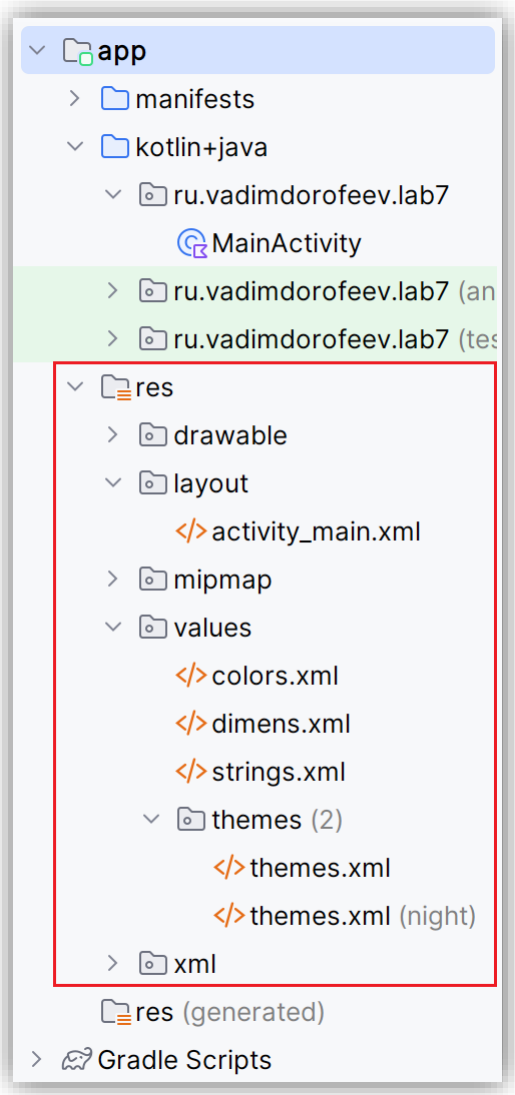
Ресурсы

Ресурсы – это контент и файлы, которые используются в программе, но хранятся отдельно от кода. Это могут быть изображения (растровые или векторные), текстовые строки, файлы разметки и т. д.

Использование ресурсов вместо жёсткого кодирования соответствующей информации в коде позволяет работать с ними независимо друг от друга. Например, файл с текстовыми строками можно отдать переводчику или корректору, а когда он будет готов – заменить исходный файл на новый. Это не потребует внесения каких-либо правок в код.

Другое преимущество использования ресурсов – альтернативные конфигурации. Если жёстко закодировать текстовые строки в коде, то локализация программы на другие языки может оказаться весьма трудоёмкой. В случае же использования ресурсов система автоматически подключит нужный языковой файл в зависимости от конфигурации устройства пользователя.

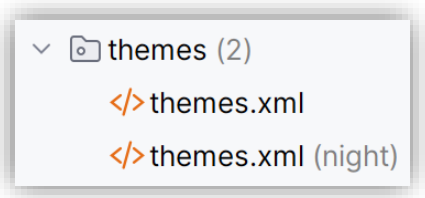
В Android Studio ресурсы размещаются в папке res в дереве проекта:



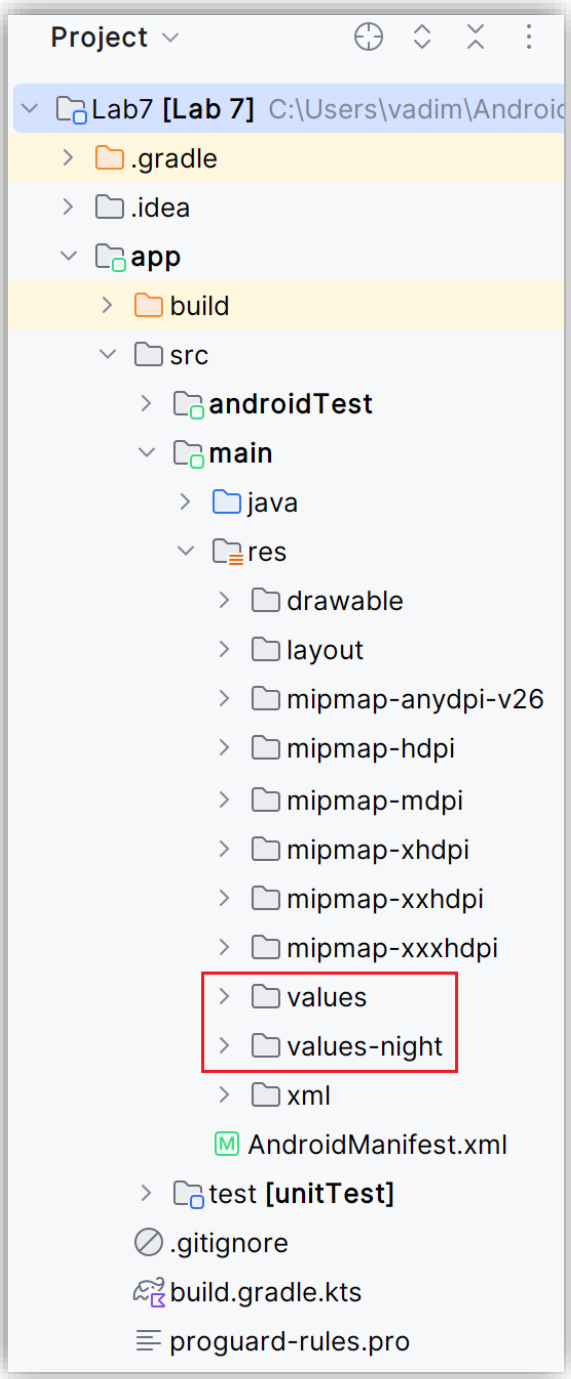
Все ресурсы сгруппированы в виртуальных папках согласно их типу, наиболее часто используются следующие папки:

- `drawable` – графические растровые или векторные изображения
- `mipmap` – значок программы в разных разрешениях
- `layout` – разметка пользовательского интерфейса в формате XML
- `raw` – файлы в произвольном формате, программа должна понимать их формат при чтении
- `values` – файлы с простыми значениями в формате XML, это могут быть строки, числа и цвета. Данная папка может содержать несколько файлов с разными типами ресурсов, например:
    - `arrays.xml` для массивов
    - `colors.xml` для цветов
    - `dimens.xml` для размеров и отступов
    - `strings.xml` для строковых значений

Справа от имени папки может добавляться название альтернативной конфигурации, например, первый файл `themes.xml` будет использоваться если в системе установлена светлая тема, а второй, у которого присутствует спецификатор `(night)` – если используется тёмная тема:



На самом деле, на диске это два отдельных файла, которые хранятся в разных папках. Можно переключить дерево проекта из режима Android в режим Project, тогда ветки дерева будут соответствовать физическим файлам и папкам:



К базовому имени папки добавляется спецификатор, например, язык (`-ru` или `-de`) или, как в примере выше, значения для ночного режима (`-night`).

Доступ к ресурсам

Ко всем ресурсам, добавленным в программу, можно получить доступ в коде с помощью особого класса `R`. Это автоматически генерируемый класс, он заново формируется каждый раз при сборке программы. В этот файл заносятся не сами ресурсы, а только ссылки на них, точнее, числовые поля, по которым система сможет определить и загрузить нужный ресурс.

Ссылки на ресурсы в классе `R` группируются согласно типу ресурсов, например, строки представляются идентификаторами `R.string.XXX`, а изображения – `R.drawable.XXX`., где `XXX` – идентификатор конкретного ресурса. Например, если в строковых ресурсах есть строка с идентификатором `hello`, то обратиться к ней из кода можно следующим образом:

```
val s = resources.getString(R.string.hello)
```

После выполнения этой строки в переменной `s` окажется строка из ресурсов, причем, сразу на нужном языке (конечно, если файл со строковыми ресурсами на этом языке присутствует).

В примере выше используется специальная системная переменная `resources`, она содержит методы для получения доступа к ресурсам. Однако, некоторые подобные методы есть и в классах, реализующих *контекст*, например, в активностях или сервисах, поэтому в таких классах можно обратиться к ресурсам и без использования переменной `resources`:

```
val s = getString(R.string.hello)
```

Для ссылки на ресурс в разметке интерфейса используется следующий синтаксис: `@string/XXX` или `@drawable/XXX`. Например, у кнопки можно задать текст и цвет фона:

```
<Button
    ...
    android:backgroundTint="@color/button_color"
    android:text="@string/button_title"/>
```

Строковые ресурсы

Строковые ресурсы хранятся в файле `strings.xml` в следующем формате:

```
<resources>
    <string name="app_name">Lab 7</string>
    <string name="choose_fruit">Выберите фрукт:</string>
    <string name="order_total">Выбрано фруктов: %d, на сумму: %d</string>
    <string name="confirm">Подтвердить заказ!</string>
</resources>
```

По умолчанию в строковых ресурсах присутствует строка с идентификатором `app_name`, она используется в качестве названия программы в манифесте и ряде других мест. Не нужно её удалять, это вызовет ошибку при сборке программы.

В файле разметки можно использовать строку следующим образом:

```
<TextView
    ...
    android:text="@string/choose_fruit"/>
```

А в коде получить строку так:

```
val s = getString(R.string.order_total)
```

Подстановки

В ресурсных строках, как и в обычных строках, допускается использовать специальные символы для подстановки: `%d` для целых чисел, `%f` для чисел с плавающей точкой, и `%s` для строковых значений. В такую строку можно подставить значения с помощью метода `format` в классе `String`. Например, в примере выше строка, полученная из ресурса с идентификатором `order_total`, содержит два символа подстановки `%d`, и можно подставить в эти позиции числа:

```
val s1 = String.format(s, 3, 15)
```

Следует, однако, помнить, что в разных языках такие параметры могут оказываться на разных местах: например, в одном языке быть принято сначала ставить название товара, а затем цену, а в других наоборот. Поэтому лучше вместе с символами подстановки указывать и порядковый номер аргумента:

```
<string name="item">Товар %2$s стоит %1$f</string>
```

В таком формате после знака `%` ставится номер аргумента и знак `$`, а затем идёт тип аргумента: `%2$s` означает что в данном месте будет использоваться второй аргумент (`2$`) строкового типа (`s`):

```
val s1 = String.format(s, 19.99, "Хлеб")
```

Хотя в функции `format` указывается сначала цена, а затем название, в результате получится строка «Товар Хлеб стоит 19.99», потому что при подстановке явно задано какие по индексу аргументы использовать в каждой подстановке.

Строковые массивы

Иногда бывает нужно задать не одну строку, а сразу много – например, названия дней недели:

```
<string-array name="day_names">
    <item>Понедельник</item>
    <item>Вторник</item>
    <item>Среда</item>
    <item>Четверг</item>
    <item>Пятница</item>
    <item>Суббота</item>
    <item>Воскресенье</item>
</string-array>
```

Получить такой массив в коде можно следующим образом:

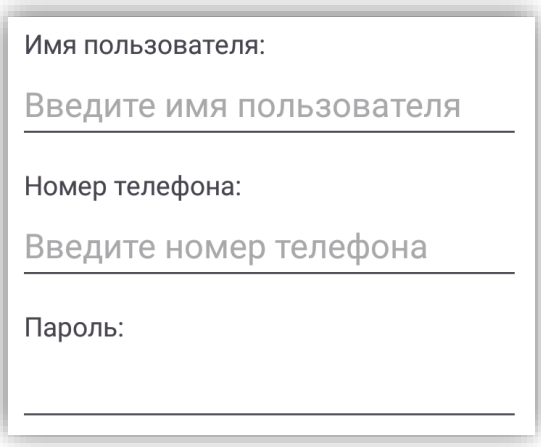
```
val days = resources.getStringArray(R.array.day_names)
```

Два важных момента: во-первых, метод `getStringArray` есть только в системной переменной `resources`, в данном случае её название нельзя опустить, а во-вторых, хоть ресурс и строковый, но в классе `R` он находится не в разделе `R.string`, а в разделе `R.array`.

Поместить такой ресурс можно как в файл `strings.xml`, так и в файл `arrays.xml`: компилятору разницы нет, разделение на файлы осуществляется для большей понятности и удобства.

Размеры и отступы

В интерфейсе нередко приходится указывать отступы между элементами (margin), отступы содержимого внутри элементов от их границ (padding), размеры шрифтов, а иногда и самих элементов. Такие значения часто повторяются: например, если на экране выведено несколько однотипных элементов, то почти наверняка их размеры и отступы будут одинаковыми:



В этом случае можно вынести повторяющиеся значения в файл `dimens.xml`:

```
<resources>
    <dimen name="label_size">16sp</dimen>
    <dimen name="left_margin">6dp</dimen>
    <dimen name="textview_top_margin">10dp</dimen>
</resources>
```

А в разметке использовать ссылки на них:

```
<TextView
    ...
    android:textSize="@dimen/label_size"
    android:layout_marginTop="@dimen/textview_top_margin"
    android:layout_marginStart="@dimen/left_margin"/>

<EditText
    ...
    android:layout_marginStart="@dimen/left_margin"/>
```

Преимуществом такого подхода является лёгкость модификации: для увеличения отступа перед названием поля достаточно изменить значение в одном файле `dimens.xml`, а не править его по всем файлам разметки.

Цвета

Работа с цветовыми ресурсами в целом аналогична работе с другими ресурсами, они задаются в файле `colors.xml` в форматах `#RRGGBB` или `#AARRGGBB`, где в шестнадцатеричном формате заданы `AA` – значение альфа-канала, `RR` – значение красной компоненты, `GG` – значение зелёной компоненты, и `BB` – значение синей компоненты. Значение альфа-канала в файле `colors.xml` можно опускать, в этом случае предполагается максимальное значение `FF`:

```
<resources>
    <color name="eatable">#40C040</color>
    <color name="non_eatable">#C04040</color>
</resources>
```

В разметке на цвета можно ссылаться обычным для ресурсов образом:

```
<Button
    ...
    android:backgroundTint="@color/eatable"/>
```

Цвет фона обычно задаётся с помощью атрибута `backgroundTint`, а цвет текста – с помощью `textColor`.

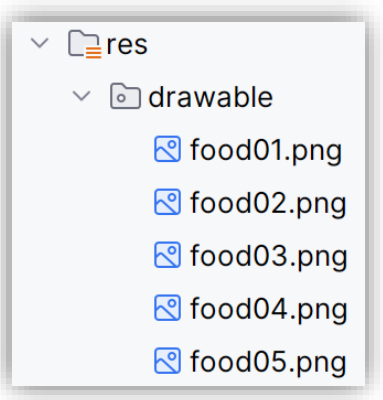
Программно можно получить цвет следующим образом:

```
val c = getColor(R.color.eatable)
tvMyTextView.setTextColor(c)
```

Графические ресурсы

Графические ресурсы размещаются в папке `drawable`. Имена файлов должны отвечать правилам именования переменных в Kotlin: в них не должно быть пробелов, они должны начинаться с буквы и т.д.

Растровые ресурсы могут быть в форматах PNG, JPG, WEBP или GIF. Графические файлы можно просто перетащить из Проводника в папку `drawable` в дереве проекта:



Хотя названия файлов в дереве отображаются с расширением (в данном случае .png), в класс `R` они помещаются без расширения, и при ссылке на них из разметки расширение также не должно указываться.

Чтобы использовать графический ресурс в разметке, на него можно сослаться с помощью атрибута `src`:

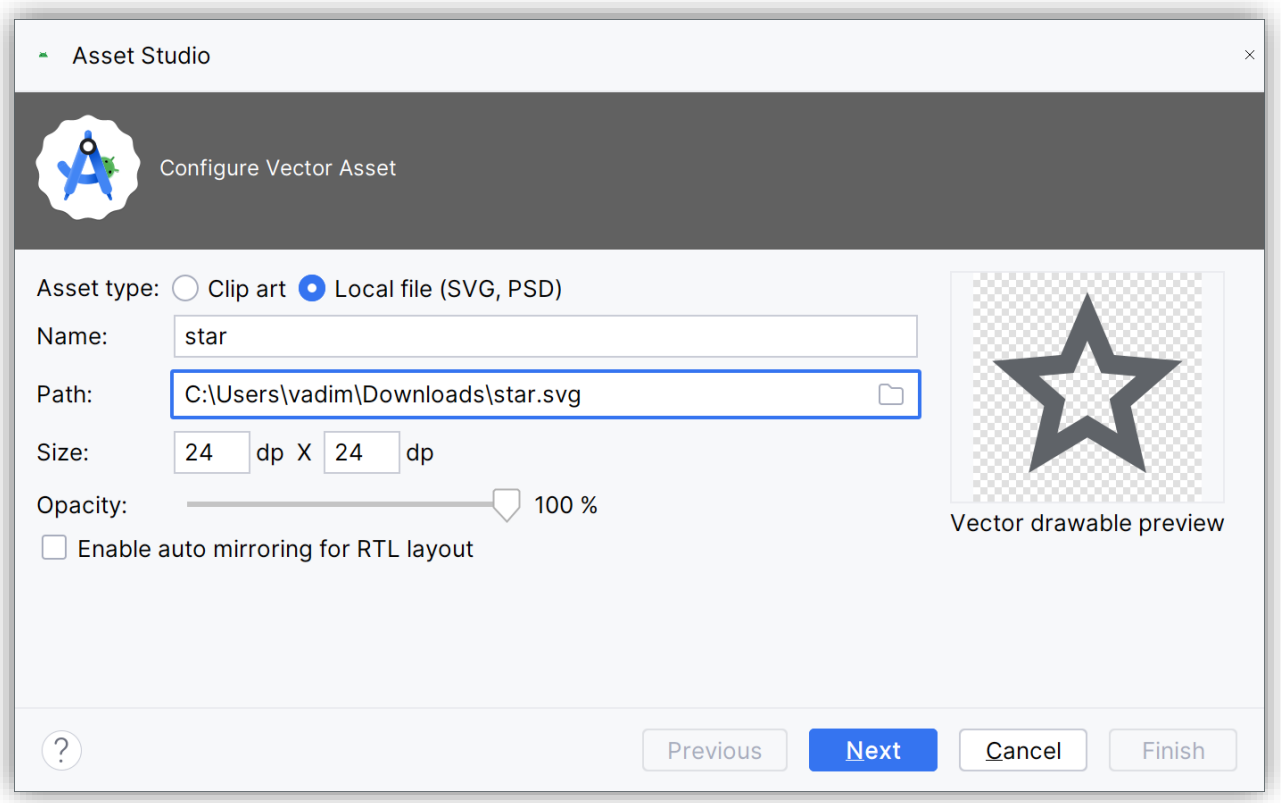
```
<ImageView
    ...
    android:src="@drawable/food01"/>
```

В коде программы изображение можно получить так:

```
val d = ContextCompat.getDrawable(this, R.drawable.food01)
```

Векторные ресурсы должны быть в формате SVG. Их нельзя просто перетащить в папку `drawable` как растровые изображения, каждый файл должен быть импортирован отдельно. Для этого нужно щёлкнуть правой кнопкой по папке res и в меню выбрать New → Vector

Asset, появится окно Asset Studio. В нём в переключателе Asset type нужно выбрать значение Local file, а затем в поле Path указать путь к импортируемому файлу:

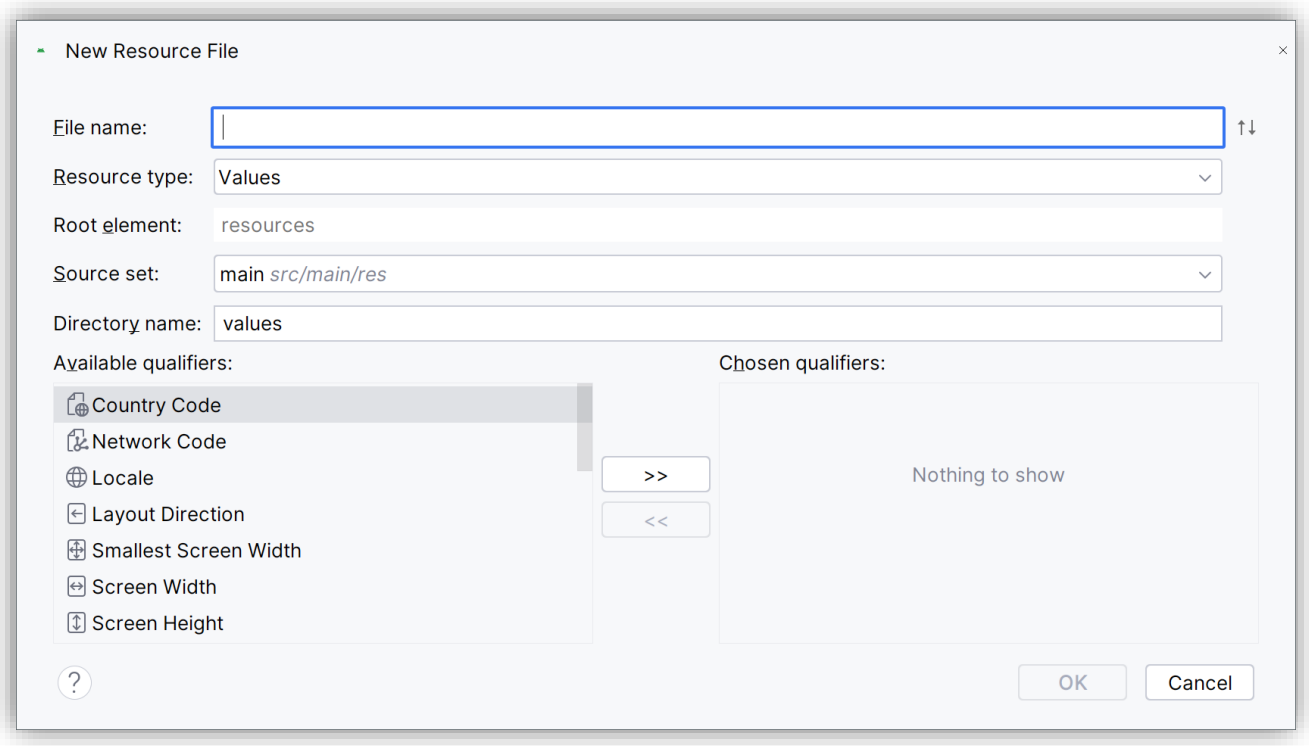


Android Studio преобразует SVG-файл во внутренний формат XML, и после этого с векторным изображением можно будет работать так же как и с растровым.

Альтернативные конфигурации

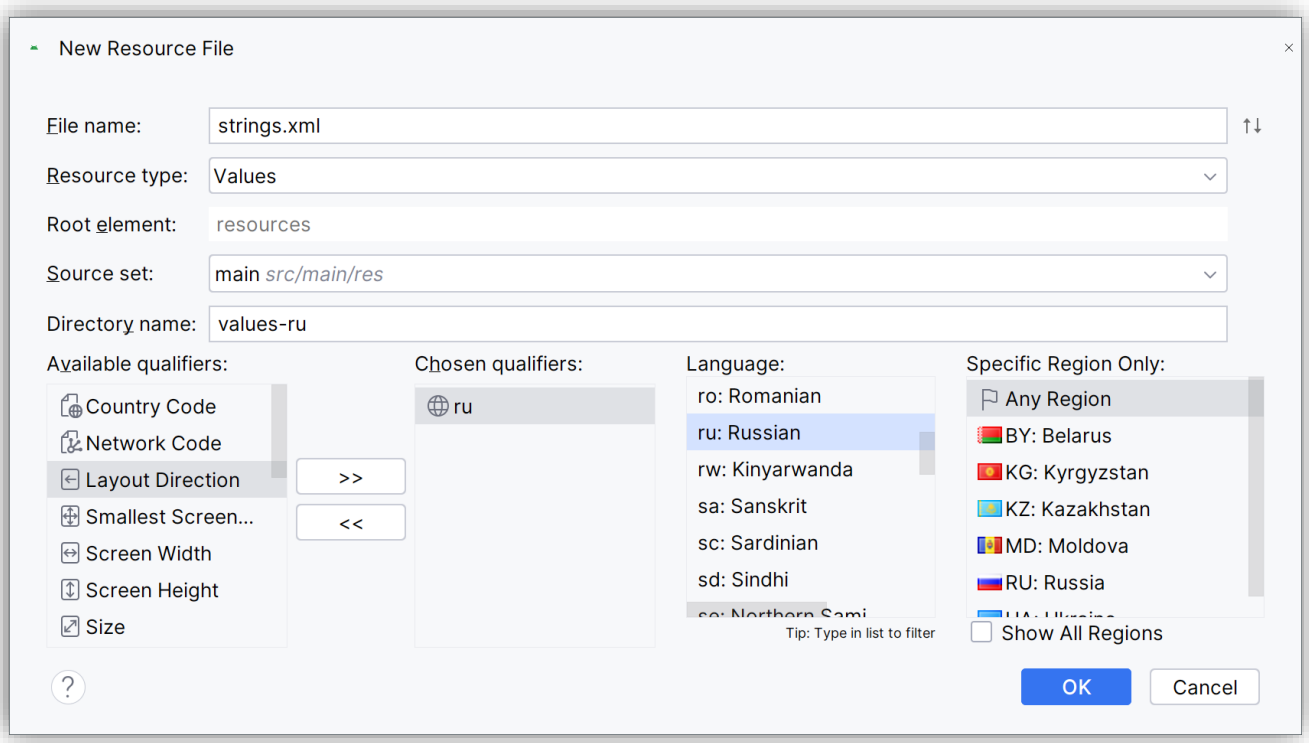
Если разработчик включил несколько вариантов ресурса, система автоматически постарается подобрать наиболее подходящий.

Для добавления альтернативной конфигурации ресурса нужно щёлкнуть правой кнопкой мышки на папке res и в меню выбрать New → Android Resource File, появится окно New Resource File:

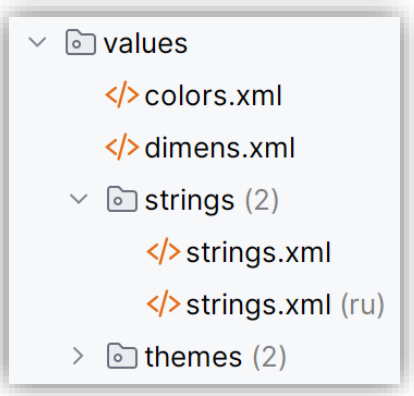


В поле File name нужно ввести имя файла, оно должно быть в точности таким же как оригинальное, для которого создаётся альтернатива. Например, если есть файл со строковыми ресурсами `strings.xml`, и нужно добавить такой файл для другого языка, то для него файл также должен называться `strings.xml`.

Далее следует проверить тип ресурса (строки должны иметь тип Values, файлы разметки Layout и т.д.), а затем указать спецификатор конфигурации (*англ.* qualifier) – именно он определяет какой файл ресурсов должен быть загружен. Например, чтобы создать строковый ресурсный файл для русского языка, нужно заполнить окно следующим образом (выбран спецификатор Locale):



После этого в дереве проекта появится новый файл, у которого справа от имени будет указано при какой альтернативной конфигурации система его задействует:



При запуске программы система определит какой язык выбран у пользователя основным. Для русского языка будет загружен файл со спецификатором `(ru)`, а для всех остальных – основной файл без спецификатора.

Важно помнить, что для всех альтернативных языков требуется одинаковый набор строк: не получится в альтернативном файле пропустить некоторые строки, Android Studio будет считать это за ошибку.

Ещё один пример использования альтернативных конфигураций – разная разметка для разных ориентаций устройства. За это отвечает спецификатор Orientation: в вертикальном (портретном) режиме он имеет значение Portrait, а в горизонтальном (альбомном) – Landscape. Имя файла разметки, как указывалось ранее, тоже должно быть одинаковым для всех альтернатив, например, `activity_main.xml`.

Задание

Разработайте игру «Съедобно – несъедобно». Приложение должно показывать картинку, на которой может быть съедобный или несъедобный предмет. Игрок нажимает одну из двух кнопок: «Съедобно» или «Несъедобно». Если игрок прав, программа увеличивает количество правильных ответов, если ошибся – увеличивает количество неправильных. В любом случае осуществляется переход к следующему случайно выбранному изображению.

Главная активность приложения может выглядеть примерно следующим образом:

В качестве примера изображений для игры можно взять бесплатные изображения из файла 08_goods_icons.zip, который приложен к странице с заданием.

Начиная с этой лабораторной работы все строки, цвета, размеры элементов, их отступы и т. д. должны браться из ресурсов приложения. Использование в разметке или в коде программы жёстко закодированных значений будет приравниваться к ошибке.