



APLIKACJE MOBILNE

Wykład 02

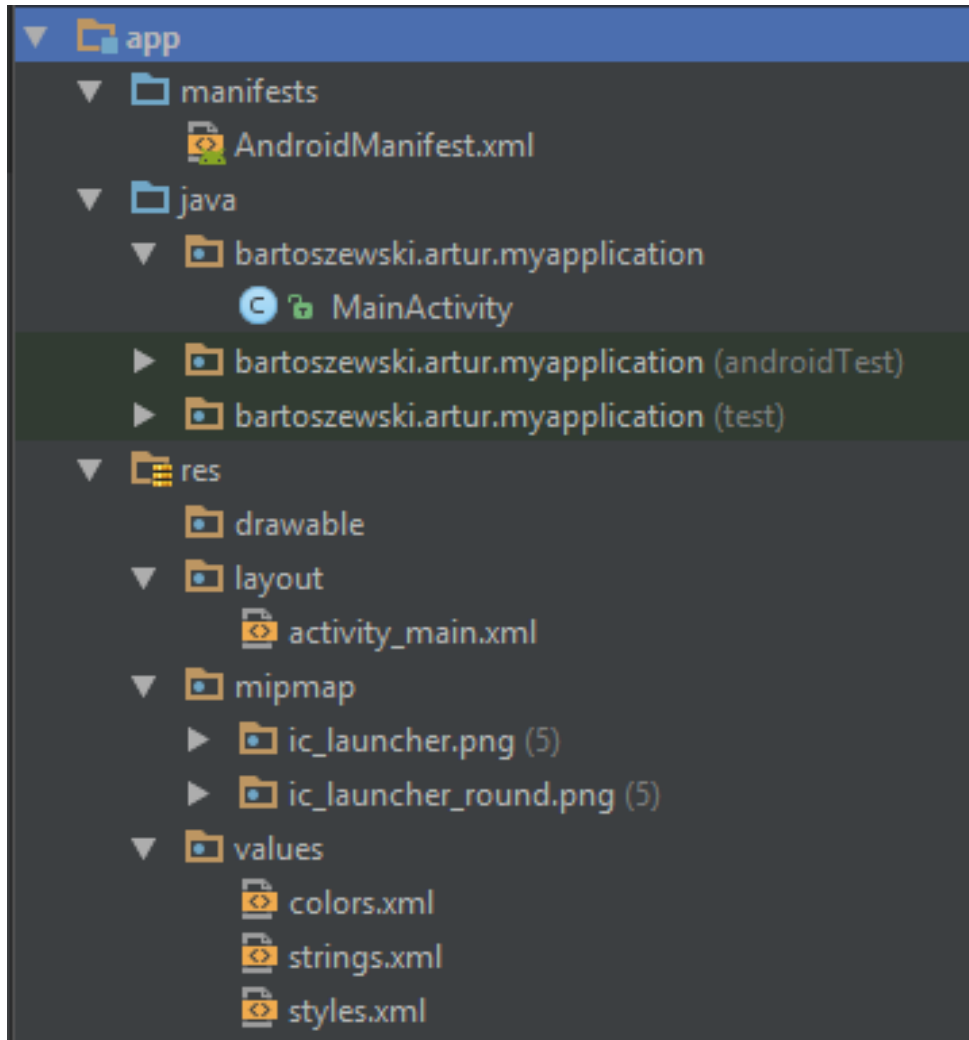
dr Artur Bartoszewski



ANDROID STUDIO



Struktura projektu



Sekcja APP

Struktura projektu

AndroidManifest.xml - Manifest – Plik sterujący, który zawiera informacje o charakterze aplikacji oraz każdym jej komponencie. Manifest wykorzystywany jest do zapisywania informacji na temat uprawnień, poszczególnych aktywności, czy też informacji na temat wykorzystywanej wersji API.

Java - w tym katalogu przechowywane są wszystkie nasze pliki zawierające kod Javy.

Res - katalog res (resources) zawiera wszystkie statyczne zasoby – obrazki, pliki dźwiękowe, video itp

Layout - Katalog przeznaczony na pliki xml odpowiadające za interfejs użytkownika w aplikacji (layout).

Struktura projektu

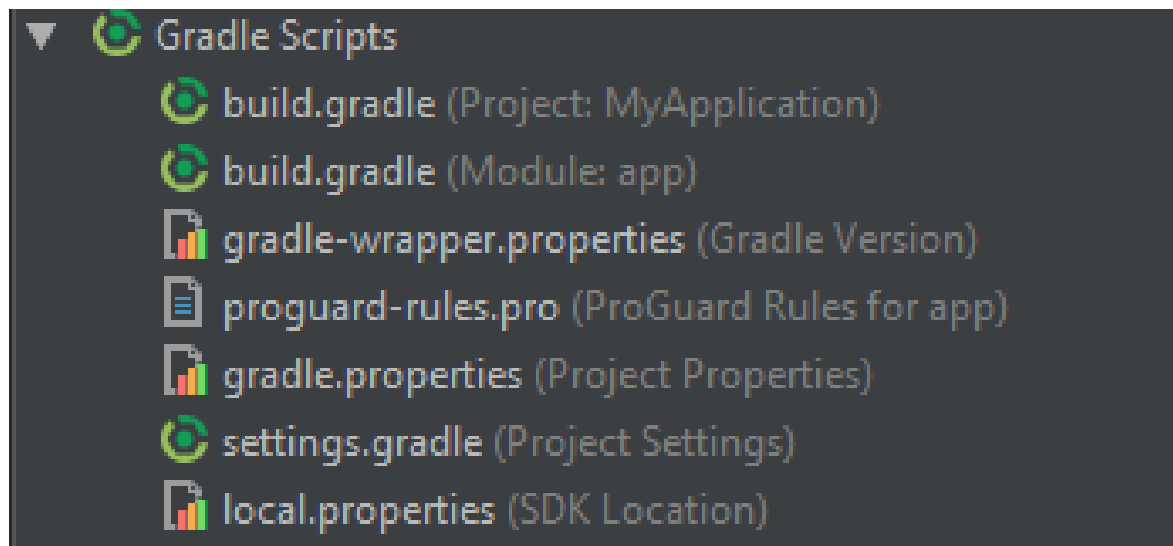
Menu - katalog dla plików xml definiujących menu w aplikacji.

Minimap - katalog przeznaczony do przechowywania ikony aplikacji.

Values – katalog zawierające wartości różnego rodzaju zmiennych i stałych aplikacji. Zawiera:

- **colors.xml** - plik w którym zdefiniowane są kolory, które później możemy użyć w różnych miejscach aplikacji.
- **dimens.xml** - plik w którym definiujemy marginesy wykorzystywane w layoucie.
- **strings.xml** - plik w którym definiujemy wszystkie rzeczy tekstowe jakie będą zawarte w interfejsie użytkownika. Dzięki temu w przyszłości łatwo będziemy mogli dorobić obsługę innych języków.

Sekcja GRADLE



Struktura projektu

Skrypty GRADLE - skrypty „budujące” aplikację.

build.gradle - plik zawierający informacje dotyczące kompilacji aplikacji. Można go edytować aby dodać własne moduły, biblioteki czy też zdefiniować miejsce przechowywania kluczy. Jest on integralną częścią projektu.

gradle.properties - ustawienia plików „Gradle”

settings.gradle – plik zawiera informacje o wszystkich podprojektach jakie muszą zostać skompilowane przy kompilacji aplikacji.

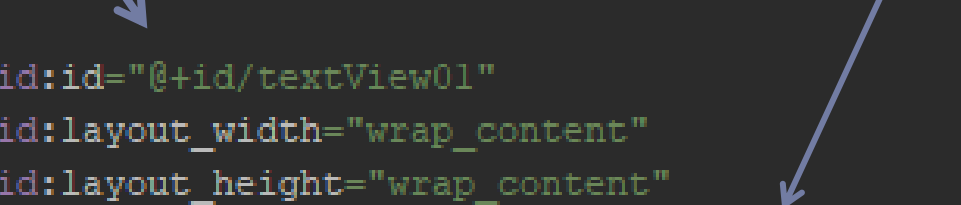
**WIDOKI:
BUTTON,
EDITTEXT,
TEXTVIEW**



Obsługa widoków - TextView

Nadanie widokowi Id jest konieczne, jeżeli będziemy się do niego odwoływać w kodzie Java

