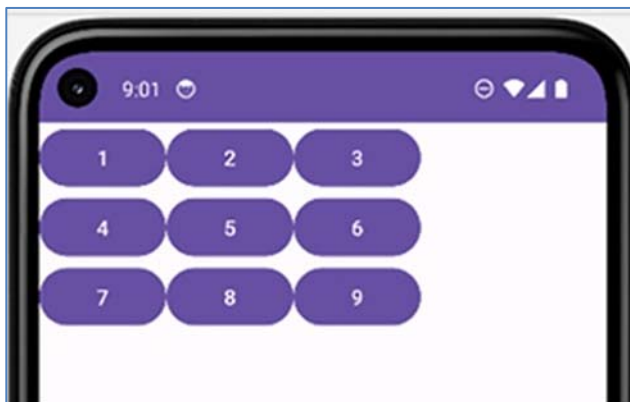


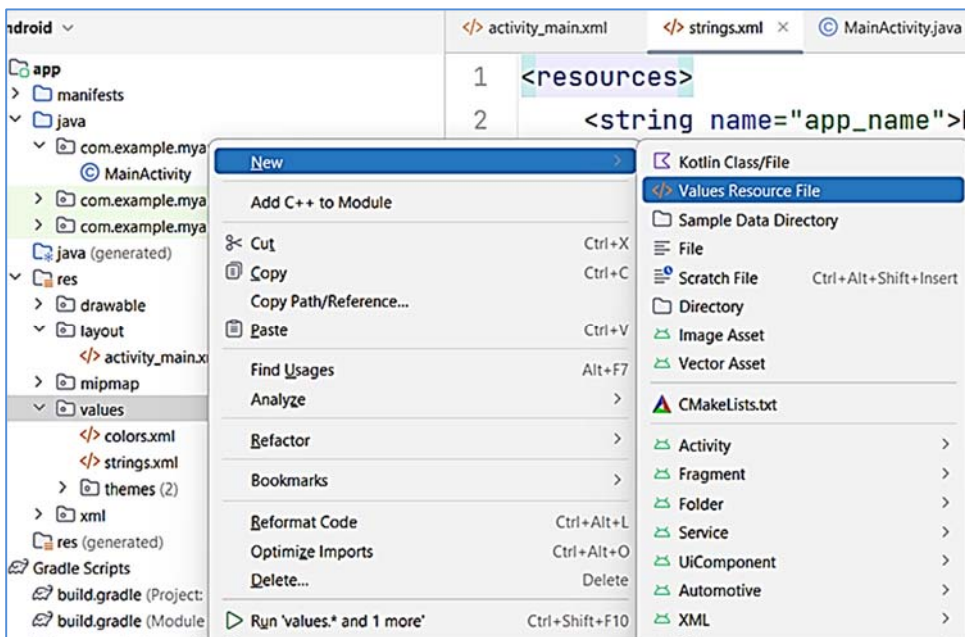
Różnić się będą tylko wartości w parametrach `android:layout_column` i `android:layout_row`.

Aplikacja po uruchomieniu będzie wyglądać tak:



Rysunek 6.78 Aplikacja ze stylem domyślnym

Style można tworzyć w pliku `styles.xml`. Jest to plik zasobów aplikacji i znaleźć go można w drzewie po lewej stronie. Jeżeli plik się tam nie znajduje trzeba go utworzyć samodzielnie. Klikamy prawym przyciskiem myszy na **gałąź values** i z menu, które nam się otwiera wybieramy **New → Values Resource File**.



Rysunek 6.79 Tworzenie nowego pliku zasobów – styles.xml

Nadajemy plikowi nazwę **styles**. Rozszerzenie **xml** zostanie nadane automatycznie. Utworzymy **styl** o nazwie **Szachownica**.

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

<style name="Szachownica">
<item name="android:layout_width">130dp</item>
<item name="android:layout_height">130dp</item>
<item name="android:background">#FFFFFF</item>
<item name="android:textColor">#FF0000</item>
<item name="android:typeface">serif</item>
<item name="android:textSize">55sp</item>
</style>

</resources>
```

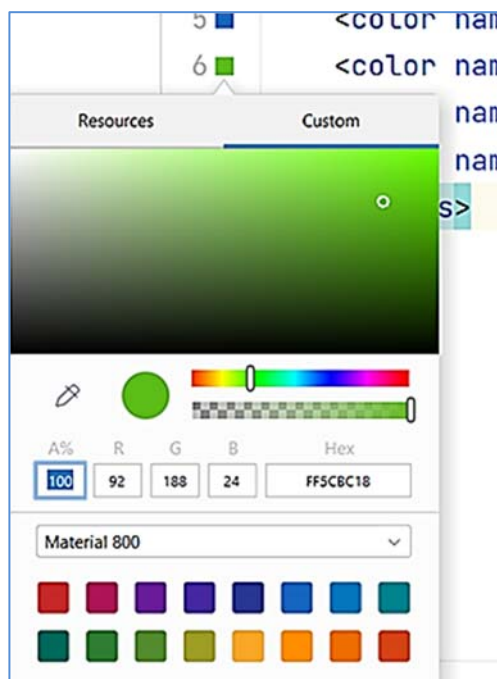
Listing 6.114 Definicja stylu Szachownica

Po pierwsze nadajemy nazwę stylu: **name="Szachownica"**. Kolejne atrybuty informują o **szerokości** i **długości** widoku. Natomiast o kolorze tła zadecyduje atrybut **android:background**. Jest on ustawiony na biały, gdyż chcemy, aby wszystkie przyciski były białe. **Kolor** możemy zmienić poprzez kliknięcie **małego kwadratu** znajdującego się po lewej stronie okna.



Rysunek 6.80 Zawartość pliku colors.xml

Kliknięcie spowoduje wyświetlenie **palety**.



Rysunek 6.81 Okno wyboru koloru

Kolor wybieramy poprzez kliknięcie w **odpowiedni obszar**. Możemy to zrobić również przesuwając **dostępne suwaki**. Pierwszy steruje **odcieniem**, drugi **przezroczystością**. Skorzystać możemy również z przygotowanej listy, a także wpisać wartości **RGB ręcznie**. Dla koloru białego będzie to wartość **255, 255, 255**. Zapis w pliku **styles.xml** jest **zapisem szesnastkowym**.

Kolejne atrybuty to ustawienia koloru tekstu:

```
<item name="android:textColor">#FF0000</item>
```

Listing 6.115 Ustawienie koloru czcionki

Określa rodzinę czcionek:

```
<item name="android:typeface">serif</item>
```

Listing 6.116 Ustawienie rodziny czcionek

Określa wielkość czcionki:

```
<item name="android:textSize">55sp</item>
```

Listing 6.117 Ustawienie rozmiaru czcionki

W pliku układu możemy teraz przypisać style do odpowiednich kwadratów. Zrobimy to za pomocą poniższego kodu:

```
style="@style/NazwaStylu"
```

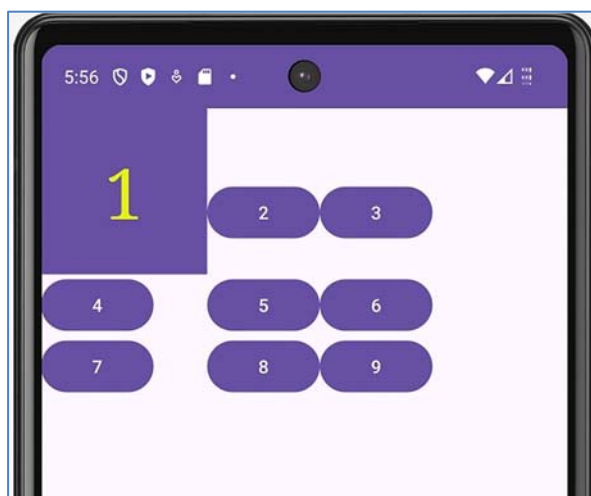
Listing 6.118 Przypisanie stylu do widoku.

w miejscu **NazwaStylu** wpisujemy nazwę **Szachownica**. Zmienimy ustawienie stylu dla pierwszego przycisku. Kod przycisku będzie wyglądał następująco:

```
<Button
  style="@style/Szachownica"
  android:text="1"
  android:layout_column="0"
  android:layout_row="0"
/>
```

Listing 6.119 Kod przycisku Button z zadeklarowanym stylem

Aplikacja po uruchomieniu powinna przypominać poniższy rysunek:



Rysunek 6.82 Przycisk 1 po zastosowaniu stylu

Na rysunku widać, że przycisk zmienił **rozmiar**, a także **kształt**. Zmienił się również **rozmiar i kolor czcionki**. Nie zmienił się jednak **kolor przycisku**. Aplikacja działa w domyślnym stylu, który jest zapisany w pliku **Manifest.xml**. Dlatego musimy zmienić to ustawienie. Na początek jednak musimy ustalić swój motyw aplikacji. W pliku **styles.xml** umieszczamy następujący kod:

```
<style name="NowyTemat"
  parent="Theme.AppCompat.Light.NoActionBar">
</style>
```

Listing 6.120 Stworzenie nowego stylu aplikacji

Tworzymy nowy styl o nazwie **NowyTemat**. W atrybucie **parent** wpisujemy nazwę stylu **nadrzędnego** – **rodzica**, który nieco zmieni wygląd aplikacji.

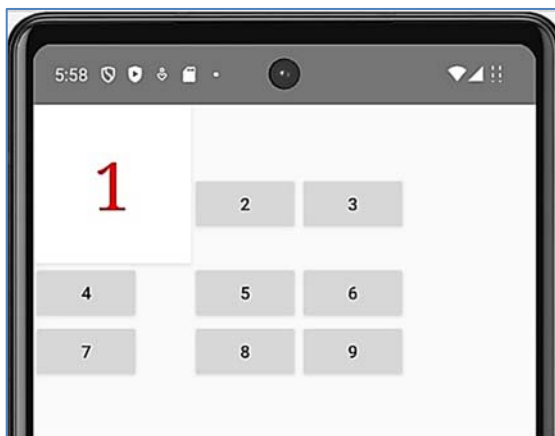
Ustawienia motywu będą szerzej omówione w następnym rozdziale. W tym momencie wystarczą nam tylko te dwie powyższe linijki. Następnie przechodzimy do pliku **manifestu** i zmieniamy tam jego nazwę.

```
<application
    android:allowBackup="true"

    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/NowyTemat"
    tools:targetApi="31">
```

Listing 6.121 Fragment pliku manifestu

W miejscu **android: theme** zmieniamy domyślną nazwę na nazwę naszego stylu czyli: **NowyTemat**. Po uruchomieniu aplikacji nasz przycisk zmieni kolor na właściwy.



Rysunek 6.83 Aplikacja po zastosowaniu własnego tematu aplikacji

Zauważyć można również, że zmianę pozostałych przycisków, a także, zmianę koloru paska. Wynika to z faktu, że zmieniliśmy cały temat kolorystyczny aplikacji.

6.10.2 Dziedziczenie stylów

Kod poniższej aplikacji znajdziemy w folderze **Aplikacja24a**.

Podczas tworzenia większych i bardziej skomplikowanych aplikacji wykorzystanie pojedynczej grupy styli może być niewystarczające. Często

potrzebne jest wiele tego typu grup, które często różnią się tylko pewnymi elementami. Wówczas możemy wykorzystać część atrybutów z jednego stylu.

Jako przykład zmodyfikujemy poprzednią aplikację. Na początek przypiszmy styl **Szachownica** do wszystkich widoków w układzie. Aplikacja będzie wyglądać tak jak na poniższym rysunku.



Rysunek 6.84 Aplikacja z zastosowaniem własnego stylu

Wszystkie pola są **koloru białego**. Teraz musimy znaleźć sposób jak najprościej i najszybciej dokonać zmian w kodzie. Wykorzystamy w tym celu **dziedziczenie stylów**. Dziedziczenie jak wspomniano na początku umożliwia **zmianę** tylko **wybranych parametrów**. Resztę zostawimy bez zmian.

Tworzymy styl o nazwie **Szachownica.Czarne**. Nazwa zawiera informacje o tym, że głównym stylem jest styl o nazwie **Szachownica**, a **kropka** informacje, że styl **Czarne** po nim dziedziczy. W stylu **Szachownica.Czarne** wpisujemy informacje o kolorze tła jakie ma mieć widok. Kod pliku **styles.xml** będzie wyglądał tak jak poniżej:

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

    <style name="NowyTemat"
parent="Theme.AppCompat.Light.NoActionBar">
    </style>

    <style name="Szachownica">
<item name="android:layout_width">130dp</item>
<item name="android:layout_height">130dp</item>
<item name="android:background">#FFFFFF</item>
<item name="android:textColor">#FF0000</item>
<item name="android:typeface">serif</item>
<item name="android:textSize">55sp</item>
    </style>

    <style name="Szachownica.Czarne">
<item name="android:background">#000000</item>
    </style>
</resources>
```

Listing 6.122 Definicja odziedziczonego stylu o nazwie Szachownica.Czarne

W pliku układu trzeba dopisać informację o **nowym stylu** dla **widoków**, które mają mieć **czarny kolor**. Kod pliku układu będzie miał po zmianach następującą postać:

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        style="@style/Szachownica"
        android:text="1"
        android:layout_column="0"
        android:layout_row="0"
    />

    <Button
        style="@style/Szachownica.Czarne"
        android:layout_column="1"
        android:layout_row="0"
        android:text="2" />
```

```
<Button
    style="@style/Szachownica"
    android:text="3"
    android:layout_column="2"
    android:layout_row="0"
    />
<Button
    style="@style/Szachownica.Czarne"
    android:layout_column="0"
    android:layout_row="1"
    android:text="4" />
<Button
    style="@style/Szachownica"
    android:text="5"
    android:layout_column="1"
    android:layout_row="1"
    />
<Button
    style="@style/Szachownica.Czarne"
    android:layout_column="2"
    android:layout_row="1"
    android:text="6" />
<Button
    style="@style/Szachownica"
    android:text="7"
    android:layout_column="0"
    android:layout_row="2"
    />
<Button
    style="@style/Szachownica.Czarne"
    android:text="8"
    android:layout_column="1"
    android:layout_row="2"
    />
<Button
    style="@style/Szachownica"
    android:text="9"
    android:layout_column="2"
    android:layout_row="2"
    />
</GridLayout>
```

Listing 6.123 Kod układu z przypisanymi stylami

Aplikacja będzie wyglądać tak jak na poniższym rysunku:



Rysunek 6.85 Aplikacja wykorzystująca dziedziczenie stylu

6.11 Tematy, motywy.

Tematy lub motywy omawiane już były w poprzednim rozdziale. Zagadnienie to jest nieco szersze i przybliżymy je w tym rozdziale.

Temat lub **motyw** jest **stylem** stosowanym w **aplikacji**. Temat podobnie jak styl można zdefiniować i wykorzystać w całym programie. Style wykorzystuje się do części aplikacji np. **widoku**. Temat stosuje się do całej aplikacji. Są to ustawienia wyglądu aktywności od strony graficznej: **kolory**, **rodzaje czcionek**, a także **kolory pasków w oknie**.

6.11.1 Korzystanie z wbudowanych tematów

Tworzymy nową aplikację z plikiem układu zawierającym pole **tekstowe** i **przycisk**.

```
<?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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="32dp"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.125"
        android:textSize="25pt"/>

    <Button
        android:layout_width="260dp"
        android:layout_height="97dp"
        android:layout_marginTop="150dp"
        android:text="Przycisk"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.071" />

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

Listing 6.124 Plik układu aplikacji

Następnie dodajemy plik **styles.xml**. Potrzebny również będzie plik **manifestu**. W pliku **styles.xml** tworzymy nowy **motyw**. Nadajmy mu nazwę **NowyTemat2** i tą właśnie nazwę wpiszemy w pliku manifestu. Plik **styles.xml** powinien wyglądać teraz tak:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="NowyTemat2" parent="Theme.AppCompat">
        </style>
</resources>
```

Listing 6.125 Utworzenie nowego stylu

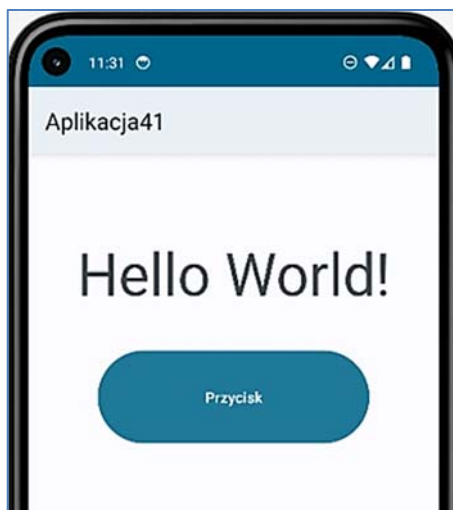
W atrybucie **parent** możemy wybrać jeden z wielu dostępnych motywów aplikacji. Najlepszym wyborem będą **tematy** z kolekcji **Material3**. Są one bardzo nowoczesne i oparte są na technologii **Material Design wersji 3**. Poniżej znajdują się przykłady aplikacji wykorzystujące różne wbudowane motywy.

Theme.AppCompat:



Rysunek 6.86 Aplikacja po zastosowaniu motywu "Theme.AppCompat"

Theme.Material3.DynamicColors.Light:



Rysunek 6.87 Aplikacja po zastosowaniu motywu "Theme.Material3.DynamicColors.Light"

Theme.Material3.Dark:



Rysunek 6.88 Aplikacja po zastosowaniu motywu "Theme.Material3.Dark"

6.11.2 Tworzenie własnego motywu

Kod aplikacji znajduje się w folderze o nazwie **Aplikacja25**.

Tworzenie własnego **motywu** trzeba zacząć od ustawienia dowolnego z domyślnych **tematów**. Robimy to w pliku **styles.xml**. Pamiętaj również, żeby w pliku **manifestu** dodać **główną nazwę stylu**.

Wszystkie kolory, które będziemy używać w aplikacji należy wcześniej zadeklarować i ustawić w pliku **colors.xml**. Znajduje się on w drzewie aplikacji w gałęzi **Values**. W pliku można dopisać wartości do tych, które już tam się znajdują. Zajmiemy się teraz edycją tego pliku. Utworzymy wyraziste, kolory które silnie będą akcentować elementy aplikacji. Dopisujemy następujące kolory:

```
<color name="zielony">#7CB342</color>
<color name="czerwony">#D50000</color>
<color name="zolty">#FFD600</color>
<color name="jasnyniebieski">#55B7F1</color>
<color name="ciemnoniebieski">#283593</color>
<color name="różowy">#D1119C</color>
```

Listing 6.126 Plik colors.xml

Następnie w pliku **styles.xml** zmieniamy ustawienia kolorów elementów występujących w aplikacji. Kod w pliku **styles.xml** wygląda tak jak poniżej: