Лабораторная работа № 2 по теме:
«Работа с ресурсами и активами»
Теоретическая часть

Ресурсы (англ. resource) и активы (англ. assets) — неотъемлемая часть Android-приложений. Это внешние элементы, которые включаются в приложение: изображения, аудио, видео, строки, компоновки, темы и т. д. Ресурсы хранятся в отдельных файлах во вложенных папках в каталоге res. Активы хранятся в папке assets. Ресурсы легко поддерживать, обновлять и редактировать. Ресурсы используются чаще, чем активы. Различие между ними заключается в следующем:

- Доступ к ресурсам осуществляется через класс R, который автоматически создается инструментом aapt. Ресурсы легко доступны для использования в коде программы. Во время компиляции приложения инструмент aapt создает класс R, в котором находятся идентификаторы для всех ресурсов в каталоге res/. Для каждого типа ресурсов предусмотрен подкласс R (например, класс R.drawable для изображений), а для каждого ресурса указанного типа создаётся статическая целочисленная переменная. Эта переменная как раз и служит идентификатором ресурса, который можно использовать для его получения.
- Чтение файлов активов осуществляется с помощью AssetManager. Этот класс представляет низкоуровневый API, который позволяет открывать и читать необработанные файлы, связанные с приложением, в виде простого потока байтов.

В таблице 1 представлены типы ресурсов, которые могут использоваться в Android-приложениях. Каталог res/ содержит все ресурсы в подкаталогах, названия которых приведены в левом столбце.

Таблица 1. Типы ресурсов, использующиеся в Android-приложениях.

Папка	**Описание ресурса**
animator	Файлы XML, которые определяют анимации свойств, например, анимацию изменения цвета.
anim	Файлы XML, которые определяют анимации преобразований (<alpha>, <scale>, <translate>, <rotate>). (Анимации свойств также можно сохранять в этом каталоге, но для них предпочтительнее использовать каталог animator/, чтобы различать эти два типа).
color	Файлы XML, которые определяют список состояний цветов.
drawable	Файлы растровых изображений (.png, .9.png, .jpg, .gif) или файлы XML, например, Shape Drawable.
mipmap	Графические файлы для иконок приложения для экранов с различной плотностью.
layout	Файлы XML, которые определяют макет пользовательского интерфейса (компоновки).
menu	Файлы XML, которые определяют меню приложения, такие как меню параметров, контекстные меню или вложенные меню.
raw	Произвольные файлы для сохранения в исходной форме. Открыть эти ресурсы можно с помощью InputStream, вызвав Resources.openRawResource() и передав в данный метод идентификатор ресурса (R.raw.filename). В том случае, если требуется получить доступ к исходным именам файлов и иерархии файлов, можно сохранять ресурсы в каталоге assets/ (вместо каталога res/raw/). Файлы в каталоге assets/ не получают идентификатора ресурса.
values	Файлы XML, которые содержат простые значения, такие как строки, целые числа и цвета. Тогда как XML-файлы ресурсов в других подкаталогах каталога res/ определяют отдельные ресурсы на базе имени файла XML, файлы в каталоге values/ описывают несколько ресурсов. Для файла в этом каталоге каждый дочерний элемент элемента <resources> определяет один ресурс. Например, элемент <string> создает ресурс R.string, а элемент <color> создает ресурс R.color. Так как каждый ресурс определяется с помощью своего собственного элемента XML, можно назначать имя файла по своему усмотрению и помещать ресурсы разных типов в один файл. Тем не менее, существуют соглашения для имен файлов ресурсов: • arrays.xml для ресурсов-массивов (массивы с указанием типа) • colors.xml для значений цветов • dimens.xml для значений единиц измерений • strings.xml для строковых значений • styles.xml для стилей.

xml	Произвольные XML-файлы, которые можно читать в режиме выполнения вызовом метода Resources.getXML().

Доступ к ресурсу можно получить из java-кода и из xml. Из кода доступ получается с помощью статической целочисленной переменной из подкласса вашего класса R, например:

R.string.name

string — это тип ресурса, а name — это имя ресурса.

Чтобы использовать ресурс в коде, можно передать идентификатор ресурса в виде параметра метода. Например, с помощью метода setImageResource() можно указать использование виджетом ImageView ресурса res/drawable/image.png:

Листинг 1. Задание источника изображения в ImageView в коде

```
ImageView myImage = (ImageView) findViewById(R.id.imageview);
myImage.setImageResource(R.drawable.image);
```

Из XML доступ получается с помощью особого синтаксиса, который также соответствует идентификатору ресурса, заданному в классе R, например:

@string/name

string — это тип ресурса, а name — это его имя. Этот синтаксис можно использовать в любом фрагменте ресурса XML, где ожидается значение, указанное вами в ресурсе.

Для задания значений для некоторых атрибутов и элементов XML можно использовать ссылку на существующий ресурс. Это зачастую требуется при создании компоновки при указании строк и изображений для виджетов. Например, при добавлении в компоновку элемента ImageView можно указать ссылку на источник изображения в атрибуте android:src (листинг 2).

Листинг 2. Задание источника изображения в ImageView в xml

```
<ImageView
    android:id="@+id/imageview"
    android:src="@drawable/image"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Альтернативные ресурсы

Для ресурсов любого типа можно указать ресурс *по умолчанию* и несколько *альтернативных* ресурсов для приложения:

- Ресурсы по умолчанию должны использоваться независимо от конфигурации устройства или в том случае, когда отсутствуют альтернативные ресурсы, соответствующие текущей конфигурации.
- Альтернативные ресурсы предназначены для работы с определенными конфигурациями. Чтобы указать, что группа ресурсов предназначена для определенной конфигурации, необходимо добавить соответствующий квалификатор к имени каталога.

Например, в качестве квалификатора может быть указана **плотность экрана устройства** (ldpi, mdpi, hdpi, xhdpi и т.д.) Такие квалификаторы часто указываются для папок drawable и mipmap. Так в папки drawable-mdpi, drawable-hdpi, drawable-xhdpi размещают одинаковые изображения с одинаковыми именами, но с разным разрешением для качественного отображения на устройствах с разной плотностью экрана. В зависимости от плотности экрана реального устройства изображения автоматически подгружаются из соответствующей папки.

Используются следующие квалификаторы, указывающие для экранов с какой плотностью предназначены ресурсы:

- ldpi: экраны низкой плотности; приблизительно 120 dpi.
- mdpi: экраны средней плотности; приблизительно 160 dpi.
- hdpi: экраны высокой плотности; приблизительно 240 dpi.
- xhdpi: экраны очень высокой плотности; приблизительно 320 dpi. Добавлено в API уровня 8.

- xxhdpi: экраны сверхвысокой плотности; приблизительно 480 dpi. Добавлено в API уровня 16.
- xxxhdpi: использование исключительно высокой плотности (только иконка); приблизительно 640 dpi. Добавлено в API уровня 18.
- nodpi: этот режим можно использовать для растровых графических ресурсов, которые не требуется масштабировать в соответствии с плотностью устройства.
- tvdpi: экраны промежуточной плотности между mdpi и hdpi; приблизительно 213 dpi. Этот режим не считается «основной» группой плотности. Он главным образом предназначен для телевизоров, и большинство приложений в нем не нуждается.

Шесть основных уровней плотности (ldpi:mdpi:hdpi:xhdpi:xxhdpi:xxxhdpi) соотносятся как 3:4:6:8:12:16. Так, растровое изображение 9x9 пикселей в ldpi представляется как 12x12 пикселей в mdpi, 18x18 в hdpi, 24x24 в xhdpi и т.д.

Другим примером квалификаторов могут служить квалификаторы, указывающие **язык и регион**, для которых предназначены данные ресурсы. С помощью этих квалификаторов можно легко создавать мультиязычные приложения.

Язык задается двухбуквенным кодом языка ISO 639-1 (например, ru, en, fr), к которому можно добавить двухбуквенный код региона ISO 3166-1-alpha-2 с предшествующей строчной буквой "r" (например, en-rUS). Коды не зависят от регистра, нельзя указывать только код региона.

В случае использования ресурсов с указанными квалификаторами языка и региона, эти ресурсу будут подгружаться в соответствии с тем, какой язык и регион задан пользователем в системных настройках устройства.

Также в качестве квалификатора может быть указана **ориентация экрана устройства**: port (устройство в портретной (вертикальной)

ориентации) и land (устройство в книжной (горизонтальной) ориентации). Ориентация может измениться за время работы приложения, если пользователь поворачивает экран. Таким образом, можно создавать компоновки с различным расположением элементов пользовательского интерфейса в зависимости от ориентации экрана.

Например, несмотря на то, что макет пользовательского интерфейса по умолчанию сохранен в каталоге res/layout/, можно указать другой макет для использования на экране с альбомной ориентацией, сохранив его в каталоге res/layout-land/. Android автоматически применяет соответствующие ресурсы, сопоставляя текущую конфигурацию устройства с именами каталогов ресурсов.

Также в качестве квалификаторов можно задавать:

- Код страны для мобильной связи (MCC), за которым может следовать код сети мобильной связи (MNC) из SIM-карты устройства.
- Направление макета для приложения (ldrtl для направления макета справа налево и ldltr для направления слева направо, которое используется по умолчанию).
- Основной размер экрана, указывающий минимальный размер доступной области экрана (минимальная ширина устройства).
- Минимальная доступная ширина экрана в единицах dp, для которой должен использоваться ресурс. Это значение конфигурации будет изменяться в соответствии с текущей фактической шириной при изменении альбомной/книжной ориентации.
- Минимальная доступная высота экрана в пикселах, для которой должен использоваться ресурс. Это значение конфигурации будет изменяться в соответствии с текущей фактической высотой при изменении альбомной/книжной ориентации.
- Размер экрана.
- Формат экрана (long для длинных экранов и notlong для недлинных).

