Widgets

android

Вьюшки и виджеты

View class represents the basic building block for user interface components. A View occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for **widgets**, which are used to create interactive UI components (buttons, text fields, etc.). The ViewGroup subclass is the base class for layouts, which are invisible containers that hold other Views (or other ViewGroups) and define their layout properties.

View

A **View** is a base class for all UI elements. It therefore covers many different classes and concepts, including Widgets, ViewGroups and Layouts. There is a root View attached to a Window instance which forms the basis of the View hierarchy. In general, the word View is usually used to describe UI elements in general, or to refer to abstract or base UI classes such as ViewGroups.

Widget

There are various definitions for this term, but most refer to a 'ready to use' UI element, be it a Button, ImageView, EditText, etc. Note that some people consider Widgets to be UI elements that are complete (not abstract) and are not containers (such as ViewGroups (Layouts/ListViews).

Основные виджеты

- Надпись (TextView)
- Текстовые поля (EditText)
- Кнопка (Button)
- Двухпозиционная кнопка (ToggleButton)
- Выключатель (Switch)
- Флажок (CheckBox)
- Переключатели (RadioButtons)
- Раскрывающийся список (Spinner)
- Графическое представление (ImageView)
- Графическая кнопка (ImageButton)
- Полосы прокрутки (ScrollView, HorizontalScrollView)

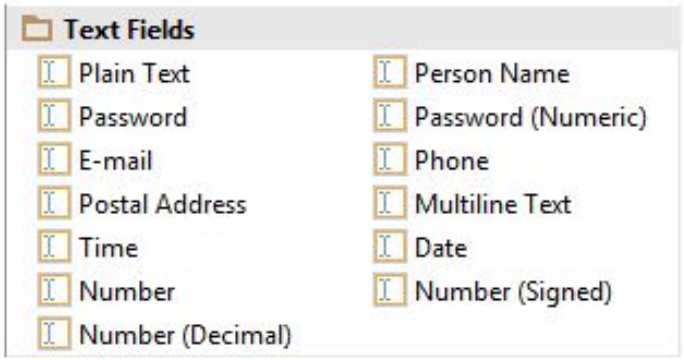
Надпись (TextView)

Используется для вывода текста.

```
<TextView
    android:id="@+id/text_view"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/text" />
```

Текстовое поле (EditText)

Аналог надписи, но с возможностью редактирования.



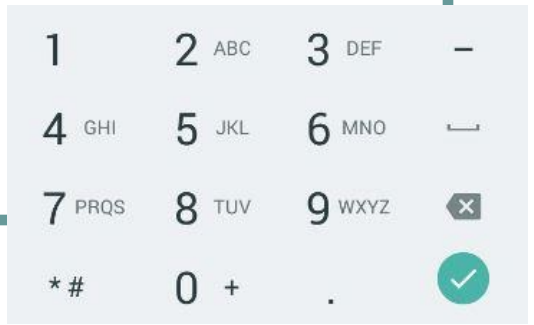

```
<EditText
    android:id="@+id/edit_text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:hint="@string/hint" />
```

Текстовое поле (EditText)

Атрибут **android:inputType** определяет тип данных, которые должны вводиться в поле. Эта информация позволяет помочь пользователю в процессе ввода. Например, если поле предназначено для ввода числовых данных, используется атрибут **android:inputType="number"**

EditText – атрибут inputType

- **phone** - предоставляет клавиатуру для ввода телефонных номеров
- **textPassword** - предоставляет клавиатуру для ввода текста, вводимые данные маскируются
- **textCapSentences** - первое слово в предложении начинается с прописной буквы (textCapWords, textCapCharacters)
- **textAutoCorrect** - автоматически исправляет вводимый текст



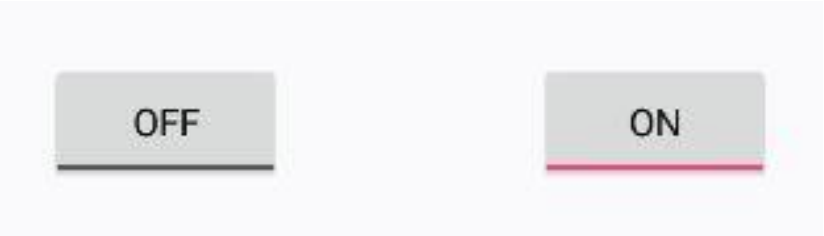
Кнопка (Button)

Обычно используется для того, чтобы приложение выполняло какие-либо действия при щелчке на кнопке.

```
<Button
    android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/button_text" />
```

ToggleButton

Щёлкая на двухпозиционной кнопке, пользователь выбирает одно из двух состояний.



<ToggleButton
android:id="@+id/toggle_button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textOn="@string/on"
android:textOff="@string/off" />

ToggleButton

Чтобы двухпозиционная кнопка реагировала на щелчки, нужно включить атрибут **android:onClick** в XML макета.

Для получения состояния кнопки:

boolean on = ((ToggleButton)view).isChecked();

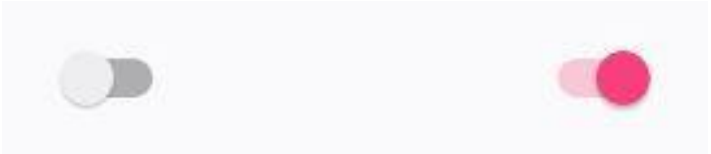
MainActivity implements CompoundButton.OnCheckedChangeListener

Установка в нажатое состояние из XML:

```
<ToggleButton
    android:id="@+id/toggle_button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_columnWeight="1"
    android:layout_gravity="center"
    android:checked="true"
    android:textOff="OFF"
    android:textOn="ON" />
```

Выключатель (Switch)

Выключатель представляет собой рычажок, который работает по тому же принципу, что и ToggleButton.

