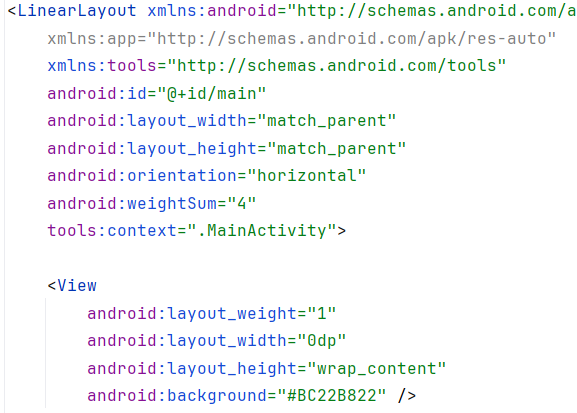
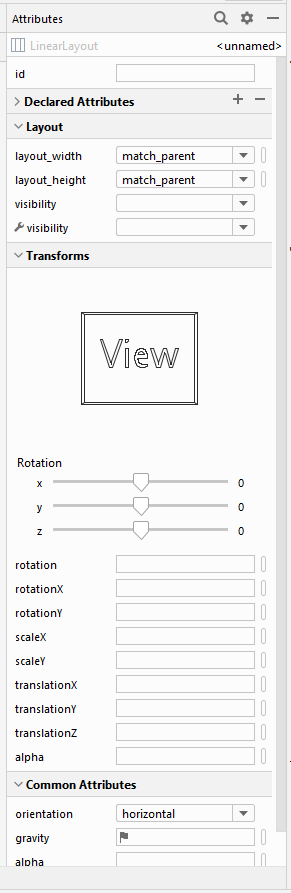
**Добавление и настройка свойств компонентов**

Мы уже изучили что есть три способа создания макетов интерфейса:

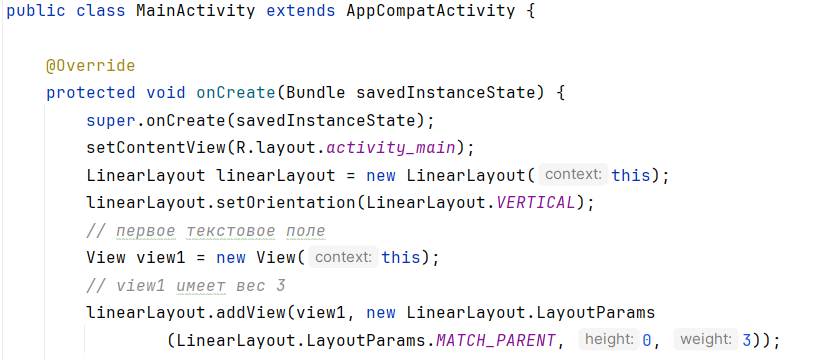
Свойства элементов можно настраивать через XML код



Свойства в конструкторе



и Java код



Давайте рассмотрим основные свойства элементов которые часто будут использоваться в нашей работе

**Определение размеров элементов**

При разработке приложений под Android мы можем использовать различные типы измерений:

* px: пиксели текущего экрана. Однако эта единица измерения не рекомендуется, так как реальное представление внешнего вида может изменяться в зависимости от устройства; каждое устройство имеет определенный набор пикселей на дюйм, поэтому количество пикселей на экране может также меняться
* dp: (device-independent pixels) независимые от плотности экрана пиксели. Абстрактная единица измерения, основанная на физической плотности экрана с разрешением 160 dpi (точек на дюйм). В этом случае 1dp = 1px. Если размер экрана больше или меньше, чем 160dpi, количество пикселей, которые применяются для отрисовки 1dp соответственно увеличивается или уменьшается. Например, на экране с 240 dpi 1dp=1,5px, а на экране с 320dpi 1dp=2px. Общая формула для получения количества физических пикселей из dp: px = dp \* (dpi / 160)
* sp: (scale-independent pixels) независимые от масштабирования пиксели. Допускают настройку размеров, производимую пользователем. Рекомендуются для работы со шрифтами.
* pt: 1/72 дюйма, базируются на физических размерах экрана
* mm: миллиметры
* in: дюймы

Предпочтительными единицами для использования являются dp. Это связано с тем, что мир мобильных устройств на Android сильно фрагментирован в плане разрешения и размеров экрана. И чем больше плотность пикселей на дюйм, тем соответственно больше пикселей нам будет доступно. Используя же стандартные физические пиксели, мы можем столкнуться с проблемой, что размеры элементов также будут сильно варьироваться в зависимости от плотности пикселей устройства.

**Ширина и высота элементов**

Все визуальные элементы, которые мы используем в приложении, как правило, упорядочиваются на экране с помощью контейнеров.

Для организации элементов внутри контейнера используются параметры разметки. Для их задания в файле xml используются атрибуты, которые начинаются с префикса layout\_. В частности, к таким параметрам относятся атрибуты **layout\_height** и **layout\_width**, которые используются для установки размеров и могут использовать одну из следующих опций:

Растяжение по всей ширине или высоте контейнера с помощью значения **match\_parent** (для всех контейнеров кроме ConstraintLayout) или 0dp (для ConstraintLayout)

Растяжение элемента до тех границ, которые достаточны, чтобы вместить все его содержимое с помощью значения **wrap\_content**

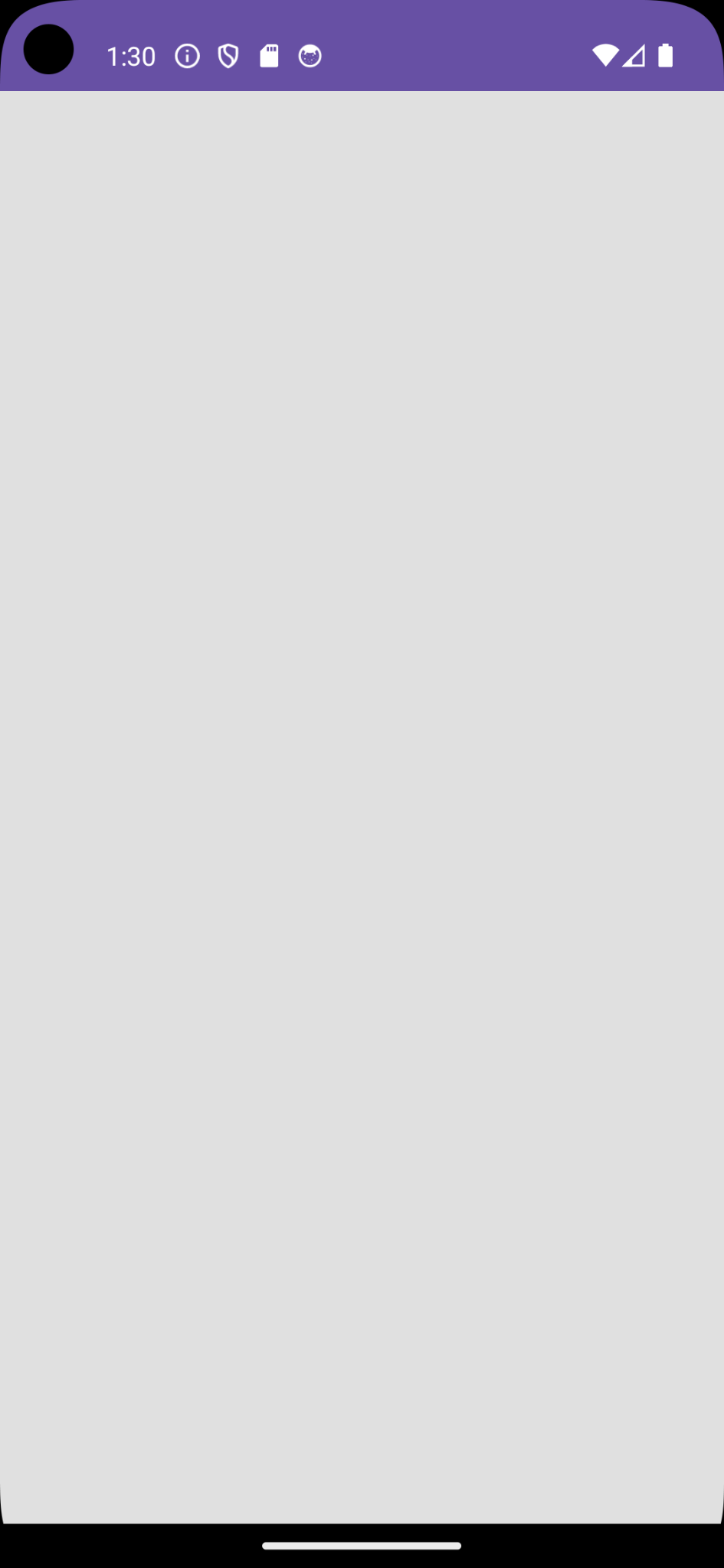
**match\_parent**

Установка значения match\_parent позволяет растянуть элемент по всей ширине или высоте контейнера. Стоит отметить, что данное значение применяется ко всем контейнерам, кроме ConstraintLayout. Например, рястянем элемент TextView по всей ширине и высоте контейнера LinearLayout:



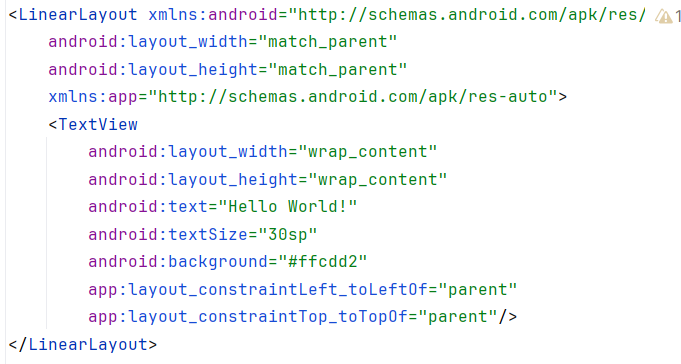
