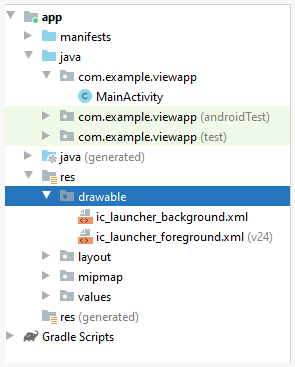
**Использование элементов для отображения изображений**

**Добавление изображений в проект**

Одним из наиболее распространенных источников ресурсов являются файлы изображений. Android поддерживает следующие форматы файлов изображений: .png (предпочтителен), .jpg (приемлем), .gif (нежелателен). Для графических файлов в проекте по умолчанию создана папка res/drawable. По умолчанию она уже содержит ряд файлов - пару файлов иконок:

****

При добавлении графических файлов в эту папку для каждого из них Android создает ресурс Drawable. После этого мы можем обратиться к ресурсу следующим образом в коде Java:

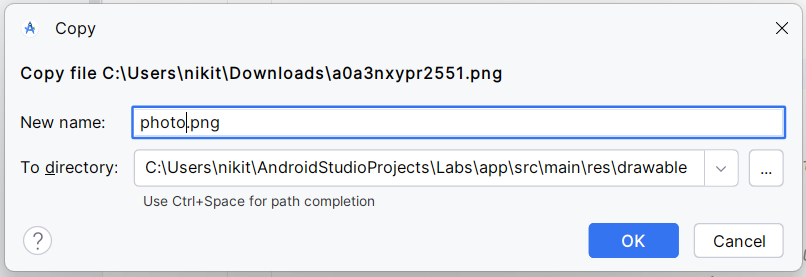
R.drawable.имя\_файла

Или в коде xml:

@[имя\_пакета:]drawable/имя\_файла

Добавьте в папку res/drawable какой-нибудь файл изображения. Скопируйте любое доступное изображение с расширением png или jpg и вставьте его в папку res/drawable.

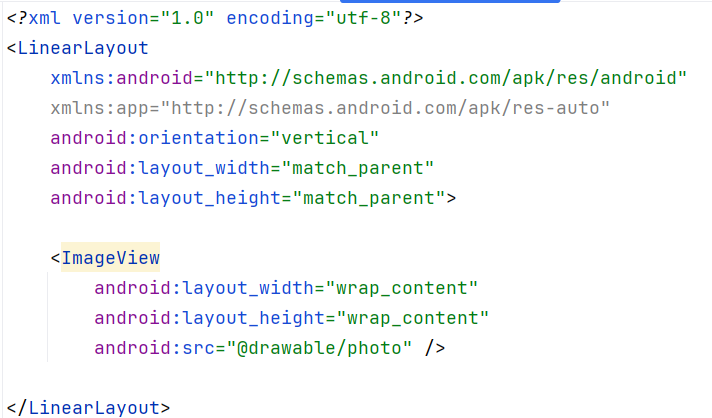
Далее будет предложено изменить название файла и изменить местоположение в которое файл будет скопирован.



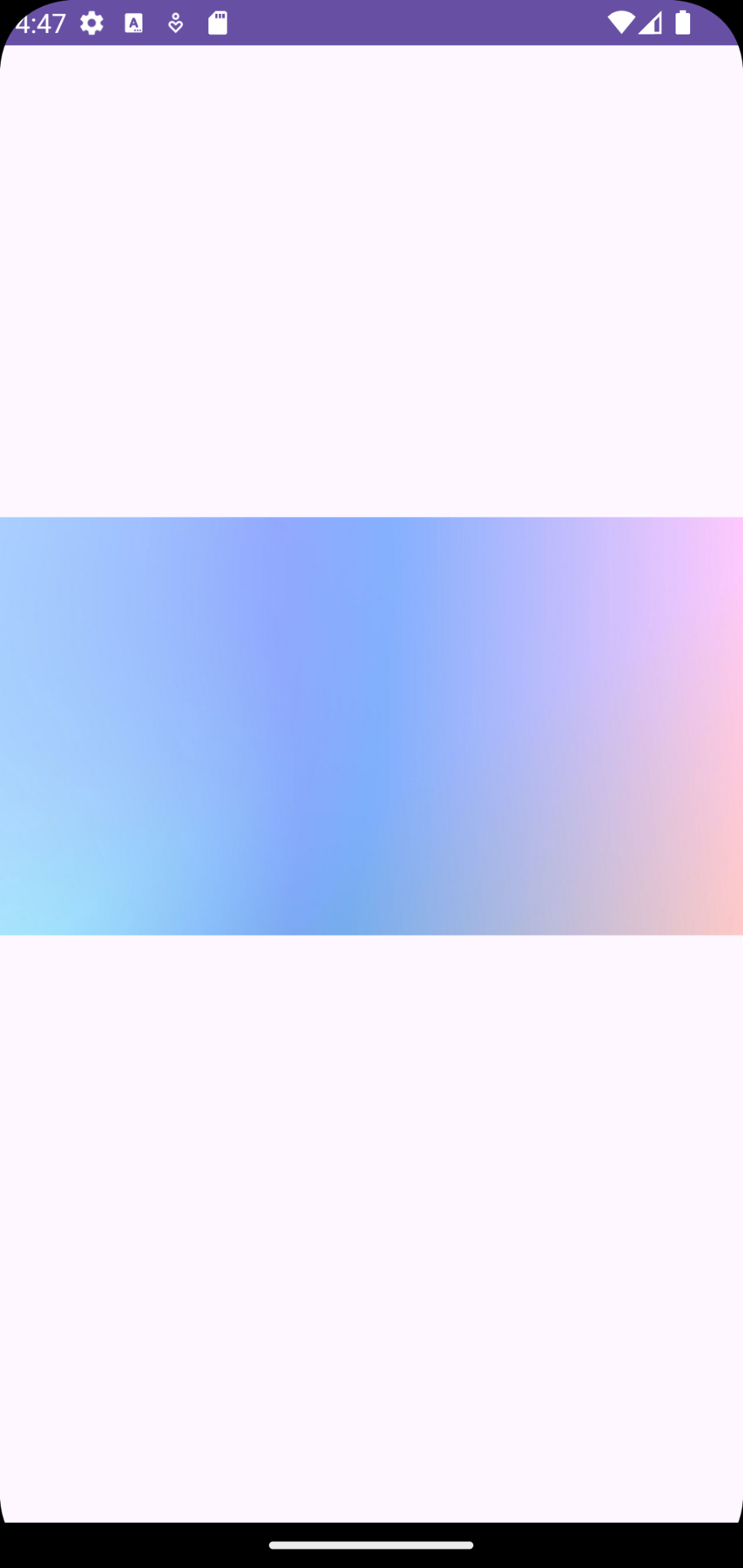
Сразу стоит учесть, что файл изображения будет добавляться в приложение, тем самым увеличивая его размер. Кроме того, большие изображения отрицательно влияют на производительность. Поэтому лучше использовать небольшие и оптимизированные (сжатые) графические файлы.

Для работы с изображениями можно использовать различные элементы, но непосредственно для вывода изображений предназначен ImageView.

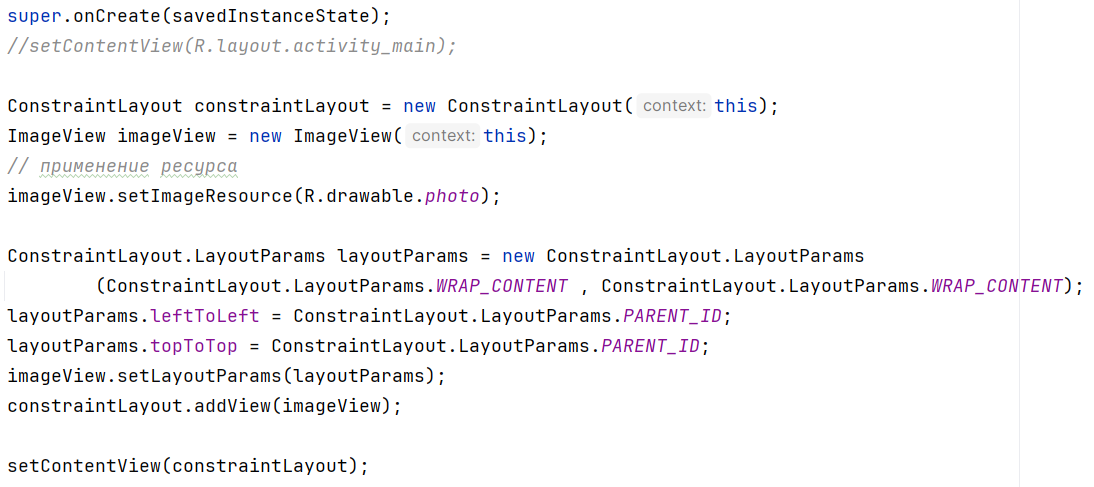
Создайте следующий макет интерфейса:



