

Яндекс



App Components

Base UI

Activity Lifecycle

Fragments

Станислав Куликов

# Основные компоненты приложения





# Основные компоненты приложения

- › Activity
- › Broadcast Receiver
- › Content Provider
- › Service

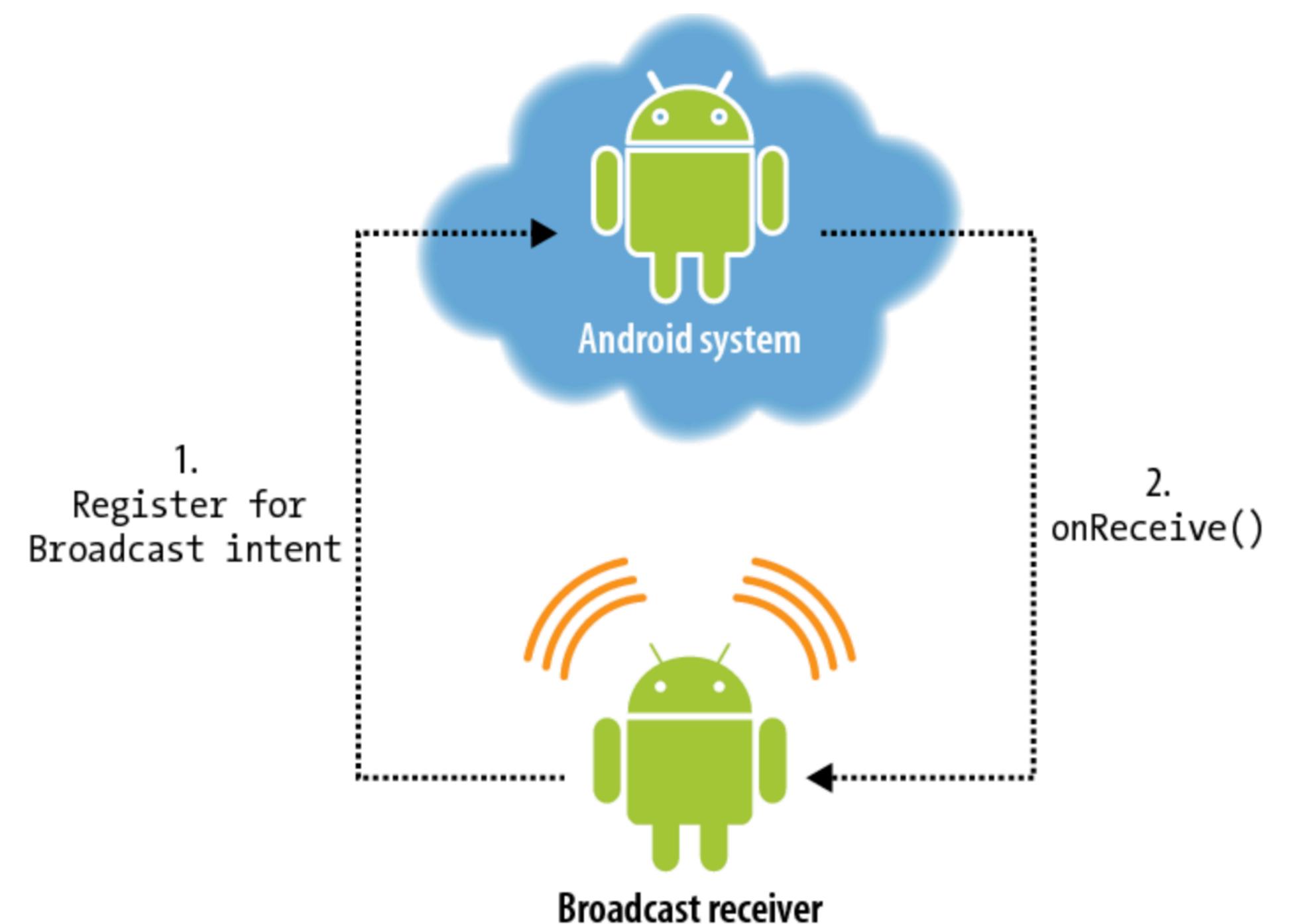
# Activity

Экран приложения, с которым пользователи могут взаимодействовать для выполнения каких-либо действий



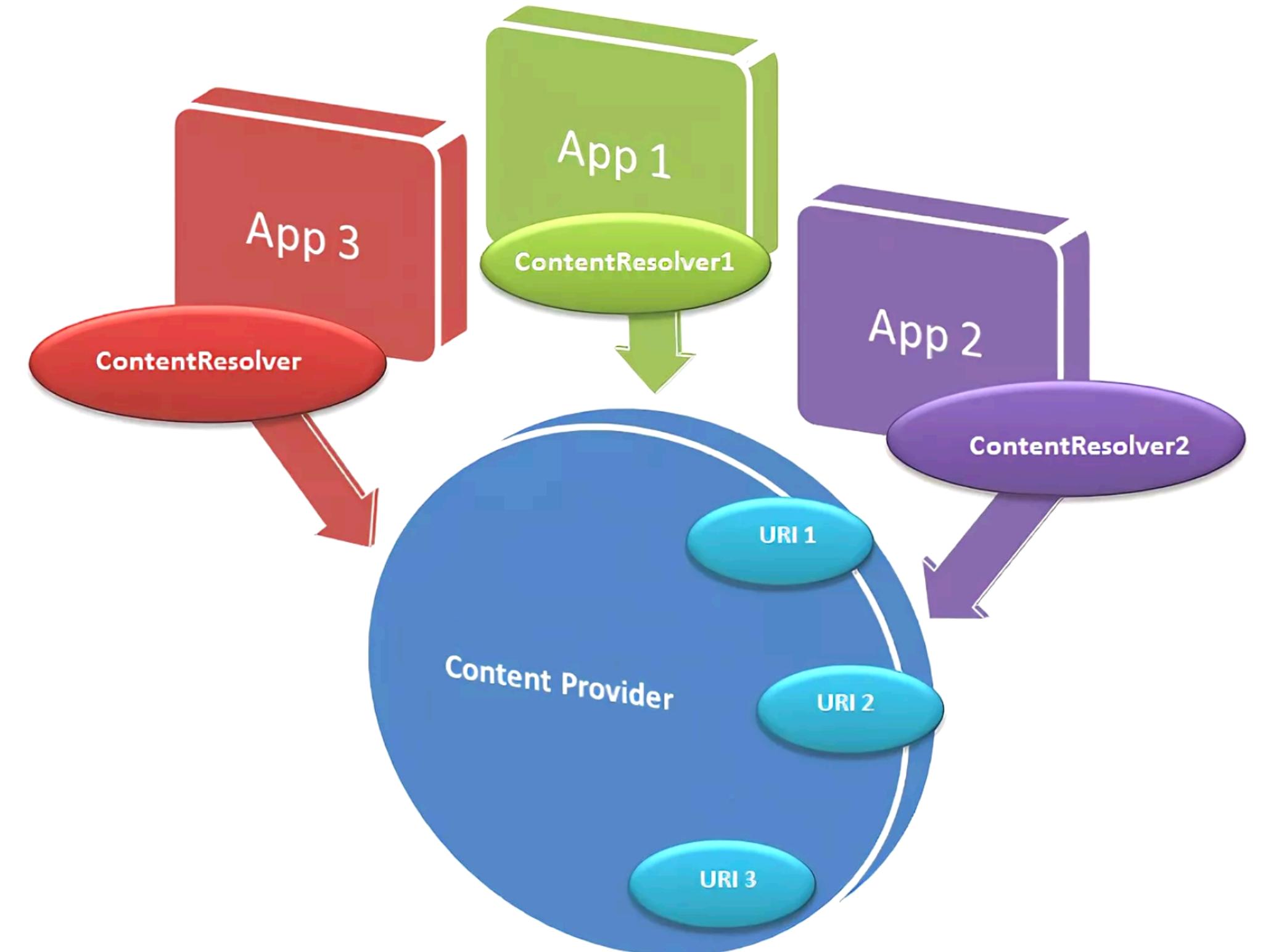
# Broadcast Receiver

Компонент, позволяющий реагировать на события, отправляемые системой



# Content Provider

| Предназначен для доступа к данным



# Service

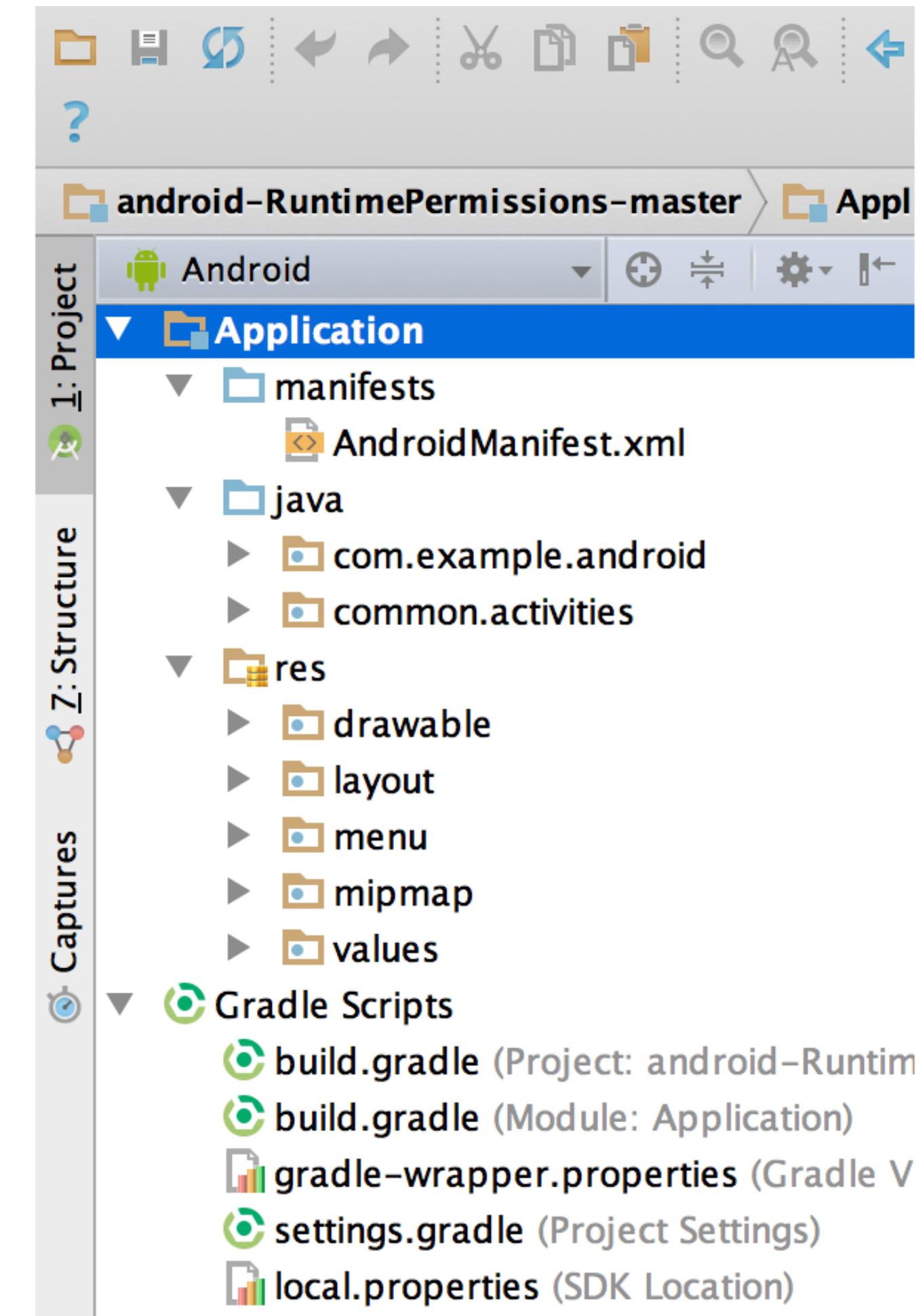
Предназначен для выполнения длительных операций в фоне

# Android Manifest



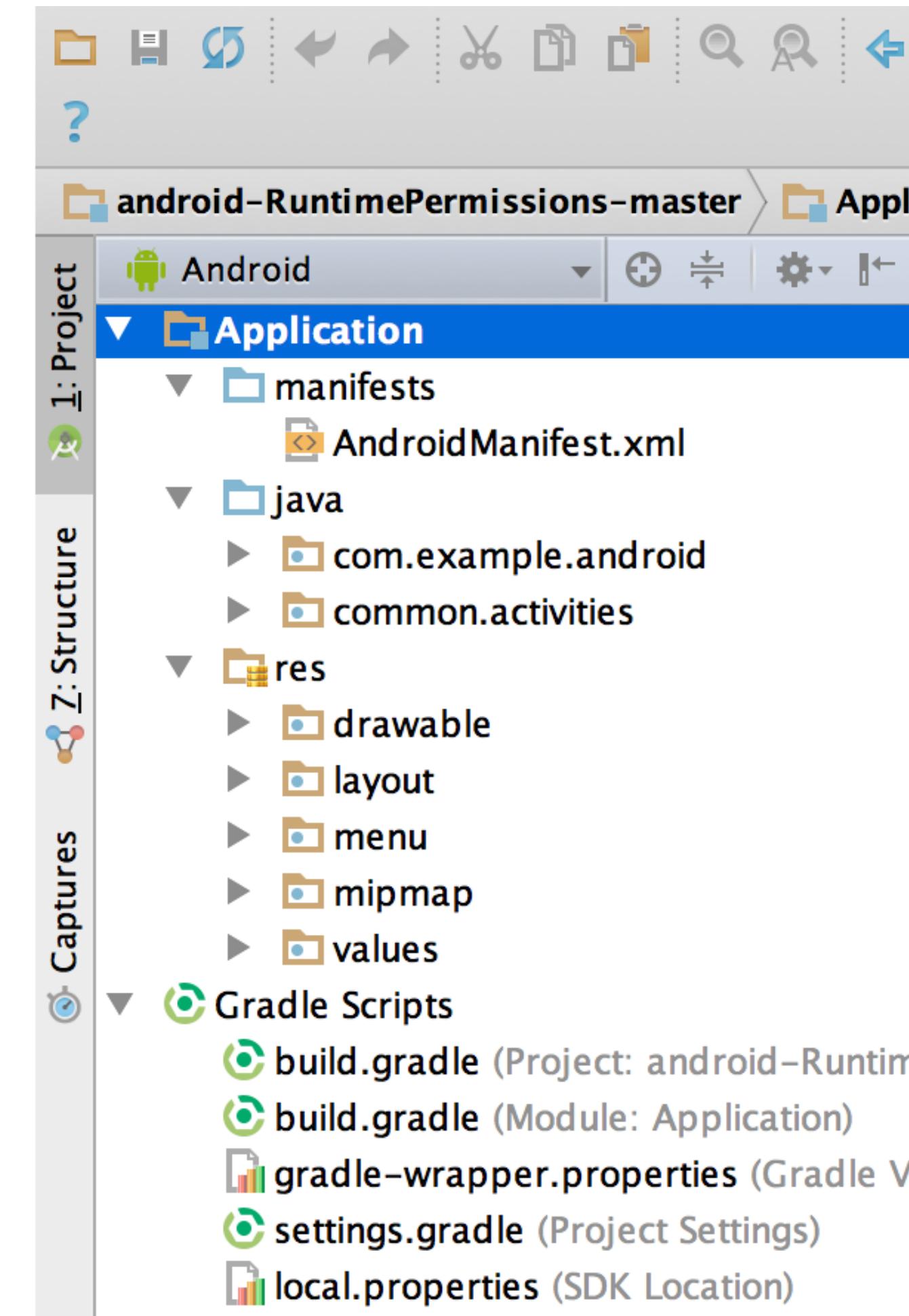
# Android Manifest

AndroidManifest.xml  
предоставляет системе основную  
информацию о приложении.



# Android Manifest

- › Информация о приложении
- › Объявление компонентов
- › Объявление требований приложения



# Android Manifest

**<permission>** - описывает права, которыми должно обладать другое приложения для получения доступа к вашему приложению.

**<uses-permission>** - описывает права, которые вашему приложению должны быть предоставлены системой для его нормального функционирования.

Имеет атрибут **android:name**, определяющий имя разрешения.  
Наиболее распространенные разрешения:

**INTERNET** - доступ к интернету

**READ\_CONTACTS** - чтение (но не запись) данных из адресной книги пользователя

**WRITE\_CONTACTS** - запись (но не чтение) данных из адресной книги пользователя

**RECEIVE\_SMS** - обработка входящих SMS

**ACCESS\_COARSE\_LOCATION** - использование приблизительного определения местонахождения при помощи вышек сотовой связи или точек доступа Wi-Fi

**ACCESS\_FINE\_LOCATION** - точное определение местонахождения при помощи GPS

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest>
```

```
    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />
```

```
<application>
```

...

```
</application>
```

```
</manifest>
```

# Android Manifest

**<uses-sdk>** - объявляет совместимость приложения с указанной версией Android SDK.

**<uses-configuration>** - указывает требуемую для приложения аппаратную и программную конфигурацию мобильного устройства (reqNavigation).

**<uses-feature>** - объявляет определенную функциональность, требующуюся для работы приложения (camera).

**<supports-screens>** - определяет разрешение экрана, требуемое для функционирования приложения.

**<application>** - содержит описание всех компонентов приложения.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest>
```

```
    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />
```

```
    <application>
```

```
    ...
```

```
    </application>
```

```
</manifest>
```

# Android Manifest

**<intent-filter>** - определяет типы интентов, на которые компонент может реагировать.

**<action>** - определяет действие, на которое Android компонент может реагировать;

**<category>** - определяет категорию интентов, которые компонент может обработать;

**<data>** - определяет тип данных, которые компонент может обработать.

Intent — это совокупность информации, описывающей требуемое действие

```
<application>
    <activity>
        <intent-filter>
            <action />
            <category />
            <data />
        </intent-filter>
        <meta-data />
    </activity>

    <activity-alias>
        <intent-filter> . . . </intent-filter>
        <meta-data />
    </activity-alias>

    <service>
        <intent-filter> . . . </intent-filter>
        <meta-data />
    </service>

    <receiver>
        <intent-filter> . . . </intent-filter>
        <meta-data />
    </receiver>

    <provider>
        <grant-uri-permission />
        <meta-data />
        <path-permission />
    </provider>

    <uses-library />
</application>
```

# Android Layout



# Android Layout

| Определяет визуальную структуру пользовательского интерфейса

- › FrameLayout
- › LinearLayout
- › RelativeLayout
- › ConstraintLayout

# FrameLayout

Предназначен для блокировки области на экране для отображения единственного дочернего элемента.

FrameLayout можно заполнить несколькими дочерними элементами, но в таком случае каждый из них будет попадать в стек и накладываться друг на друга.

Причём последний добавленный элемент будет находиться сверху.

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

    <View
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:layout_gravity="center"
        android:background="#ddd"/>

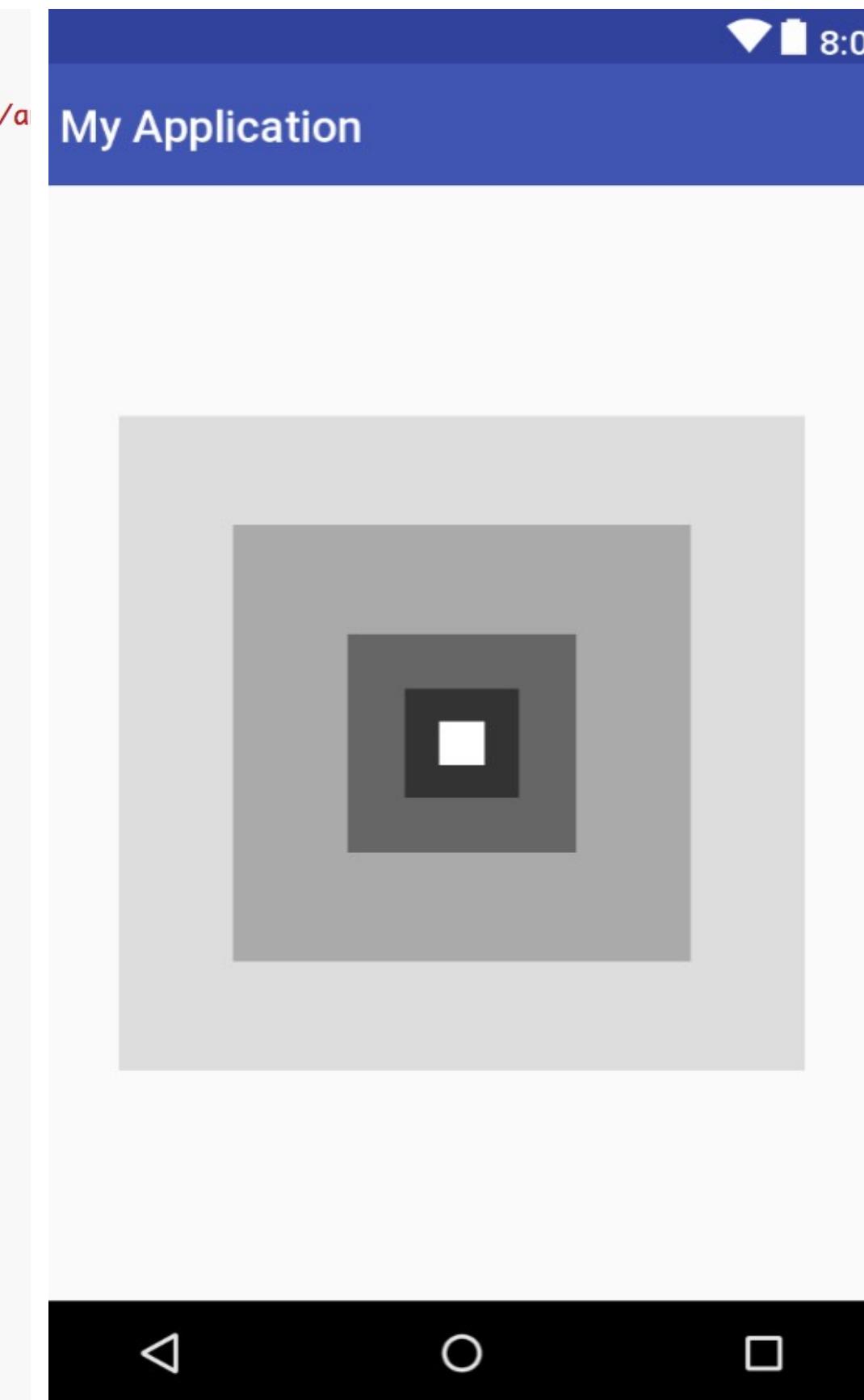
    <View
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_gravity="center"
        android:background="#aaaaaa"/>

    <View
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_gravity="center"
        android:background="#666666"/>

    <View
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_gravity="center"
        android:background="#333333"/>

    <View
        android:layout_width="20dp"
        android:layout_height="20dp"
        android:layout_gravity="center"
        android:background="@android:color/white"/>

</FrameLayout>
```



# LinearLayout

Выравнивает все дочерние  
объекты в одном  
направлении — вертикально  
или горизонтально.

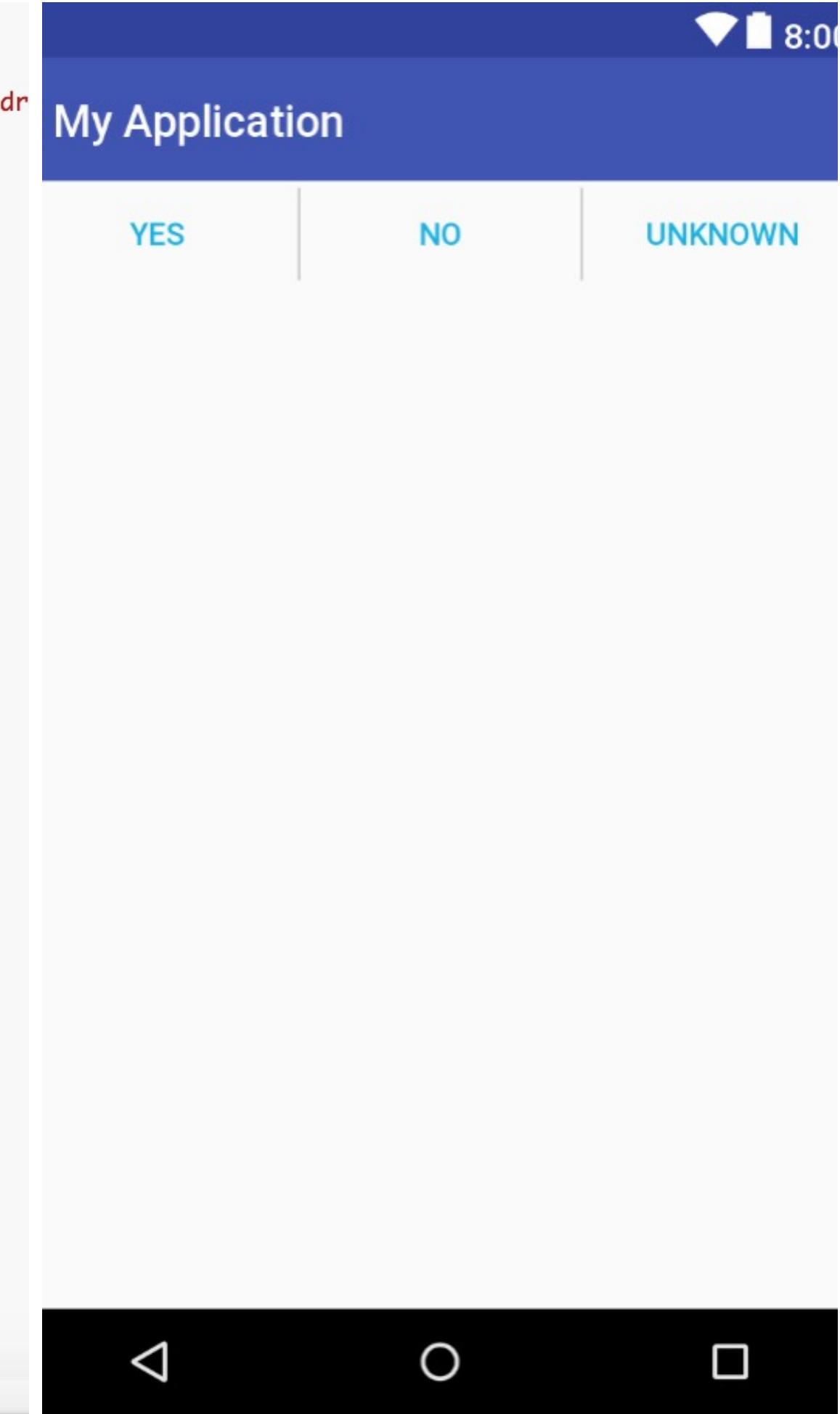
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:adjustViewBounds="true"
    android:divider="@drawable/sePARATOR"
    android:dividerPadding="3dp"
    android:orientation="horizontal"
    android:showDividers="middle" >

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        style="?android:attr/buttonBarButtonStyle"
        android:text="Yes"
        android:layout_weight="1"
        android:id="@+id/button1"
        android:textColor="#00b0e4" />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        style="?android:attr/buttonBarButtonStyle"
        android:text="No"
        android:layout_weight="1"
        android:id="@+id/button2"
        android:textColor="#00b0e4" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        style="?android:attr/buttonBarButtonStyle"
        android:text="Unknown"
        android:layout_weight="1"
        android:id="@+id/button3"
        android:textColor="#00b0e4" />

</LinearLayout>
```

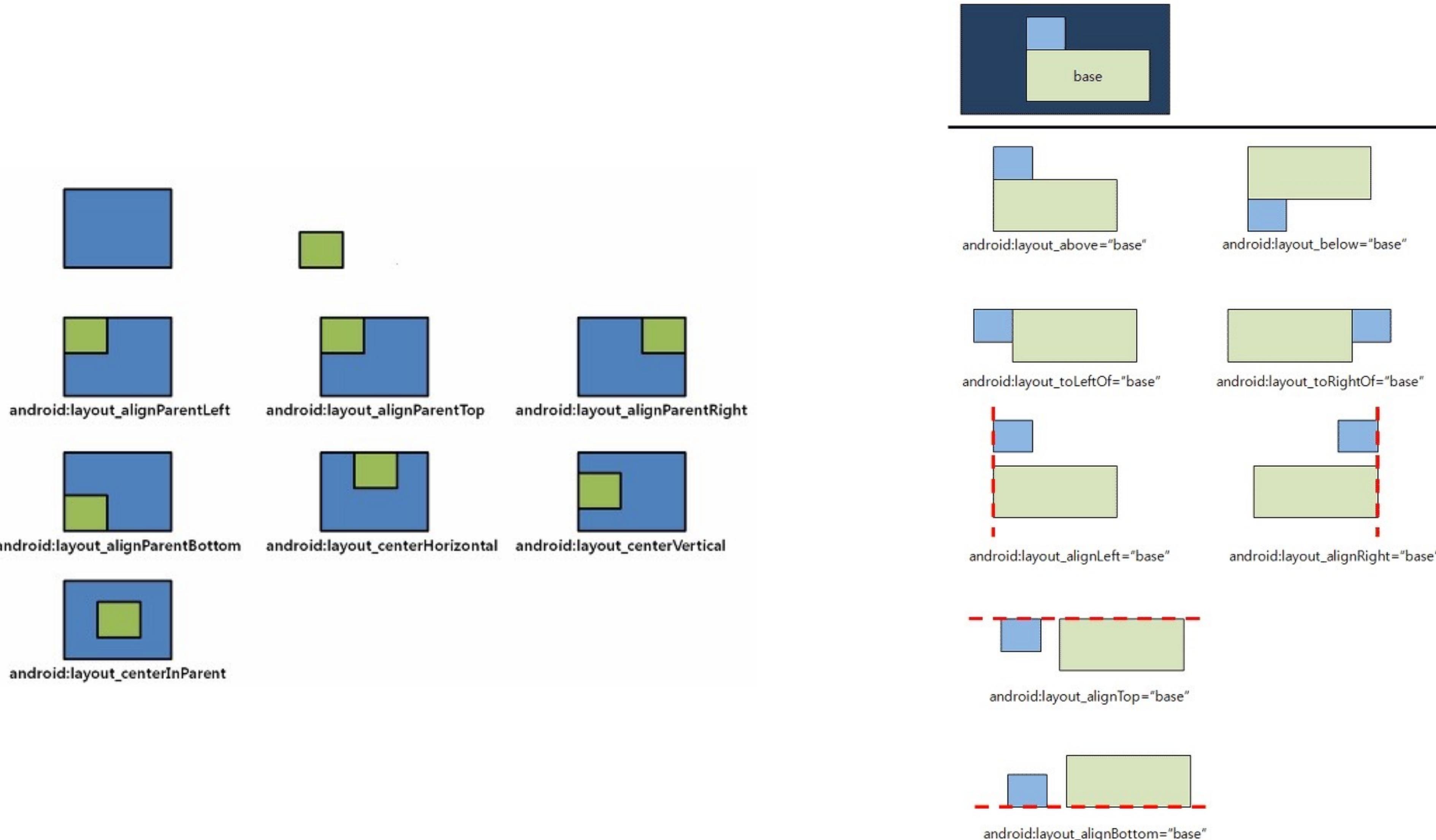


# RelativeLayout

Позволяет дочерним компонентам определять свою позицию относительно родительского компонента или относительно соседних дочерних элементов.



# RelativeLayout

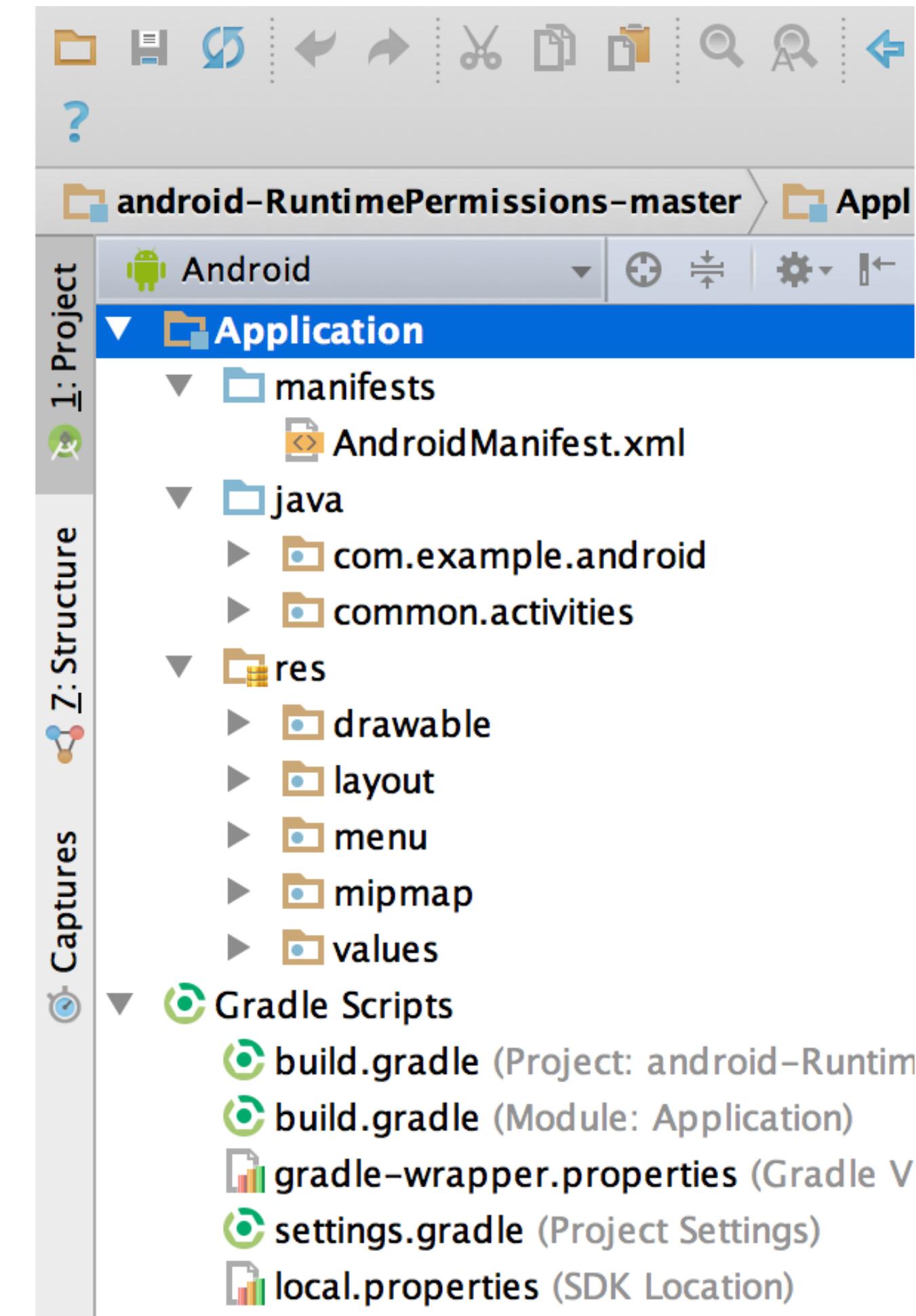


# Android Resources



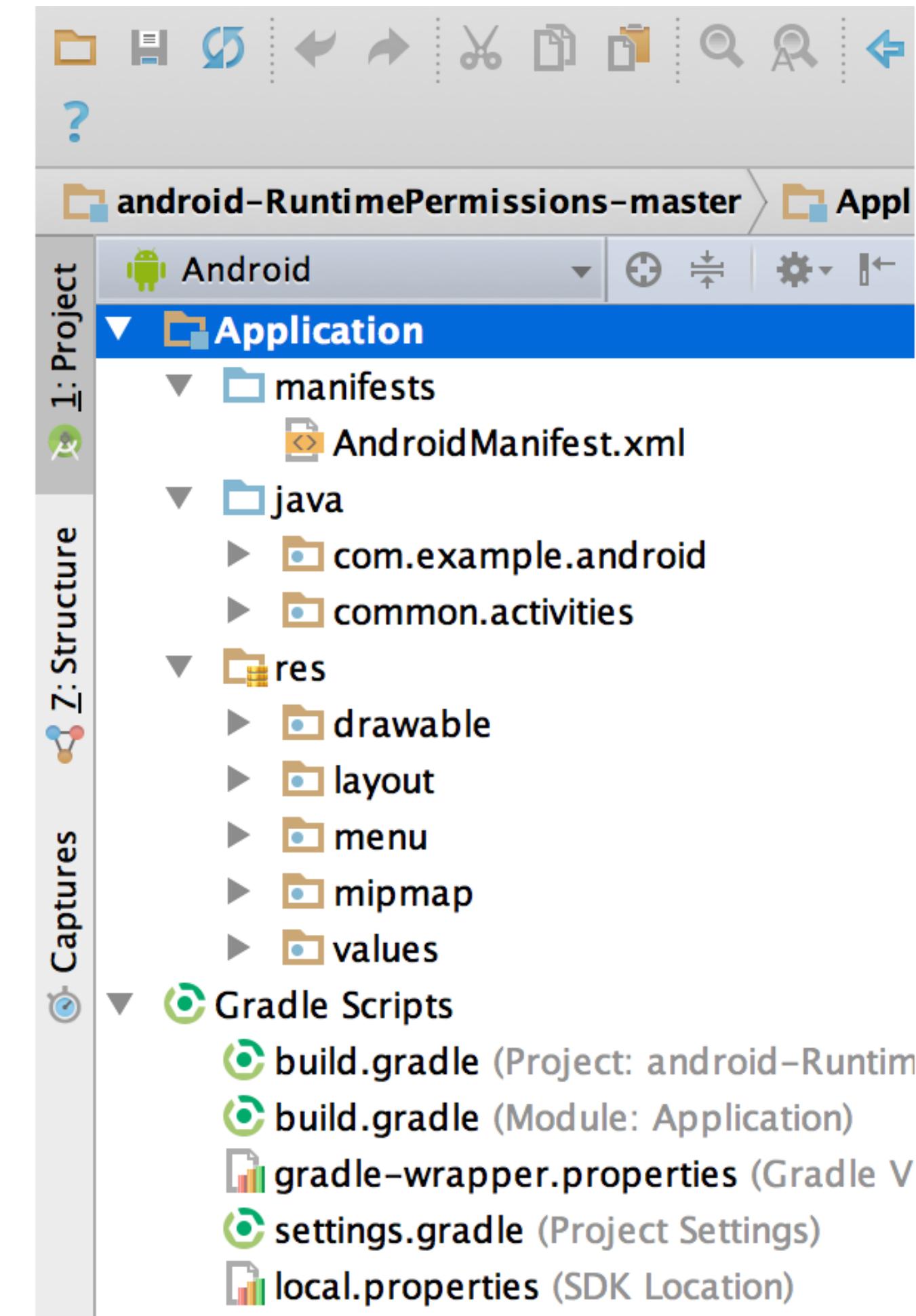
# Android Resources

Позволяют приложению динамически выбирать данные для разных конфигураций, языков и регионов.



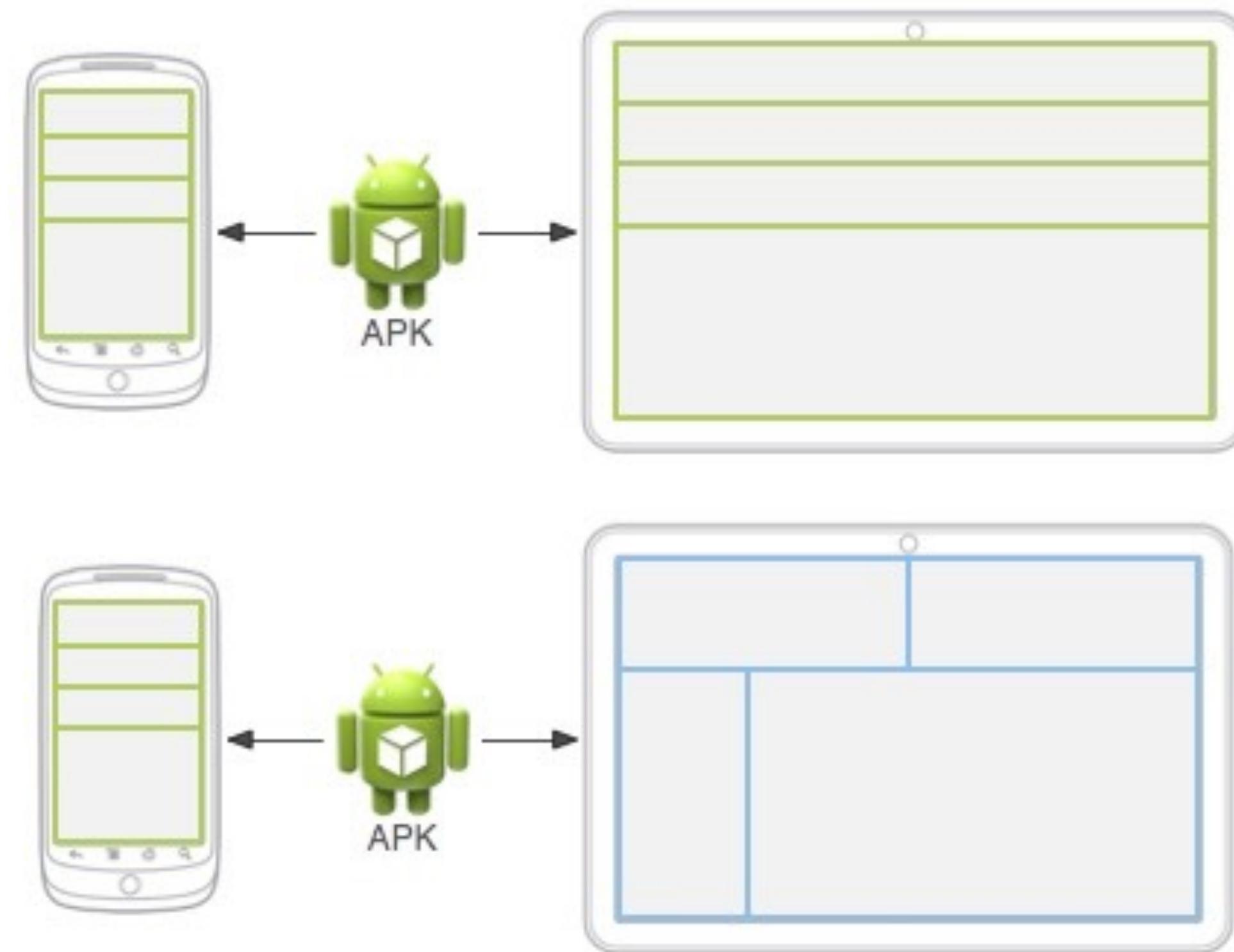
# Android Resources

- › Анимация
- › Меню
- › Растроевые изображения
- › Графические файлы
- › Ресурсы разметки
- › Строковые данные



# Android Resources

- › Ресурсы по умолчанию
- › Альтернативные ресурсы  
(определяются с помощью квалификаторов)



Конфигурация	Значения квалификатора	Описание
Язык и регион	Примеры: <code>en</code> <code>fr</code> <code>en-rUS</code> <code>fr-rFR</code> <code>fr-rCA</code> и т. д.	Язык задается двухбуквенным кодом языка <a href="#">ISO 639-1</a> , к которому можно добавить двухбуквенный код региона <a href="#">ISO 3166-1-alpha-2</a> (которому предшествует строчная буква "r").
Направление макета	<code>ldrtl</code> <code>ldltr</code>	Направление макета для приложения. Квалификатор <code>ldrtl</code> означает «направление макета справа налево». Квалификатор <code>ldltr</code> означает «направление макета слева направо» и используется по умолчанию.  <i>Добавлено в API уровня 17.</i>
<code>smallestWidth</code>	<code>sw&lt;N&gt;dp</code>  Примеры: <code>sw320dp</code> <code>sw600dp</code> <code>sw720dp</code> и т. д.	Основной размер экрана, указывающий минимальный размер доступной области экрана.  <i>Добавлено в API уровня 13.</i>
Размер экрана	<code>small</code> <code>normal</code> <code>large</code> <code>xlarge</code>	<code>small</code> : Экраны, подобные по размеру экрану QVGA низкой плотности.  <code>normal</code> : Экраны, подобные по размеру экрану HVGA средней плотности.  <code>large</code> : Экраны, подобные по размеру экрану VGA средней плотности.  <code>xlarge</code> : Экраны значительно крупнее обычного экрана HVGA средней плотности. Минимальный размер макета для очень большого экрана составляет приблизительно 720x960 пикселов.  <i>Добавлено в API уровня 9.</i>  <i>Добавлено в API уровня 4.</i>
Ориентация экрана	<code>port</code> <code>land</code>	<code>port</code> : Устройство в портретной (вертикальной) ориентации  <code>land</code> : Устройство в книжной (горизонтальной) ориентации

Конфигурация	Значения квалификатора	Описание
Режим пользовательского интерфейса	<code>car</code> <code>desk</code> <code>television</code> <code>appliance</code> <code>watch</code>	<p><code>car</code>: Устройство подсоединенено к автомобильной док-станции</p> <p><code>desk</code>: Устройство подсоединенено к настольной док-станции</p> <p><code>television</code>: Устройство подсоединенено к телевизору, обеспечивая взаимодействие с расстояния «три метра», когда пользовательский интерфейс находится на большом экране, находящемся вдалеке от пользователя, ориентированное на управление с помощью навигационной клавиши или другого устройства без указателя</p> <p><code>appliance</code>: Устройство служит в качестве прибора без дисплея</p> <p><code>watch</code>: Устройство с дисплеем для ношения на запястье</p> <p><i>Добавлено в API уровня 8, телевизор добавлен в API 13, часы добавлены в API 20.</i></p>
Ночной режим	<code>night</code> <code>notnight</code>	<p><code>night</code>: Ночное время</p> <p><code>notnight</code>: Дневное время</p> <p><i>Добавлено в API уровня 8.</i></p>
Плотность пикселов на экране (dpi)	<code>ldpi</code> <code>mdpi</code> <code>hdpi</code> <code>xhdpi</code> <code>xxhdpi</code> <code>xxxhdpi</code> <code>nodpi</code> <code>tvdpi</code>	<p><code>ldpi</code>: Экраны низкой плотности; приблизительно 120 dpi.</p> <p><code>mdpi</code>: Экраны средней плотности (обычные HVGA); приблизительно 160 dpi.</p> <p><code>hdpi</code>: Экраны высокой плотности; приблизительно 240 dpi.</p> <p><code>xhdpi</code>: Экраны очень высокой плотности; приблизительно 320 dpi. <i>Добавлено в API уровня 8.</i></p> <p><code>xxhdpi</code>: Экраны сверхвысокой плотности; приблизительно 480 dpi. <i>Добавлено в API уровня 16.</i></p> <p><code>xxxhdpi</code>: Использование исключительно высокой плотности (только значок запуска, см. <a href="#">примечание</a> в документе <i>Поддержка нескольких экранов</i>); приблизительно 640 dpi. <i>Добавлено в API уровня 18.</i></p> <p><code>nodpi</code>: Этот режим можно использовать для растровых графических ресурсов, которые не требуется масштабировать в соответствии с плотностью устройства.</p> <p><code>tvdpi</code>: Экраны промежуточной плотности между mdpi и hdpi; приблизительно 213 dpi. Этот режим не считается «основной» группой плотности. Он главным образом предназначен для телевизоров, и большинство приложений в нем не нуждается — при условии, что ресурсов mdpi и hdpi достаточно для большинства приложений, и система будет масштабировать их при необходимости. Этот квалификатор введен в API уровня 13.</p>

# Доступ к ресурсам

- › R.drawable
- › R.layout
- › R.string
- › ...

Доступ к ресурсам осуществляется через класс R, который создается во время компиляции приложения

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="opaque_red">#f00</color>
    <string name="hello_message">Hello!</string>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<EditText xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:textColor="@color/opaque_red"
    android:text="@string/hello_message" />
```

```
// Set the text on a TextView object using a resource ID
TextView msgTextView = (TextView) findViewById(R.id.msg);
msgTextView.setText(R.string.hello_message);
```

# Доступ к ресурсам

```
<Button  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/button_text"  
    android:textColor="@color/button_text" />
```

```
<?xml version="1.0" encoding="utf-8"?>  
<selector xmlns:android="http://schemas.android.com/apk/res  
    <item android:state_pressed="true"  
        android:color="#ffff0000"/> <!-- pressed -->  
    <item android:state_focused="true"  
        android:color="#ff0000ff"/> <!-- focused -->  
    <item android:color="#ff000000"/> <!-- default -->  
/>
```

Color State List Resource

**res/color/button\_text.xml**

```
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:textColor="#00FF00"  
    android:typeface="monospace"  
    android:text="@string/hello" />
```

```
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:textAppearance="@style/CodeFont"  
    android:text="@string/hello" />
```

Styles and Themes

**res/values/styles.xml**

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <color  
        name="color_name"  
        >hex_color</color>  
</resources>
```

- #RGB
- #ARGB
- #RRGGBB
- #AARRGGBB

Colors Resource

**res/values/colors.xml**

# Единицы измерения

## Единицы измерения:

- › **dp** или **dip** (density-independent pixels) — это не зависящий от разрешения пиксель, равный физическому пикселю на экране с плотностью 160 точек/дюйм;
- › **in** (inches) — дюймы, базируются на физических размерах экрана.
- › **mm** (millimeters) — миллиметры, базируются на физических размерах экрана.
- › **pt** (points) — 1/72 дюйма, базируются на физических размерах экрана;
- › **px** (pixels) — пиксели. Точки на экране - минимальные единицы измерения;
- › **sp** (scale-independent pixels) — это не зависящий от разрешения пиксель, равный физическому пикселю на экране с плотностью 160 точек/дюйм, но масштабируется на основе выбранного пользователем размера текста. Следует применять для указания величины шрифта;

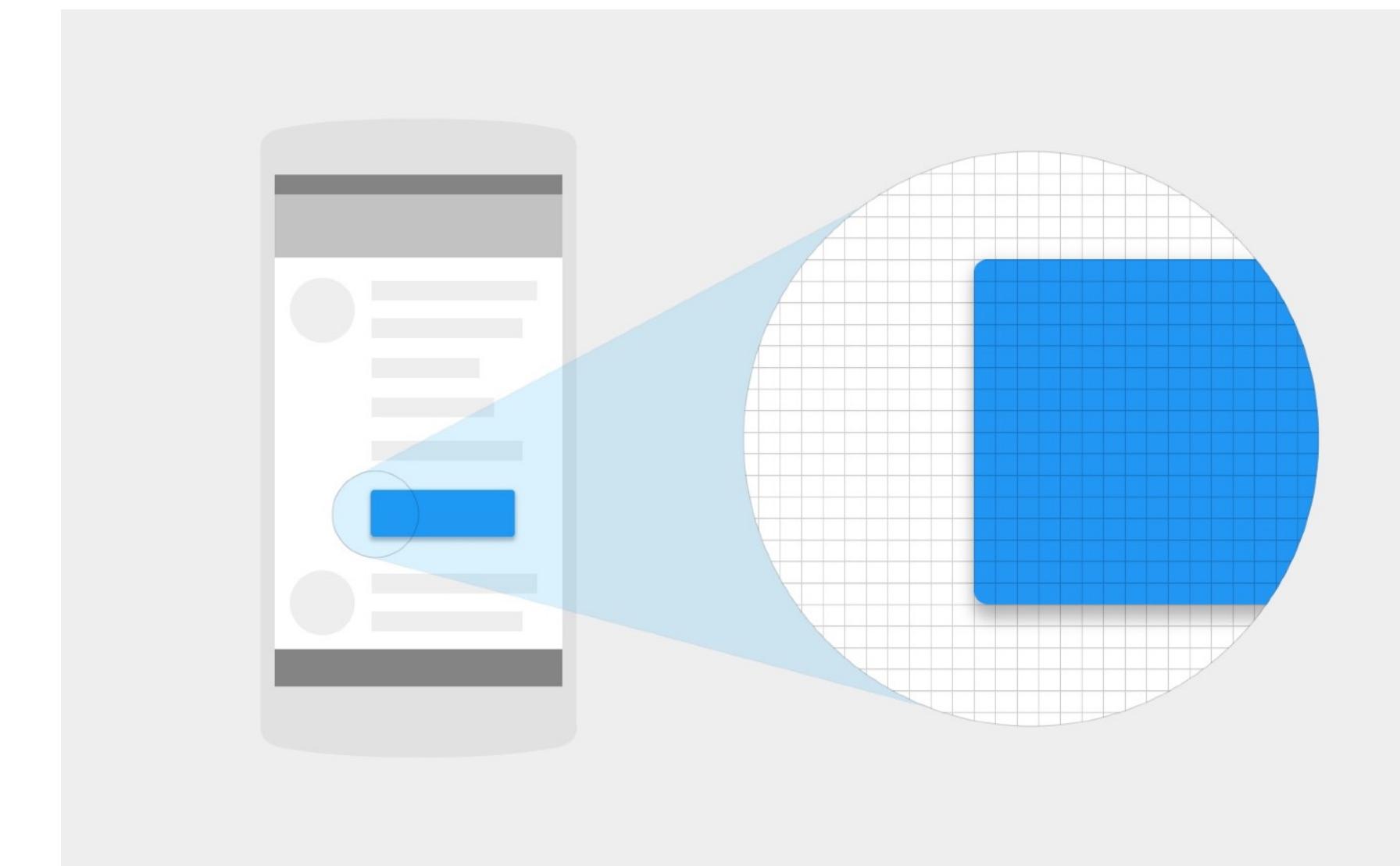
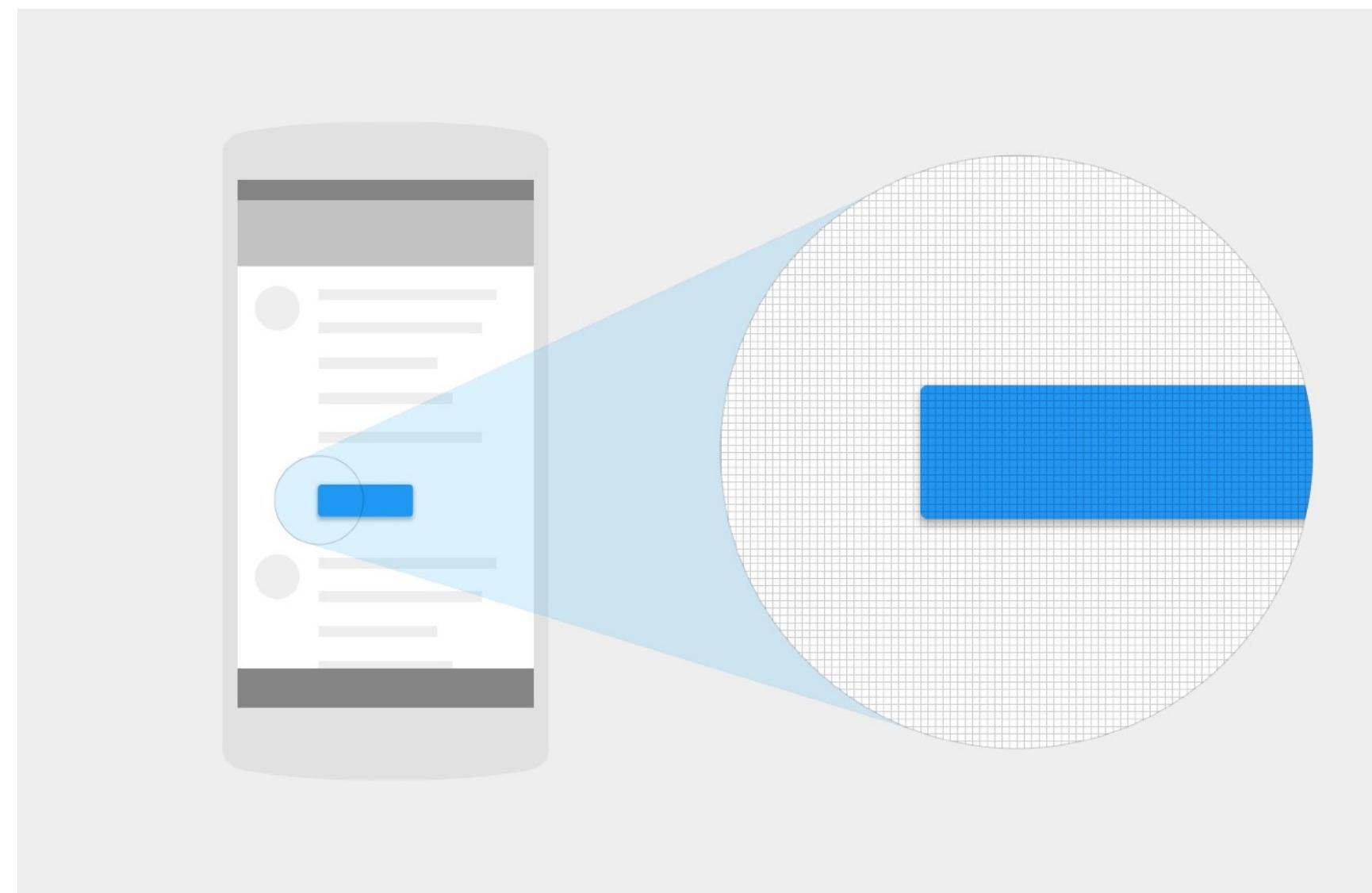
**in**, **mm** и **pt** – неизменны относительно друг друга. Всегда  $1 \text{ in} = 25,4 \text{ mm}$  и  $1 \text{ in} = 72 \text{ pt}$ . Исторически так сложилось, что разработчики всегда

# Плотность пикселей

**Плотность пикселей** (ppi) - количество пикселей, помещающихся в одном дюйме.

Элементы UI (например, кнопки) выглядят крупнее на устройствах с экранами с низкой плотностью и меньше на экранах с высокой плотностью.

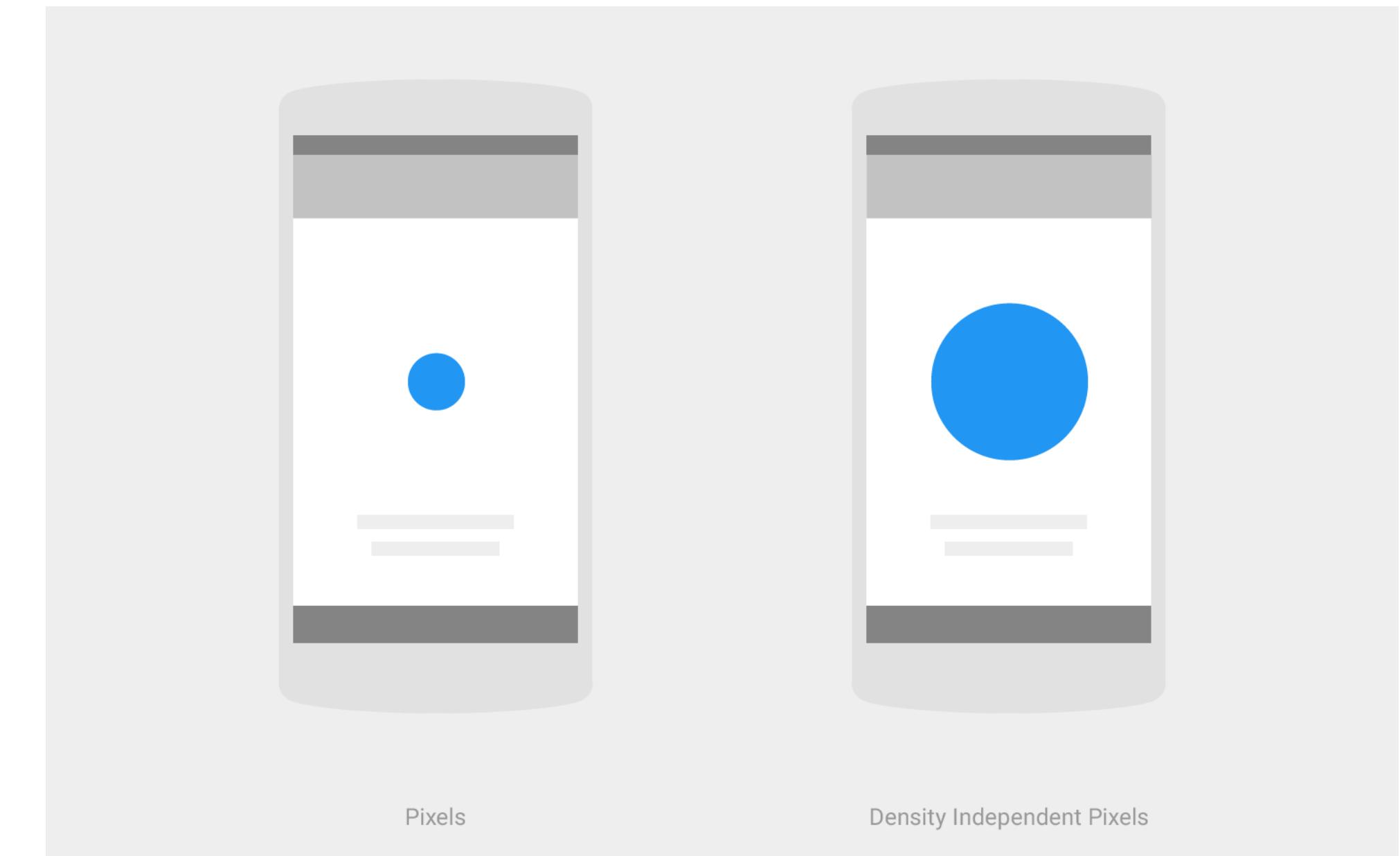
**Плотность экрана** (dpi) - ширина (или высота) экрана в пикселях / ширина (или высота) экрана в дюймах



# Не зависящие от плотности пиксели

1 dp равен одному физическому пикслю на экране с плотностью 160 dpi

$$1 \text{ dp} = 1 \text{ px} * 160 / \text{dpi}$$



**Не зависящие от плотности пиксели (dp)** – это гибкие единицы измерения, которые на любом экране масштабируются до одного и того же размера.

