ТЕХНОТРЕК

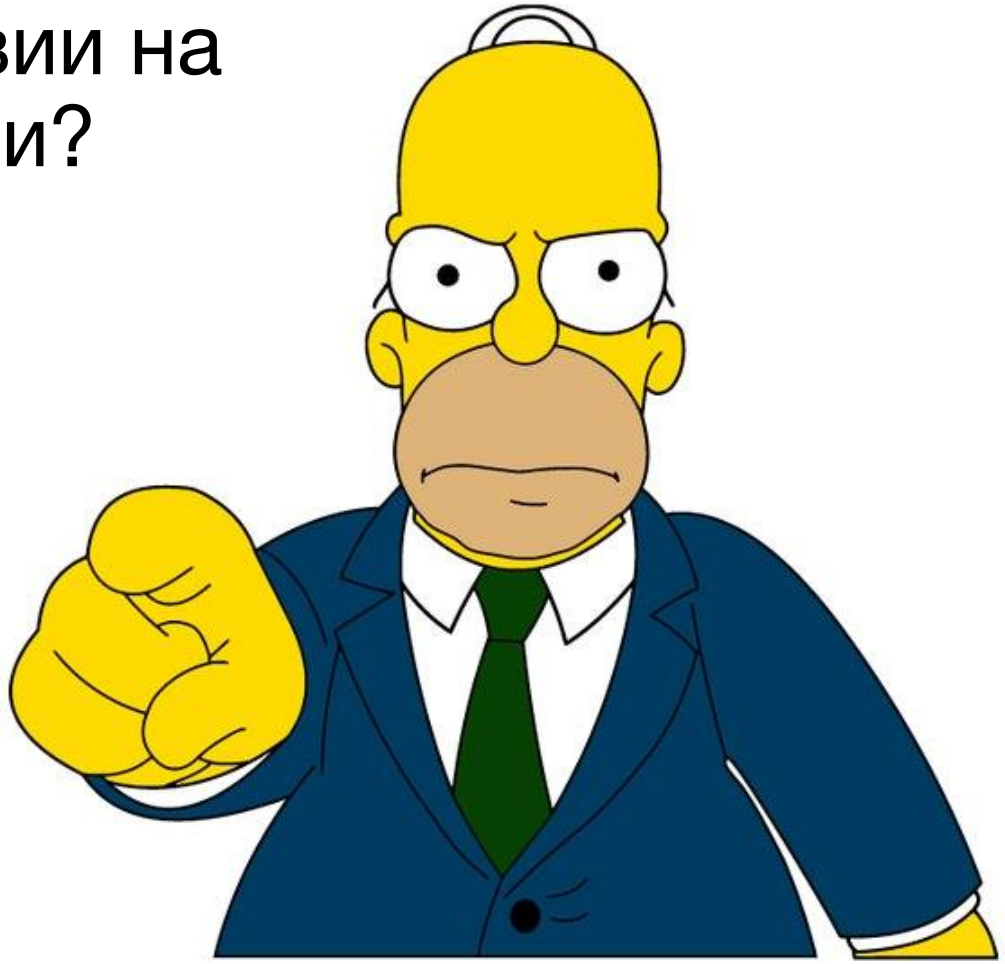
Занятие №2

Разработка приложений на Android

Юрий Береза
Кирилл Филимонов

Напоминание

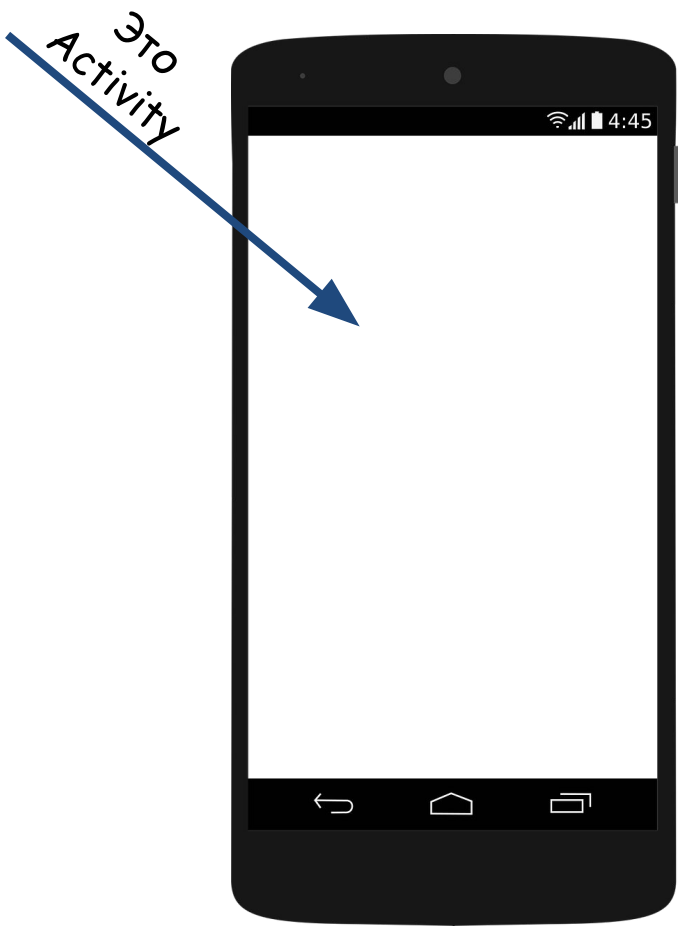
А ты отметился о присутствии на занятии?



Agenda

1. Activity и его жизненный цикл
2. Task и Back stack
3. Implicit intent и explicit intent
4. Фрагменты и их жизненный цикл
5. Support Library
6. GUI элементы
7. Layouts
8. Gravity

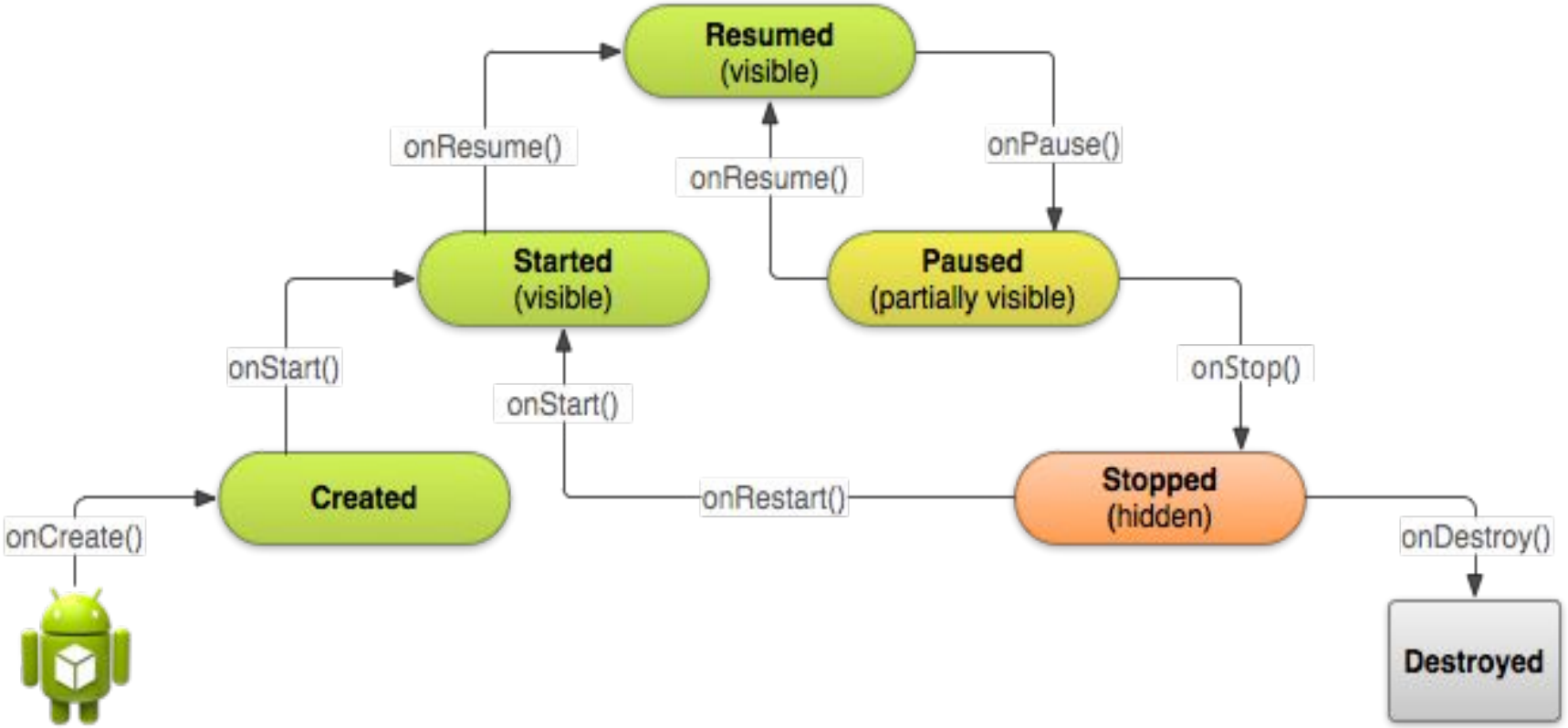
Activity



А это
объявление
Activity
в AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="ru.mail.sample" >

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

    <activity
      android:name="ru.mail.sample.MainActivity"
      android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

Activity и его жизненный цикл



Activity и его жизненный цикл

- `onCreate(Bundle saveInstanceState)`
  - Вызывается, когда создается activity
  - Получает сохраненное состояние (если оно есть)
  - Как конструктор
- `onResume()`
  - Вызывается перед тем, как activity станет видимым пользователю
- `onPause()`
  - Вызывается перед тем, как у другой activity вызовется onResume()
  - Здесь все завершающие операции
  - Не делать долгих операций!
- `onStop()`
  - Вызывается, когда activity уже не видима пользователю
- `onDestroy()`
  - Вызывается перед уничтожением activity

Activity - сохранение состояния

Когда уничтожается Activity:
- был вызван метод finish()
- пользователь нажал кнопку “Назад” на телефоне
- нехватка памяти, система освобождает ресурсы
- поворот экрана

- Сохранение - `onSaveInstanceState(Bundle state)`
  - Не забывайте вызвать метод `super.onSaveInstanceState(state);`
- Восстановление - `onRestoreInstanceState(Bundle state)`
  - Не забывайте вызвать метод `super.onRestoreInstanceState(state);`