В примере для отображения файла в ImageView устанавливается атрибут android:src. В его значении указывается имя графического ресурса, которое совпадает с именем файла без расширения (вместо photo.png просто photo).

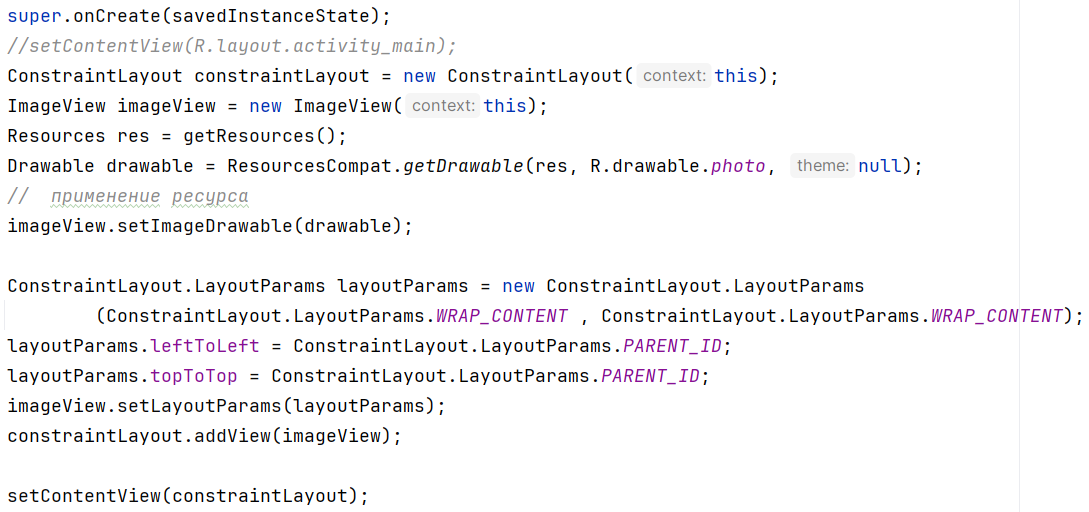


Приведем пример создания и настройки ImageView в программном коде:



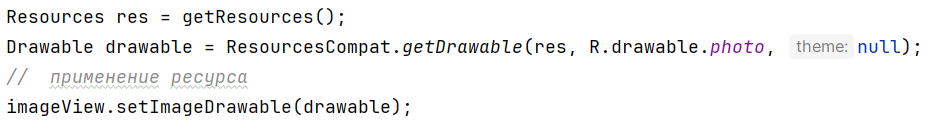
В приведенном примере ресурс drawable напрямую передается в метод imageView.setImageResource(), и таким образом устанавливается изображение.

В случае если потребуется как либо обработать ресурс перед использованием, необходимо сначала получить его как объект Drawable и затем использовать для выполнения задачи:

****

Для получения ресурса применяется метод ResourcesCompat.getDrawable(), в который передается объект Resources, идентификатор ресурса и тема. В данном случае тема нам не важна, поэтому для нее передаем значение null. Возвращается ресурс в виде объекта Drawable.

Затем, ресурс передается ImageView через метод setImageDrawable():



**Использование ImageView**

ImageView - элемент, который используется для отображения изображения (или любого другого файла графики) на экране.



ImageView обрабатывает всю загрузку и масштабирование изображения за вас. Обратите внимание на атрибут scaleType, который определяет, как изображения будут масштабироваться, чтобы соответствовать вашему макету. В примере с использованием scaleType "center" изображение будет отображаться в своем собственном разрешении и по центру, независимо от того, сколько места занимает элемент.

