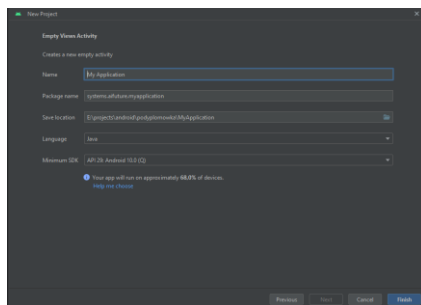
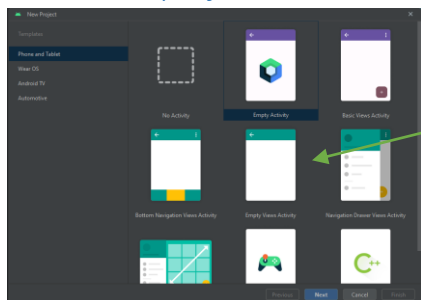


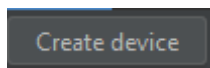
Tworzenie projektu



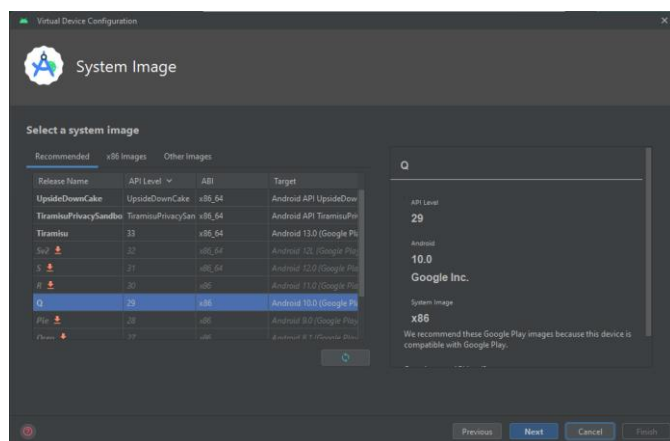
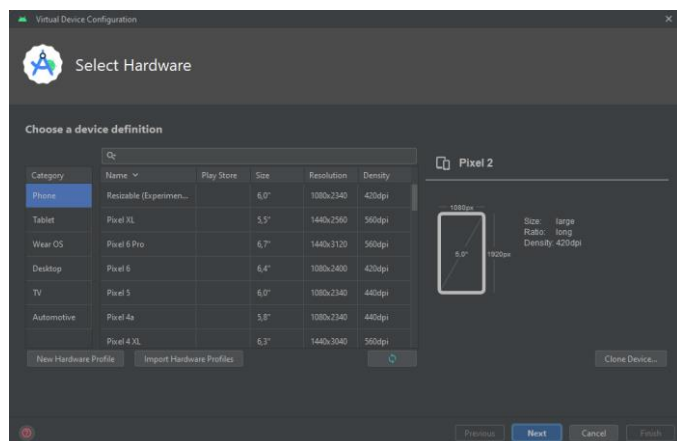
- Wybór Activity
- Dodanie nazwy aplikacji
- Dodanie domeny
- Ustawienie lokalizacji
- Ustawienie języka programowania (JAVA)
- Wybór minimalnej wersji Android (SDK)

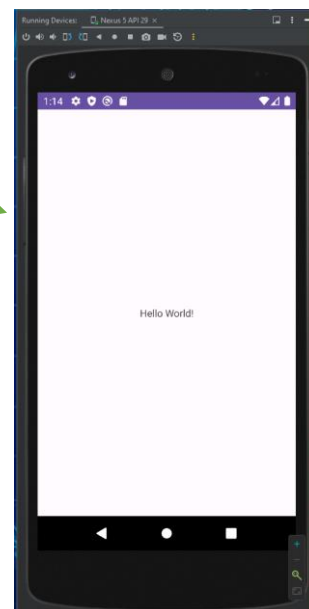
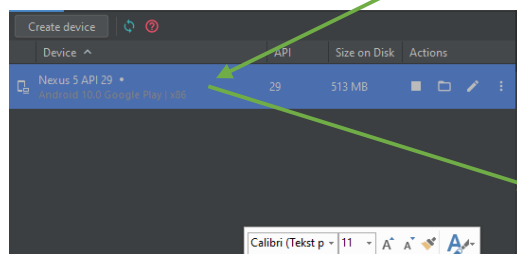
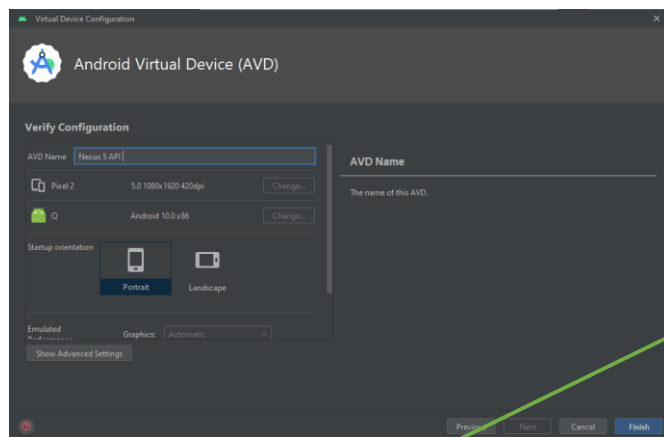
PODCZAS TWORZENIA PROJEKTU MOŻE ZOSTAĆ WYŚWIETLONY
ALERT ZAPORY WINDOWS – NALEŻY GO ZAAKCEPTOWAĆ

Tworzenie wirtualnej maszyny



- Wybór: Device Manager
- Następnie Create Device
- Wybór urządzenia
- Wybór Androida na urządzeniu (SDK)
- Dalej

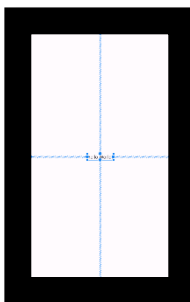




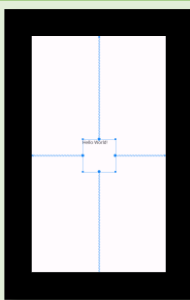
- Nazwa urządzenia
- Orientacja
- Finish
- Uruchomienie z listy menadżera urządzeń

Widgety – przykład zastosowania

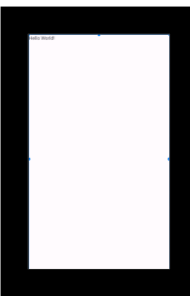
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"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



```
<TextView
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```



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

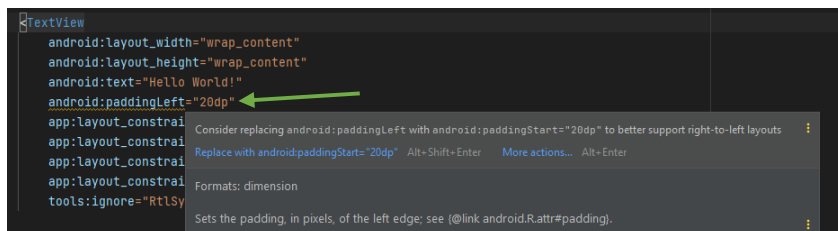


- **wrap_content** - dostosowuje wrapper do „jego zawartości/treści”
- **100dp** – nadaje bezwzględna wielkość dla wrappera
- **match_parent** – zajmuje całą szerokość „rodzica (parent)”

Ustawienie

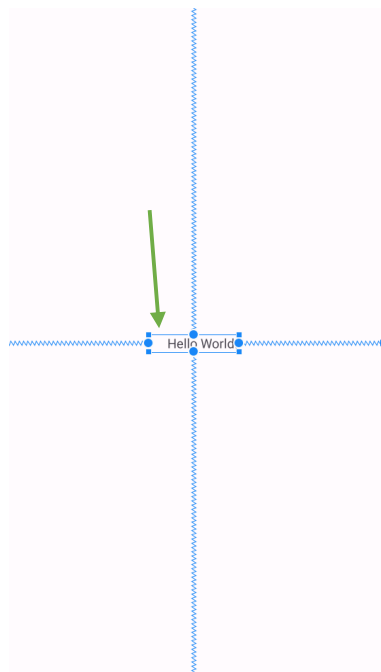
```
android:paddingLeft="20dp"
```

Spowoduje rozpoczęcie tekstu od wewnętrznego wcięcia marginesu z lewej strony o 20dp (**Density-independent Pixels** czyli jednostka niezależna od gęstości pikseli.)



Obecnie system Android Studio „doradza” używanie

```
android:paddingStart="20dp"
```



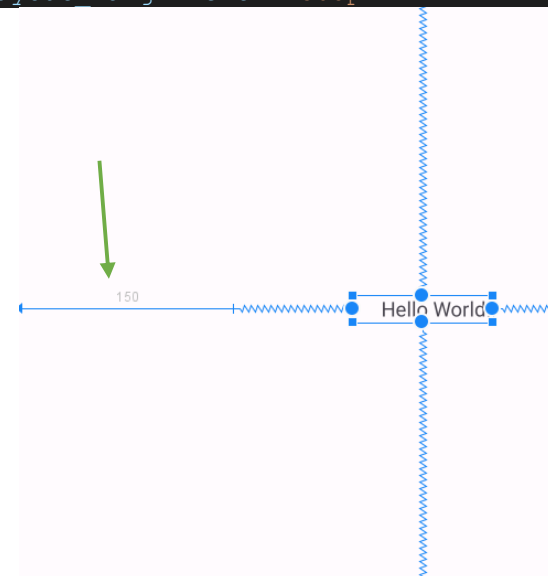
Analogicznie można określać resztę wewnętrznych stylów

```
padding
paddingBottom
paddingEnd
paddingHorizontal
paddingLeft
paddingRight
paddingTop
```

Padding – odległości od wewnętrznych ścian

Margin – odległość zewnętrznych ścian

```
android:layout_marginLeft="150dp"
```



Podobnie jak w padding właściwość margin posiada więcej ustawień