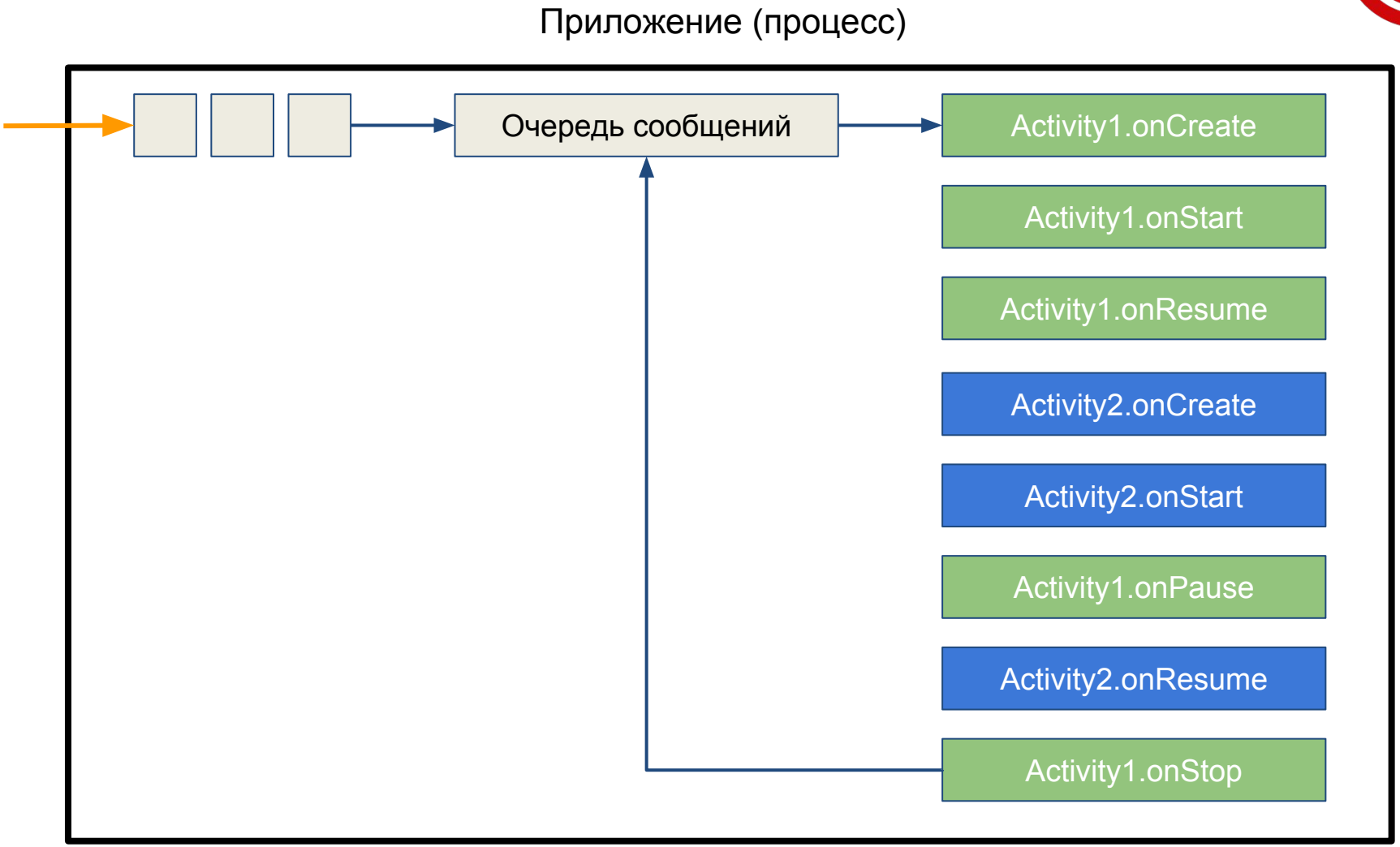
Важно! Если был вызван метод onSaveInstanceState, это еще не означает, что будет вызван метод onRestoreInstanceState

Activity - сохранение состояния

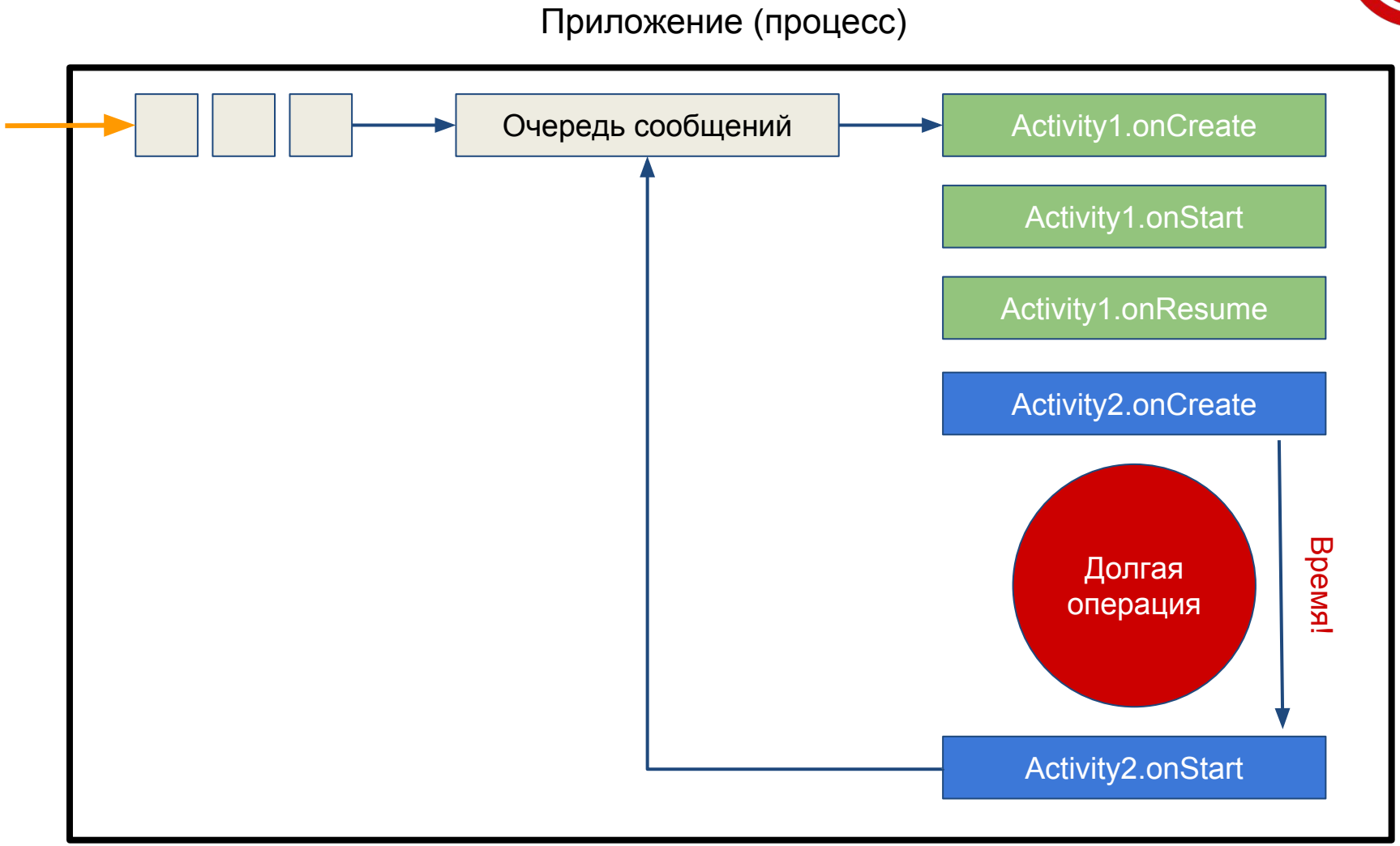
```
static final String STATE_SCORE = "playerScore";
static final String STATE_LEVEL = "playerLevel";

@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
   // Save the user's current game state
   savedInstanceState.putInt(STATE_SCORE, mCurrentScore);
   savedInstanceState.putInt(STATE_LEVEL, mCurrentLevel);

   // Always call the superclass so it can save the view hierarchy state
   super.onSaveInstanceState(savedInstanceState);
}
```

```
@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
    // Always call the superclass so it can restore the view hierarchy
    super.onRestoreInstanceState(savedInstanceState);

    // Restore state members from saved instance
    mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
    mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
}
```

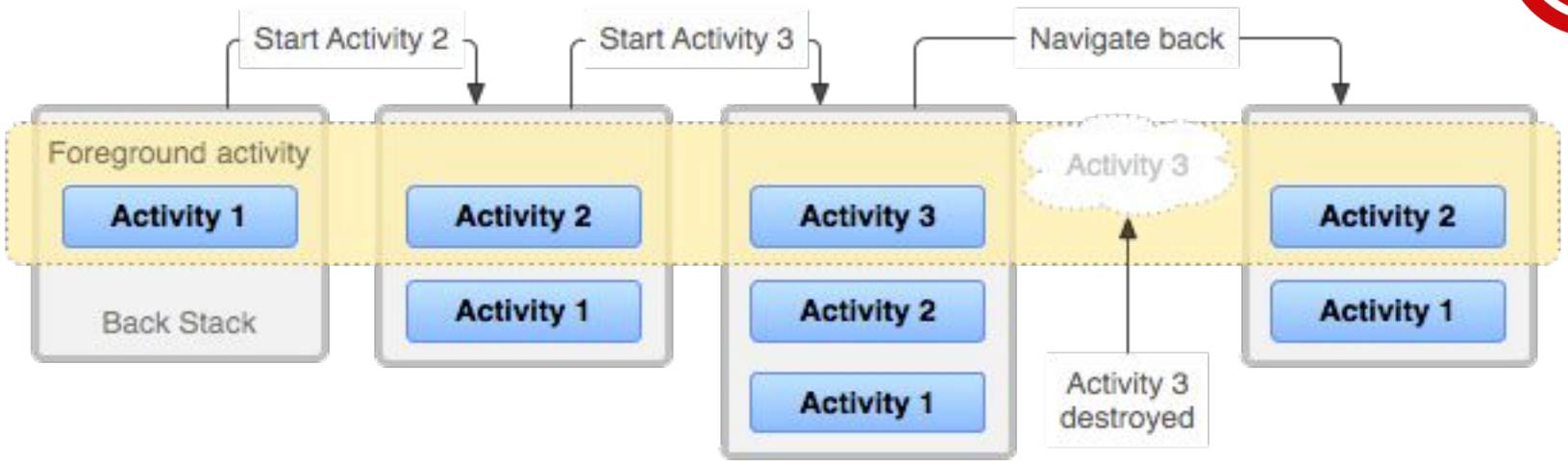
Приложение и Activity



Приложение и Activity



Task и Back Stack



- Task - набор Activity
- Каждый task имеет свой стек
- Одна и та же Activity может иметь сколько угодно экземпляров в стеке
- Новая activity пушится в стек, а у предыдущей вызывается onStop()
- По кнопке back верхняя activity достается из стека и уничтожается, а у activity под ней
- вызывается onResume()

Explicit Intent

- Адресуются конкретному компоненту (с помощью component name)
- Обычно используются для запуска внутренних компонентов

```
Intent intent = new Intent(this, WordsWriter.class);
startActivity(intent);
```

Implicit Intent

- Не имеют конкретного адресата
- Обычно используются для запуска компонентов сторонних приложений
- Система находит наиболее подходящие компоненты (или несколько)

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://mail.ru"));
startActivity(intent);
```

Получение результата от Activity

Activity 1 (получатель)

```
Intent intent = new Intent(this, WordsWriter.class);
//Используйте константы!!!
startActivityForResult(intent, 12345);

@Override
void onActivityResult(int code, int result, Intent data) {
    if (code == 12345 && result == RESULT_OK) {
        if (data.hasExtra("result")) { //что-то делаете }
    }
}
```

Activity 2 (отправитель)

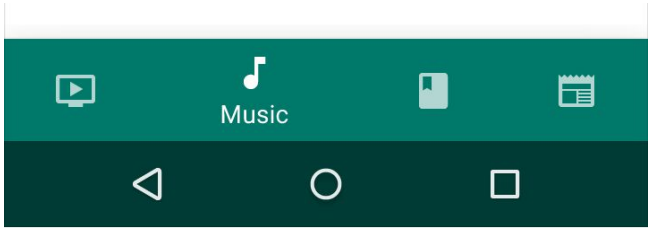
```
Intent data = new Intent();
data.putExtra("result", "111-пыщ-пыщ");
setResult(RESULT_OK, data);
```

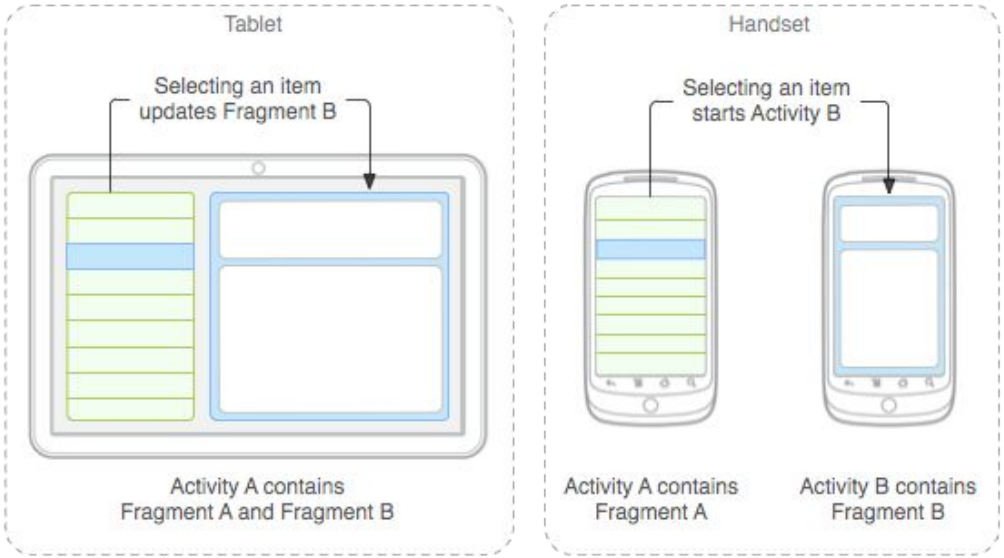
Фрагменты

- Что такое фрагмент
- Жизненный цикл
- Менеджер фрагментов
- Как использовать
- Как использовать Support Fragments

Что такое фрагменты

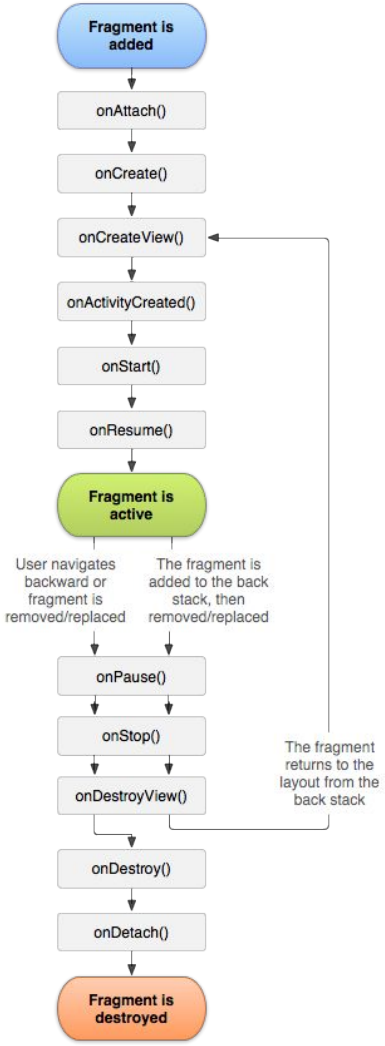
- Fragments – представляет собой часть пользовательского интерфейса в Activity
- Можно в одном Activity использовать несколько фрагментов эмулирую многопанельный интерфейс
- Жизненный цикл фрагмента привязан к циклу activity





Жизненный цикл фрагмента

- Код почти как в Activity
- onAttach – фрагмент прикрепили к activity
- onCreate - аналог аналогичной функци у activity, но пока нет доступа к элементам UI
- onCreateView – тут создается View фрагмента который надо отдать системе
- onActivityCreate – фрагмент создан есть доступ к UI
- onDetach – фрагмент откреплися от activity



Как работать с фрагментами

1. В ресурсы layout можно добавить атрибут <fragment>

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.news.ArticleListFragment"
              android:id="@+id/list"
```

- 2. Создать класс унаследованный от Fragment реализовать нужные методы
- 3. Если нужно динамически менять фрагменты или как иначе управлять ими, надо использовать FragmentManager и FragmentTransaction

```
// Create new fragment and transaction
Fragment newFragment = new ExampleFragment();
FragmentTransaction transaction = getFragmentManager().beginTransaction();

// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack
transaction.replace(R.id.fragment_container, newFragment);
transaction.addToBackStack(null);

// Commit the transaction
transaction.commit();
```

FragmentManager

- Позволяет управлять фрагментами и историей фрагментов
- Позволяет добавить, удалить или заменить фрагмент в Activity
- Позволяет работать со стеком транзакций фрагментов
- Вся работа идет через FragmentTransaction
- Для того чтобы действие свершилось надо вызвать commit у транзакции.

Фрагменты в Support Library

- Для того чтобы использовать фрагменты из Support Library надо Activity наследовать от AppCompatActivity
- Чтобы получить FragmentManager надо у activity вызвать getSupportFragmentManager
- Нельзя использовать одновременно фрагменты из support library и обычные
- Во всем остальном код одинаков

Подробнее о Support Library

- Позволяет на старых версиях Android использовать новый API
- Делает код проще без ветвления по версиям устройств
- Делает приложение единым для всех версий Android
- Экономит много времени программиста
- Легко добавляется в приложение в виде готовых библиотек

Поддержка разных версий API

```
@SuppressWarnings("deprecation")
public class ViewCompat {

    public static void postOnAnimation(View view, Runnable runnable) {
        if (VERSION.SDK_INT >= VERSION_CODES.JELLY_BEAN) {
            SDK16.postOnAnimation(view, runnable);
        } else {
            view.postDelayed(runnable, 16);
        }
    }

    public static void setBackground(View view, Drawable background) {
        if (VERSION.SDK_INT >= VERSION_CODES.JELLY_BEAN) {
            SDK16.setBackground(view, background);
        } else {
            view.setBackgroundDrawable(background);
        }
    }

    @TargetApi(16)
    static class SDK16 {

        public static void postOnAnimation(View view, Runnable runnable) {
            view.postOnAnimation(runnable);
        }

        public static void setBackground(View view, Drawable background) {
            view.setBackground(background);
        }

    }

}
```

Support Library что и как

- Легко встраивать в проект просто одна строчка в скрипте сборки
- Сейчас есть несколько версий которые можно использовать параллельно
- v4 разработана для использования начиная с Android 1.6 (API level 4)
- v7 разработана для использования начиная с Android 2.6(API level 7)
- v8 и v13 и v17 соответственно
- Multidex Support Library - позволяет обойти ограничение 65к методов в одном dex файле

Support Library v4

- Fragments - фрагменты
- NotificationCompat – продвинутые нотификации
- ViewPager - контейнер где можно «свайпаться» между элементами
- Loader - класс для асинхронной загрузки данных
- FileProvider – предоставляет поддержку для шаринга частных файлов приложения между другими приложениями

Support Library v7

- v7 appcompat library
  - ActionBar – элемент интерфейса для навигации и пр
  - ActionBarActivity – Activty со встроенным ActionBar
- v7 cardview library
  - CardView – карточка
- v7 gridlayout library
  - GridLayout
- v7 mediarouter library
  - MediaRouter, MediaRouteProvider
- v7 palette library
  - Palette
- v7 recyclerview library
  - RecyclerView

Support Library v8 и v13

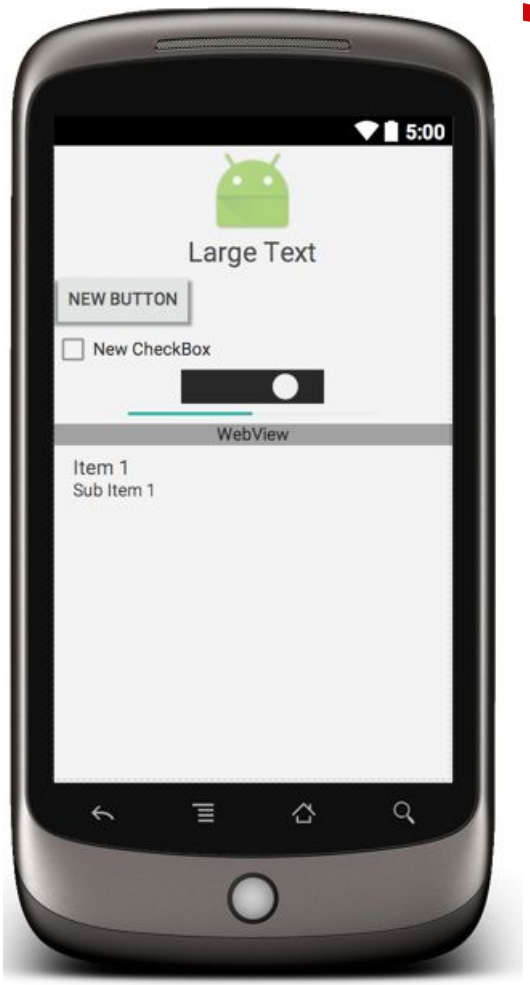
- v8
  - RenderScript - просто чтобы знать этот термин
- v13 реализация фрагментов для Android API level 13
  - FragmentCompat
  - FragmentPagerAdapter
  - FragmentTabHost

Support Library v17

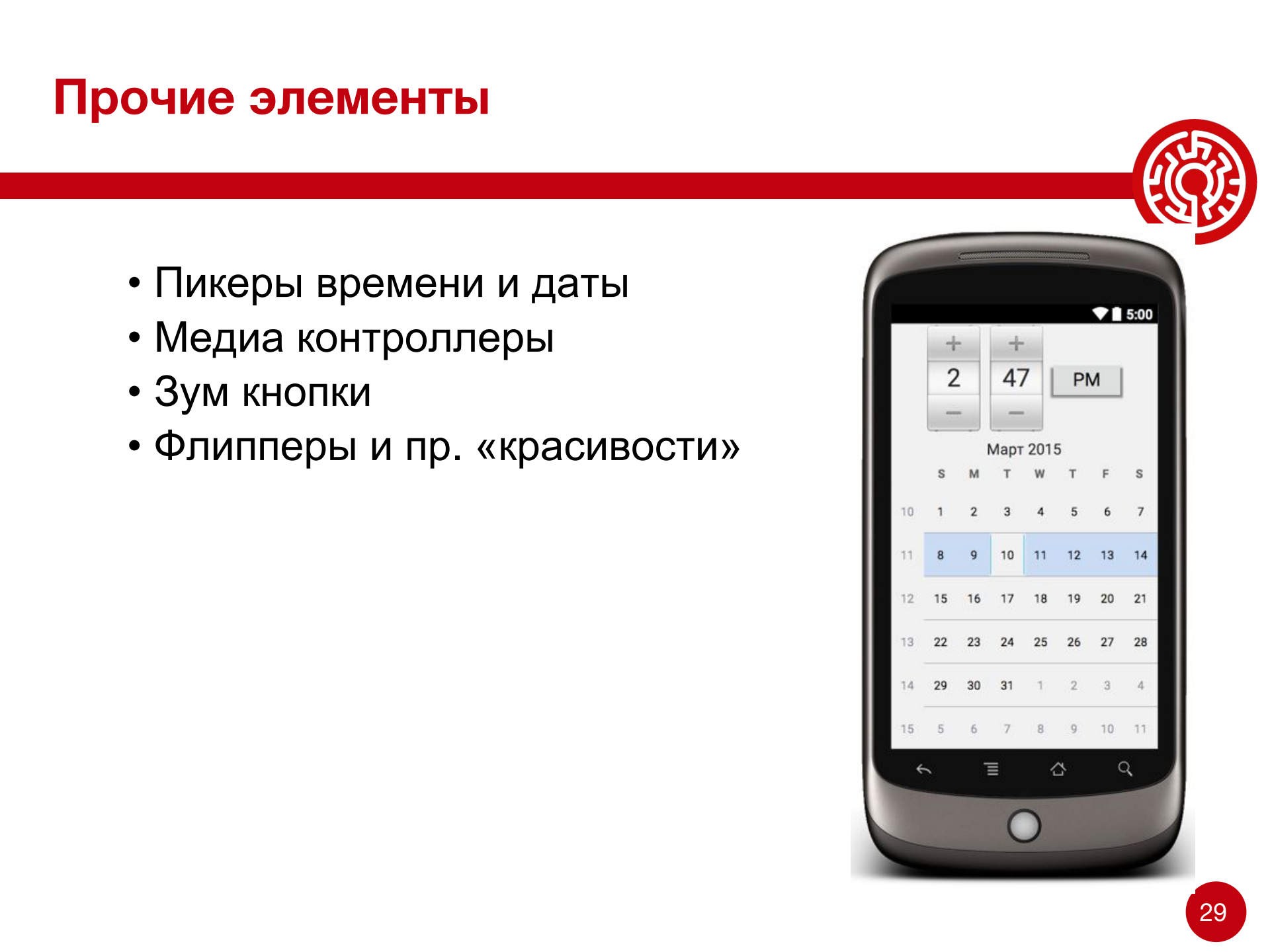
- v17 Leanback Library
  - Поддержка TV устройств
  - BrowseFragment – каталог медиа контента
  - DetailsFragment – фрагмент для отображения детальной информации
  - PlaybackOverlayFragment – фрагмент для отображения медиа контроллера
  - SearchFragment – фрагмент для поиска

Простые UI элементы

- View
- Image
- Button
- ImageButton
- TextView
- RadioButton
- CheckBox
- Switch
- ProgressBar
- WebView
- Spinner



Прочие элементы



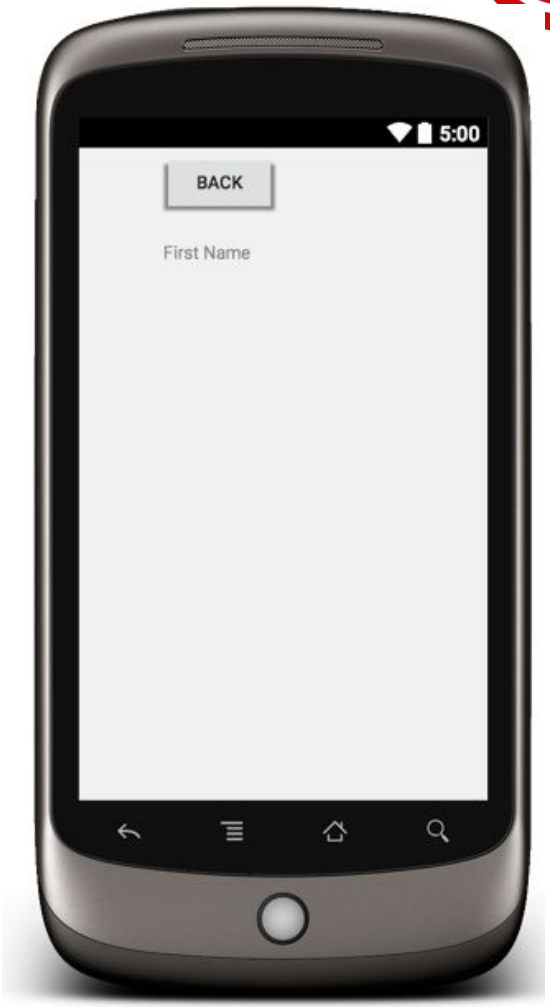
- Пикеры времени и даты
- Медиа контроллеры
- Зум кнопки
- Флипперы и пр. «красивости»

Разбираем GUI layoutы

- AbsoluteLayout (deprecated)
- FrameLayout
- LinearLayout
- RelatedLayout
- TableLayout

AbsoluteLayout

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
<Button
    android:id="@+id/backbutton"
    android:text="Back"
    android:layout_x="100px"
    android:layout_y="5px"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<TextView
    android:layout_x="100px"
    android:layout_y="110px"
    android:text="First Name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</AbsoluteLayout>
```



- каждый элемент верстки будет иметь абсолютную позицию относительно верхнего левого угла экрана, задаваемую с помощью координат x и y.
- верхний левый угол экрана при AbsoluteLayout имеет координаты x = 0, y = 0.
- позиция указывается в атрибутах элемента android:layout_x и android:layout_y.

FrameLayout

- тип верстки, внутри которого может отображаться только один элемент в строке.
- если внутри FrameLayout вы поместите несколько элементов, то следующий будет отображаться поверх предыдущего.

```
<FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <ImageView
        android:src="@drawable/homer"
        android:scaleType="fitCenter"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent"/>
    <TextView
        android:text="Hello World"
        android:textSize="36sp"
        android:textColor="#000000"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent"
        android:gravity="center"/>
</FrameLayout>
```

TableLayout

- Табличная верстка.
- Организует элементы в строки и столбцы таблицы.
- Для организации строк служит таг <TableRow>, а количество столбцов определяется максимальным количеством элементов внутри одного из <TableRow>.
- В случае, если элемент должен занимать несколько ячеек, используется атрибут android:layout_span.
- По умолчанию <TableRow> организует строки таблицы, если мы хотим организовывать не строки, а столбцы, то нужно использовать атрибут android:layout_column

RelativeLayout

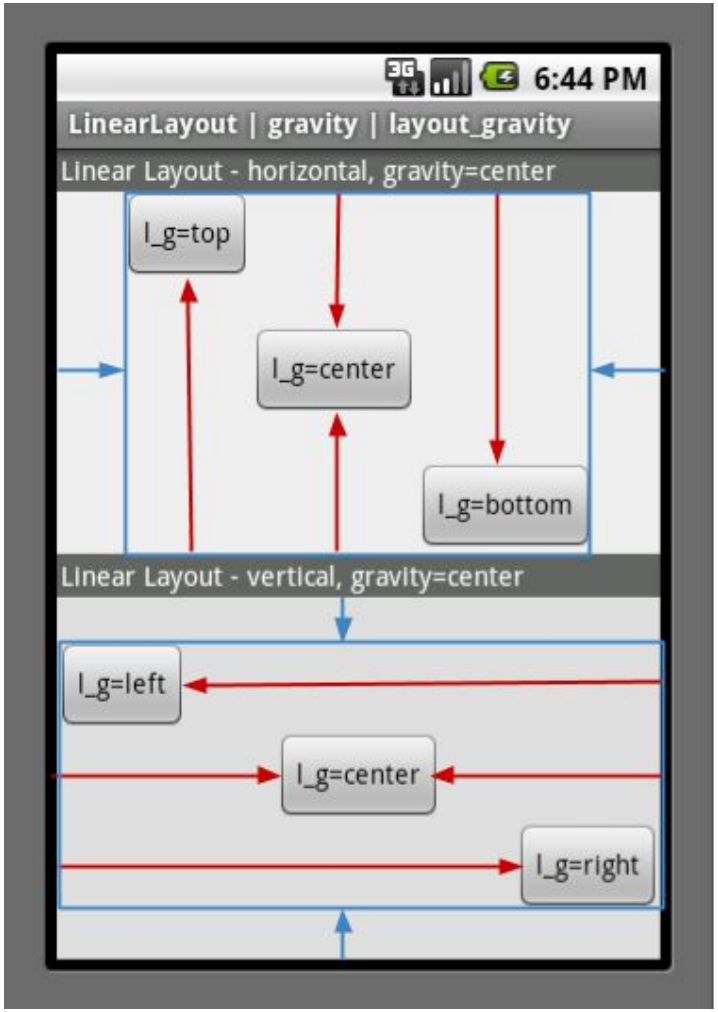
- Тип верстки, при котором позиционирование элементов происходит относительно друг друга и относительно главного контейнера.
- За то, каким образом будут позиционироваться элементы, отвечают следующие группы атрибутов:
  - Атрибуты позиционирования относительно контейнера
  - Атрибуты позиционирования относительно других элементов.
  - Выравнивание относительно других элементов.

LinearLayout

- тип верстки, при котором область верстки делится на строки, и в каждую строку помещается один элемент.
- разбиение может быть вертикальное или горизонтальное, тип разбиения указывается в атрибуте LinearLayout android:orientation.


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

    <Button
        android:id="@+id/button1"
        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" />
</LinearLayout>
```

Немного о гравитации



layout_gravity

gravity

- android:gravity – расположение контента внутри контейнера
- android:layout_gravity – расположение относительно родителя

Попробуем создать тестовое приложение

- Сделаем тестовое приложение, в котором есть по каждому виду слоя и каждому элементу

- Можно взять готовое приложение
- https://bitbucket.org/ybereza/technoparklection2

ТЕХНОТРЕК

Спасибо за внимание!

Юрий Береза – ybereza@gmail.com

Кирилл Филимонов - kirill.filimonov@gmail.com