, где должны отображаться вьюшки, и сколько строк или столбцов они должны занимать.



# Relative Layout

Относительный макет определяется элементом XML **<RelativeLayout>**.

Обязательно будет необходимо указать ширину и высоту макета атрибутами **layout\_width** и **layout\_height**!

Значениями этих атрибутов могут быть **match\_parent**, **wrap\_content** или конкретные размеры, например **10dp**.

# wrap\_content и match\_parent

Значение "**wrap\_content**" означает, что размеры макета должны быть минимально достаточными для того, чтобы разместить все вьюшки, а "**match\_parent**" означает, что размеры макета выбираются по размерам родителя — например, это может быть размер экрана без учёта отступов.

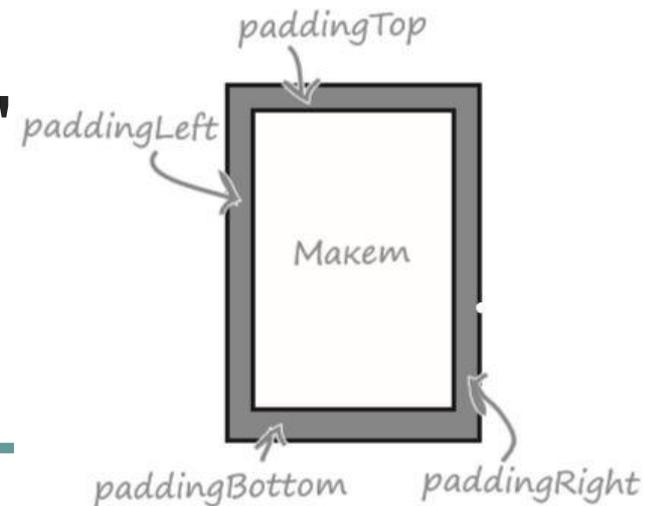
# Аппаратно-независимые пикселы

Некоторые устройства создают очень чёткие изображения за счёт использования очень маленьких пикселов. Другие устройства обходятся дешевле в производстве, потому что они используют меньшее количество более крупных пикселов. Чтобы интерфейсы не были слишком мелкими на одних устройствах и слишком крупными на других, можно использовать аппаратно-независимые пикселы (**dp**). Размеры, выраженные в аппаратно-независимых пикселах, приблизительно одинаковы на всех устройствах.

# Отступы (padding)

Если нужно, чтобы макет окружало некоторое пустое пространство, применяются атрибуты **padding**.

```
<RelativeLayout  
  android:paddingBottom="16dp"  
  android:paddingLeft="16dp"  
  android:paddingRight="16dp"  
  android:paddingTop="16dp">  
</RelativeLayout>
```



# Позиция вьюшки относительно родителя

- android:**layout\_alignParentBottom**="true"
- android:**layout\_alignParentTop**="true"
- android:**layout\_alignParentLeft**="true"
- android:**layout\_alignParentRight**="true"
- android:**layout\_centerInParent**="true"
- android:**layout\_centerHorizontal**="true"
- android:**layout\_centerVertical**="true"
- **ПРАКТИКА (проверка свойств)**

# Позиция относительно другой вьюшки

- android:**layout\_above**="@+id/view\_id"
- android:**layout\_below**="@+id/view\_id"
- android:**layout\_alignTop**="@+id/view\_id"
- android:**layout\_alignBottom**="@+id/view\_id"
- android:**layout\_alignLeft**="@+id/view\_id"
- android:**layout\_alignRight**="@+id/view\_id"
- android:**layout\_toLeftOf**="@+id/view\_id"
- android:**layout\_toRightOf**="@+id/view\_id"
- **ПРАКТИКА (проверка свойств)**

# Интервалы (margin)

Когда применяются атрибуты для размещения, вьюшки располагаются вплотную друг к другу. Чтобы вьюшки разделялись промежутками, нужно добавить интервалы.

- android:**layout\_marginTop**="20dp"
- android:**layout\_marginBottom**="20dp"
- android:**layout\_marginLeft**="20dp"
- android:**layout\_marginRight**="20dp"

# Linear Layout

Линейный макет определяется при помощи элемента XML `<LinearLayout>`.

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

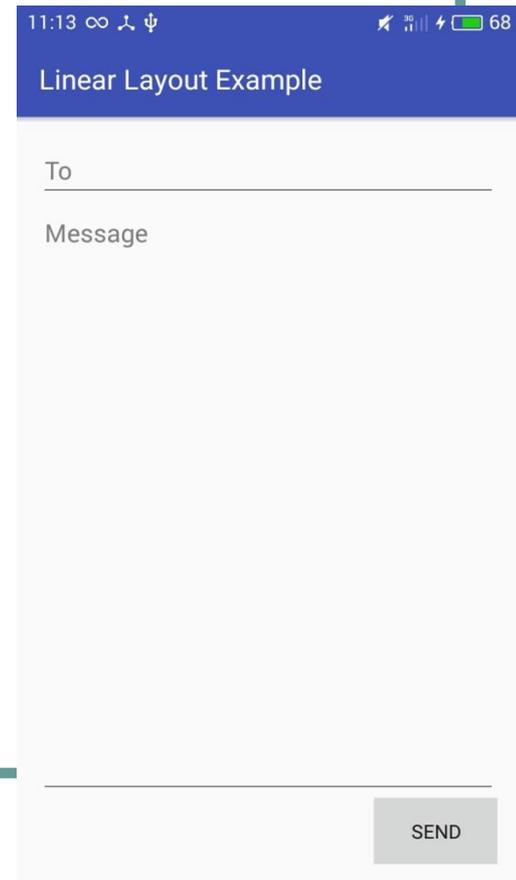
# Последовательность отображения

При определении линейного макета вьюшки включаются в XML-макет в том порядке, в котором они должны следовать на экране. То есть, если нужно, чтобы надпись размещалась над кнопкой, надпись **должна** определяться первой в разметке. В линейном макете айдишники вьюшек понадобятся только в том случае, если нужно будет явно обращаться к ним из кода активити. Чтобы указать, где должна размещаться вьюшка, разработчику не нужно обращаться к другим вьюшкам.

# Пример линейной разметки

<https://git.io/vilMy>

- `android:hint="To"`
- `android:layout_weight="1"`
- `android:gravity="top"`
- `android:layout_gravity="right"`



# Подсказки (hint)

В двух текстовых полях выводятся подсказки “To” и “Message”. Подсказка представляет собой временный текст, который выводится в пустом текстовом поле. Этот текст дает пользователю представление о том, какие данные следует вводить в этом поле. Текст определяется при помощи атрибута **android:hint**.

# Весовой коэффициент

Чтобы текстовое поле Message растянулось по вертикали, занимая всё свободное пространство макета, нужно было назначить этой вьюшке **весовой коэффициент**, или **вес**. Назначение весов — способ приказать вьюшке занять дополнительное пространство в макете. Для назначения веса вьюшке используется атрибут **android:layout\_weight**.

**Практика: назначить полю To вес 2.**

# Выравнивание содержимого

Атрибут **android:gravity** позволяет указать, как содержимое должно размещаться внутри вьюшки. Например, как текст должен позиционироваться в текстовом поле. Если нужно, чтобы текст выводился у верхнего края, следующий фрагмент кода обеспечит нужный эффект: **android:gravity="top"**.  
Другие значения: bottom, left, right, center\_vertical, center\_horizontal, center, fill\_vertical, fill\_horizontal, fill.

# Выравнивание самой вьюшки

Атрибут **android:layout\_gravity** позволяет указать, в какой части внешнего пространства должна находиться вьюшка в линейном макете. Например, атрибут может использоваться для смещения вьюшки вправо или для горизонтального выравнивания по центру. Для смещения кнопки вправо в её разметку включается следующий атрибут:

**android:layout\_gravity="right"**

# Grid Layout

В табличном макете экран разбивается на строки и столбцы, а вьюшки связываются с ячейками.

Количество столбцов в табличном макете задаётся следующим атрибутом:

**`android:columnCount="2"`**.

Количество строк лучше доверить системе - Android создаст столько строк, сколько потребуется для отображения всех вьюшек.