Tekst może być wypisany na poziome layoutu, lecz można go modyfikować w kodzie Javy



```
<TextView
    android:id="@+id/textView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="To jest widok wyświetlający tekst na ekranie"
    android:textSize="10pt"
    android:textColor="@android:color/holo_red_dark"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Obsługa widoków - EditText

Nadanie widokowi Id jest konieczne, jeżeli będziemy się do niego odwoływać w kodzie Java (czyli praktycznie zawsze)

```
<EditText
    android:id="@+id/editText01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10"
    android:hint="Tu wpisz dane"
    android:inputType="text"
/>
```

Przewidywana
ilość znaków
(szerokość)

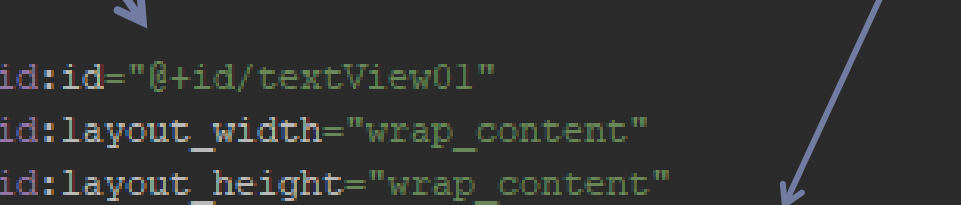
Typ okna dialogowego

Podpowiedź

Obsługa widoków - Button

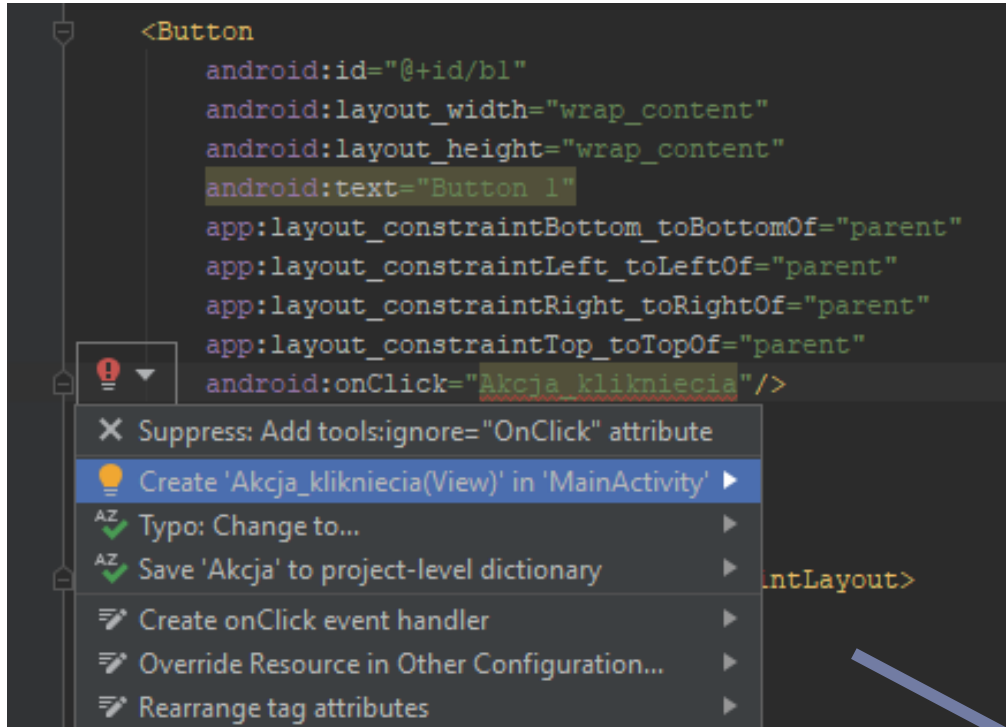
Nadanie widokowi Id jest konieczne, jeżeli będziemy się do niego odwoływać w kodzie Java

Tekst może być wypisany na poziome layoutu, lecz można go modyfikować w kodzie Javy



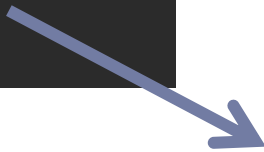
```
<TextView
    android:id="@+id/textView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="To jest widok wyświetlający tekst na ekranie"
    android:textSize="10pt"
    android:textColor="@android:color/holo_red_dark"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Obsługa widoków - Button



Do przycisku dodać
można atrybut **onClick**

W kodzie Javy
utworzona zostanie
metoda obsługi
kliknięcia przycisku



```
public void Akcja_klikniecia(View view) {
}

}
```

Wypisanie tekstu z poziomu Javy

```
TextView tView01;  
tView01 = findViewById(R.id.textView01);
```

Pierwszym krokiem zawsze jest utworzenie w kodzie zmiennej (obiektu) typu TextView i powiązanie jej z Id widoku na layoucie

```
tView01.setText("Tekst do wypisania");
```

Tekst wyświetlany zmienić możemy za pomocą metody setText tej zmiennej

```
int x = 100;  
tView01.setText(String.valueOf(x));
```

Aby wypisać liczbę należy zamienić ją na łańcuch

Wczytanie tekstu z poziomu Javy

```
EditText editText01;  
editText01 = findViewById(R.id.editText01);
```

Pierwszym krokiem zawsze jest utworzenie w kodzie zmiennej (obiektu) typu EditText i powiązanie jej z Id widoku na layoucie

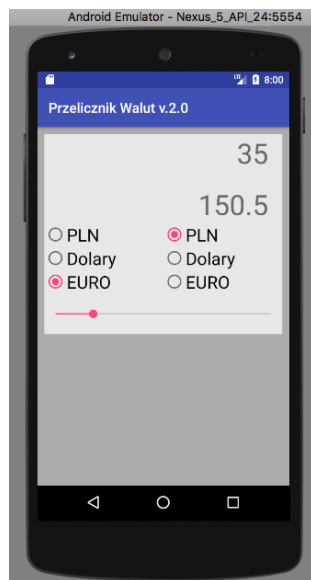
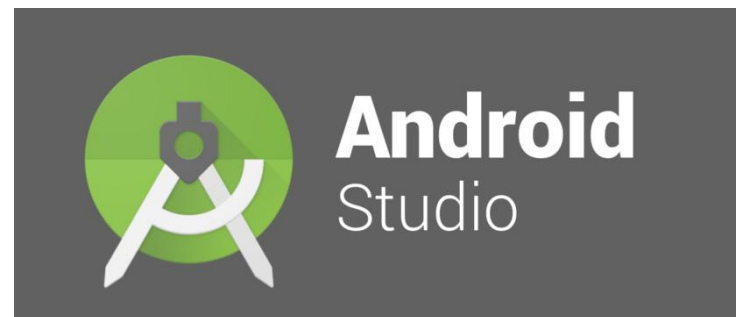
```
String dana;  
dana = editText01.getText().toString();
```

Tekst wczytać można za pomocą metody
getText().toString()

```
int danaLiczbowa = Integer.parseInt(dana);
```

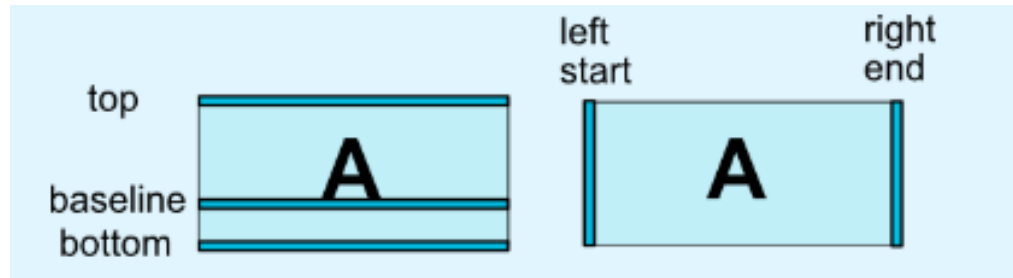
Aby wczytać liczbę należy wczytać tekst i
zamienić go na liczbę

LAYOUT APLIKACJI



Constraint Layout

Constraint Layout daje możliwość podłączenia każdej z krawędzi widoku do jednej z dwóch krawędzi dowolnego innego widoku, linii pomocniczej lub krawędzi ekranu.



Źródło: <https://developer.android.com>

Podstawowe polecenia:

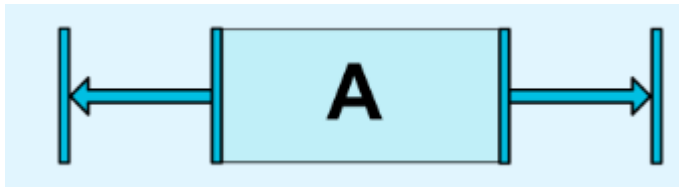
- `layout_constraintTop_toBottomOf`
- `layout_constraintBottom_toBottomOf`
- `layout_constraintBottom_toTopOf`
- `layout_constraintTop_toTopOf`
- `layout_constraintStart_toStartOf`
- `layout_constraintStart_toEndOf`
- `layout_constraintEnd_toEndOf`
- `layout_constraintEnd_toStartOf`

Parametrem każdego z tych poleceń jest **Id** innego widoku

Constraint Layout

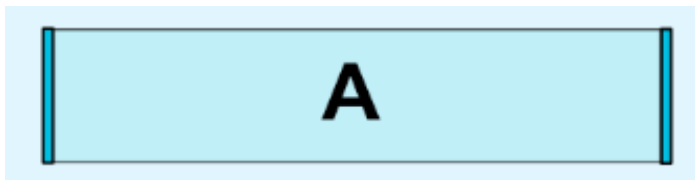
Wyśrodkowanie vs. rozciągnięcie na cały ekran

Wyśrodkowanie



```
android:layout_width="wrap_content"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
```

Dopasowanie do szerokości ekranu



Źródło: <https://developer.android.com>

```
android:layout_width="0dp"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
```

Ustawienie szerokości obiektu na 0dp oznacza, że powinna być ona wyliczona z innych ustawień

Constraint Layoutu

Guideline – linia pomocnicza - pozwala na ustawienie linii odniesienia do pozycjonowania pozostałych widoków. Może zostać ustawiona procentowo lub na konkretną wartość.

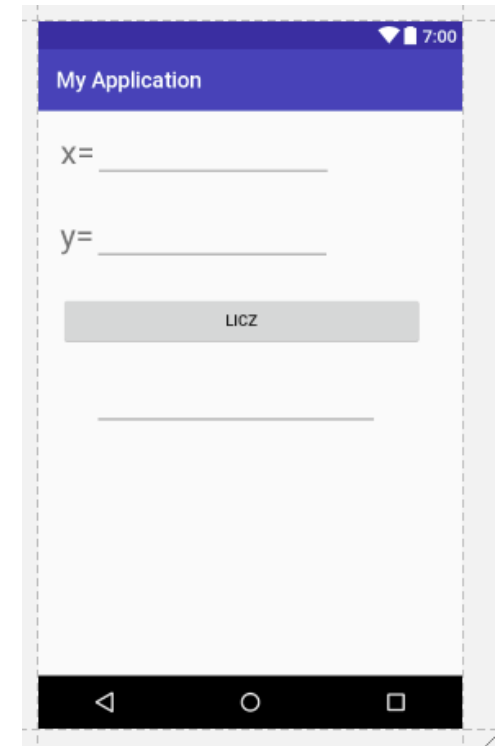
```
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_begin="20dp" />
```

```
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent="0.5" />
```

ZADANIE PRAKTYCZNE:

Zadanie:

- aplikacja posiada dwa pola edycji w które wprowadzamy liczby
- po kliknięciu na przycisk licz są one sumowane i wypisywane w trzecim polu edycji



Budowa interfejsu

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:padding="20dp"
8      tools:context="pl.uniwersytetradom.bartoszewski.artur.myapplication.MainActivity"
9
10     <android.support.constraint.Guideline
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:id="@+id/guideline"
14         app:layout_constraintGuide_begin="32dp"
15         android:orientation="vertical" />
16
17     <TextView...>
23
24     <EditText...>
31
32     <TextView...>
41
42     <EditText...>
51
52     <Button...>
61
62     <EditText...>
71
72 </android.support.constraint.ConstraintLayout>
73

```

Budowa interfejsu

<TextView

