Работа с изображениями

Компонент Image

Для выведения изображений в приложении в Jetpack Compose предназначен компонент Image. Он имеет ряд версий. Первая версия компонента:

```
@Composable
fun Image(
    bitmap: ImageBitmap,
    contentDescription: String?,
    modifier: Modifier = Modifier,
    alignment: Alignment = Alignment.Center,
    contentScale: ContentScale = ContentScale.Fit,
    alpha: Float = DefaultAlpha,
    colorFilter: ColorFilter? = null,
    filterQuality: FilterQuality = DefaultFilterQuality
): @Composable Unit
```

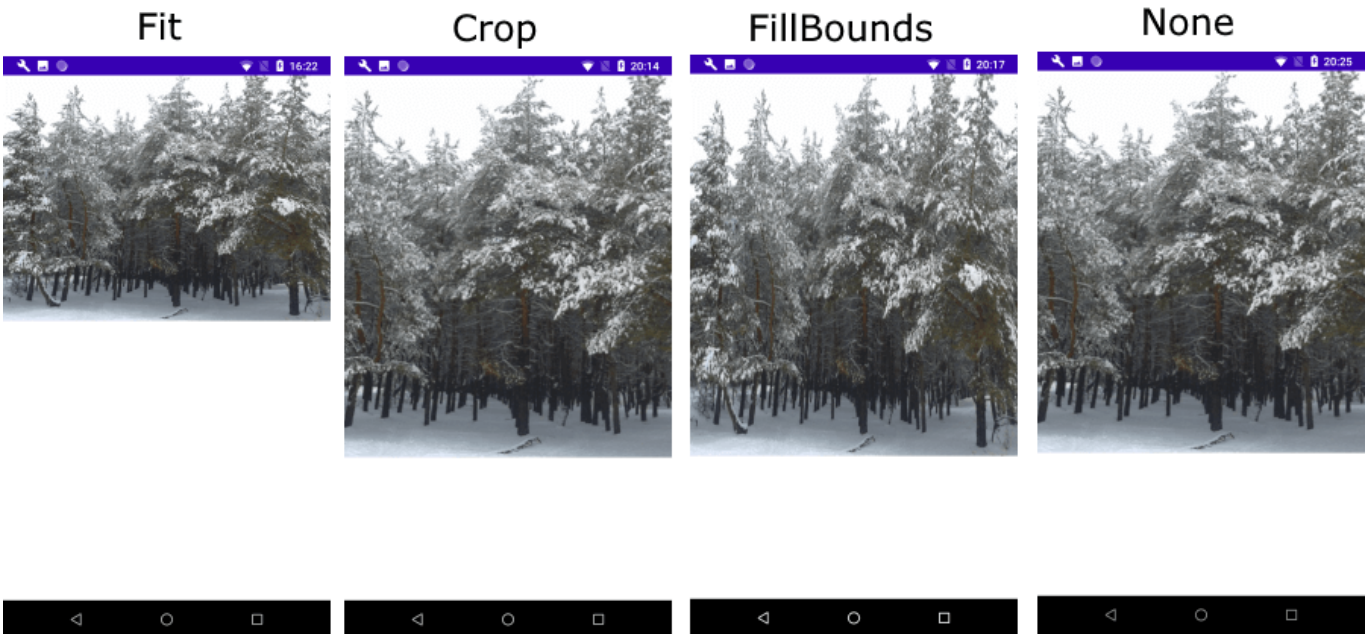
Параметры функции компонента:

- bitmap: объект типа ImageBitmap, которое собственно представляет изображение для отрисовки
- contentDescription: представляет строковое описание для изображения, которое применяется сервисами accessibility в служебных целях
- modifier: представляет объект Modifier, который определяет модификаторы компонента
- alignment: представляет объект типа Alignment, которое задает выравнивание изображения. Значение по умолчанию - Alignment.Center.
- contentScale: объект типа ContentScale, который принцип масштабирования изображения. По умолчанию равно ContentScale.Fit

  Может принимать следующие значения:

  - ContentScale.Crop: масштабирует изображение с сохранением аспектного отношения (отношение высоты и ширины) таким образом, что ширина и высота оказываются равными или больше сторон контейнера.
  - ContentScale.FillBounds: неравномерно масштабирует изображение для полного заполнения пространства контейнера.
  - ContentScale.FillHeight: масштабирует изображение с сохранением аспектного отношения таким образом, что высота изображения равна высоте контейнера.
  - ContentScale.FillWidth: масштабирует изображение с сохранением аспектного отношения таким образом, что ширина изображения равна ширине контейнера.