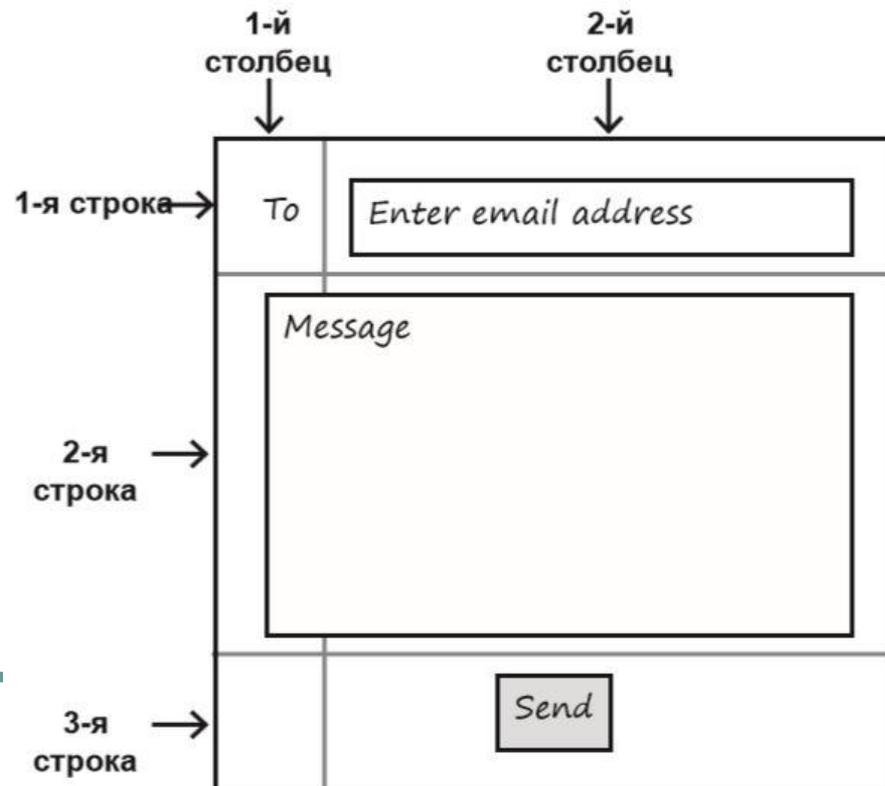


# Особенности Grid Layout

Как и в случае с линейным макетом, не обязательно назначать вьюшкам айдишники – т.к. им не придётся обращаться друг к другу в макете. По умолчанию табличный макет размещает вьюшки в порядке их следования в XML. Если создать табличный макет с двумя столбцами, табличный макет поместит первую вьюшку в ячейку 0-0, вторую – в ячейку 0-1, и тд. У такого решения есть один недостаток: исключение одной из вьюшек из макета может привести к серьёзному изменению внешнего вида макета. Чтобы избежать подобных проблем, можно указать явно, где должна быть каждая вьюшка и сколько столбцов она будет занимать.

# Построение эскиза

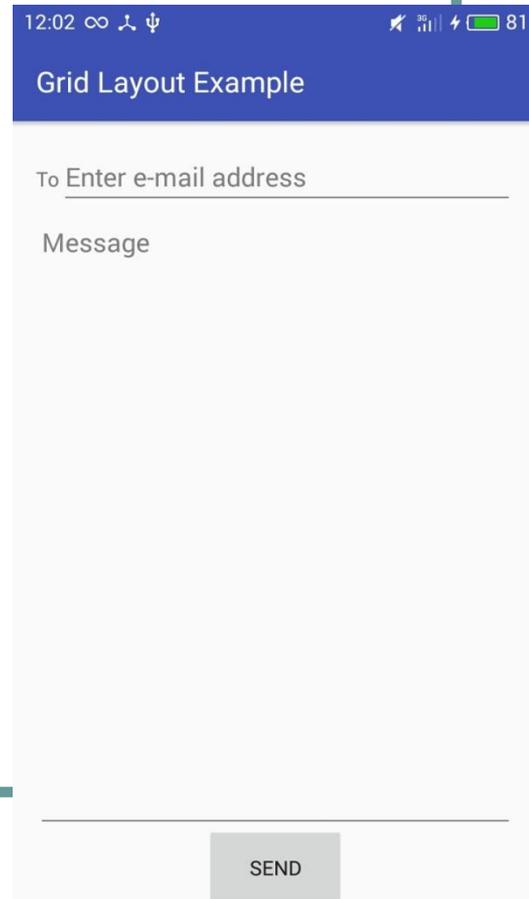
Создание нового табличного макета начинается с построения эскиза. Это поможет понять, сколько строк и столбцов потребуется, где должна находиться каждая вьюшка и сколько столбцов она будет занимать.



# Пример табличной разметки

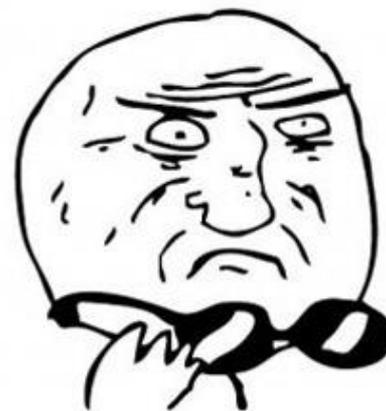
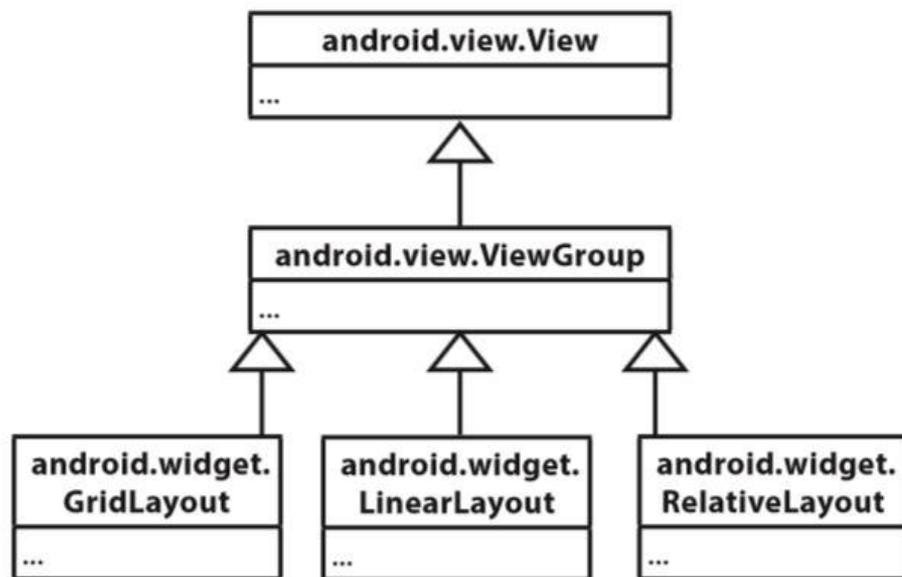
<https://git.io/vilH2>

- `android:layout_column="0"`
- `android:layout_row="0"`
- `android:layout_columnSpan="2"`



# ViewGroup

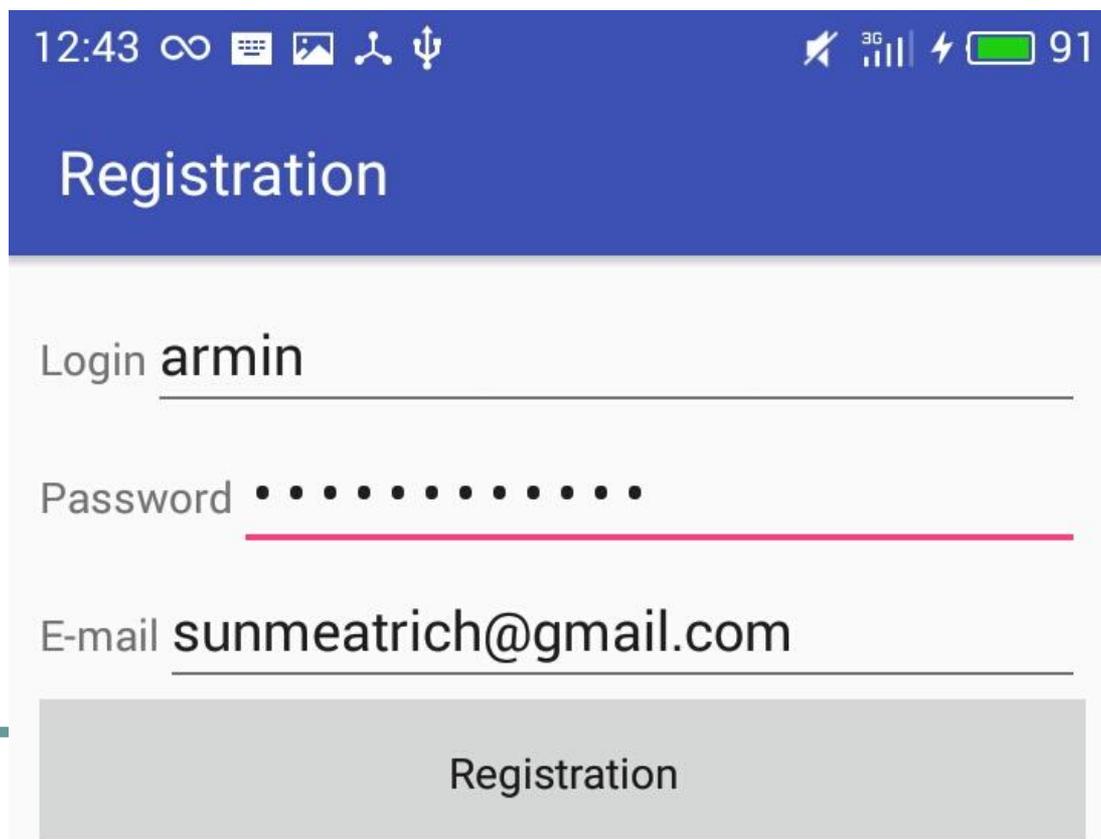
Не только компоненты графического интерфейса (вроде кнопок и текстовых полей) являются специализациями класса `View`. В иерархии Android макет является специализацией класса `android.view.ViewGroup`. Группа представлений — особая разновидность вьюшек, способных содержать другие вьюшки. А это значит, что элементом макета может быть другой макет!



