Практическая работа 3. Добавление изображений в приложение

Прежде чем начать

В этом руководстве вы узнаете, как добавить изображения в приложение с помощью Image composable.

Необходимые условия

- Базовые знания о том, как создать и запустить приложение в Android Studio.
- Базовые знания о том, как добавлять элементы пользовательского интерфейса, такие как текстовые композиты.

Что вы узнаете

- Как добавить изображение или фотографию в приложение для Android.
- Как отобразить изображение в приложении с помощью композита Image.
- Лучшие практики использования ресурсов String.

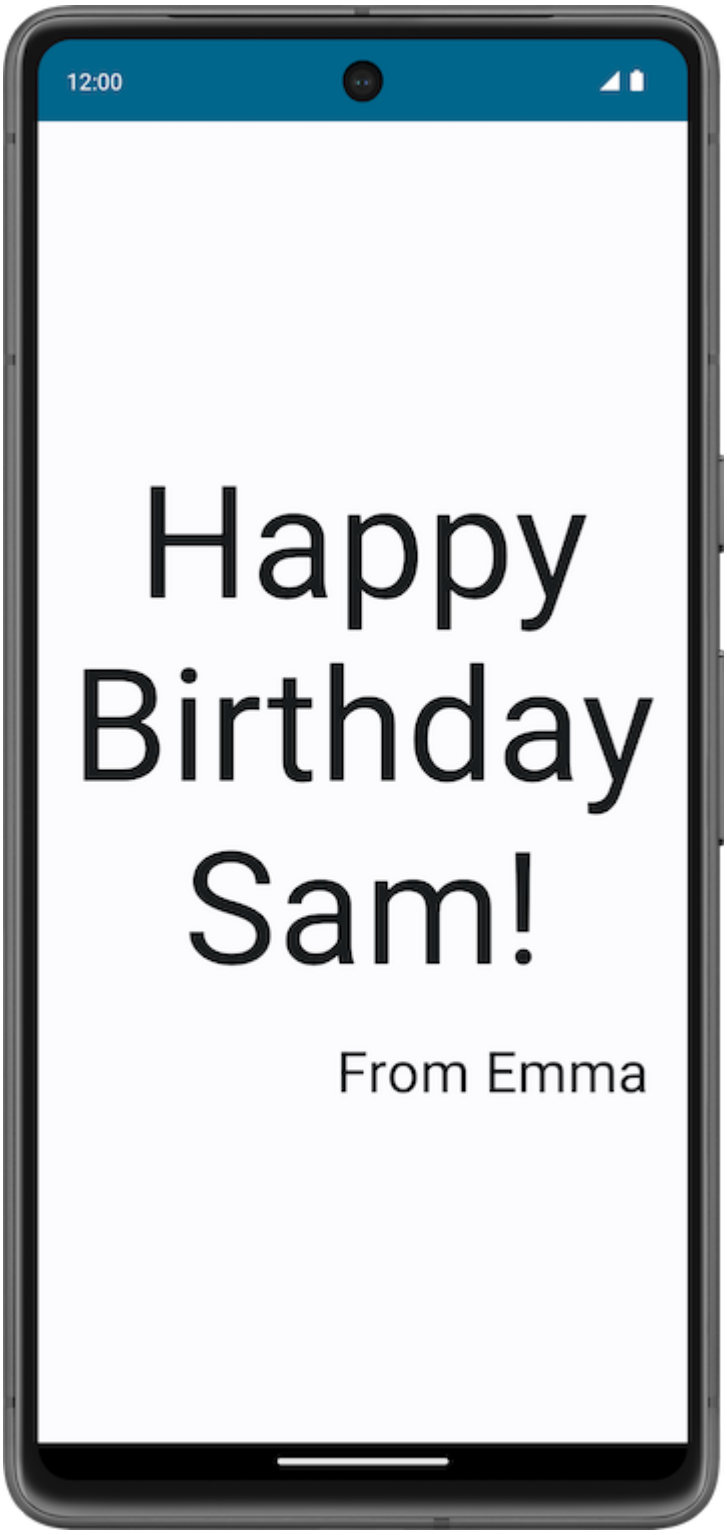
Что вы создадите

- Усовершенствуйте приложение `Happy Birthday`, добавив в него изображение.

Настройте ваше приложение

Откройте проект `Happy Birthday` из предыдущей практической работы в Android Studio.

Когда вы запустите приложение, оно должно выглядеть как на этом скриншоте.

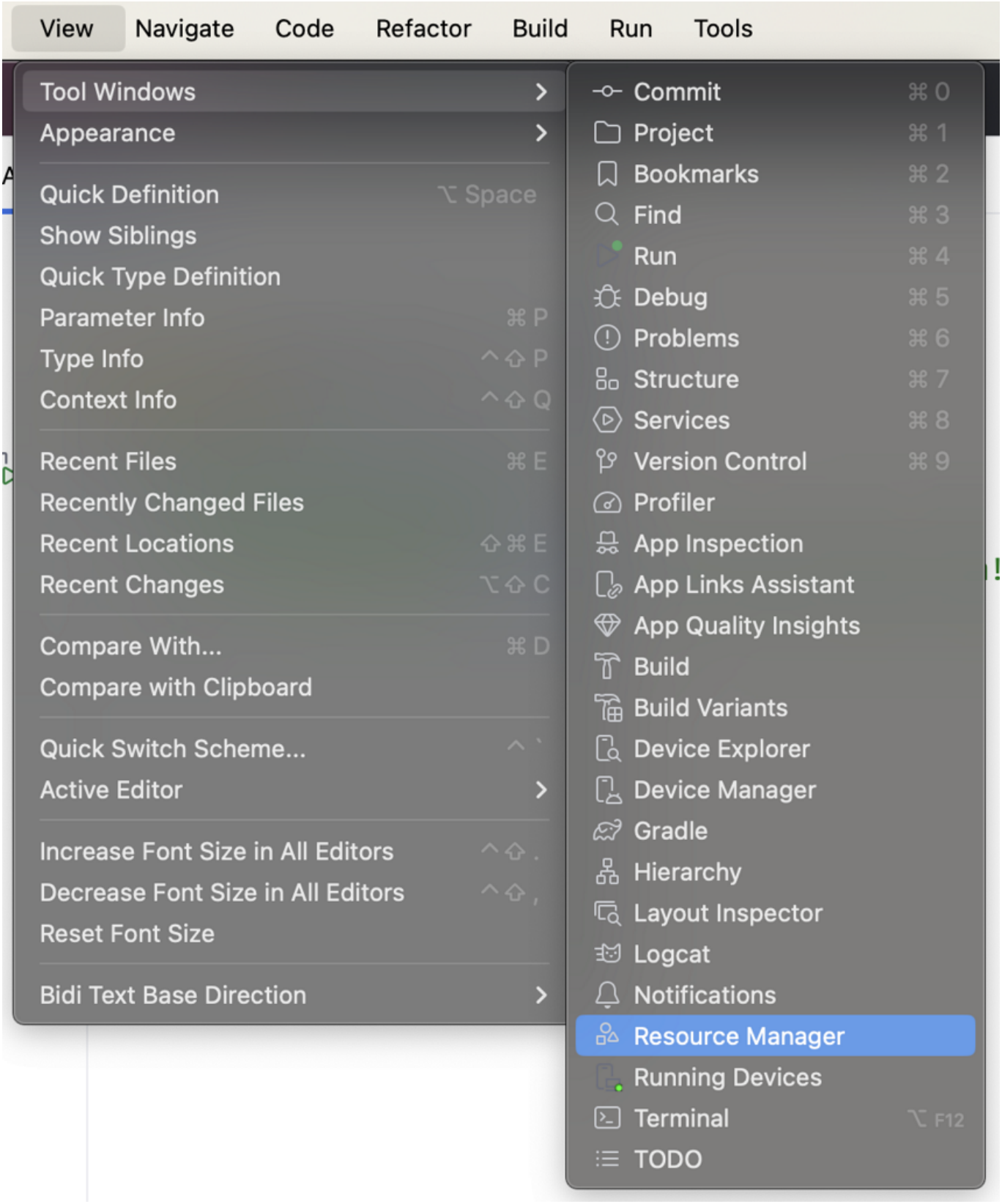


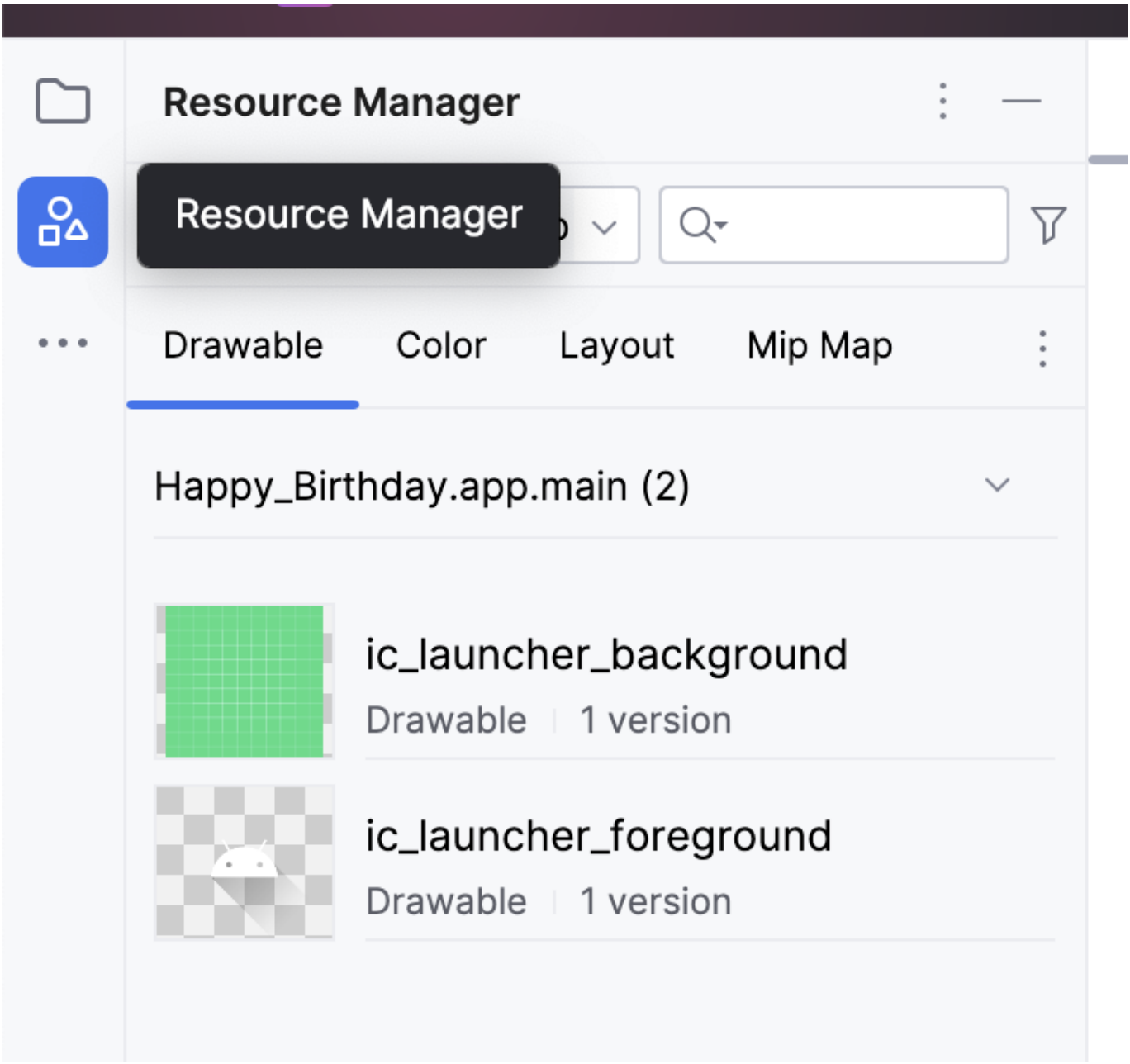
Добавьте изображение в свой проект

В этом задании вы загрузите изображение из ресурсов и добавите его в приложение «С днем рождения».

- Щелкните правой кнопкой мыши на изображении, а затем сохраните файл на компьютере под именем `androidparty.png`.
- Запомните, куда вы сохранили изображение.

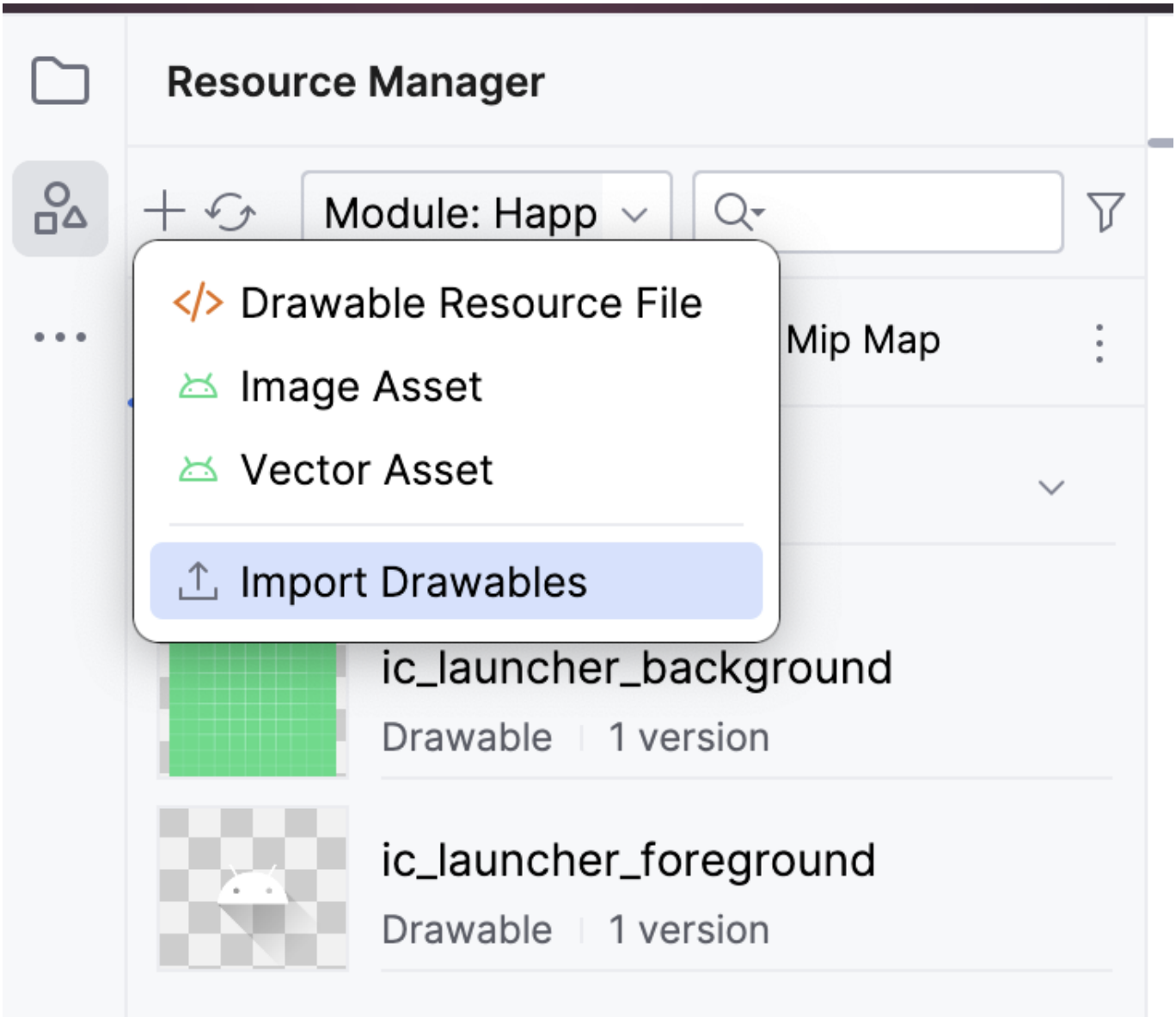
В Android Studio нажмите `View > Tool Windows > Resource Manager` или перейдите на вкладку `Resource Manager` рядом с окном `Project`.


View
Navigate
Code
Refactor
Build
Run
Tools
Tool Windows
Appearance
Quick Definition ⌥ Space
Show Siblings
Quick Type Definition
Parameter Info ⌘ P
Type Info ^ ⇧ P
Context Info ^ ⇧ Q
Recent Files ⌘ E
Recently Changed Files
Recent Locations ⇧ ⌘ E
Recent Changes ⌥ ⇧ C
Compare With... ⌘ D
Compare with Clipboard
Quick Switch Scheme... ^ `
Active Editor
Increase Font Size in All Editors ^ ⇧ .
Decrease Font Size in All Editors ^ ⇧ ,
Reset Font Size
Bidi Text Base Direction
Commit ⌘ 0
Project ⌘ 1
Bookmarks ⌘ 2
Find ⌘ 3
Run ⌘ 4
Debug ⌘ 5
Problems ⌘ 6
Structure ⌘ 7
Services ⌘ 8
Version Control ⌘ 9
Profiler
App Inspection
App Links Assistant
App Quality Insights
Build
Build Variants
Device Explorer
Device Manager
Gradle
Hierarchy
Layout Inspector
Logcat
Notifications
Resource Manager
Running Devices
Terminal ⌥ F12
TODO



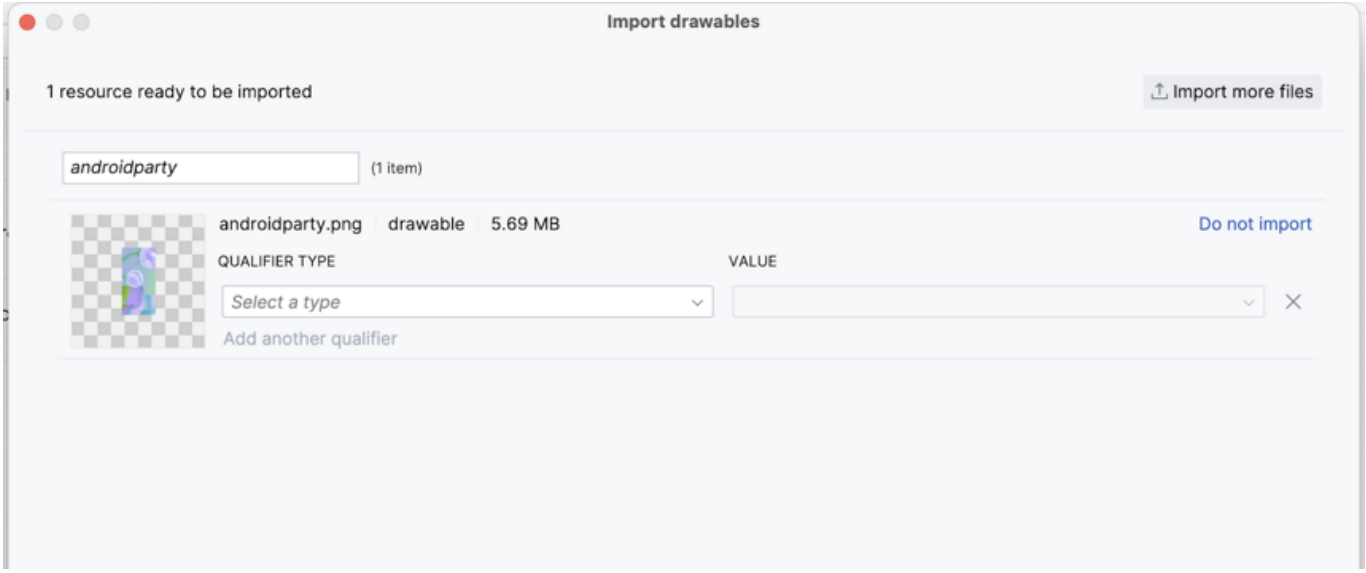
> Примечание: Менеджер ресурсов - это окно инструментов, позволяющее импортировать, создавать, управлять и использовать ресурсы в вашем приложении.

Нажмите + (Добавить ресурсы в модуль) > Импорт рисунков.

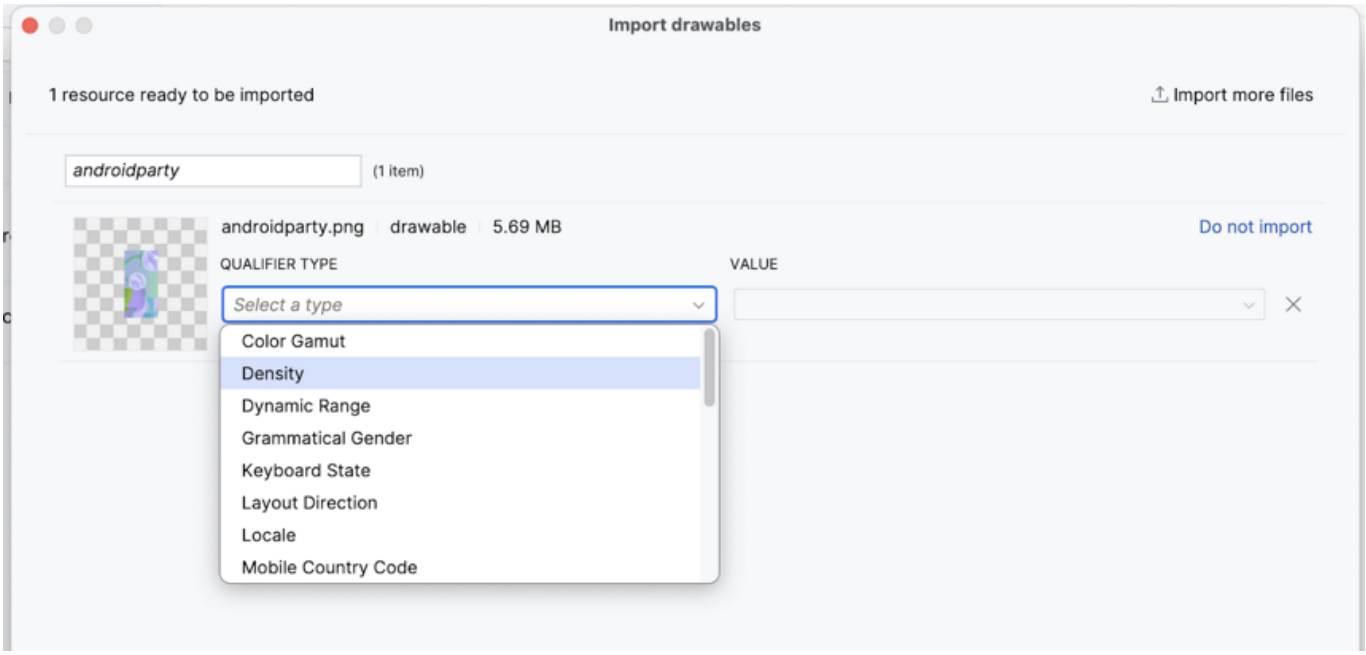


В браузере файлов выберите файл изображения, который вы загрузили, а затем нажмите кнопку Открыть.

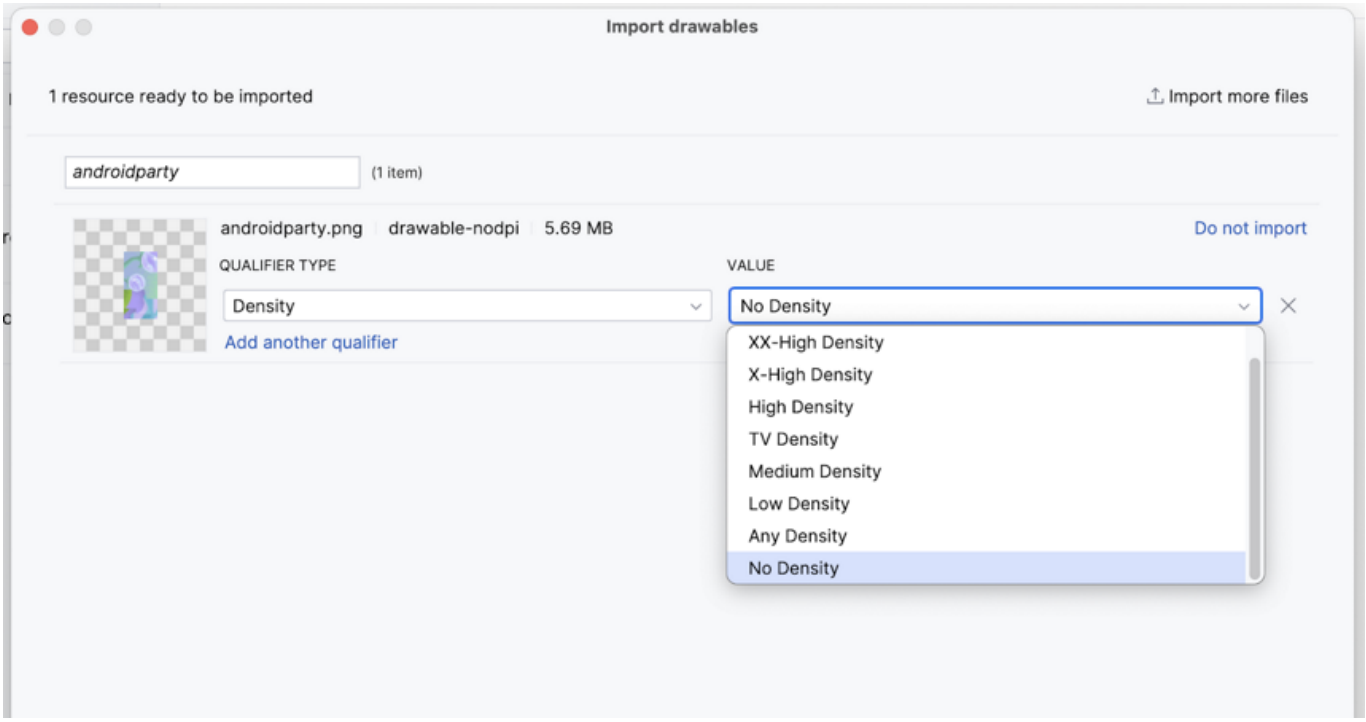
Это действие открывает диалоговое окно Импорт чертежей.



Android Studio показывает предварительный просмотр изображения. В раскрывающемся списке ТИП КАЧЕСТВА выберите Плотность. О том, зачем это нужно, вы узнаете из следующего раздела.



Выберите `No Density` в списке `VALUE`.



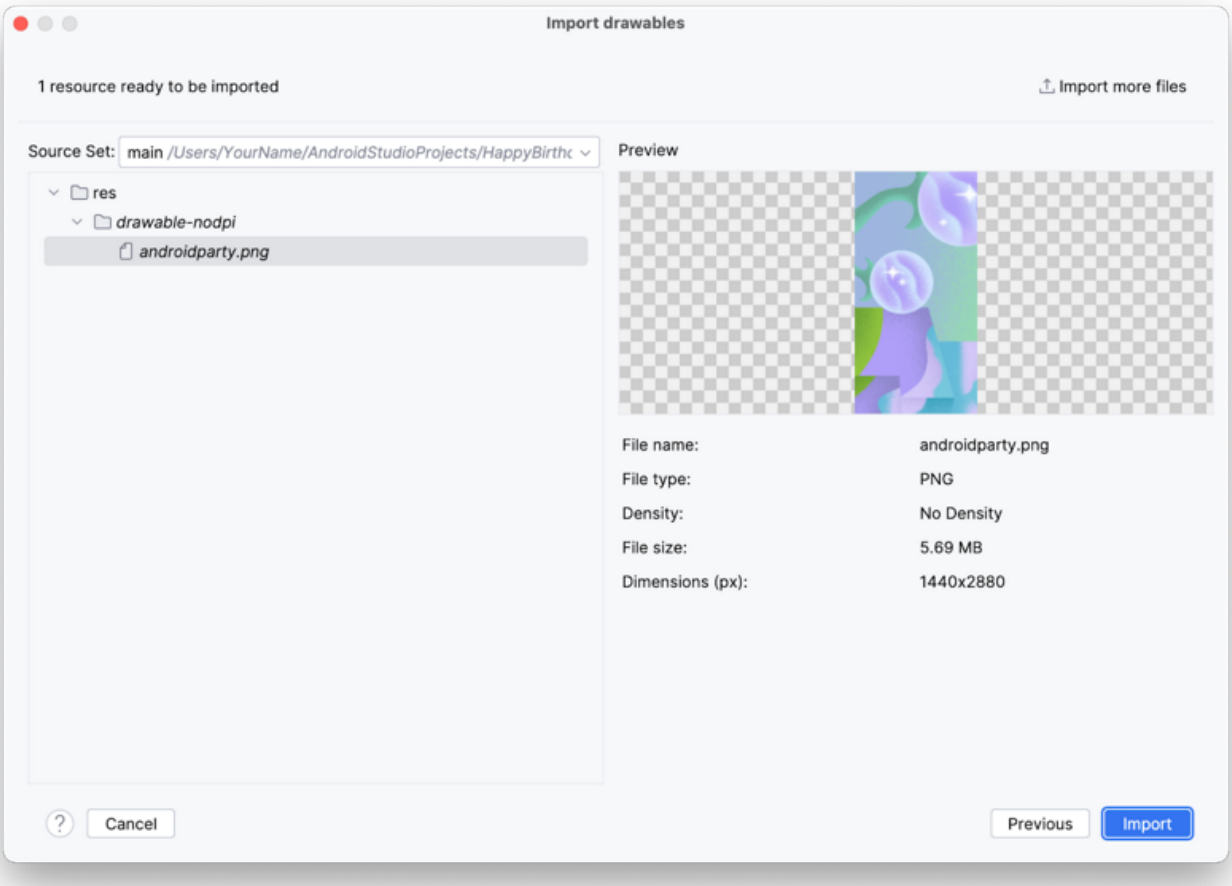
Устройства Android имеют разные размеры экрана (телефоны, планшеты, телевизоры и т. д.), и их экраны также отличаются по размеру пикселей. То есть, если на одном устройстве на квадратный дюйм приходится 160 пикселей, то на другом на том же пространстве помещается 480 пикселей. Если не учитывать эти различия в плотности пикселей, система может изменить масштаб изображений, что приведет к размытым изображениям, большим изображениям, потребляющим слишком много памяти, или изображениям с неправильным размером.

При изменении размера изображений, превышающих возможности системы Android, возникает ошибка нехватки памяти. Для фотографий и фоновых изображений, таких как текущее изображение

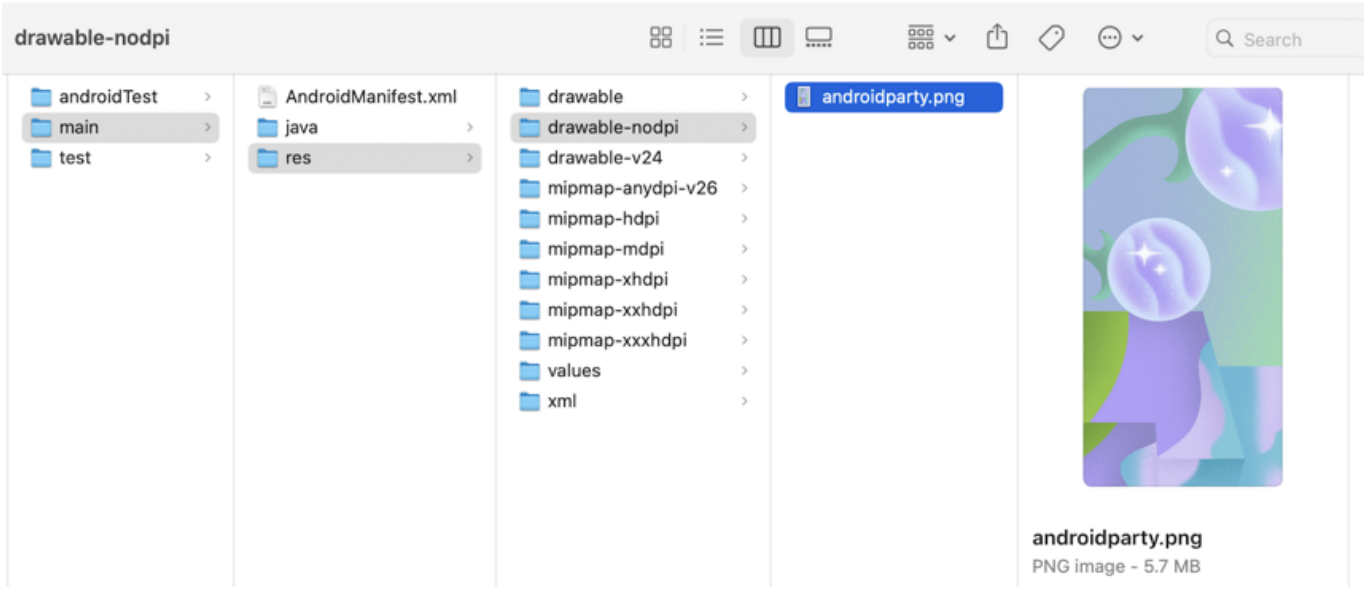
androidparty.png, следует поместить их в папку drawable-nodpi, что остановит поведение изменения размера.

Дополнительные сведения о плотности пикселей см. в разделе Поддержка различных плотностей пикселей.

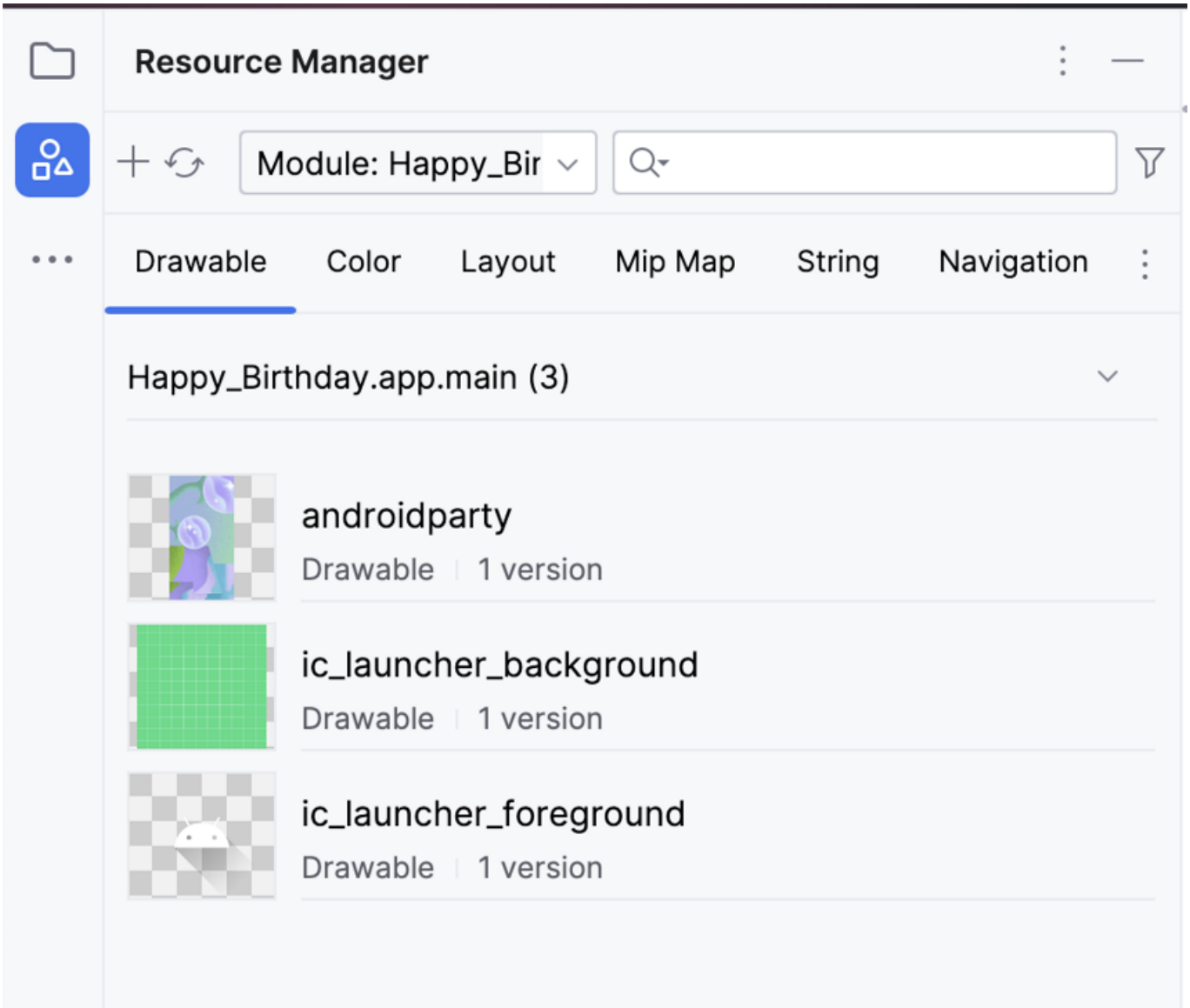
Нажмите кнопку Далее. Android Studio показывает структуру папок, в которые будет помещено изображение. Обратите внимание на папку drawable-nodpi. Нажмите Импорт(C).



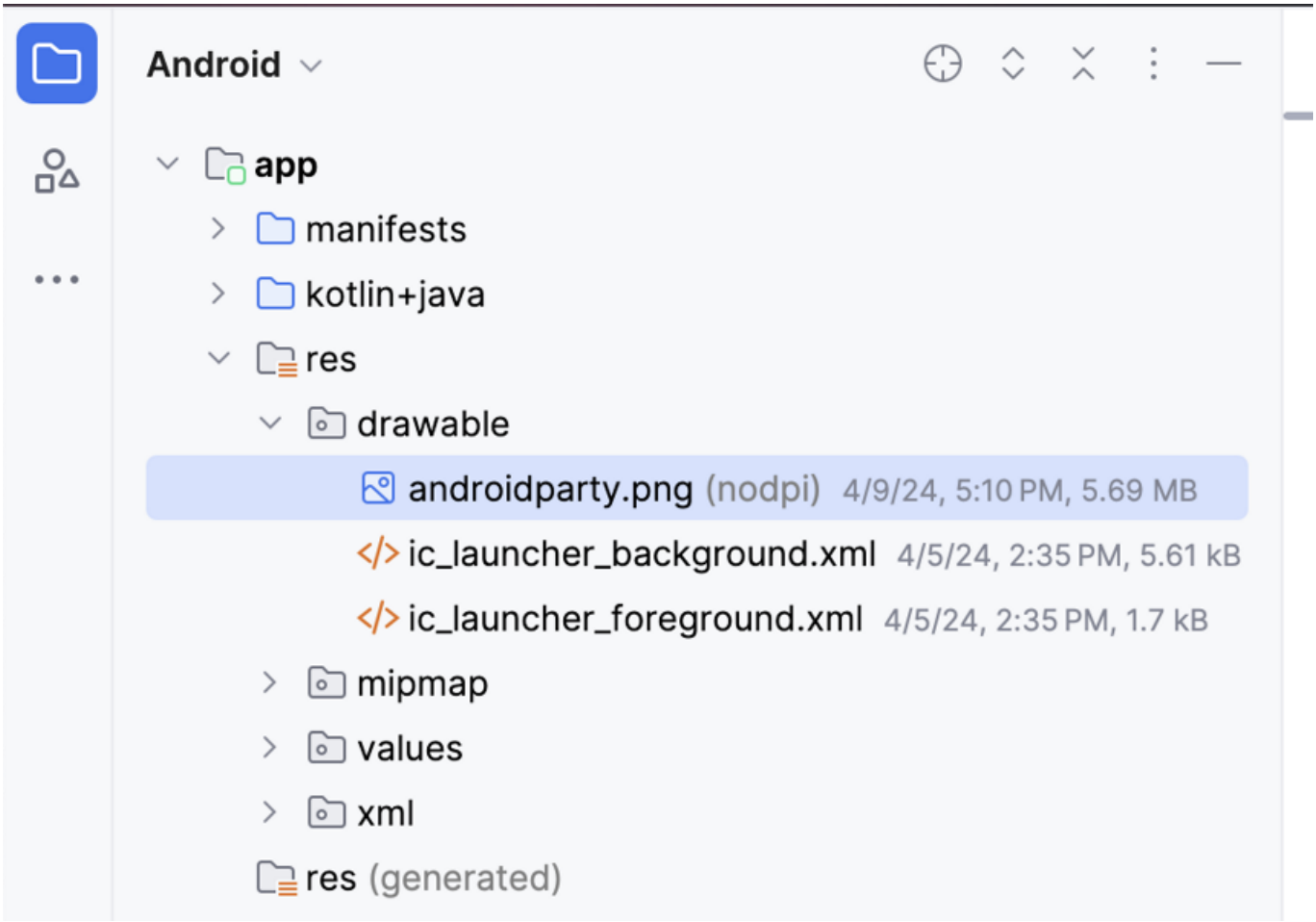
Android Studio создает папку drawable-nodpi и помещает в нее ваше изображение.В представлении проекта Android Studio имя ресурса отображается как androidparty.png (nodpi).В файловой системе компьютера Android Studio создала бы папку drawable-nodpi.



Если изображение успешно импортировано, Android Studio добавляет его в список на вкладке Drawable.В этот список входят все ваши изображения и иконки для приложения.Теперь вы можете использовать это изображение в своем приложении.



Переключитесь обратно в вид проекта, щелкните View > Tool Windows > Project или перейдите на вкладку Project в крайнем левом углу. Нажмите app > res > drawable, чтобы убедиться, что изображение находится в папке drawable.



3. Добавление композитного изображения

Чтобы отобразить изображение в приложении, ему нужно место для отображения. Точно так же, как вы используете компонент Text для отображения текста, вы можете использовать компонент Image для отображения изображения.

В этом задании вы добавите Image composable в свое приложение, установите его изображение на загруженное вами изображение, расположите его и отрегулируйте его размер так, чтобы оно заполняло экран.

Добавьте функцию composable для добавления изображения В файле MainActivity.kt добавьте составную функцию GreetingImage() после функции GreetingText(). Передайте функции GreetingImage() два параметра String: один - message - для поздравления с днем рождения, другой - from - для вашей подписи.

```
@Composable
fun GreetingImage(message: String, from: String) {
}
```

Каждая композитная функция должна принимать необязательный параметр Modifier. Модификаторы указывают элементу пользовательского интерфейса, как он должен располагаться, отображаться или вести себя в родительском макете. Добавьте еще один параметр в составную функцию GreetingImage().

```
@Composable
fun GreetingImage(message: String, from: String, modifier: Modifier = Modifier) {
}
```

Ресурсы в Jetpack Compose Ресурсы - это дополнительные файлы и статический контент, которые использует ваш код, например растровые изображения, строки пользовательского интерфейса, инструкции по анимации и многое другое. Более подробную информацию о ресурсах в Android вы найдете в разделе Обзор ресурсов приложения.

Вы всегда должны отделять ресурсы приложения, такие как изображения и строки, от вашего кода, чтобы вы могли поддерживать их независимо. Во время выполнения Android использует соответствующий ресурс в зависимости от текущей конфигурации. Например, вам может потребоваться предоставить различный макет пользовательского интерфейса в зависимости от размера экрана или различные строки в зависимости от языковых настроек.