Контейнер самого верхнего уровня, в качестве которого в данном случае выступает LinearLayout, для высоты и ширины имеет значение match\_parent, то есть он будет заполнять всю область для activity - как правило, весь экран.

И View также принимает подобные атрибуты. Значение android:layout\_width="match\_parent" обеспечивает растяжение по ширине, а android:layout\_height="match\_parent" - по вертикали. Для наглядности в View применяет атрибут android:background, который представляет фон и в данном случае окрашивает элемент в цвет "#e0e0e0", благодаря чему мы можем увидеть занимаемую им область.



**wrap\_content**

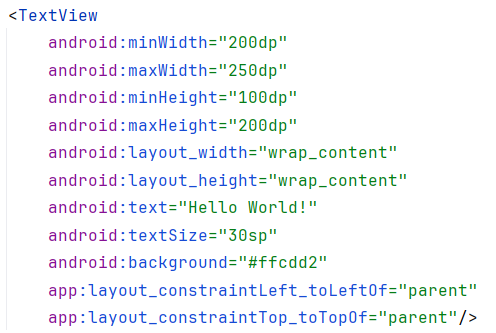
Значение wrap\_content устанавливает те значения для ширины или высоты, которые необходимы, чтобы разместить на экране содержимое элемента:



Здесь элемент TextView растягивается до тех значений, которые достаточны для размещения его текста.



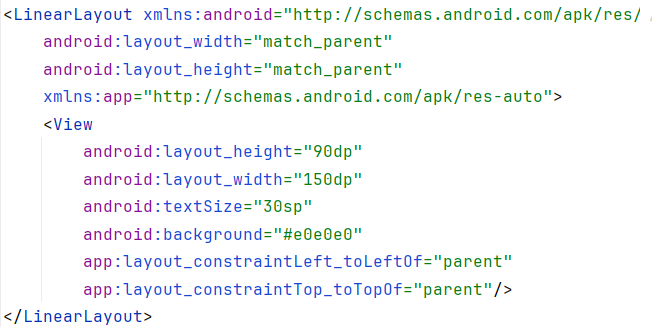
Мы можем дополнительно ограничить минимальные и максимальные значения с помощью атрибутов **minWidth/maxWidth** и **minHeight/maxHeight**:



В этом случае ширина TextView будет такой, которая достаточна для вмещения текста, но не больше значения maxWidth и не меньше значения minWidth. То же самое для установки высоты.

**Установка точных значений**

Также вы можете установить точные значения:

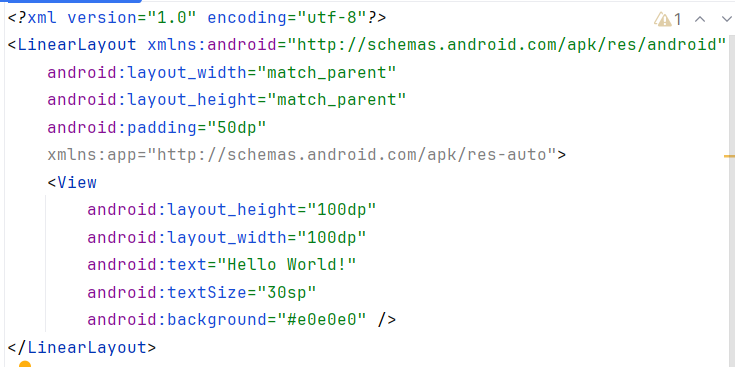


**Внутренние и внешние отступы**

Параметры разметки позволяют задать отступы как от внешних границ элемента до границ контейнера, так и внутри самого элемента между его границами и содержимым.

**Padding**

Для установки внутренних отступов применяется атрибут **android:padding**. Он устанавливает отступы контента от всех четырех сторон контейнера. Можно устанавливать отступы только от одной стороны контейнера, применяя следующие атрибуты: **android:paddingLeft**, **android:paddingRight**, **android:paddingTop** и **android:paddingBottom**.



У контейнера LinearLayout был установлен только один общий внутренний отступ в 50 единиц. Вложенный элемент View позиционируется в левом верхнем углу контейнера. Поэтому View будет отодвигаться от начальной точки (левый верхний угол контейнера LinearLayout ) вниз и влево на 50 единиц. Кроме того, такие же отступы будут действовать справа и снизу, если элемент будет примыкать к нижней или правой границе контейнера.



Установка одного отступа

android:padding="50dp"

Будет аналогична установке четырёх отступов

android:paddingTop="50dp"

android:paddingLeft="50dp"

android:paddingBottom="50dp"

android:paddingRight="50dp"

**Margin**

Для установки внешних отступов используется атрибут layout\_margin. Данный атрибут имеет модификации, которые позволяют задать отступ только от одной стороны: android:layout\_marginBottom, android:layout\_marginTop, android:layout\_marginLeft и android:layout\_marginRight (отступы соответственно от нижней, верхней, левой и правой границ):



Здесь у View задаются отступы от двух сторон (слева 60 единиц и сверху 50 единиц):

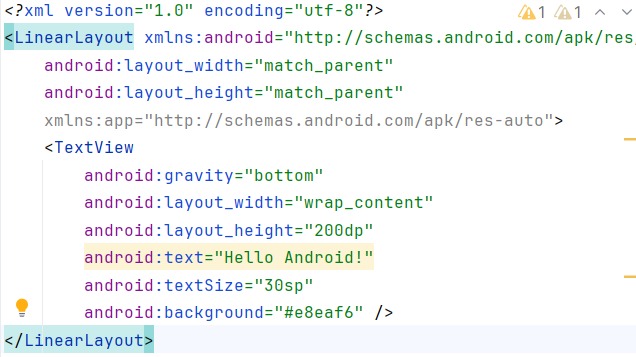


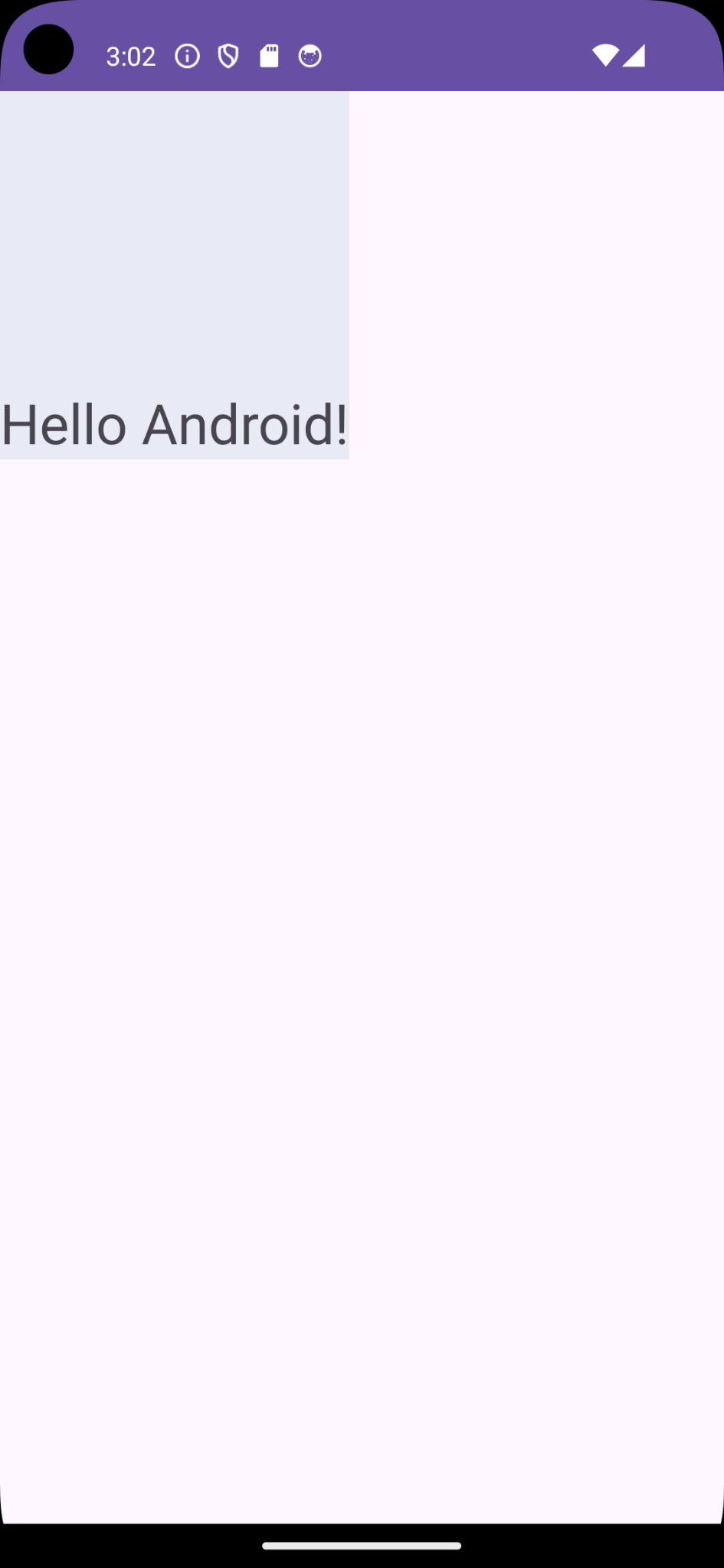
**Gravity**

Атрибут gravity задает позиционирование содержимого внутри визуального элемента. Он может принимать следующие значения:

* top: элементы размещаются вверху
* bottom: элементы размещаются внизу
* left: элементы размещаются в левой стороне
* right: элементы размещаются в правой стороне контейнера
* center\_vertical: выравнивает элементы по центру по вертикали
* center\_horizontal: выравнивает элементы по центру по горизонтали
* center: элементы размещаются по центру
* fill\_vertical: элемент растягивается по вертикали
* fill\_horizontal: элемент растягивается по горизонтали
* fill: элемент заполняет все пространство контейнера
* clip\_vertical: обрезает верхнюю и нижнюю границу элементов
* clip\_horizontal: обрезает правую и левую границу элементов
* start: элемент позиционируется в начале (в верхнем левом углу) контейнера
* end: элемент позиционируется в конце контейнера(в верхнем правом углу)

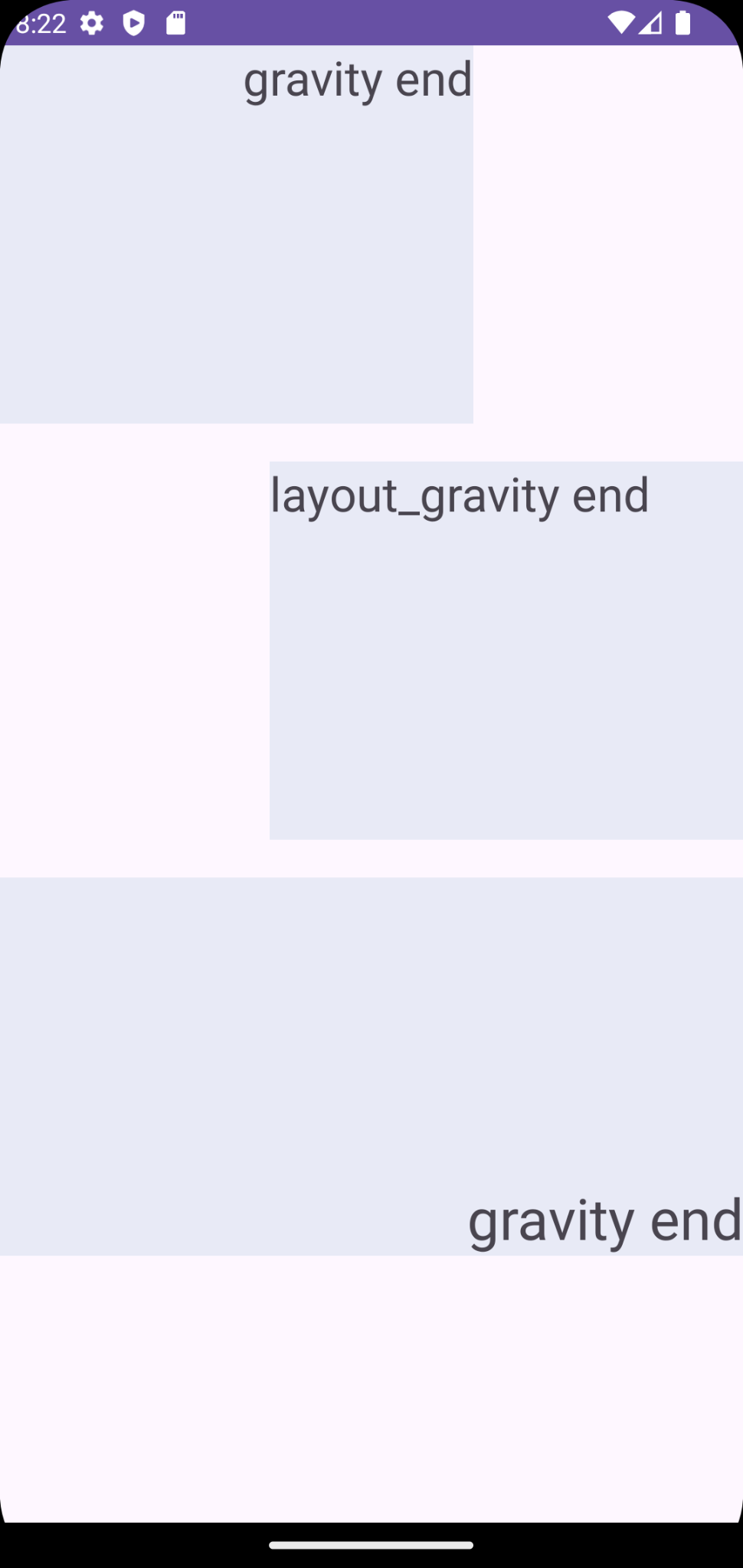
Например, поместим текст в самый низ в элементе TextView:





При необходимости мы можем комбинировать значения, разделяя их вертикальной чертой:





**Задание для самостоятельной работы:**

1. Создайте макет интерфейса соответствующий следующему макету:

