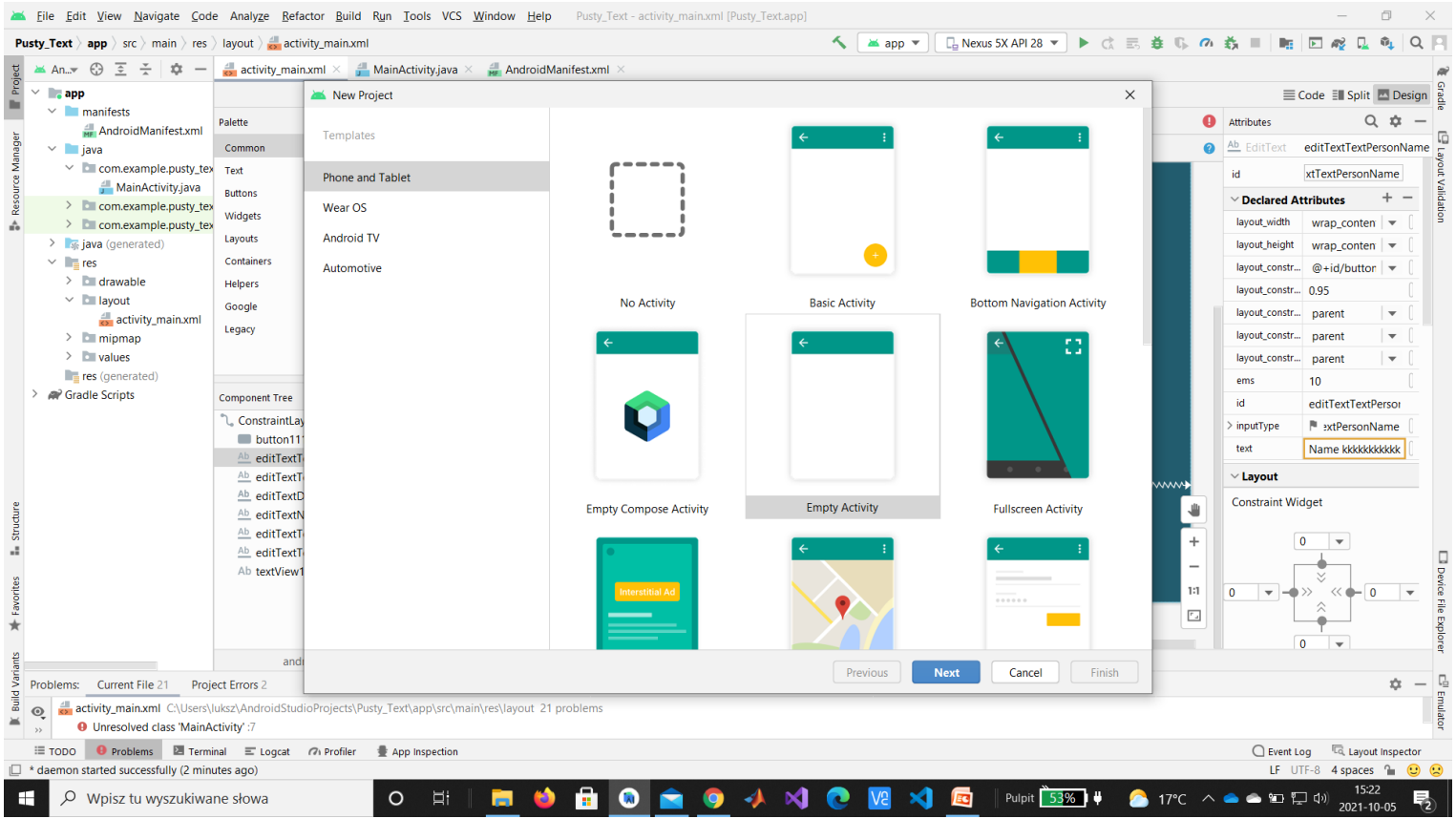
Aplikacja Android

PUM

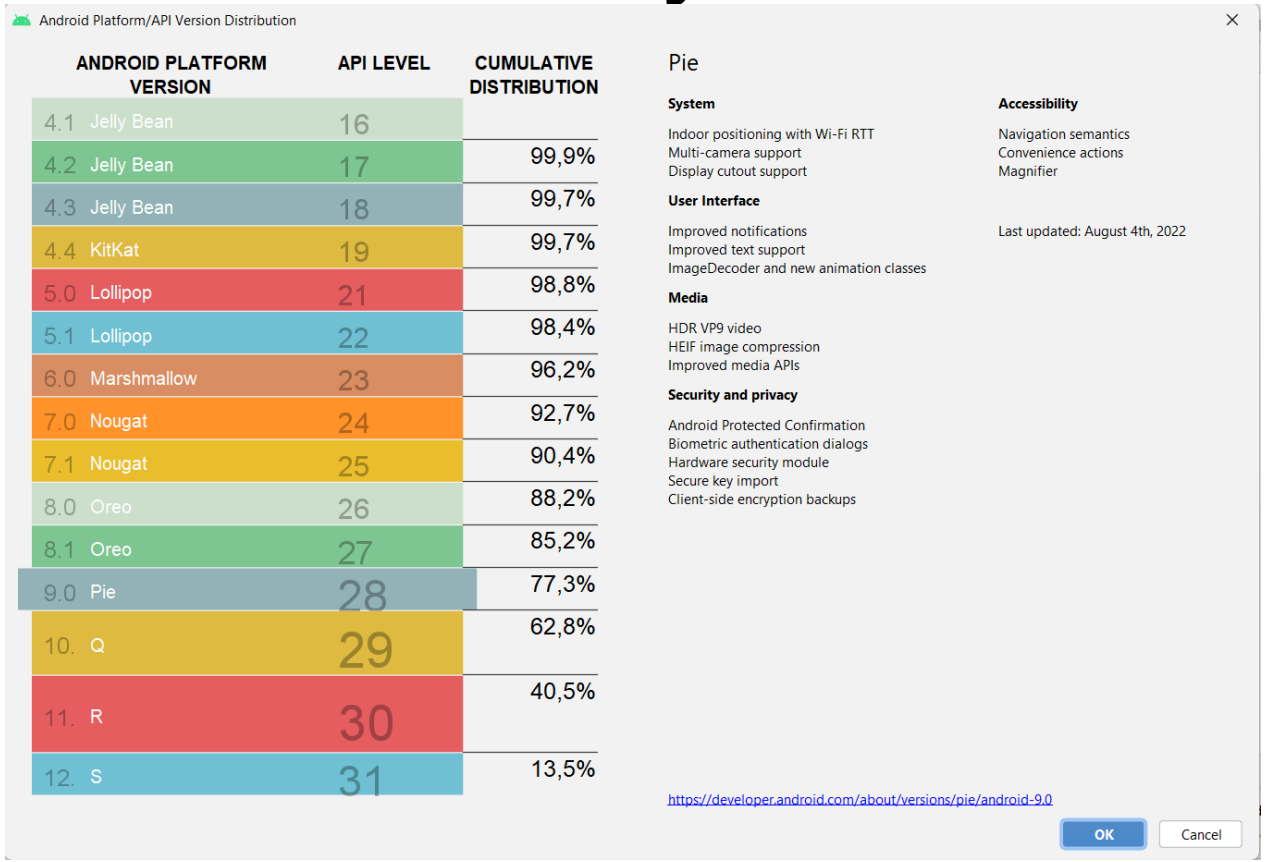
Android Studio

- Oficjalne środowisko programistyczne na platformę Android. Android to około 90% rynku urządzeń mobilnych, system iOS to około 10% 20%.
- Wolne oprogramowanie, na licencji Apache License, autorstwa firmy Google i JetBrains.
- Języki programowania Java (8, 12?), Kotlin (preferowany) i C++.
- Edytor typu WYSIWIG do edycji układu ekranu.
- Pierwsze wydanie grudzień 2014r. Poprzednik to środowisko Eclipse.
- Działa na komputerach z systemem operacyjnym Windows, Linux i macOS.
- Wymagania: 64-bitowy procesor ze wsparciem dla wirtualizacji, co najmniej 8 GB pamięci RAM i tyle samo miejsca na dysku.
- Aplikacje można testować bez dostępu do urządzenia mobilnego na emulatorze Androida.
- Aplikacje skompilowane przez Android Studio można publikować w Google Play Store.

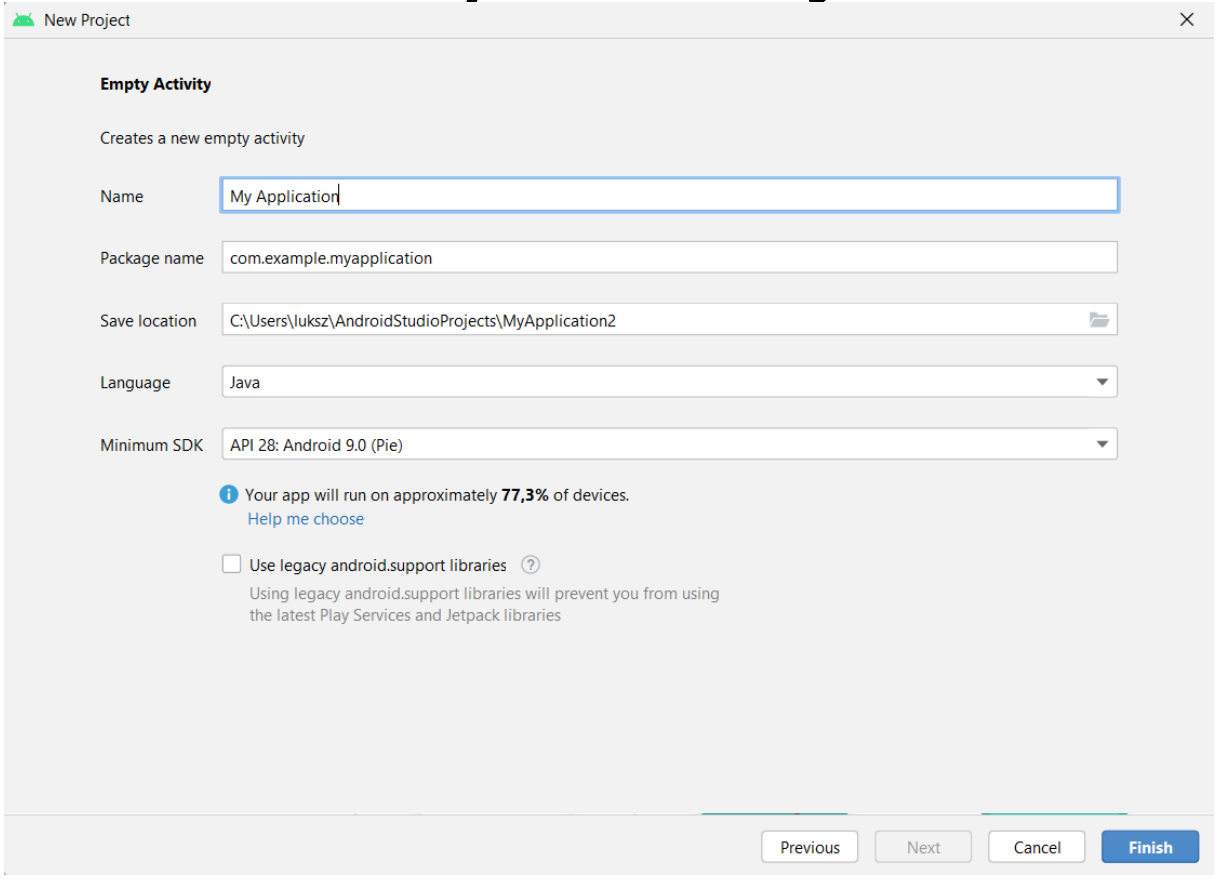
Android studio – nowy projekt



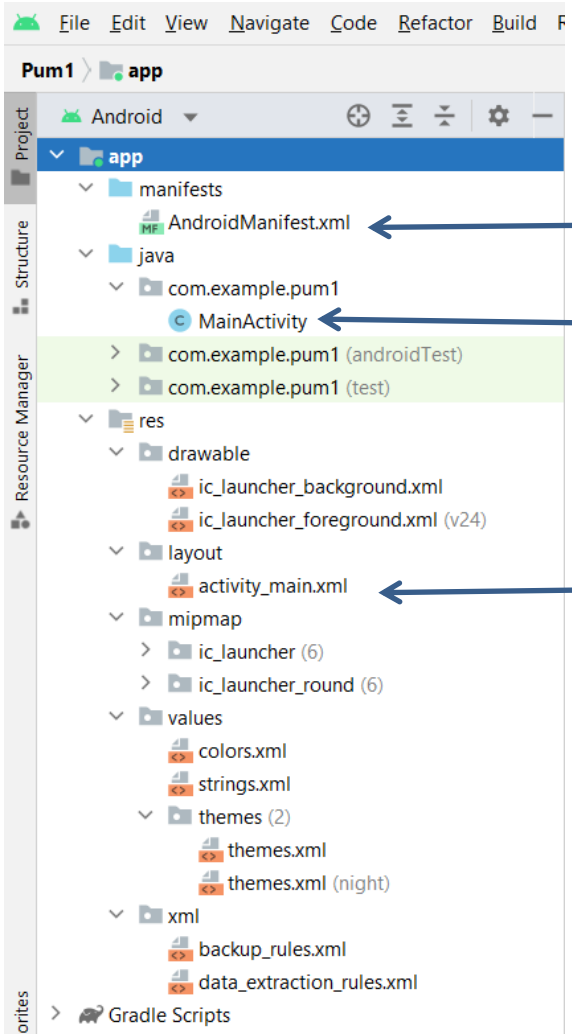
Wersje Androida



Wybór nazwy i wersji Androida



Katalog projektu



Manifest - XML

Kod aktywności - Java

Układ aktywności- XML

Pliki zasobów

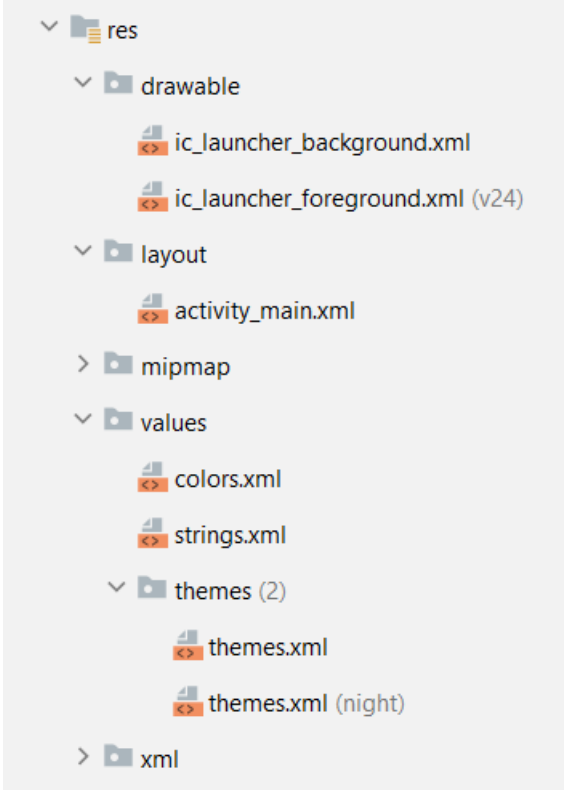
Plik AndroidManifest.xml

Główny plik konfiguracyjny projektu

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

        <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/Theme.MyApplication"
            tools:targetApi="31">
            <activity
                android:name=".MainActivity"
                android:exported="true">
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />

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

    </manifest>
```

Zasoby - resources

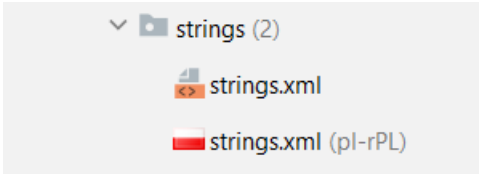
- Katalog `res` zawiera
  - Układy ekranu
  - Teksty z ich tłumaczeniami na różne języki
  - Pliki graficzne
  - Style
  - Pliki dźwiękowe



Układ – Layout: 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"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello World!"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

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

Zasoby: strings.xml



Android jest wielojęzyczny

•Aby ułatwić lokalizację aplikacji powinno się przestrzegać następujących zasad:

1. Wszystkie teksty umieszczać w zasobach
2. Domyślnie pisać po angielsku

•Potem można łatwo dokonać lokalizacji wpisując tłumaczenia w różnych językach.

•Android Studio ułatwia przeniesienie tekstów do zasobów w trakcie refaktoryzacji, czyli jak nie przestrzegamy ww. zasad, to i tak sobie poradzimy.

•Teksty są przechowywane w pliku: res/values/strings/strings.xml

Edytor tekstów z dodanymi tłumaczeniami na język polski

SHOW ALL KEYS | SHOW ALL LOCALES

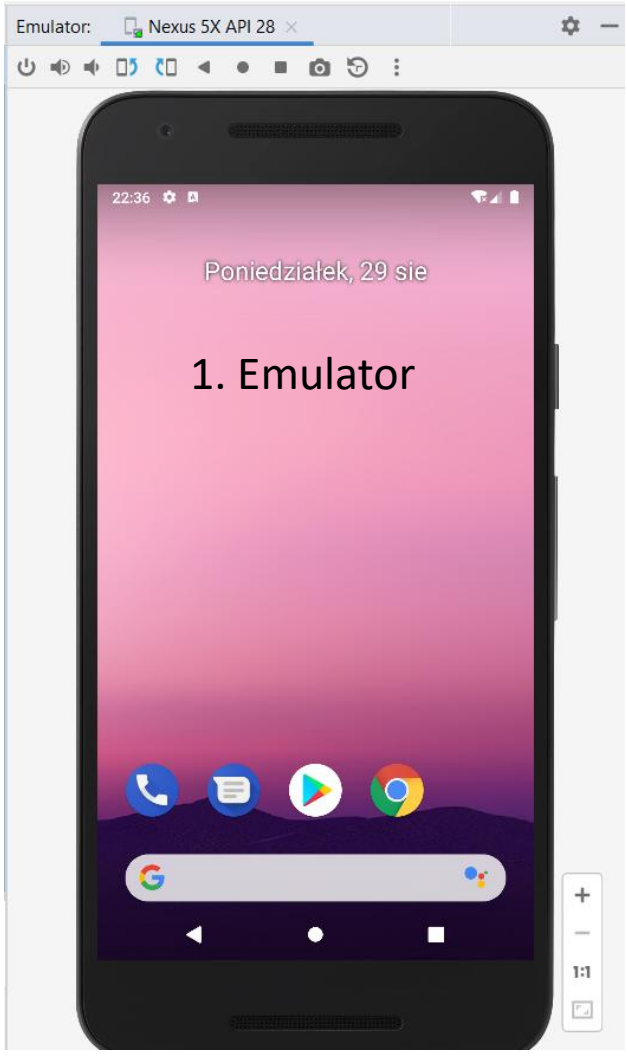
Key	Resource Folder	Untranslatable	Default Value	Polish (...
app_name	app/src/main/res	☐	My Application	Moja aplikacj
textview	app/src/main/res	☐	Text View	Okienko teks
button	app/src/main/res	☐	Button	Przycisk
button2	app/src/main/res	☐	Button 2	Przycisk 2
textview2	app/src/main/res	☐	Go home	Do domu

Kod aktywności: MainActivity.java

```
package com.example.pum4;

    import androidx.appcompat.app.AppCompatActivity;

    import android.os.Bundle;

    public class MainActivity extends AppCompatActivity
    {

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

Początkowa treść pliku, potrzebna do uruchomienia startowego programu

Metoda onCreate() jest wywoływana zawsze po otwarciu aktywności. Dodany do niej kod zostanie uruchomiony bezpośrednio po otwarciu aktywności.

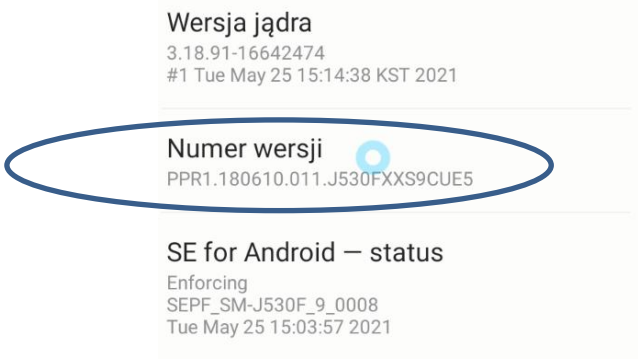


Debugowanie

2. Urządzenie fizyczne

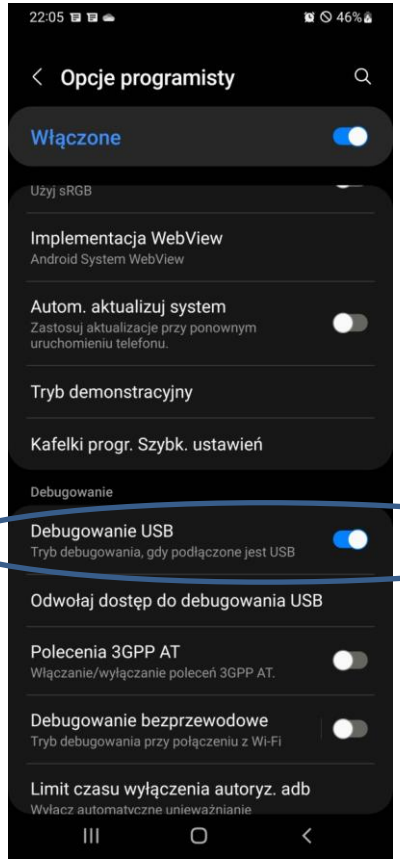
Aby umożliwić debugowanie na fizycznym urządzeniu należy uruchomić opcje programisty.

Można to zrobić klikając 7 razy na numer wersji

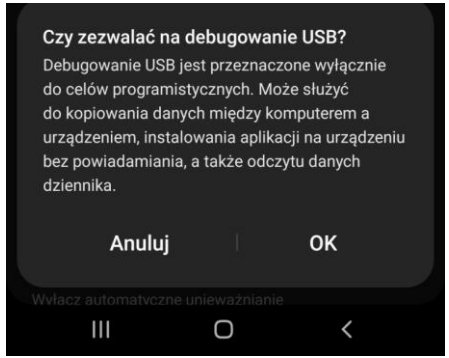


Zgoda na debugowanie USB

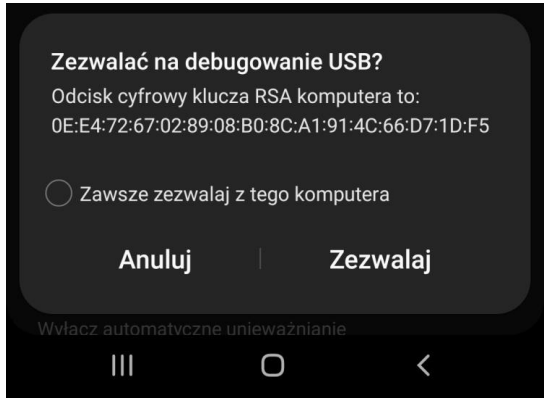
1



2

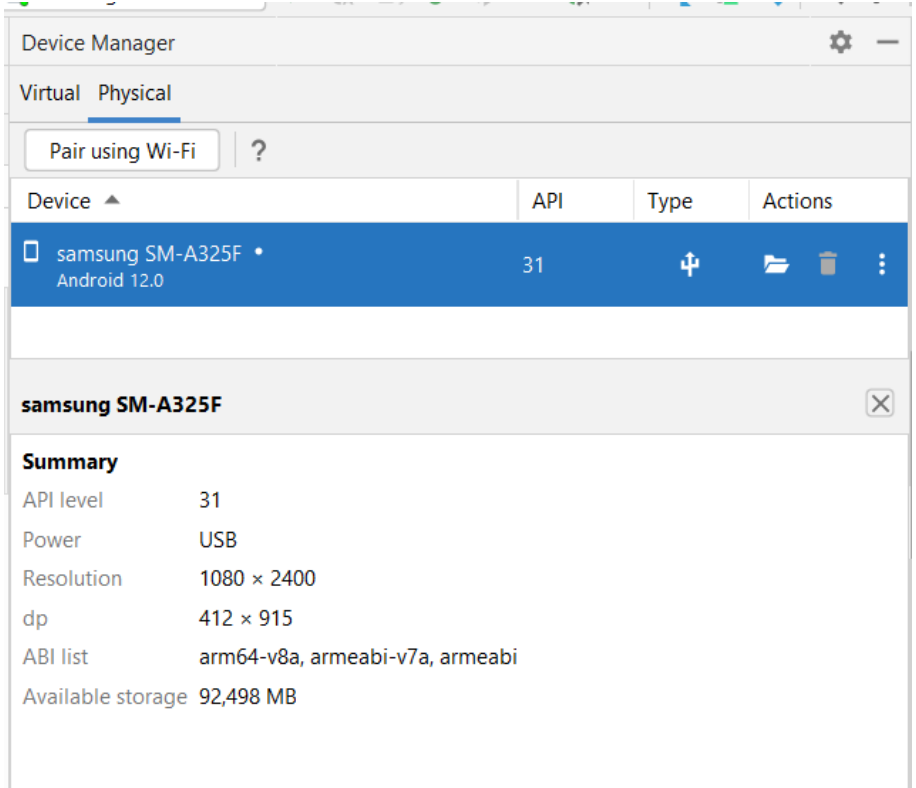


3



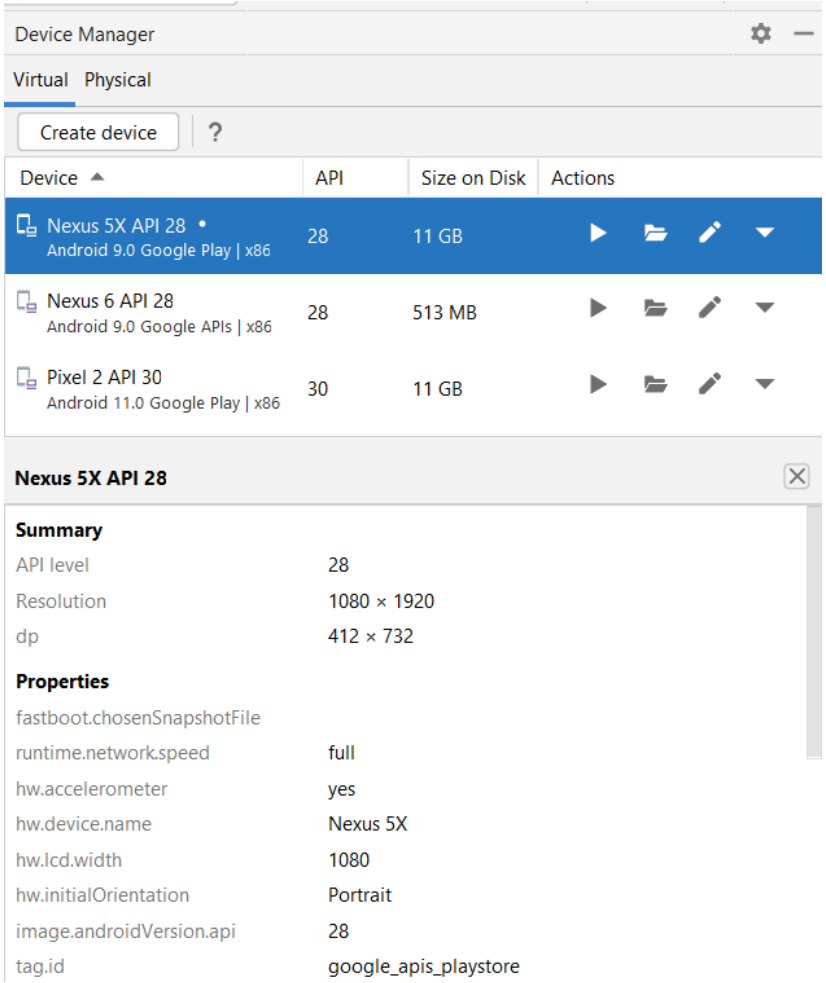
Device Manager

Device Manager służy do administrowania urządzeniami na których można debugować nasz program.
Urządzenia to emulatory i urządzenia fizyczne podłączone do Android Studio.
W Device Manager można zobaczyć dane urządzeń wirtualnych i fizycznych.



Konfiguracja Emulatorów

W Device Manager
można utworzyć dowolną liczbę urządzeń wirtualnych, różnych typów z różnymi systemami operacyjnymi.



GRAFICZNY INTERFEJS UŻYTKOWNIKA

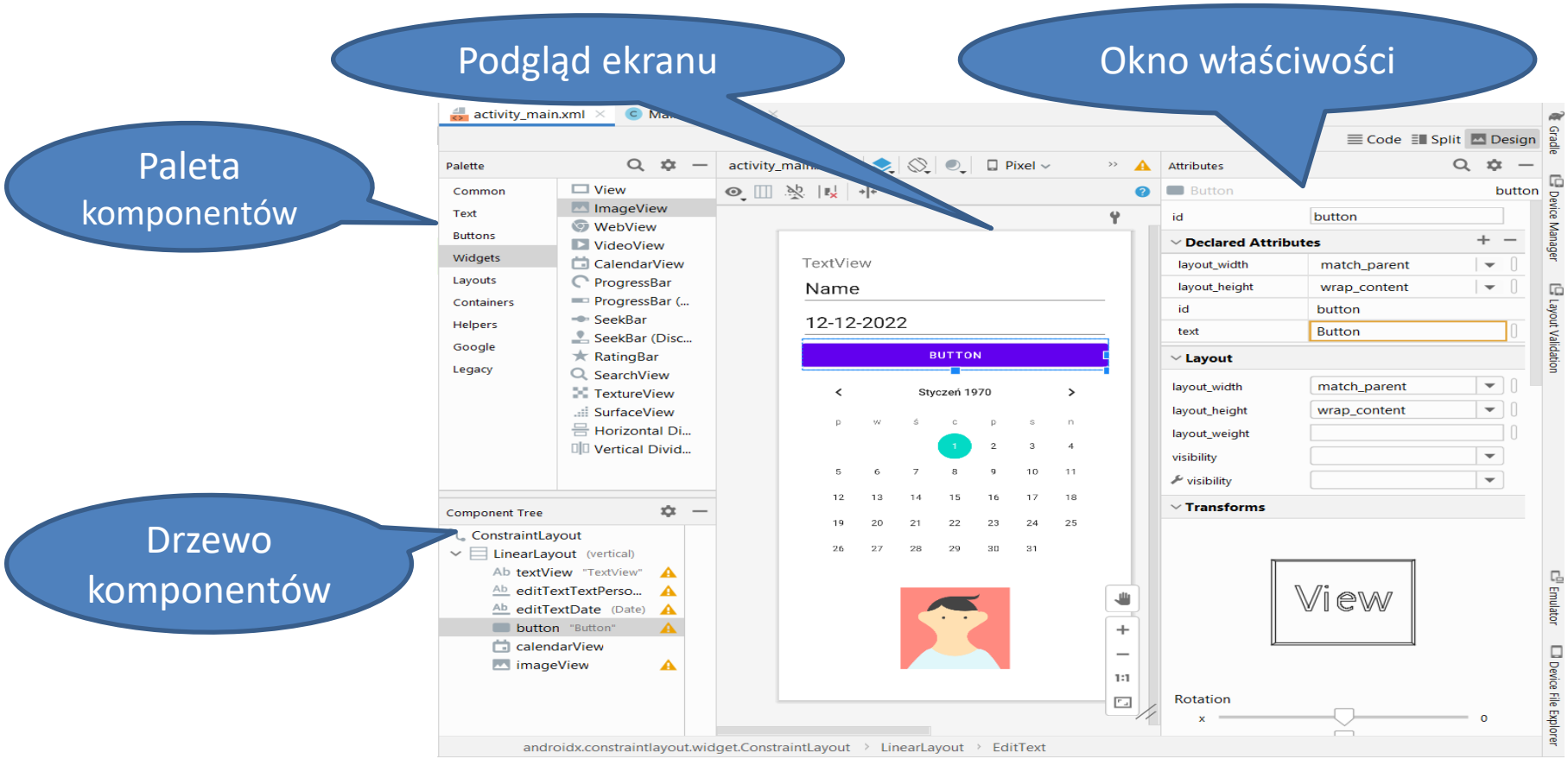
Specyfika aplikacji mobilnych

- Mały ekran, na którym bardzo trudno wszystko pomieścić
- Obracanie ekranu i różne układy pionowy i poziomy
- Urządzenie mobilne to przede wszystkim narzędzie do komunikacji, zastosowanie to przekazywanie wiadomości i Internet rzeczy
- Konieczność oszczędzania energii
- Wielojęzyczność
- Zanikający dostęp do internetu
- Większa współpraca aplikacji między sobą niż w innych systemach
- Dostęp do czujników

Różna rozdzielczość różnych telefonów

- **W Androidzie praktycznie nie używa się pikseli jako jednostki wielkości.**
- Zamiast pikseli używa się jednostki „dp” czyli „density independent pixel”, co można przetłumaczyć na pixel niezależny od gęstości.
- Tworząc interfejs deklaruje się na przykład, że przycisk ma 30dp szerokości i Android dba o to, żeby ten przycisk był jednakowej wielkości zarówno na urządzeniu z rozdzielczością HVGA jak i 4K.
- Rozdzielczość urządzenia w dp można zobaczyć w Device Manager

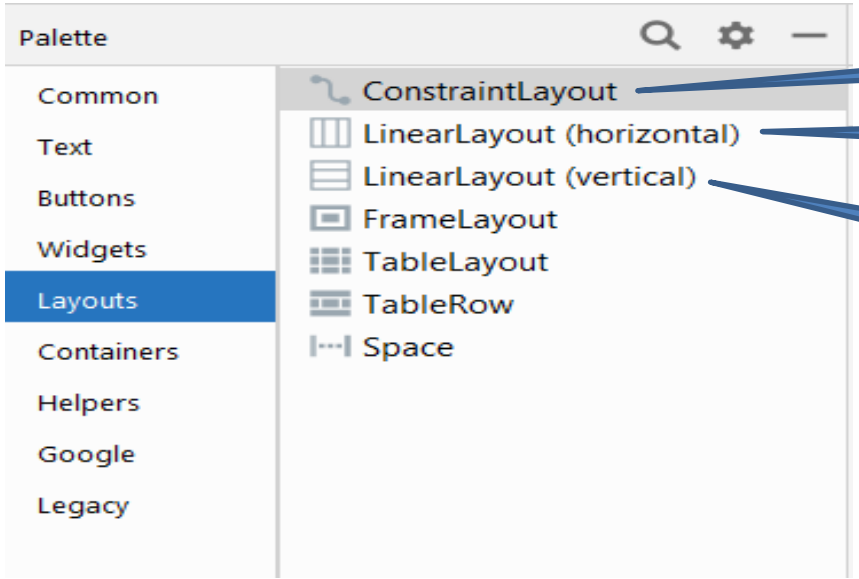
Edytor typu WYSIWIG do edycji układu, umożliwia tworzenie interfejsów metodą przeciągnij i upuść



Klasa View

- Klasa View reprezentuje podstawowy komponent Androida, za pomocą którego można tworzyć graficzne interfejsy użytkownika. Stanowi kontener dla wszystkich elementów, które można wyświetlić.
- Widok zajmuje prostokątną przestrzeń na ekranie i jest odpowiedzialny za rysowanie jak i przetwarzanie zdarzeń.
- Widżety (ang.widgets) są potomkami klasy View. Używane są do stworzenia interaktywnych komponentów graficznych takich jak przyciski, etykiety, pola tekstowe itp., które w dalszej części wykładu będą nazywane kontrolkami lub komponentami.
- Układy (ang. layouts) to niewidzialne kontenery umożliwiające pozycjonowanie innych widoków oraz zagnieżdżonych układów.

Układy - Layouty



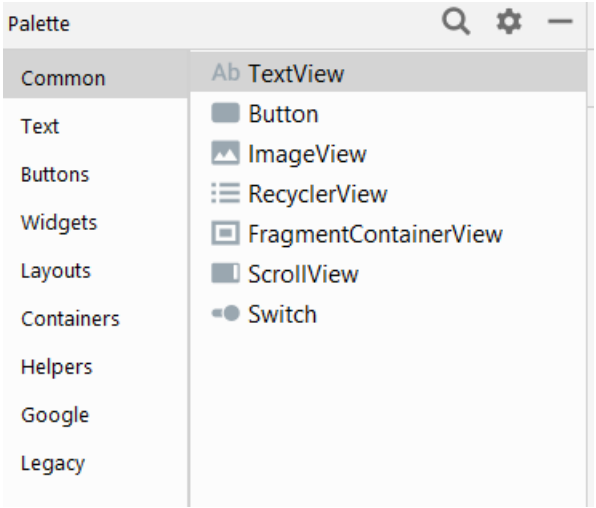
Układ ze sprężynkami

Układ poziomy – jeden element obok drugiego

Układ pionowy – jeden element na drugim

Układy można zagnieżdżać

Kontrolki

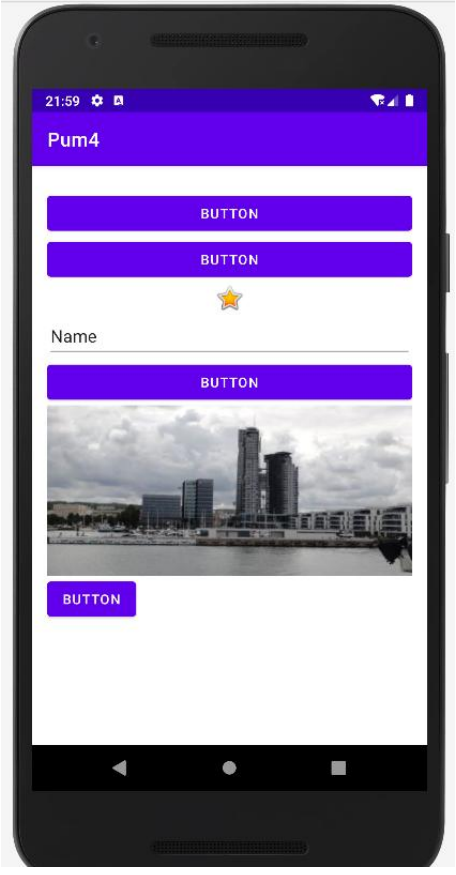


Do budowy pierwszych aplikacji wystarczą trzy kontrolki:

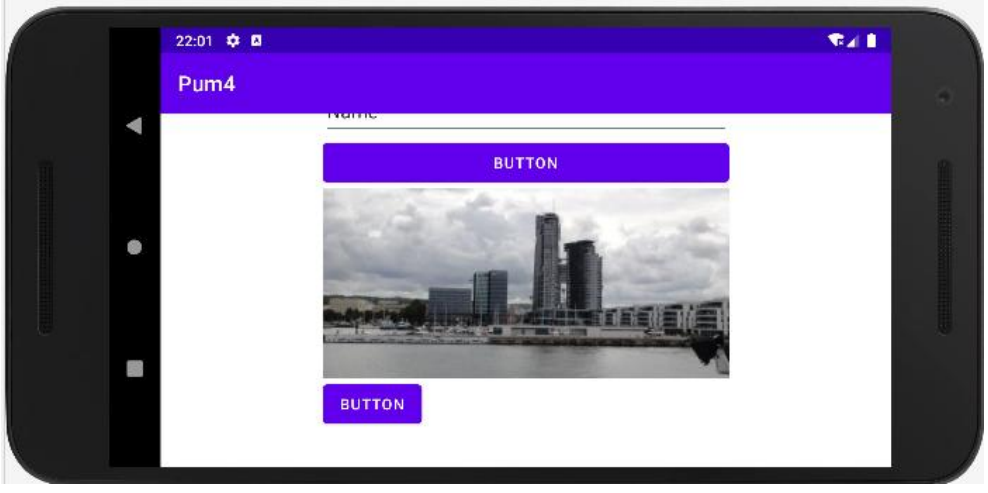
1. **TextView** (etykieta) do drukowania na ekranie,
2. **Button** do generowania zdarzeń,
3. **Plain Text** do wprowadzania danych na ekranie.

Przykład budowy układu

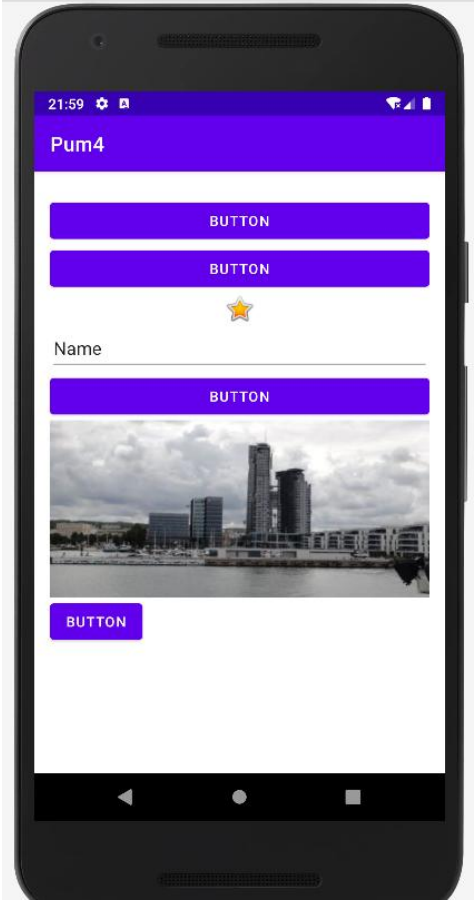
Układ liniowy pionowy w 2. pozycjach

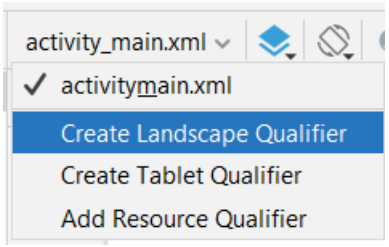


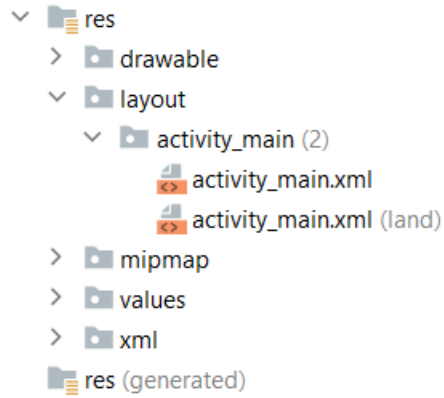
Jest to ten sam układ w 2 pozycjach
pionowej (portret),
poziomej (landscape).
W układzie poziomym część kontrolek
może być niewidoczna

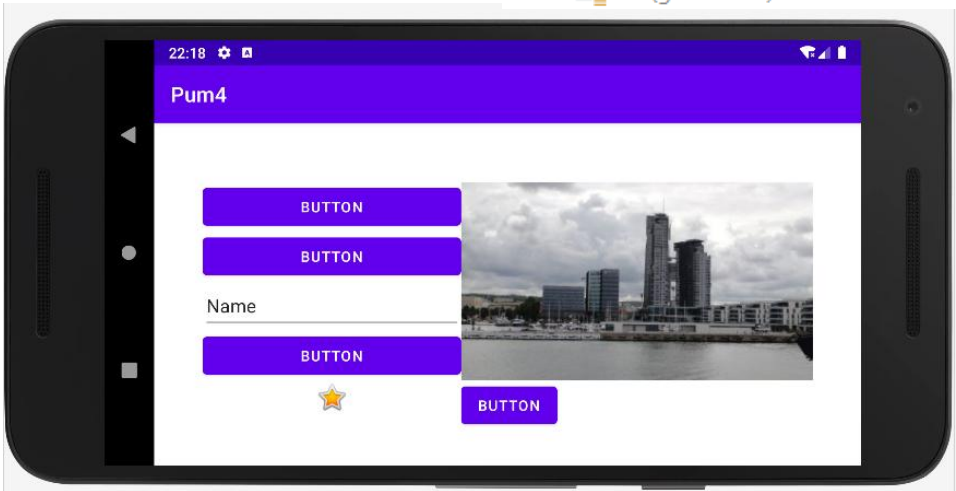


Różne układy pionowy i poziomy









Lokalizacja aplikacji – tłumaczenie na różne języki

- Wszystkie napisy powinny być odwołaniami do zasobu strings.xml.
- Domyślne napisy najlepiej w języku angielskim.
- Po dodaniu tłumaczeń napisy w aplikacji będą w języku skonfigurowanym na urządzeniu, jeśli jest odpowiednie tłumaczenie.
- Jeśli nie ma tłumaczenia w danym języku wyświetlone będą napisy w języku domyślnym.

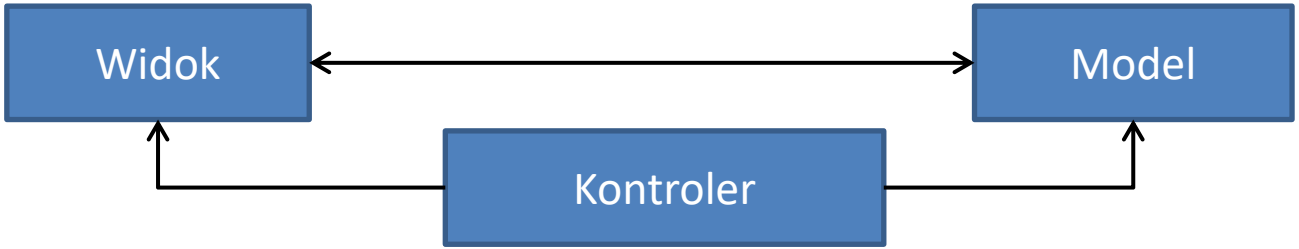
Odwołanie do napisu text_1 w kodzie programu to: `R.string.text_1`
Odwołanie w oknie właściwości obiektu to: `@string/text_1`

Android Studio pomaga przenieść wszystkie napisy do zasobów podczas refaktoryzacji programu; Wystarczy kliknąć na tekst, wybrać właściwą pozycję z menu kontekstowego nazwać zasób i wszystko.

KOMUNIKACJA Z APLIKACJI Z UŻYTKOWNIKIEM

Wzorzec projektowy MVC

- Wzorzec Model-Widok(Prezentacja)-Kontroler *(MVC)*
- **Model** składa się z kodu w języku JAVA i odwołań API, które zarządzają zachowaniem danych aplikacji; warstwa biznesowa?.
- **Widok** (warstwa prezentacji) to zestaw ekranów, które użytkownik widzi i wchodzi w interakcję; graficzny interfejs użytkownika – GUI.
- **Kontroler** implementowany m. in. poprzez system operacyjny jest odpowiedzialny za interpretację zdarzeń użytkownika i systemowych. Zdarzenia mogą pochodzić z różnorakich źródeł: ekranów dotykowych, modułu GPS, Broadcast receivera, usług działających w tle. Kontroler dokonuje w modelu i/lub widoku odpowiedne zmiany.



Konwencja nadawania nazw w języku Java

- Nazwy klas z dużej litery
- Nazwy obiektów z małej litery
- Nazwy funkcji z małej litery
- Stałe – wszystko dużymi literami

Toast - mała chmurka u dołu ekranu

Poniższy kod można umieścić wewnątrz metody onCreate(),
która jest wywoływana przy pierwszym uruchomieniu aktywności

Najprostsza wersja:

```
Toast.makeText(getApplicationContext(),
               "Pojawiam się po uruchomieniu programu",
               Toast.LENGTH_LONG).show();
```

Wersja z argumentami metody makeText() utworzonymi wcześniej jako zmienne:

```
Context context = getApplicationContext();
CharSequence text = "Tekst wiadomości!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

Snackbar - inna chmurka u dołu ekranu

Poniższy kod można umieścić wewnątrz metody `onCreate()`,
która jest wywoływana przy pierwszym uruchomieniu aktywności

Najprostsza wersja:

```
Snackbar.make(view, "Replace with your own action",
    Snackbar.LENGTH_LONG)
            .setAction("Action", null).show();
```

Przykład:

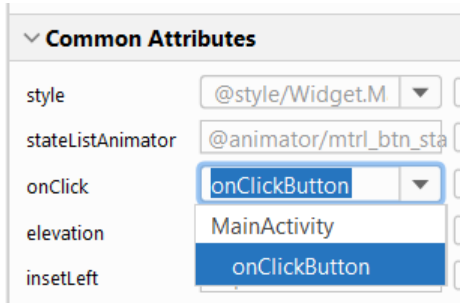
```
((Button)findViewById(R.id.button)).setOnClickListener(view ->
        Snackbar.make((ImageView)findViewById(R.id.imageView),
                "Replace with your own action", Snackbar.LENGTH_LONG)
        .setAction("Action", null).show());
```

Obsługa zdarzenia kliknięcia przycisku

1. Wstawienie kodu funkcji (metody) obsługi zdarzenia do pliku MainActivity.java

```
public void onClickButton(View view) {
        Toast.makeText(getApplicationContext(),
                "Pojawiam się po kliknięciu przycisku",
                Toast.LENGTH_LONG).show();
    }
```

2. Skojarzenie funkcji ze zdarzeniem kliknięcia przycisku w pliku układu `activity_main.xml`




```
<Button
    android:id="@+id/button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="onClickButton"
    android:text="Button 1" />
```

Obsługa zdarzenia zdefiniowana w kodzie aktywności

np. wewnątrz metody onCreate()

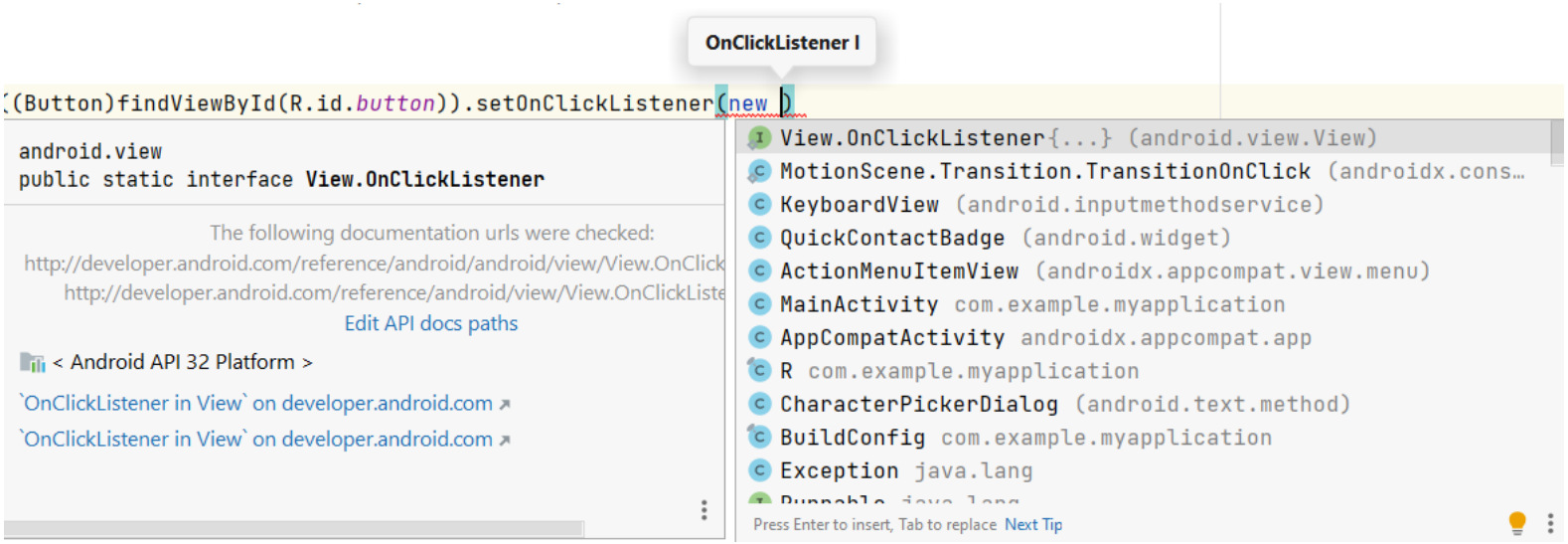
```
// Deklaracja i inicjacja zmiennej button
Button button = (Button)findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // kod metody
    }
});
```

Metoda `findViewById(R.id.button)` odszukuje w zasobach kontrolkę po jej identyfikatorze

Przy tworzeniu „nasłuchiwacza” (listenera) korzystamy z podpowiedzi, jak na następnym slajdzie

Obsługa zdarzenia zdefiniowana w kodzie aktywności

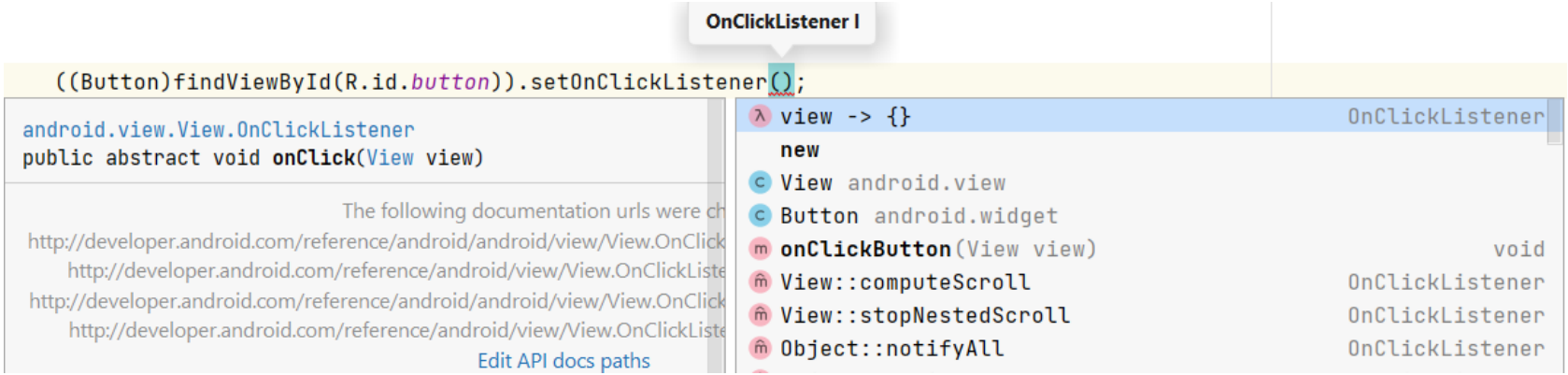
bez deklarowania przycisku




```
((Button)findViewById(R.id.button)).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

    }
});
```

Obsługa zdarzenia zdefiniowana w kodzie aktywności

notacja Lambda




```
((Button)findViewById(R.id.button)).setOnClickListener(view ->
        Toast.makeText(this, "Tekst",Toast.LENGTH_LONG).show());
```

```
((Button)findViewById(R.id.button)).setOnClickListener(view -> {
    // Nawiasy {} gdy jest więcej niż jedna linia kodu
    Toast.makeText(this, "Tekst", Toast.LENGTH_SHORT).show();
});
```

Dostęp do kontrolek z kodu programu

- W kodzie programu należy zadeklarować obiekt odpowiedniego typu, następnie związać go z kontrolką za pomocą funkcji `findViewById()`.
- Parametrem funkcji jest identyfikator kontrolki znajdującej się w zasobach: `R.id.`*`identyfikator_kontrolki`*

Przykład:

```
Button button;  // Deklaracja przycisku
// Przypisanie kontrolki
button = (Button)findViewById(R.id.button);
button.setText("Przycisk testowy"); // Zmiana napisu
```

Zmiana tekstu kontrolki w kodzie programu

Metoda `getText()` odczytuje i zwraca tekst kontrolki tekstowej.
Metoda `setText(String)` wpisuje tekst do kontrolki tekstowej.

Przykład odczytu tekstu z kontrolki *editText* i umieszczenie go kontrolce *textView*:

```
EditText editText = findViewById(R.id.editTextText);
String text = editText.getText().toString();
TextView textView = findViewById(R.id.textView);
textView.setText(text);
```

To samo można wykonać za pomocą jednej, ale długiej linii kodu:

```
((TextView)findViewById(R.id.textView))
        .setText(((EditText)findViewById(R.id.editTextText))
                .getText().toString());
```

Dodanie do układu kontrolki z kodu programu

Przykład dodania do układu liniowego nowego przycisku z kodu programu. Właściwości przycisku można zmieniać przed dodaniem go do GUI lub po dodaniu, pod warunkiem zapamiętania referencji.

```
LinearLayout layout = ((LinearLayout) findViewById(R.id.linear1));
Button button = new Button(getApplicationContext());
button.setText("Nowy przycisk z kodu");
layout.addView(button);
button.setOnClickListener(v -> Toast.makeText(this,
        "Przycisk", Toast.LENGTH_SHORT).show());
```

Zadanie 1. Układy

- Stworzyć układ z 3 przyciskami, różny dla orientacji pionowej i poziomej
- Napisy na przyciskach w 2. językach
- Po kliknięciu przycisku pojawia się tost w jednym z 2. języków
- Przetestować aplikację z 2. ustawieniami języka w telefonie

Zadanie 2. Rozwiązywanie równań kwadratowych

- Stworzyć aplikację obliczającą pierwiastki równania kwadratowego
- Współczynniki a, b i c w polach Number (Decimal)
- Wynik w dowolnym polu tekstowym i/lub w toście
- Obliczenia są wykonywane po kliknięciu przycisku
- Program powinien wyznaczać pierwiastki rzeczywiste i zespolone sprzężone, jednostkę urojoną oznaczyć jako *j*.