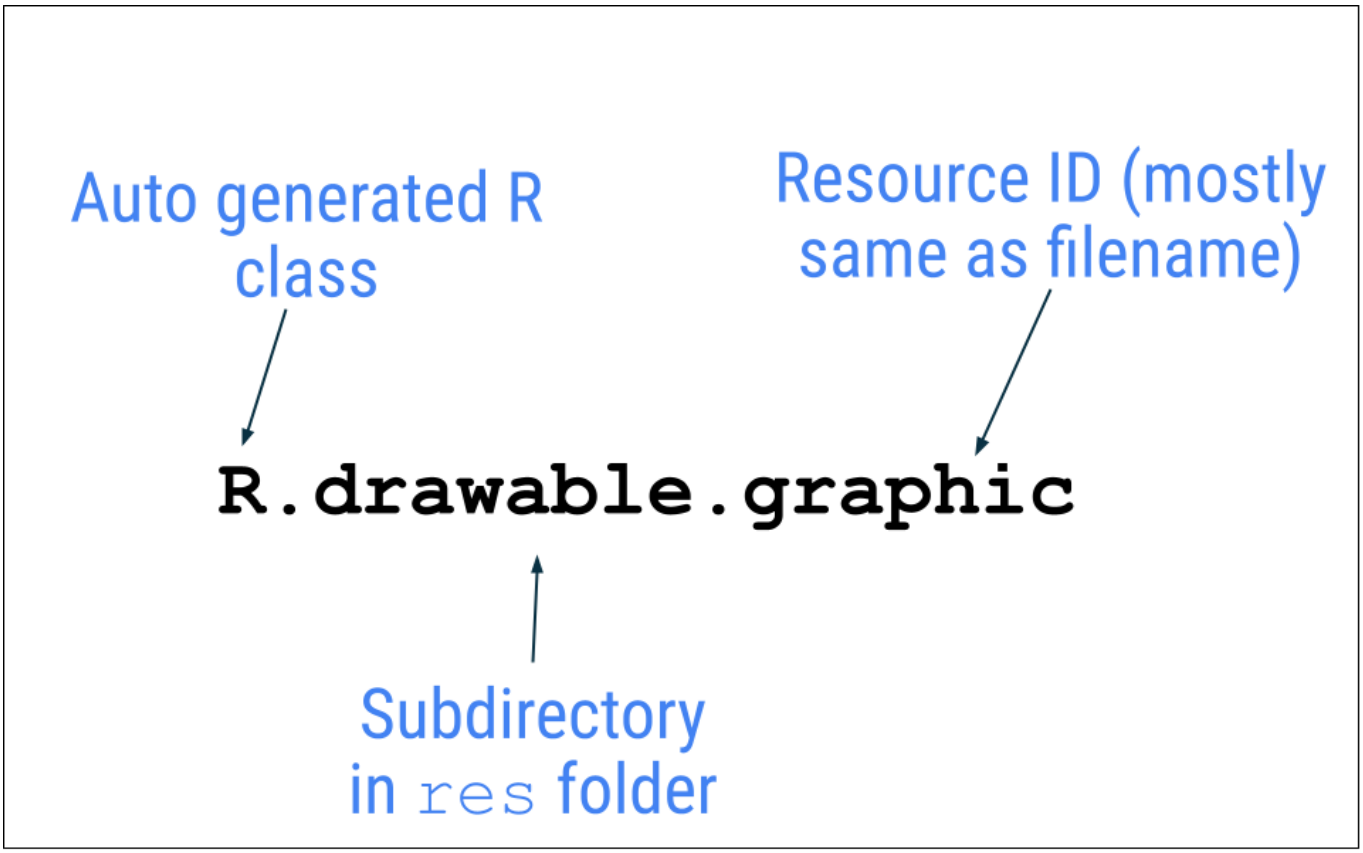
Группировка ресурсов Каждый тип ресурсов всегда следует помещать в определенный подкаталог каталога res/ вашего проекта. Например, вот иерархия файлов для простого проекта:

```
MyProject/
    src/
        MyActivity.kt
    res/
        drawable/
            graphic.png
        mipmap/
            icon.png
        values/
            strings.xml
```

Как видно из примера, каталог res/ содержит все ресурсы в подкаталогах, в том числе каталог drawable/ для ресурсов изображений, каталог mipmap/ для иконок пусковой установки и каталог values/ для строковых ресурсов. Чтобы узнать больше об использовании, формате и синтаксисе ресурсов приложений, смотрите раздел Обзор типов ресурсов.

Доступ к ресурсам Jetpack Compose может получить доступ к ресурсам, определенным в вашем проекте Android. Доступ к ресурсам осуществляется с помощью идентификаторов ресурсов, которые генерируются в R-классе вашего проекта.

R-класс - это автоматически генерируемый Android класс, который содержит идентификаторы всех ресурсов в проекте. В большинстве случаев идентификатор ресурса совпадает с именем файла. Например, к изображению в предыдущей иерархии файлов можно получить доступ с помощью этого кода:

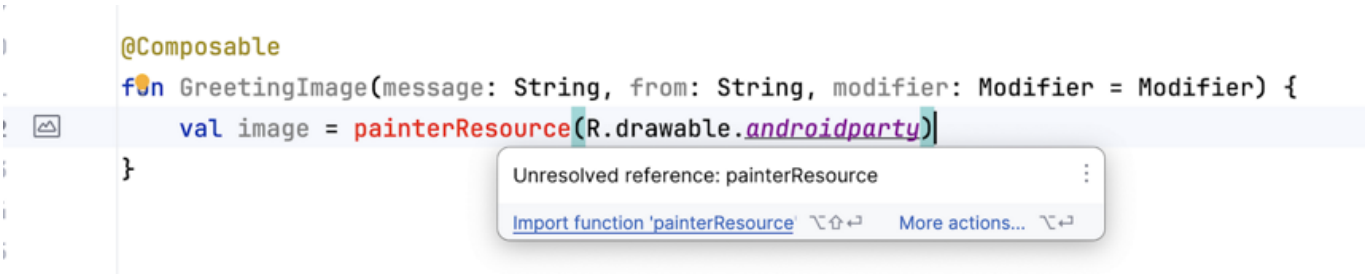


В следующем задании вы используете изображение, файл androidparty.png, который вы добавили в предыдущем задании.

В функции GreetingImage() объявите свойство val и назовите его image. Выполните вызов функции painterResource(), передав в нее ресурс androidparty. Присвойте возвращаемое значение переменной image.

```
val image = painterResource(R.drawable.androidparty)
```

Android Studio выделяет код .painterResource, поскольку для компиляции приложения необходимо импортировать эту функцию.



Щелкните .painterResource, выделенный Android Studio. Нажмите Import во всплывающем окне, чтобы добавить импорт для androidx.compose.ui.res.painterResource. Функция painterResource() загружает ресурс рисуемого изображения и принимает в качестве аргумента идентификатор ресурса (в данном случае R.drawable.androidparty).

После вызова функции painterResource() добавьте композит Image, а затем передайте изображение в качестве именованного аргумента для painter.

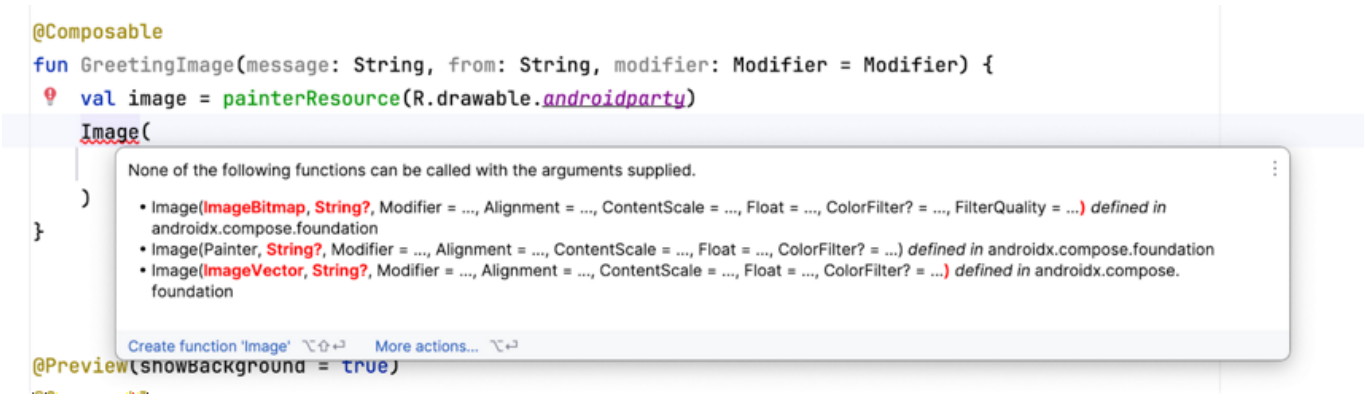
```
Image(
    painter = image
)
```

Android Studio выделяет код Image, потому что для компиляции приложения необходимо импортировать функцию.

Чтобы исправить это предупреждение, добавьте следующий импорт в верхнюю часть файла MainActivity.kt:

```
import androidx.compose.foundation.Image
```

Первоначальное предупреждение теперь устранено, но если навести курсор на слово Image, Android Studio выводит новое предупреждение, которое гласит: «Ни одна из следующих функций не может быть вызвана с указанными аргументами». Это происходит потому, что предоставленный аргумент не соответствует ни одной из сигнатур функций Image.



Это предупреждение будет исправлено в следующем разделе.

Проверьте свое приложение на доступность Соблюдая правила кодирования для обеспечения доступности, вы позволяете всем вашим пользователям, включая людей с ограниченными возможностями, легче ориентироваться и взаимодействовать с вашим приложением.

> Примечание: Android предоставляет множество инструментов для пользователей. Например, TalkBack - это программа для чтения с экрана Google, включенная в устройства Android. TalkBack предоставляет пользователям голосовую обратную связь, чтобы они могли пользоваться

> устройством, не глядя на экран. Чтобы узнать больше о доступности, смотрите раздел Создание доступных приложений.

Android Studio предоставляет подсказки и предупреждения, которые помогут вам сделать ваше приложение более доступным. Описание содержимого определяет назначение элемента пользовательского интерфейса, что делает ваше приложение более удобным для использования с TalkBack.

Однако изображение в этом приложении включено только в декоративных целях. Добавление описания содержимого для изображения сделает его более удобным для использования с TalkBack в данном конкретном случае. Вместо того чтобы задавать описание содержимого, которое сообщается пользователю, вы можете установить аргумент contentDescription изображения в null, чтобы TalkBack пропустил композит Image.

В составной части Image добавьте еще один именованный аргумент contentDescription и установите его значение в null.

```
Image(
    painter = image,
    contentDescription = null
)
```

Предварительный просмотр композитного изображения В этом задании вы предварительно просмотрите изображение в композитном виде и запустите приложение на эмуляторе или устройстве.

В функции BirthdayCardPreview() замените вызов функции GreetingText() на вызов функции GreetingImage(). Ваша функция должна выглядеть так, как показано в этом фрагменте кода:

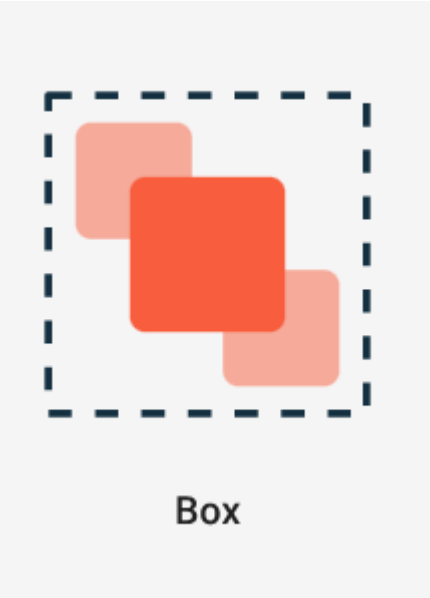
```
@Preview(showBackground = true)
@Composable
fun BirthdayCardPreview() {
    HappyBirthdayTheme {
        GreetingImage(
            message = "Happy Birthday Sam!",
            from = "From Emma"
        )
    }
}
```

Панель Design должна автоматически обновиться, если этого не произошло, нажмите 609ccb451d05cf6b.png для создания. Обратите внимание, что текст больше не виден, потому что в новой функции есть только Image composable, но нет Text composable.

4. Добавление макета Box

Три основных стандартных элемента макета в Compose - это композиты Column, Row и Box. О композитах Column и Row вы узнали в предыдущих коделабах, а теперь вы узнаете больше о композите Box.

Вставка - это один из стандартных элементов компоновки в Compose. С помощью компоновки Box можно располагать элементы друг над другом. Макет Box также позволяет настроить специфическое выравнивание элементов, которые он содержит.



В функции GreetingImage() добавьте компонент Box вокруг компонента Image, как показано на рисунке:

```
@Composable
fun GreetingImage(message: String, from: String, modifier: Modifier = Modifier) {
    val image = painterResource(R.drawable.androidparty)
    Box {
        Image(
            painter = image,
            contentDescription = null
        )
    }
}
```

Импортируйте функцию androidx.compose.foundation.layout.Box при появлении запроса в Android Studio. Добавьте код для передачи параметра модификатора в компонуемый Box.

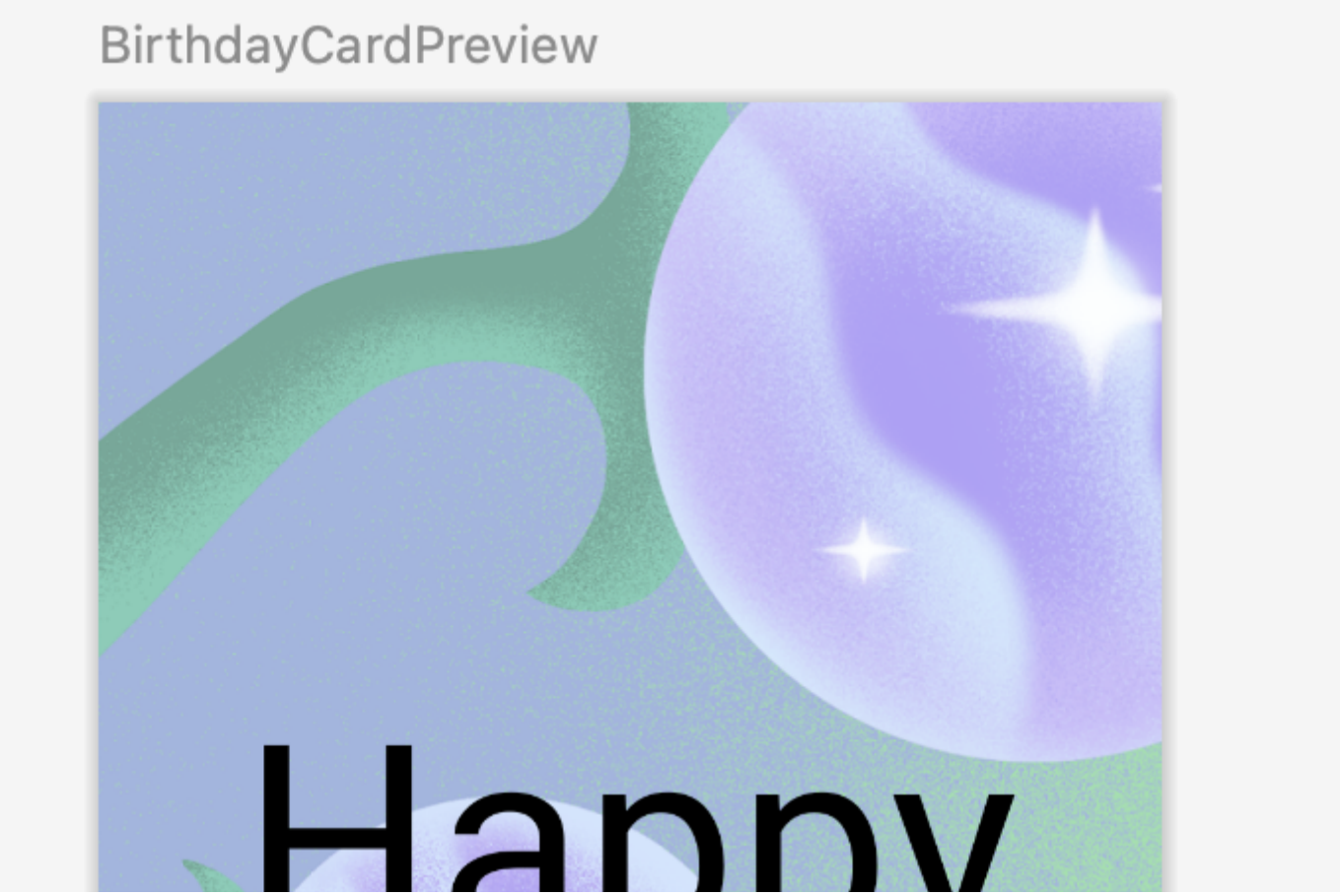
```
@Composable
fun GreetingImage(message: String, from: String, modifier: Modifier = Modifier) {
    val image = painterResource(R.drawable.androidparty)
    Box(modifier) {
        Image(
            painter = image,
            contentDescription = null
        )
```

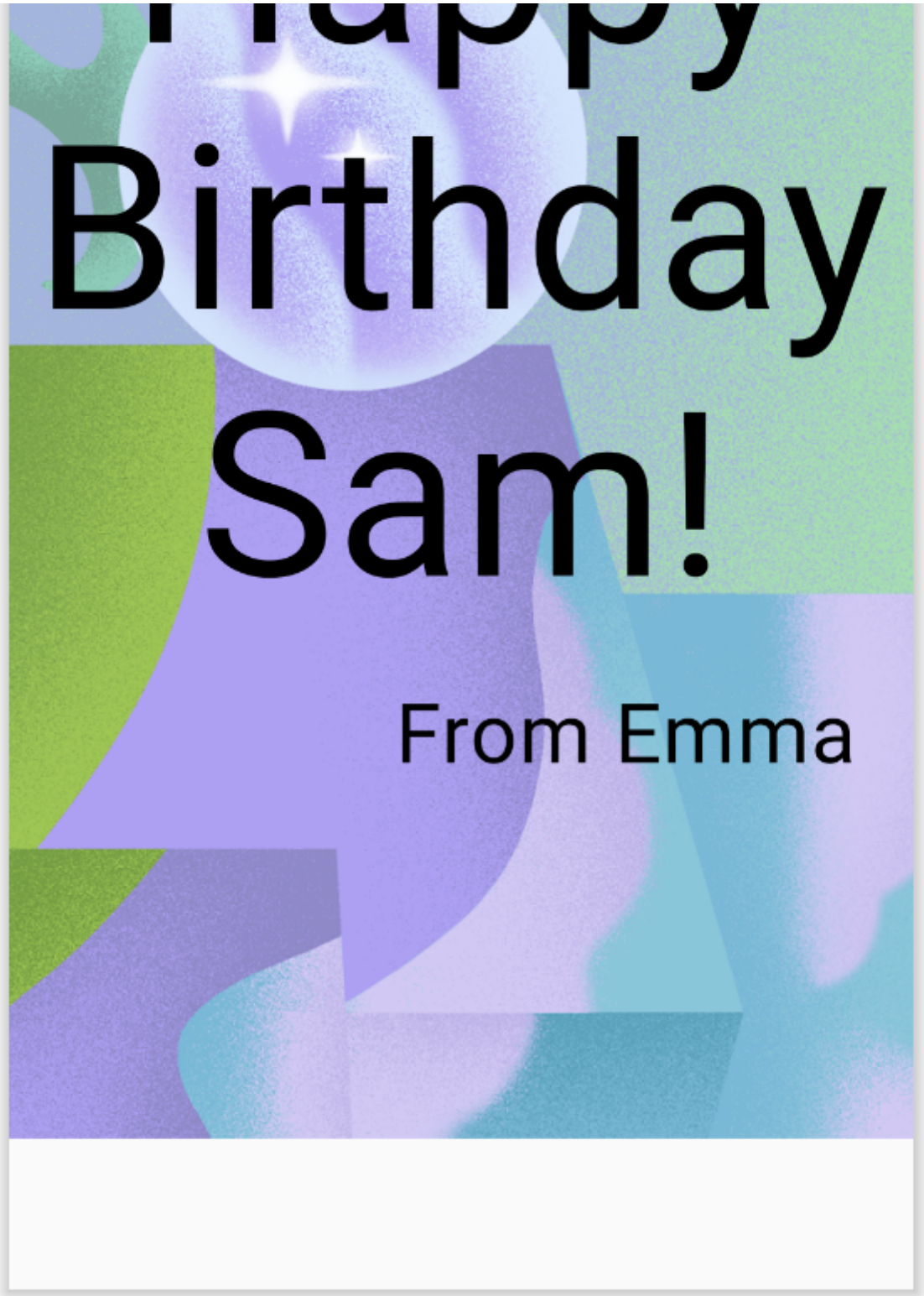
```
    }
}
```

В конце композитного блока Box вызовите функцию GreetingText() и передайте ей сообщение о дне рождения, подпись и модификатор, как показано на рисунке:

```
@Composable
fun GreetingImage(message: String, from: String, modifier: Modifier = Modifier) {
    val image = painterResource(R.drawable.androidparty)
    Box(modifier) {
        Image(
            painter = image,
            contentDescription = null
        )
        GreetingText(
            message = message,
            from = from,
            modifier = Modifier
                .fillMaxSize()
                .padding(8.dp)
        )
    }
}
```

Обратите внимание на обновленный предварительный просмотр в панели Design. Вы должны увидеть текст и изображение.





Чтобы описанные выше изменения отразились в эмуляторе или на устройстве, в функции onCreate() замените вызов функции GreetingText() на вызов функции GreetingImage(). Ваш блок setContent должен выглядеть так, как показано в этом фрагменте кода:

```
setContent {
    HappyBirthdayTheme {
        // A surface container using the 'background' color from the theme
        Surface(
```

```
            modifier = Modifier.fillMaxSize(),
            color = MaterialTheme.colorScheme.background
        ) {
            GreetingImage(
                message = "Happy Birthday Sam!",
                from = "From Emma"
            )
        }
    }
}
```

Обратите внимание, что изображение имеет такую же ширину, как и экран, но оно привязано к верхней части экрана. В нижней части экрана остается белое пространство, которое выглядит не очень привлекательно. В следующем задании вы заполните ширину и высоту экрана, а также масштабируете изображение, чтобы оно заполнило весь экран.

5. Изменение непрозрачности и масштабирование изображения

В этом задании вы сделаете изображение полноэкранным, чтобы украсить ваше приложение. Для этого вы используете параметры ContentScale.

Масштабировать содержимое Вы добавили изображение в приложение и разместили его. Теперь вам нужно настроить тип масштаба изображения, который указывает, как изменить размер изображения, чтобы сделать его полноэкранным.

Существует довольно много типов ContentScale. Вы используете параметр ContentScale.Crop, который равномерно масштабирует изображение для сохранения соотношения сторон, чтобы ширина и высота изображения были равны или больше соответствующих размеров экрана.

Добавьте к изображению именованный аргумент ContentScale.

```
Image(
    painter = image,
    contentDescription = null,
    contentScale = ContentScale.Crop
)
```

Импортируйте интерфейс androidx.compose.ui.layout.ContentScale при появлении запроса в Android Studio. Посмотрите на панель Design. Теперь изображение должно заполнить весь экран предварительного просмотра, как показано на этом скриншоте:

Изменить непрозрачность Чтобы улучшить контрастность приложения, измените непрозрачность фонового изображения.

Добавьте параметр alpha в составное изображение Image и установите его на 0,5F.

```
Image(
    painter = image,
    contentDescription = null,
    contentScale = ContentScale.Crop,
    alpha = 0.5F
)
```

Обратите внимание на изменение изображения opacity.

https://developer.android.com/static/codelabs/basic-android-kotlin-compose-add-images/img/951cdc313bfd120d_856.png

Как много кода! Пришло время просмотреть всю вашу тяжелую работу.

Запустите приложение Запустите приложение на устройстве или эмуляторе.

9d1416521733e8c.png

Хорошая работа с полноэкранным изображением и текстовым сообщением. Вы также изменили непрозрачность изображения.

Модификаторы макета Модификаторы используются для украшения или добавления поведения к элементам пользовательского интерфейса Jetpack Compose. Например, вы можете добавить фон, подложку или поведение для строк, текста или кнопок. Чтобы задать их, компонуемый элемент или макет должен принять модификатор в качестве параметра.

В предыдущем кодельном уроке вы узнали о модификаторах и использовали модификатор padding (Modifier.padding) для добавления пространства вокруг Text composable. Модификаторы могут многое, и вы увидите это в этом и последующих уроках.

Например, у этого Text composable есть аргумент Modifier, который меняет цвет фона на зеленый.

```
// Example
Text(
    text = "Hello, World!",
    // Solid element background color
    modifier = Modifier.background(color = Color.Green)
)
```

Аналогично приведенному выше примеру, вы можете добавлять модификаторы к макетам, чтобы позиционировать дочерние элементы с помощью свойств расположения и выравнивания.

Чтобы задать положение дочерних элементов в строке, задайте аргументы horizontalArrangement и verticalAlignment. Для столбца задайте аргументы verticalArrangement и horizontalAlignment.

Свойство arrangement используется для упорядочивания дочерних элементов, когда размер макета больше, чем сумма его дочерних элементов.

Например, когда размер колонки больше суммы размеров ее дочерних элементов, можно задать вертикальное выравнивание (VerticalArrangement), чтобы определить расположение дочерних элементов внутри колонки. Ниже показаны различные варианты вертикального расположения:

одинаковая высота, пространство между, пространство вокруг, пространство равномерно, сверху, по центру и снизу

Аналогично, когда размер Строки больше суммы размеров ее дочерних элементов, можно указать горизонтальное расположение (horizontalArrangement), чтобы определить расположение дочерних элементов внутри Строки. Ниже показаны различные варианты горизонтального расположения:

равный вес, пространство между, пространство вокруг, пространство равномерно, конец, центр и начало

Свойство alignment используется для выравнивания дочерних элементов по началу, центру или концу макета.

6. Выравнивание и расположение текста

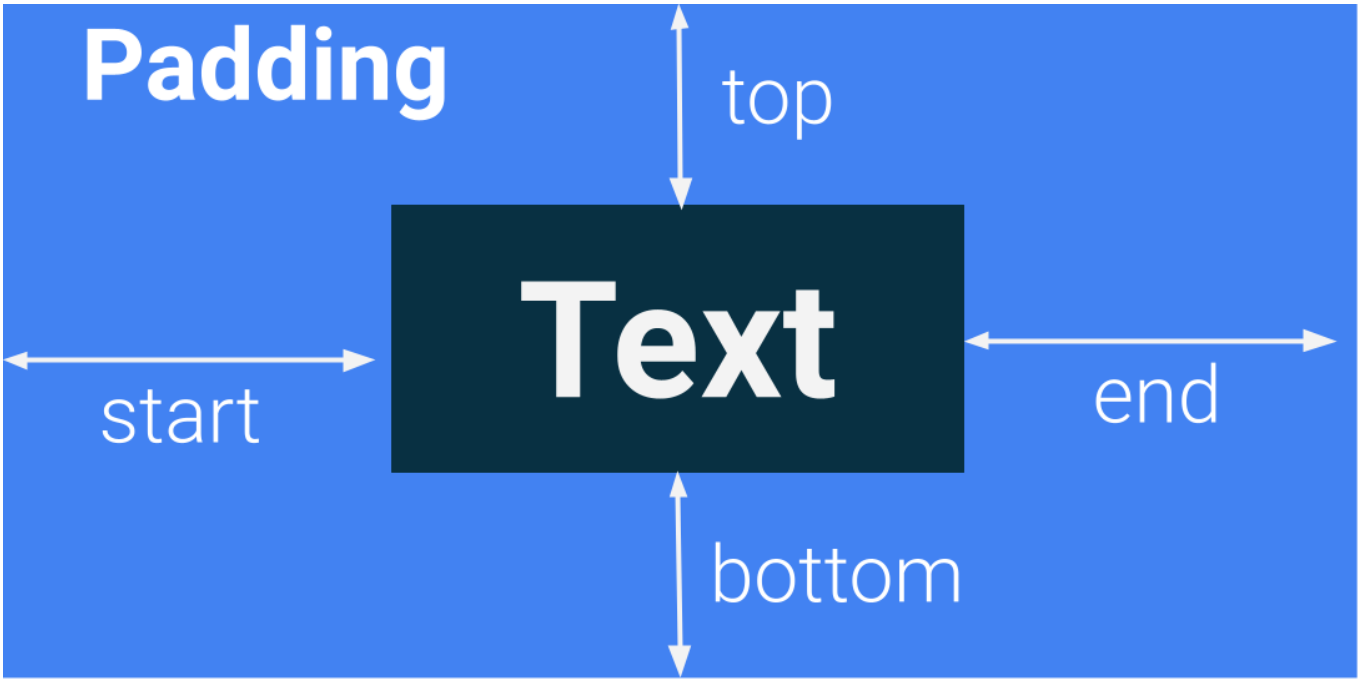
В этом задании вы будете наблюдать за кодом, который вы добавили в предыдущем codelab, чтобы расположить текст в приложении.

В файле MainActivity.kt прокрутите страницу до функции GreetingText(). Свойство verticalArrangement в колонке имеет значение Arrangement.Center. Таким образом, содержимое текста будет отцентрировано на экране.

```
@Composable
fun GreetingText(message: String, from: String, modifier: Modifier = Modifier) {
    Column(
        verticalArrangement = Arrangement.Center,
        modifier = modifier
    ) {
        Text(
            text = message,
            fontSize = 100.sp,
            lineHeight = 116.sp,
            textAlign = TextAlign.Center
        )
        Text(
            text = from,
            fontSize = 36.sp,
            modifier = Modifier
                .padding(16.dp)
                .align(alignment = Alignment.End)
        )
    }
}
```

Padding Элемент пользовательского интерфейса оборачивается вокруг своего содержимого. Чтобы не допустить слишком плотного обхвата, можно задать величину подкладки с каждой стороны.

Padding используется как модификатор, что означает, что вы можете применить его к любому композиту. Для каждой стороны композита модификатор padding принимает необязательный аргумент, определяющий количество набивки.




```
// This is an example.
Modifier.padding(
    start = 16.dp,
    top = 16.dp,
    end = 16.dp,
    bottom = 16.dp
)
```

Ваша очередь! В файле MainActivity.kt прокрутите страницу до того места, где вызывается функция GreetingText(), и обратите внимание на атрибут padding.

```
modifier = Modifier
    .fillMaxSize()
    .padding(8.dp)
```

Аналогично обратите внимание на то, что внутри функции GreetingText() подложка для подписи Text является составной.

