Практическая работа 10

Методические материалы

Стили и темы. Меню. Группы в меню и подменю

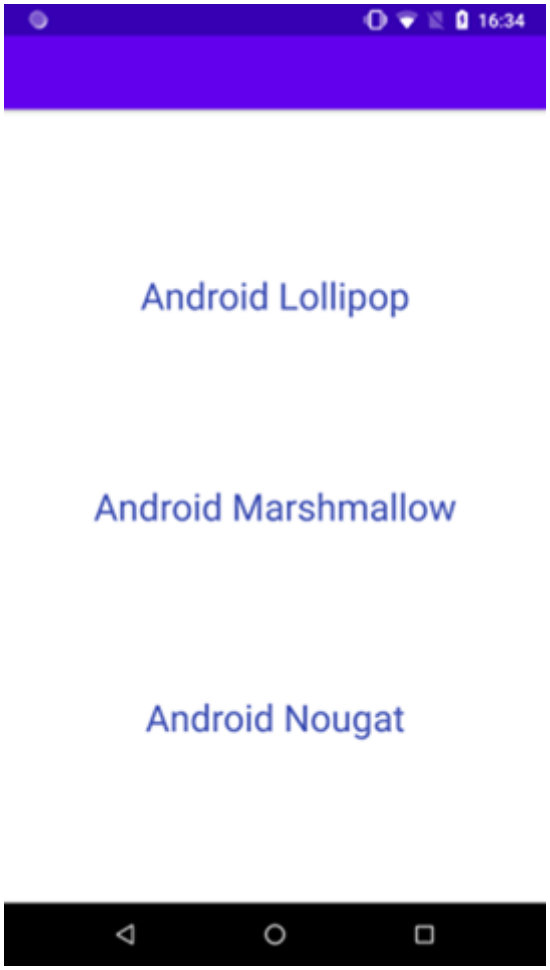
Мы можем настроить элемент с помощью различных атрибутов, которые задают высоту, ширину, цвет фона, текста и так далее. Но если у нас несколько элементов используют одни и те же настройки, то мы можем объединить эти настройки в стили.

Например, пусть у нас есть несколько элементов TextView:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textSize="28sp"
        android:textColor="#3f51b5"
        android:text="Android Lollipop"
```

```
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toTopOf="@+id/textView2"
    />
<TextView
    android:id="@+id/textView2"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:textSize="28sp"
    android:textColor="#3f51b5"
    android:text="Android Marshmallow"

    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintBottom_toTopOf="@+id/textView3"
    app:layout_constraintTop_toBottomOf="@+id/textView1"
    />
<TextView
    android:id="@+id/textView3"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:textSize="28sp"
    android:textColor="#3f51b5"
    android:text="Android Nougat"

    app:layout_constraintLeft_toLeftOf="parent"
```

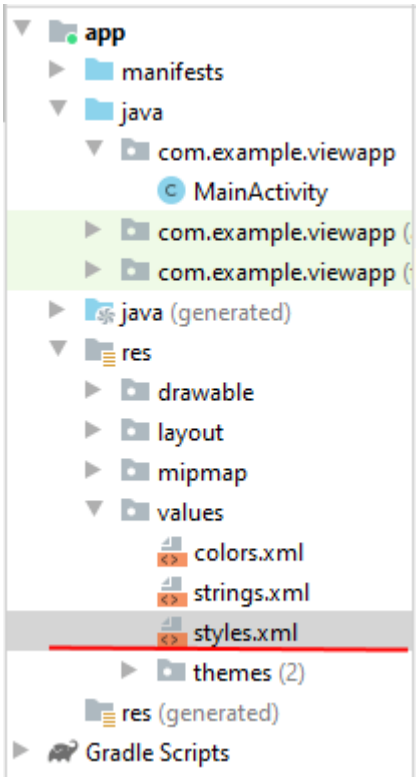
```
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```



Стили в Android

Все эти TextView имеют одинаковый набор свойств, и, к примеру, если нам захочется изменить цвет текста, то придется менять его у всех трех TextView. Данный подход не оптимален, а более оптимальный подход представляет использование стилей, которые определяются в проекте в папке **res/values**.

Итак, добавим в проект в папку **res/values** новый элемент **Value Resourse File**, который назовем **styles.xml**:



styles.xml в Android Studio

Определим в файле styles.xml следующее содержимое:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="TextViewStyle">
        <item name="android:layout_width">0dp</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#3f51b5</item>
        <item name="android:textSize">28sp</item>
        <item name="android:gravity">center</item>
    </style>
</resources>
```

Здесь определен новый стиль TextViewStyle, который с помощью элементов item задает значения для атрибутов TextView.

Стиль задается с помощью элемента <style>. Атрибут name указывает на название стиля, через которое потом можно ссылаться на него. Необязательный атрибут parent устанавливает для данного стиля родительский стиль, от которого дочерний стиль будет наследовать все свои характеристики.

С помощью элементов item устанавливаются конкретные свойства виджета, который принимает в качестве значения атрибута name имя устанавливаемого свойства.

Теперь применим стиль, изменим файл activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView1"

        style="@style/TextViewStyle"

        android:text="Android Lollipop"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/textView2"
        />
    <TextView
        android:id="@+id/textView2"
        style="@style/TextViewStyle"
```

```
        android:text="Android Marshmallow"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/textView3"
        app:layout_constraintTop_toBottomOf="@+id/textView1"
        />
    <TextView
        android:id="@+id/textView3"
        style="@style/TextViewStyle"
        android:text="Android Nougat"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2"
        />
 </androidx.constraintlayout.widget.ConstraintLayout>
```

Используя определение style="@style/TextViewStyle" текстовое поле связывается с определением стиля. Итоговый результат буде тот же, что и раньше, только кода становится меньше. А если мы захотим поменять какие-то характеристики, то достаточно изменить нужный элемент item в определении стиля.

Темы

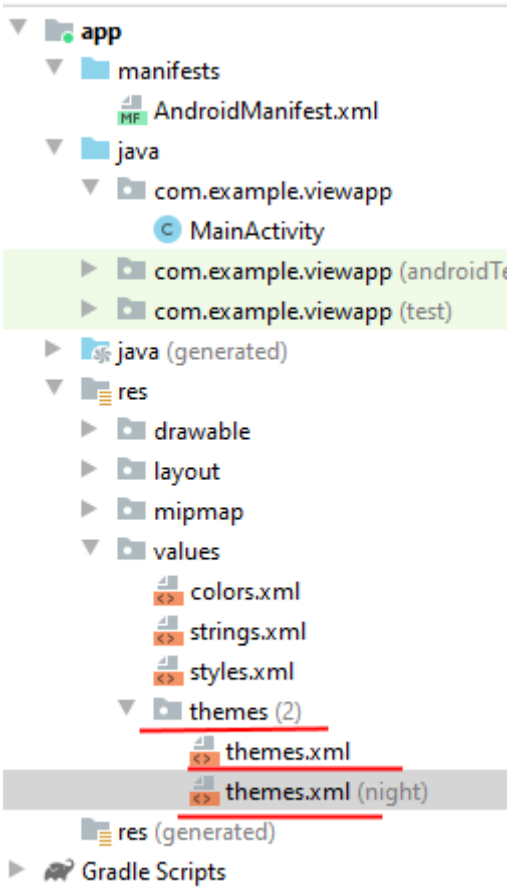
Кроме применение отдельных стилей к отдельным элементам, мы можем задавать стили для всего приложения или activity в виде тем. Тема предтавляет коллекцию атрибутов, которые применяются в целом ко всему приложению, классу activity или иерархии виджетов.

Мы можем сами создать тему. Однако Android уже предоставляет несколько предустановленных тем для стилизации приложения, например, Theme.AppCompat.Light.DarkActionBar и ряд других.

По умолчанию приложение уже применяет темы. Так, откроем файл **AndroidManifest.xml**. В нем мы можем увидеть следующее определение элемента application, представляющего приложение:

```
<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/Theme.ViewApp">
```

Задание темы происходит с помощью атрибута android:theme. В данном случае используется ресурс, который называется в моем случае **Theme.ViewApp**. По умолчанию файлы тем определены в папке **res/values**. В частности, здесь можно найти условный каталог **themes**, в котором по умолчанию есть два элемента: **themes.xml**:



Темы и стили в Android Studio и Java

Один файл представляет светлую тему, а другой - темную. Например, откроем файл themes.xml со светлой темой:

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.ViewApp"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
```

```
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Здесь мы можем увидеть, что тема определяется, как и стиль с помощью элемента **style**. Атрибут **parent** указывает на родительскую тему, от которой текущая тема берет все стилевые характеристики. То есть тема "Theme.ViewApp" использует другую тему - "Theme.MaterialComponents.DayNight.DarkActionBar". И кроме того, определяет ряд своих собственных стилей.

Также можно заметить, что здесь определяются не только характеристики для атрибутов, но и семантические имена, например, colorPrimary, которому сопоставлен ресурс "@color/purple_500".

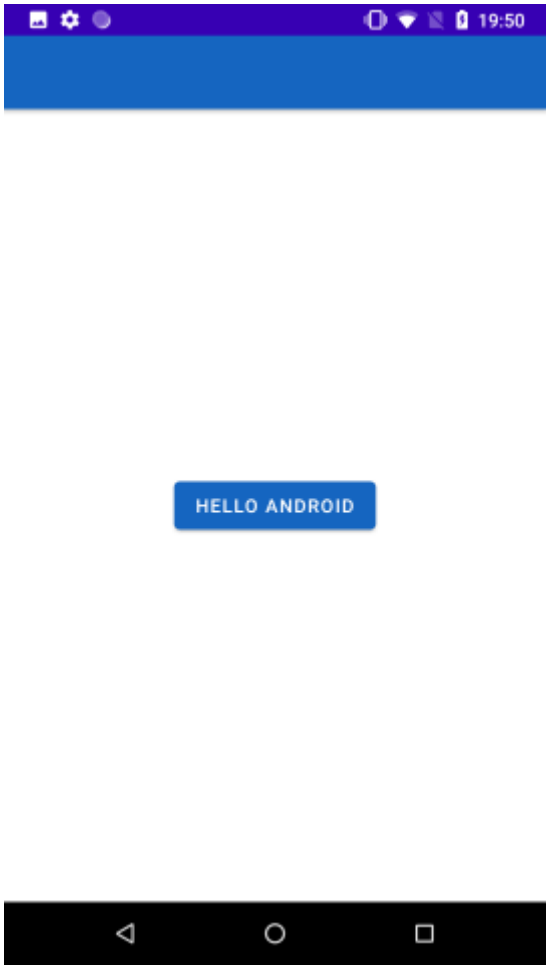
При необходимости мы можем изменить эти характеристики или дополнить тему новыми стилевыми характеристиками. Например, изменим цвет свойства colorPrimary, которое применяется в том числе в качестве фонового цвета заголовка и кнопки:

```
<item name="colorPrimary">#1565C0</item>
```

И соответственно изменится цвет по умолчанию для фона заголовка и кнопки:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

```
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello Android"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

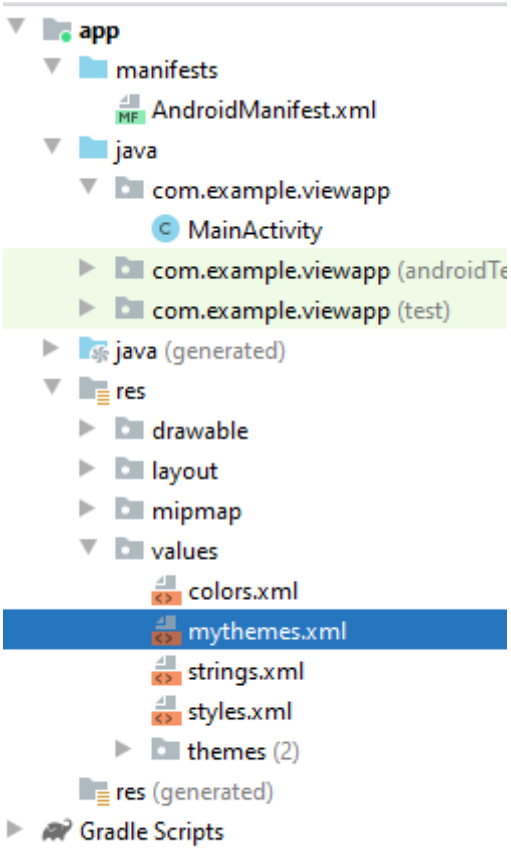


Theme Editor in Android Studio

Создание собственной темы

Вместо использования встроенных тем мы можем создать свою. Для этого добавим в папку res/values новый файл mythemes.xml и определим в нем следующее содержимое:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="MyTheme" parent="Theme.AppCompat.Light">
        <item name="android:textColor">#FF018786</item>
        <item name="android:textSize">28sp</item>
    </style>
</resources>
```



Определение своей темы

Итак, мы создали стиль "MyTheme", который унаследован от стиля Theme.AppCompat.Light. В этом стиле мы переопределили два свойства: высоту шрифта (textSize) - 28sp, а также цвет текста (textColor) - #FF018786.

Теперь определим этот стиль в качестве темы приложения в файле **AndroidManifest.xml:**

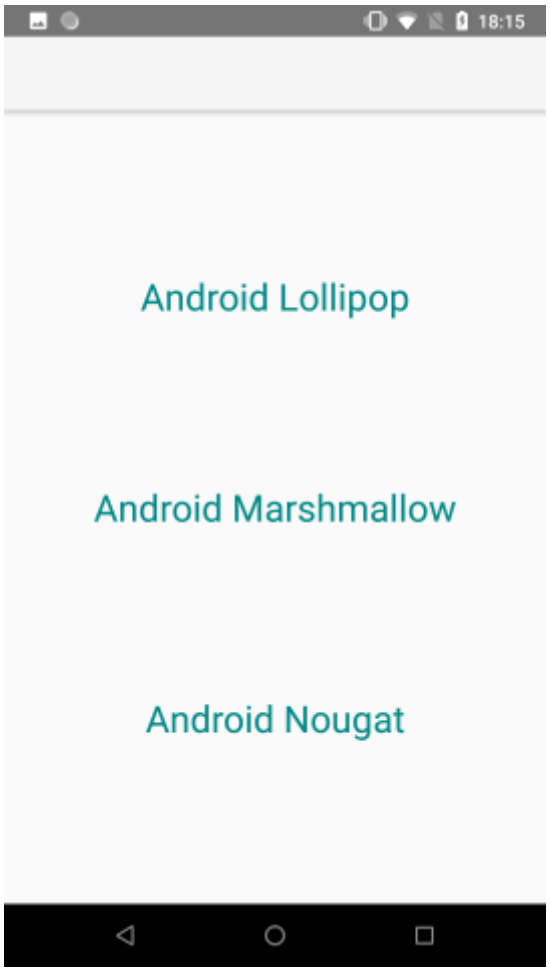
```
<application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"

        android:theme="@style/MyTheme"><!-- применение темы -->
```

Пусть у нас будет следующая разметка в **activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android Lollipop"
        app:layout_constraintLeft_toLeftOf="parent"
```

```
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/textView2"
        />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android Marshmallow"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/textView3"
        app:layout_constraintTop_toBottomOf="@+id/textView1"
        />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android Nougat"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2"
        />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Как видно, для элементов TextView не устанавливается атрибут textSize и textColor, однако поскольку они определены в теме, которая применяется глобально к нашему приложению, то элементы TextView будут подхватывать эти стилевые характеристики:



Создание новой темы для Android

Применение темы к activity

Выше темы применялись глобально ко всему приложению. Но также можно применить их к отдельному классу Activity. Для этого надо подкорректировать файл манифеста AndroidManifest. Например:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.viewapp"
    android:versionCode="1"
```

```
    android:versionName="1.0">

    <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/Theme.ViewApp">
        <activity android:name=".MainActivity"
android:theme="@style/MyTheme">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Атрибут **android:theme** элемента <activity> указывает на применяемую к MainActivity тему. То есть глобально к приложению применяется тема "Theme.ViewApp", а к MainActivity - "MyTheme".

Применение темы к иерархии виджетов

Также можно применить тему к иерархии виджетов, установив атрибут **android:theme** у элемента, к которому (включая его вложенные элементы)

мы хотим применить тему. Например, примение темы к ConstraintLayout и ее элементам:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    android:theme="@style/MyTheme">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Android Lollipop"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

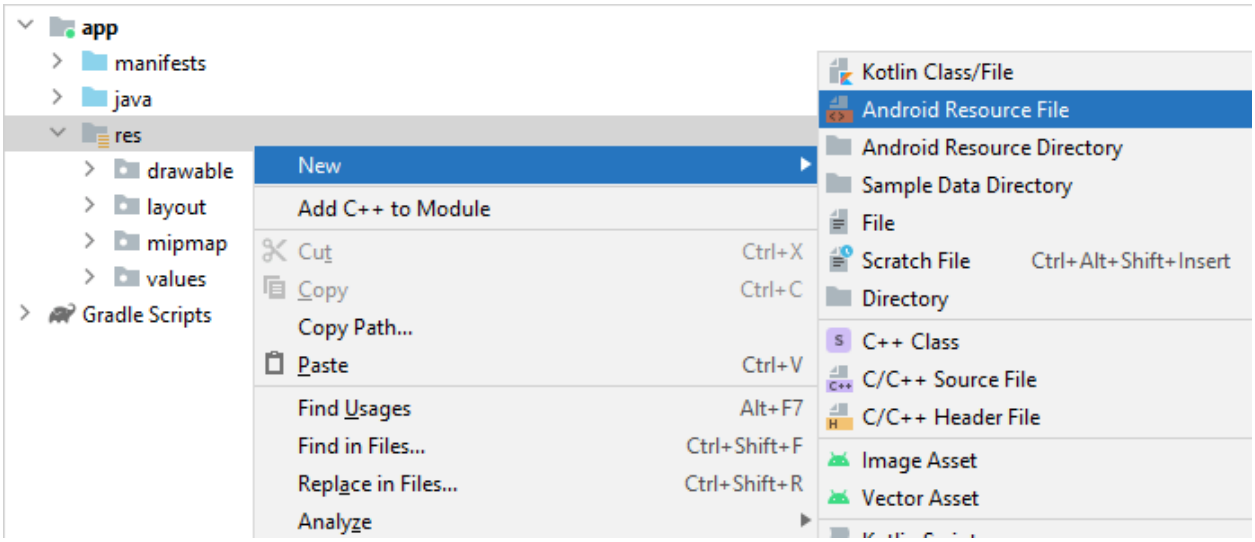
Меню

Создание меню

Меню в приложениях представляет класс **android.view.Menu**, и каждая activity ассоциируется с объектом этого типа. Объект android.view.Menu может включать различное количество элементов, а те в свою очередь могут хранить подэлементы.

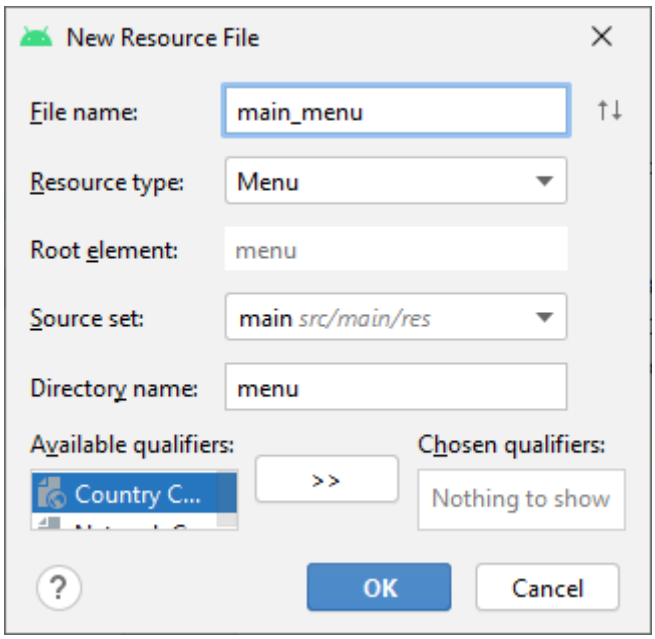
Определение меню в xml

Меню, как и файлы интерфейса или изображений, также представляет собой ресурс. Однако при создании нового проекта с Empty Activity по умолчанию нет никаких ресурсов меню, поэтому при необходимости их нужно добавлять вручную. Так, для определения ресурсов меню в проекте нажмем правой кнопкой мыши в проекте на каталог res и далее в открывшемся списоке выберем пункт New -> Android Resource File:



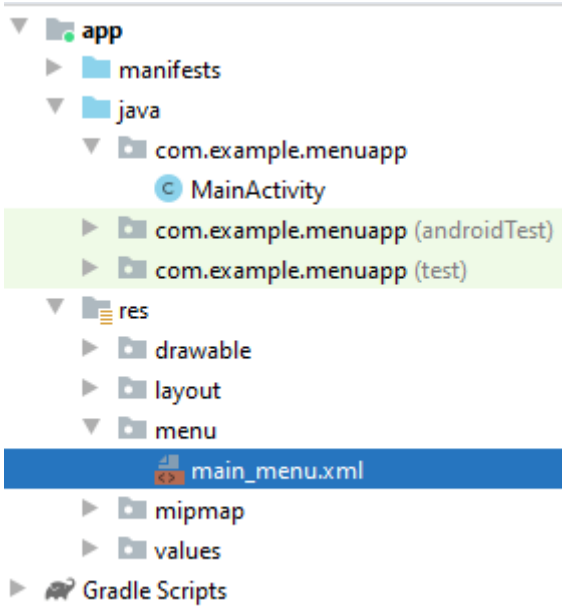
Добавление ресурсов меню в Android Studio

Далее в появившемся окне укажем для имени файла название **main_menu**, а для поля **Resource Type** (тип ресурса) выберем **Menu**:



Создание ресурса меню в Android Studio

После этого в каталоге res будет создан подкаталог menu, в котором будет находиться файл **main_menu.xml**.



Определение меню в Android Studio и Java

По умолчанию этот файл определяет один пустой элемент menu:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
</menu>
```

Изменим содержимое файла, определив несколько пунктов:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item
      android:id="@+id/action_settings"
      android:orderInCategory="1"
      android:title="Настройки" />
   <item
      android:id="@+id/save_settings"
      android:orderInCategory="3"
      android:title="Сохранить" />
   <item
      android:id="@+id/open_settings"
      android:orderInCategory="2"
      android:title="Открыть" />
</menu>
```

Тег <menu> является корневым узлом файла и определяет меню, состоящее из одного или нескольких элементов <item> и <group>.

Элемент <item> представляет объект MenuItem, которой является одним из элементов меню. Этот элемент может содержать внутренний подэлемент <menu>, с помощью которого создается подменю.

Элемент <item> включает следующие атрибуты, которые определяют его внешний вид и поведение:

- **android:id**: уникальный id элемента меню, который позволяет его опознать при выборе пользователем и найти через поиск ресурса по id

- **android:icon**: ссылка на ресурс drawable, который задает изображение для элемента (android:icon="@drawable/ic_help")

- **android:title**: ссылка на ресурс строки, содержащий заголовок элемента. По умолчанию имеет значение "Settings"

- **android:orderInCategory**: порядок следования элемента в меню

Наполнение меню элементами

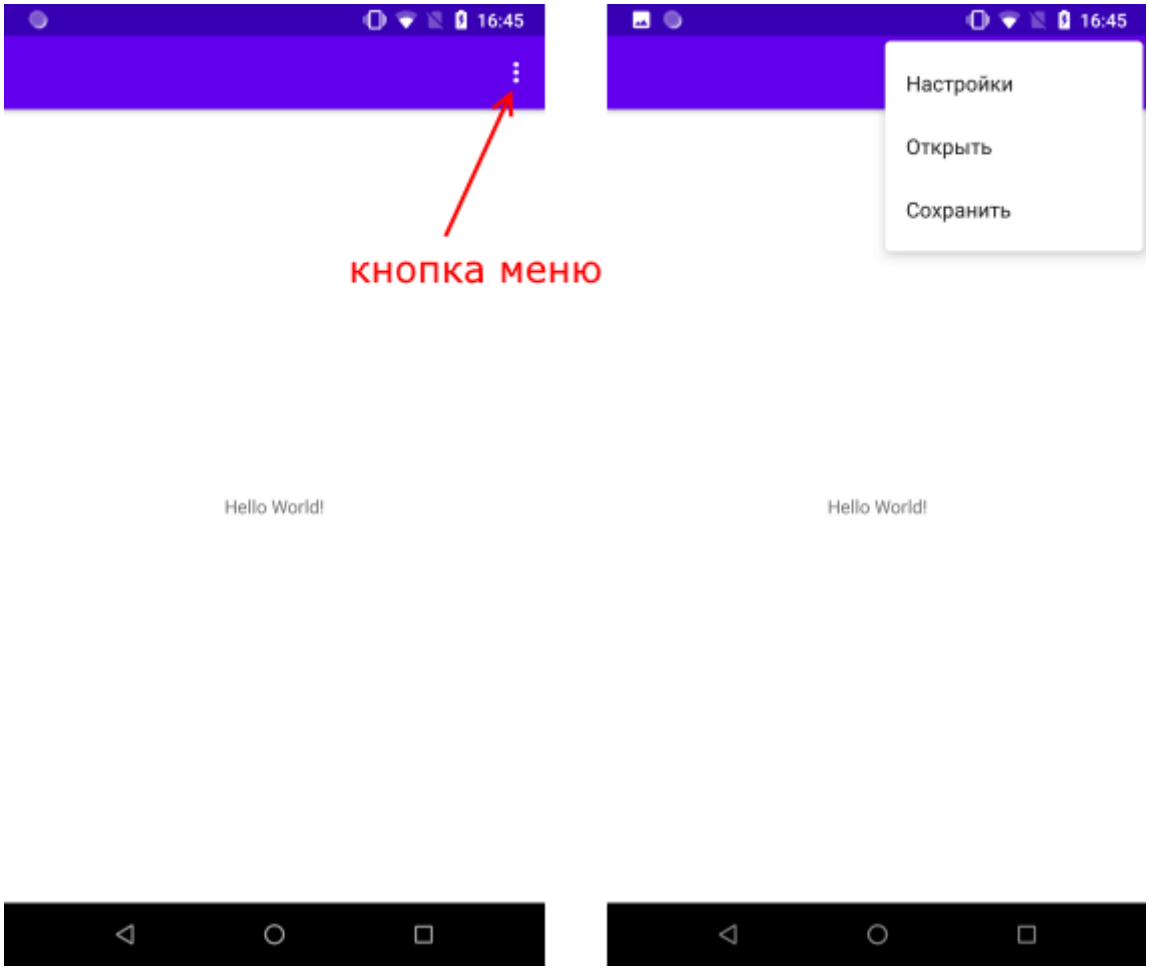
Мы определили меню с тремя элементами, но само определение элементов в файле еще не создает меню. Это всего лишь декларативное описание. Чтобы вывести его на экран, нам надо использовать его в классе Activity. Для этого надо переопределить метод onCreateOptionsMenu. Итак, перейдем к классу MainActivity и изменим его следующим образом:

```
package com.example.menuapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.Menu;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
```

```
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }
}
```

Метод getMenuInflater возвращает объект MenuInflater, у которого вызывается метод inflate(). Этот метод в качестве первого параметра принимает ресурс, представляющий наше декларативное описание меню в xml, и наполняет им объект menu, переданный в качестве второго параметра.

Запустим приложение по умолчанию и нажмем на кнопку меню в правом верхнем углу:



Меню по умолчанию в Android

Обработка нажатий в меню

Если мы нажмем на любой из пунктов меню, то ничего не произойдет. Чтобы привязать к меню действия, нам надо переопределить в классе activity **onOptionsItemSelected.**

Для вывода выбранного элемена меню в файле **activity_main.xml** определим текстовое поле с id=header:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/selectedMenuItem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:textSize="28sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

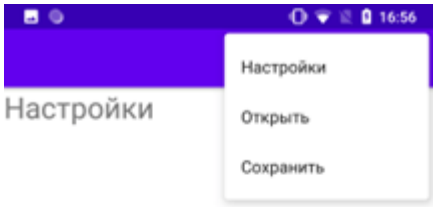
И изменим класс MainActivity:

```
package com.example.menuapp;
```

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        TextView headerView = findViewById(R.id.selectedMenuItem);
        switch(id){
            case R.id.action_settings :
                headerView.setText("Настройки");
```

```
            return true;
        case R.id.open_settings:
            headerView.setText("Открыть");
            return true;
        case R.id.save_settings:
            headerView.setText("Сохранить");
            return true;
    }
    //headerView.setText(item.getTitle());
    return super.onOptionsItemSelected(item);
  }
}
```

Чтобы понять, какой пункт меню выбран, вначале получаем его идентификатор int id = item.getItemId(). Затем пробегаемся в конструкции switch..case и выбираем нужный вариант и в зависимости от выбора производим определенные действия - в данном случае устанавливаем текст TextView.



Меню и onOptionsItemSelected в Android и Java

Стоит отметить, что в данном случае, если наша задача заключалась, чтобы просто в выводе текста выбранного пункта меню, то мы вместо конструкции switch просто могли написать так:

```
headerView.setText(item.getTitle());
```

Программное создание меню

Кроме определения элементов меню в xml, можно также создать меню программным способом. Для добавления новых пунктов меню используется метод **add()** класса Menu.

Например, изменим код **MainActivity**:

```
package com.example.menuapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.widget.TextView;


public class MainActivity extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        menu.add("Настройки");
```

```
        menu.add("Открыть");
        menu.add("Сохранить");
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        String title = item.getTitle().toString();
        TextView headerView = findViewById(R.id.selectedMenuItem);
        headerView.setText(title);

        return super.onOptionsItemSelected(item);
    }
}
```

Использованная версия метода add() принимает заголовок для пункта меню.

Группы в меню и подменю

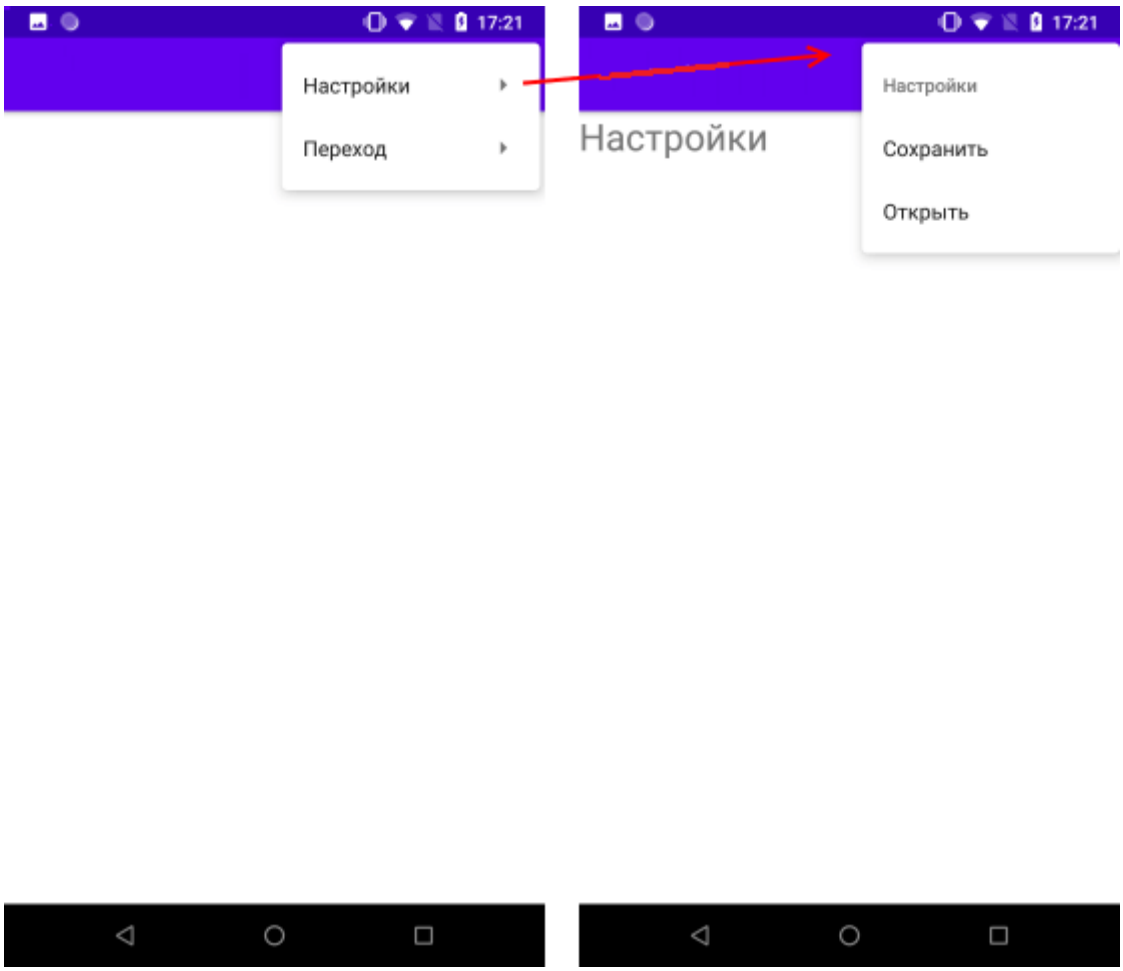
Создание подменю

Для создания подменю в файле разметки меню определим внутренний элемент menu:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_settings"
        android:title="Настройки">
        <menu>
            <item android:id="@+id/save_settings"
```

```
            android:title="Сохранить" />
        <item android:id="@+id/open_settings"
            android:title="Открыть" />
    </menu>

  </item>
  <item
    android:id="@+id/action_move"
    android:title="Переход">
    <menu>
      <item android:id="@+id/forward"
        android:title="Вперед" />
      <item android:id="@+id/back"
        android:title="Назад" />
    </menu>

  </item>
</menu>
```

После нажатия на меню отобразятся элементы верхнего уровня, по нажатию на которые мы можем перейти к подменю:



Подменю в Android

Группы в меню

Использование элемента group позволяет оформить элементы меню в группу:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <group android:checkableBehavior="single">
        <item
            android:id="@+id/action_settings"
            android:title="Настройки"
            android:checked="true" />
        <item android:id="@+id/save_settings"
```

```
            android:title="Сохранить" />
        <item android:id="@+id/open_settings"
            android:title="Открыть"  />
    </group>
</menu>
```

В определении группы мы можем установить атрибут **android:checkableBehavior**. Этот атрибут может принимать следующие значения: single(у каждого элемента создается радиокнопка), all (для каждого элемента создается флажок) и none.

В данном случае для каждого элемента будет создаваться радиокнопка (визуально кружок). И для первого элемента устанавливается отмеченная радиокнопка (android:checked="true").

В файле разметки интерфейса **activity_main.xml** также пусть будет определено текстовое поле:

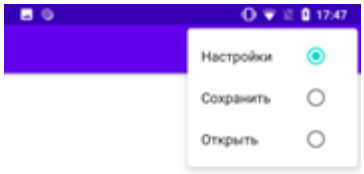
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/selectedMenuItem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="28sp"
        app:layout_constraintLeft_toLeftOf="parent"
```

```
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

А в классе MainActivity определим выделение радиокнопки у выбранного пункта меню:

```
package com.example.menuapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        TextView headerView = findViewById(R.id.selectedMenuItem);
        switch(id){
            case R.id.action_settings :
                headerView.setText("Настройки");
                return true;
            case R.id.open_settings:
                headerView.setText("Открыть");
                return true;
            case R.id.save_settings:
                headerView.setText("Сохранить");
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```



Группы в меню в Android

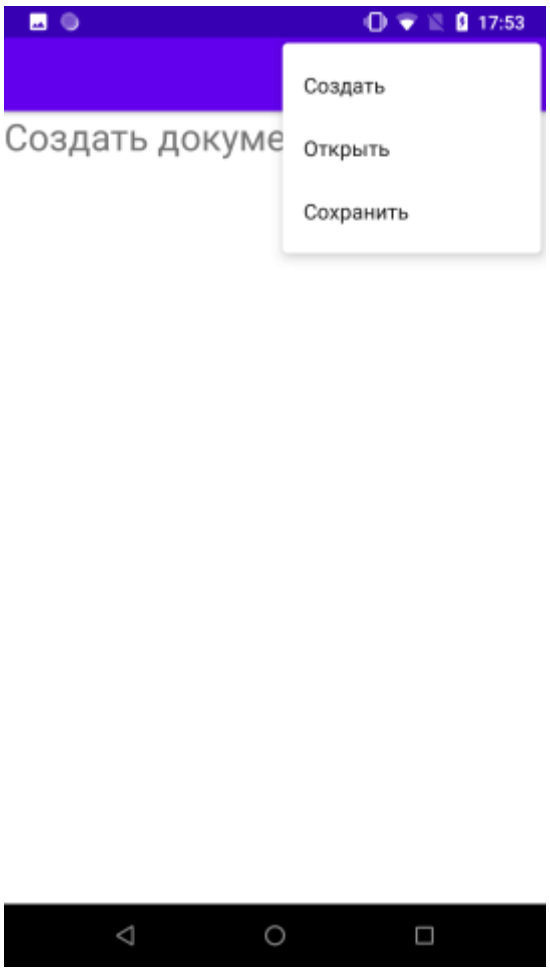
Программное создание групп в меню и подменю

Также группы и подменю можно создавать программным способом. Так, изменим код **MainActivity**:

```
package com.example.menuapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);

        menu.add(0        // Группа
                ,1        // id
                ,0        //порядок
                ,"Создать");  // заголовок
```

```
        menu.add(0,2,1,"Открыть");
        menu.add(0,3,2,"Сохранить");
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        TextView headerView = findViewById(R.id.selectedMenuItem);

        switch(id){
            case 1 :
                headerView.setText("Создать документ");
                return true;
            case 2:
                headerView.setText("Открыть документ");
                return true;
            case 3:
                headerView.setText("Сохранить документ");
                return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

Использованная здесь версия метода add() добавляет пункт в меню, принимая следующие параметры: номер группы, id, порядок элемента в меню и заголовок элемента.



Программное создание меню в Android

Задание

1. Реализовать пример с применением технологий стилей
2. Реализовать пример с применением технологий тем
3. Реализовать пример создания собственной темы.
4. Реализовать пример с применением темы к activity
5. Реализовать пример с применением темы к иерархии виджетов
6. Реализовать пример с применением определения меню в xml
7. Реализовать пример с наполнением меню элементами
8. Реализовать обработку нажатий в меню.
9. Реализовать программное создание меню
10. Реализовать программное создание подменю.
11. Реализовать группы в меню
12. Реализовать программное создание групп в меню и подменю