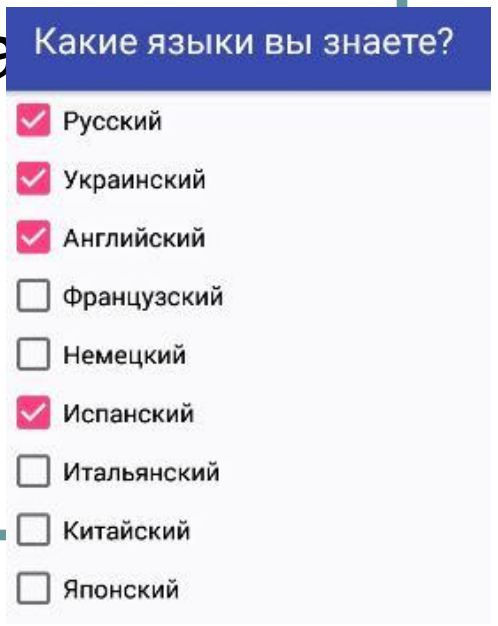


```
<Switch
    android:id="@+id/switch_view"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textOn="@string/on"
android:textOff="@string/off" />
```

Флажок (CheckBox)

Флажки (check boxes) предоставляют пользователю набор независимых вариантов. Пользователь может выбрать любые варианты по своему усмотрению. Каждый флажок может устанавливаться или сниматься независимо от всех оста флажков.

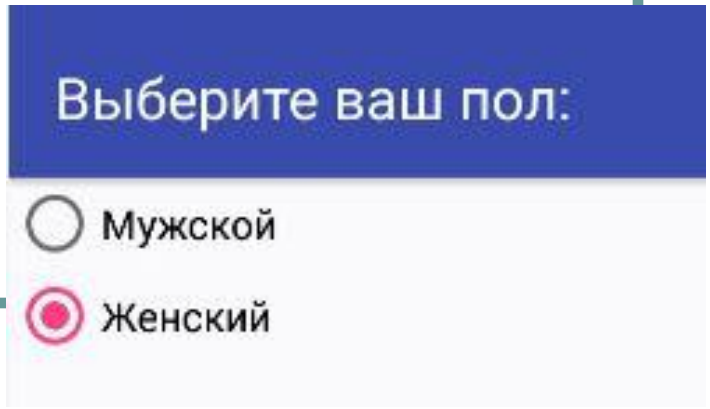
```
<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:checked="true"
    android:text="Украинский" />
```



Переключатель (RadioButton)

Переключатели (radio buttons) предоставляют набор вариантов, из которого пользователь может выбрать ровно один вариант.

```
<RadioGroup
    android:id="@+id/radio_group"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton
        android:id="@+id/man"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Мужской" />
    <RadioButton
        android:id="@+id/woman"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Женский" />
</RadioGroup>
```



Какой переключатель выбран?

```
RadioGroup radioGroup =
findViewById(R.id.radioGroup);


int id = radioGroup.
getCheckedRadioButtonId(); if (id == -1)
{
    // ничего не выбрано
} else {
    RadioButton radioButton =
findViewById(id);
}
```

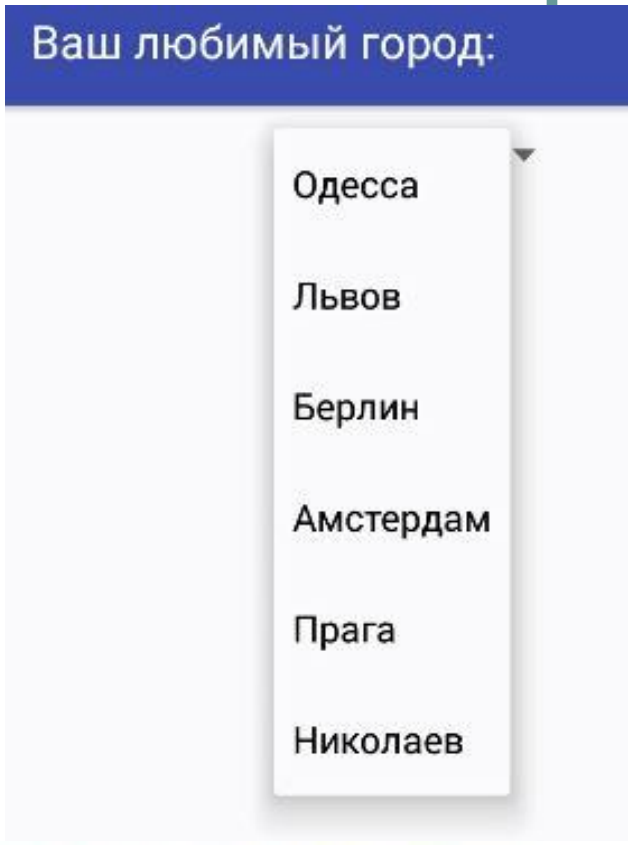
Практика

- Сделать четыре переключателя с текстами RED, GREEN, BLUE, YELLOW. Выбор переключателя меняет цвет фона (MainActivity implements RadioGroup.OnCheckedChangeListener).
- Сделать четыре флажка и обычную кнопку под ними. Нажатие на кнопку показывает тексты отмеченных флажков.