**MOTHER OF GOD**

# Пример вложенности макетов

<https://git.io/vi1FH>



12:43 ∞ [icons] 3G [signal] [battery] 91

## Registration

Login

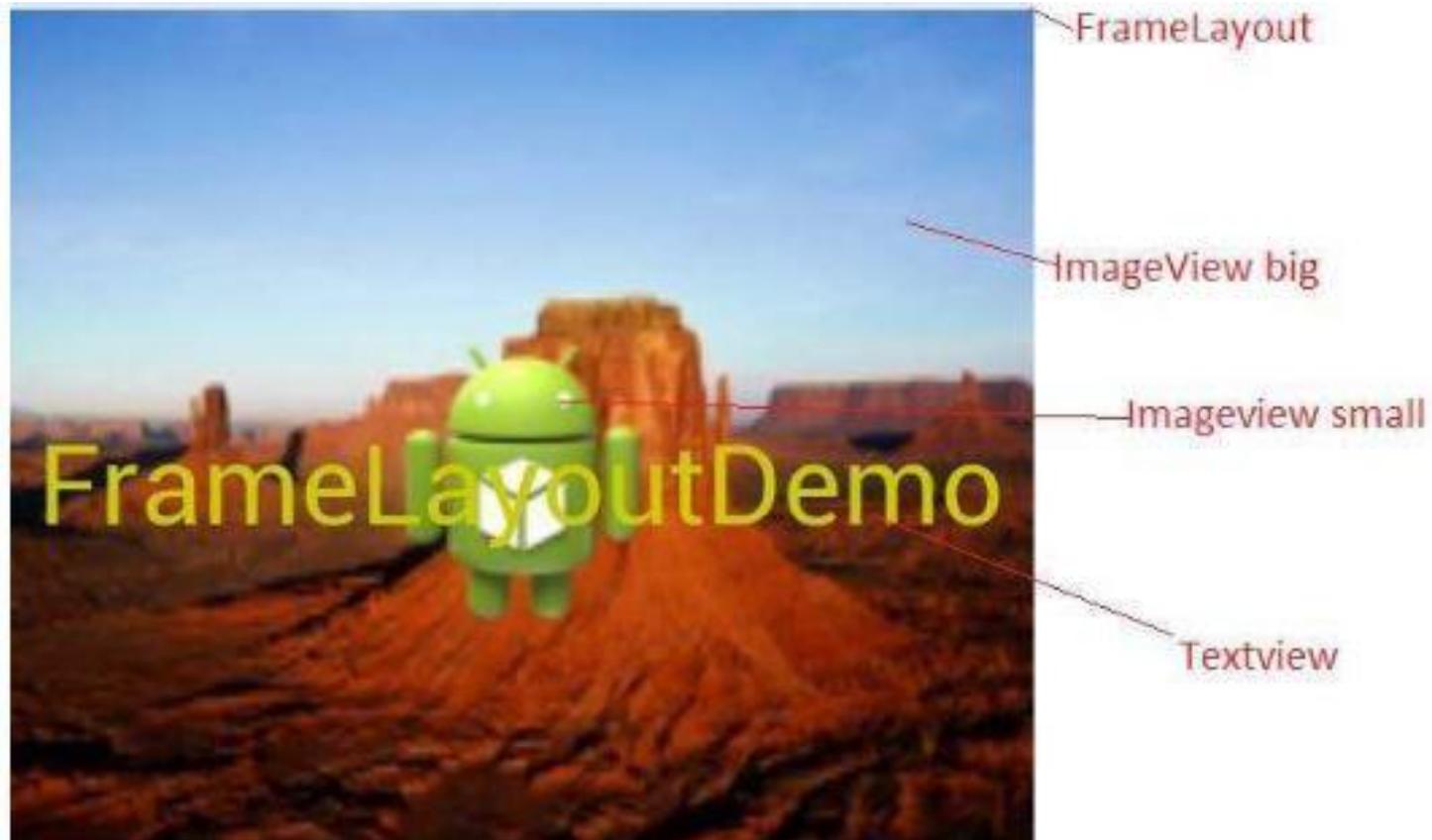
Password

E-mail

Registration

# Frame Layout

- [http://www.tutorialspoint.com/android/android\\_frame\\_layout.htm](http://www.tutorialspoint.com/android/android_frame_layout.htm)



# Frame Layout

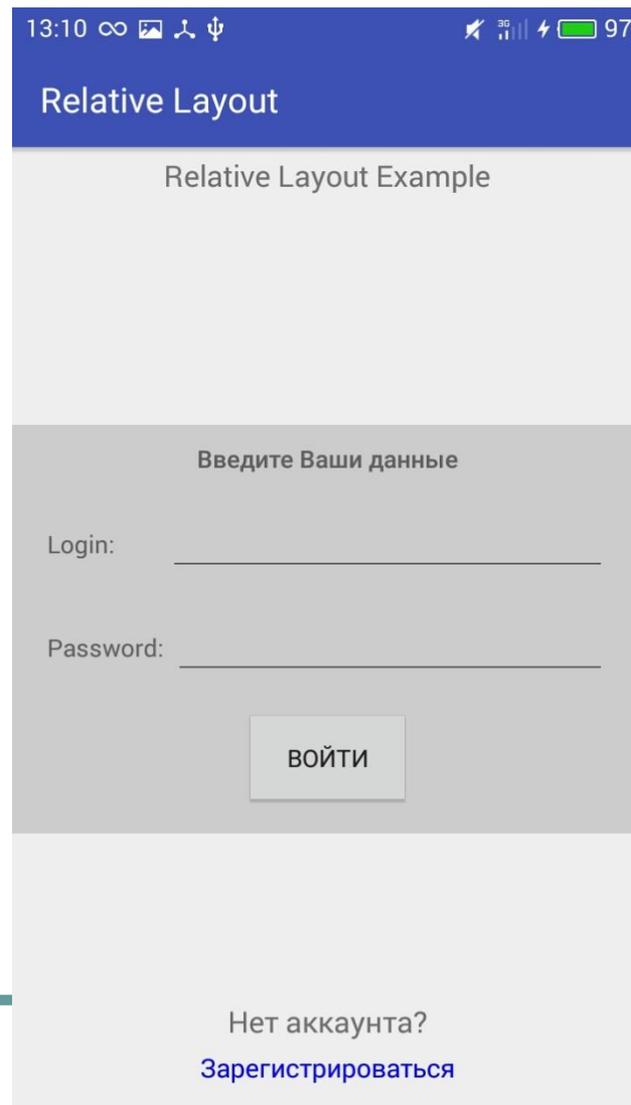
**FrameLayout** является самым простым типом разметки. Обычно это такое пространство на экране, которое можно заполнить только дочерними объектами **View** или **ViewGroup**. Все дочерние элементы **FrameLayout** прикрепляются к верхнему левому углу экрана. В разметке **FrameLayout** нельзя определить особое местоположение для дочернего объекта. Последующие дочерние объекты **View** будут просто рисоваться поверх предыдущих компонентов, частично или полностью перекрывая их.

# Практика

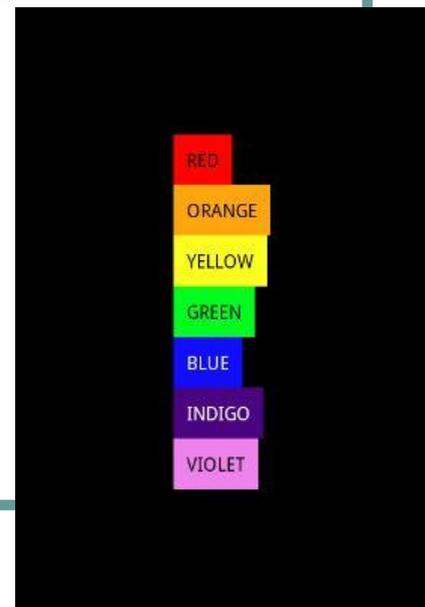
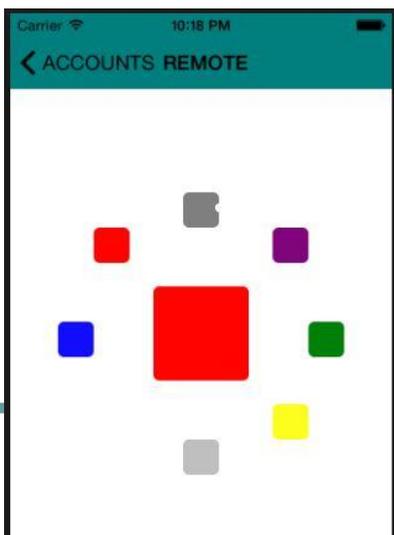
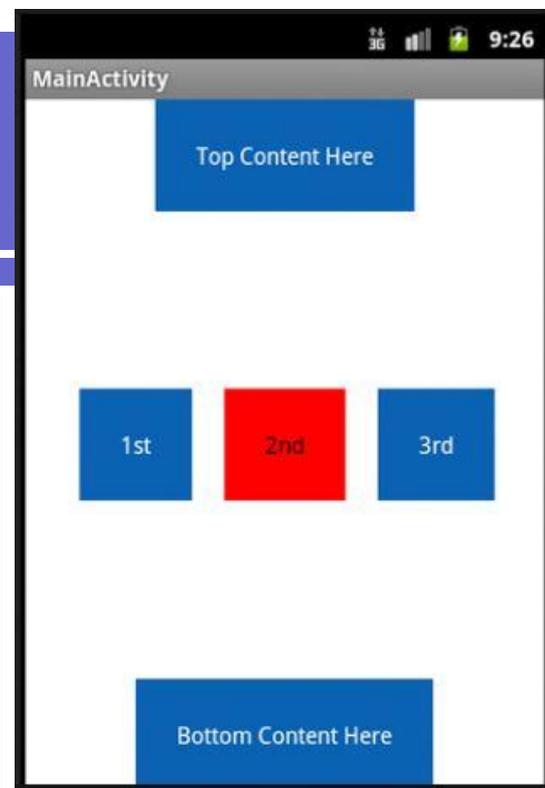
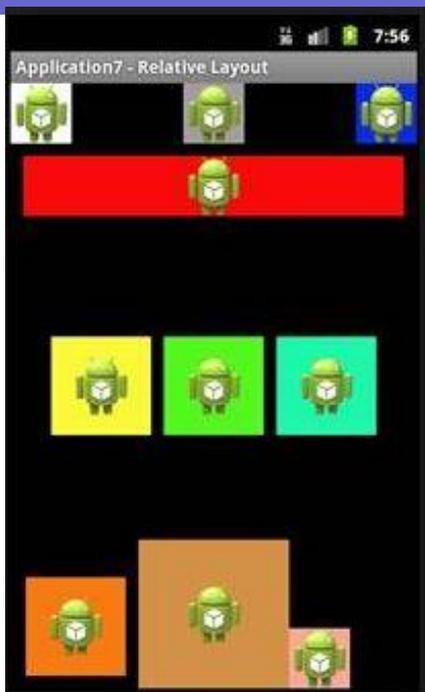
- Сделать кнопку, на которой есть и фоновая картинка, и текст (элемент `Button`, не `ImageButton`).
- Добиться эффекта, как на картинке с 33-го слайда: сделать `FrameLayout`, в нём разместить два элемента `ImageView` и один элемент `TextView`.

# Пример на Relative Layout

git.io/vilx0



# Практика RL (любые 2)



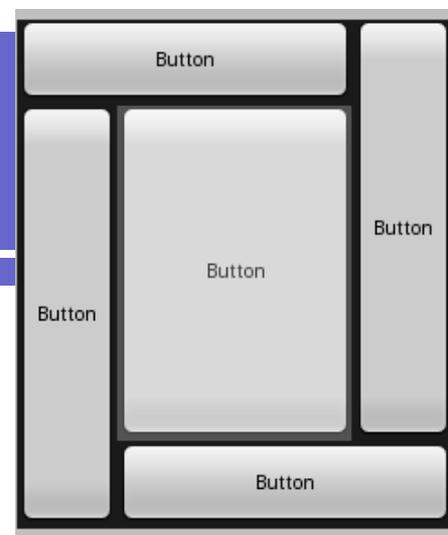
# Практика LL

	Cupcake 1.5
	Eclair 2.0-2.1
	Froyo 2.2-2.2.3
	Gingerbread 2.3-2.3.7
	Honeycomb

## SCHEDULE

Thu	Fri	Sat
12.00		13.00
MAIN STAGE		
		<b>THE JOY FOR MIDA...</b> ☆ 12:35-13:15
MARQUEE		
<b>DRY THE RIVER</b> ☆ 12:00-12:40		
DANCE HALL		
		<b>DISCLOSURE LIVE</b> ☆ 12:35-13:15

# Практика GL (любые 2)



Simple Workout

Menu Save

Sets/Reps  Time/Distance

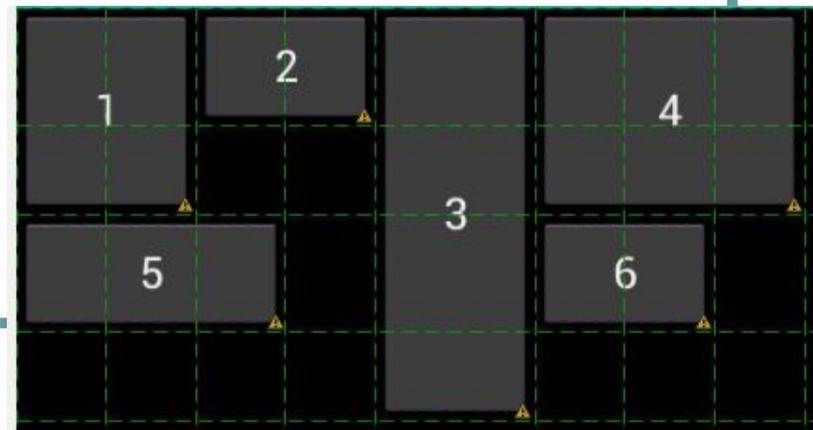
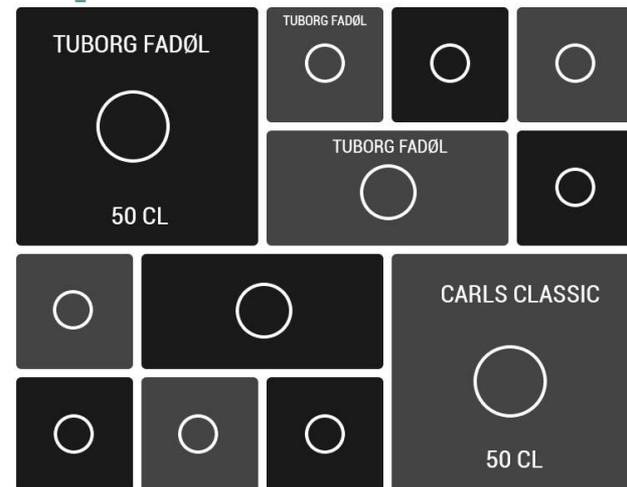
Sets 5

