



# PROGRAMOWANIE APLIKACJI MOBILNYCH

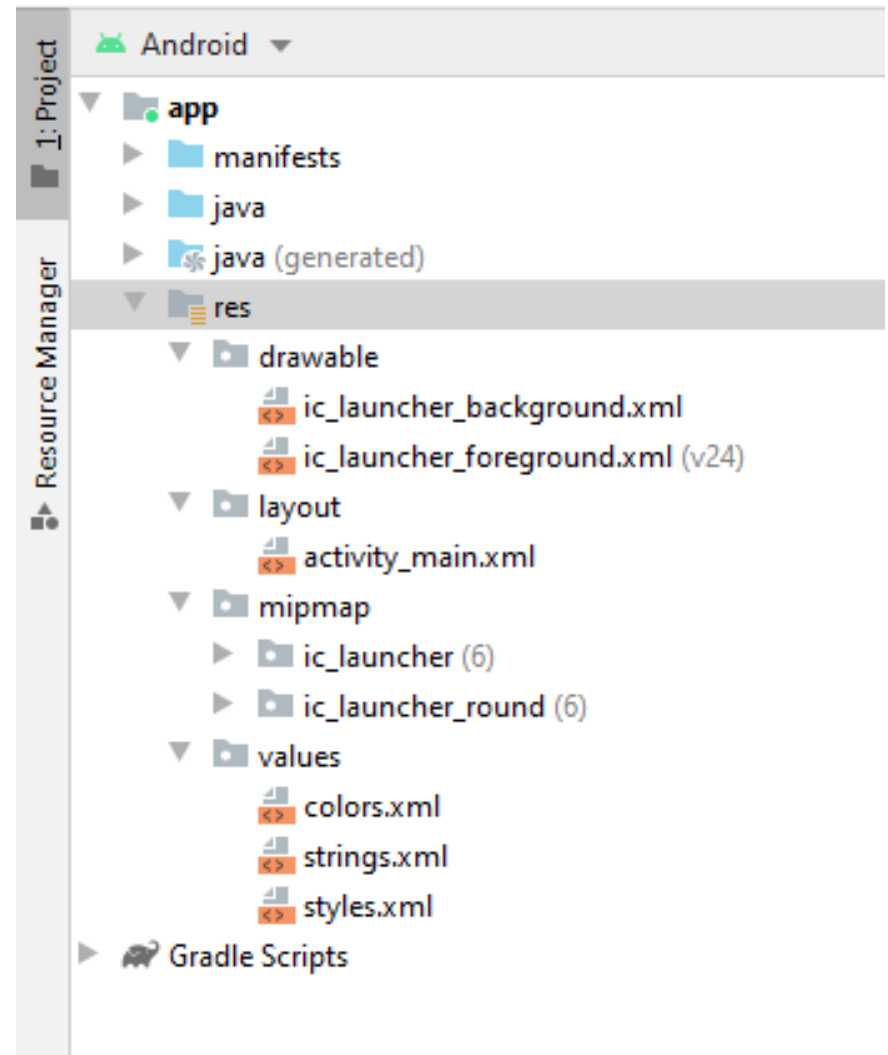
## Wykład

dr Artur Bartoszewski



## Zasoby aplikacji

Aby w projekcie skorzystać z zasobów, należy umieścić je podkatalogach folderu **res**.



- ✓ **Drawable** - lokalne obrazy (pliki JPG i PNG,) należy umieścić w folderze **drawable**. Tam również umieszcza się pliki XML, definiujące obiekty wektorowe i kształty.
- ✓ **Mipmap** - katalog mipmap akceptuje elementy bitmapowe, ale w szczególności przechowuje ikonę aplikacji.
- ✓ **Layout** - katalog layout zawiera pliki XML, które definiują budowę interfejsu aplikacji.



- ✓ **Color** - katalog color jest używany do przechowywania plików XML, które kojarzą informacje o stanie z określonymi kolorami. To jest przydatne, np. gdy musisz zmienić kolor tekstu View lub tła po stuknięciu, aby dostarczyć informację zwrotną użytkownikowi. (kolory można zapisać też w katalogu values).
- ✓ **Menu** - pliki XML menu są używane do definiowania paska akcji, menu nawigacji i submenu. Te zasoby znajdują się w katalogu zasobów „menu”.

- ✓ **Raw** - katalog raw przechowuje pliki surowe dla aplikacji (pliki audio, wideo i tekstowe). To umożliwia łatwy dostęp do plików, jednakże jeśli chcesz uzyskać dostęp do oryginalnej nazwy pliku lub hierarchii katalogów, powinieneś rozważyć umieszczenie plików surowych w katalogu asset Androida.
- ✓ **Xml** - ten katalog przechowuje dowolne pliki XML używane przez Androida.

**Katalog values** - może zawierać wiele plików XML składających się z kluczy wartości, które są używane w aplikacji.

- **Arrays** - obiekty tablic,
- **Colors** – kolory (z własnymi nazwami), które mogą być użyte w wielu miejscach w twojej aplikacji.
- **Dimens** – rozmiary - mogą określać cokolwiek związanego z rozmiarem (np.: tekst, marginesy).
- **Integers** - jeśli istnieją konkretne liczby całkowite (stałe), które chcesz wykorzystać w swojej aplikacji, można zebrać je w pliku zasobu

- **Strings** -. zamiast rozpraszać łańcuchy znaków (teksty) w kodzie, można przechowywać je w pliku strings.xml.
- **Plurals** - podobne do ciągów, plurals umożliwiają dostarczanie alternatywnych ciągów, gdy liczba jest przekazywana do funkcji wydobywania.
- **Styles** – definiując style (plik styles.xml) można przygotować domyślny wygląd komponentów View – style wykorzystujemy budując układy.



## Dostęp do zasobów

Zasoby przechowywane w aplikacji posiadają identyfikowalną nazwę składającą się z typu zasobu (takiego jak np.: **drawable** lub **layout**) oraz **id** (w generowanym automatycznie pliku **R**)



## Dostęp do zasobów – z poziomu Java

Wiele obiektów w systemie Android akceptuje **id** zasobu w celu jego wykorzystania.

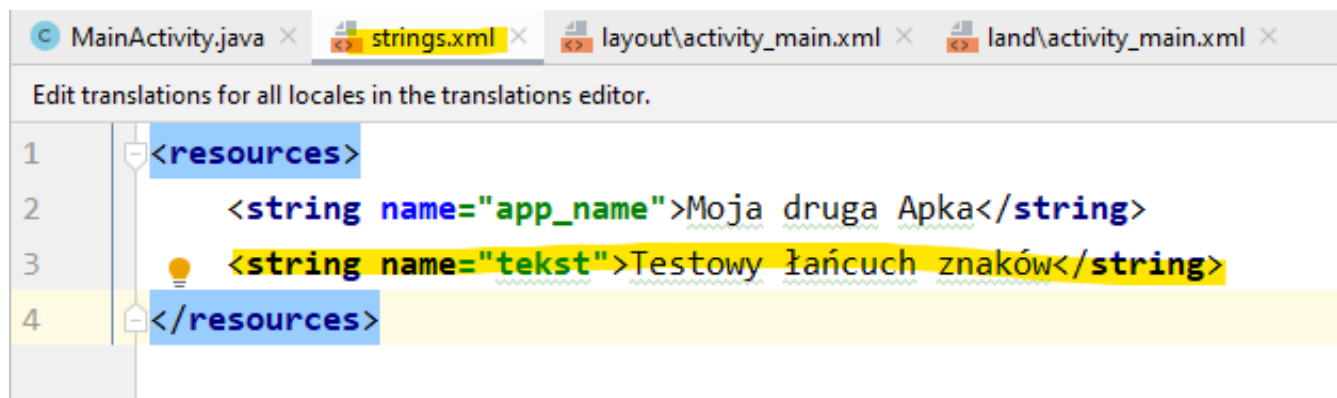
Np.: ustawienie źródła obrazu dla ImageView:

```
ImageView.setImageResource(R.drawable.nazwa) ;
```

Jeżeli trzeba uzyskać oryginalny zasób w kodzie. Można to zrobić za pomocą metody getResources( )

```
getResources().getColor(R.color.kolor) ;
```

## Dostęp do zasobów – strings.xml



```
1 <resources>
2     <string name="app_name">Moja druga Apka</string>
3     <string name="tekst">Testowy łańcuch znaków</string>
4 </resources>
```

```
String s = getResources().getString(R.string.tekst);
t1.setText(s);
```

## Dostęp do zasobów z pliku „strings.xml” – tablica łańcuchów

MainActivity.java x strings.xml x layout\activity\_ma

t translations for all locales in the translations editor.

```
<resources>
  <string-array name="tab">
    <item>"pozycja 1</item>
    <item>"pozycja 2</item>
    <item>"pozycja 3</item>
    <item>"pozycja 4</item>
  </string-array>
```

```
</resources>
```

```
String[] tablica = getResources().getStringArray(R.array.tab);
t1.setText(tablica[0]);
```

```
String[] tablica = getResources().getStringArray(R.array.tab);
String s="";
for (String temp: tablica)
{
  s=s+temp+" ";
}
t1.setText(s);
```



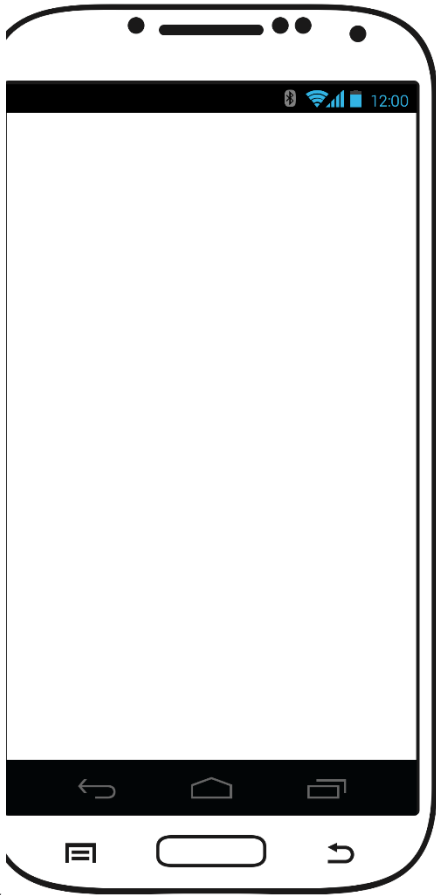
### Alternatywne wersje zasobów - rozdzielczości

Dołączając kwalifikatory takie jak: **-ldpi**, **-mdpi**, **-hdpi** do nazw folderów, (jak widać w tworzonym automatycznie folderze minimaps) możemy przygotować zestawy grafik dla różnych klas urządzeń

# Aplikacije mobilne

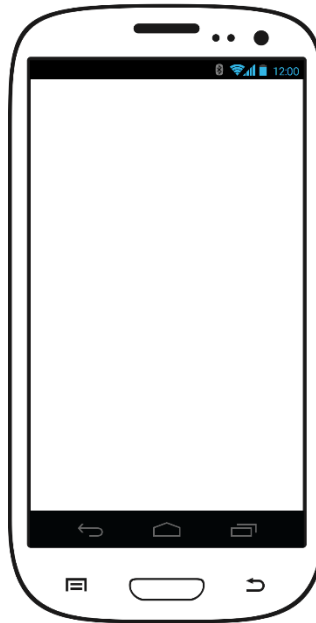


XLHDPi  
3.0  
1dp = 3px



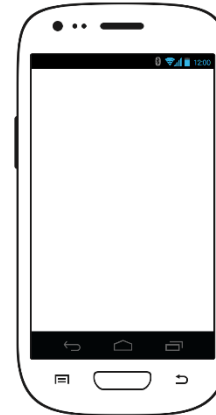
GALAXY S4

XHDPi  
2.0  
1dp = 2px



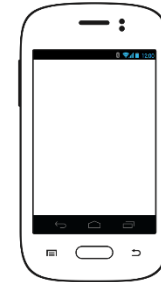
GALAXY S3

HDPi  
1.5  
1dp = 1.5px



GALAXY S3 MINI

MDPi  
1.0  
1dp = 1px



GALAXY YOUNG

LDPi  
0.75  
1dp = 0.75px

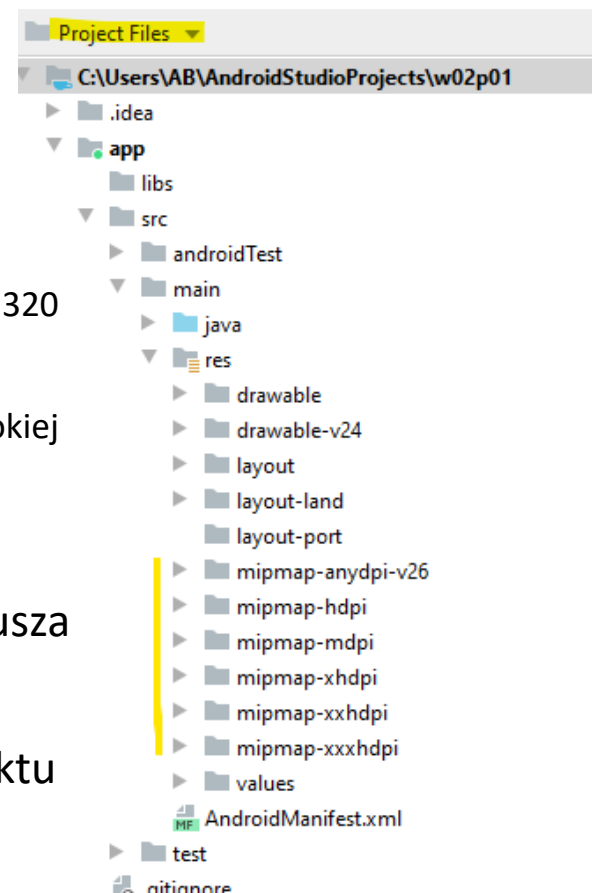


GALAXY POCKET

## Alternatywne wersje zasobów – rozdzielczości plików graficznych

- **drawable** pliki graficzne wspólne dla wszystkich rozdzielczości,
- **drawable-ldpi** - pliki dla ekranów o niskiej rozdzielczości - 120 dpi,
- **drawable-mdpi** - pliki dla ekranów o średniej rozdzielczości - 160 dpi,
- **drawable-hdpi** - pliki dla ekranów o wysokiej rozdzielczości - 240 dpi,
- **drawable-xhdpi** - pliki dla ekranów o ekstra wysokiej rozdzielczości - 320 dpi,
- **drawable-xxhdpi** - pliki dla ekranów o super ekstra wysokiej rozdzielczości - 480 dpi

1. Tworzymy wersje pliku graficznego, o różnych rozdzielczościach. UWAGA: wszystkie wersje pliku muszą mieć identyczne nazwy.
2. Umieszczamy pliki w odpowiednich katalogach projektu





### Alternatywne wersje zasobów - języki

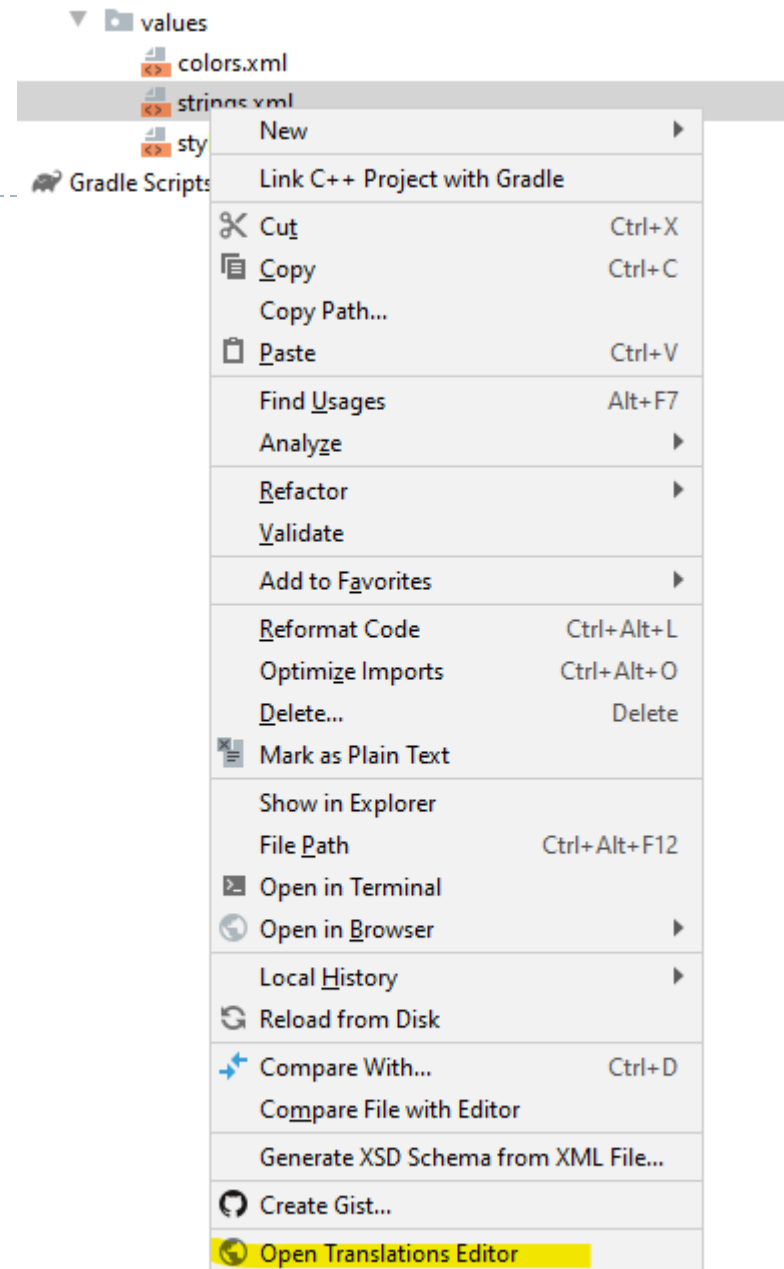
Można stworzyć osobne katalogi wartości dla każdej wersji językowej aplikacji - dołączając kod języka na końcu nazwy katalogu (na przykład **values-de** dla niemieckiego lub **values-pl**)

# Aplikacje mobilne

## Alternatywne wersje zasobów - języki

Mając bazową wersję pliku string.xml możemy dodać jego wersje językowe (tłumaczenia)

1. Znajdujemy w zasobach plik strings.xml i klikamy prawym przyciskiem myszy
2. Z menu kontekstowego wybieramy „Open Translation Editor”