- Режим пользовательского интерфейса (car — устройство подсоединено к автомобильной док-станции; desk — устройство подсоединено к настольной док-станции; television — устройство подсоединено к телевизору, обеспечивая взаимодействие с расстояния «три метра», когда пользовательский интерфейс находится на большом экране, находящемся вдалеке от пользователя; appliance — устройство служит в качестве прибора без дисплея; watch — устройство с дисплеем для ношения на запястье).
- Ночной режим (night для ночного времени и notnight для дневного времени).
- Тип сенсорного экрана (notouch — устройство не оснащено сенсорным экраном, finger — устройство оснащено сенсорным экраном, предназначенным для ввода с помощью пальцев пользователя).
- Доступность клавиатуры (аппаратной или экранной).
- Основной способ ввода текста (nokeys, qwerty, 12key).
- Доступность клавиш перемещения.
- Основной несенсорный способ перемещения курсора.
- Версия платформы (уровень API) (например, v3, v4, v7 и т.д.)

Если не предусмотрены альтернативные ресурсы с квалификаторами, лучше подходящими к текущей конфигурации устройства, система может использовать любые наиболее подходящие ресурсы.

Ресурсы Drawable

Как было сказано ранее, ресурсы Drawable — это файлы растровых изображений (.png, .9.png, .jpg, .gif) или файлы XML, задающие изображение.

Из перечисленных растровых форматов формат PNG — рекомендуемый, JPG — приемлемый и GIF — нежелательный.

Формат NinePatch (.9.png) — это специальная разновидность PNG-файлов, которые содержат рамку, описывающую область, которая может растягиваться при изменении размеров изображения. Изменения размеров изображения будут пропорционально влиять на относительные размеры участков изображения, помеченных как растягиваемые. Название NinePatch происходит от сетки 3x3, которая условно разбивает изображение на 9 частей (рисунок 1).

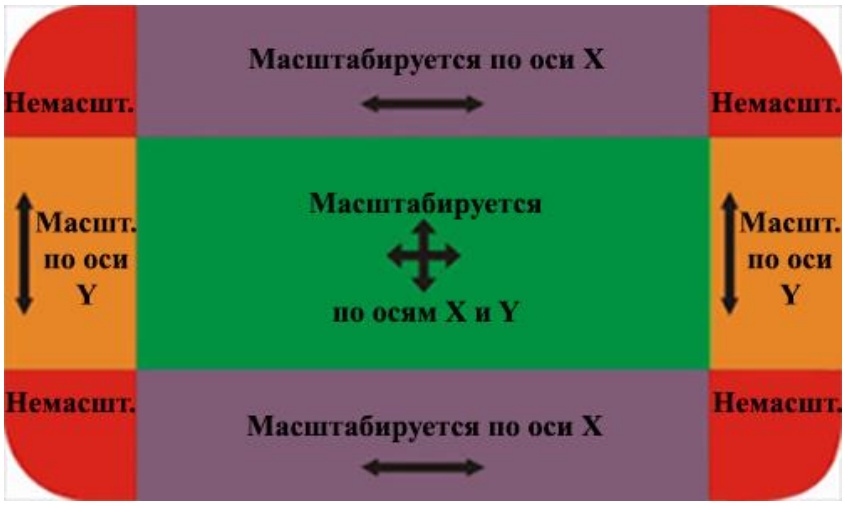

Рисунок 1. Формат NinePatch.

Файлы растровых изображений могут быть автоматически оптимизированы с использованием сжатия без потерь с помощью инструмента aapt в процессе сборки проекта. Например, PNG, для которого не требуется более 256 цветов, может быть преобразован в 8-битный PNG с цветовой палитрой. В результате будет получено изображение равного качества, но для которого потребуется меньше памяти. Эту особенность ресурсов Drawable надо учитывать, если вы планируете читать изображение как битовый поток. В таком случае, следует помещать изображения в папку res/raw/, где они не будут оптимизированы.

Файлы XML также могут использоваться в качестве графических ресурсов. Используются следующие подтипы графических ресурсов из XML-файлов:

- State List — XML-файл, который ссылается на разные растровые изображения для разных состояний (например, для использования другого изображения в качестве фона кнопки при её нажатии).
- Level List — XML-файл, определяющий ресурс Drawable, который управляет несколькими альтернативными объектами Drawable, каждому из которых соответствует числовое значение (например, для использования нескольких изображений соответствующих разным уровням сигнала или заряда батареи устройства).
- Transition Drawable — XML-файл, определяющий ресурс Drawable, в котором может осуществляться плавный переход (англ. cross-fade) между двумя объектами Drawable.
- Inset Drawable — XML-файл, определяющий ресурс Drawable, который включает в себя другой объект Drawable с отступам. Это полезно, когда для объекта View требуется рисовать фон, размер которого меньше фактических границ View.
- Clip Drawable — XML-файл, определяющий объект Drawable, который обрезает другой объект Drawable по горизонтали и (или) вертикали на основе текущего значения атрибута level — уровня. Чаще всего используется для реализации индикаторов выполнения задачи.
- Scale Drawable — XML-файл, определяющий объект Drawable, который изменяет размер (масштабирует) другого объекта Drawable в зависимости от текущего значения уровня. Чаще всего используется для реализации индикаторов выполнения задачи.
- Shape Drawable — XML-файл, определяющий геометрическую форму, включая цвета и градиенты.
- Vector Drawable — векторная графика, определенная в файле XML как набор точек, линий и кривых вместе с соответствующей информацией о цвете. Основным преимуществом использования Vector Drawable является масштабируемость изображения. Его можно масштабировать

без потери качества отображения, что означает, что размер одного и того же изображения изменяется для разной плотности экрана без потери качества изображения. Это приводит к уменьшению размера APK-файлов и упрощает разработку приложения. Поддержка Vector Drawable добавлена Android 5.0 (API уровень 21).

Стили и темы

Стили и темы — это такие же ресурсы, как и строки, изображения и т.д. Android обеспечивает некоторые заданные по умолчанию стили и темы, которые вы можете использовать в приложениях. При необходимости вы можете определить свой собственный стиль и тему для создаваемого приложения.

Стиль — это один или несколько сгруппированных атрибутов форматирования, которые вы можете применить как модуль к отдельным элементам пользовательского интерфейса в XML-схеме компоновки для Activity. Например, вы можете определить стиль, который устанавливает определенный размер текста и цвет фона, а затем применить его к выбранным компонентам пользовательского интерфейса.

Листинг 3. Задание стиля

```
<resources>
<style name="SpecialText" parent="@style/Text">
    <item name="android:textSize">28sp</item>
    <item name="android:textColor">#548f00</item>
    <item name="android:background">#b7e4fb</item>
</style>
</resources>
```

Вы можете использовать элементы <item>, чтобы установить определенные значения форматирования для стиля. В атрибуте name указывается название параметра, значение которого задаётся внутри данного элемента item.

Обратите внимание на атрибут parent в элементе <style>. Этот атрибут позволяет определять ресурс, от которого текущий стиль наследует значения свойств. Стиль может наследоваться от любого типа ресурса,

который содержит требуемый стиль. Ваши собственные стили должны всегда наследоваться, прямо или косвенно, от стандартных стилей Android. В этом случае необходимо только определить значения, которые требуется изменить в наследуемом стиле.

Для ссылки на стиль используется атрибут style. Например, для элемента TextView в компоновке можно указать атрибут style="@style/SpecialText".

Тема — это группа из одного или нескольких атрибутов форматирования, которые вы можете применить как модуль к одному или ко всем активити в приложении. Например, можно определить тему, которая устанавливает цвет для фона, размеры текста и цвета для меню.

Листинг 4. Задание темы

```
<resources>
<style name="MyTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="android:windowBackground">@drawable/background</item>
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
</resources>
```

Темы задаются так же, как и стили. Темы объявляются в XML-файле элементами <style> и ссылаются на них тем же самым способом. Различие состоит в том, что тема добавляется ко всему приложению или к отдельному Activity через элементы <application> и <activity> в файле манифеста приложения AndroidManifest.xml.

AndroidManifest.xml — это XML-файл, который содержит важную информацию о приложении, требующуюся системе Android, и задает конфигурацию приложения: объявляет компоненты приложения, перечисляет любые библиотеки, связанные с приложением (помимо библиотек Android, связанных по умолчанию), объявляет разрешения, которые требуются для работы приложения (например, доступ в сеть, разрешение на отправку SMS) и т.д.

Чтобы установить тему для всех активити приложения, в файле AndroidManifest.xml в элемент <application> необходимо добавить атрибут android:theme с названием темы:

android:theme="@style/MyTheme"

Если вы хотите определить тему, которая будет загружаться только для одного активити, добавьте атрибут android:theme с темой в элемент <activity> данного активити.

Наряду с другими встроенными ресурсами, в Android есть несколько стандартных тем, которые можно загрузить в приложение.

Меню

Меню — это стандартный элемент пользовательского интерфейса, который используется в различных приложениях. На платформе Android для реализации меню используется класс Menu. Существуют различные типы меню: меню параметров, контекстное меню и всплывающее меню.

Для описания пунктов меню рекомендуется использовать xml-ресурс, размещаемый в папке res/menu. Использовать ресурсы меню удобнее, чем объявлять пункты меню в программном коде, потому что это позволяет проще визуализировать структуру меню; отделять контент для меню от java-кода; создавать альтернативные ресурсы меню.

В XML-файле меню могут располагаться три типа элементов:

- <menu> — корневой элемент меню;
- <group> — контейнерный элемент, определяющий группу меню; он является необязательным и невидимым. Он позволяет разделять пункты меню на категории и назначать им одинаковые свойства, такие как активное состояние и видимость. Элементы <item> и <group> могут быть дочерними элементами <group>.
- <item> — элемент, определяющий пункт меню. Создает класс MenuItem. Этот элемент может содержать вложенный элемент <menu> для создания вложенных меню.

Корневой узел любого файла меню должен быть элементом <menu>.

У элемента <item> есть несколько атрибутов, с помощью которых можно определить внешний вид и поведение пункта меню:

- android:id — уникальный идентификатор ресурса.
- android:icon — ссылка на графический ресурс, который будет использоваться в качестве значка пункта меню.
- android:title — ссылка на строку, которая будет использоваться в качестве названия пункта меню.
- android:showAsAction — указывает, когда и как этот пункт должен отображаться в строке действий (англ. app bar или action bar).

Для использования меню в активити необходимо загрузить ресурс меню, то есть преобразовать xml-ресурс в объект с помощью метода MenuInflater.inflate ().

Для добавления меню параметров в активити необходимо переопределить метод onCreateOptionsMenu (). В этом методе можно загрузить собственный ресурс меню в класс Menu (листинг 5).

Листинг 5. Добавления меню параметров

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu, menu);
    return true;
}
```

Когда пользователь выбирает пункт меню параметров, система вызывает метод onOptionsItemSelected (). Этот метод передает выбранный класс MenuItem. Идентифицировать пункт меню можно, вызвав метод getItemId (), который возвращает уникальный идентификатор пункта меню. Этот идентификатор можно сопоставить с известными пунктами меню, чтобы выполнить соответствующее действие.

В листинге 6 приведён фрагмент кода активити, в котором имеются пункты, меняющие цвет фона активити и кнопка exit завершающая активити.

Листинг 6. Обработка выбора пункта меню

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    final LinearLayout mylayout = (LinearLayout)findViewById(R.id.root);
        switch (item.getItemId()) {
        case R.id.red:
            mylayout.setBackgroundColor(Color.parseColor("#FF0000"));
            return true;
        case R.id.green:
            mylayout.setBackgroundColor(Color.parseColor("#00FF00"));
            return true;
        case R.id.blue:
            mylayout.setBackgroundColor(Color.parseColor("#0000FF"));
            return true;
        case R.id.exit:
            finish();
            return true;
        default:
            return super.onOptionsItemSelected(item);
        }
}
```

Активы

Активы — это файлы произвольного типа и содержания, располагаемые в каталоге assets/. Этот каталог находится на одном уровне с каталогом res/ в дереве проекта. Особенностью этого каталога является то, что файлы в assets/ не генерируют идентификаторы ресурса в классе R. В программном коде необходимо явно определять путь к файлу для доступа к нему. Путь к файлу — это относительный путь, начинающийся с assets/. Для обращения к этим файлам используется класс AssetManager. Этот каталог, в отличие от подкаталога res/, может иметь произвольную глубину подкаталогов и произвольные имена файлов и подкаталогов.

Рассмотрим в качестве примера работу со сторонними шрифтами (листинг 7). Шрифты необходимо разместить в папке assets/fonts. Можно использовать любые шрифты в формате ttf или otf. В данном примере используется свободный шрифт PenguinAttackCyrillic (авторы — Dustin Norlander и Denis Ignatov). В классе Activity необходимо создать объект Typeface, используя вызов статического метода Typeface.createFromAsset (). Метод createFromAsset () принимает два параметра:

- объект AssetManager, который можно получить вызовом метода getAssets ();
- путь к файлу актива.

Листинг 7. Работа со сторонними шрифтами

```
final TextView text1 = (TextView)findViewById(R.id.text1);
text1.setTypeface(Typeface.createFromAsset(
                  getAssets(),"fonts/PenguinAttackCyrillic.otf"));
```

Задание

Разработаем приложение, использующее различные типы ресурсов (графические, строковые, меню, цвета, стили и темы) и активы.

Создайте новый проект с именем Resources. При создании проекта выберите создание пустого активити (англ. Empty Activity). Откройте папку ресурсов (рис. 2).

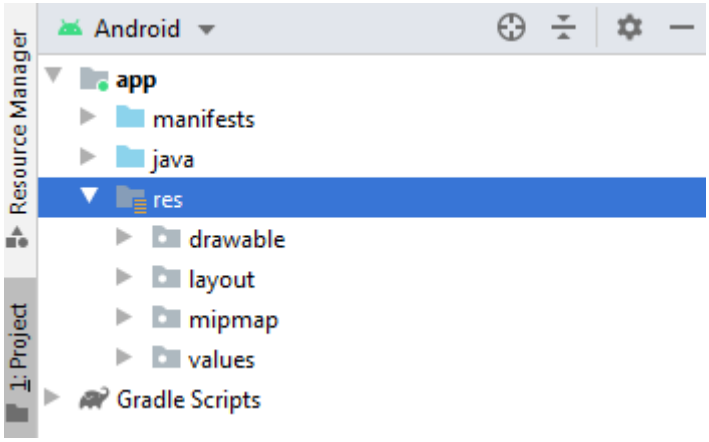

Рисунок 2. Папка ресурсов.

Скопируйте в папку drawable графический ресурс с именем background (например, background.jpg или background.png), который будет использоваться как фоновое изображение для активити.

В файле strings.xml, расположенном в папке values, создайте следующие строковые ресурсы (листинг 8).

Листинг 8. Файл strings.xml

```
<resources>
    <string name="app_name">Resources</string>
    <string name="text">Hello world!</string>
    <string name="red">Red</string>
    <string name="green">Green</string>
```

```
    <string name="blue">Blue</string>
    <string name="color">Color</string>
    <string name="exit">Exit</string>
</resources>
```

Затем создайте файл для переведённых на русский язык строковых ресурсов (рис. 3, рис. 4).

Рисунок 3. Создание нового файла ресурсов.

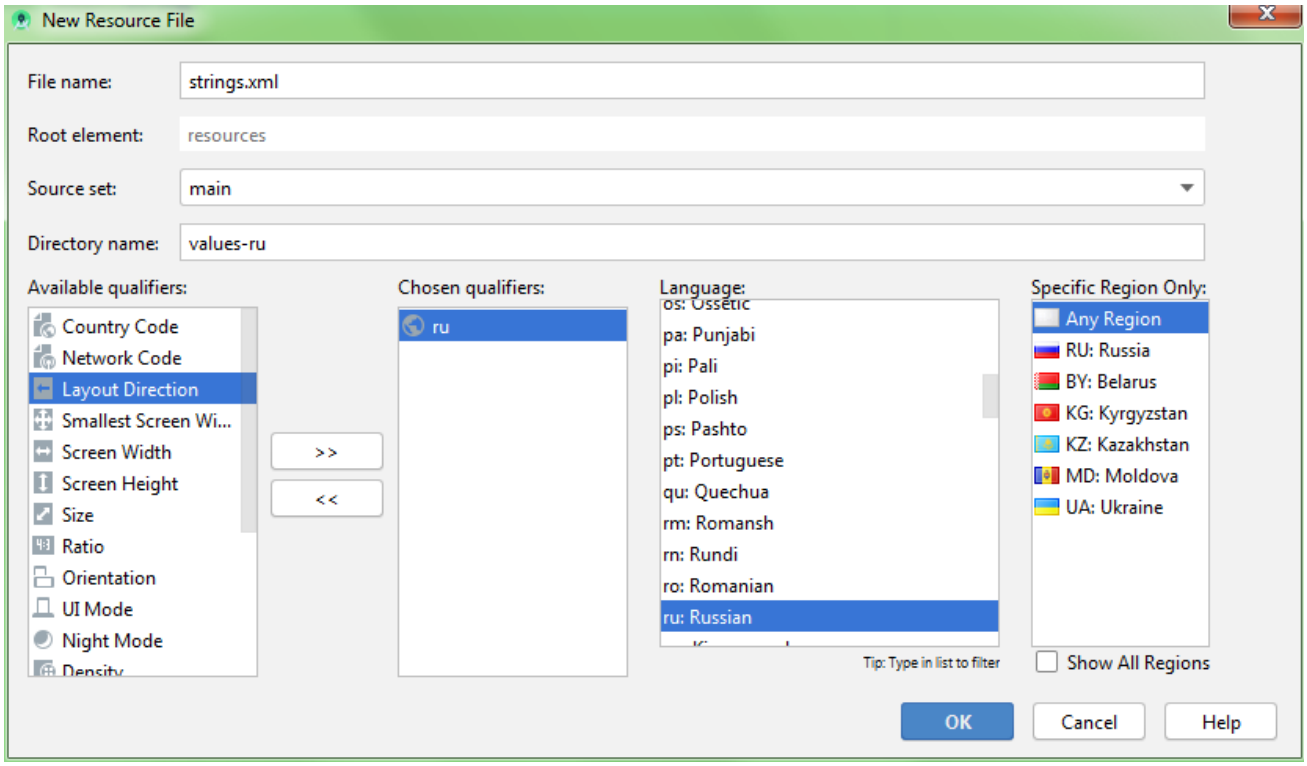

Рисунок 4. Создание строковых ресурсов для русского языка.

В результате будет создана папка values-ru с файлом stings.xml (рис. 5), в который нужно поместить переведённые строковые ресурсы (листинг 9).

Листинг 9. Файл strings.xml (ru)

```
<resources>
    <string name="app_name">Ресурсы</string>
    <string name="text">Привет, мир!</string>
    <string name="red">Красный</string>
    <string name="green">Зелёный</string>
    <string name="blue">Синий</string>
    <string name="color">Цвет</string>
    <string name="exit">Выход</string>
</resources>
```

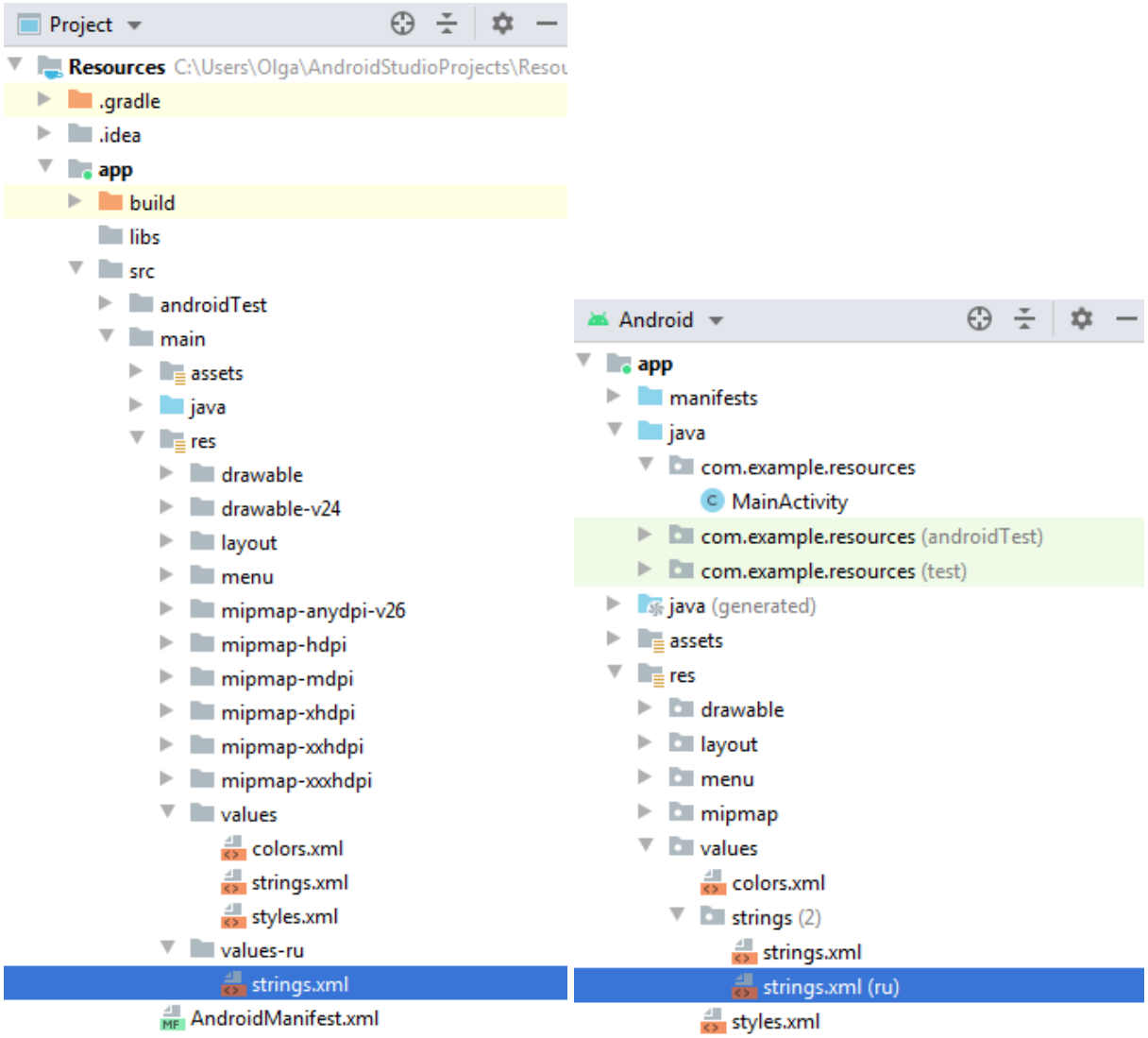

Рисунок 5. Файл strings.xml (ru)

Теперь создадим цветовые ресурсы в файле colors.xml (листинг 10).

Листинг 10. Файл colors.xml

```
<resources>
    <color name="colorPrimary">#6200EE</color>
    <color name="colorPrimaryDark">#3700B3</color>
    <color name="colorAccent">#03DAC5</color>
    <color name="red">#CC1B30</color>
    <color name="green">#009900</color>
    <color name="blue">#0099FF</color>
</resources>
```

Создадим папке menu для ресурсов меню (рис. 6).

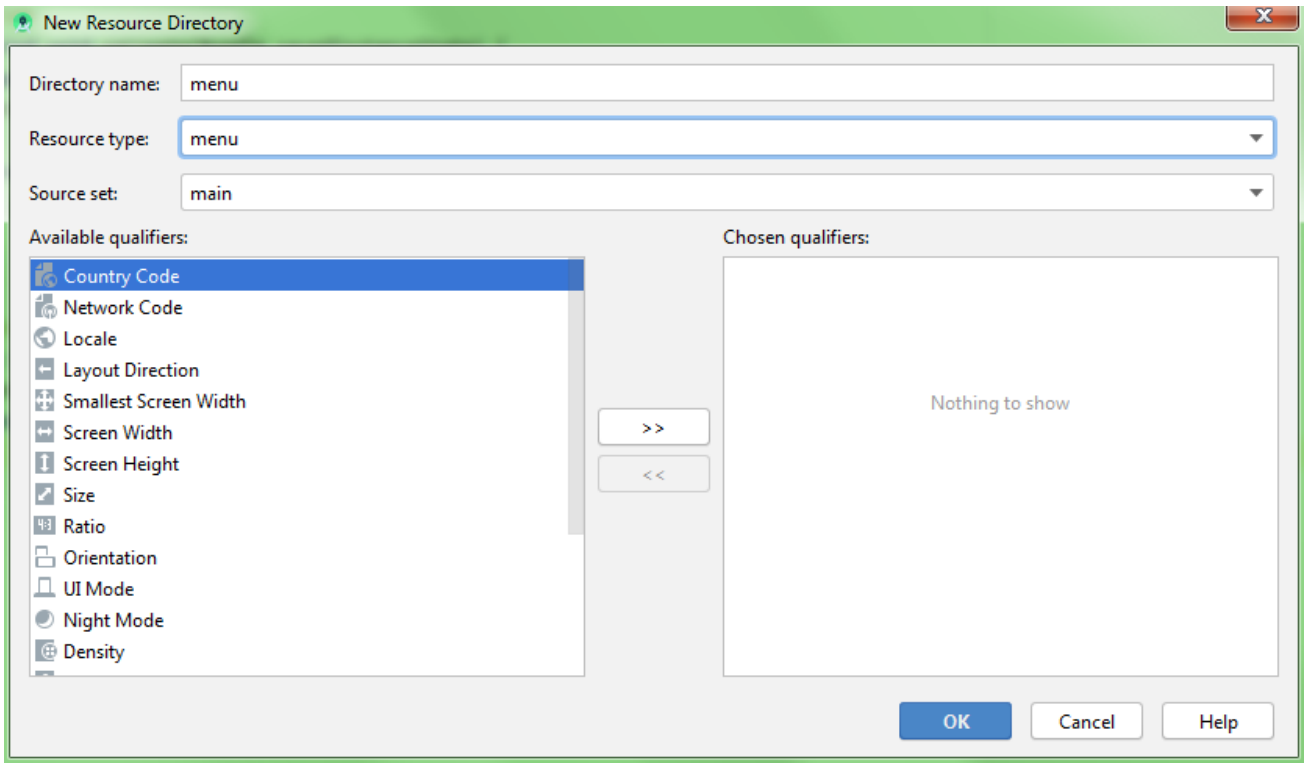

Рисунок 6. Создание папки menu

В папке menu создадим файл menu.xml и опишем пункты, которые будут в меню параметров: будет кнопка выхода из активити, а также подменю с выбором цвета фона активити (листинг 11).

Листинг 11. Файл menu.xml

```
<menu  xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/color"
          android:title="@string/color">
        <menu>
            <item android:id="@+id/blue"
                android:title="@string/blue"/>
            <item android:id="@+id/red"
                android:title="@string/red"/>
            <item android:id="@+id/green"
                android:title="@string/green"/>
        </menu>
    </item>
    <item android:id="@+id/exit"
          android:title="@string/exit"/>
</menu>
```

Теперь зададим стили и темы в файле styles.xml (листинг 12). В файле манифеста приложения должна быть объявлена ссылка на тему для приложения (листинг 13).

Листинг 12. Файл styles.xml

```
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="android:windowBackground">@drawable/background</item>
        <item name="android:windowNoTitle">true</item>
    </style>
    <style name="SpecialText" parent="TextAppearance.AppCompat.Title">
        <item name="android:textSize">36sp</item>
        <item name="android:textColor">#548f00</item>
        <item name="android:background">#b7e4fb</item>
    </style>
</resources>
```

Листинг 13. Элемент application из файла AndroidManifest.xml

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
```

В файле компоновки определим единственный элемент TextView, и для этого элемента зададим созданный в файле styles.xml стиль, используя атрибут style (листинг 14).

Листинг 14. Файл activity_main.xml

```
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
    android:id="@+id/root"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center">
    <TextView
        style="@style/SpecialText"
        android:id="@+id/my_text"
        android:text="@string/text"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:gravity="center"
        android:padding="5dp"/>
</LinearLayout>
```

Создадим папку assets для активов (рис. 7).

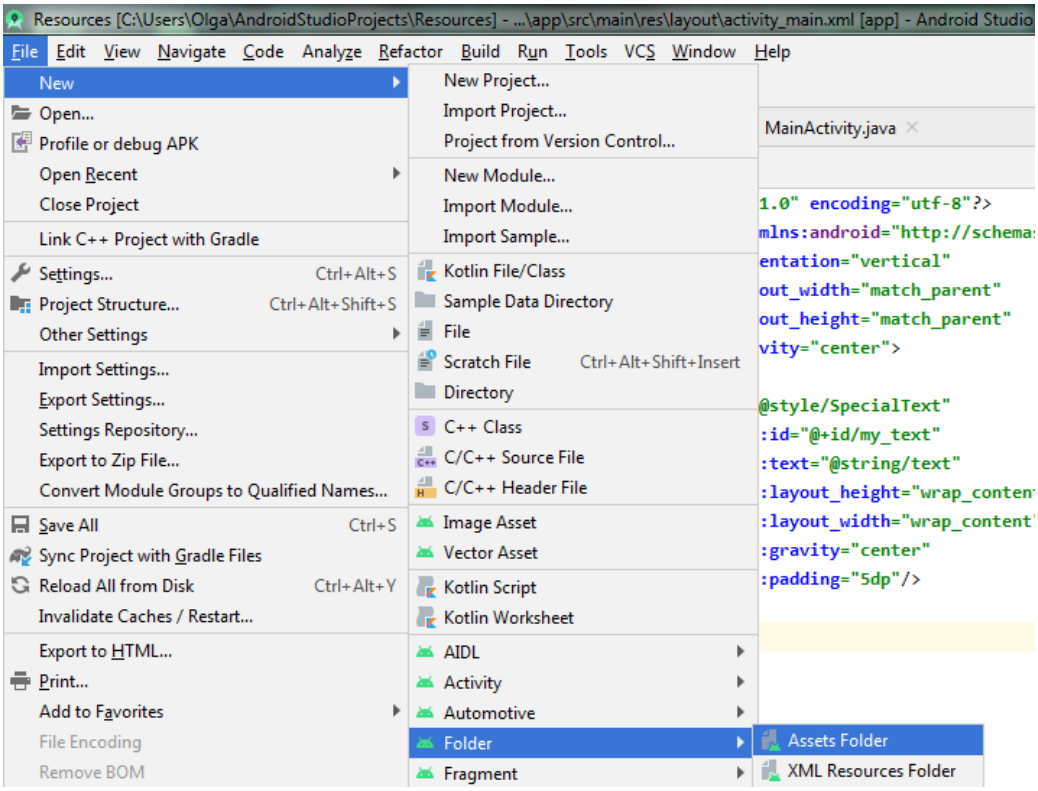

Рисунок 7. Создание папки assets

Создадим в папке assets вложенную папку fonts и скопируем в неё шрифт PenguinAttackCyrillic.otf (рис. 8).

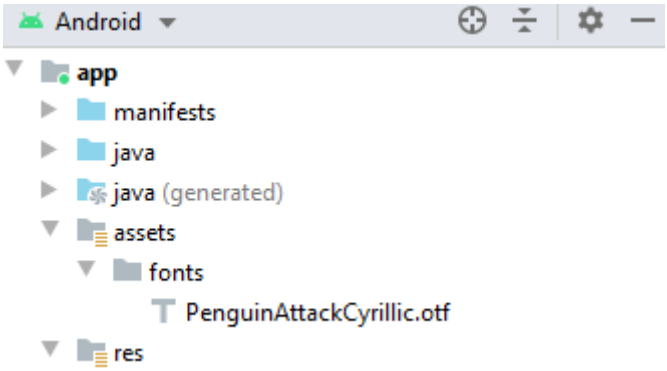

Рисунок 8. Папка с шрифтами

И наконец, отредактируем файл MainActivity.java, добавим в него код создающий шрифт на основе актива и назначающий его текстовому полю, а также загрузим меню из файла menu.xml и опишем действия при выборе пунктов меню (листинг 15).

Листинг 15. Файл MainActivity.java

```
package com.example.resources;
import androidx.appcompat.app.AppCompatActivity;
import android.graphics.Color;
import android.graphics.Typeface;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.LinearLayout;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final TextView my_text = (TextView)findViewById(R.id.my_text);
        my_text.setTypeface(Typeface.createFromAsset(
                getAssets(), "fonts/PenguinAttackCyrillic.otf"));
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        final LinearLayout mylayout =
(LinearLayout)findViewById(R.id.root);
        switch (item.getItemId()) {
            case R.id.red:

mylayout.setBackgroundColor(getResources().getColor(R.color.red));
                return true;
            case R.id.green:

mylayout.setBackgroundColor(getResources().getColor(R.color.green));
                return true;
            case R.id.blue:

mylayout.setBackgroundColor(getResources().getColor(R.color.blue));
                return true;
            case R.id.exit:
                finish();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

Запустите приложение на виртуальном или физическом устройстве Android. Посмотрите, как изменение языка по умолчанию (английский, русский) в настройках устройства влияет на приложение.

На рисунке 9 приведены скриншоты работы приложения.

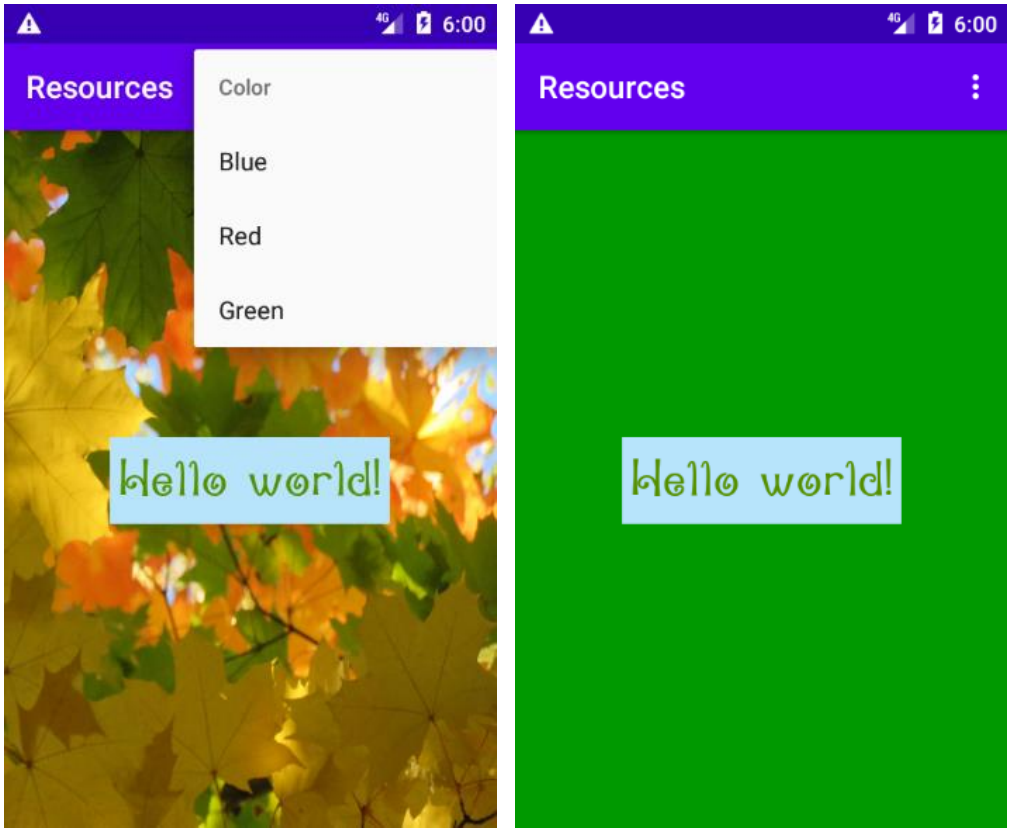

Рисунок 9. Результаты работы приложения.

Самостоятельно дополните приложение, добавьте различные фоновые картинки для горизонтальной и вертикальной ориентации экрана, добавьте новые пункты меню, создающие на компоновке дополнительные элементы (например, текстовые поля).

Контрольные вопросы:

1. Какие типы ресурсов хранятся в файле arrays.xml?
2. Какие типы ресурсов хранятся в файле colors.xml?
3. Какие типы ресурсов хранятся в файле dimens.xml?
4. Какие типы ресурсов хранятся в файле drawables.xml?
5. Какие типы ресурсов хранятся в файле strings.xml?
6. Чем стили отличаются от тем?

7. В каком файле задаются стили?
8. В каком файле задаются темы?
9. Как объявляется ссылка на тему в файле манифеста приложения?
10. Чем ресурсы отличаются от активов?
11. Какие примеры активов Вы можете привести?
12. Как реализовать использование произвольных шрифтов в приложении?
13. Как возможно осуществить локализацию ресурсов в приложении?