


Wykład 3

Komponenty wizualne – część 1

Programowanie Urządzeń Mobilnych

Dr inż. Damian Raczyński

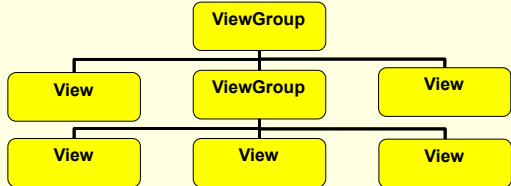


Interfejs Graficzny

Wszystkie elementy interfejsu graficznego są zbudowane w oparciu o klas `View` i `ViewGroup`.

`View` – obiekt, który „rysuje” coś na ekranie z czym użytkownik może wejść w interakcję.


`ViewGroup` – obiekt, który przechowuje obiekty `View` i `ViewGroup` w pewnym porządku.



```

graph TD
    VG1[ViewGroup] --> V1[View]
    VG1 --> VG2[ViewGroup]
    VG1 --> V2[View]
    VG2 --> V3[View]
    VG2 --> V4[View]
  
```

2




Interfejs Graficzny

Układ (`Layout`) definiuje strukturę interfejsu użytkownika.


Interfejs graficzny można definiować w pliku `xml` (pliku `Layout`) lub tworzyć go dynamicznie w kodzie.

Każdy plik `Layout` musi posiadać jeden element nadrzędny (korzeń), do którego można dodawać elementy potomne.



```


<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  . . . >
  <TextView android:id="@+id/text" . . . />
  <Button android:id="@+id/button" . . . />
</LinearLayout>
  
```



Interfejs Graficzny

W procesie kompilacji każdy plik `xml` z folderu `layouts` jest przekształcany na obiekt `View`.

Kojarzenie aktywności z układem odbywa się za pomocą metody `setContentView` – przyjmuje parametr w postaci referencji z klasy `R` (`R.layout.nazwa_layoutu`).



```

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
  
```

4

Interfejs Graficzny



ID – każdy obiekt View może mieć przypisane id
`android:id="@+id/button1"`
 @ - informacja dla parsera XML aby przetworzył zapis,
 + - nowy zasób, który musi zostać utworzony i dodany do klasy R

Za pomocą id możliwe jest pobranie referencji do obiektu w kodzie

Plik układu

```
<Button android:id="@+id/button1"
... />
```

Kod aktywności

```
Button x = (Button) findViewById(R.id.button1);
```

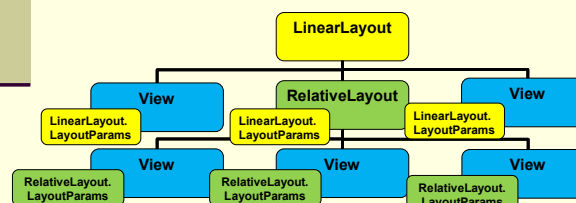
5

Interfejs Graficzny



Obiekty View mogą definiować atrybuty układu `layout_*`, które określają ustawienia dostępne dla nich w Layout'cie nadrzędnym (ViewGroup).

Obiekty ViewGroup zawierają atrybuty (związane z klasą zagnieżdżoną `ViewGroup.LayoutParams`) definiujące rozmiar i pozycję dla każdego obiektu wewnętrznego.



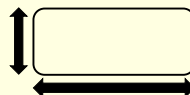
6

Interfejs Graficzny



Wszystkie Layout'y wymuszają na obiektach wewnętrznych użycia atrybutów

`layout_width` - szerokość,
`layout_height` - wysokość.



Wartości mogą zostać ustawione na sztywno (piksele, ...), wygodnie jest użyć stałych:

`wrap_content` - obiekt View zajmie tyle miejsca ile zajmuje jego zawartość,

TEKST

`match_parent` - obiekt będzie tak duży, na ile zezwoli mu Layout.

TEKST

```

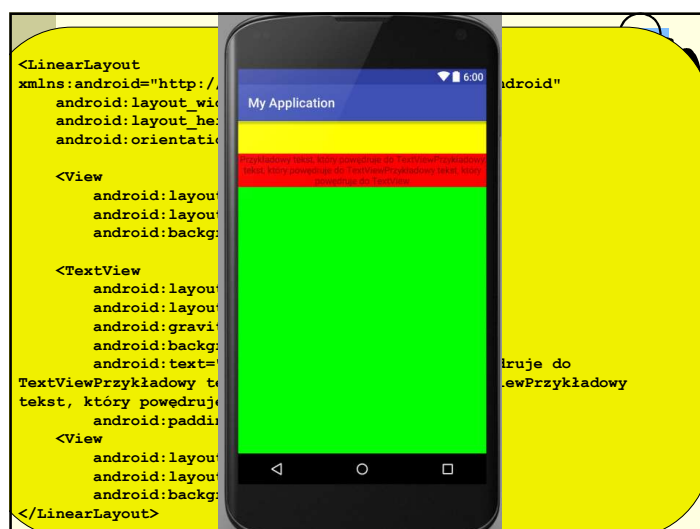
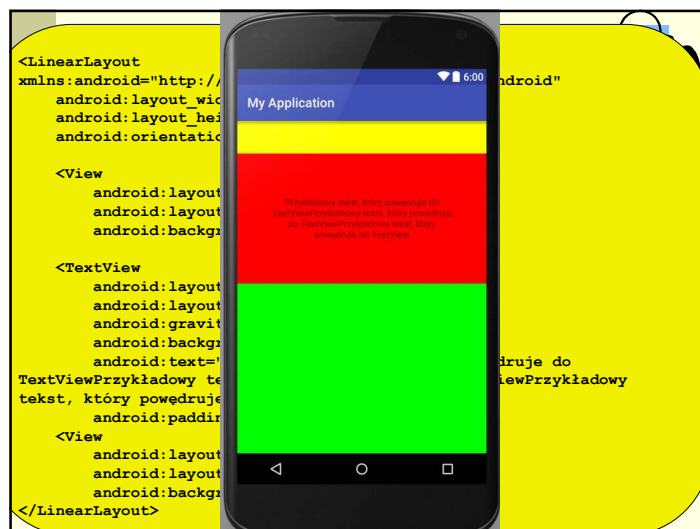
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

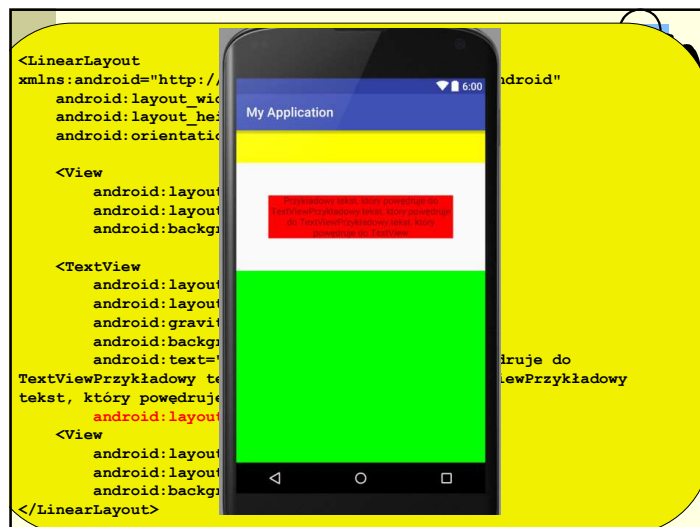
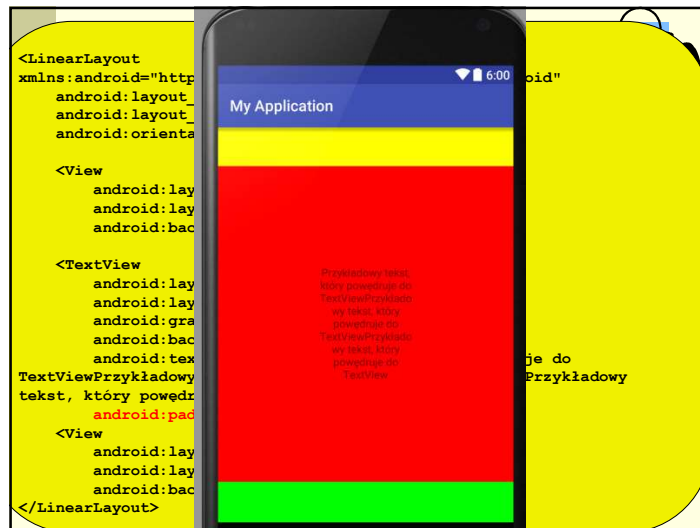
  <View
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:background="#ffff00"/>

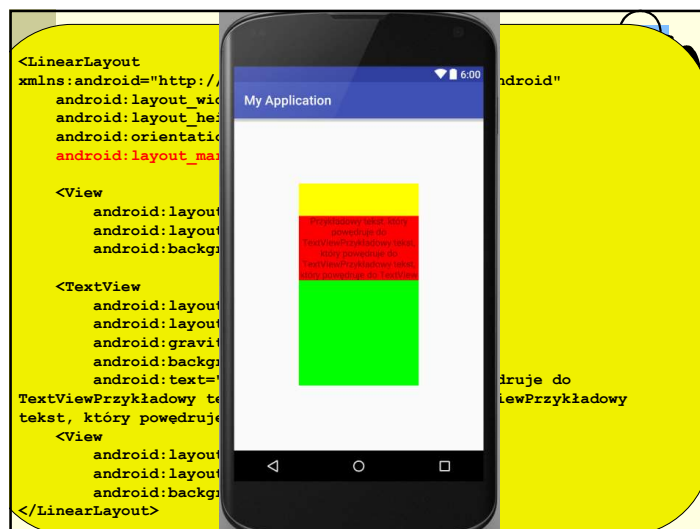
  <TextView
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:gravity="center"
    android:background="#ff0000"
    android:text="Przykładowy tekst, który powoduje do
    TextViewPrzykładowy tekst, który powoduje do TextViewPrzykładowy
    tekst, który powoduje do TextView"
    android:padding="50dp"/>

  <View
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#00ff00"/>

</LinearLayout>
  
```

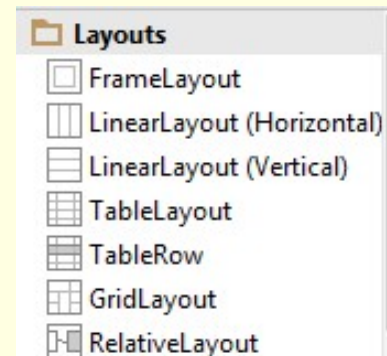






Układy

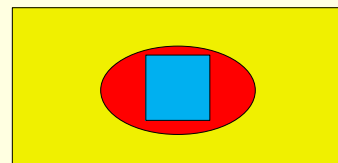
UKŁADY MOŻNA DOWOLNIE ZAGNIEŹDZAĆ.



Układy

Układ FrameLayout

Układ przeznaczony do przechowywania kontrolek
zajmujących ten sam obszar, umieszczonych jedna nad
drugą.



Układy

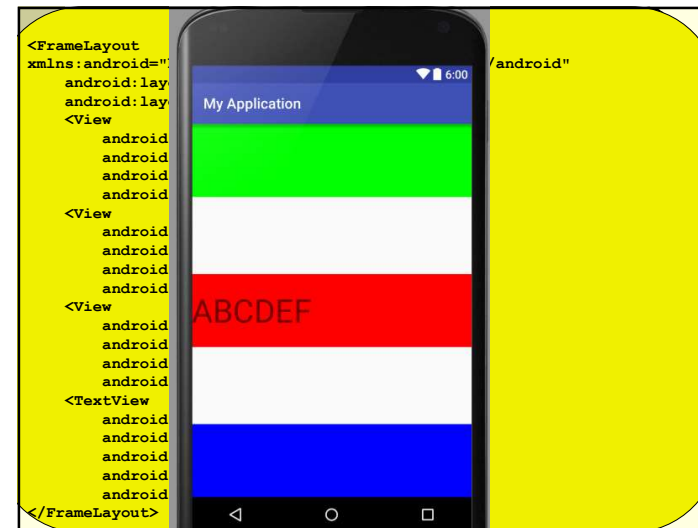
Wybrane atrybuty (należy poprzedzić **android:**):

layout_gravity – atrybut kontrolki podrzędnej. Określa wyrównanie kontrolki wewnątrz układu. wartości: top, bottom, left, right, center_vertical, fill_vertical, center_horizontal, fill_horizontal, center, fill.

```

<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <View
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:background="#FF0000"
        android:layout_gravity="center"/>
    <View
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:background="#00FF00"
        android:layout_gravity="top"/>
    <View
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:background="#0000FF"
        android:layout_gravity="bottom"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_gravity="center"
        android:text="ABCDEF"
        android:textSize="20pt"/>
</FrameLayout>

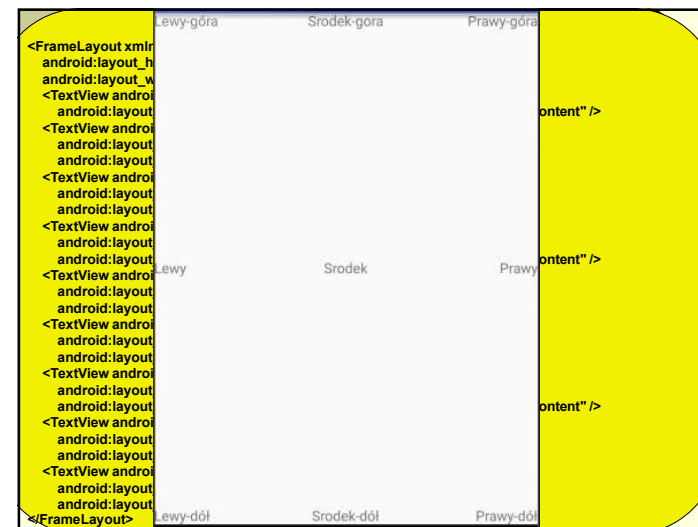
```



```

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent">
    <TextView android:text="Lewy-góra"
        android:layout_width="wrap_content"    android:layout_height="wrap_content" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"    android:text="Prawy-góra"
        android:layout_gravity="top|right" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"    android:text="Srodek-gora"
        android:layout_gravity="top|center_horizontal" />
    <TextView android:text="Lewy"
        android:layout_gravity="left|center_vertical"
        android:layout_width="wrap_content"    android:layout_height="wrap_content" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"    android:text="Prawy"
        android:layout_gravity="right|center_vertical" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"    android:text="Srodek"
        android:layout_gravity="center" />
    <TextView android:text="Lewy-dół"
        android:layout_gravity="left|bottom"
        android:layout_width="wrap_content"    android:layout_height="wrap_content" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"    android:text="Prawy-dół"
        android:layout_gravity="right|bottom" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"    android:text="Srodek-dół"
        android:layout_gravity="center|bottom" />
</FrameLayout>

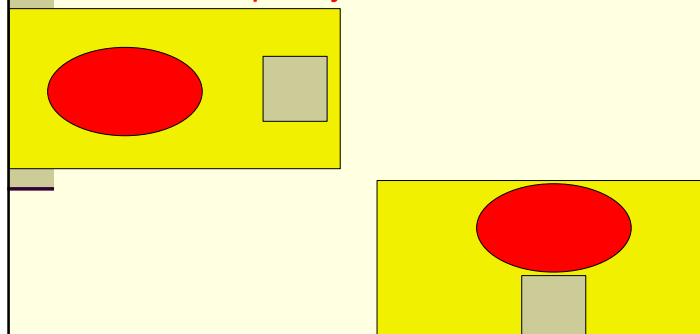
```



Układy

Układ LinearLayout

Umieszcza komponenty w wierszu lub w kolumnie



Układy

Wybrane atrybuty (należy poprzedzić **android:**):

orientation – kontrolka obiektu nadrzędnego. określa czy będzie układ horyzontalny czy wertykalny. wartości: **horizontal** lub **vertical**.

gravity – wyrównanie kontrolki umieszczonej w układzie (do obiektu nadrzędnego). Wartości: **top**, **bottom**, **left**, **right**, **center_vertical**, **fill_vertical**, **center_horizontal**, **fill_horizontal**, **center**, **fill**.

layout_gravity - -||- do obiektu podrzędnego

layout_weight – do kontrolki podrzędnych. Waga kontrolki. Waga używana jest do określenia proporcji obszarów układu zajmowanych przez kontrolki (jak dużo dodatkowego miejsca zajmie).

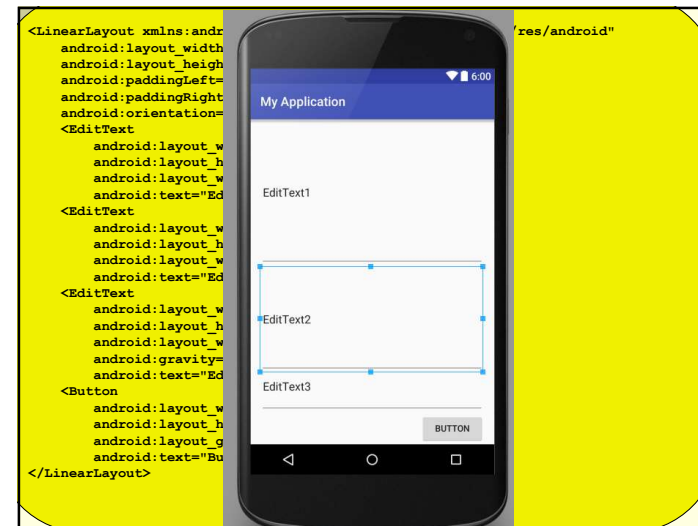
```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="EditText1"/>
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="EditText2"/>
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:text="EditText3"/>
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="Button"/>
</LinearLayout>
```

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="EditText1"/>
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="EditText2"/>
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:text="EditText3"/>
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="Button"/>
</LinearLayout>
```

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="0.3"
        android:text="EditText1"/>
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="0.2"
        android:text="EditText2"/>
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="0.1"
        android:gravity="top"
        android:text="EditText3"/>
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="Button"/>
</LinearLayout>

```

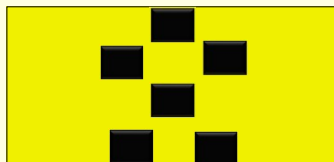


Układy

Układ RelativeLayout

Pozwala na określenie położenia umieszczonych komponentów względem innych – np. aby jeden komponent został wyświetlony poniżej drugiego, z jego prawej lub lewej strony.

Obiekt odniesienia ustalany na podstawie identyfikatora.



Układy

Wybrane atrybuty (należy poprzedzić **android:**):

gravity – wyrównanie kontrolki umieszczonej w układzie (do obiektu nadrzędnego). Wartości: top, bottom, left, right, center_vertical, fill_vertical, center_horizontal, fill_horizontal, center, fill.

layout_centerInParent – kontrolki podrzędne. Umieszcza kontrolkę podrzędną w środku nadrzędnej.

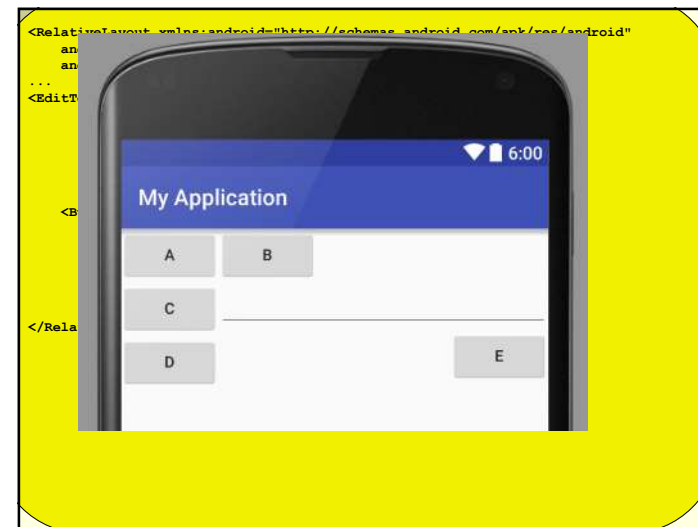
layout_alignParentX – X={Top, Bottom, Left, Right} – wyrównanie kontrolki podrzędnej w określony sposób

layout_X – X={above, below, toLeftOf, toRightOf} – podobnie jak poprzednio.


```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <Button android:id="@+id/btnButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"         android:text="A"/>
    <Button android:id="@+id/btnButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"         android:text="B"
        android:layout_toRightOf="@+id/btnButton1"/>
    <Button android:id="@+id/btnButton3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"         android:text="C"
        android:layout_below="@+id/btnButton1"/>
    <Button android:id="@+id/btnButton4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"         android:text="D" />
    <EditText android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_toRightOf="@+id/btnButton3" />
    <Button
        android:id="@+id/btnButton5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/editText1"         android:text="E" />
</RelativeLayout>

```

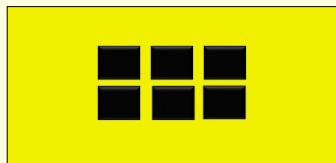


Układy

Układ TableLayout

Poszczególne elementy dodaje się do konkretnych wierszy tabeli układu TableLayout (TableRow).

Wymiary kolumn dostosowują się do wymiarów **największego** elementu.



Układy

Wybrane atrybuty (należy poprzedzić **android:**):

collapseColumns – dotyczy TableLayout – lista indeksów kolumn (od zera, rozdzielone przecinkami), które mają być ukryte (np. „0,3,5”)

shrinkColumns – dotyczy TableLayout – lista kolumn (jak poprzednio), które należy zwęzić.

stretchColumns – -||- , które należy rozszerzyć.

layout_column – dotyczy kontrolki podrzędnej TableRow – indeks kolumny, w której zostanie wyświetlona kontrolka podrzędna

layout_span – dotyczy TableRow – liczba kolumn, w których należy wyświetlić kontrolkę

36

```

<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1">
    <TableRow>
        <TextView
            android:text="Aaaaaa" />
        <TextView
            android:text="Bbbbb"
            android:gravity="right"/>
    </TableRow>
    <TableRow>
        <TextView
            android:text="Ccccc" />
        <TextView
            android:text="Ddddd"
            android:gravity="right" />
    </TableRow>
</TableLayout>

```

```

<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="0">
    <TableRow>
        <TextView
            android:text="Aaaaaa" />
        <TextView
            android:text="Bbbbb"
            android:gravity="right"/>
    </TableRow>
    <TableRow>
        <TextView
            android:text="Ccccc" />
        <TextView
            android:text="Ddddd"
            android:gravity="right" />
    </TableRow>
</TableLayout>

```

```

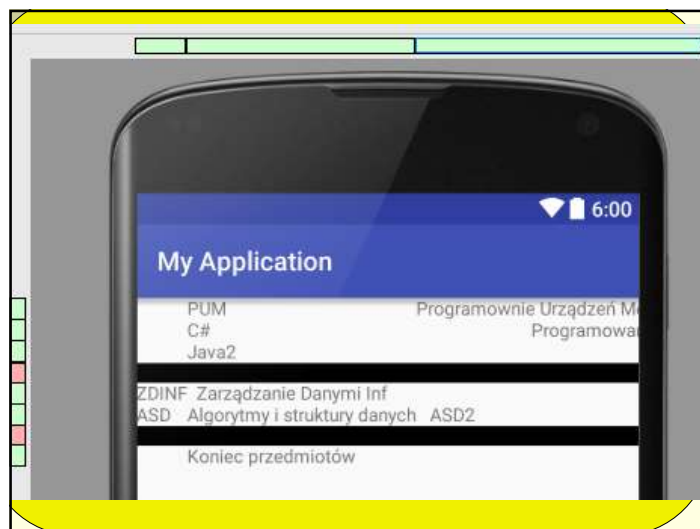
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:stretchColumns="1">
    <TableRow>
        <TextView android:layout_column="1" android:text="PUM" />
        <TextView android:text="Programownie Urzadzeń Mobilnych"
            android:gravity="right" />
    </TableRow>
    <TableRow>
        <TextView android:layout_column="1" android:text="C#" />
        <TextView android:text="Programowanie .NET" android:gravity="right" />
    </TableRow>
    <TableRow>
        <TextView android:layout_column="1" android:text="Java2" />
        <TextView android:text="JEE" android:gravity="right" />
    </TableRow>
    <View
        android:layout_height="15dip" android:background="#000000" />
    <TableRow>
        <TextView android:text="ZDINF" />
        <TextView android:text=" Zarządzanie Danymi Inf" />
    </TableRow>
    <TableRow>
        <TextView android:text="ASD" />
        <TextView android:text="Algorytmy i struktury danych" />
        <TextView android:text=" ASD2" android:gravity="left" />
    </TableRow>
    <View
        android:layout_height="15dip" android:background="#000000" />
    <TableRow>
        <TextView android:layout_column="1" android:text="Koniec przedmiotów" />
    </TableRow>
</TableLayout>

```

```

<View
    android:layout_height="15dip"
    android:background="#000000" />
<TableRow>
    <TextView
        android:layout_height="15dip"
        android:background="#000000" />
    <TableRow>
        <TextView
            android:layout_column="1"
            android:text="Koniec przedmiotów" />
    </TableRow>
</TableLayout>

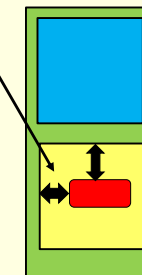
```



Interfejs Graficzny

Pozycja, rozmiar, marginesy

Metoda	Opis
getLeft()	Przesunięcie od lewej w px względem rodzica
getTop()	Przesunięcie od góry w px względem rodzica
getWidth()	Szerokość
getHeight()	Wysokość
getRight()	getLeft()+getWidth()
getBottom()	getTop()+getHeight()
setPadding(int,int,int,int)	Margines wewnętrzny (left,top,right,bottom)
getPaddingRight(), getPaddingBottom(),...	Pobiera rozmiar marginesu wewnętrznego



42

Pojemniki

Jeżeli zawartość Layoutu jest dynamiczna lub nieznana w czasie kompilacji można wykorzystać klasy pochodne AdapterView do wypełnienia Layoutu obiektami klasy View w trakcie działania programu.

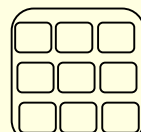
Pochodne AdapterView wykorzystują obiekty klasy Adapter do zbindowania danych do ich Layoutów.

Obiekty klasy Adapter konwertują dane do obiektów klasy View, które mogą zostać umieszczone w Layoutach AdapterView.

ListView



GridView



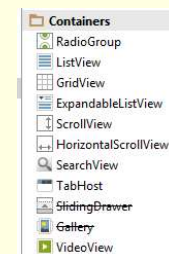
43

Pojemniki

ListView – przewijana w pionie lista komponentów typu View (zajmuje całą szerokość)

GridView – siatka obiektów View – może być przykładowo użyta do wyświetlania ikon

GalleryView – lista obiektów View przewijana w poziomie



Powyższe obiekty dziedziczą po klasie AdapterView. Elementy podrzędne mogą być dodane poprzez źródło danych Adapter.

44

Pojemniki



Przykład:

ArrayAdapter - w przypadku, gdy źródło danych pochodzi z tablicy. Tworzy obiekt View dla każdego elementu (wykorzystuje metodę toString), umieszczając zawartość w obiekcie TextView.

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, tablica);
ListView listView = (ListView) findViewById(R.id.listView);
listView.setAdapter(adapter);
```

Parametry:

- Kontekst,
- Domyślny układ, który zawiera TextView dla każdego Stringa,
- Tablica Stringów.

45

Adapte

```
<ListView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/listView"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true" />
```

layout
activity_main.xml
content_main.xml
przykl.xml

```
<TextView
    xmlns:android="http://schemas.android.com/apk/res/
    android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Text"
    android:id="@+id/textView2"

    android:layout_gravity="center_horizontal">
</TextView>
```

Adaptery



```
ListView listView1 = (ListView) findViewById(R.id.listView);

String[] elem = { "PUM", "JEE", "ZDINF", "Progr", "Zaocz" };
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    R.layout.przykl, elem);
listView1.setAdapter(adapter);
```

My Application

PUM
JEE
ZDINF
Progr
Zaocz

47

Adaptery



```
ListView listView1 = (ListView) findViewById(R.id.listView);

String [] elem = new String [100];
for(int i=0; i<100; i++) elem[i]="element "+i;

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    R.layout.przykl, elem);
listView1.setAdapter(adapter);
```

Adaptery

```

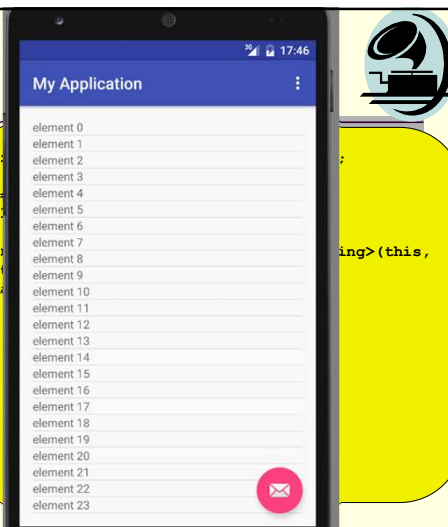
ListView listView1 = (L

String [] elem =
for(int i=0; i<

ArrayAdapter<Str
R.layout
listView1.setAd

ing>(this,

```



Pojemniki

Elementy `ListView`, `Gallery`, `GridView` zapewniają możliwość obsługi zdarzeń kliknięcia.

Należy w tym celu wywołać metodę `setOnItemClickListener()` obiektu `AdapterView`, przekazując obiekt typu `AdapterView.OnItemClickListener`.

Kolejno należy zaimplementować metodę `onItemClick`. Parametry metody `onItemClick`:

- `parent (arg0)` – obiekt `AdapterView`, którego element został kliknięty,
- `view (arg1)` – konkretna kliknięta kontrolka,
- `position (arg2)` - położenie kontrolki na liście (od zera),
- `id (arg3)` – pole `id`

50

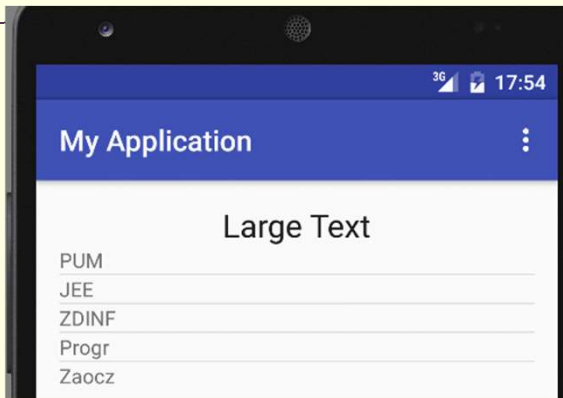
Pojemniki

```

ListView listView1 = (ListView) findViewById(R.id.listView);
String[] elem = { "PUM", "JEE", "ZDINF", "Progr", "Zaocz" };
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
R.layout.przykl, elem);
listView1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
long arg3) {
// TODO Auto-generated method stub
TextView napis= (TextView) findViewById(R.id.textView);
napis.setText("Wybrano pozycje numer: "+String.valueOf(arg2));
}
});
listView1.setAdapter(adapter);

```

Pojemniki



32

Długie przytrzymanie pozycji

```

ListView listView1 = (ListView) findViewById(R.id.listView1);

String[] elem = { "PUM", "JEE", "ZDINF", "Progr", "Zaocz" };
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    R.layout.przykl, elem);
listView1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
        long arg3) {
        // TODO Auto-generated method stub
        TextView napis= (TextView) findViewById(R.id.textView1);
        napis.setText("Wybrano pozycję numer: "+String.valueOf(arg2));
    }
});
listView1.setAdapter(adapter);

listView1.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> arg0, View arg1,
        int arg2, long arg3) {
        TextView napis= (TextView) findViewById(R.id.textView1);
        napis.setText("Długo przycisnieto element: "+String.valueOf(arg2));
        return false;
    }
});

```

ScrollView

ScrollView (HorizontalScrollView) – element zapewnia możliwość przewijania ekranu

Kontrolka może mieć wyłącznie jeden element podrzędny – przykładowo można przypisać do niej layout zawierający inne kontrolki.

Scroll

```

<ScrollView
    android:id="@+id/scrollView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView1"
    android:layout_marginLeft="98dp"
    android:layout_marginTop="170dp"
    android:layout_toRightOf="@+id/textView1" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button" />

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

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

        <Button
            android:id="@+id/button4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button" />

        <Button
            android:id="@+id/button5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button" />

    </LinearLayout>

</ScrollView>

```

SwitchView

SwitchView - Kontrolka zawiera dwa podrzędne obiekty View, w danym momencie może być widoczny tylko jeden z nich.

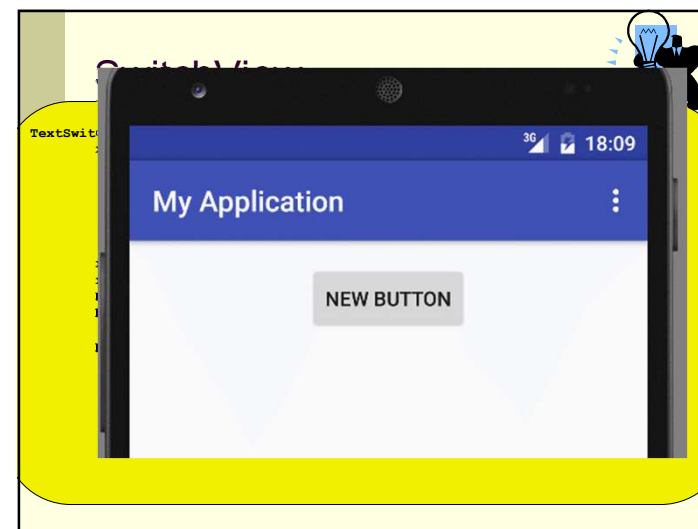
ImageSwitcher – przełączanie obrazków,

TextSwitcher – przełączanie tekstu.

```

public class MainActivity extends AppCompatActivity {
    String teksty[]={„tekst1“, „tekst2“, „tekst3“};
    int indeks=0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        TextSwitcher x = (TextSwitcher) findViewById(R.id.textSwitcher);
        x.setFactory(new ViewSwitcher.ViewFactory() {
            public View makeView() {
                TextView textV = new TextView(MainActivity.this);
                textV.setGravity(Gravity.CENTER);
                return textV;
            }
        });
        x.setInAnimation(this, android.R.anim.fade_in);
        x.setOutAnimation(this, android.R.anim.fade_out);
        Button b = (Button) findViewById(R.id.button);
        b.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                TextSwitcher x = (TextSwitcher) findViewById(R.id.textSwitcher);
                x.setText(teksty[++indeks%3]);
            }
        });
    }
}

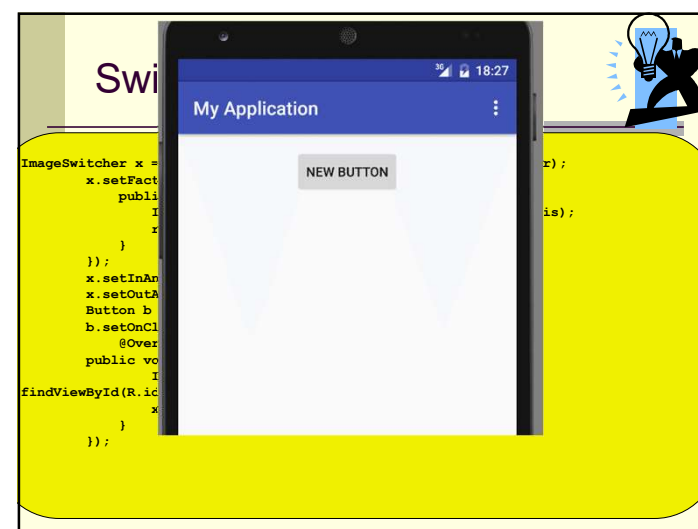
```



```

public class MainActivity extends AppCompatActivity {
    int obrazy[]={R.drawable.a, R.drawable.b};
    int indeks=0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ImageSwitcher x = (ImageSwitcher) findViewById(R.id.imageSwitcher);
        x.setFactory(new ViewSwitcher.ViewFactory() {
            public View makeView() {
                ImageView ImageV = new ImageView(MainActivity.this);
                return ImageV;
            }
        });
        x.setInAnimation(this, android.R.anim.slide_in_left);
        x.setOutAnimation(this, android.R.anim.slide_out_right);
        Button b = (Button) findViewById(R.id.button);
        b.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ImageSwitcher x = (ImageSwitcher)
                findViewById(R.id.imageSwitcher);
                x.setBackgroundResource(obrazy[++indeks%2]);
            }
        });
    }
}

```



Menu



Typy menu:

Menu **opcji** - akcje, które mają globalny wpływ na aplikację (niezależnie od wybranej aktywności - przycisk telefonu),

Menu **kontekstowe** - dotyczy określonego elementu (np. ListView, GridView ...),

floating - pojawia się po dłuższym przyciśnięciu elementu (lista wyboru),

contextual - zazwyczaj umieszczone w pasku, który jest wyświetlany w górnej części ekranu.

Menu typu **popup** - przypisane do konkretnego obiektu View.

W celu zdefiniowania wyglądu menu należy stworzyć odpowiedni plik XML w res/menu.

61

Menu



Znacznik	opis
<menu>	Kontener dla elementów menu
<item>	Odpowiada obiektom MenuItem - pojedyncza opcja menu
<group>	Niewidoczny kontener dla elementów <item>. Umożliwia współdzielenie właściwości pomiędzy elementami

Wybrane atrybuty <item>:

Znacznik	opis
android:id	ID
android:icon	ikona
android:title	Wyświetlana etykieta
android:showAsAction	

62

Menu



W celu określenia menu dla aktywności należy nadpisać metodę onCreateOptionsMenu().

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

Możliwe jest również zarządzanie menu w trakcie działania programu:

add() - dodaje elementy,
findItem() - zwraca element.

63

Menu Opcji



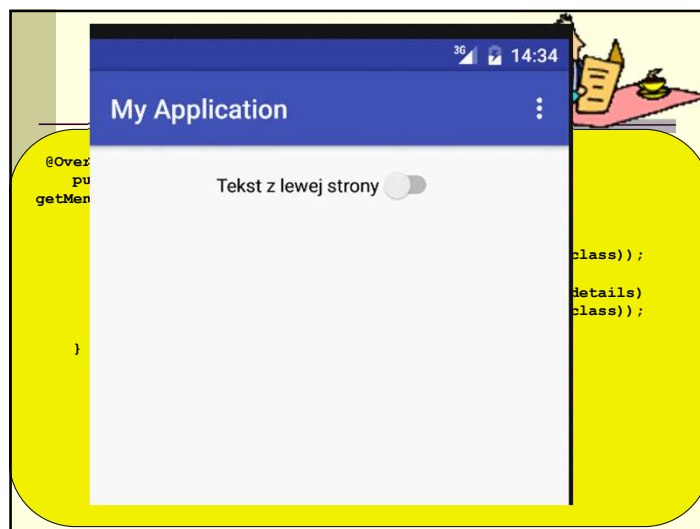
Kolejne pozycje menu należy dodać za pomocą metody add(String param)

Metody:

- setIcon - pozwala ustawić ikonę (obrazek ikony powinien znajdować się w zasobach drawable),
- setIntent - ustawia obiekt Intent (intencję)

Po wybraniu jednej z opcji zostanie uruchomiona aktywność określona obiektem Intent.

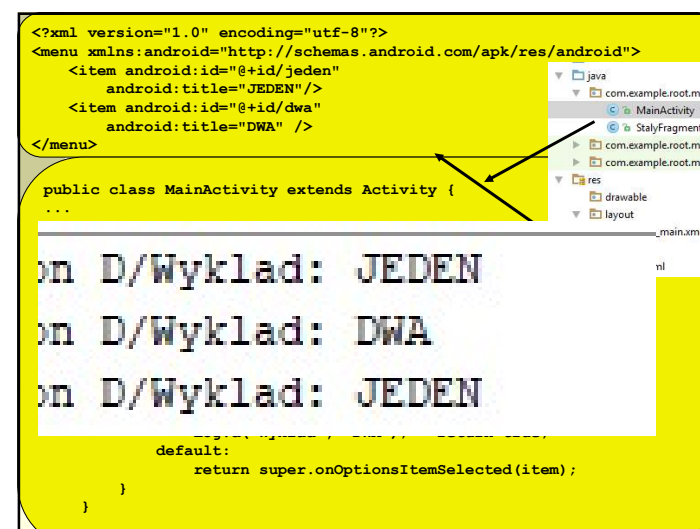
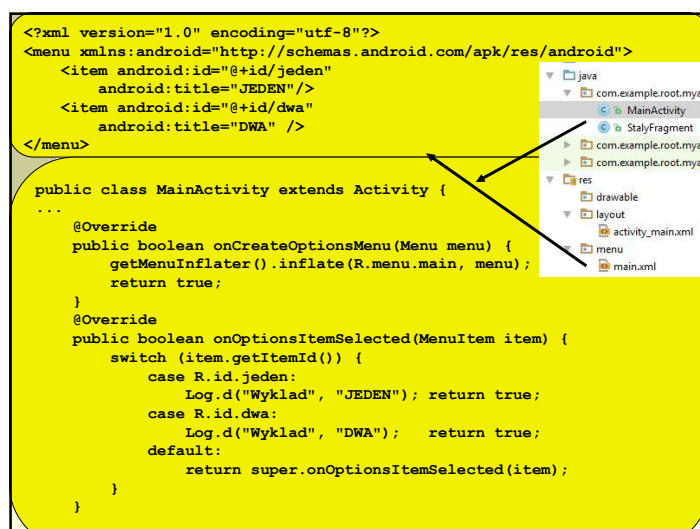
```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    menu.add("Opcja 1")
        .setIcon(android.R.drawable.ic_menu_edit)
        .setIntent(new Intent(this, MainActivity.class));
    menu.add("Opcja 2")
        .setIcon(android.R.drawable.ic_menu_info_details)
        .setIntent(new Intent(this, MainActivity.class));
    return true;
}
```

Menu Opcji

`onOptionsItemSelected()` - metoda wywoływana po wybraniu opcji menu.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.jeden:
            newGame();
            return true;
        case R.id.dwa:
            showHelp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```



Menu kontekstowe



Menu kontekstowe zostanie uruchomione, w przypadku, gdy użytkownik naciśnie i przez chwilę przytrzyma komponent View.

W celu stworzenia menu kontekstowego dla wybranej kontrolki należy wykonać szereg czynności:

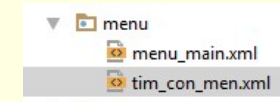
- W aktywności (lub fragmencie) aplikacji zaimplementować metodę `onCreateContextMenu`
- W aktywności (lub fragmencie) aplikacji zaimplementować metodę `onContextItemSelected`
- Stworzyć plik XML zawierający opis wszystkich opcji menu i ich identyfikatory
- zarejestrować fakt, że dany obiekt wykorzystuje menu kontekstowe – metoda `registerForContextMenu`

69

Menu kontekstowe



W metodzie `onCreateContextMenu` należy wykorzystać metodę `getMenuInflater().inflate` w celu przetworzenia pliku XML zawierającego opis menu do postaci obiektu menu.



```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/start" android:title="start"></item>
  <item android:id="@+id/stop" android:title="stop"></item>
</menu>
```

Menu kontekstowe



Metoda wywoływana za każdym razem, gdy zarejestrowany obiekt View zostanie kliknięty "długo"

Metoda w klasie aktywności głównej:

```
public void onCreateContextMenu(ContextMenu menu, View v,
    ContextMenu.ContextMenuInfo mi)
{
    super.onCreateContextMenu(menu, v, mi);
    if (v.getId() == R.id.chronometer)
    {
        getMenuInflater().inflate(R.menu.tim_con_men, menu);
        menu.setHeaderIcon(android.R.drawable.ic_media_play);
        menu.setTitle("Opcje dla minutnika");
    }
}
```

71

Menu kontekstowe



Metoda w klasie aktywności głównej:

```
public boolean onContextItemSelected(Menu.Item item)
{
    super.onContextItemSelected(item);
    boolean r = false;
    Chronometer t = (Chronometer) findViewById(R.id.chronometer);
    if (R.id.start == item.getItemId())
    {
        t.start();
        r = true;
    } else if (R.id.stop == item.getItemId())
    {
        t.stop();
        r = false;
    }
    return r;
}
```

Menu kontekstowe




```
Rejestracja w metodzie onCreate aktywności:  
Chronometer c = (Chronometer) findViewById(R.id.chronometer);  
registerForContextMenu(c);
```

73

Menu

My Application

00:00



74