```
layout_margin  
layout_marginBottom  
layout_marginEnd  
layout_marginHorizontal  
layout_marginLeft  
layout_marginRight  
layout_marginStart  
layout_marginTop  
layout_marginVertical
```

Właściwość android:gravity

Android:gravity to atrybut używany w widokach Android, który kontroluje rozmieszczenie treści wewnątrz danego widoku. Określa on, w jaki sposób treść powinna być wyśrodkowana lub wyrównana wewnątrz widoku.

Atrybut ten przyjmuje różne wartości, które można łączyć za pomocą operatora | (bitowego OR) w celu uzyskania pożądanego efektu. Oto kilka przykładów najczęściej używanych wartości:

- center: Centruje treść wewnątrz widoku w pionie i poziomie.
- center_vertical: Centruje treść wewnątrz widoku tylko w pionie.
- center_horizontal: Centruje treść wewnątrz widoku tylko w poziomie.
- top: Wyrównuje treść do górnego brzegu widoku.
- bottom: Wyrównuje treść do dolnego brzegu widoku.
- left: Wyrównuje treść do lewego brzegu widoku.
- right: Wyrównuje treść do prawego brzegu widoku.

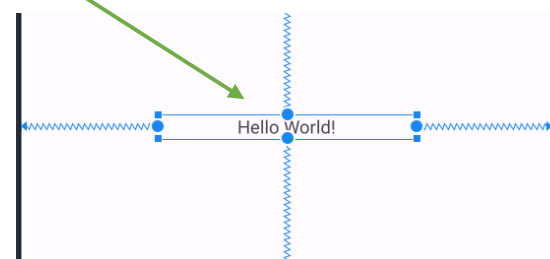
Można również zastosować kombinację tych wartości, na przykład center_vertical|right, aby wyśrodkować treść w pionie i wyrównać ją do prawego brzegu widoku.

Atrybut android:gravity jest często używany w połączeniu z android:layout_gravity, który kontroluje rozmieszczenie samego widoku wewnątrz jego rodzica.

Ustawienie

```
<TextView  
    android:layout_width="200dp"  
    android:layout_height="wrap_content"  
    android:text="Hello World!"  
    android:gravity="center"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent"  
    tools:ignore="RtlSymmetry" />
```

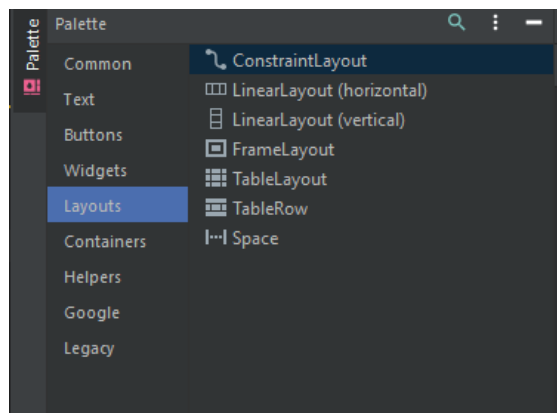
Spowoduje wyśrodkowanie napisu (trzeba pozbyć się marginesów aby zobaczyć efekt – tak jak w CSS)



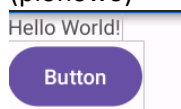
Layouts

Określają porządek rozmieszczenia widgetów na planszy (activity)

Na palecie należy przejść do sekcji Layouts



- RelativeLayout – opiera się o wzajemne zależności widgetów
- LinearLayout (vertical) – widgety będą układały się pod sobą (pionowo)



- LinearLayout (horizontal) – widgety będą układały się obok siebie (poziomo)



LinearLayout posiada wagi, które odpowiednio od swojej wartości skalują szerokość rozmieszczonych widgetów.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="horizontal"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="0dp"
        android:layout_weight="2"
        android:layout_height="wrap_content"
        android:text="Button" />

    <Button
        android:id="@+id/button2"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text="Button" />

    <Button
        android:id="@+id/button3"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text="Button" />

</LinearLayout>
```

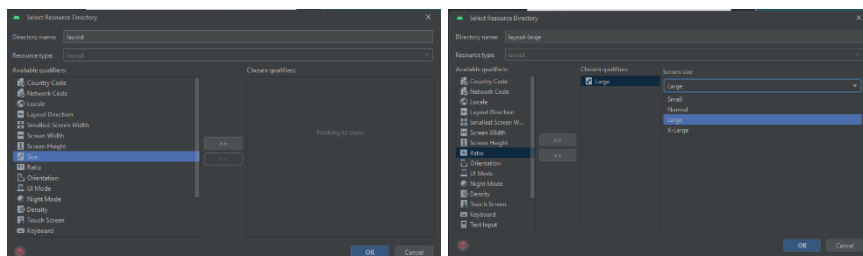
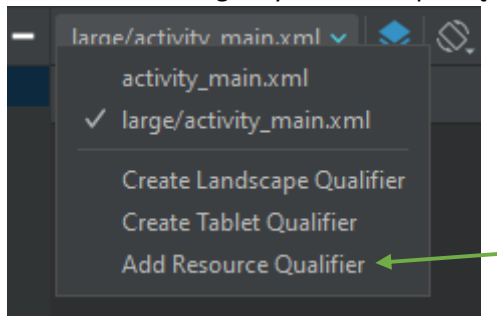
Odp zostaje ustawione dla szerokości na każdym przycisku (button)

```
android:layout_weight="2"
android:layout_weight="1"
android:layout_weight="1"
```

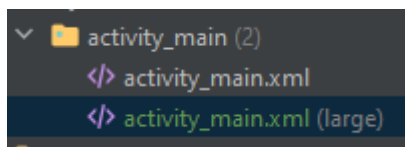
$2 + 1 + 1 = 4$ // Pierwszy przycisk zajmuje $\frac{2}{4}$ szerokości ekranu a przyciski drugi i trzeci po $\frac{1}{4}$ szerokości ekranu



Dodawanie nowego layout-u dla np. większych ekranów



Stosując to ustawienie zostanie stworzony layout dla większych ekranów
[activity-main.xml(large)]



W ten sposób należy tworzyć layouts dla szerokości ekranów.

- małe
- średnie
- duże

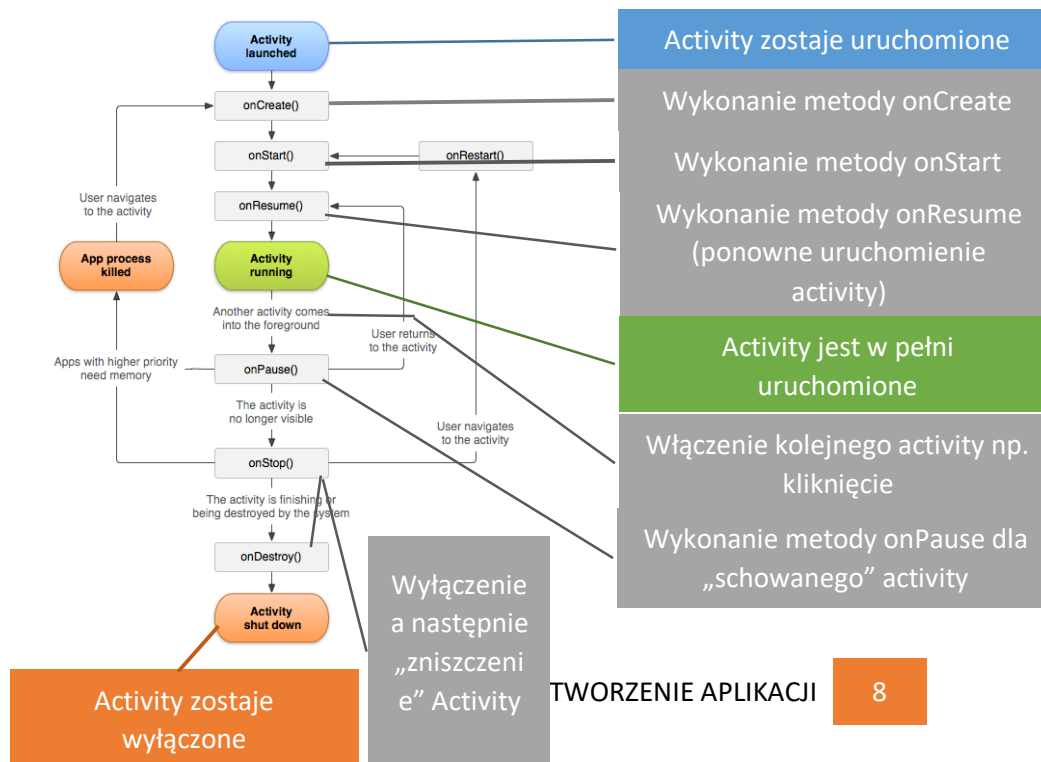
Widgety, Activity

Activity – jest jednym z kluczowych elementów wchodzących w skład budowy aplikacji na system Android.

Metoda – onCreate jest dziedziczona z klasy nadrzędnej Activity.
Uruchamiane jest podczas procesu tworzenia i generowania samego Activity (na samym początku cyklu życia)

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

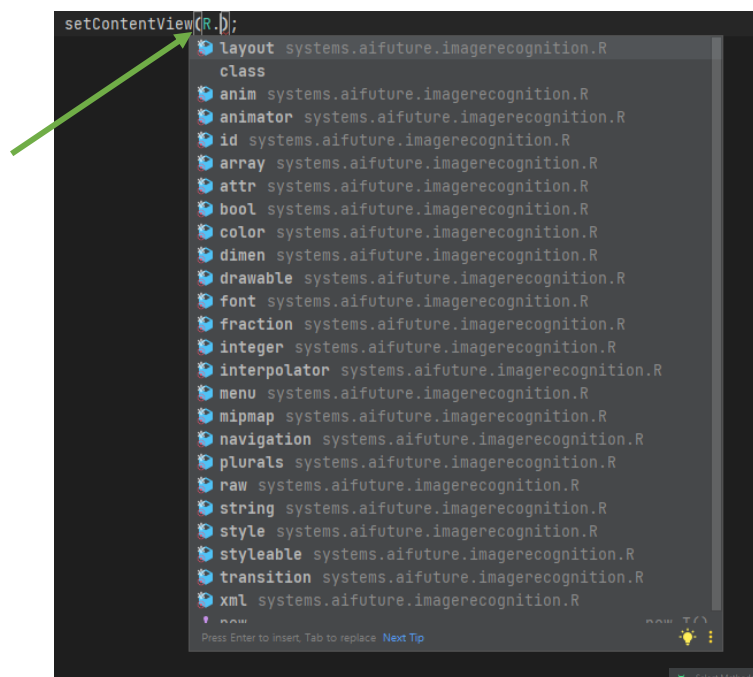
Diagram życia Activity / skopiowany z dokumentacji Androida /



Powiązanie activity „java - owego” z layoutem (np. activity_mian.xml)

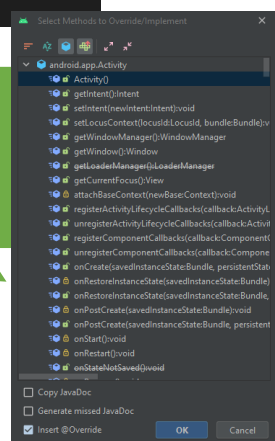
```
setContentView(R.layout.activity_main);
```

Klasa R nawiązuje do wszystkich resource aplikacji. Pozwala na odwoływanie się do animacji, kolorów do **id widgetów**, layoutów



W Android Studio skrót

CTRL + O otwiera kolekcję metod, które można dziedziczyć po klasie nadrzędnej

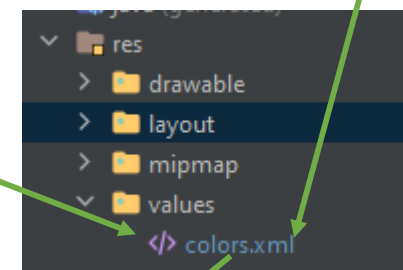


Dla widgetu TextView można ustawić kolor, czcionkę itd. (**podkreślenia czerwone** wynikają z ustawień języka w MSWord)

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="Hello World!"
    android:textSize="30sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0"
    android:textColor="@color/primary"
/>
```

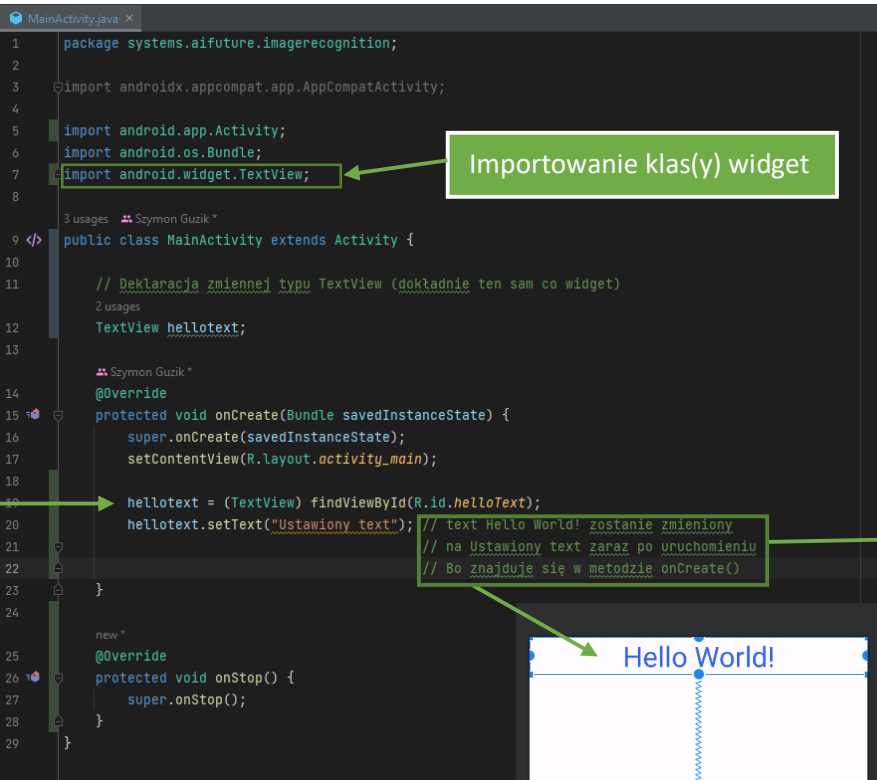
Kolor odwołuje się automatycznie do pliku w katalogu

- res
 - values
 - colors.xml



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <color name="black">#FF000000</color>
4   <color name="white">#FFFFFFF</color>
5   <color name="primary">#2E63FF</color>
6 </resources>
```

Odwoływanie się do elementów



The screenshot shows the `MainActivity.java` file. A green box highlights the import statement `import android.widget.TextView;` with the annotation "Importowanie klas(y) widget". Another green box highlights the line `hellotext = (TextView) findViewById(R.id.helloText);` with the annotation "Wyszukiwanie widgetu odbywa się za pomocą metody findViewById()". A third green box highlights the line `hellotext.setText("Ustawiony_text");` with the annotation "text Hello World! zostanie zmieniony na Ustawiony text zaraz po uruchomieniu // Bo znajduje się w metodzie onCreate()". Below the code, a green box contains the text "Wyszukiwanie widgetu odbywa się za pomocą metody findViewById() Np. `findViewById(R.id.helloText)`". To the right, a visual representation of the app's UI is shown as a white rectangle with a blue border and the text "Hello World!" at the top.

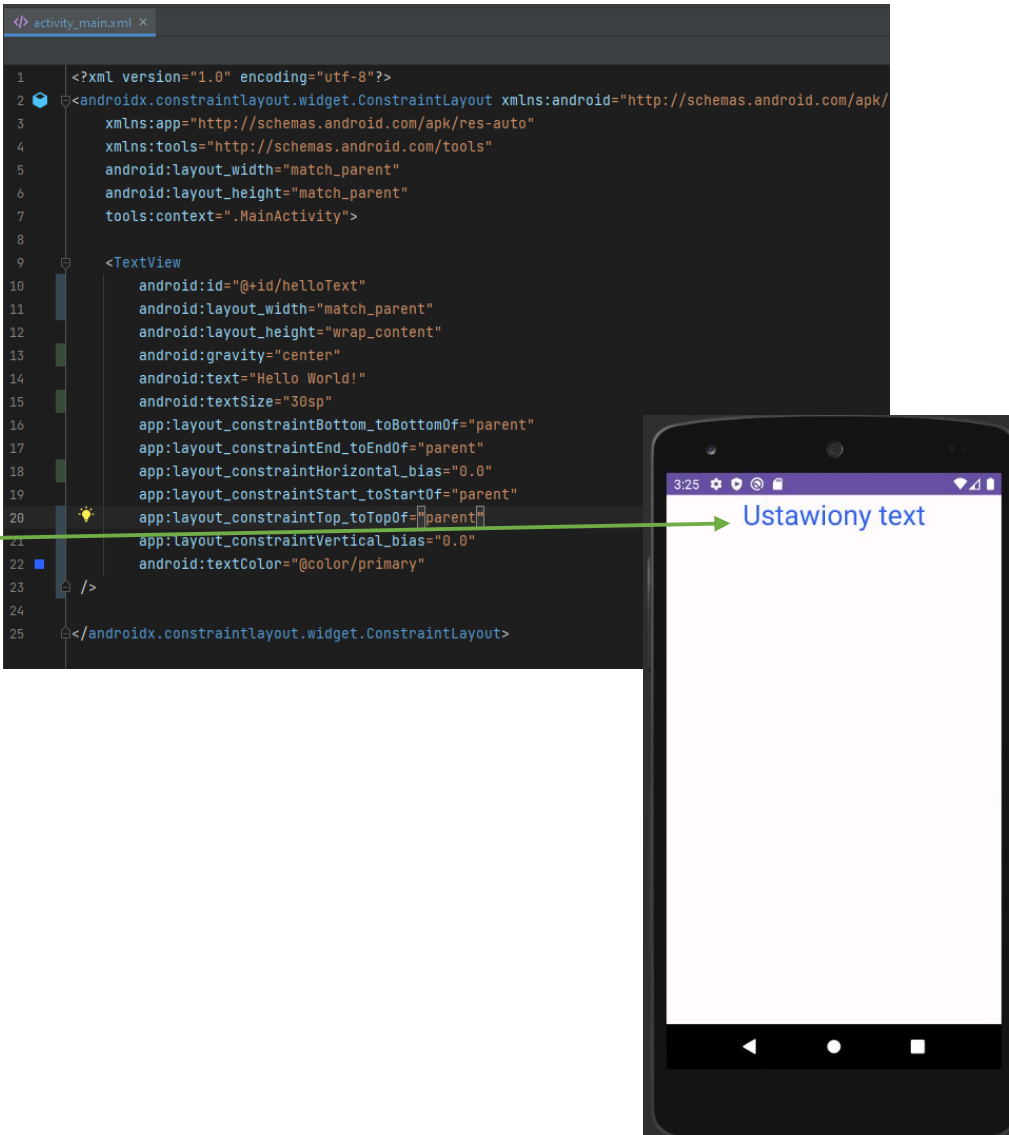
Importowanie klas(y) widget

Wyszukiwanie widgetu odbywa się za pomocą metody `findViewById()`

Np.

```
findViewById(R.id.helloText)
```

Hello World!



The screenshot shows the `activity_main.xml` file. The XML code defines a `TextView` with the text "Hello World!". A green arrow points from the `android:text="Hello World!"` attribute in the XML to a mobile phone mockup on the right. The mockup shows the text "Ustawiony text" on its screen, with the annotation "Ustawiony text" next to it.

Ustawiony text

Ustawianie nasłuchiwanie

Na kliknięcie

```

public class MainActivity extends Activity implements View.OnClickListener {

    // Deklaracja zmiennej typu TextView (dokładnie ten sam co widget)
    2 usages
    TextView helloText;
    3 usages
    Button buttonClick;

    Szymon Guzik *
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        helloText = (TextView) findViewById(R.id.helloText);
        helloText.setText("Ustawiony text"); // text Hello World! zostanie zmieniony
        // na Ustawiony text zaraz po uruchomieniu
        // Bo znajduje się w metodzie onCreate()

        buttonClick = (Button) findViewById(R.id.buttonClick);
        buttonClick.setText("Zmieniłem text");
        buttonClick.setOnClickListener(this); // Jako argument przyjmuje klasę, która implementuje
        // interfejs OnClickListener
        // Do tej klasy został dopisany taki interfejs więc
        // w nawiasie podajemy referencję do naszej klasy
        // this (referencja jest naszym argumentem
        // bo ta klasa ma implementację ww. interfejsu)
    }

    new *
    @Override
    protected void onStop() {
        super.onStop();
    }

    new *
    @Override
    public void onClick(View v) {
        Toast.makeText( context: this, text: "Kliknięto mnie", Toast.LENGTH_LONG).show();
        // Toast to krótka wiadomość w formie dymku na dole ekranu
        // Toast jako argument przyjmuje referencję do kontekstu (kontekst to jest activity)
        // Treść dymku w stringu
        // Długość trwania Toast (dymku) [Toast.LENGTH_SHORT, Toast.LENGTH_LONG]
    } // Metoda Show odpowiada za pokazanie dymku użytkownikowi
}

```

Implementacja OnClickListener wymusza metodę onClick, która zostaje wywołana za pomocą metody setOnClickListener

Podłączanie przycisków pod metodę i wybieranie „co ma się stać” po kliknięciu

```

public class MainActivity extends Activity implements View.OnClickListener {

    // Deklaracja zmiennej typu TextView (dokładnie ten sam co widget)
    2 usages
    TextView helloText;
    4 usages
    Button buttonClickLongToast;
    4 usages
    Button buttonClickShortToast;
}

```

Następnie w onCreate

```

buttonClickLongToast = (Button) findViewById(R.id.buttonClickLongToast);
buttonClickLongToast.setText("Pokaż długi Toast");
buttonClickLongToast.setOnClickListener(this);

```

```

buttonClickShortToast = (Button) findViewById(R.id.buttonClickShortToast);
buttonClickShortToast.setText("Pokaż krótki Toast");
buttonClickShortToast.setOnClickListener(this);

```

W metodzie onClick()

```

@Override
public void onClick(View v) {

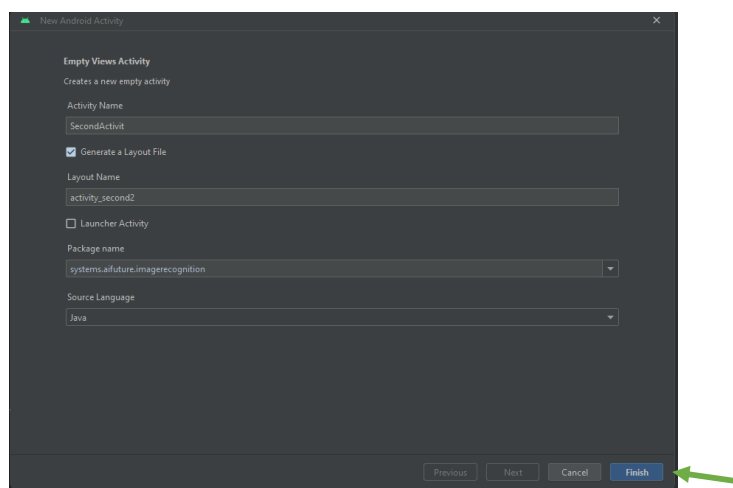
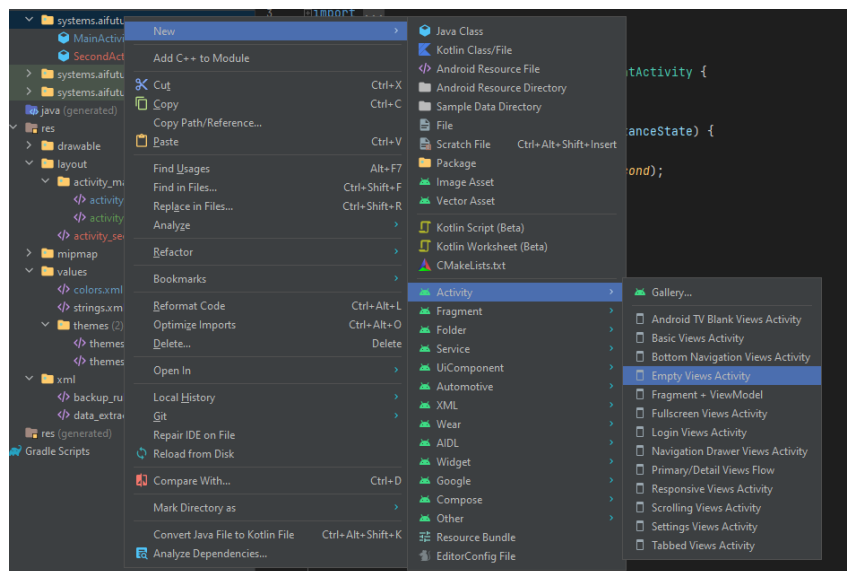
    if(v.getId() == buttonClickLongToast.getId()) {
        Toast.makeText( context: this, text: "Kliknięto Long", Toast.LENGTH_LONG).show();
    } else if (v.getId() == buttonClickShortToast.getId()) {
        Toast.makeText( context: this, text: "Kliknięto Short", Toast.LENGTH_SHORT).show();
    }
}

```

Metody analogicznie obsługiwane na innych Listenerach

Intenty

Tworzenie kolejnego Activity



SecondActivity.java

Tak wygląda kolejne Activity zaraz po utworzeniu

```

1 package systems.aifuture.imagerecognition;
2
3 import ...
4
5 2 usages
6
7
8 </> public class SecondActivity extends Activity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_second);
14     }
15 }
  
```

Przechodzenie pomiędzy Activity realizuje Intent

```

public class MainActivity extends Activity implements View.OnClickListener

// Deklaracja zmiennej typu TextView (dokładnie ten sam co widget)
4 usages
TextView helloText;
4 usages
Button buttonClickLongToast;
4 usages
Button buttonClickShortToast;

2 usages
Button buttonSecondActivity;
  
```

Deklaracja przycisku do przejścia (zmienna typu Button)

W metodzie onCreate tworzymy powiązanie oraz wywołanie `setOnClickListener`

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

    buttonSecondActivity = (Button) findViewById(R.id.buttonSecondActivity);
    new *
    buttonSecondActivity.setOnClickListener(new View.OnClickListener() {
        new *
        @Override
        public void onClick(View v) {
            Intent intent = new Intent( packageContext: MainActivity.this, SecondActivity.class);
            startActivity(intent);
        }
    });
}
```

Klasa Intent pozwala na przejścia pomiędzy Activity.

Ponieważ jesteśmy w klasie zagnieżdżonej to nie możemy odwołać się bezpośrednio do `this` ale możemy wskazując Nazwę klasy `this`

```
Intent intent = new Intent( packageContext: MainActivity.this, SecondActivity.class);
startActivity(intent);
```

- Pierwszy argument to klasa (Activity) z której wychodzimy
- Drugi argument to klasa (Activity) do którą chcemy otworzyć
- Wyjście z referencji obecnej czyli `this` do referencji docelowej `class`
- Wywołanie metody dziedziczonej `start_activity` z argumentem typu `Intent`

Analogicznie ustawienia powinny być zastosowane dla klasy drugiego Activity (SecondActivity.java)

```

1 package systems.aifuture.imagerecognition;
2
3 import ...
4
5
6
7
8
9
10
11 4 usages
12 <> public class SecondActivity extends Activity {
13
14
15
16 2 usages
17 Button buttonBackToFirstActivity;
18
19
20
21 @Override
22 protected void onCreate(Bundle savedInstanceState) {
23     super.onCreate(savedInstanceState);
24     setContentView(R.layout.activity_second);
25
26     buttonBackToFirstActivity = (Button) findViewById(R.id.buttonBackToFirstActivity);
27     buttonBackToFirstActivity.setOnClickListener(new View.OnClickListener() {
28         @Override
29         public void onClick(View v) {
30             Intent intent = new Intent(packageContext, SecondActivity.this, MainActivity.class);
31             startActivity(intent);
32         }
33     });
34 }

```

activity_second.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=".SecondActivity">

    <Button
        android:id="@+id/buttonBackToFirstActivity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Pomôc"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.898"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.023" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Są dwa rodzaje Intentów

Jawny	
W sposób wyraźny wskazane są dwie klasy	<code>Intent(SecondActivity.this, MainActivity.class);</code> Z jednego Activity do drugiego Activity (Sposób jawny - kolejny element ma być otwarty we wskazanym Activity – Android nie wybiera docelowego programu (automatycznie))
Niejawny	-----
Wskazać należy co trzeba zrobić ale bez podania „sposobu jak to zrobić”	<code>Intent(Intent.ACTION_VIEW, Uri.parse("http://wsb.pl"));</code> <code>startActivity(intent);</code> Android „sam”(automatycznie) wybierze program do przeglądania w tym przypadku WWW

```

public class MainActivity extends Activity {
    // Deklaracja zmiennej typu TextView (dokładnie ten sam co widget)

    2 usages
    Button wwwButton;

    "Szymon Guzik"
    @SuppressWarnings("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        wwwButton = (Button) findViewById(R.id.buttonWSB);
        new "
        wwwButton.setOnClickListener(new View.OnClickListener() {
            new "
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://wsb.pl"));
                startActivity(intent);
            }
        });
    }
}

```

Poniżej kod implementujący **Niejawny** sposób (Jawny jest pokazany wyżej)

Przesyłanie danych pomiędzy Activity

```

@Override
public void onClick(View v) {
    // Przechodzenie w sposób jawny do innego Activity
    Intent intent = new Intent( packageContext MainActivity.this, SecondActivity.class);
    // Przekazanie danych do innego activity (klucz -> wartość)
    intent.putExtra( name: "name", value: "Szymon Guzik");
    startActivity(intent);
}

```

Pobieranie danych (Przypisanie do zmiennej String więc na `getIntent()` jest wywołana metoda do zwracania zmiennej ze stringiem `getStringExtra()`).

Na samym końcu jest deklaracja krótkiego Toast (drugi argument to **odczytana zmienna**) wyświetlanego zaraz po przejściu na wybrane Activity

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_second);
    // Pobieranie danych przekazanych przez Intent
    String name = getIntent().getStringExtra( name: "name");
    Toast.makeText( context: this, name, Toast.LENGTH_SHORT).show();
    buttonBackToFirstActivity = (Button) findViewById(R.id.buttonBackToFirstActivity);
    buttonBackToFirstActivity.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent( packageContext: SecondActivity.this, MainActivity.class);
            startActivity(intent);
        }
    });
}

```