```
android:id="@+id/textView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="x="
android:textSize="30dip" />
```

<EditText

```
android:id="@+id/editText"
android:layout_width="215dp"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="number"
app:layout_constraintStart_toEndOf="@id/textView" />
```

<Button

```
android:id="@+id/b_licz"
style="@style/Widget.AppCompat.Button"
android:layout_width="328dp"
android:layout_height="wrap_content"
android:text="LICZ"
android:layout_marginTop="28dp"
app:layout_constraintTop_toBottomOf="@+id/editText2"
android:onClick="liczenie" />
```

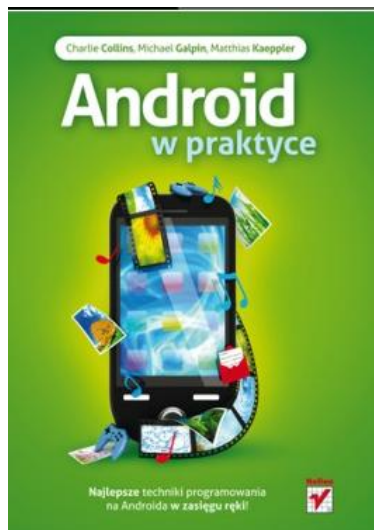
Kod Java

```

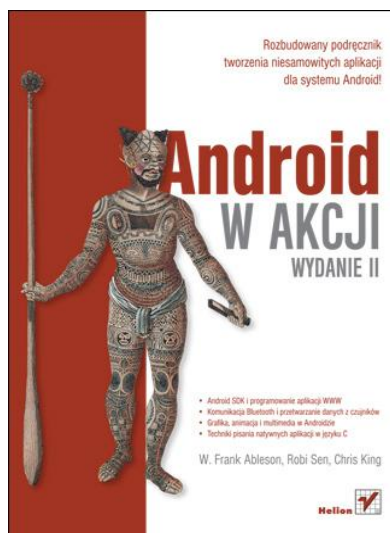
1  package pl.uniwersytetradom.bartoszewski.artur.myapplication;
2
3  import android.support.v7.app.AppCompatActivity;
4  import android.os.Bundle;
5  import android.view.View;
6  import android.widget.EditText;
7
8  import static java.lang.StrictMath.sqrt;
9  import static pl.uniwersytetradom.bartoszewski.artur.myapplication.R.id.editText2;
10
11  public class MainActivity extends AppCompatActivity {
12
13      @Override
14      protected void onCreate(Bundle savedInstanceState) {
15          super.onCreate(savedInstanceState);
16          setContentView(R.layout.activity_main);
17      }
18
19      public void liczenie (View view) {
20          double x=0, y=0, wynik;
21          int z;
22          String a,b;
23          EditText poleX = (EditText) findViewById(R.id.editText);
24          EditText poleY = (EditText) findViewById(R.id.editText2);
25          EditText poleWynik = (EditText) findViewById(R.id.wynik);
26
27          a = poleX.getText().toString();
28          x = Double.parseDouble(a);
29          b = poleY.getText().toString();
30          y = Double.parseDouble(b);
31
32          wynik= sqrt(x*x + y*y);
33          poleWynik.setText(String.valueOf(wynik));
34      }
35  }

```

```
19 public void liczenie (View view) {  
20     double x=0, y=0, wynik;  
21     int z;  
22     String a,b;  
23     EditText poleX = (EditText) findViewById(R.id.editText);  
24     EditText poleY = (EditText) findViewById(R.id.editText2);  
25     EditText poleWynik = (EditText) findViewById(R.id.wynik);  
26  
27     a = poleX.getText().toString();  
28     x = Double.parseDouble(a);  
29     b = poleY.getText().toString();  
30     y = Double.parseDouble(b);  
31  
32     wynik= sqrt(x*x + y*y);  
33     poleWynik.setText(String.valueOf(wynik));  
34 }
```



<https://developer.android.com>



<https://javastart.pl/baza-wiedzy/android/>

<https://forum.android.com.pl>