- ContentScale.Fit: масштабирует изображение с сохранением аспектного отношения (отношение высоты и ширины) таким образом, что ширина и высота оказываются равными или меньше сторон контейнера.
- ContentScale.Inside: масштабирует изображение с сохранением аспектного отношения (отношение высоты и ширины) таким образом, чтобы вместить изображение внутри контейнера, если ширина и(или) высота изобажения больше ширины и(или) высоты контейнера.
- ContentScale.None: масштабирование отсутствует



- alpha: задает прозрачность изображения в виде значения типа Float
- colorFilter: устанавливает применяемые к изображению цветовые фильтры в виде объекта ColorFilter
- filterQuality: задает алгоритм выборки пикселей из изображения. Представляет объект типа FilterQuality и по умолчанию имеет значение FilterQuality.Low

Вторая версия Image:

```
@Composable
fun Image(
    imageVector: ImageVector,
    contentDescription: String?,
    modifier: Modifier = Modifier,
    alignment: Alignment = Alignment.Center,
    contentScale: ContentScale = ContentScale.Fit,
    alpha: Float = DefaultAlpha,
    colorFilter: ColorFilter? = null
): @Composable Unit
```

Здесь применяются практические те же параметры, кроме первого - imageVector, который устанавливает отображаемый рисунок и представляет объект ImageVector.

Третья версия:

```
@Composable
fun Image(
    painter: Painter,
    contentDescription: String?,
    modifier: Modifier = Modifier,
    alignment: Alignment = Alignment.Center,
    contentScale: ContentScale = ContentScale.Fit,
    alpha: Float = DefaultAlpha,
    colorFilter: ColorFilter? = null
): @Composable Unit
```

Здесь опять же меняется только первый параметр, который задает изображение. В данном случае это параметр painter, который представляет объект Painter.

Какой бы вариант компонента Image мы не выбрали, в любом случае необходимо установить первый параметр, который задает изображение, и параметр contentDescription.

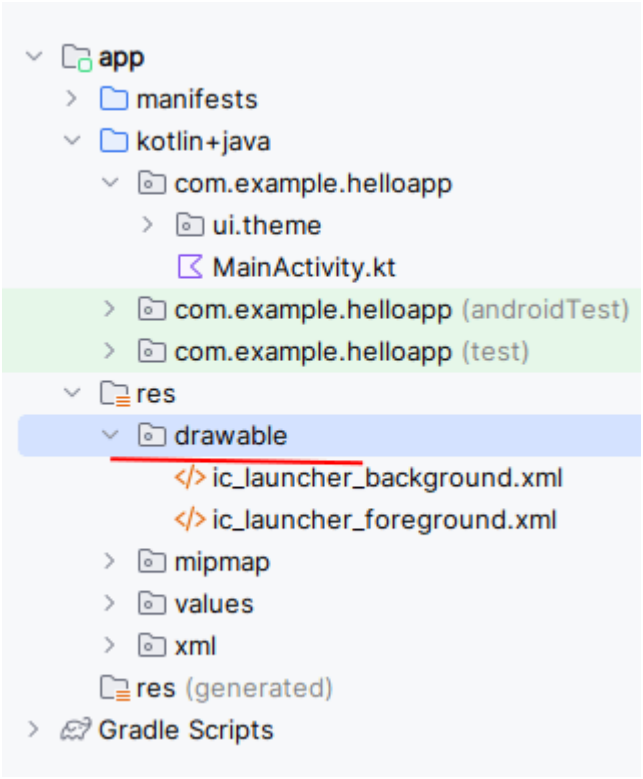
Простейший пример:

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.painter.ColorPainter

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Image(painter = ColorPainter(Color.Red), contentDescription = "Красный
прямоугольник")
        }
    }
}
```

В данном случае для установки изображения применяется параметр painter, который в качестве значения принимает объект ColorPainter. ColorPainter по сути представляет полотно, закрашенное цветом. Цвет передается в качестве параметра в ColorPainter. То есть в данном случае мы фактически увидем компонент Image, окрашенный в красный цвет.

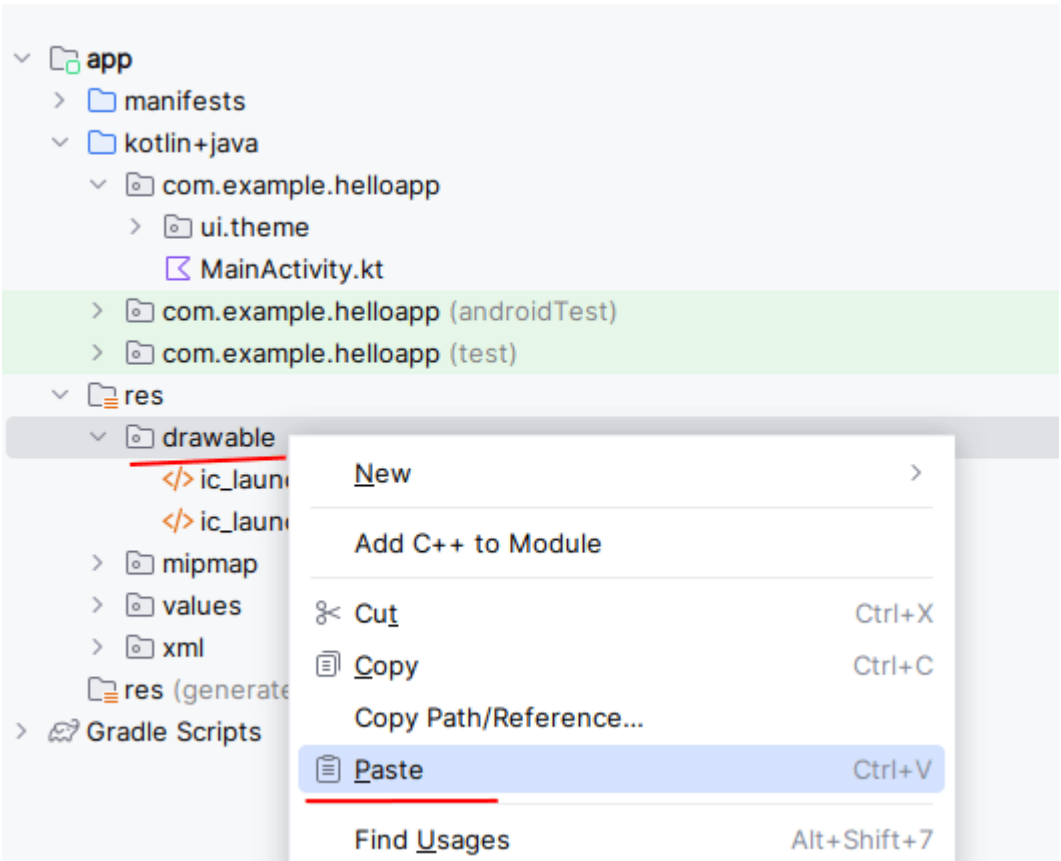
Ресурсы изображений и ImageBitmap

Изображение, которое мы хотим отобразить в нашем приложении, может располагаться в различных местах - это может быть сетевой проект, но также изображение может храниться в самом приложении. Для хранения изображений в проекте предназначена папка res/drawable:

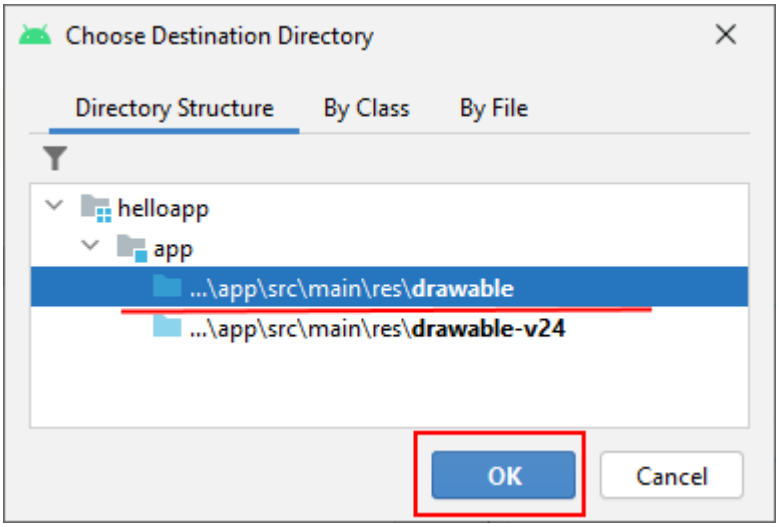


По умолчанию в этой папке уже имеется определения векторной графики в виде файлов ic_launcher_background.xml и ic_launcher_foreground.xml, которые применяются для создания иконок приложения.

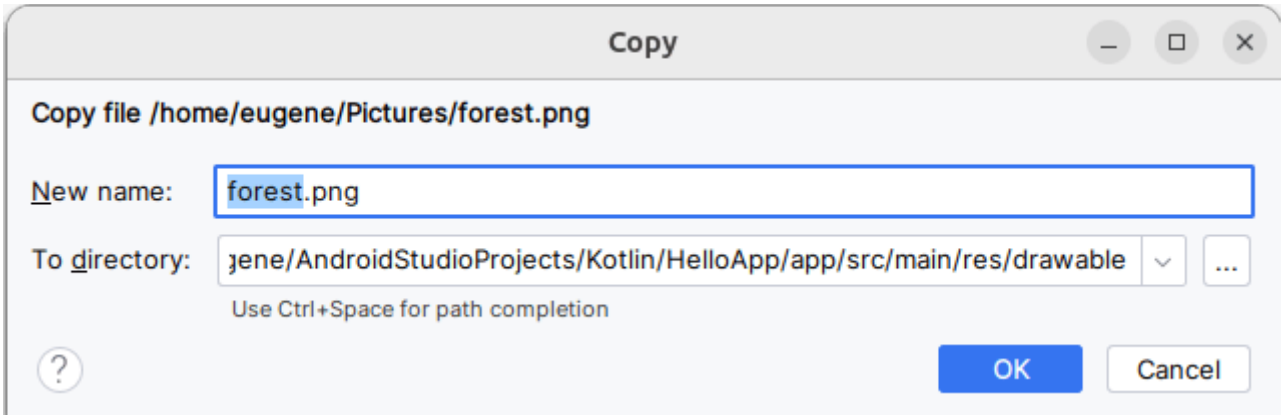
Теперь добавим в эту папку какой-нибудь файл изображения. Для этого скопируем файл изображения с расширением png или jpg и с помощью стандартной комбинации клавиш Ctrl+V добавим его в папку res/drawable. Также можно нажать на папку правой кнопкой мыши и в появившемся меню выбрать пункт Paste:



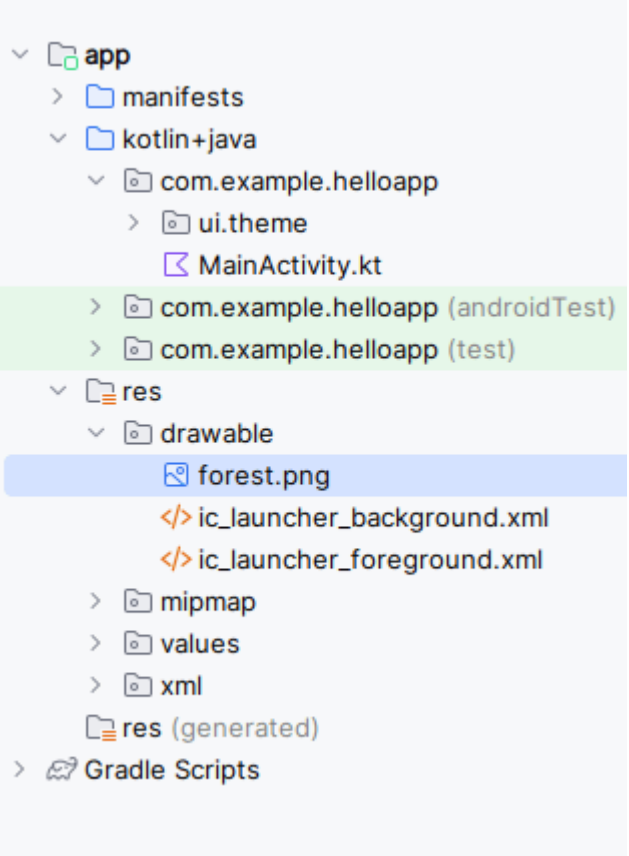
Далее нам будет предложено выбрать папку - drawable или drawable-24. Для добавления обычных файлов изображений выберем drawable:



При копировании файла нам будет предложено установить для него новое имя (по умолчанию подставляется текущее имя файла)



Можно изменить название файла, а можно оставить так как есть. В моем случае файл называется forest.png. И затем нажмем на кнопку OK. И после этого в папку drawable будет добавлен выбранный нами файл изображения.



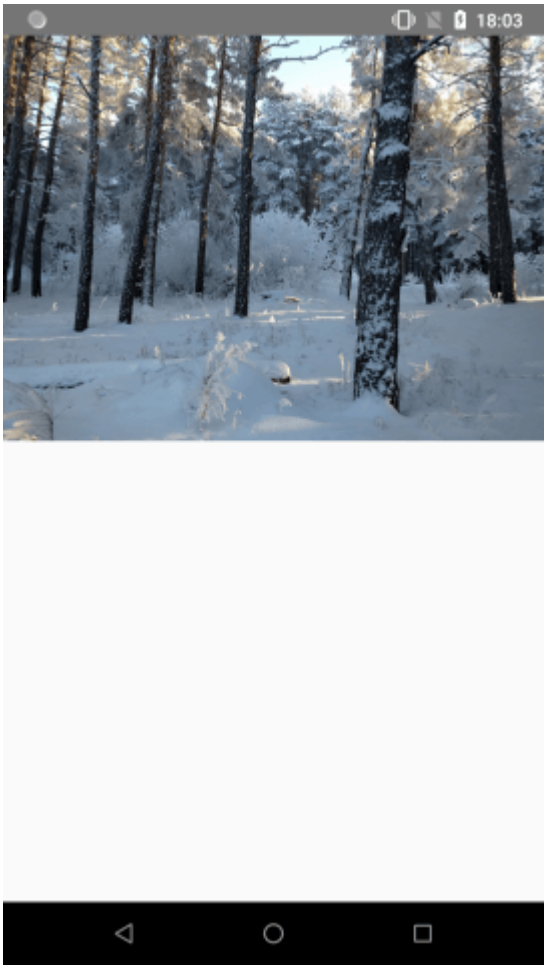
При добавлении графических файлов в эту папку для каждого из них Android создает ресурс Drawable. После этого мы можем обратиться к ресурсу следующим образом в коде Kotlin:

```
R.drawable.имя_файла
```

Для отображения растровых изображений, а именно файлов png и jpg, в компоненте Image применяется интерфейс ImageBitmap. Этот интерфейс предоставляет статический метод imageResource(идентификатор_ресурса) для получения объекта ImageBitmap из ресурса drawable.

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.ui.graphics.ImageBitmap
import androidx.compose.ui.res.imageResource

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Image(
                bitmap = ImageBitmap.imageResource(R.drawable.forest),
                contentDescription = "Зимний лес"
            )
```

```
        }
    }
}
```



BitmapPainter

Также для выведения изображения можно использовать класс Painter, а точнее его класс-наследник BitmapPainter, который отрисовывает изображение. Данное изображение передается в виде объекта ImageBitmap в конструктор BitmapPainter в качестве параметра:

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.ui.graphics.ImageBitmap
import androidx.compose.ui.graphics.painter.BitmapPainter
import androidx.compose.ui.res.imageResource

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Image(
```

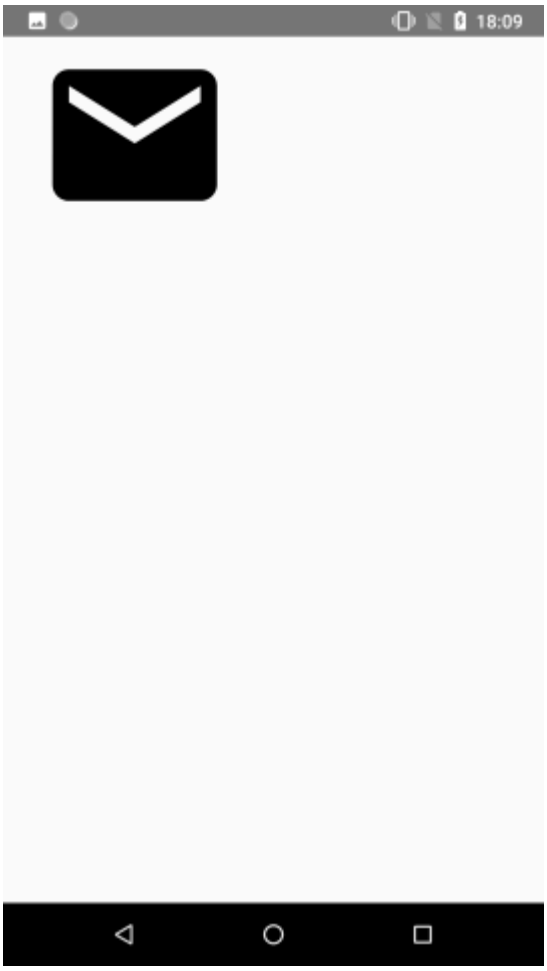
```
                painter =
BitmapPainter(ImageBitmap.imageResource(R.drawable.forest)),
                contentDescription = "Зимний лес"
            )
        }
    }
}
```

Векторная графика и ImageVector

Для отображения векторной графики в компоненте Image у этого компонента применяется параметр imageVector, который представляет объект класса ImageVector.

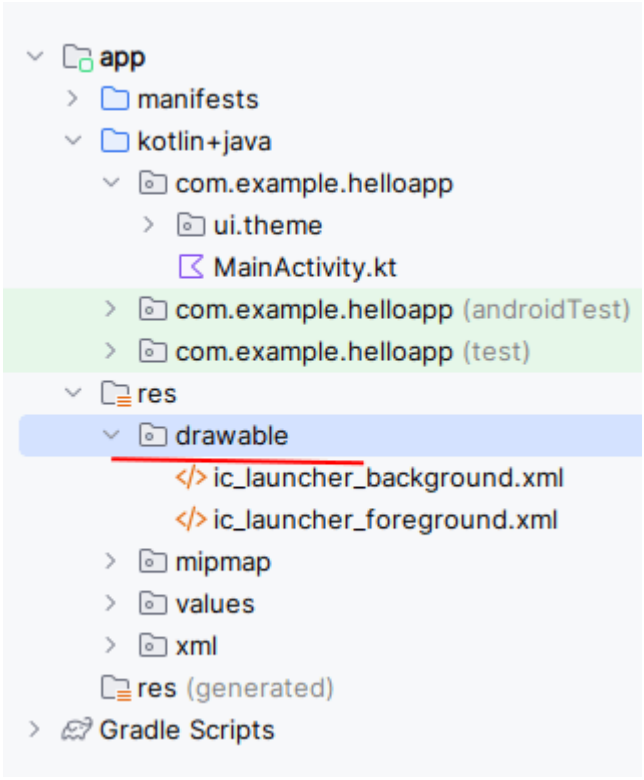
Например, встроенные в Jetpack Compose иконки как раз представляют векторную графику, которую мы можем вывести в компоненте Image:

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.size
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Email
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Image(
                imageVector = Icons.Filled.Email,
                contentDescription = "Значок электронной почты",
                modifier = Modifier.size(200.dp, 150.dp)
            )
        }
    }
}
```



Ресурсы векторной графики

Также мы можем использовать ImageVector для загрузки векторной графики, которая хранится в ресурсах приложения. Для хранения векторной графики, как и вообще изображений, в проекте предназначена папка res/drawable:

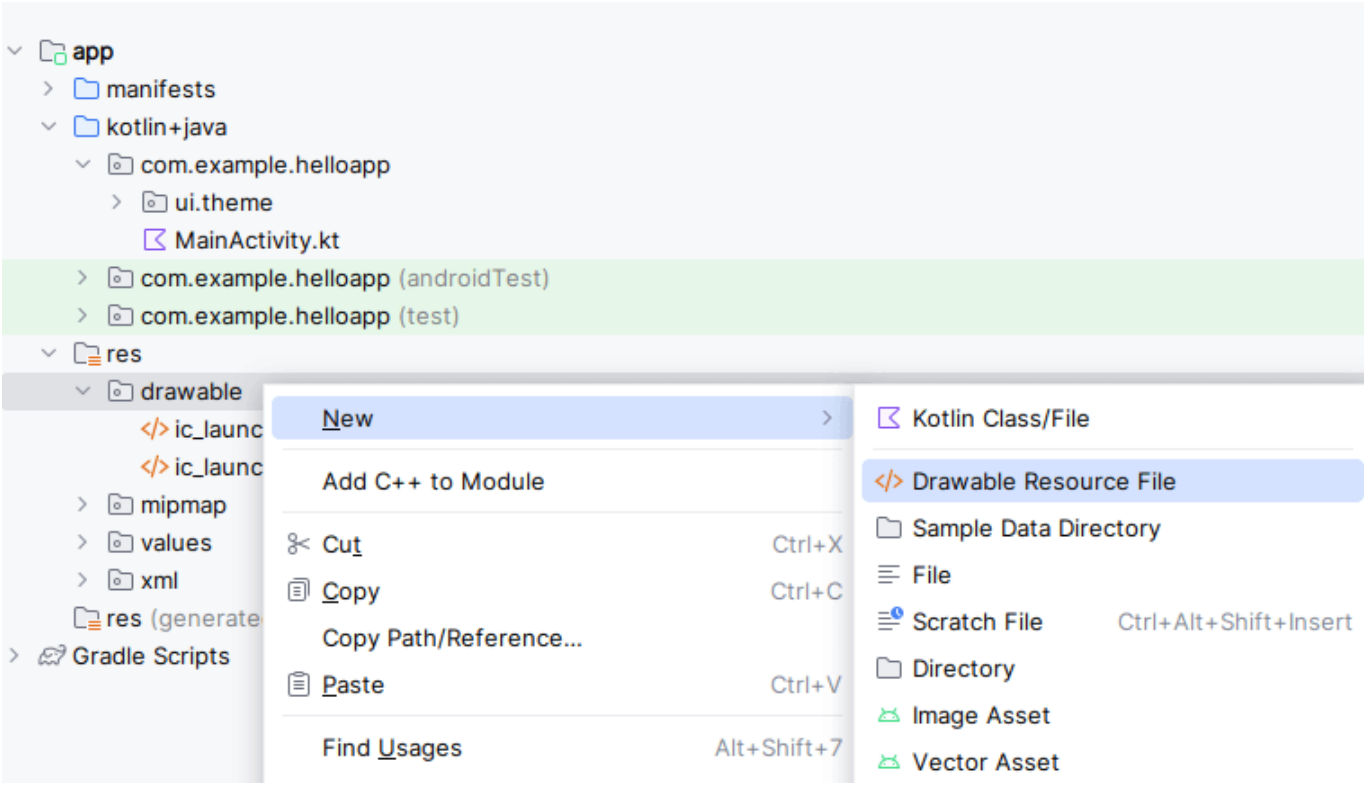


По умолчанию в этой папке уже имеется ресурсы векторной графики в виде файлов ic_launcher_background.xml и ic_launcher_foreground.xml, которые применяются для создания иконок приложения.

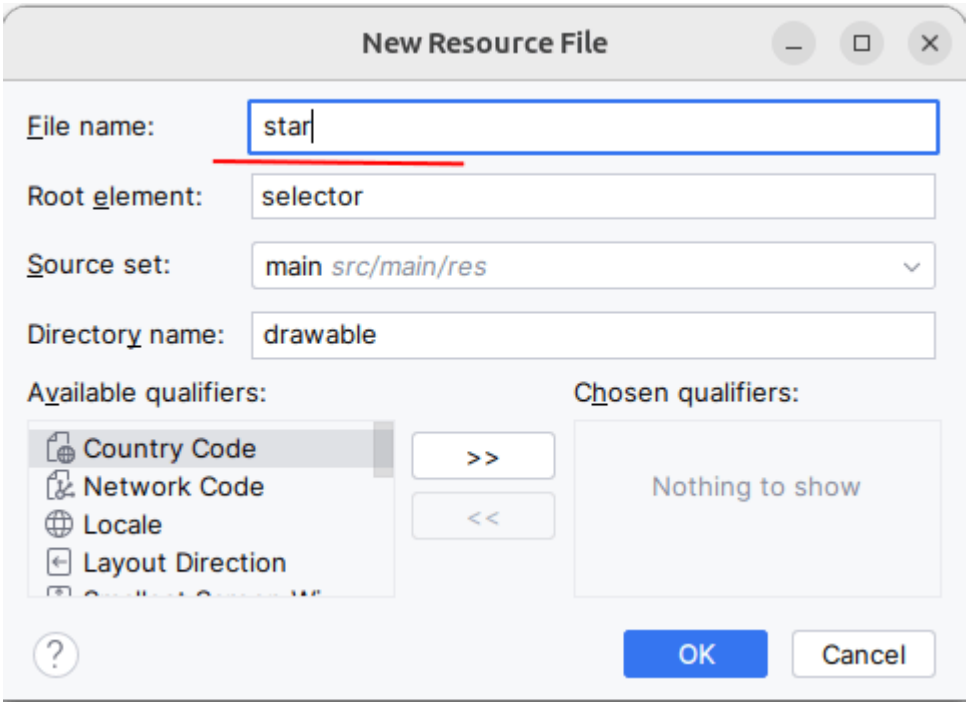
Для загрузки ресурса векторной графики и создания из него объекта ImageVector применяется функция ImageVector.vectorResource(), в которую передается идентификатор загружаемого ресурса. Например, загрузим имеющийся по умолчанию файл ic_launcher_background.xml

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.ui.graphics.vector.ImageVector
import androidx.compose.ui.res.vectorResource

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Image(
                imageVector =
ImageVector.vectorResource(R.drawable.ic_launcher_background),
                contentDescription = "Android"
            )
        }
    }
}
```

Для примера определим свой ресурс векторной графики. Для этого в папку res/drawable добавим новый ресурс. Для этого нажмем правой кнопкой мыши на папку res/drawable и в появившемся контекстном меню выберем пункт New -> Drawable Resource File:

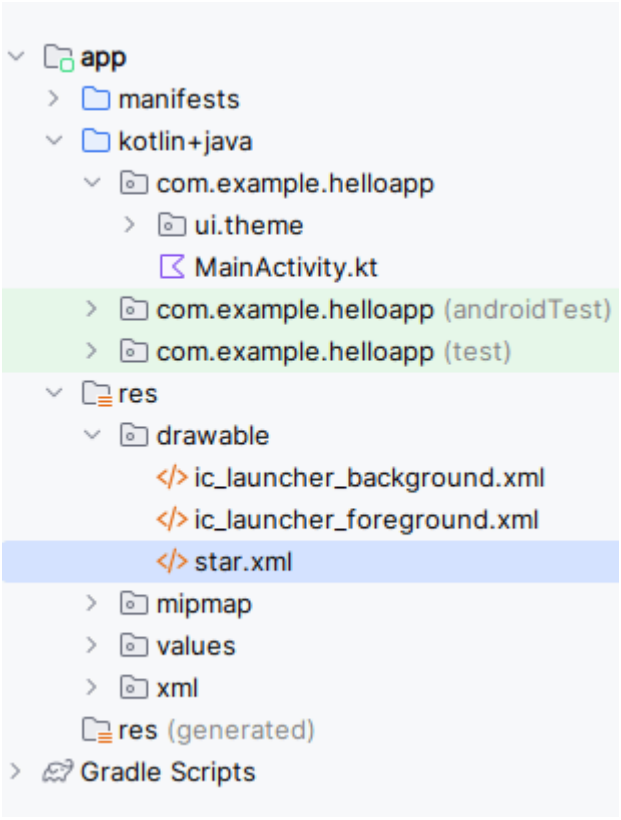


Далее в появившемся окошке в поле File name введем star - это будет название файла:



Остальные настройки оставим по умолчанию и нажмем на OK.

После этого в папку res/drawable будет добавлен файл star.xml:



Изменим содержимое этого файла на следующий код:

```
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:aapt="http://schemas.android.com/aapt"
    android:viewportWidth="260"
    android:viewportHeight="245"
    android:width="260dp"
    android:height="245dp">
    <path
        android:pathData="M56 237l74 -228 74 228L10 96h240"
        android:fillColor="#aa0000" />
</vector>
```

Я не буду подробно останавливаться на том, как создавать подобную графику, но вкратце данная графика представляет красную звезду. Для простоты можно взять какое-нибудь изображение в формате svg, который довольно распространен, и конвертировать его в данный формат с помощью онлайн-конвертеров.

Теперь изменим код в файле MainActivity.kt для загрузки этой звезды:

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.ui.graphics.vector.ImageVector
```

```
import androidx.compose.ui.res.vectorResource

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Image(
                imageVector = ImageVector.vectorResource(R.drawable.star),
                contentDescription = "Красная звезда"
            )
        }
    }
}
```