Раскрывающийся список (Spinner)

Раскрывающийся список содержит набор значений, из которых пользователь может выбрать только одно.

```
<Spinner
    android:layout_marginTop="10dp"
    android:layout_gravity="center"
    android:id="@+id/spinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_values" />
```



MainActivity.java × strings.xml ×

Edit translations for all locales in the translations editor.

```
<resources>
    <string name="app_name">Ваш любимый город:</string>

    <string-array name="spinner_values">
        <item>Одесса</item>
        <item>Львов</item>
        <item>Берлин</item>
        <item>Амстердам</item>
        <item>Прага</item>
        <item>Николаев</item>
    </string-array>

</resources>
```

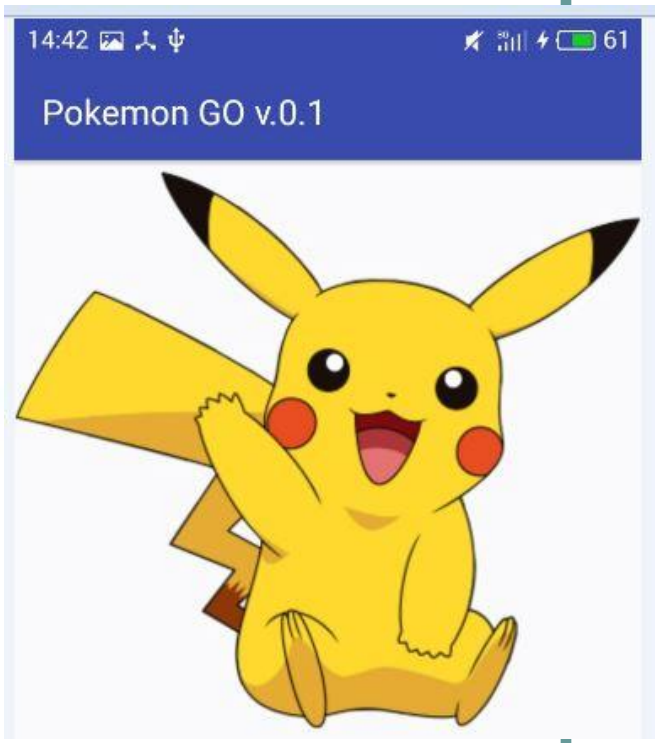
```
Spinner spin = (Spinner) findViewById(R.id.spinner);
String item = String.valueOf(spin.getSelectedItem());
```

ImageView

Графическое представление используется для вывода изображений. Изображение включается в проект как ресурс. В папке app/src/main/res находится папка с именем drawable. Она используется по умолчанию для хранения ресурсов изображений. Чтобы добавить файл с изображением, его перетаскивают в эту папку, либо просто Ctrl+C по файлу, Ctrl+V на папке.

Обращение к ImageView

```
<ImageView
    android:id="@+id/pikachu"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:src="@drawable/pikachu" />
```




```
ImageView pokemon = (ImageView) findViewById(R.id.pikachu);
int image = R.drawable.pikachu;
pokemon.setImageResource(image);
```

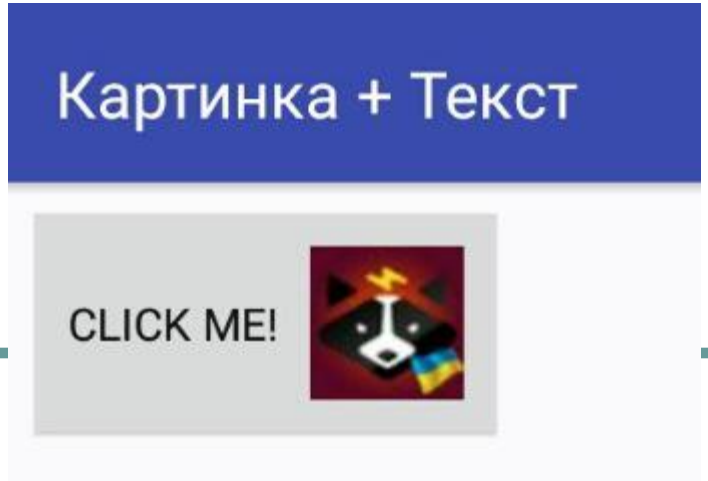
ImageButton

Графическая кнопка почти не отличается от обычной — просто на ней выводится только изображение, **без текста**.

```
<ImageButton
    android:id="@+id/img_button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/button_pic />
```

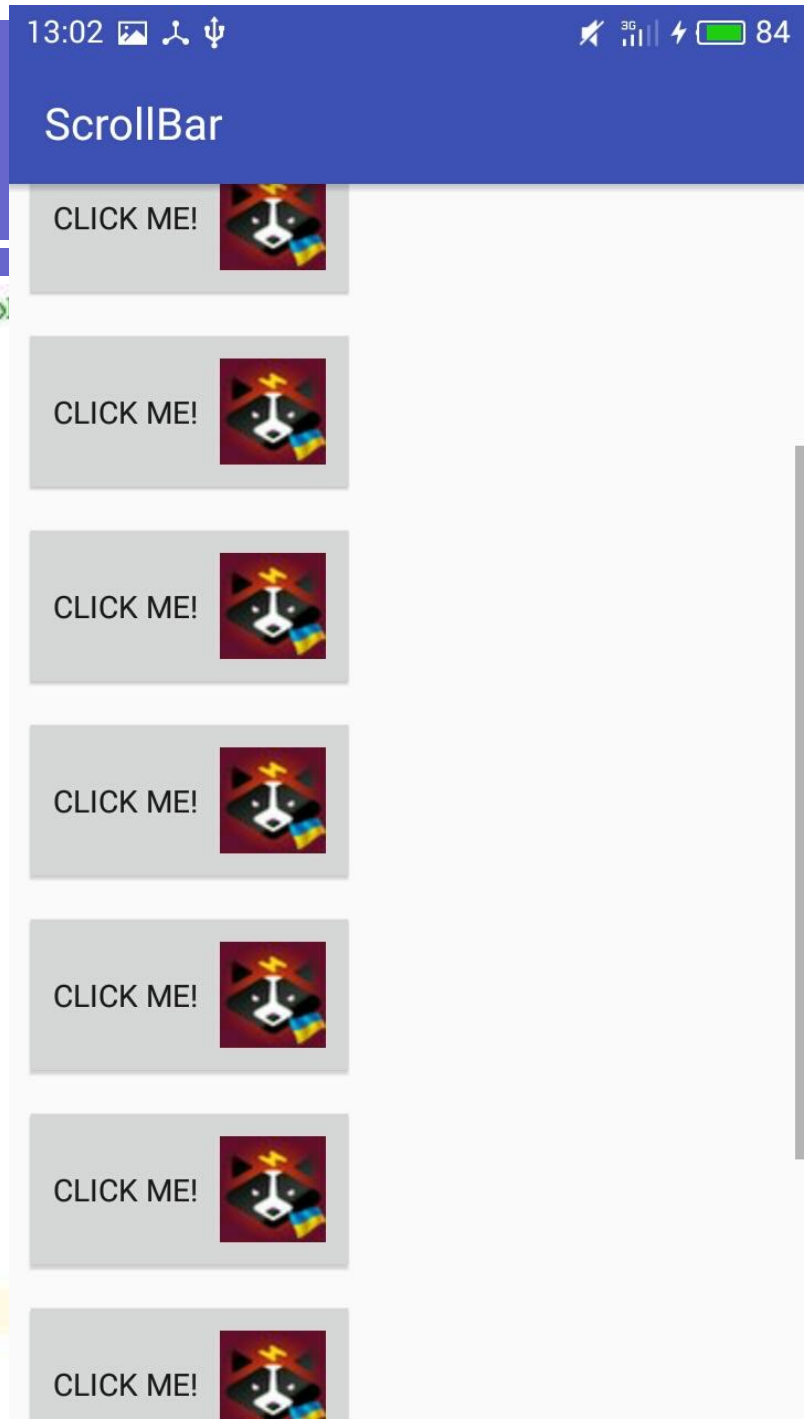
Картинка + текст на кнопке

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:drawableRight="@drawable/enot"
    android:drawablePadding="10dp"
    android:padding="10dp"
    android:text="CLICK ME!" />
```

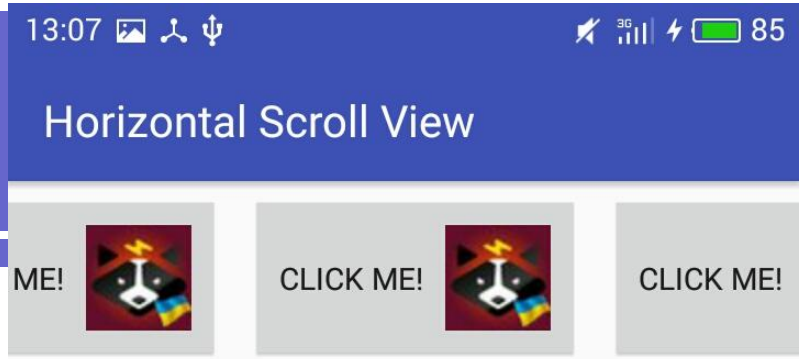


ScrollView

```
<ScrollView xmlns:android="http://schemas.android.com/ap
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:drawablePadding="10dp"
            android:drawableRight="@drawable/enot"
            android:padding="10dp"
            android:text="CLICK ME!" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
```



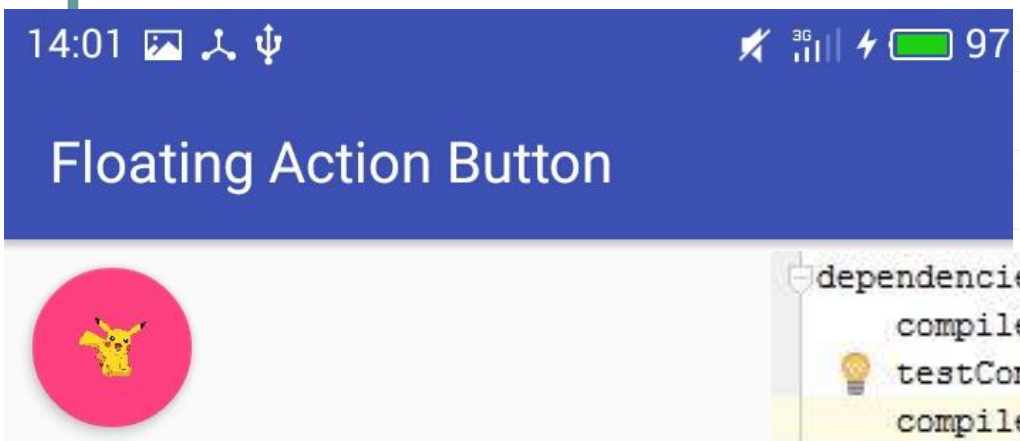
HorizontalScrollView

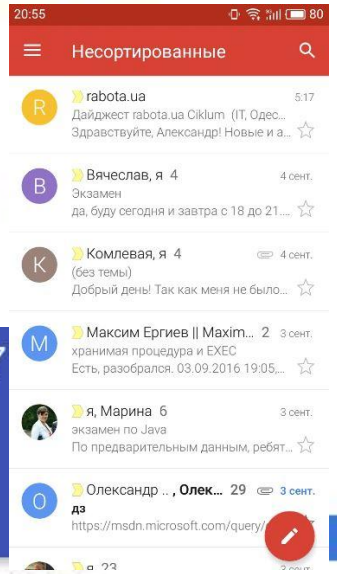


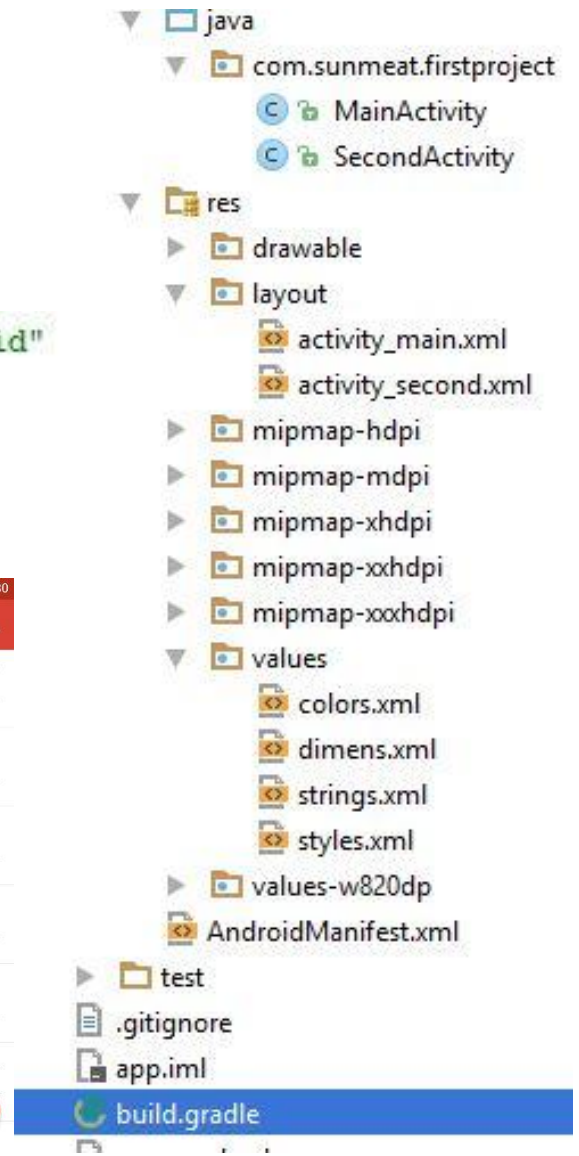

```
<HorizontalScrollView xmlns:android="http://schemas
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

Floating Action Button

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"

  <android.support.design.widget.FloatingActionButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_margin="10dp"
      app:backgroundTint="@color/colorAccent"
      android:src="@drawable/pikachu" />
```





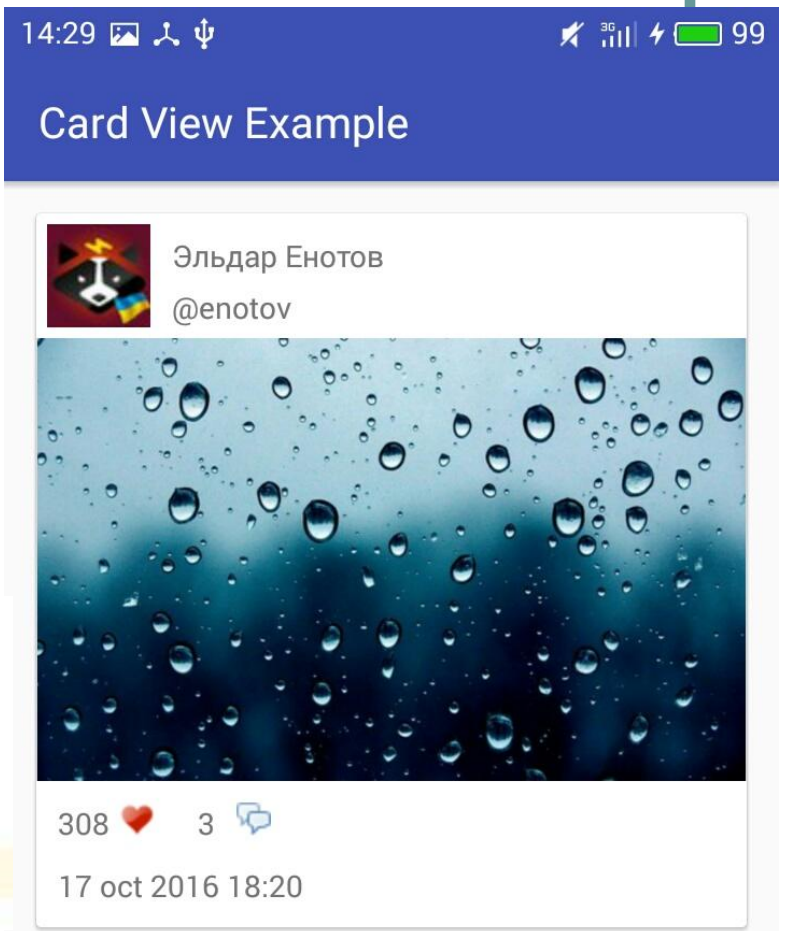



```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:24.2.0'
    compile 'com.android.support:design:24.2.0'
}
```

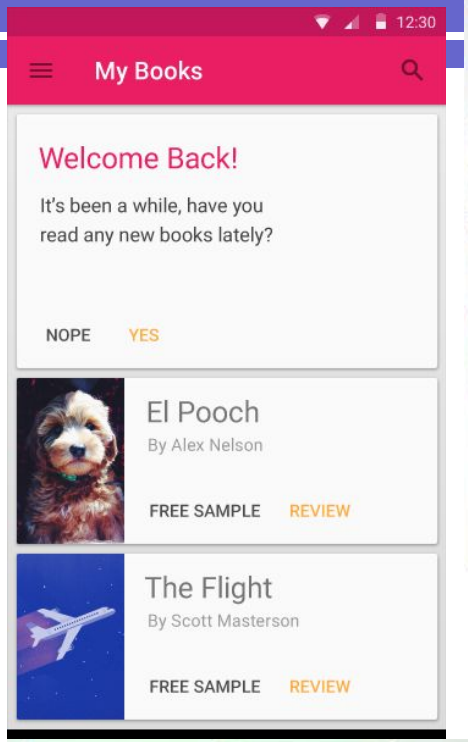
CardView

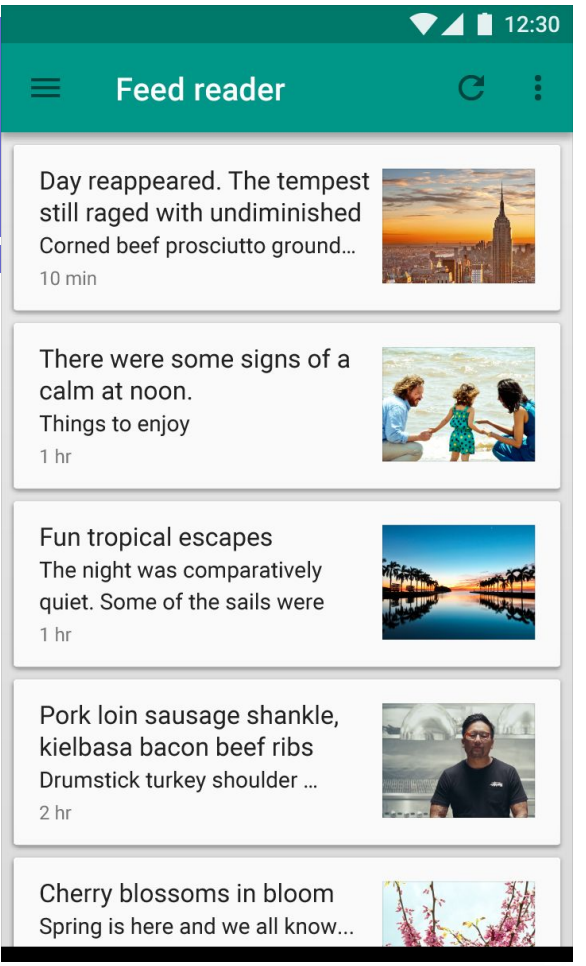
https://git.io/vi3j0

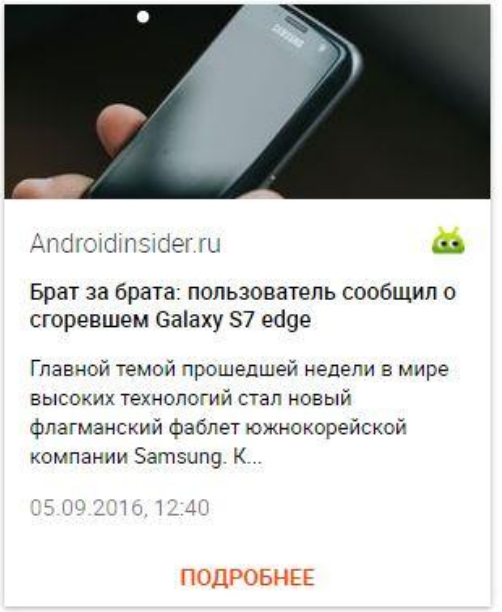
```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:24.2.0'
    compile 'com.android.support:cardview-v7:24.0.0'
}
```

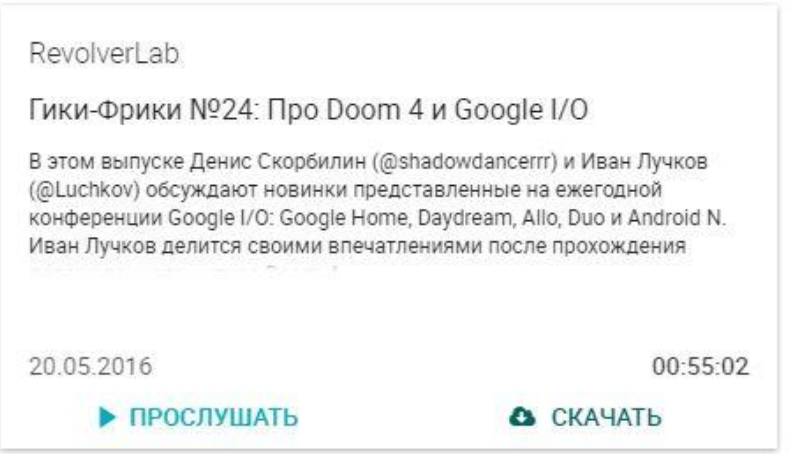


Практика на CardView



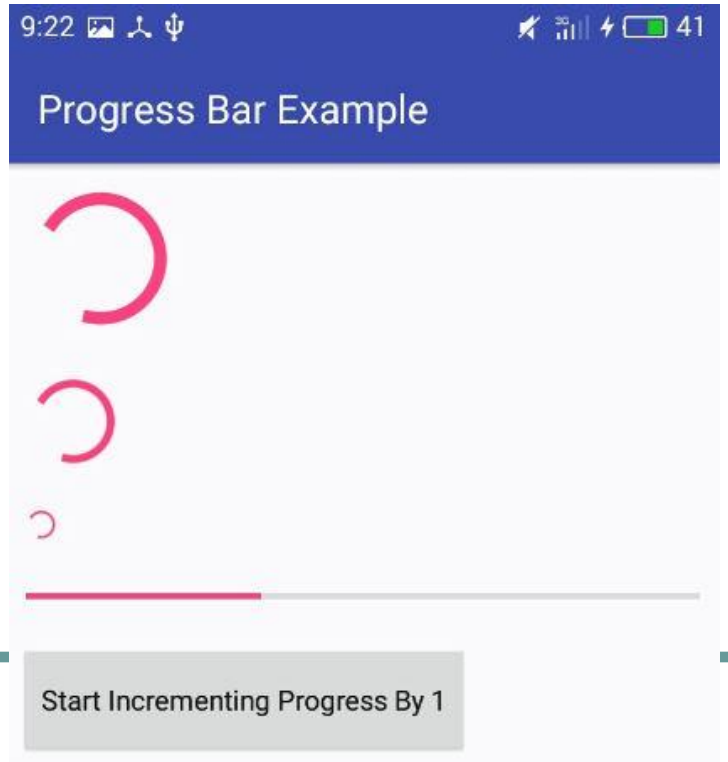






ProgressBar

- https://git.io/viG28 (XML)
- https://git.io/viG22 (Java)



- https://developer.android.com/reference/android/widget/ProgressBar.html
- https://www.mkyong.com/android/android-progress-bar-example/
- http://www.tutorialspoint.com/android/android_progressbar.htm
- http://www.journaldev.com/9652/android-progressdialog-example

SeekBar

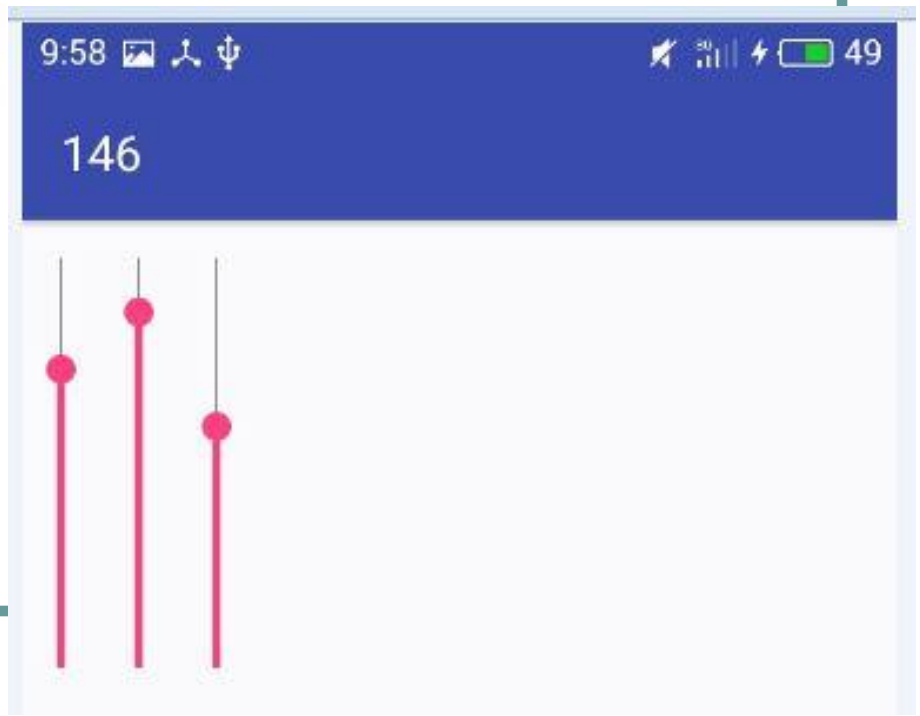
https://git.io/viG2Q