Разрешение экрана	Ширина экрана в пикселях (dpi * ширина в дюймах)	Ширина экрана в не зависящих от плотности пикселях
120 dpi	180 px	240 dp
160 dpi	240 px	
240 dpi	360 px	

# Масштабируемость

**Масштабируемые пиксели** (sp) – независящие от плотности пиксели, учитывающие пользовательские настройки шрифта

## Масштабирование изображений

Используя следующие соотношения сторон, можно масштабировать изображения, чтобы они выглядели одинаково на экранах с различным разрешением:

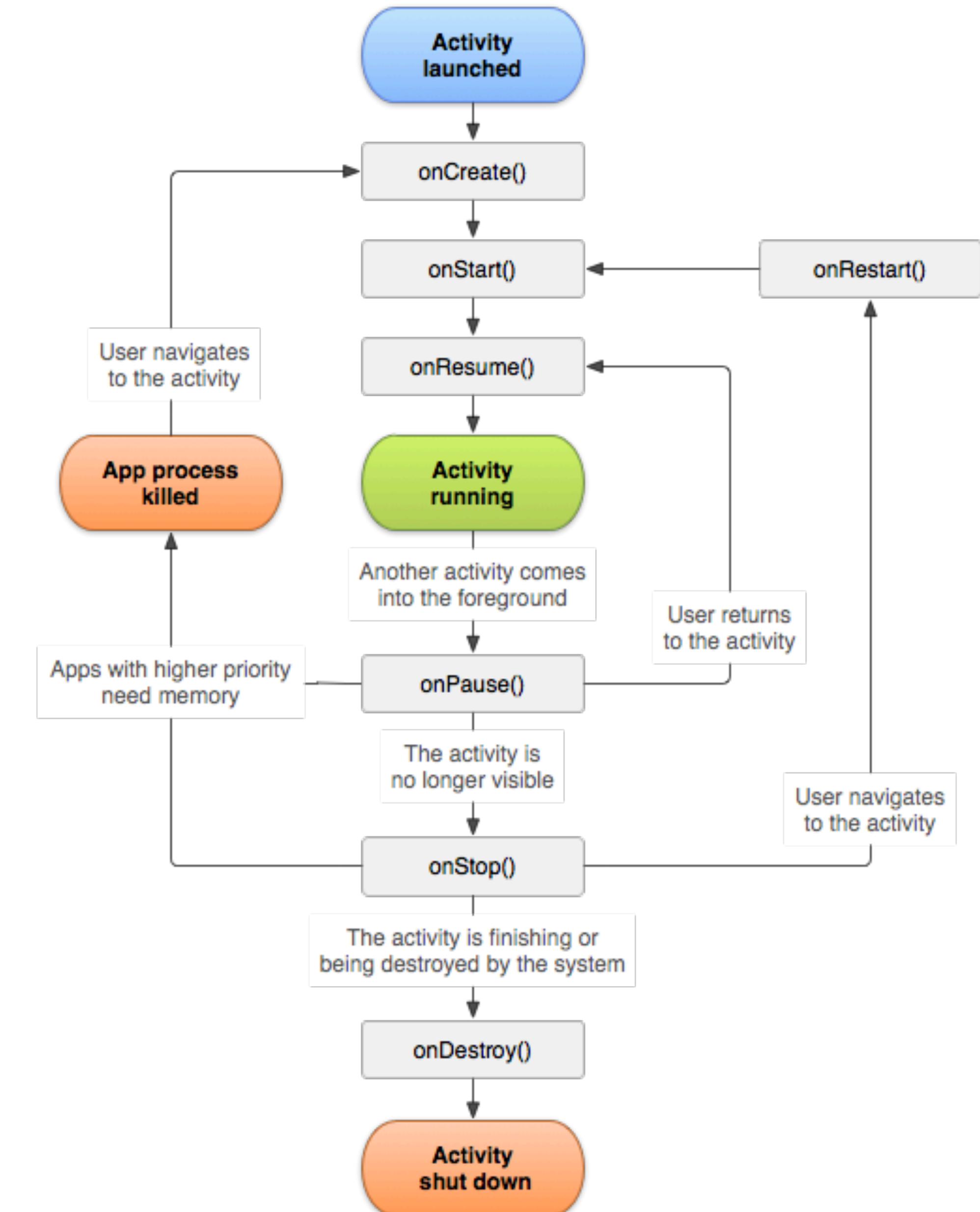
Разрешение	dpi	Пиксельный коэффициент	Размер изображения (в пикселях)
xxxhdpi	640	4.0	400 x 400
xxhdpi	480	3.0	300 x 300
xhdpi	320	2.0	200 x 200
hdpi	240	1.5	150 x 150
mdpi	160	1.0	100 x 100

# Activity



# Activity

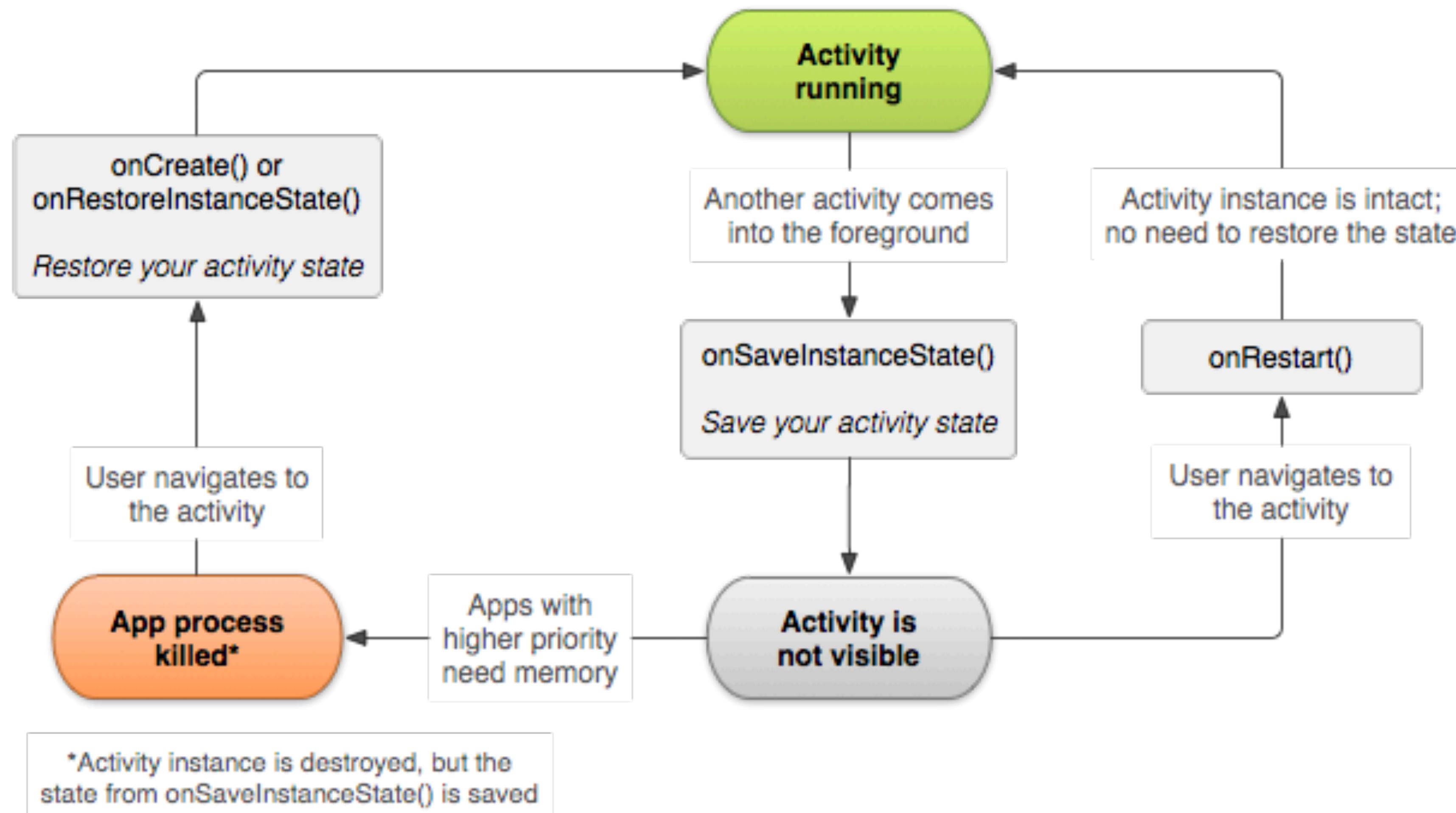
Экран приложения, с которым пользователи могут взаимодействовать для выполнения каких-либо действий



# Activity

- › **onCreate()** - вызывается при создании операции
- › **onStart()** - вызывается непосредственно перед тем, как операция становится видимой для пользователя
- › **onResume()** - вызывается непосредственно перед тем, как операция начинает взаимодействие с пользователем
- › **onPause()** - вызывается, когда система собирается возобновить другую операцию
- › **onStop()** - вызывается в случае, когда операция больше не отображается для пользователя
- › **onDestroy()** - вызывается перед тем, как операция будет уничтожена
- › **onRestart()** - вызывается после остановки операции непосредственно перед ее повторным запуском
- › **onSaveInstanceState()** - сохранение состояния и данных
- › **onRestoreInstanceState()** - восстановление состояния и данных

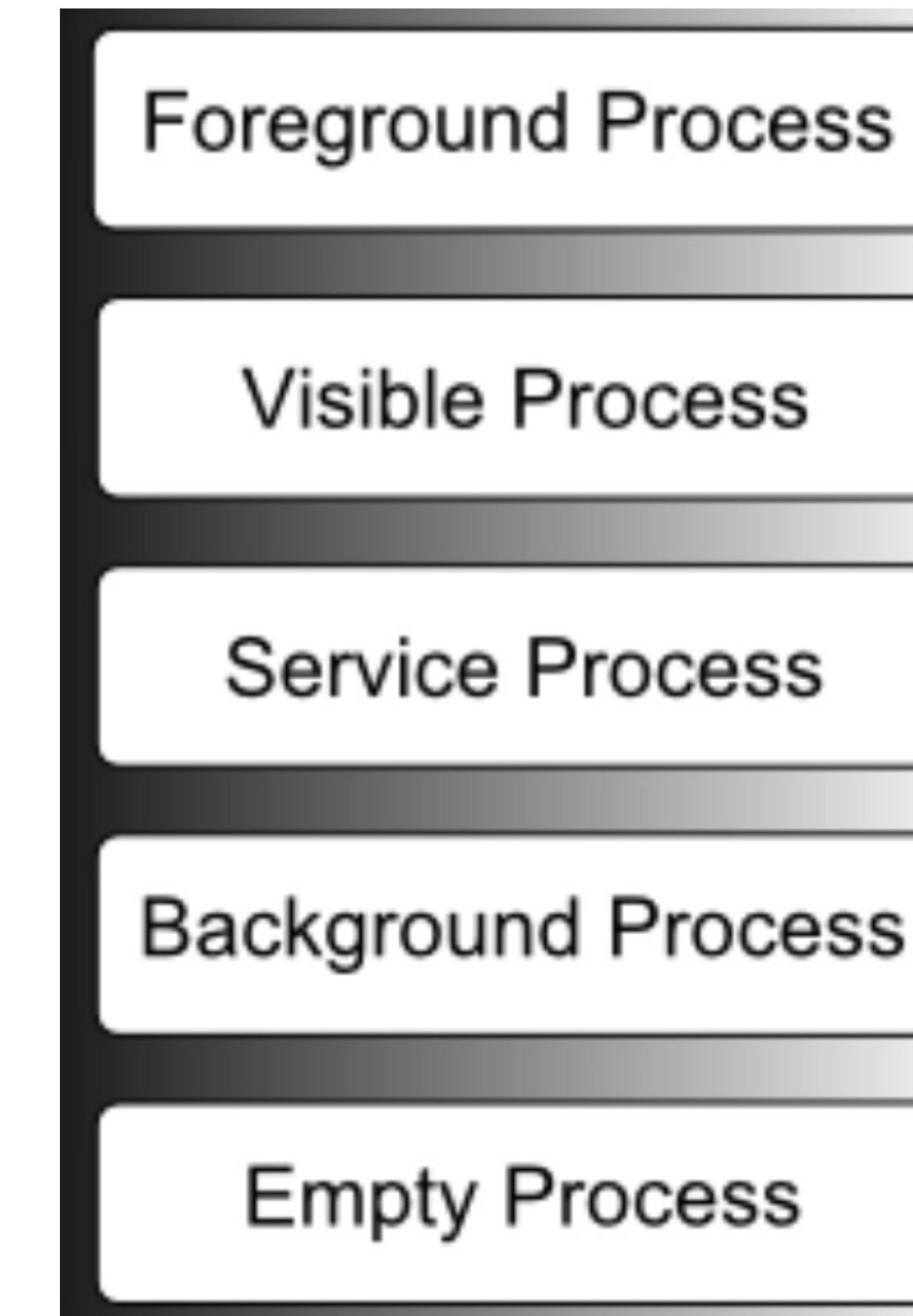
# Activity



# Процессы и потоки

По умолчанию все компоненты одного приложения работают в одном процессе и потоке.

Процессы с самым низким уровнем важности исключаются в первую очередь.



Highest Priority

Lowest Priority

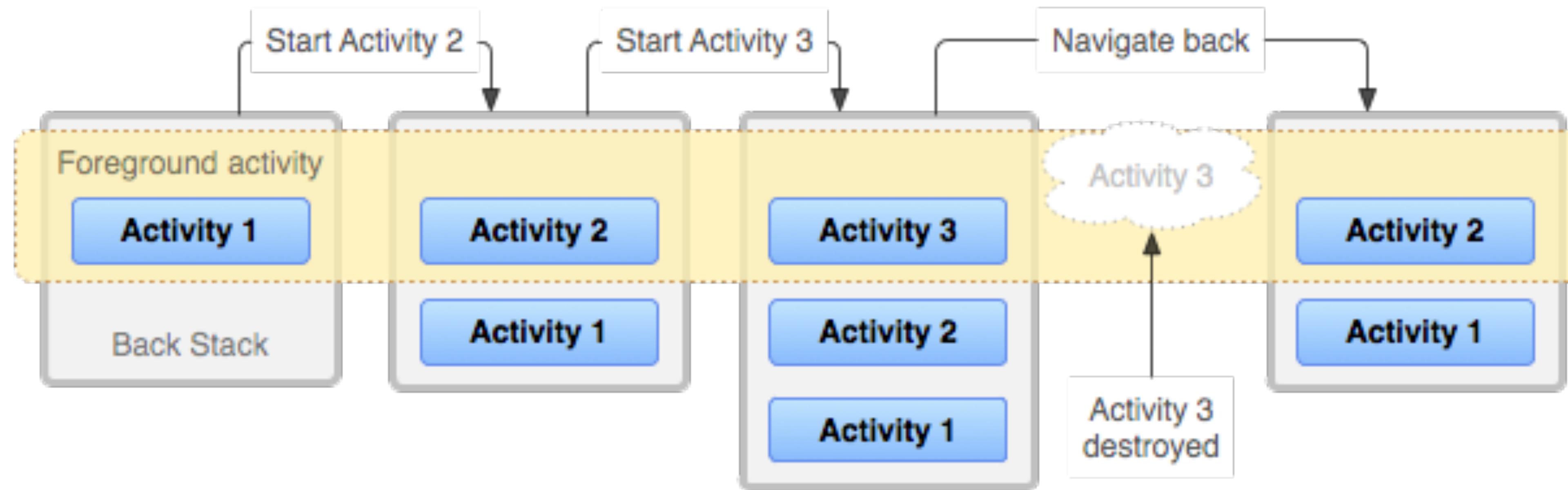
# Tasks and Back Stack



# Tasks and Back Stack

Задача — это коллекция операций, с которыми взаимодействует пользователь при выполнении определенного задания. Операции упорядочены в виде стека (стека переходов назад), в том порядке, в котором открывались операции

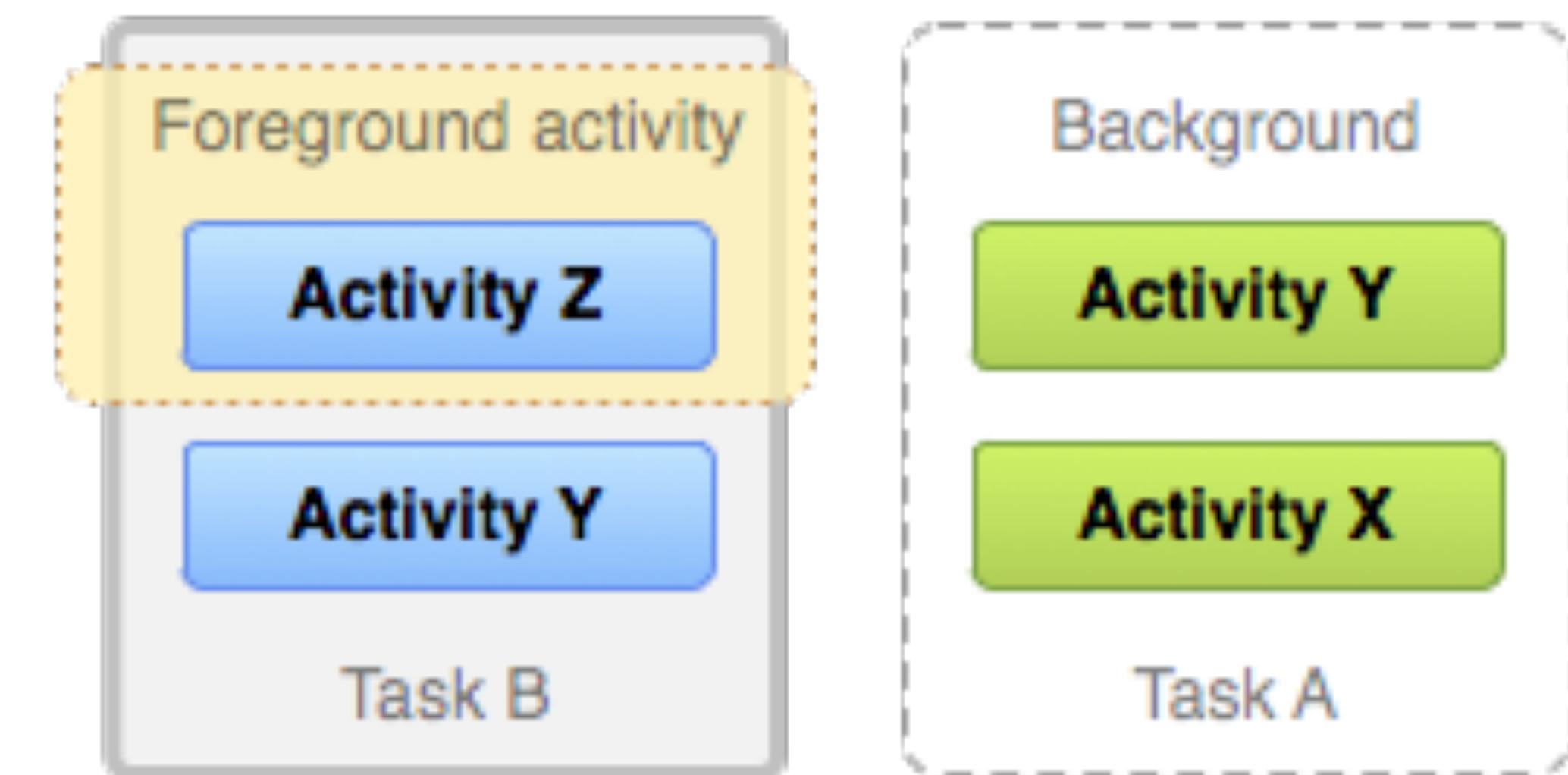
# Tasks and Back Stack



# Tasks and Back Stack

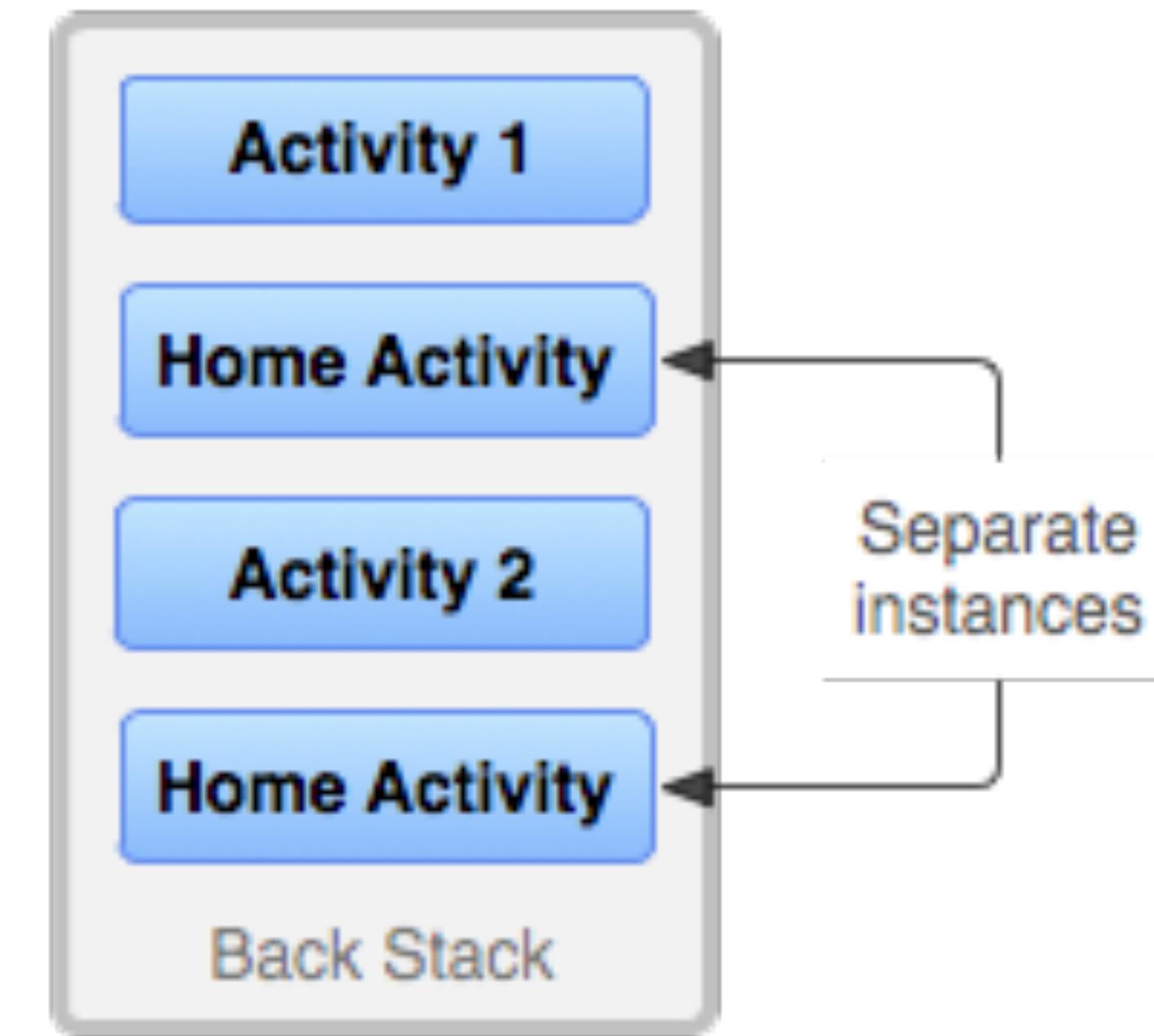
Задача — это связанный блок, который может переходить в фоновый режим, когда пользователи начинают новую задачу или переходят на главный экран с помощью кнопки Домой.

В фоновом режиме все операции задачи остановлены, но стек обратного вызова для задачи остается неизменным.



# Tasks and Back Stack

Операции в стеке никогда не переупорядочиваются. Для одной операции вашего приложения может быть создано несколько экземпляров (даже из разных задач).



# Tasks and Back Stack

- › Когда Операция А запускает Операцию В, Операция А останавливается, но система сохраняет ее состояние (например, положение прокрутки и текст, введенный в формы). Если пользователь нажимает кнопку Назад в Операции В, Операция А возобновляет работу из сохраненного состояния.
- › Когда пользователь выходит из задачи нажатием кнопки Домой, текущая операция останавливается и ее задача переводится в фоновый режим. Система сохраняет состояние каждой операции в задаче. Если пользователь впоследствии возобновляет задачу, выбирая значок запуска задачи, она переводится на передний план и возобновляет операцию на вершине стека.
- › Если пользователь нажимает кнопку Назад, текущая операция удаляется из стека и уничтожается. Возобновляется предыдущая операция в стеке. Когда операция уничтожается, система не сохраняет состояние операции.
- › Можно создавать несколько экземпляров операции, даже из других задач.

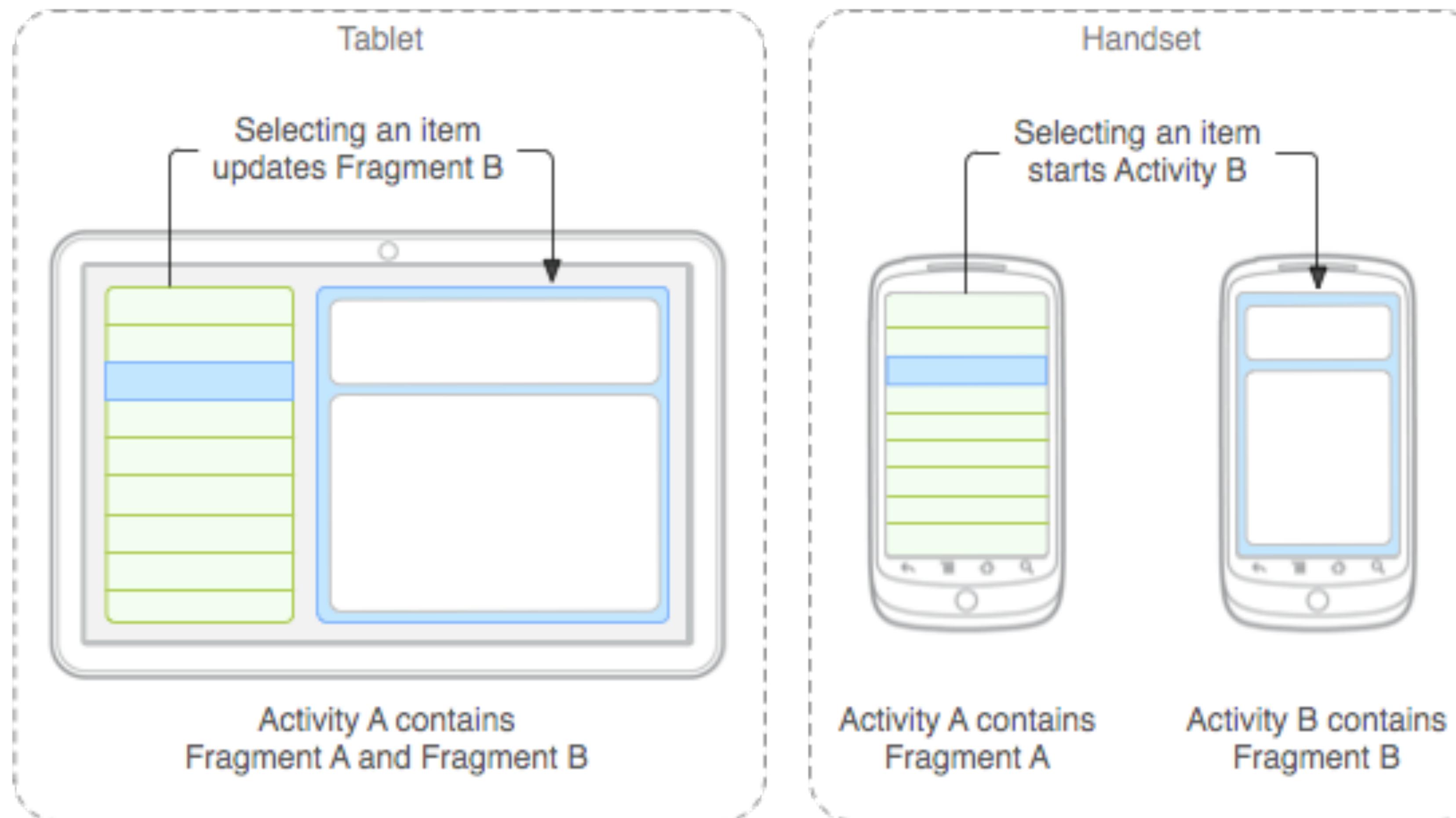
# Fragment



# Fragment

Фрагмент представляет поведение или часть пользовательского интерфейса в операции (класс Activity). Разработчик может объединить несколько фрагментов в одну операцию для построения многопанельного пользовательского интерфейса и повторного использования фрагмента в нескольких операциях.

# Fragment



# Fragment

Фрагмент всегда должен быть встроен в операцию, и на его жизненный цикл напрямую влияет жизненный цикл операции. Например, когда операция приостановлена, в том же состоянии находятся и все фрагменты внутри нее, а когда операция уничтожается, уничтожаются и все фрагменты. Однако пока операция выполняется (это соответствует состоянию возобновлена жизненного цикла), можно манипулировать каждым фрагментом независимо, например добавлять или удалять их.

# Fragment

## Возобновлен

Фрагмент виден во время выполнения операции.

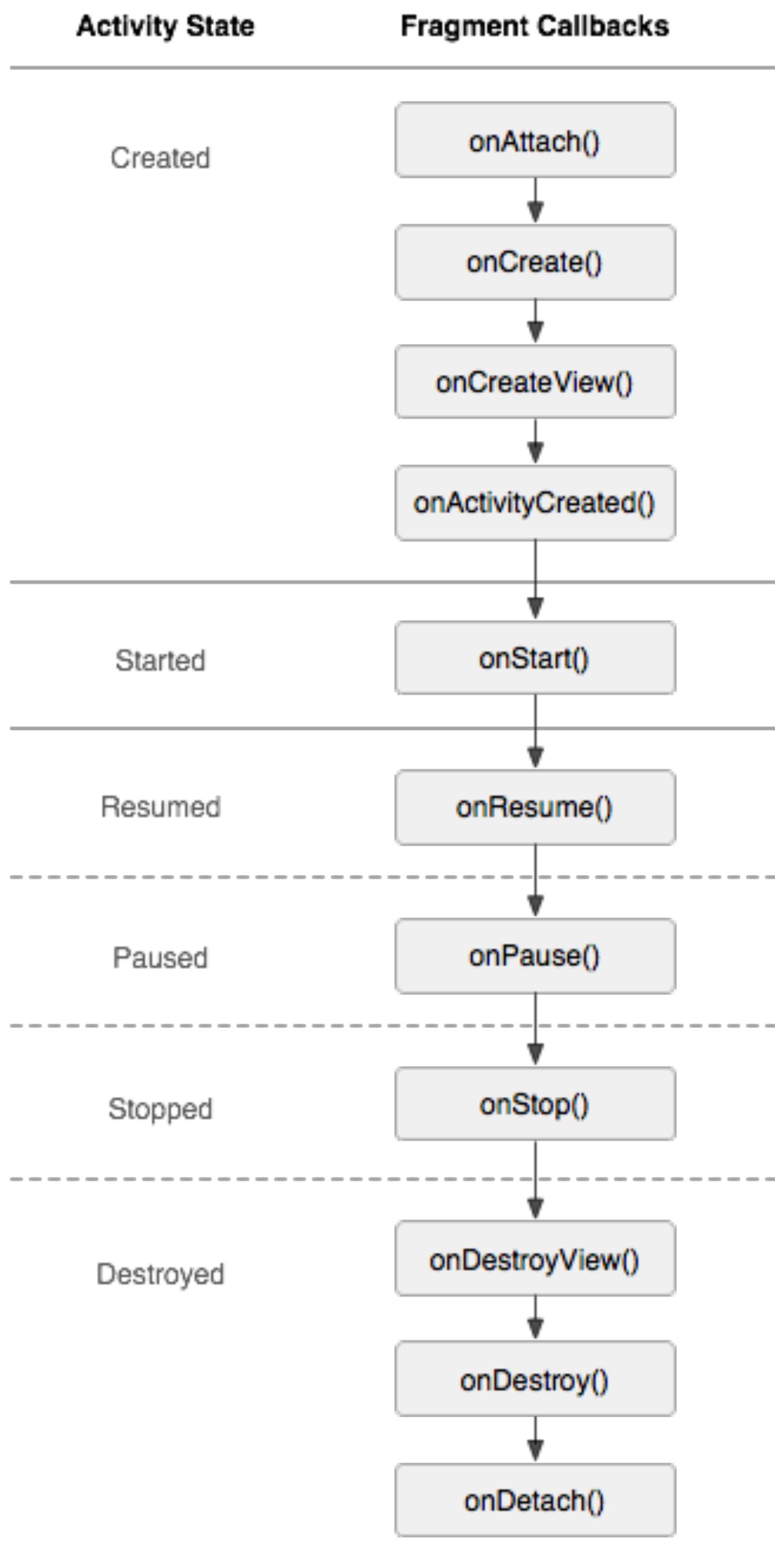
## Приостановлен

На переднем плане выполняется и находится в фокусе другая операция, но операция, содержащая данный фрагмент, по-прежнему видна (операция переднего плана частично прозрачна или не занимает весь экран).

## Остановлен

Фрагмент не виден. Либо контейнерная операция остановлена, либо фрагмент удален из нее, но добавлен в стек переходов назад.

Остановленный фрагмент по-прежнему активен (вся информация о состоянии и элементах сохранена в системе). Однако он больше не виден пользователю и будет уничтожен в случае уничтожения операции.



# Fragment

## onAttach()

Вызывается, когда фрагмент связывается с операцией (ему передается объект Activity).

## onCreateView()

Вызывается для создания иерархии представлений, связанной с фрагментом.

## onActivityCreated()

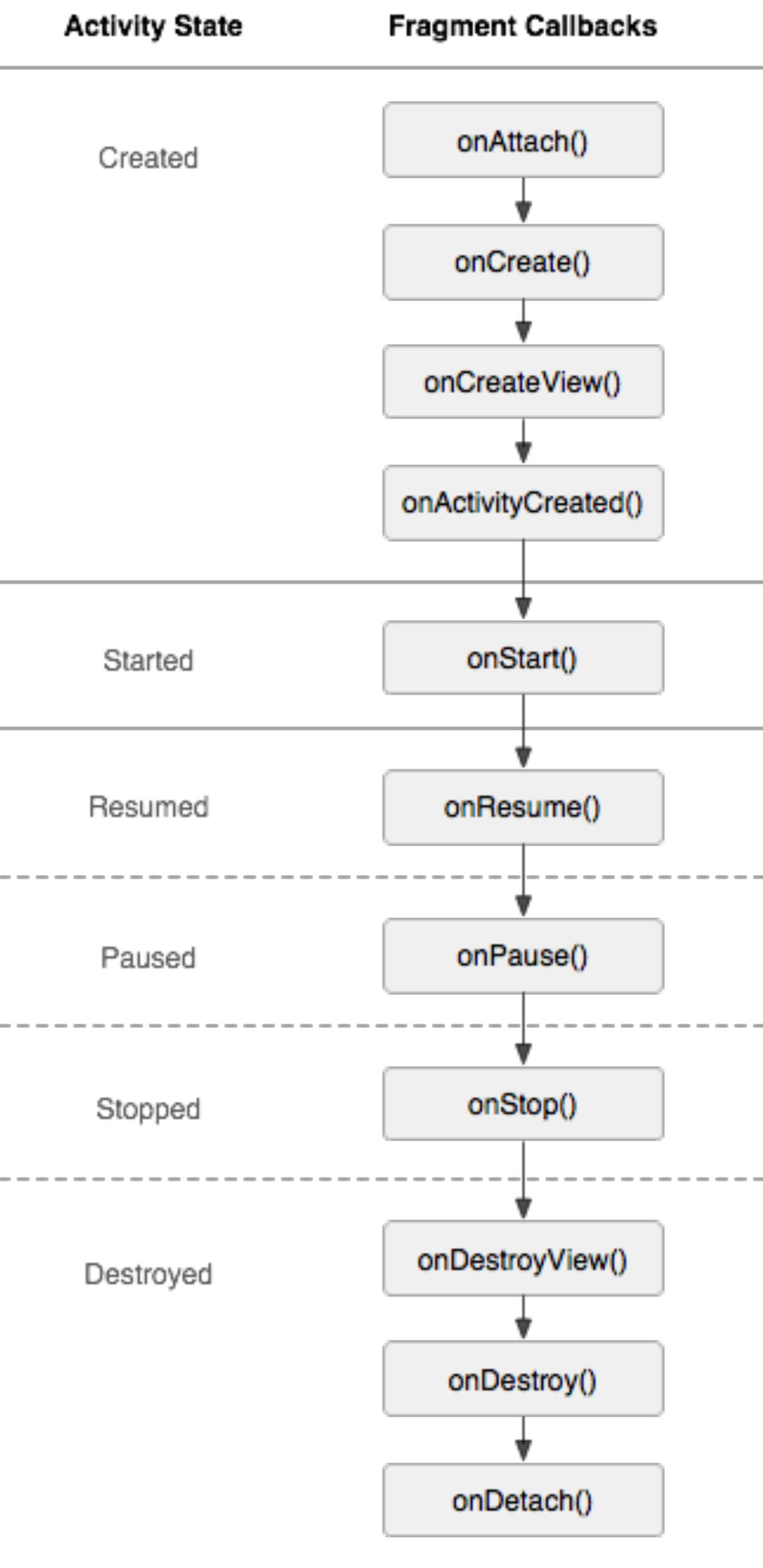
Вызывается, когда метод onCreate(), принадлежащий операции, возвращает управление.

## onDestroyView()

Вызывается при удалении иерархии представлений, связанной с фрагментом.

## onDetach()

Вызывается при разрыве связи фрагмента с операцией.



# Fragment

## **Готовые реализации фрагментов:**

- › DialogFragment
- › ListFragment
- › PreferenceFragment

# Fragment

```
public static class ExampleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.example_fragment, container, false)
    }
}
```

```
<?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"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
    <fragment android:name="com.example.news.ArticleReaderFragment"
        android:id="@+id/viewer"
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
</LinearLayout>
```

# Fragment

Большим достоинством использования фрагментов в операции является возможность добавлять, удалять, заменять их и выполнять другие действия с ними в ответ на действия пользователя. Любой набор изменений, вносимых в операцию, называется транзакцией.

- › `add()` - добавляет фрагмент
- › `remove()` - удаляет фрагмент
- › `replace()` - заменяет текущий фрагмент
- › `addToBackStack()` - добавляет фрагмент в стек переходов
- › `popBackStack()` - снимает фрагмент со стека переходов

# Fragment

```
// 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();
```

# Communicating with the Activity

```
public static class FragmentA extends ListFragment {  
    ...  
    // Container Activity must implement this interface  
    public interface OnArticleSelectedListener {  
        public void onArticleSelected(Uri articleUri);  
    }  
    ...  
}
```

```
public static class FragmentA extends ListFragment {  
    OnArticleSelectedListener mListener;  
    ...  
    @Override  
    public void onAttach(Context context) {  
        super.onAttach(context);  
        try {  
            mListener = (OnArticleSelectedListener) context;  
        } catch (ClassCastException e) {  
            throw new ClassCastException(context.toString() + " must :");  
        }  
    }  
    ...  
}
```

# Спасибо за внимание

Станислав Куликов



satsakul@yandex-team.ru