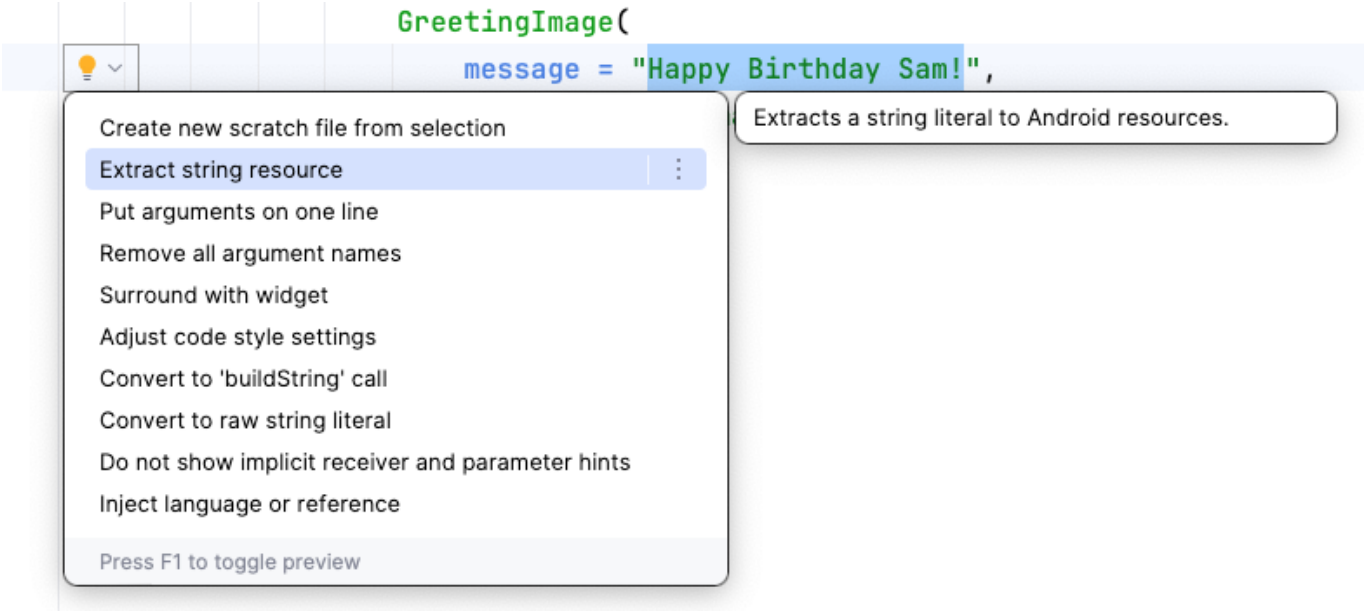
```
modifier = Modifier
    .padding(16.dp)
    .align(alignment = Alignment.End)
```

7. Применяйте передовые методы работы с кодом

Перевод Когда вы пишете приложения, важно помнить, что в какой-то момент они могут быть переведены на другой язык. Как вы узнали из предыдущего коделаба, тип данных String - это последовательность символов, например «Happy Birthday Sam!».

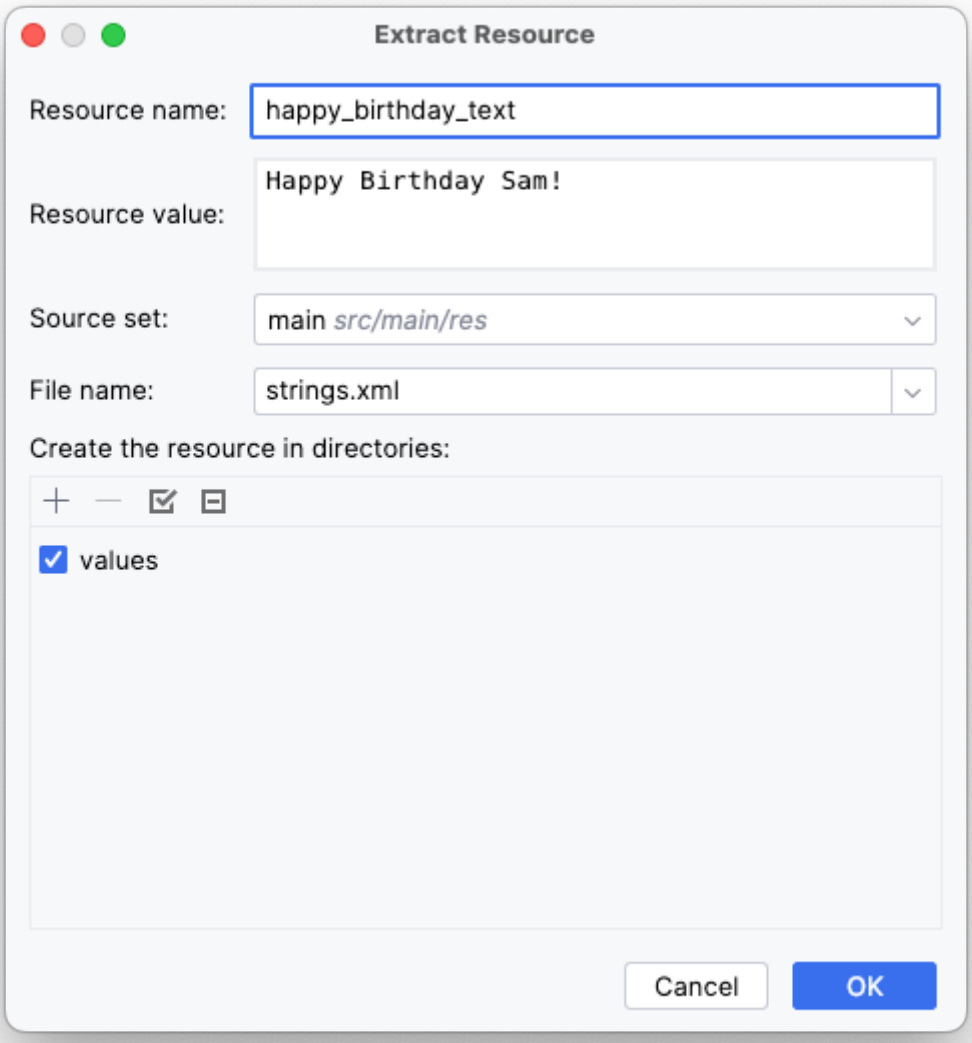
Жестко закодированная строка - это строка, которая записана непосредственно в коде вашего приложения. Жестко закодированные строки усложняют перевод вашего приложения на другие языки и затрудняют повторное использование строк в разных местах вашего приложения. Чтобы решить эти проблемы, можно извлекать строки в файл ресурсов. Вместо того чтобы жестко кодировать строки в коде, вы помещаете их в файл, называете строковые ресурсы и используете эти имена везде, где хотите использовать строки. Имя остается неизменным, даже если вы измените строку или переведете ее на другой язык.

В файле MainActivity.kt прокрутите страницу до функции onCreate(). Выберите поздравление с днем рождения, строку Happy Birthday Sam! без кавычек. Щелкните по лампочке в левой части экрана. Выберите Извлечь строковый ресурс.



Android Studio открывает диалоговое окно Extract Resource. В этом диалоге вы можете настроить, как будет называться ваш строковый ресурс, и некоторые детали его хранения. В поле Resource name вы вводите название строки. В поле «Значение ресурса» вводится сама строка.

В диалоговом окне Извлечение ресурса измените имя ресурса на happy_birthday_text. Строковые ресурсы должны иметь строчные имена, а несколько слов должны быть разделены знаком подчеркивания. Остальные настройки оставьте по умолчанию.



Нажмите кнопку OK. Обратите внимание на изменения в коде. Теперь жестко закодированная строка заменена вызовом функции getString().

```
GreetingImage(
    message = getString(R.string.happy_birthday_text),
    from = "From Emma",
    modifier = Modifier.padding(8.dp)
)
```

> Примечание: Некоторые версии Android Studio заменяют жестко закодированную строку функцией getString(). В таких случаях необходимо вручную изменить функцию на stringResource().

При необходимости добавьте import androidx.compose.ui.res.stringResource в раздел импорта.

В панели Project откройте файл strings.xml по пути app > res > values > strings.xml и заметите, что Android Studio создала строковый ресурс happy_birthday_text.

```
<resources>
    <string name="app_name">Happy Birthday</string>
    <string name="happy_birthday_text">Happy Birthday Sam!</string>
</resources>
```

Файл strings.xml содержит список строк, которые пользователь увидит в вашем приложении. Обратите внимание, что название вашего приложения также является строковым ресурсом. Поместив все строки в одно место, вы сможете легче перевести весь текст в вашем приложении и легче повторно использовать строки в разных частях вашего приложения.

Выполните те же шаги, чтобы извлечь текст для композита «Текст подписи», но на этот раз введите signature_text в поле Resource name. Ваш готовый файл должен выглядеть так, как показано в этом фрагменте кода:

```
<resources>
    <string name="app_name">Happy Birthday</string>
    <string name="happy_birthday_text">Happy Birthday Sam!</string>
    <string name="signature_text">From Emma</string>
</resources>
```

Обновите функцию BirthdayCardPreview(), чтобы использовать stringResource() и извлеченные строки.

```
@Preview(showBackground = true)
@Composable
fun BirthdayCardPreview() {
    HappyBirthdayTheme {
        GreetingImage(
            message = stringResource(R.string.happy_birthday_text),
            from = stringResource(R.string.signature_text)
        )
    }
}
```

> Примечание: Если Android Studio выдает предупреждение «Unresolved reference: stringResource» при наведении курсора на stringResource, вам нужно добавить оператор импорта для androidx.compose.ui.res.stringResource, чтобы использовать функцию stringResource().

Запустите приложение еще раз, чтобы убедиться, что оно по-прежнему работает.

8. Попробуйте решить эту задачу

Вы отлично справились с добавлением изображения в ваше приложение. Вот вам задание:

Расположите или выровняйте текст подписи Compose так, чтобы он был выровнен по центру экрана.

> Подсказка: Compose предлагает модификатор align для индивидуального расположения дочерних компонент, не подчиняясь правилам выравнивания, навязанным родительским

> макетом. Соедините аргумент .align(alignment = Alignment.CenterHorizontally) с модификатором text composable.

Ваше приложение должно выглядеть примерно так:

Вот код решения для функции GreetingText() для справки:

```
@Composable
fun GreetingText(message: String, from: String, modifier: Modifier = Modifier) {
    Column(
        verticalArrangement = Arrangement.Center,
        modifier = modifier
    ) {
        Text(
            text = message,
            fontSize = 100.sp,
            lineHeight = 116.sp,
            textAlign = TextAlign.Center
        )
        Text(
            text = from,
            fontSize = 36.sp,
            modifier = Modifier
                .padding(16.dp)
                .align(alignment = Alignment.CenterHorizontally)
        )
    }
}
```

9. Получите код решения

Код решения для приложения Happy Birthday находится на GitHub.

GitHub - это сервис, позволяющий разработчикам управлять кодом для своих программных проектов. Он использует Git, систему контроля версий, которая отслеживает изменения, сделанные для каждой версии кода. Если вы когда-нибудь видели историю версий документа в Google Docs, вы можете увидеть, какие правки и когда были сделаны. Точно так же вы можете отслеживать историю версий кода в проекте. Это полезно, когда вы работаете над проектом в одиночку или в команде.

У GitHub также есть веб-сайт, позволяющий просматривать и управлять проектом. По этой ссылке GitHub вы можете просмотреть файлы проекта Happy Birthday онлайн или загрузить их на свой компьютер.

10. Заключение

Вы добавили изображение в приложение Happy Birthday, выровняли текст с помощью модификаторов, соблюли рекомендации по доступности и облегчили перевод на другие языки! Что еще более важно, вы завершили создание своего собственного приложения Happy Birthday! Поделитесь своей работой в социальных сетях и используйте хэштег #AndroidBasics, чтобы мы могли ее увидеть!

Резюме

Вкладка Resource Manager в Android Studio помогает добавлять и организовывать изображения и другие ресурсы. Композитное изображение - это элемент пользовательского интерфейса, который отображает изображения в вашем приложении. Изображение должно иметь описание содержимого, чтобы сделать ваше приложение более доступным. Текст, который отображается пользователю, например поздравление с днем рождения, должен быть извлечен в строковый ресурс, чтобы было проще перевести приложение на другие языки.