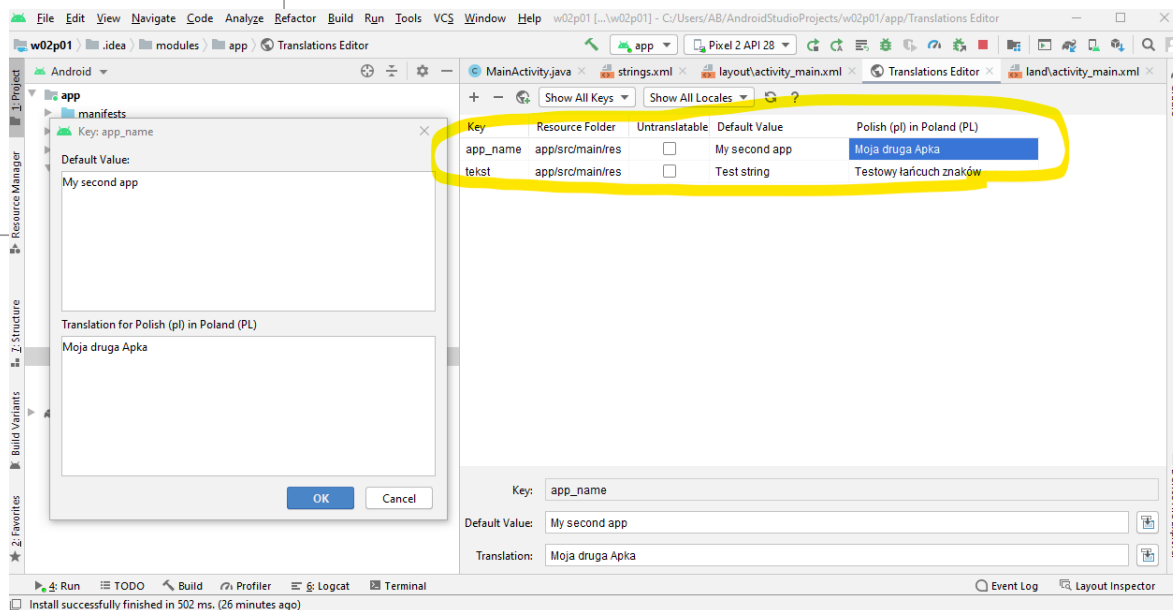
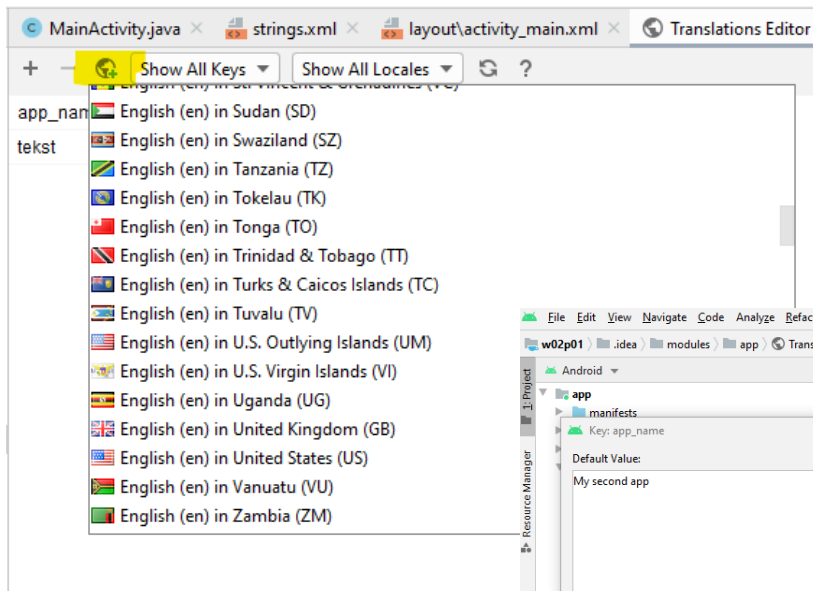




## Alternatywne wersje zasobów - języki

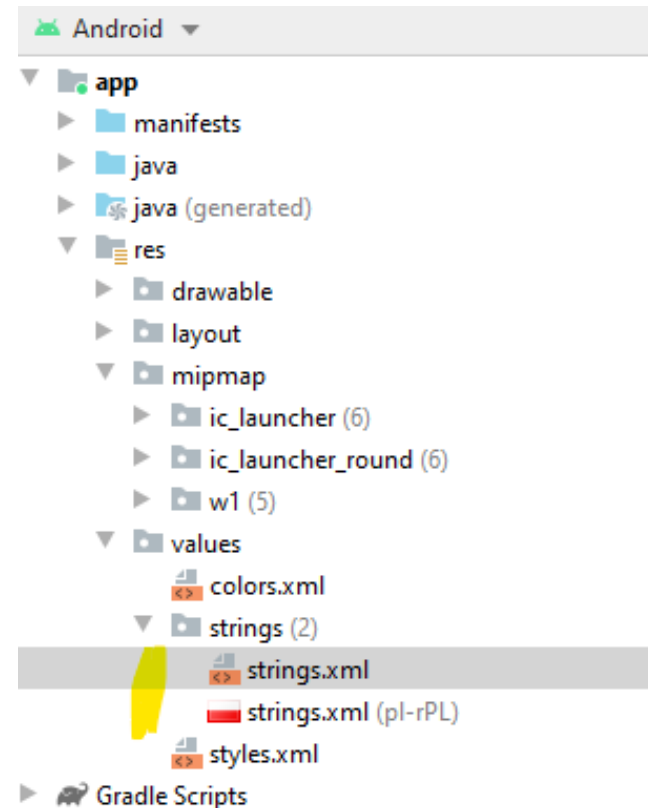
3. W edytorze wybieramy język tłumaczenia

4. Otrzymujemy listę łańcuchów zawartych w pliku – dodajemy do nich tłumaczenia



## Alternatywne wersje zasobów - języki

W widoku projektu otrzymujemy nowy plik strings.xml oznaczony nazwą języka.



## Alternatywne wersje zasobów – orientacja ekranu

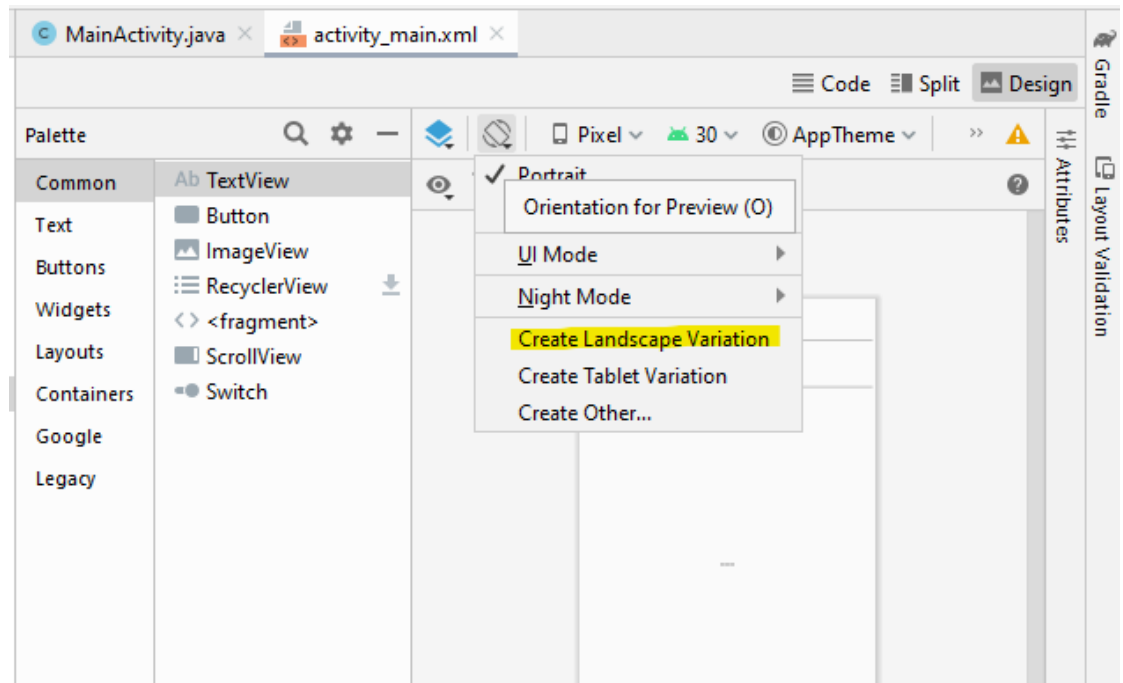
Można dodać kwantyfikikator **-land** do katalogu zasobu, aby zdefiniować domyślne wartości, dla pozycji poziomej.

Można zdefiniować również minimalną szerokość urządzenia (-sw600dp lub -sw720dp)

## Alternatywne wersje layoutów – orientacja ekranu

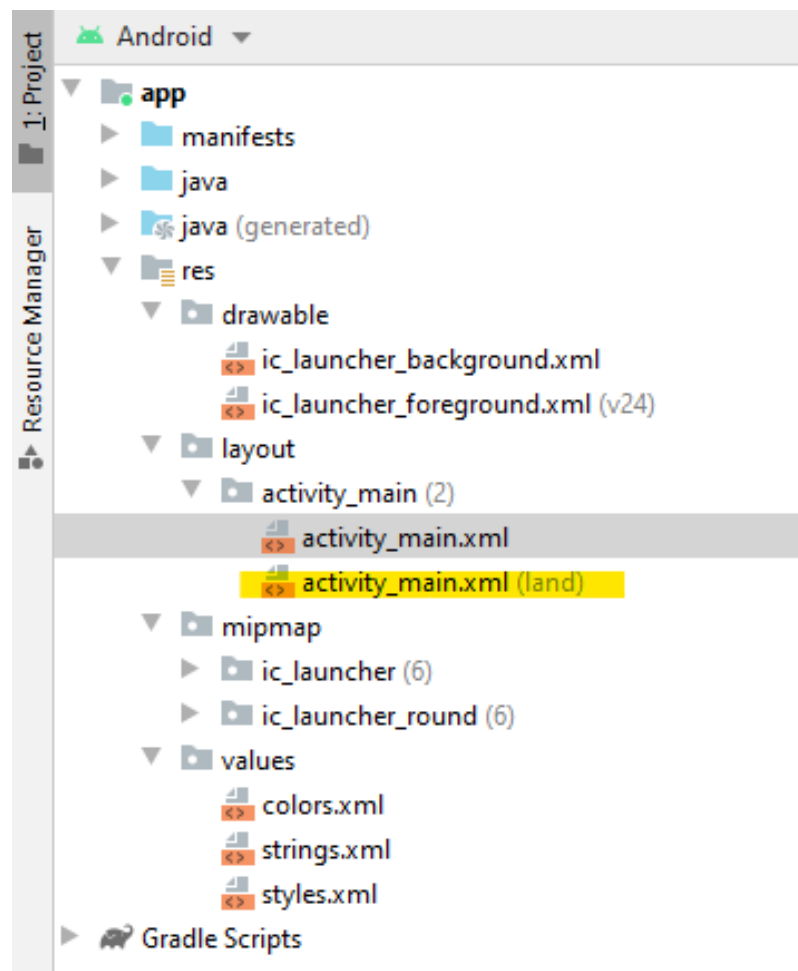
Aby dodać poziomy layout należy:

- otworzyć edytor z domyślnym layoutem,
- w prawym górnym rogu kliknąć ikonę telefonu,
- z rozwiniętego menu wybrać opcję „*Create Landscape Variation*„



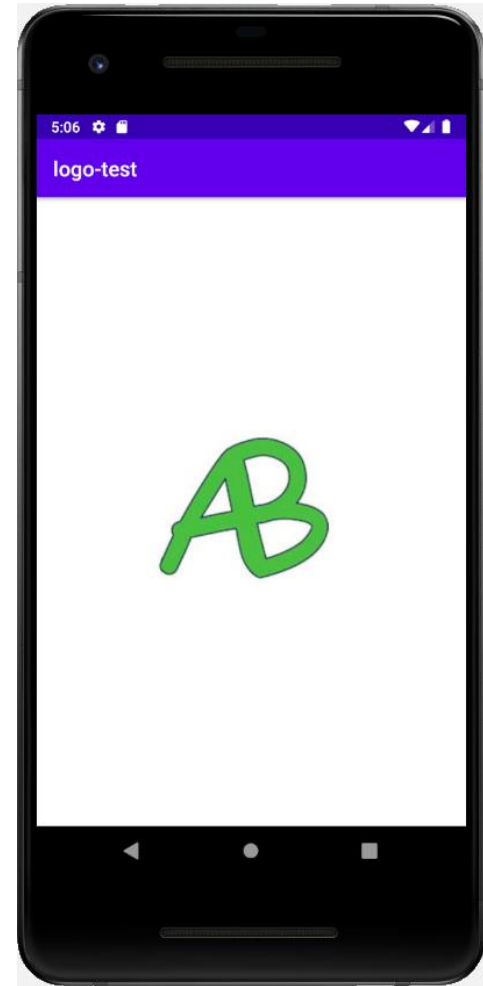
## Alternatywne wersje layoutów – orientacja ekranu

Utworzone zostaną dwa pliki `activity_main.xml` z tym, że jeden z nich oznaczony jest jako widok poziomy (ang. *land*).



# ZADANIE PRAKTYCZNE:

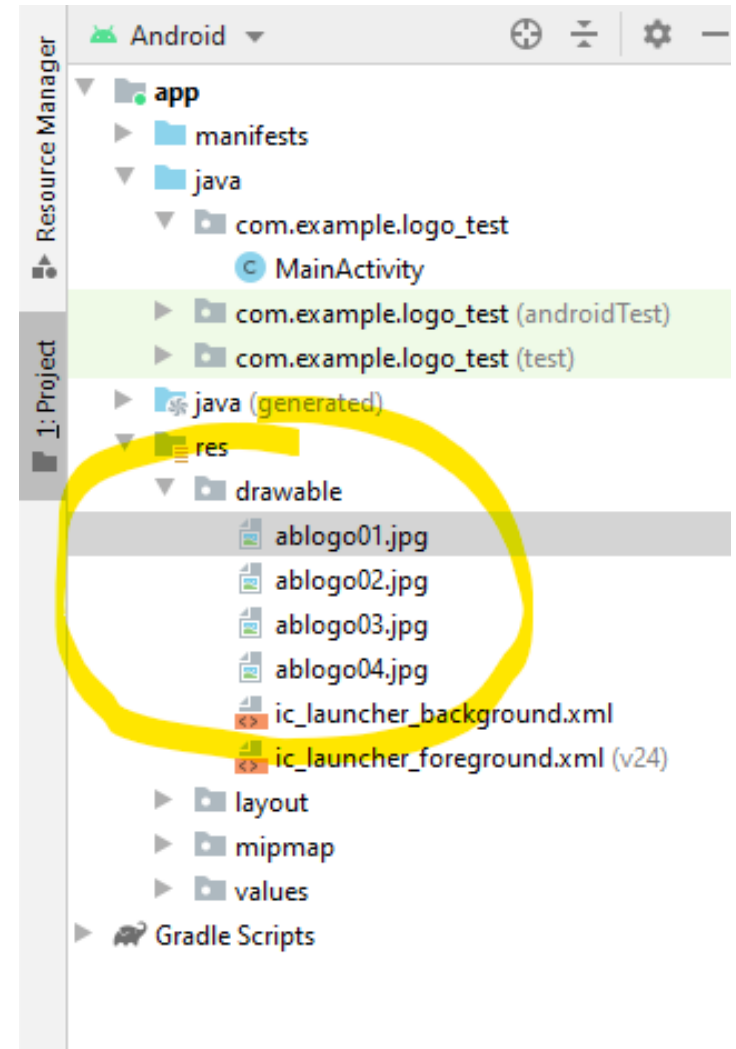
1. Aplikacja wyświetla grafikę.
2. Po kliknięciu na nią, grafika zmienia się.
3. W zasobach aplikacji są 4 wersje kolorystyczne obrazka. Po kliknięciu przeskakujemy na następną. Po czwartym kliknięciu wracamy do pierwszej wersji itd..



# Przykład

## I. Dodawanie zasobów graficznych

Pliki graficzne umieszczamy w katalogu „drawable” projektu (najprościej metodą przeciągnij i upuść)



# Przykład

## I. Dodawanie zasobów graficznych

Dostęp do plików  
graficznych z poziomu  
Java

```
<ImageView
    android:id="@+id/imageView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/abLogo01"
    app:layout_c...
    app:layout_c...
    app:layout_c...
    app:layout_c...
    android:click...
    android:onClick...
/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Press Enter to insert, Tab to replace



# Przykład

## I. Tworzenie layout-u

Do wyświetlania  
grafiki służy  
komponent  
ImageView

```
<ImageView  
    android:id="@+id/imageViw01"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@drawable/abLogo01"  
    app:layout_constraintTop_toTopOf="parent"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintRight_toRightOf="parent"  
    android:clickable="true"  
    android:onClick="zmien"  
/>
```

Aby reagował na kliknięci  
ustawiamy opcje „clickable” na  
true oraz podpinamy metodę  
obsługi zdarzenia „onClick

# Przykład

## I. Kod Java

```
11     ImageView ramka;  
12     int[] obrazki = new int[] { R.drawable.abLogo01,  
13                                     R.drawable.abLogo02,  
14                                     R.drawable.abLogo03,  
15                                     R.drawable.abLogo04};  
16     int numerObrazka = 0;
```

Przygotowujemy:

- referencję do komponentu ImageView
- tablicę zawierającą indeksy obrazków (indeksy są zmiennymi typu int)
- zmienną przechowującą numer wyświetlanego obrazka

# Przykład

## I. Kod Java

```
22      ramka = findViewById(R.id.imageView01);
```

Odnajdujemy uchwyt do ImageView

```
25      public void zmien(View view) {  
26          numerObrazka++;  
27          if (numerObrazka >= 4) numerObrazka = 0;  
28          ramka.setImageResource(obrazki[numerObrazka]);  
29      }
```

W metodzie „zmien” (podpiętej pod onClick) zwiększy numer obrazka i ustawiamy nowy obrazek dla ImageView

# ZADANIE PRAKTYCZNE:

Program wczytuje dane i po kliknięciu przyciski „Zapisz” składa je w jeden tekst i wyświetla poniżej.