```
<SeekBar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/seekBar" />
```

Vertical SeekBar

https://git.io/viG2A

```
<com.sunmeat.firstproject.VerticalSeekBar
    android:id="@+id/seekBar1"
    android:layout_width="wrap_content"
    android:layout_height="200dp" />
```



Практика

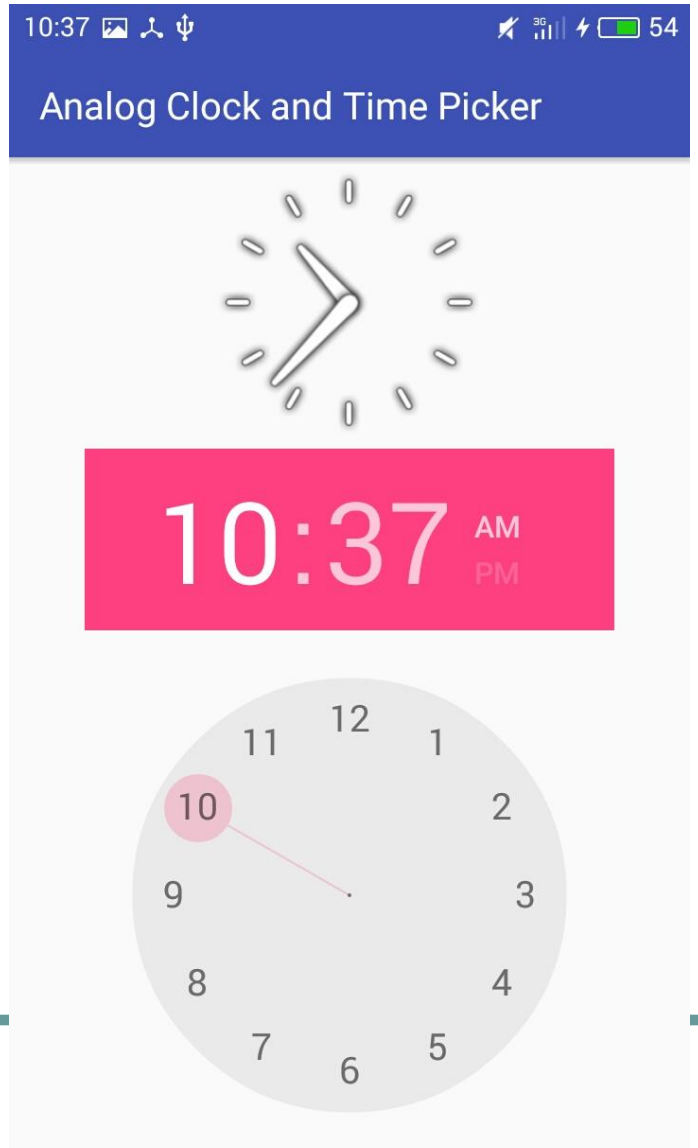
- Сделать три сикбара с диапазоном от 0 до 255, для каналов R, G и B. При изменении положения сикбаров меняется фон приложения.

RatingBar and DatePicker

```
<RatingBar
    android:id="@+id/ratingBar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_margin="10dp"
    android:progressTint="@color/colorGold" />

<DatePicker
    android:id="@+id/datePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_margin="10dp" />
```

```
    rb = (RatingBar) findViewById(R.id.ratingBar);
    rb.setOnRatingBarChangeListener(this);
}

@Override
public void onRatingChanged(RatingBar ratingBar, float v, boolean b) {
    Toast.makeText(this, v + "", Toast.LENGTH_LONG).show();
}
```

AnalogClock and TimePicker



Программное создание вьюшек

```
LinearLayout layout = (LinearLayout) findViewById(R.id.layout);
Button b = new Button(this);
b.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT,
        LinearLayout.LayoutParams.WRAP_CONTENT));
b.setText("Button");
layout.addView(b);
```

Предполагается, что в XML-файле разметки есть тэг LinearLayout с атрибутом android:id=«@+id/layout».

Практика: проверить программный способ размещения одной кнопки на LinearLayout

Практика

Отобразить на экране устройства программно созданный двумерный массив кнопок размерностью M x N. Размеры кнопок по ширине и высоте, отступы между кнопками задаются программно. На каждой кнопке должен быть её порядковый номер. При нажатии на любую кнопку появляется уведомление с текстом "строка X, столбец Y", после чего кнопка деактивируется. Обеспечить наличие скроллов для просмотра всех кнопок.

Результаты


11:19
52
Program Buttons in GridLayout
0 1 2 3 4 5
10 11 12 13 14 15
20 21 22 23 24 25
30 31 32 33 34 35
40 41 42 43 44 45
50 51 52 53 54 55
60 61 62 63 64 65
70 71 72 73 74 75
80 81 82 83 84 85

Практика. Крестики-нолики

Любым способом (программно либо разметкой) создать игровое поле, состоящее из 9 кнопок (3х3). При нажатии на кнопку на ней появляется картинка (крестик или нолик). Нельзя поставить крестик или нолик в уже занятую клетку. Предусмотреть отдельную кнопку, которая начинает новую игру. Предусмотреть 2-3 уровня сложности (использовать переключатели). Предусмотреть флажок, который определяет, кто ходит первым – человек или компьютер.

Практика. 16 кнопок

Написать игру, смысл которой состоит в следующем. На игровом поле есть 16 кнопок и прогрессбар. На кнопки необходимо разместить 16 случайных чисел из диапазона от 0 до 100. Задача состоит в том, чтобы за указанное время (например, за 1 минуту), пока не заполнится весь прогресс-бар, щёлкнуть по всем кнопкам в порядке возрастания чисел. Если нажать на кнопку, где число не является следующим по возрастанию – отнимается одна секунда времени. Если все кнопки нажаты в правильном порядке – вывести уведомление «Победа». Если время закончилось – вывести уведомление «Вы проиграли». Таймер стартует при первом нажатии на кнопку.

Результаты

Практика

Реализовать простую версию игры «Сапёр» (без меню и анимаций).