**ЛЕКЦІЯ 13. ОФОРМЛЕННЯ ІНТЕРФЕЙСУ ПРОГРАМИ**

**8.1 Стилі**

Ми можемо налаштувати елемент за допомогою різних атрибутів, які задають висоту, ширину, колір фону, тексту і так далі. Але якщо у нас декілька елементів використовують одні і ті ж налаштування, то ми можемо об’єднати ці налаштування в стилі. Наприклад, нехай у нас є кілька елементів *TextView*:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:andr[oid="http://schemas.android.com/apk/res/androi](http://schemas.android.com/apk/res/android)d" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:orientation="vertical">

<TextView android:layout\_width="match\_parent" android:layout\_height="50dp" android:gravity="center" android:textSize="22sp" android:textColor="#3f51b5" android:text="Android Lollipop"/>

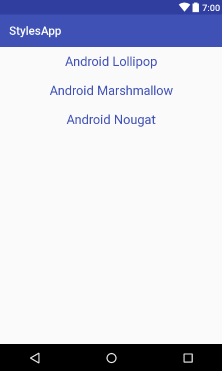
<TextView android:layout\_width="match\_parent" android:layout\_height="50dp" android:gravity="center" android:textSize="22sp" android:textColor="#3f51b5" android:text="Android Marshmallow"/>

<TextView android:layout\_width="match\_parent" android:layout\_height="50dp" android:gravity="center" android:textSize="22sp" android:textColor="#3f51b5"

android:text="Android Nougat"/>

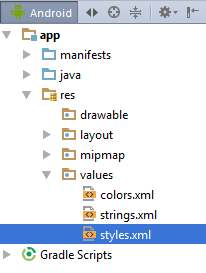
</LinearLayout>

які на екрані користувачу дають наступну розмітку:



Всі ці *TextView* мають однаковий набір властивостей і, наприклад, якщо нам захочеться змінити колір тексту, то доведеться міняти його у всіх трьох *TextView*. Більш оптимальним є підхід з використанням стилів.

За замовчуванням при створенні проекту в Android Studio в папку *res/values* вже додається файл для стилів *styles.xml*:



Якщо такого файлу немає, то його можна додати, або можна додати ще один файл стилів. Головне щоб стилі перебували в каталозі проекту *res/values/* в файлі xml. При цьому файл може мати довільне ім’я, необов’язково *styles.xml*.

Наявний файл *styles.xml* має наступне визначення:

<resources>

<!-- Base application theme. -->

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

<!-- Customize your theme here. -->

<item name="colorPrimary[">@color/](mailto:@color)colorPrimary</item>

<item name="colorPrimaryDark"[>@color/](mailto:@color)colorPrimaryDark</item>

<item name="colorAccent"[>@color/](mailto:@color)colorAccent</item>

</style>

</resources>

Стиль задається за допомогою елемента *<style>*. Атрибут **name** вказує на назву стилю, через яке потім можна посилатися на нього. Необов’язковий атрибут parent встановлює для даного стилю батьківський стиль, від якого дочірній стиль буде наслідувати всі свої характеристики.

За допомогою елементів **item** встановлюються конкретні властивості віджета, який приймає в якості значення атрибута **name** ім’я встановлюваної властивості.

Повернемося до нашого завдання по стилізації елементів *TextView* і для її вирішення змінимо файл *styles.xml*:

<resources>

<!-- Base application theme. -->

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

<!-- Customize your theme here. -->

<item name="colorPrimary[">@color/](mailto:@color)colorPrimary</item>

<item name="colorPrimaryDark"[>@color/](mailto:@color)colorPrimaryDark</item>

<item name="colorAccent"[>@color/](mailto:@color)colorAccent</item>

</style>

<style name="TextViewStyle">

<item name="android:layout\_width">match\_parent</item>

<item name="android:layout\_height">50dp</item>

<item name="android:textColor">#3f51b5</item>

<item name="android:textSize">22sp</item>

<item name="android:gravity">center</item>

</style>

</resources>

Тут визначено новий стиль *TextViewStyle*, який за допомогою елементів **item**

задає значення для атрибутів *TextView*.

Тепер застосуємо стиль, змінимо файл *activity\_main.xml*:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="[http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:orientation="vertical">

<TextView style=["@style/T](mailto:@style)extViewStyle" android:text="Android Lollipop"/>

<TextView style=["@style/T](mailto:@style)extViewStyle" android:text="Android Marshmallow"/>

<TextView style=["@style/T](mailto:@style)extViewStyle" android:text="Android Nougat"/>

</LinearLayout>

Використовуючи визначення style=["@style](mailto:@style)/TextView", текстове поле пов’язується із визначенням стилю. Підсумковий результат буде той же, що і раніше, тільки коду стає менше. А якщо ми захочемо поміняти якісь характеристики, то досить змінити потрібний елемент item у визначенні стилю.

**8.2 Теми**

**8.2.1 Застосування теми**

Крім застосування окремих стилів до окремих елементів, ми можемо задавати стилі для всього програми або *activity* у вигляді тем. Ми можемо самі створити тему. Однак Android вже надає кілька попередньо встановлених тем для стилізації застосунків, наприклад, *Theme.AppCompat.Light.DarkActionBar* і ряд інших.

Для визначення тем застосунку відкриємо файл *AndroidManifest.xml*. У ньому ми можемо побачити наступне визначення елемента *application*, що представляє додаток:

<application

android:allowBackup="true" android:icon=["@mipmap/ic\_l](mailto:@mipmap)auncher" android:label=["@strin](mailto:@string)g/app\_name" android:roundIcon=["@mipmap/ic\_l](mailto:@mipmap)auncher\_round" android:supportsRtl="true" android:theme=["@style/App](mailto:@style)Theme">

Задання теми відбувається за допомогою атрибута **android:theme**. В даному випадку використовується ресурс, визначений в стилях – в файлі *res/values/ styles.xml*:

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

<item name="colorPrimary[">@color/](mailto:@color)colorPrimary</item>

<item name="colorPrimaryDark[">@color/](mailto:@color)colorPrimaryDark</item>

<item name="colorAccent"[>@color](mailto:@color)/colorAccent</item>

</style>

Стиль AppTheme використовує вбудовану тему *Theme.AppCompat.Light.Dark ActionBar*, яка надає візуальні характеристики нашого застосунку. Тепер визначимо стиль, який використовує іншу тему:

<resources>

<style name="NoActionBarTheme" parent="Theme.AppCompat.DayNight.NoActionBar">

</style>

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

<!-- Customize your theme here. -->

<item name="colorPrimary[">@color/](mailto:@color)colorPrimary</item>

<item name="colorPrimaryDark"[>@color/](mailto:@color)colorPrimaryDark</item>

<item name="colorAccent"[>@color/](mailto:@color)colorAccent</item>

</style>

<style name="TextViewStyle">

<item name="android:layout\_width">match\_parent</item>

<item name="android:layout\_height">50dp</item>

<item name="android:textColor">#3f51b5</item>

<item name="android:textSize">22sp</item>

<item name="android:gravity">center</item>

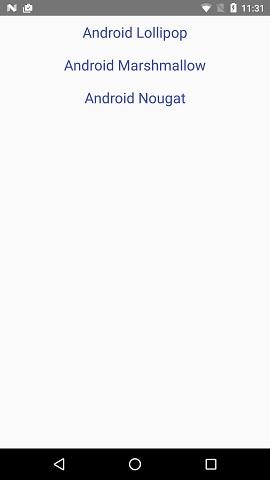
</style>

</resources>

Нехай новий стиль називається NoActionBarTheme, який посилається на тему Theme.AppCompat.DayNight.NoActionBar. Тепер встановимо його в якості теми застосунку в файлі *AndroidManifest.xml*:

<application android:theme=["@styles/NoA](mailto:@styles)ctionBarTheme"

В результаті отримаємо:



**8.2.2 Створення власної теми**

Замість використання вбудованих тем, ми можемо створити свою. Для цього створимо в файлі *res/values/styles.xml* новий стиль:

<resources>

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

<!-- Customize your theme here. -->

<item name="colorPrimary[">@color/](mailto:@color)colorPrimary</item>

<item name="colorPrimaryDark"[>@color/](mailto:@color)colorPrimaryDark</item>

<item name="colorAccent"[>@color/](mailto:@color)colorAccent</item>

</style>

<style name="TextViewStyle">

<item name="android:layout\_width">match\_parent</item>

<item name="android:layout\_height">50dp</item>

<item name="android:gravity">center</item>

</style>

<style name="CustomTheme" parent="Theme.AppCompat.Light">

<item name="android:textColor">#333333</item>

<item name="android:textSize">22sp</item>

</style>

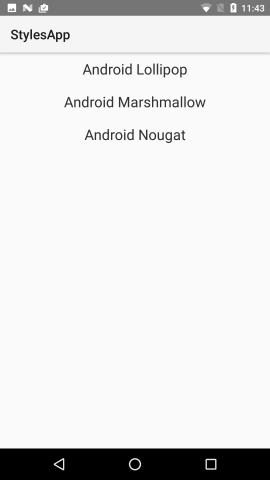
</resources>

Отже, ми створили стиль "*CustomTheme*", який успадкований від стилю *Theme.AppCompat.Light*. У цьому стилі ми перевизначили дві властивості: висоту шрифту (**textSize**) – 22*sp*, а також колір тексту (**textColor**) – він тепер світло-сірий.

Тепер визначимо цей стиль як тему додатка в файлі *AndroidManifest.xml*:

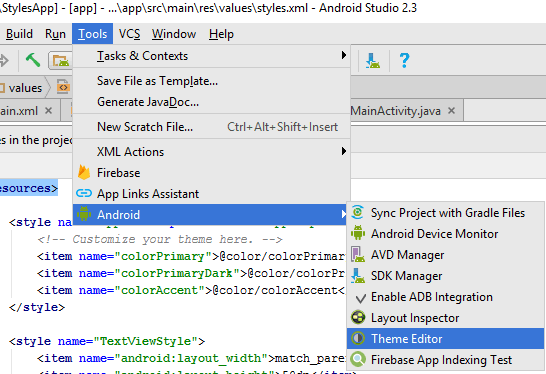
<application android:theme=["@style/Custo](mailto:@style)mTheme"

В результаті отримаємо:



**8.2.3 Редактор тем**

Для спрощення процесу визначення тем в Android Studio є вбудований графічний редактор тем. Для переходу до нього необхідно вибрати пункт меню *Tools->Android->Theme Editor*:



Після цього відкриється редактор тем, де ми зможемо вибрати будь-яку тему і підредагувати її окремі значення, наприклад, колір:

