



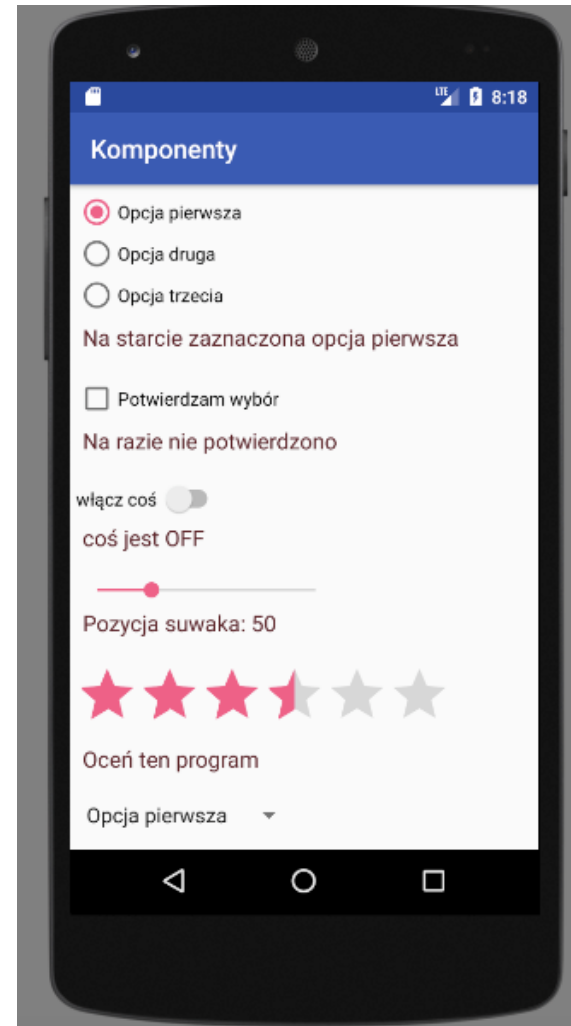
PROGRAMOWANIE APLIKACJI MOBILNYCH

Wykład

dr Artur Bartoszewski

Kontrolki systemu

Android udostępnia kilka standardowych kontrolek



Suwak - SeekBar



Suwak posiada własności:

- **progres** odpowiadającą za aktualną wartość,
- **max i min** wyznaczające jej rozpiętość.

```
<SeekBar  
  android:id="@+id/seekBar01"  
  android:layout_width="0dp"  
  android:layout_height="wrap_content"  
  app:layout_constraintLeft_toLeftOf="parent"  
  app:layout_constraintRight_toRightOf="parent"  
  app:layout_constraintTop_toBottomOf="@id/textView01"  
  android:min="0"  
  android:max="200"  
  android:progress="30"  
>
```

Suwak - SeekBar

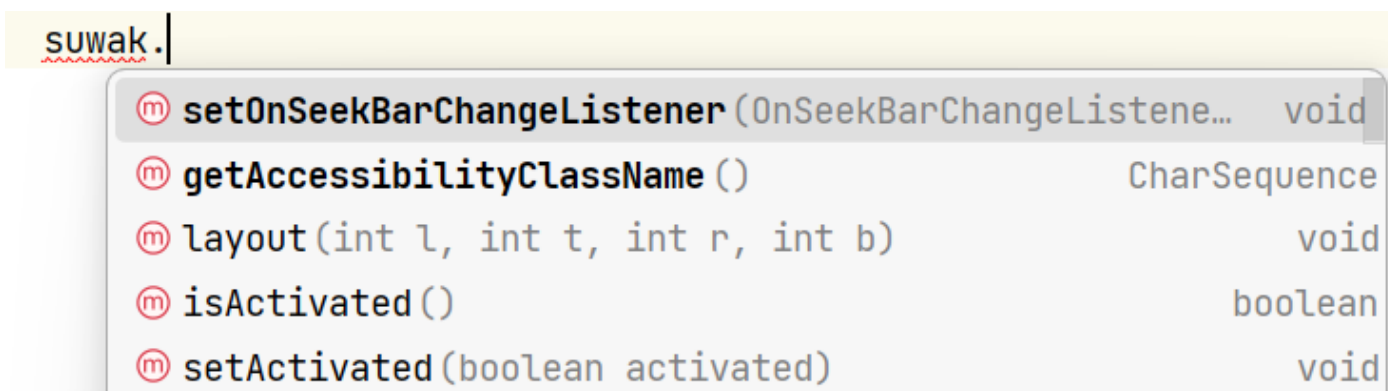


Pierwszym krokiem, jak w przypadku, każdej kontrolki jest stworzenie jej referencji.

```
SeekBar suwak;  
suwak = findViewById(R.id.seekBar01);
```

Następnie należy dodać słuchacza zdarzeń.

Używamy dedykowanego dla suwaka ka słuchacza `OnSeekBarChangeListener()`



Suwak - SeekBar



Słuchacz posiada 3 metody obsługi zdarzeń

```
suwak.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
    @Override  
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
        // metoda wywoływana jest cyklicznie w trakcie przesuwania suwaka.  
        // Aktualna wartość suwaka dostępna jest poprzez parametr „progress”  
    }  
    @Override  
    public void onStartTrackingTouch(SeekBar seekBar) {  
        // metoda wywoływana jest jeden raz, przy rozpoczęciu przesuwania suwaka  
    }  
    @Override  
    public void onStopTrackingTouch(SeekBar seekBar) {  
        // metoda wywoływana jest jeden raz, po zakończeniu przesuwania suwaka  
    }  
});
```

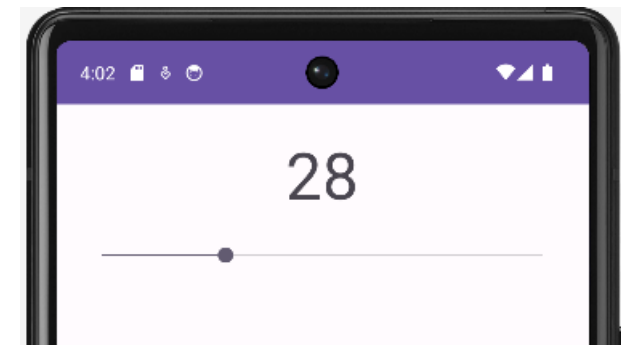
Suwak - SeekBar



Przykład:

- W trakcie przesuwania suwaka jego wartość będzie wypisywana w polu tekstowym.
- Przy rozpoczęciu i zakończeniu ruchu wywołane zostaną okienka komunikatów

```
suwak.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
    @Override  
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
        wynik.setText(String.valueOf(progress));  
    }  
    @Override  
    public void onStartTrackingTouch(SeekBar seekBar) {  
        Toast.makeText(MainActivity.this, "Rozpoczęto suwanie", Toast.LENGTH_SHORT).show();  
    }  
    @Override  
    public void onStopTrackingTouch(SeekBar seekBar) {  
        Toast.makeText(MainActivity.this, "Zakończono suwanie", Toast.LENGTH_SHORT).show();  
    }  
});
```



Suwak – Slider



Często używana jest także druga wersja suwaka –Slider

Różnice:

- wyświetlanie dymku z aktualną wartością w trakcie przesuwania
- możliwość ustawienia kroku ułamkowego (stepSize)
- inne nazewnictwo
 - value – aktualna wartość,
 - valueFrom - minimum,
 - valueTo - maksimum

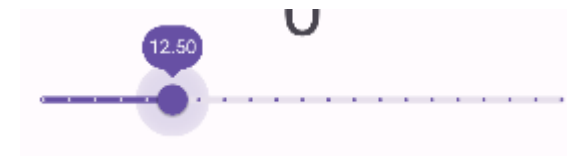


```
<com.google.android.material.slider.Slider  
    android:id="@+id/slider01"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:valueFrom="10"  
    android:valueTo="20"  
    android:value="15"  
    android:stepSize="0.5"  
>
```

Suwak - Slider



Dedykowane dla niego słuchacz zdarzeń nie posiada metody która wykonywana jest cyklicznie w trakcie przesuwania.



Slider suwak;

```
suwak = findViewById(R.id.slider01);
```

```
suwak.addOnSliderTouchListener(new Slider.OnSliderTouchListener() {
```

```
    @Override
```

```
    public void onStartTrackingTouch(@NonNull Slider slider) {
```

```
        // metoda uruchamiana na starcie
```

```
    }
```

```
    @Override
```

```
    public void onStopTrackingTouch(@NonNull Slider slider) {
```

```
        // metoda uruchamiana po zakończeniu gestu
```

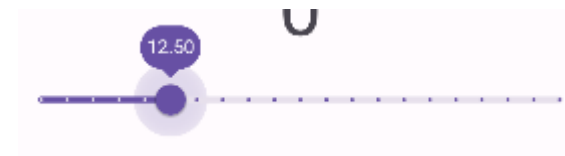
```
    }
```

```
});
```


Suwak - Slider



Wartość suwaka jest typu zmiennoprzecinkowego (float) i należy ją pobrać przy pomocy metody `getValue()`

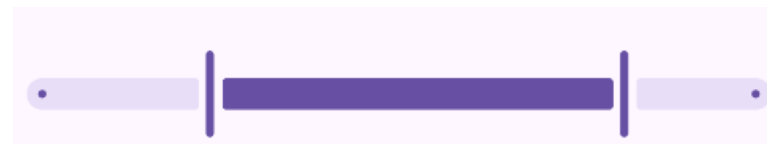


```
suwak.addOnSliderTouchListener(new Slider.OnSliderTouchListener() {  
    @Override  
    public void onStartTrackingTouch(@NonNull Slider slider) {  
        wynik.setText(String.valueOf(suwak.getValue()));  
        // metoda uruchamiana na starcie  
    }  
    @Override  
    public void onStopTrackingTouch(@NonNull Slider slider) {  
        wynik.setText(String.valueOf(suwak.getValue()));  
        // metoda uruchamiana po zakończeniu gestu  
    }  
});
```

Suwak - RangeSlider



Kontrolka `com.google.android.material.slider.RangeSlider` służy do wybierania zakresu wartości z ciągłego przedziału. Użytkownik może interaktywnie przesunąć dwa uchwyty na suwaku, aby określić minimalną i maksymalną wartość w wybranym zakresie.



Kontrolka ta zwraca wynik zapisany jako dwu elementowa tablica liczb typów `float`.

Aby zdefiniować początkową pozycję suwaków należy przygotować taką tablicę i umieścić ją w zasobach aplikacji.

Utwórz plik `res/values/arrays.xml` i zdefiniuj początkowe wartości:

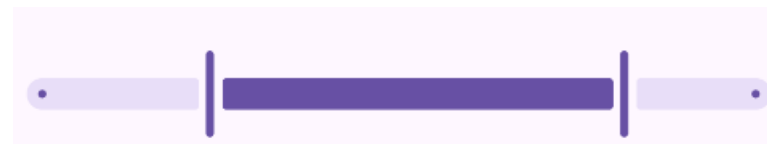
```
<array name="initial_slider_values">  
  <item>25.0</item>  
  <item>75.0</item>  
</array>
```

Nazwa tablicy oraz
wartości przykładowe.