Reps 15

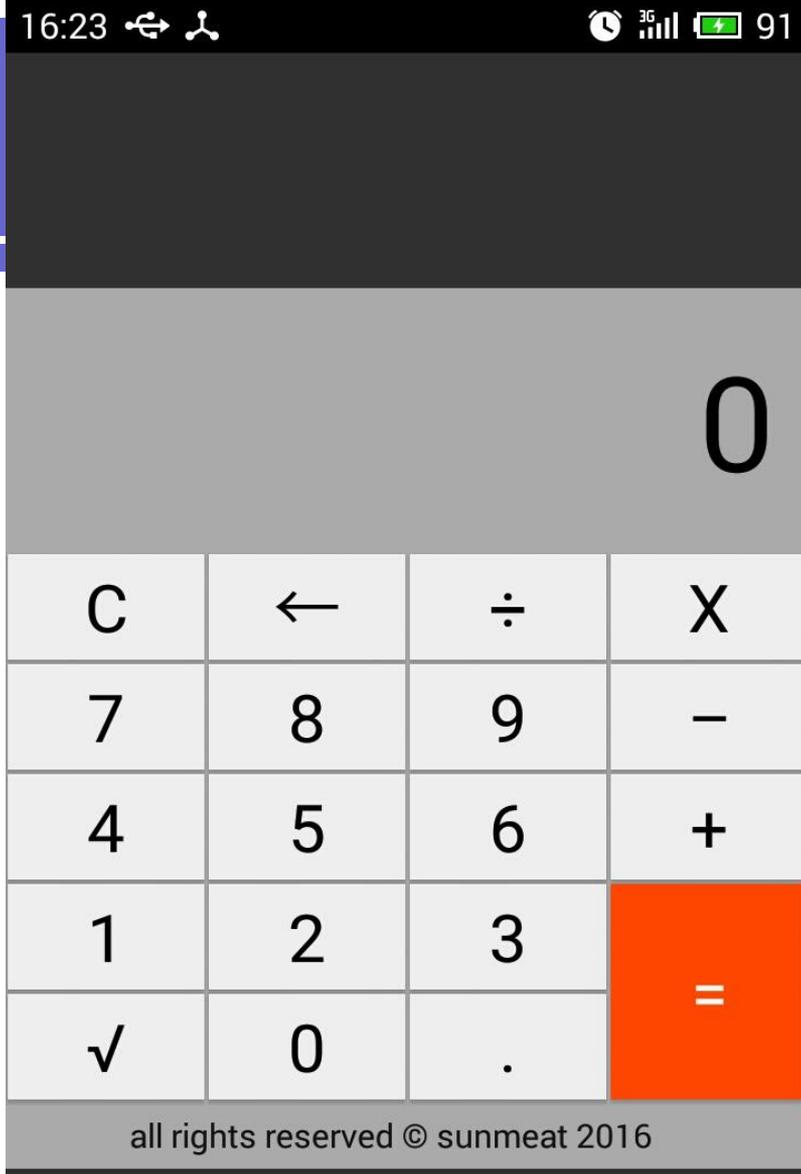
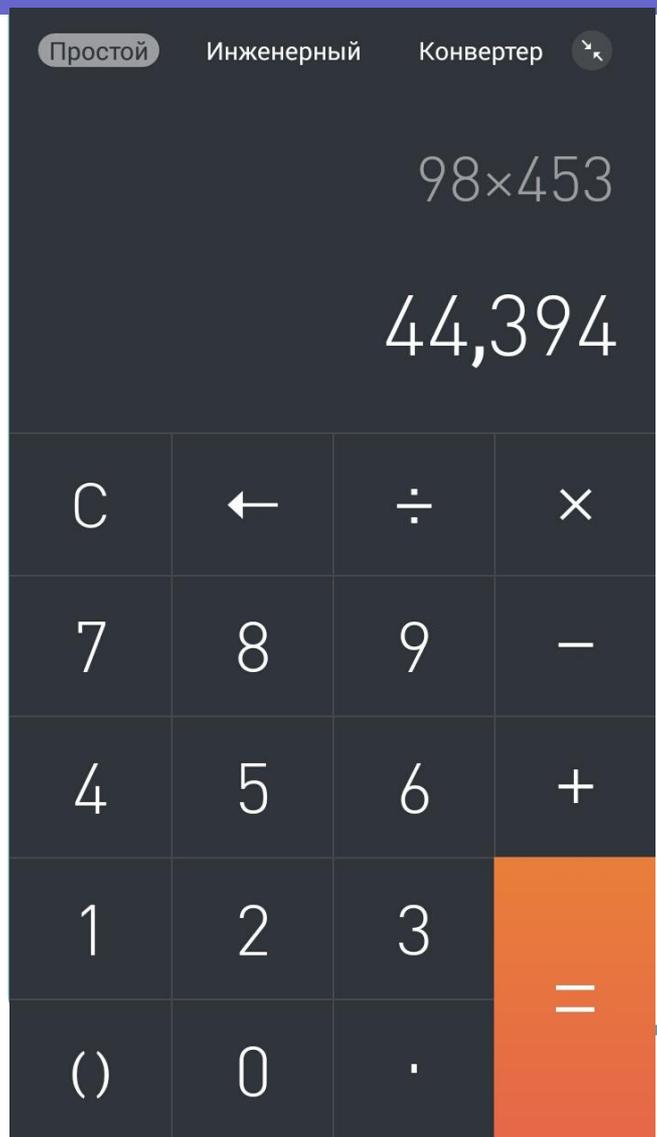
Weight 35 lbs

Time hr mi ss ms

Distance 3.1 miles



# Калькулятор



```
// AndroidManifest.xml
<activity
    android:name=".MainActivity"
    android:theme="@style/Theme.AppCompat.NoActionBar">
```

# Constraint Layout

- <https://codelabs.developers.google.com/codelabs/constraint-layout/index.html#0>
- <https://developer.android.com/training/constraint-layout/index.html>
- <https://medium.com/exploring-android/exploring-the-new-android-constraintlayout-eeed37fe8d8f1#.sh4gzn6om>
- <https://medium.com/google-developer-experts/first-impressions-of-androids-new-constraintlayout-c6d081b2bc2a#.m3b4c8imi>