Само изображение:

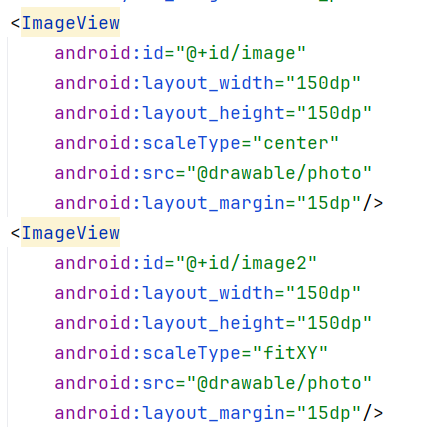


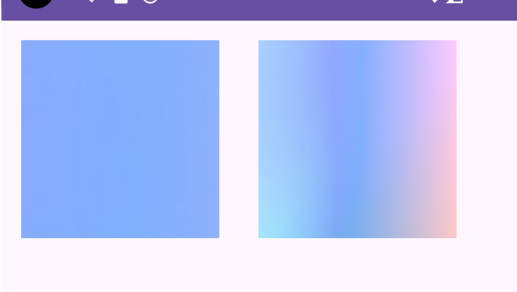
Отображение изображение с помощью ImageView из примера:



**Изменение размера элементов управления ImageView**

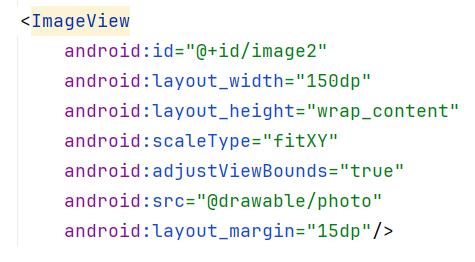
По умолчанию содержимое элемента управления ImageView имеет определенный размер равный размеру изображения. Они также могут быть ограничены атрибутами layout\_width и layout\_height:





На втором ImageView для атрибута scaleType было установлено значение fitXY, которое устанавливает высоту и ширину вверх или вниз, чтобы соответствовать указанным максимальным размерам.

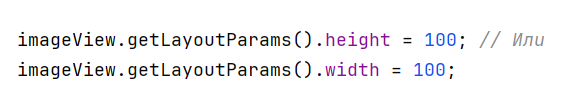
Однако исправление ширины и высоты означает, что пропорции исходного изображения, будут изменены. Вы можете воспользоваться параметром adjustViewBounds, чтобы сохранить это соотношение сторон. Однако вы должны либо разрешить регулировать высоту и / или ширину (т. е. Используя maxWidth и используя wrap\_content для измерения). В противном случае размеры не могут быть скорректированы для соответствия требуемому соотношению сторон.





Объединяя эти свойства вместе, вы можете управлять приблизительным размером изображения и по-прежнему корректировать его в соответствии с правильным соотношением сторон.

Вы также можем изменять размер ImageView во время выполнения в нашем исходном коде Java, изменяя ширину или высоту внутри getLayoutParams():



В определенных случаях изображение необходимо масштабировать, чтобы оно соответствовало ширине родительского элемента, а высоту необходимо отрегулировать пропорционально, для этого используйте расширенный класс ResizableImageView.

**Некоторые методы класса ImageView:**

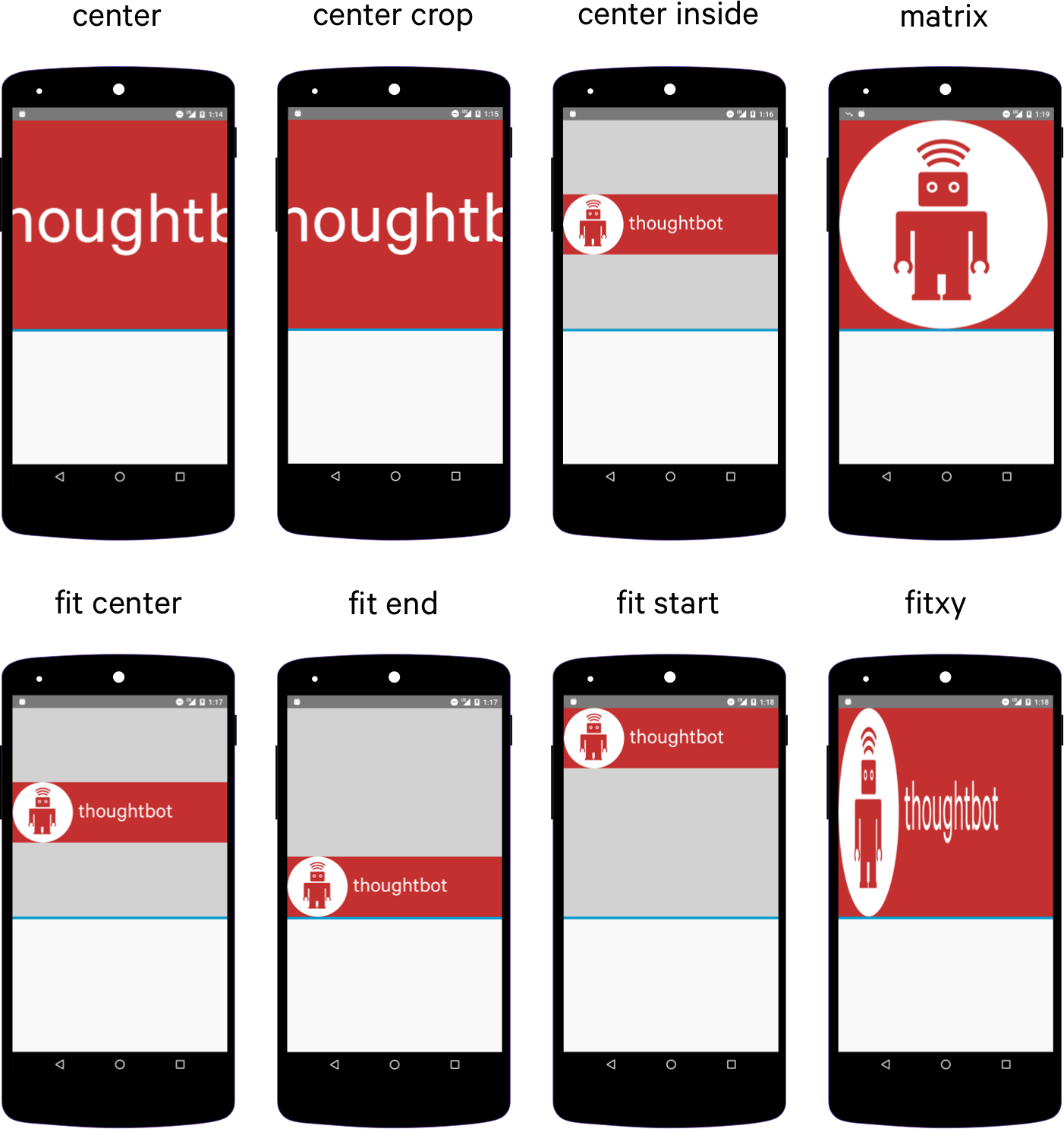
* getDrawable(): возвращает ресурс Drawable, который связан с данным ImageView (или null, если ресурс для ImageView не установлен)
* getScaleType(): возвращает значение перечисления ImageView.ScaleType, которое указывает, как масштабируется изображение относительно границ элемента ImageView
* setImageDrawable(Drawable drawable): устанавливает ресурс изображения с помощью объекта Drawable
* setImageResource(int resId): устанавливает ресурс изображения с помощью идентификатора ресурса Drawable
* setImageURI(Uri uri): устанавливает ресурс изображения с помощью адреса Uri этого ресурса
* setScaleType(ImageView.ScaleType scaleType): задает масштабирование изображения принимает значения scaleType вида: ImageView.ScaleType.CENTER, ImageView.ScaleType.CENTER\_INSIDE, ImageView.ScaleType.FIT\_XY и т.д.
* setImageAlpha(int alpha): задает прозрачность изображения - значение от 0.0 до 1.0

**Типы масштабирования**

Одним из способов регулировать отображение изображения в ImageView является атрибут scaleType. Ранее вы использовали fitXY значение вместе с adjustViewBounds для соответствия соотношению сторон изображения. Далее приведен список всех наиболее распространенных типов:

| **Тип** | **Описание** |
| --- | --- |
| center | Изображение отображается по центру представления без масштабирования. |
| centerCrop | Масштабирует изображение таким образом, чтобы размеры x и y были больше или равны элементу, сохраняя при этом аспектное соотношение сторон изображения, центрирует изображение. Если какая-то часть не помещается в пределы экрана, то она обрезается. |
| centerInside | Масштабирует изображение так, чтобы оно помещалось внутри элемента, сохраняя при этом соотношение сторон изображения. Если изображение уже меньше, чем элемент, то работает как center. Изображение не может быть больше элемента. |
| fitCenter | Масштабирует изображение так, чтобы оно помещалось внутри элемента, сохраняя при этом соотношение сторон изображения. По крайней мере, одна ось будет точно соответствовать элементу, а результат будет центрирован. |
| fitStart | То же, что и в fitCenter, но выровнено по верхнему краю элемента (верхний при портретной ориентации и левый край - при альбомной). |
| fitEnd | То же, что и в fitCenter, но выровнено по нижнему краю представления. (нижный при портретной ориентации и правый край - при альбомной) |
| fitXY | Масштабирует размеры x и y, чтобы они точно соответствовали размеру изображения, без сохранения аспектного отношения между шириной и высотой. |
| matrix | Масштабирует изображение с помощью предоставленного класса Matrix. Матрица может быть предоставлена с помощью метода setImageMatrix. Класс Matrix может использоваться для применения к изображению преобразований, таких как поворот изображения. |

**Примечание:** Тип fitXY scale позволяет вам установить точный размер изображения в вашем макете. Однако помните о возможных искажениях изображения из-за масштабирования. Если вы создаете приложение для просмотра фотографий, вы, вероятно, захотите использовать типы масштаба center или fitCenter.



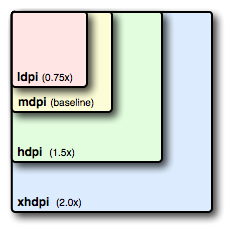
**Поддержка нескольких форматов размерности**

Поскольку мобильные телефоны имеют так много экранов разного размера, разрешения и плотности, существует мощная система для выбора правильного ресурса изображения для устройства. Для каждой категории плотности устройства существуют определенные папки, включая: ldpi (низкий), mdpi (средний), hdpi (высокий) и xhdpi (сверхвысокий). Обратите внимание, что в каждом приложении есть папки для отрисовки изображений, например, drawable-mdpi которые предназначены для "средних точек на один дюйм".

Чтобы создавать альтернативные растровые изображения для разных плотностей, вы должны соблюдать масштабное соотношение 3:4:6:8 между четырьмя обобщенными плотностями. Обратитесь к таблице ниже:

| **Плотность** | **DPI** | **Пример** | **Масштабирование** | **Значение в px** |
| --- | --- | --- | --- | --- |
| ldpi | 120 | Galaxy Y | 0.75x | 1dp = 0.75px |
| mdpi | 160 | Galaxy Tab | 1.0x | 1dp = 1px |
| hdpi | 240 | Galaxy S II | 1.5x | 1dp = 1.5px |
| xhdpi | 320 | Nexus 4 | 2.0x | 1dp = 2px |
| xxhdpi | 480 | Nexus 5 | 3.0x | 1dp = 3px |
| xxxhdpi | 640 | Nexus 6 | 4.0x | 1dp = 4px |

Это означает, что если вы создаете изображение 100x100 для mdpi (1x baseline), то вы должны сгенерировать тот же ресурс размером 150x150 для hdpi (1.5x), изображение 200x200 для устройств xhdpi (2.0x), изображение 300x300 для xxhdpi (3.0x) и изображение 75x75 для устройств ldpi (0.75x).



**Задание для самостоятельной работы:**

1. Разработайте приложение с интерфейсом соответствующим заданному макету. Приложение позволяет пользователю изменить изображение на другое доступное случайным образом или изменить его масштабирование. Добавьте в приложение 5-ть картинок, которые будут случайно меняться между собой при нажатии на кнопку “Случайная картинка”. Текстовые поля выводят актуальную информацию по изображению.

|  |  |
| --- | --- |