Suwak - RangeSlider



Kontrolka `com.google.android.material.slider.RangeSlider` służy do wybierania zakresu wartości z ciągłego przedziału. Użytkownik może interaktywnie przesunąć dwa uchwyty na suwaku, aby określić minimalną i maksymalną wartość w wybranym zakresie.



```
<com.google.android.material.slider.RangeSlider
    android:id="@+id/slaidler"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    android:valueFrom="0.0"
    android:valueTo="100.0"
    app:values="@array/initial_slider_values"
/>
```

Zakres suwaka

Wartości początkowe - zapisane jako tablica

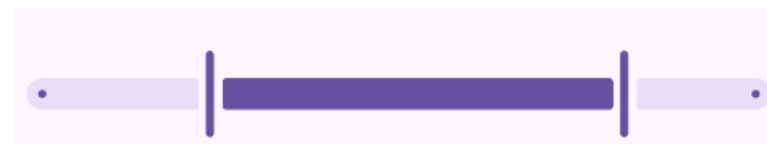
Suwak - RangeSlider



W kodzie java aktualne wartości suwaka odczytać możemy za pomocą metod:

```
.getValues().get(0)  
.getValues().get(1)
```

Można je umieścić w słuchaczu zdarzeń reagującym na zmiany wartości suwaka



```
com.google.android.material.slider.RangeSlider slider = findViewById(R.id.slaider);
```

```
slider.addOnChangeListener(new RangeSlider.OnChangeListener() {  
    @Override  
    public void onValueChange(@NonNull RangeSlider slider, float value, boolean fromUser) {  
        textView.setText("Od: " + slider.getValues().get(0) + "\nDo: " + slider.getValues().get(1));  
    }  
});
```



RatingBar

```
9  <RatingBar
10      android:id="@+id/ratingBar01"
11      android:layout_width="wrap_content"
12      app:layout_constraintTop_toTopOf="parent"
13      android:layout_height="wrap_content"
14      app:layout_constraintLeft_toLeftOf="parent"
15      app:layout_constraintRight_toRightOf="parent"
16      android:numStars="7"
17      android:rating="3.5"
18  />
```

Właściwości:

numStars – maksymalna liczba gwiazdek

rating – aktualna ocena (wyświetlana z dokładnością do pół gwiazdki)

Pole wyboru - CheckBox

W pliku XML – tworzymy CheckBox.



Opis pola wyboru

```
CheckBox checkBox01;  
checkBox01 = findViewById(R.id.checkBox01);  
    android:text="Opis pola wyboru"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:checked="true"  
/>
```

W pliku JAVA – rozpoczynamy od znalezienie referencji do kontrolki

```
CheckBox checkBox01;  
checkBox01 = findViewById(R.id.checkBox01);
```

Pole wyboru - CheckBox

Kontrolkę CheckBox można obsługiwać z zewnątrz.
To znaczy, że jeśli mamy referencję do kontrolki (jej uchwyt) możemy w każdej chwili sprawdzić jej aktualny stan (czy jest wybrana)



Opis pola wyboru

```
if(checkBox01.isChecked())  
    // akcja jeżeli jest wybrany  
else  
    //akcja jeżeli nie jest wybrany
```

W takiej sytuacji reakcja następuje nie po kliknięciu na CheckBox, a dopiero po sprawdzeniu jego stanu.

Na przykład po wybraniu wszystkich pól formularza i kliknięciu na przycisk „Wyślij”

Pole wyboru - CheckBox

Kontrolce CheckBox można też przypisać słuchacza zdarzenia.



Opis pola wyboru

```
checkBox01.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  
    @Override  
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {  
        if(isChecked)  
            // akcja jeżeli jest wybrany  
        else  
            //akcja jeżeli nie jest wybrany  
    }  
});
```

W takiej sytuacji reakcja następuje natychmiast po kliknięciu na CheckBox

W takiej sytuacji reakcja następuje natychmiast po kliknięciu na CheckBox

Przełącznik - Switch

<Switch

```
    android:id="@+id/switch01"  
    android:text="Przełącznik"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:checked="true"/>
```

Przełącznik



Przełącznik



<com.google.android.material.switchmaterial.SwitchMaterial

```
    android:id="@+id/switch01"  
    android:text="Przełącznik"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
/>
```

Przełącznik



Przełącznik



W takiej sytuacji reakcja następuje natychmiast po kliknięciu na CheckBox

Przełącznik - Switch

Podobnie jak CheckBox przełącznik możemy obsługiwać z zewnątrz.


```
Switch switch01;
switch01 = findViewById(R.id.switch01);
```


```
if (switch01.isChecked())
```

```
    //akcja jeżeli jest włączony
```

```
else
```

```
    //akcja jeżeli jest wyłączony
```

Przełącznik 

Przełącznik 

```
switch01.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
```

```
    @Override
```

```
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
```

```
        if (isChecked)
```


```
            //akcja jeżeli jest włączony
```


```
        else
```

```
            //akcja jeżeli jest wyłączony
```

```
    }
```

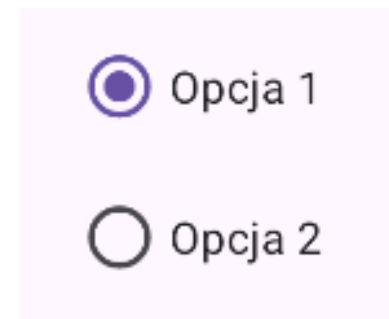
```
});
```

Przełącznik 

Przełącznik 

Lub dodać do niego słuchacza zdarzeń

- ✓ Kontrolki **RadioButton** są używane, gdy użytkownik ma do wyboru jedną opcję z grupy wzajemnie wykluczających się opcji.
- ✓ **RadioGroup** jest kontenerem dla grupy kontrolek RadioButton, zapewniającym, że tylko jeden przycisk radiowy może być wybrany w danym czasie.
- ✓ Pozycjonujemy kontener RadioGroup. Przyciski są automatycznie rozkładane wewnątrz niego.
- ✓ Może istnieć kilka niezależnych tych od siebie kontenerów - grup przycisków.



RadioGroup i RadioButton

```
<RadioGroup
  android:id="@+id/radioGroup"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:orientation="vertical">

  <RadioButton
    android:id="@+id/radioButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Opcja 1,"
    android:checked="true"/>

  <RadioButton
    android:id="@+id/radioButton2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Opcja 2" />

</RadioGroup>
```

ID nadajemy zarówno kontenerowi, jak i każdej z kontrolek.

orientation: Określa orientację kontrolek wewnątrz RadioGroup. Może przyjmować wartość vertical lub horizontal.

text: Tekst wyświetlany obok RadioButton

checked: Kontrolka „wybrana”. Tylko jedna kontrolka w grupie może mieć opcję checked ustawioną na true

☒ Opcja 1

☐ Opcja 2

```
RadioButton radioButton1 = findViewById(R.id.radioButton1);  
RadioButton radioButton2 = findViewById(R.id.radioButton2);  
RadioGroup radioGroup = findViewById(R.id.radioGroup);
```

Obsługę RadioButton-ów w kodzie Java rozpoczynamy od znalezienia uchwytów (referencji) do kontrolki oraz do kontenera

Słuchacza zdarzeń dodajemy dla całego kontenera.

```
radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {  
    @Override  
    public void onCheckedChanged(RadioGroup group, int checkedId) {  
        if(checkedId == R.id.radioButton1){  
            //akcja po wybraniu opcji 1  
        }  
        else if(checkedId == R.id.radioButton2) {  
            //akcja po wybraniu opcji 2  
        }  
    }  
});
```

Słuchacz reaguje na zmianę kontrolki. Nie reaguje na ponowne kliknięcie w już wybraną kontrolkę.

Jednym z jego parametrów jest ID kontrolki która została wybrana.

Na podstawie ID możemy rozpoznać który RadioButton został wybrany i zaimplementować odpowiednią akcję

```
RadioButton radioButton1 = findViewById(R.id.radioButton1);  
RadioButton radioButton2 = findViewById(R.id.radioButton2);
```

Stan RadioButton-ów można sprawdzić w dowolnym miejscu kodu java. Jeżeli tylko posiadamy dostęp do uchwytów kontrolerek

```
if(radioButton1.isChecked())  
    textView.setText("Wybrano: radioButton1");  
else if(radioButton2.isChecked())  
    textView.setText("Wybrano: radioButton2");
```

Metoda `.isChecked()` zwraca wartość prawda lub fałsz w zależności od tego czy kontrolka jest wybrana czy nie.

W ten sposób można sprawdzić która kontrolka jest wybrana w dowolnym miejscu programu, często w słuchaczu zdarzeń innej kontrolki lub zdarzenia cyklu życia aplikacji

```

<?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:id="@+id/main"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity"
android:padding="20dp">

<TextView
    android:id="@+id/opis"
    android:textSize="20sp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Wybrana kontrolka: "
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
<RadioGroup
    android:id="@+id/radioGroup"
    app:layout_constraintTop_toBottomOf="@id/opis"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginTop="20dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton
        android:id="@+id/radioButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Opcja 1"
        android:checked="true"/>
    <RadioButton
        android:id="@+id/radioButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Opcja 2" />
</RadioGroup>

</androidx.constraintlayout.widget.ConstraintLayout>

```

Przykład:

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
        TextView textView = findViewById(R.id.opis);
        RadioButton radioButton1 = findViewById(R.id.radioButton1);
        RadioButton radioButton2 = findViewById(R.id.radioButton2);
        RadioGroup radioGroup = findViewById(R.id.radioGroup);

        if(radioButton1.isChecked())
            textView.setText("Wybrano: radioButton1");
        else if(radioButton2.isChecked())
            textView.setText("Wybrano: radioButton2");

        radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                if(checkedId == R.id.radioButton1){
                    Toast.makeText(MainActivity.this, "Wybrano: radioButton1", Toast.LENGTH_SHORT).show();
                    textView.setText("Wybrano: radioButton1");
                }
                else if(checkedId == R.id.radioButton2)
                {
                    Toast.makeText(MainActivity.this, "Wybrano: radioButton2", Toast.LENGTH_SHORT).show();
                    textView.setText("Wybrano: radioButton2");
                }
            }
        });
    }
}

```



Wybrano: radioButton1

☒ Opcja 1

☐ Opcja 2

Komponent ImageView w Androidzie służy do wyświetlania obrazów w aplikacji.

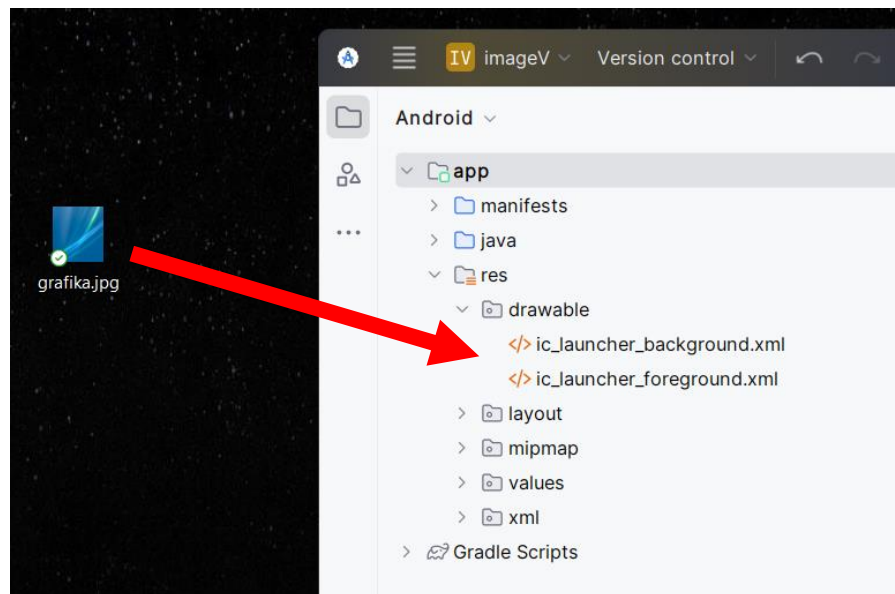


Można użyć go do wyświetlania obrazów ze źródeł, takich jak:

- **Pliki lokalne:** obrazy przechowywane w zasobach aplikacji (np. w folderze drawable).
- **Zasoby sieciowe:** obrazy pobierane z Internetu za pomocą adresu URL.
- **Bitmapy:** obrazy generowane dynamicznie w kodzie.
- **Inne źródła:** obrazy pochodzące z aparatu, galerii, czy innych źródeł danych.

Dodatkowe możliwości komponentu ImageView:

- **Skalowanie:** można kontrolować sposób skalowania obrazu, aby dopasować go do rozmiaru ImageView (np. centerCrop, fitCenter, matrix).
- **Kadrowanie:** można przyciąć obraz, aby wyświetlić tylko jego fragment.
- **Obrót:** można obracać obraz o określony kąt.
- **Dodawanie filtrów:** można stosować filtry do obrazu, aby zmienić jego wygląd.



Najprostszym sposobem mu życia kontrolki ImageView jest umieszczenie w niej plików graficznych przechowywanych w zasobach aplikacji.

Pracę rozpoczynamy od umieszczenie grafik folderze „drawable” (np. metodą przeciągnij i upuść)

Należy uważać na ograniczenia nazw plików graficznych

Niektóre ograniczenia wynikają ze sposobu budowania projektu Androida - np. nazwa pliku nie może zaczynać się od cyfry ani dużej litery

Należy też pamiętać, że Android pracuje pod kontrolą jądra systemu Linux, które inaczej przetwarza niektóre znaki (spacje czy polskie znaki diakrytyczne). Ich także należy unikać.

```
<ImageView  
    android:id="@+id/imageView01"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:src="@drawable/grafika,"  
    android:scaleType="fitCenter"  
/>
```

Kontrolkę ImageView można zpozycjonować tak jak każdy inny widok layoutu.

Własność `scaleType` pozwala wybrać sposób skalowania grafiki wewnątrz kontrolki.

Porady:

- Zwykle definiujemy rozmiar kontrolki i do niego skalujemy grafikę. Można stworzyć kontrolkę, której rozmiar dopasuje się do załadowanej grafiki, lecz w większości przypadków nie jest to polecane.
- Nie używajcie `scaleType="fitXY"` (rozciągnięcie grafiki na cały dostępny obszar bez zachowania proporcji). Chyba nic nie wygląda bardziej nieprofesjonalnie...

Grafikę można załadować także z poziomu kodu Java.

```
ImageView imageView = findViewById(R.id.imageView01);
```

Potrzebna jest referencja do kontrolki.

```
imageView.setImageResource(R.drawable.grafika);
```

Jeżeli grafika znajduje się w zasobach aplikacji użyć można metody `.setImageResource()`

Jeżeli grafika zapisana jest w obiekcie typu `drawable` użyć można metody `.setImageDrawable()`



```
<ImageButton  
    android:id="@+id/imageButton01"  
    android:layout_width="120dp"  
    android:layout_height="60dp"  
    android:src="@drawable/artr_b_logo"  
    android:scaleType="fitCenter" />
```

Z poziomu kodu ImageButton obsługujemy tak jak zwykły przycisk - za pomocą słuchacza zdarzeń `OnClickListener()`

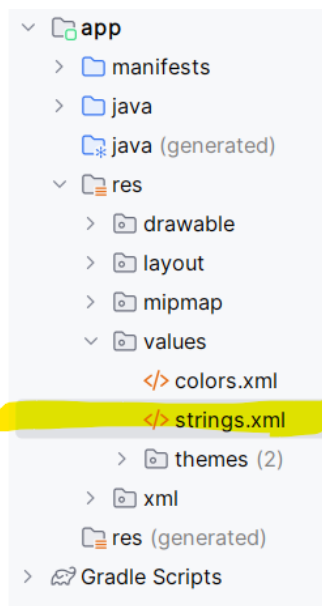
Kontrolka ImageButton to przycisk, który wyświetla obraz zamiast tekstu.

Użytkownik może na niego kliknąć, aby wywołać określoną akcję.

Główne zastosowania ImageButton:

- Ikony akcji: np. wyświetlanie ikon reprezentujących różne akcje, takie jak odtwarzanie, pauza, udostępnianie, dodawanie do ulubionych itp.
- Nawigacja: tworzenie przycisków nawigacyjnych, np. strzałek do przewijania galerii obrazów.
- Interaktywne elementy: ImageButton może być używany jako element interaktywny w grach lub innych aplikacjach, gdzie obraz lepiej oddaje funkcjonalność niż tekst.

Tworzenie i wypełnianie danymi menu rozwijanego z poziomu kodu XML



```
<string-array name="pozycje_menu">  
    <item>Pozycja 01</item>  
    <item>Pozycja 02</item>  
    <item>Pozycja 03</item>  
</string-array>
```

W zasobach aplikacji – strings.xml –
tworzymy tablicę zawierającą wszystkie
pozycje, które mają znaleźć się w menu.

```
<string name="prompt">  
    Wybierz właściwą opcję:  
</string>
```

Opcjonalnie możemy też dodać łańcuch,
który wykorzystany będzie jako
objaśnienie (tytuł) menu.

Aplikacje mobilne - Spinner – menu rozwijane



Tworzenie i wypełnianie danymi menu rozwijanego z poziomu kodu XML

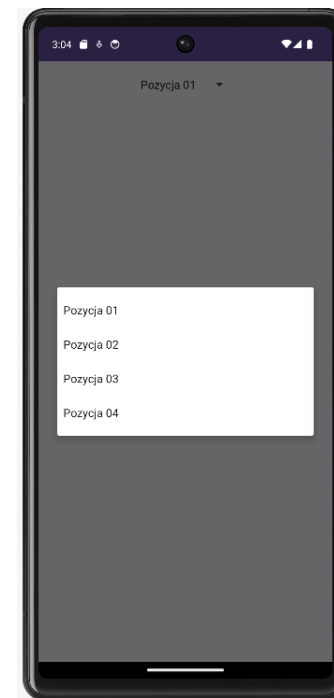
W Pliku XML definiujemy wygląd i pozycję spinnera

```
<Spinner  
    android:id="@+id/spinner01"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:entries="@array/pozycje_menu"  
    android:spinnerMode="dialog"  
    android:tooltipText="@string/prompt"  
/>
```

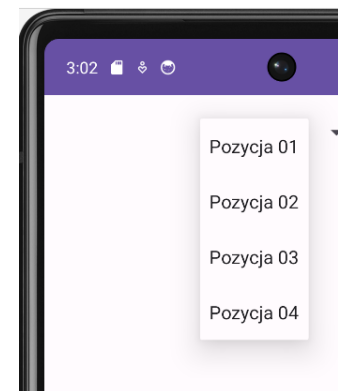
Własność entries wypełnia menu danymi z tablicy przygotowanej w zasobach aplikacji

Własność spinnerMode pozwala wybrać jeden z dwóch sposobów działania menu

Pozwala ustawić podpowiedź (opis) do menu. Działa tylko w trybie „dialog”



`android:spinnerMode="dialog"`



`android:spinnerMode="dropdown"`

Tworzenie i wypełnianie danymi menu rozwijanego z poziomu kodu **JAVA**

```
String[] pozycje_w_menu = {"Pierwsza", "Druga", "Trzecia", "Czwarta" };  
Spinner spinner01 = findViewById(R.id.spinner01);  
ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_spinner_item, pozycje_w_menu);  
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);  
spinner01.setAdapter(adapter);
```

Tworzenie i wypełnianie danymi menu rozwijanego z poziomu kodu **JAVA**

1. ***String[] pozycje_w_menu = {"Pierwsza", "Druga", "Trzecia", "Czwarta"};*** - tworzy tablicę tekstów, które będą wyświetlane jako pozycje w rozwijanej liście Spinner.
2. ***Spinner spinner01 = findViewById(R.id.spinner01);*** - pobiera referencję do kontrolki Spinner z Twojego layoutu XML, używając jej identyfikatora R.id.spinner01.
3. ***ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_spinner_item, pozycje_w_menu);*** tworzy obiekt ArrayAdapter, który będzie odpowiedzialny za dostarczanie danych do Spinnera
 - Pierwszy argument (this) odnosi się do kontekstu aktywności.
 - Drugi argument (android.R.layout.simple_spinner_item) to layout, który będzie używany do wyświetlania każdej pozycji w liście.
 - Trzeci argument (pozycje_w_menu) to tablica tekstów, które będą wyświetlane w Spinner.
4. ***adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);*** ustawia layout, który będzie używany do wyświetlania listy rozwijanej Spinner.
5. ***spinner01.setAdapter(adapter);*** przypisuje utworzony adapter do kontrolki Spinner, co powoduje, że pozycje z tablicy pozycje_w_menu zostaną wyświetlone w liście.

Aplikacje mobilne - Spinner – menu rozwijane

Reakcja na wybór opcji menu spinner

```
spinner01.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
    @Override  
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {  
        switch (position)  
        {  
            case 0: //Akcja, jeżeli wybrano opcję pierwszą;  
                break;  
            case 1: //Akcja, jeżeli wybrano opcję drugą;  
                break;  
            case 2: // Akcja, jeżeli wybrano opcję trzecią;  
                break;  
        }  
    }  
    @Override  
    public void onNothingSelected(AdapterView<?> parent) {  
        //akcje, jeżeli żadna z pozycji jest wybrana  
    }  
});
```

Menu Spinner obsługujemy za pomocą słuchacza zdarzeń **OnItemSelectedListener()**

Parametr **position** to indeks wybranej opcji (począwszy od 0)

Aplikacje mobilne - Spinner – menu rozwijane

Pozycję aktualnie wybraną w menu typu Spinner możemy odczytać także spoza słuchacza zdarzeń służą do tego metody:

- `spinner01.getSelectedItem();` - Zwraca tekst aktualnie wybranej pozycji z menu
- `spinner01.getSelectedItemId();` - Zwraca ID aktualnie wybranej pozycji z menu

Przykład:

```
String[] pozycje_w_menu = {"Pierwsza", "Druga", "Trzecia", "Czwarta" };
Spinner spinner01 = findViewById(R.id.spinner01);
ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_spinner_item, pozycje_w_menu);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner01.setAdapter(adapter);
spinner01.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        String Odpowiedz="";
        switch (position)
        {
            case 0: Odpowiedz="Wybrano pozycję Pierwszą"; break;
            case 1: Odpowiedz="Wybrano pozycję Drugą"; break;
            case 2: Odpowiedz="Wybrano pozycję Trzecią"; break;
            case 3: Odpowiedz="Wybrano pozycję Czwartą"; break;
        }
        Toast.makeText(getApplicationContext(), Odpowiedz, Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onNothingSelected(AdapterView<?> parent) {
    }
});
```



RatingBar

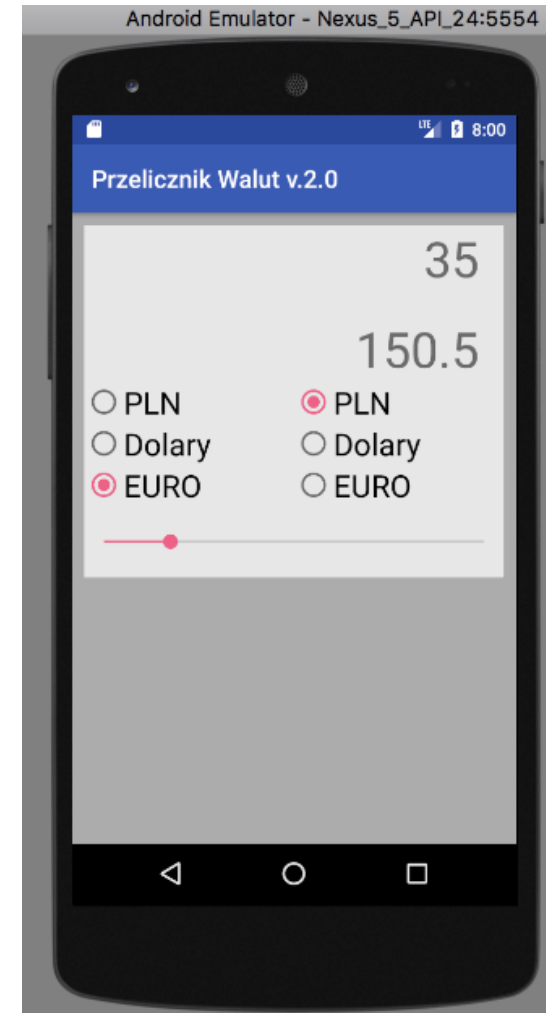
```
final RatingBar gwiazdki01 = findViewById(R.id.ratingBar01);
gwiazdki01.setOnRatingBarChangeListener(new RatingBar.OnRatingBarChangeListener() {
    @Override
    public void onRatingChanged(RatingBar ratingBar, float rating, boolean fromUser) {
        //Akcja po zmianie "ratingu"
    }
});
```

Obsługa podobnie jak SeekBar

Do odczytania wartości poza słuchaczem zdarzeń służy metoda **getRating()**

ZADANIE PRAKTYCZNE:

Przelicznik walut obsługiwany bez pomocy klawiatury ekranowej



```

13 <android.support.constraint.Guideline
14     android:layout_width="wrap_content"
15     android:layout_height="wrap_content"
16     android:id="@+id/guideline"
17     app:layout_constraintGuide_percent="0.5"
18     android:orientation="vertical" />
19
20 <TextView
21     android:id="@+id/textView01"
22     android:text="0"
23     android:layout_width="0dp"
24     android:layout_height="wrap_content"
25     app:layout_constraintLeft_toLeftOf="parent"
26     app:layout_constraintRight_toRightOf="parent"
27     android:textSize="40dp"
28     android:gravity="right"
29     android:paddingBottom="20dp"
30     android:paddingRight="20dp"
31     android:background="@color/widok"
32 />
33
34 <TextView
35     android:id="@+id/textView02"
36     android:text="0"
37     android:layout_width="0dp"
38     android:layout_height="wrap_content"
39     app:layout_constraintLeft_toLeftOf="parent"
40     app:layout_constraintRight_toRightOf="parent"
41     app:layout_constraintTop_toBottomOf="@id/textView01"
42     android:textSize="40dp"
43     android:gravity="right"
44     android:layout_marginBottom="20dp"
45     android:paddingRight="20dp"
46     android:background="@color/widok"/>

```

Wygląd aplikacji (w pliku XML)

Pola TextView
wyświetlające wartości

Aplikacje mobilne

Wygląd aplikacji (w pliku XML)

```
50 <RadioGroup
51     android:id="@+id/radioGroup01"
52     android:layout_width="0dp"
53     android:layout_height="wrap_content"
54     app:layout_constraintTop_toBottomOf="@id/textView02"
55     app:layout_constraintLeft_toLeftOf="parent"
56     app:layout_constraintRight_toLeftOf="@id/guideline"
57     android:background="@color/widok">
58
59     <RadioButton
60         android:id="@+id/radioButton01"
61         android:layout_width="wrap_content"
62         android:layout_height="wrap_content"
63         android:text="PLN"
64         android:textSize="25sp"
65         android:checked="true"
66         android:onClick="zmianaWaluty"/>
67 <RadioButton...>
74 <RadioButton...>
81 </RadioGroup>
```

Pola wyboru waluty –
waluta wejściowa

Aplikacje mobilne

Wygląd aplikacji (w pliku XML)

```
<RadioGroup
    android:id="@+id/radioGroup02"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintTop_toBottomOf="@id/textView02"
    app:layout_constraintLeft_toLeftOf="@id/guideline"
    app:layout_constraintRight_toRightOf="parent"
    android:background="@color/widok">

    <RadioButton
        android:id="@+id/radioButton04"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="PLN"
        android:textSize="25sp"
        android:onClick="zmianaWaluty"/>
    <RadioButton...>
    <RadioButton...>
</RadioGroup>
```

Pola wyboru waluty –
waluta docelowa

Aplikacje mobilne

Wygląd aplikacji (w pliku XML)

Suwak - SeekBar

```
116      <SeekBar
117          android:id="@+id/seekBar01"
118          android:layout_width="0dp"
119          android:layout_height="wrap_content"
120          app:layout_constraintTop_toBottomOf="@id/radioGroup02"
121          app:layout_constraintLeft_toLeftOf="parent"
122          app:layout_constraintRight_toRightOf="parent"
123          android:padding="20dp"
124          android:background="@color/widok"
125          android:max="200"/>
126
```

```

12 TextView dane, wynik;
13 RadioButton danePLN, daneDolar, daneEuro, wynikPLN, wynikDolar, wynikEuro;
14 SeekBar suwak;
15 private double kursDolara=3.8, kursEuro=4.3;
16 @Override
17 protected void onCreate(Bundle savedInstanceState) {
18     super.onCreate(savedInstanceState);
19     setContentView(R.layout.activity_main);
20     dane = (TextView) findViewById(R.id.textView01);
21     wynik = (TextView) findViewById(R.id.textView02);
22     danePLN = (RadioButton) findViewById(R.id.radioButton01);
23     daneDolar = (RadioButton) findViewById(R.id.radioButton02);
24     daneEuro = (RadioButton) findViewById(R.id.radioButton03);
25     wynikPLN = (RadioButton) findViewById(R.id.radioButton04);
26     wynikDolar = (RadioButton) findViewById(R.id.radioButton05);
27     wynikEuro = (RadioButton) findViewById(R.id.radioButton06);
28     suwak = (SeekBar) findViewById(R.id.seekBar01);
29     SeekBar.OnSeekBarChangeListener l1 = new SeekBar.OnSeekBarChangeListener() {
30         @Override
31         public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
32             dane.setText(String.valueOf(i));
33             wynik.setText(
34                 (String.valueOf(przelicz((double) i))));
35         }
36         @Override
37         public void onStartTrackingTouch(SeekBar seekBar) {
38         }
39         @Override
40         public void onStopTrackingTouch(SeekBar seekBar) {
41         }
42     };
43     suwak.setOnSeekBarChangeListener(l1);
44 }

```

Kod aplikacji

Przypisanie
kontrolki do
zmiennych w
programie.

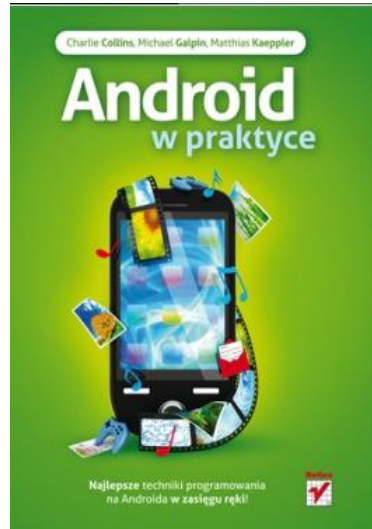
Stworzenie suwaka i
oprogramowanie
metody wykonywanej
w trakcie jego
przesuwania

Aplikacje mobilne

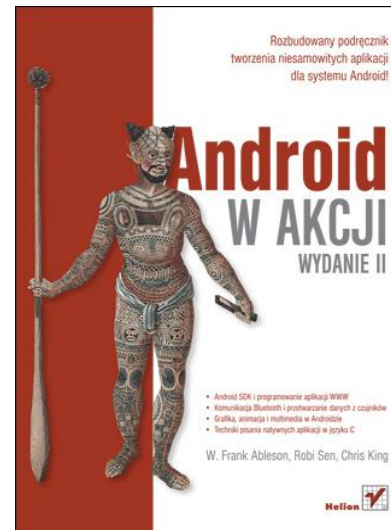
```
46 private double przelicz(double x) {
47     double kwotaPLN, wynik=0;
48     if (danePLN.isChecked()) kwotaPLN = x;
49     else if (daneDolar.isChecked()) kwotaPLN = x * kursDolara;
50     else kwotaPLN = x * kursEuro;
51
52     if (wynikPLN.isChecked()) wynik = kwotaPLN;
53     else if (wynikDolar.isChecked()) wynik= kwotaPLN / kursDolara;
54     else wynik = kwotaPLN / kursEuro;
55     return Math.round(wynik*100.0)/100.0;
56 }
57
58 public void zmianaWaluty (View view) {
59     wynik.setText(
60         (String.valueOf(
61             przelicz(Double.parseDouble(
62                 dane.getText().toString()))));
63     }
64 }
65 }
```

Metoda przeliczająca waluty

Metoda wypisująca wynik - korzysta z wyżej zdefiniowanej metody przelicz()



<https://developer.android.com>



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

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

