Лабораторная работа №4. Стили и темы

Задание: написать приложение, работающее с разными темами/стилями. Сначала создать свой стиль и применить его к какому-нибудь интерфейсному элементу, затем - свою тему, которая применяется ко всем интерфейсным элементам. Приложение при этом должно выглядеть нестандартно. Запустить на эмуляторе и убедиться, что всё работает.

Создайте новый проект с Empty Views Activity, перейдите в activity_main.xml, удалите стандартное текстовое поле с фразой «Hello world!» и поместите друг под друга 4 TextView, а затем поменяйте у них размер шрифта (свойство textSize) на 10, 14, 28 и 36 sp соответственно (не забудьте про ФИО и группу). Как мы видим, все они отличаются друг от друга. Бывают ситуации, когда всем элементам необходимо иметь один стиль (один размер шрифта, один цвет текста и т.д.). В данном случае можно поменять параметры вручную, а что делать, есть у вас 100 таких элементов? В таком случае можно создать стиль и применить его ко всем нужным элементам. Переходим в папку values и создаем там файл styles.xml (правой кнопкой мыши по папке values -> New -> Values resource file, в открывшемся окне File name: styles, и кнопка ОК), в который запишем между тэгами <resources> следующий стиль:

```
<style name="style1" parent="@android:style/TextAppearance">
   <item name="android:textColor">#000000</item>
   <item name="android:textSize">30sp</item>
   <item name="android:typeface">monospace</item>
</style>
```

TextAppearance – это стиль для текста в Android по умолчанию. Далее мы устанавливаем цвет текста чёрным (#000000), делаем текст 30 размера (30sp) и тип шрифта указываем как monospace. Теперь нужно применить этот стиль к созданным четырём элементам. Выделите все четыре TextView в интерфейсе приложения (левой кнопкой мыши с зажатой клавишей Shift) и в панели Attributes (справа) найдите свойство style, где нажмите на крошечной кнопке справа от поля, и в открывшемся окне стилей выберите стиль style1 и нажмите кнопку ОК, или просто в поле style пропишите стиль:

@style/style1

Вы увидите, что текстовые поля, кроме одного (у которого и так было 14sp по умолчанию, поэтому вы не стали ничего вводить в поле textSize), не поменялись в смысле размера. Это происходит потому, что поле textSize непустое и поэтому имеет приоритет над стилем. Если очистить все значения textSize у всех полей, то и все выделенные текстовые поля приведутся к единому виду. Кроме того, в последней версии Android Studio Koala возможно появление ошибки в свойствах (Attributes) элементов TextView – некоторые свойства будут выделены красным цветом (это баг среды), некоторые – оранжевым (это подсказка от среды, что textSize значение непустое, поэтому полностью применить стиль нельзя).

Создадим подстиль, который будет соответствовать предыдущему стилю во всём, кроме размера шрифта. В файле styles.xml ниже стиля style1 напишите:

```
<style name="style1.bigfont">
   <item name="android:textSize">50sp</item>
</style>
```

Применяем данный подстиль к любому текстовому полю и видим, что размер шрифта увеличился. Так же можно менять любые свойства любых интерфейсных элементов, не обязательно TextView, например цвет, шрифт, прозрачность и т.д.

Со стилями разобрались, теперь переходим к созданию собственной темы для приложения. В папке values уже есть папка themes, которая создаётся для каждого проекта автоматически и содержит 2 файла themes.xml – для дневной темы и для ночной темы интерфейса приложения:

```
<resources xmlns:tools="http://schemas.android.com/tools">
   <!-- Base application theme. -->
   <style name="Base.Theme.MyForthApp" parent="Theme.Material3.DayNight.NoActionBar">
      <!-- Customize your light theme here. -->
      <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
   </style>

   <style name="Theme.MyForthApp" parent="Base.Theme.MyForthApp" />
</resources>
```

Мы можем кастомизировать данную тему, о чём, собственно, и написано в комментариях в коде (там даже есть пример, как это делать – с цветом). Изменим родительскую тему приложения (**Theme.Material3.DayNight.NoActionBar**) на следующую:

```
<style name="AppTheme" parent="Base.Theme.AppCompat.Light.Dialog.Alert">
```

Теперь приложение будет иметь вид диалогового окна (это можно увидеть, если вернуться в интерфейс приложения activity_main.xml). Добавим какие-нибудь параметры между тэгами <style>, чтобы тема отличалась от стандартной:

```
<item name="android:background">#5fffb000</item>
<item name="android:textColor">#000000</item>
<item name="android:textSize">32sp</item>
<item name="android:textAllCaps">true</item>
<item name="android:typeface">monospace</item>
```

Свойства довольно понятны из названий, поэтому расписывать их не имеет смысла. Существуют онлайн-ресурсы, позволяющие подобрать какой-то цвет или даже сочетание цветов, например https://colorscheme.ru/, откуда после выбора цвета можно скопировать шестнадцатеричный код его значения, например ffb000. В Android Studio тоже есть такой миниинструмент, если найти свойство textColor интерфейсного элемента (вообще любое свойство со словом color) и кликнуть по цвету, присутствующему в поле, откроется минидиалог выбора цвета. В свойстве **android:background** у цвета в коде выше вместо RGB компонентов есть еще 2 компонента в начале – 5f, это – прозрачность. Удалите из текстовых полей предыдущие стили, и увидите, что текстовые поля сразу подстроились под параметры из темы. Дело в том, что тема AppTheme уже прописана как базовая в приложении, поэтому любые изменения сразу же отображаются на всех элементах. В поле с ФИО можно поставить свойство gravity в значение center_horizontal.

В заключение переходим в файл MainActivity и в объявлении класса оставим следующее:

Java:

```
public class MainActivity extends Activity {
```

Kotlin:

```
class MainActivity : Activity() {
```

Делается это для того, чтобы в приложении не выводилась так называемая «шапка» с названием приложения (ActionBar, или toolbar), которые присутствуют, если наследовать класс MainActivity от класса AppCompatActivity.

Запустите проект на эмуляторе, он должен иметь вид, как показано на рисунке 37. Приложение запустилось без «шапки» и в виде диалогового окна. Из-за того, что прозрачность фона установлена в 5f, а не полную непрозрачность ff, видно, что фон окна и фон текстовых элементов немного различаются.

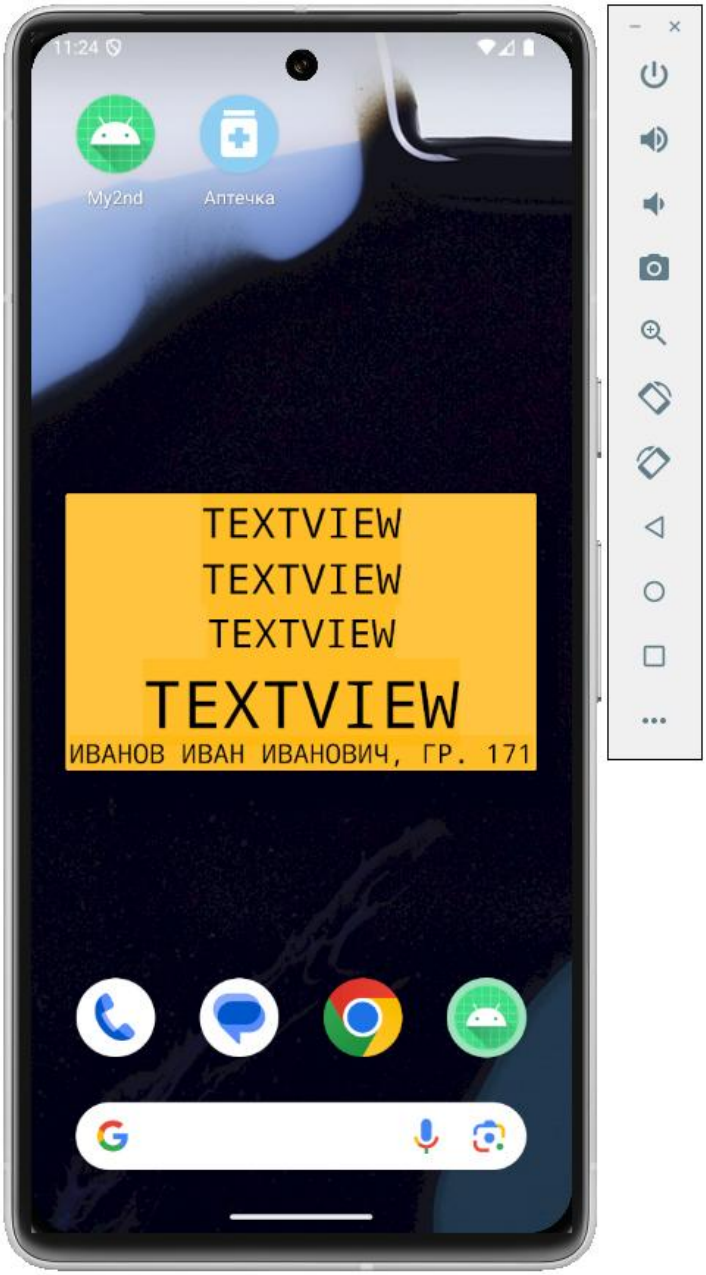

Рисунок 37 – Результат выполнения лабораторной работы №4