Программные и аппаратные средства информатики

Лекция 6. Программирование для мобильных устройств

Содержание лекции:

- Жизненный цикл Activity
- Способы добавления элементов

Преподаватель курса: Нефедов Денис Геннадьевич, к.т.н., доцент

Жизненный цикл Activity

Activity – активность – представляет каждый отдельный экран или страницу в приложении.
По умолчанию, если выбран тип проекта Empty Activity, в приложении уже имеется класс MainActivity (файл **MainActivity.java**)

```
package com.example.myapplication;
//директивы импорта
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity { //определение класса
    @Override
    protected void onCreate(Bundle savedInstanceState) {//единственный метод
        super.onCreate(savedInstanceState); //содержащий интерфейс приложения
        setContentView(R.layout.activity_main);
    }
}
```

Файл activity_main.xml - текстовый файл с разметкой xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Весь интерфейс представлен элементом-контейнером

```
<androidx.constraintlayout.widget.ConstraintLayout
```

ConstraintLayout позволяет расположить вложенные элементы в определенных местах экрана. Вначале элемента ConstraintLayout идет определение пространств имен XML. Каждое пространство имен задается следующим образом: xmlns:префикс="название_ресурса". Каждый ресурс или URI определяет некоторую функциональность, которая используется в приложении, например, предоставляют теги и атрибуты, которые необходимы для построения приложения.

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

И чтобы упростить работу с этими ресурсами, применяются префиксы.

```
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
```

Здесь определена ширина и высота контейнера (по содержащему контейнеру), и связь интерфейса с классом activity

Аналогично для текстового поля, где последние 4 атрибута приводят к расположению элемента по центру экрана

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

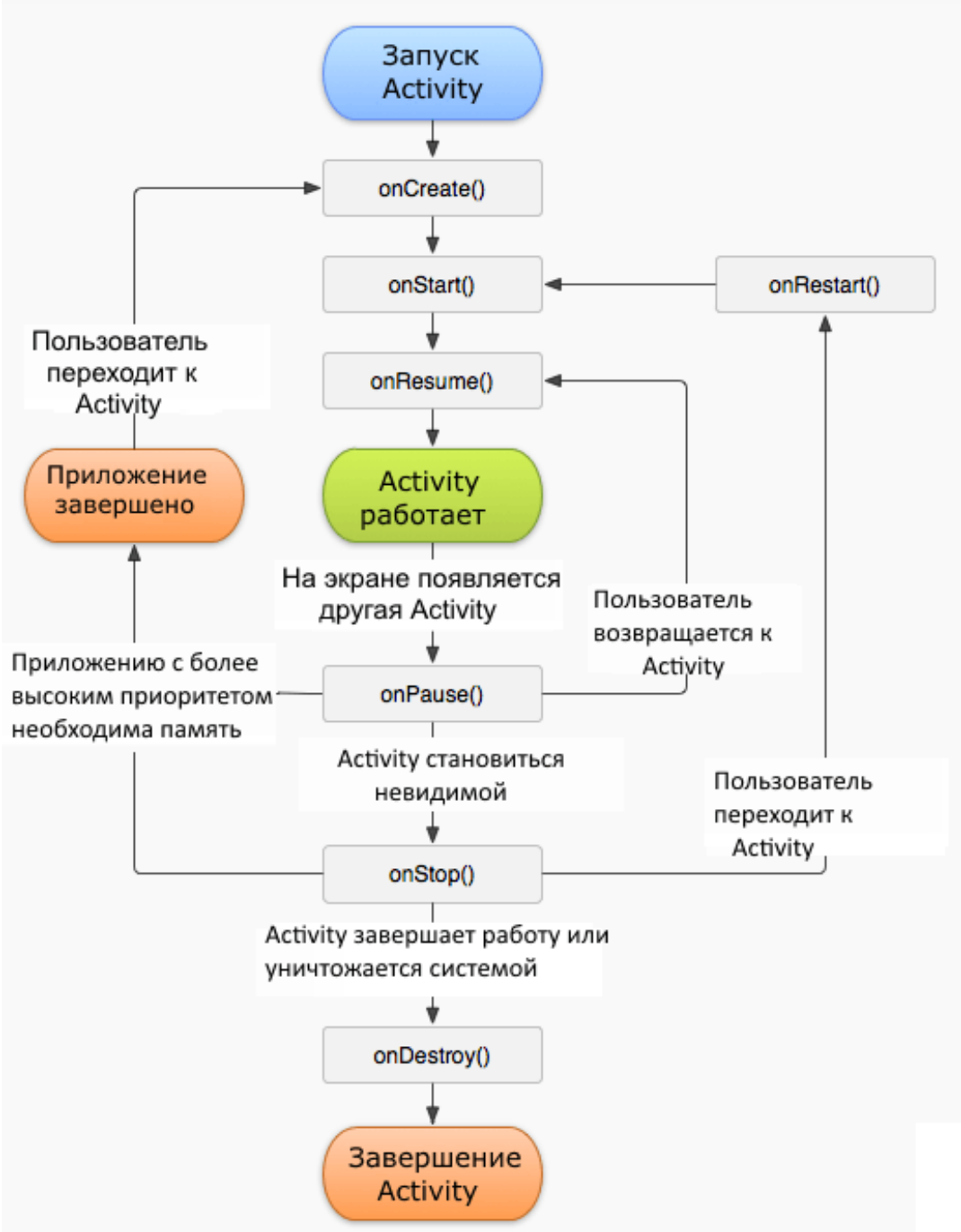
Для добавления новых активностей нажмем ПКМ в структуре проекта на папку, в котором находится класс MainActivity, и затем в контекстном меню выберем New->Activity->Empty Activity.
После этого в папке с MainActivity (по умолчанию) должен появиться файл с новой activity (xml и java)

Все объекты activity, которые есть в приложении, управляются системой в виде стека activity, который называется **back stack**. При запуске новой activity она помещается поверх стека и выводится на экран устройства, пока не появится новая activity. Когда текущая activity заканчивает свою работу (например, пользователь уходит из приложения), то она удаляется из стека, и возобновляет работу та activity, которая ранее была второй в стеке.

У Activity 6 основных **функций** жизненного цикла

onCreate(), onStart(), onResume(), onPause(), onStop(), и **onDestroy().**

Система вызывает данные функции при изменении состояния Activity.



1) onCreate()

Данная функция вызывается когда activity впервые создано. По окончанию создания activity, activity входит в состояние Created . В функции onCreate() , мы пишем код который нам нужно запустить только один раз за весь жизненный цикл activity. Данная функция принимает параметр savedInstanceState, это есть Bundle а именно это объект который содержит предыдущее состояние activity. Если activity не существовало ранее то значение объекта Bundle равно null.

2) onStart()

Когда Activity входит в состояние Started state, система вызывает данную функцию. Функция onStart() делает видимым Activity для пользователя, Activity выходит на передний план и становится интерактивным для пользователя. Например это отличное место для кода инициализации UI (User Interface) компонентов.

Функция onStart() выпллняется очень быстро, и как в случае с Created state, activity не находится в состоянии Started state. Как только функция вызвана activity сразу переходит в Resumed state, и система вызывает функцию onResume() .

3) onResume()

Это состояние в котором приложение взаимодействует с пользователем. Приложение находится в данном состоянии до тех пор, пока что либо не произойдет и оно уйдет с первого плана (например входящий звонок). Когда процесс прерван activity входит в состояние Paused state, и система вызывает функцию onPause().

Если activity возвращается в состояние Resumed state из состояния Paused state, система снова вызывает функцию onResume() . По этой причине мы должны инициализировать компоненты в функцииonResume() которые мы остановили в функции onPause(), а также инициализируем все компоненты которые нужно инициализировать в состоянии Resumed state.

4) onPause()

Система вызывает данный метод как только поймет что пользователь покидает приложение (это не означает что activity будет уничтожено); это означает что activity больше не находится на переднем плане (оно может быть видно например в multi-window mode). Используйте функцию onPause() для операций которые должны быть остановлены во время паузы пока Activity находится в состоянии Paused state, и ожидается скорое возобновление работы activity.

5) onStop()

Когда activity больше не видно на экране, оно входит в состояние Stopped state, и ситема вызывает функциюonStop() . Это происходит например когда новое activity перекрывает весь экран. а также данная функция onStop() может быть вызвана когда activity завершило.

В функции onStop() нам нужно освободить все ресурсы которые нам не нужны , когда приложение не видно на экране.

6) onDestroy()

Вызывается когда activity “разрушено” по причине:

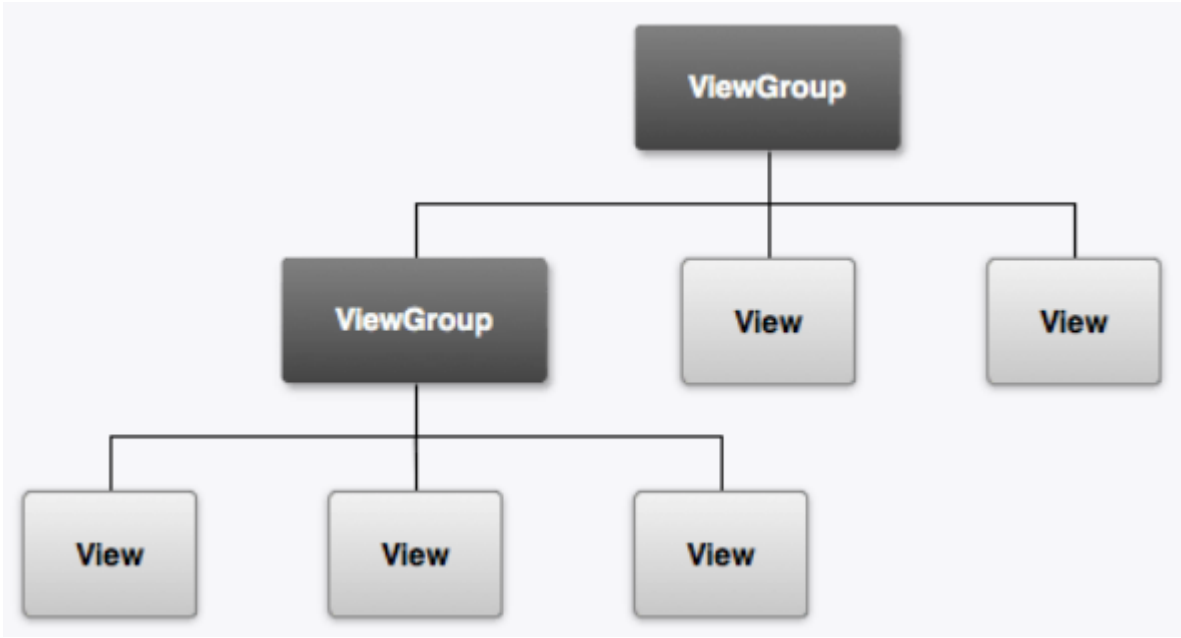
- Activity закончило свою работу (пользователь нажал на кнопку выхода из приложения или была запущена функция finish()).
- Система “разрушила” activity по причине поворота экрана или использования многооконного режима.

В функции onDestroy() должны быть освобождены все ресурсы которые не были освобождены в функции onStop().

Способы добавления элементов

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.д., через которые пользователь взаимодействует с программой:



При определении визуального у нас есть три стратегии:

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

1. В коде java.

Определим в классе **MainActivity** простейший интерфейс:

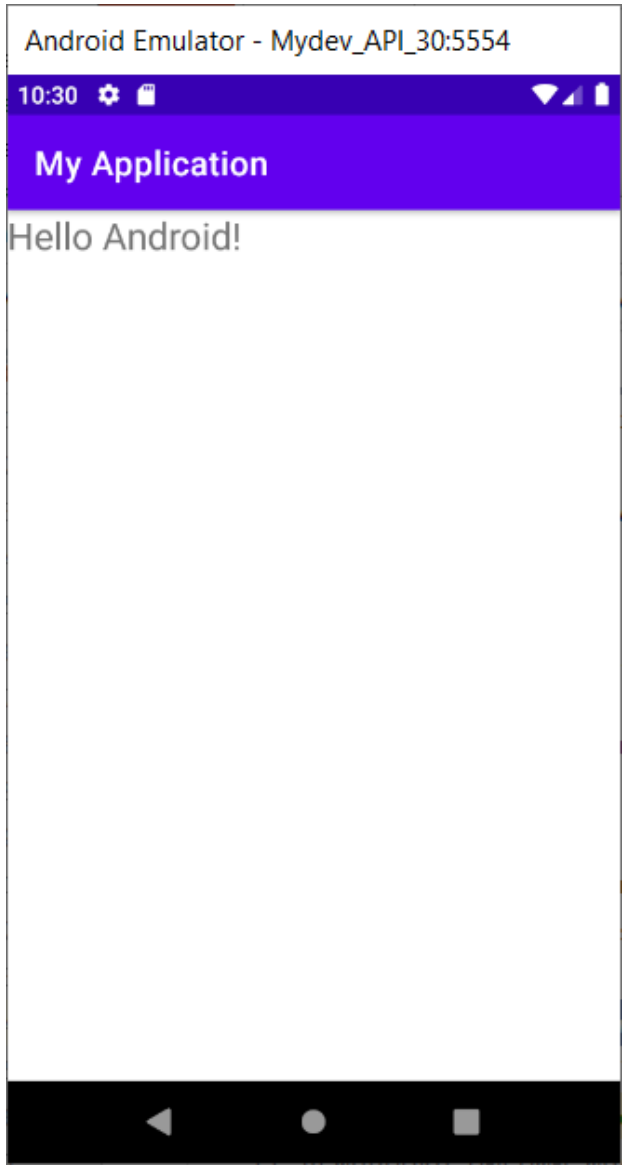
```
package com.example.myapplication;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView textView = new TextView(this); // создание TextView
        textView.setText("Hello Android!"); // установка текста в TextView
        textView.setTextSize(22); // установка высоты текста
        setContentView(textView); // установка визуального интерфейса для activity
    }
}
```

При создании виджетов в коде Java применяется их конструктор, в который передается контекст данного виджета, а точнее объект android.content.Context, в качестве которого выступает текущий класс MainActivity.

```
TextView textView = new TextView(this);
```

Здесь весь интерфейс представлен элементом TextView, который предназначен 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.



2. Описание в xml файле.

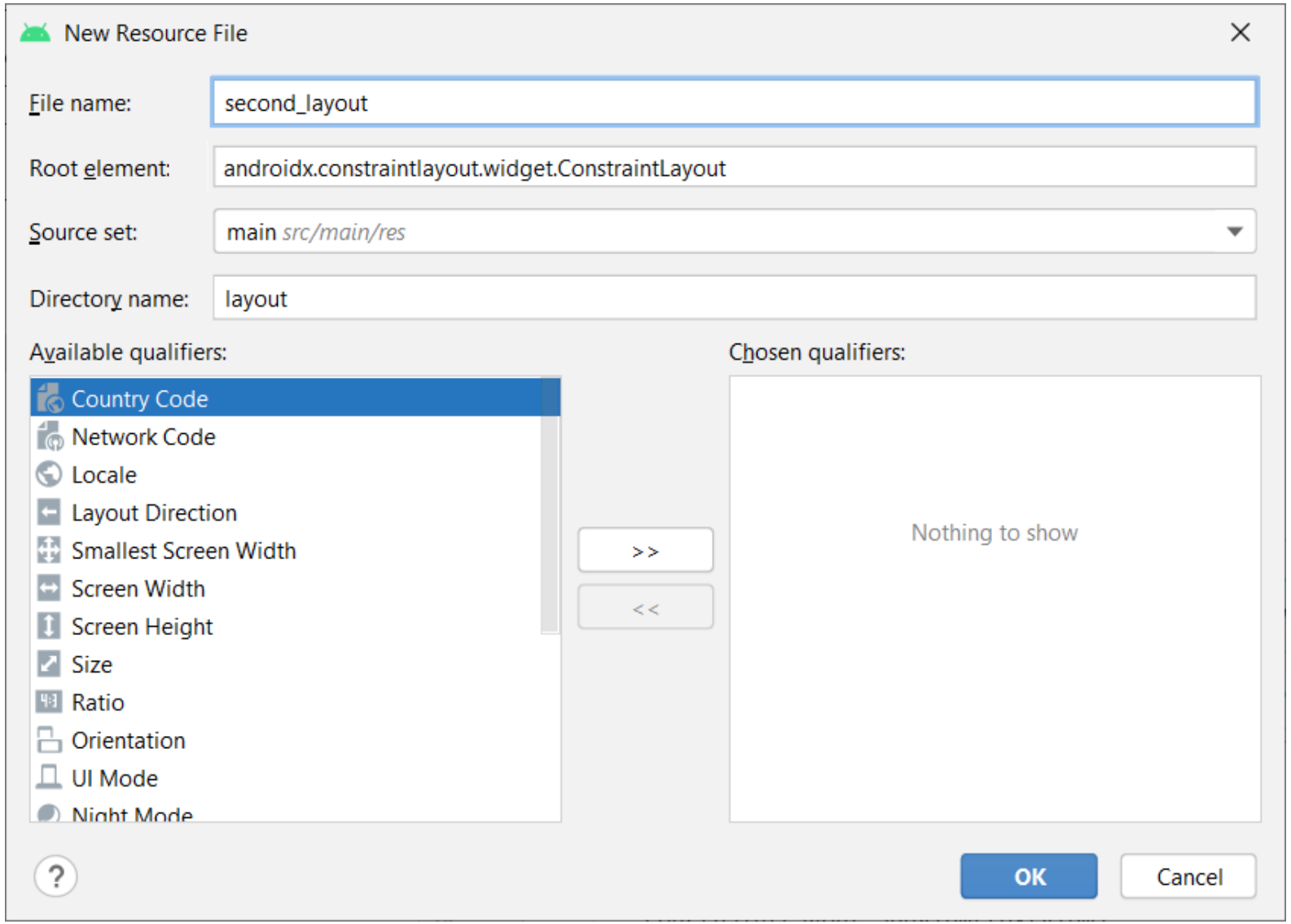
Как правило, для определения визуального интерфейса в проектах под Android используются специальные файлы xml. Эти файлы являются ресурсами разметки и хранят определение визуального интерфейса в виде кода XML. Объявление пользовательского интерфейса в файлах XML позволяет отделить интерфейс приложения от кода.

Файлы разметки графического интерфейса располагаются в проекте в каталоге

res/layout.

Как правило, каждый отдельный класс Activity использует свой файл layout. Либо для одного класса Activity может использоваться сразу несколько различных файлов layout.

К примеру, добавим в проект новый файл разметки интерфейса. Для этого нажмем на папку res/layout правой кнопкой мыши и в появившемся меню выберем пункт New -> Layout Resource File.



В поле Root element указывается корневой элемент. По умолчанию это androidx.constraintlayout.widget.ConstraintLayout.
Поле Source set указывает, куда помещать новый файл. По умолчанию это main - область проекта, с которой мы собственно работаем при разработке приложения.
Поле Directory main указывает папку в рамках каталога.

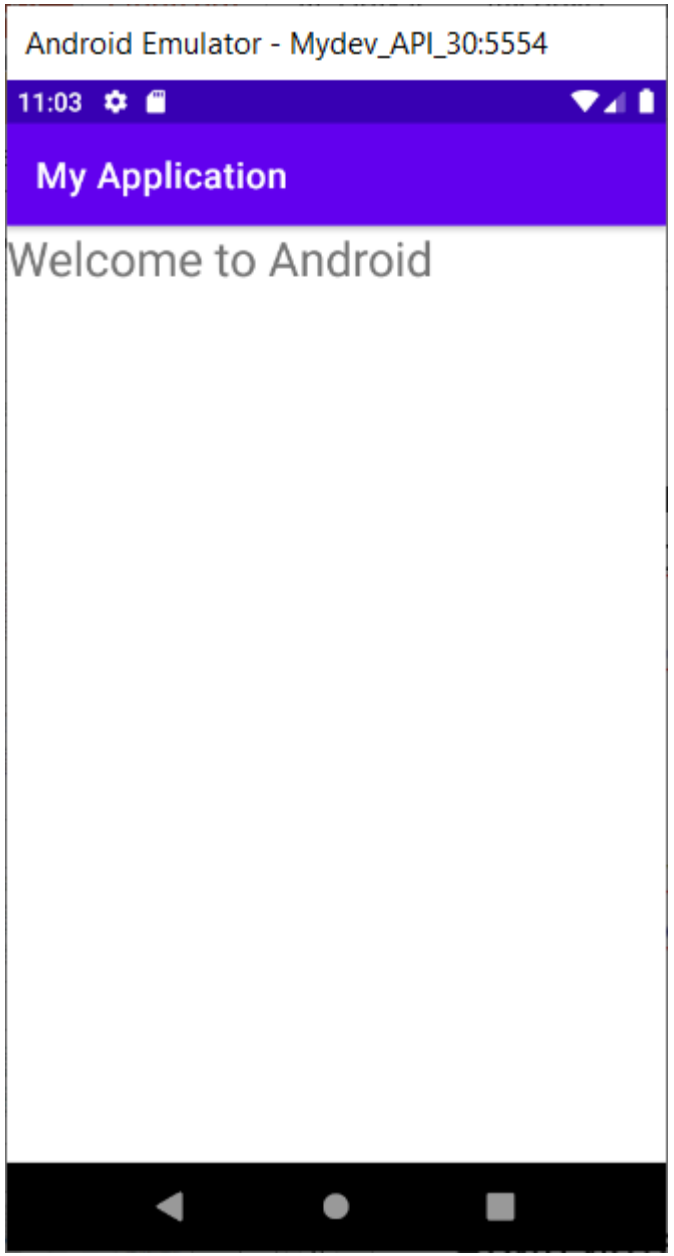
Откроем файл second_layout.xml и изменим его

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/header"
        android:text="Welcome to Android"
        android:textSize="26sp"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Здесь android:id - идентификатор элемента, через который мы сможем ссылаться на него в коде. В записи android:id="@+id/header" символ @ указывает XML-парсеру использовать оставшуюся часть строки атрибута как идентификатор. А знак + означает, что если для элемента не определен id со значением header, то его следует определить.

Применим этот файл в качестве определения графического интерфейса в классе MainActivity:

```
package com.example.myapplication;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.second_layout);
    }
}
```



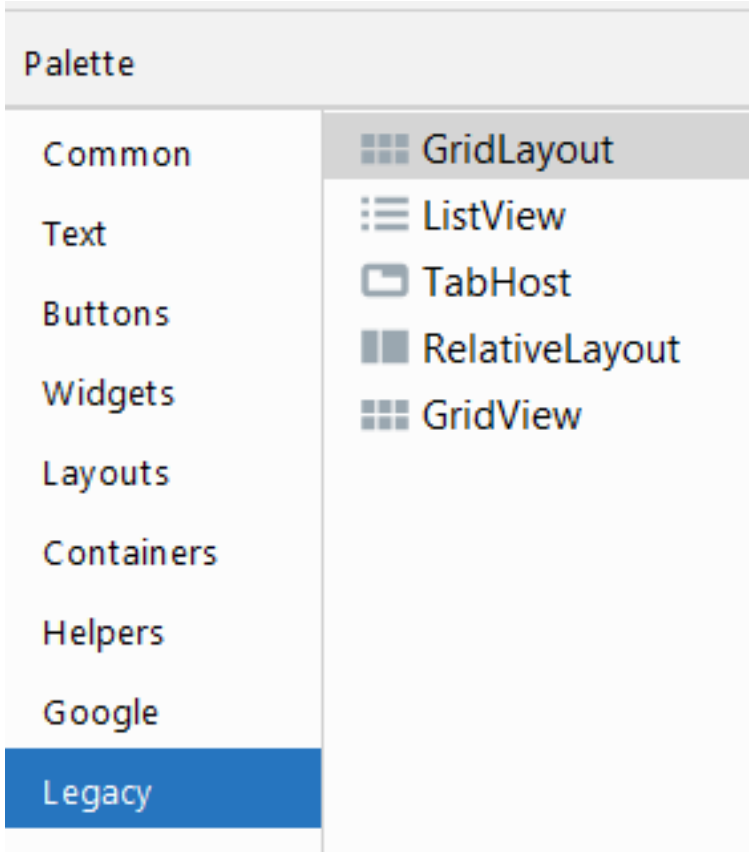
Файл интерфейса называется second_layout.xml, поэтому по умолчанию для него будет создаваться ресурс R.layout.second_layout. Соответственно, чтобы его использовать, мы передаем его в метода setContentView.

Элемент TextView имеет важный атрибут - id или идентификатор элемента. Он позволяет обращаться к элементу, который определен в файле xml, из кода Java. Например, перейдем к классу MainActivity и добавим код:

```
import android.widget.TextView;
…
        // получаем элемент textView
        TextView textView = (TextView) findViewById(R.id.header);
        // переустанавливаем у него текст
        textView.setText("Hello from Java!");
```

Для получения элементов по id класс Activity имеет метод findViewById(). В этот метод передается идентификатор ресурса в виде R.id.[идентификатор_элемента]. Этот метод возвращает объект View - объект базового класса для всех элементов, поэтому результат метода еще необходимо привести к типу TextView.
Далее мы можем что-то сделать с этим элементом, в данном случае изменяем его текст.
Причем что важно, получение элемента происходит после того, как в методе setContentView была установлена разметка, в которой этот визуальный элемент был определен.

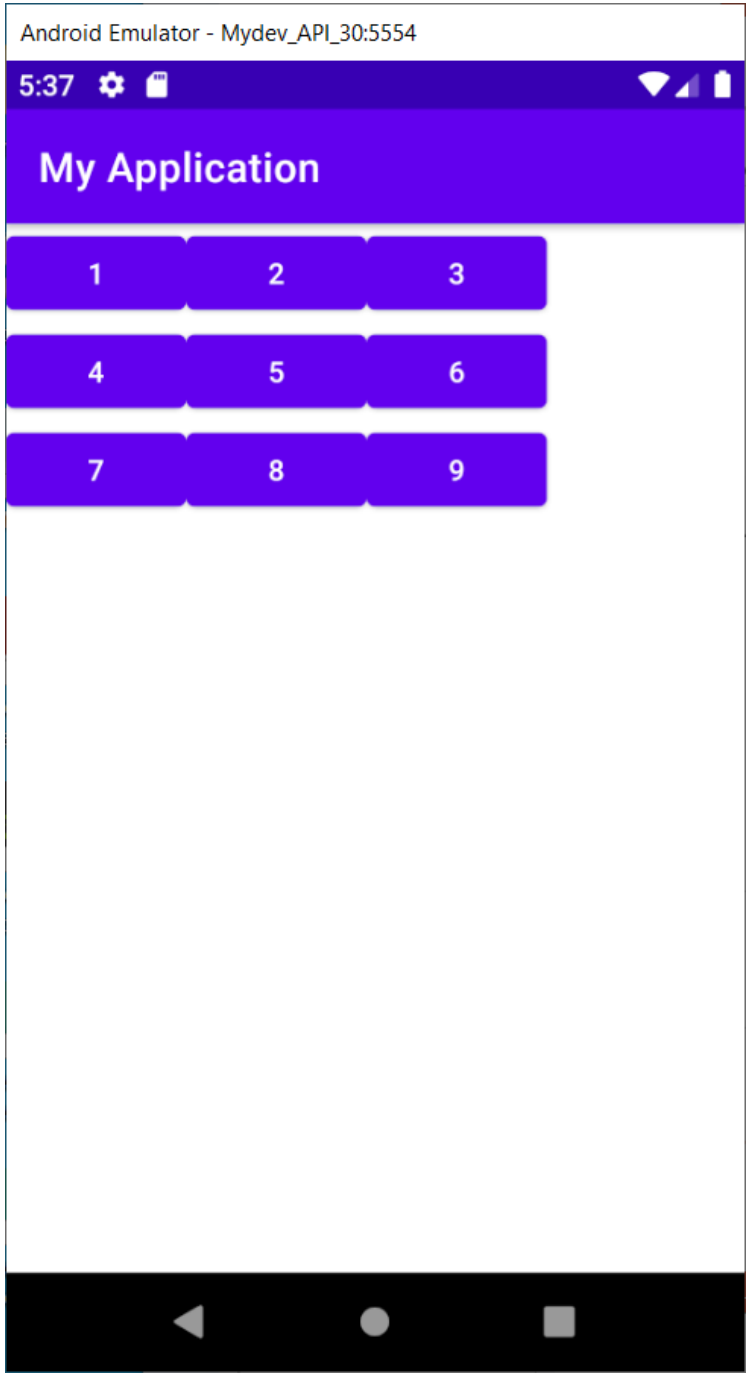
GridLayout представляет один из контейнеров, который позволяет создавать табличные представления. GridLayout состоит из коллекции строк, каждая из которых состоит из отдельных ячеек:



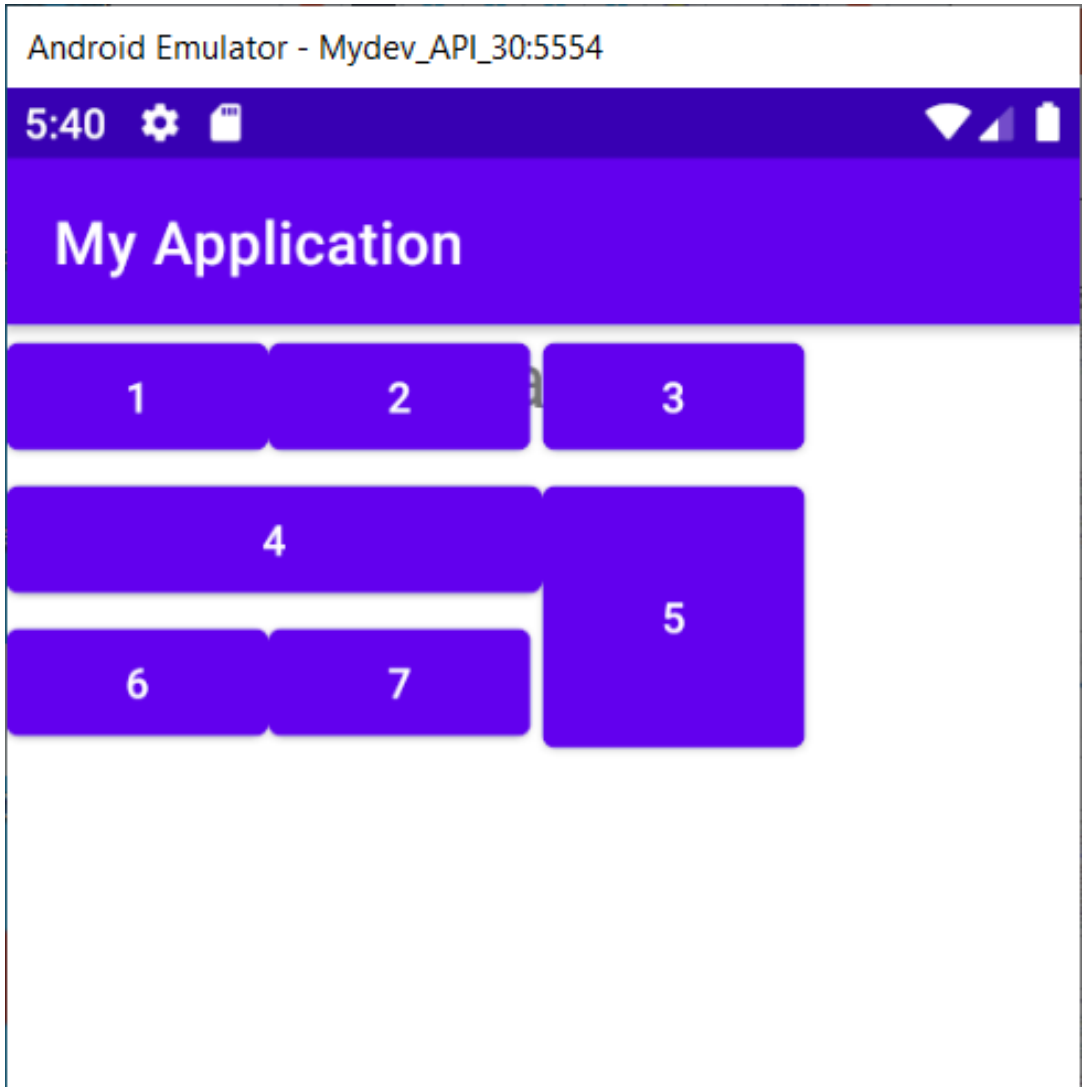

```
<GridLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="3"
    android:columnCount="3">

    <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>
```

При этом ширина столбцов устанавливается автоматически по ширине самого широкого элемента. Для явного задания размещения элементов в контейнеры можем применять следующие атрибуты:

- 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



```
<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"
        android:layout_column="0"
        android:layout_row="0" />
    <Button android:text="2"
        android:layout_column="1"
        android:layout_row="0"/>
    <Button android:text="3"
        android:layout_column="2"
        android:layout_row="0"  />
    <Button android:text="4"
        android:layout_width="180dp"
        android:layout_columnSpan="2"/>
    <Button android:text="5"
        android:layout_height="100dp"
        android:layout_rowSpan="2"/>
    <Button android:text="6" />
    <Button android:text="7"/>
</GridLayout>
```



Элемент EditText является подклассом класса TextView. Дополнительно к возможностям TextView у него есть атрибут android:hint. Он позволяет задать текст, который будет отображаться в качестве подсказки, если элемент EditText пуст. Кроме того, мы можем использовать атрибут android:inputType, который позволяет задать клавиатуру для ввода, например

- text: обычная клавиатура для ввода однострочного текста
- textMultiLine: многострочное текстовое поле
- textEmailAddress: обычная клавиатура, на которой присутствует символ @, ориентирована на ввод email
- textUri: обычная клавиатура, на которой присутствует символ /, ориентирована на ввод интернет-адресов
- textPassword: клавиатура для ввода пароля
- textCapWords: при вводе первый введенный символ слова представляет заглавную букву, остальные - строчные
- number: числовая клавиатура
- phone: клавиатура в стиле обычного телефона
- date: клавиатура для ввода даты
- time: клавиатура для ввода времени
- datetime: клавиатура для ввода даты и времени

```
<TextView
    android:id="@+id/textView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:textSize="34sp"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"/>
<EditText
    android:id="@+id/editText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Введите имя"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent" />
```

Предполагается, что введенные в EditText символы тут же будут отображаться в элементе TextView. И для этого также изменим код MainActivity:

```java
package com.example.myapplication;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        EditText editText = (EditText) findViewById(R.id.editText);
        editText.addTextChangedListener(new TextWatcher() {
            public void afterTextChanged(Editable s) {}
            public void beforeTextChanged(CharSequence s, int start,
                                          int count, int after) {
            }
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                TextView textView = (TextView) findViewById(R.id.textView);
                textView.setText(s);
            }
        });
    }
}
```

С помощью метода addTextChangedListener() здесь к элементу EditText добавляется слушатель ввода текста - объект TextWatcher. Для его использования нам надо реализовать три метода, но в реальности нам хватит реализации метода onTextChanged, который вызывается при изменении текста.
Введенный текст передается в этот метод в качестве параметра CharSequence. В самом методе просто передаем этот текст в элемент TextView.
В итоге при вводе в EditText все символы также будут отображаться в TextView: