POLITECHNIKA GDAŃSKA

Język Java i Android podstawy

Jacek Rumiński

Język Java i Android podstawy

Jacek Rumiński

Katedra Inżynierii Biomedycznej,
Wydział Elektroniki, Telekomunikacji i Informatyki
Politechnika Gdańska

Język Java i Android - podstawy
Plan prezentacji

1. Układy i komponenty
2. Obsługa zdarzeń
3. Operacje w systemie plików

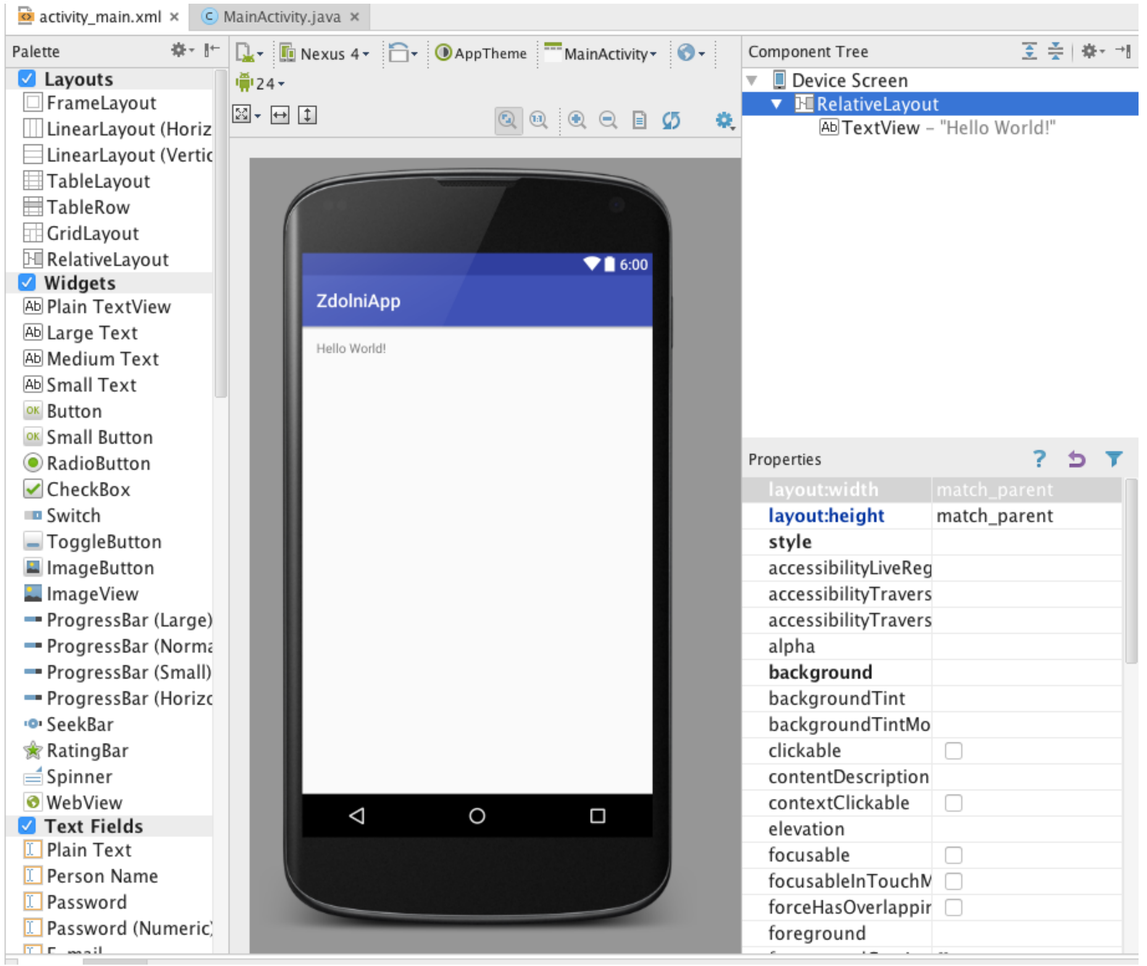
Interfejs graficzny w środowisku Android można projektować:

- bezpośrednio z kodu źródłowego (rzadko),
- za pośrednictwem pliku konfiguracyjnego w formacie XML (często),
- za pośrednictwem narzędzi graficznych (często).

activity_main.xml × MainActivity.java ×

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:paddingBottom="16dp"
7      android:paddingLeft="16dp"
8      android:paddingRight="16dp"
9      android:paddingTop="16dp"
10     tools:context="pg.eti.zdolni.zdolniapp.MainActivity">
11
12     <TextView
13         android:layout_width="wrap_content"
14         android:layout_height="wrap_content"
15         android:text="Hello World!" />
16 </RelativeLayout>
17
```

Design Text

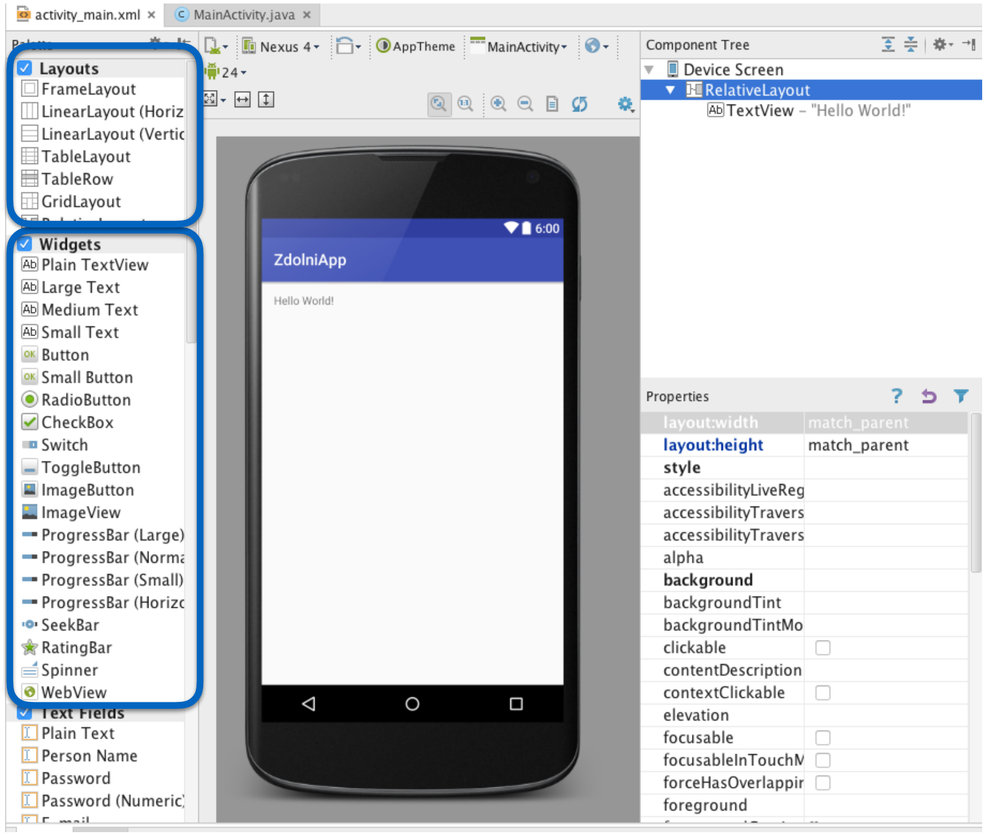


Widok (budowę) interfejsu konstruuje się wykorzystując kontener (lub ich hierarchię), zawierający komponenty. Kontenery w Androidzie to obiekty ViewGroup, określające sposoby układania komponentów. Stąd nazywane są popularnie układami (LAYOUTS). W danym układzie umieszczamy inne układy lub komponenty (widoki - View).

Układy

Widok w układzie (komponent w kontenerze)

Widok komponenty



Kluczowe metody układania komponentów:

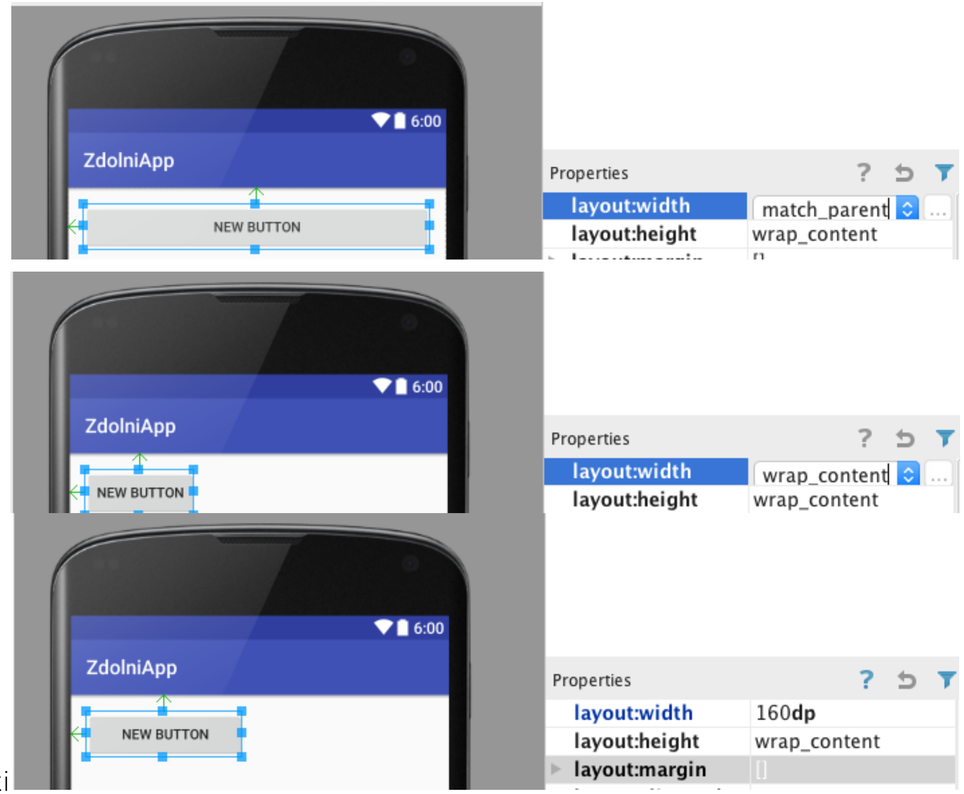
- **RelativeLayout** - umożliwia układanie komponentów z definicją wzajemnego położenia pomiędzy komponentami
- **LinearLayout** - umożliwia układanie komponentów w pojedynczej kolumnie lub w pojedynczym wierszu
- **AbsoluteLayout** - umożliwia określenie dokładnej lokalizacji komponentów (używając współrzędnych)
- inne (**GridLayout**, **FrameLayout**, własne, itp.).

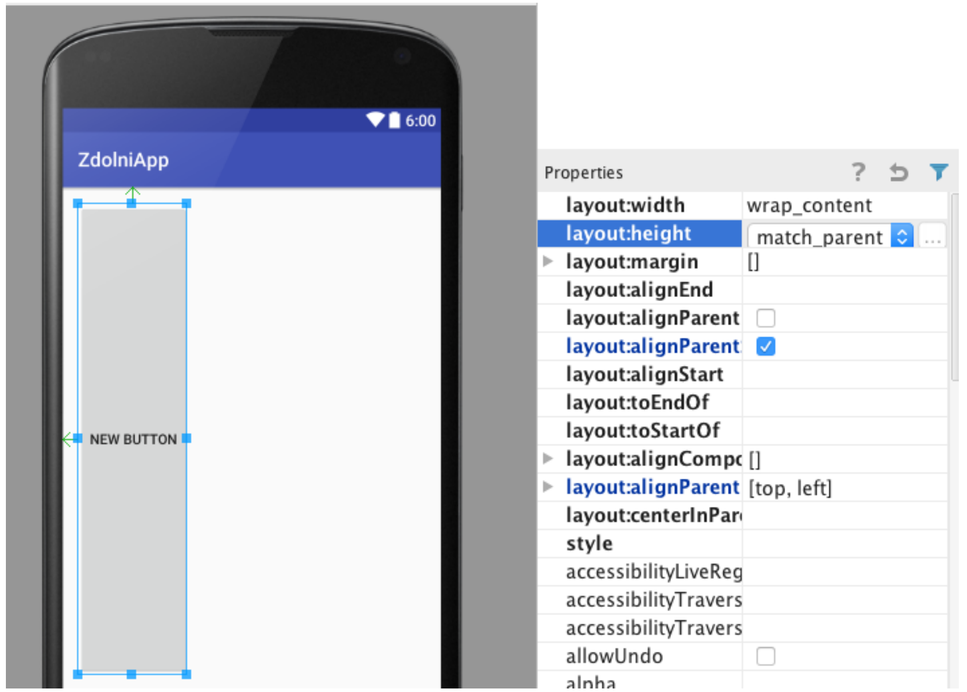
W kontenerach (układach) umieszczamy widoki - komponenty. Typowe komponenty to:

- **Button** - przycisk
- **TextView** - pole z tekstem (etykieta)
- **EditText** - pole z tekstem (wprowadzanie tekstu)
- **ImageView** - pole z obrazkiem
- inne (**CheckBox**, **RadioButton**, własne, itp.).

Rozmiar widoku umieszczonego w układzie kontrolujemy przez dobór parametrów wysokości (**height**) i szerokości (**width**) wskazując:

- **match_parent** (lub **fill_parent**) -> wypełnij dostępny obszar danego rodzica (układu)
- wrap_content -> ustaw rozmiar stosownie do treści (np. tekstu) wyświetlanego na komponencie,
- konkretną wartość w jednostkach **dp** (**D**ensity-independent **P**ixels - liczba pikseli niezależna od gęstości ekranu - automatycznie przeliczana)





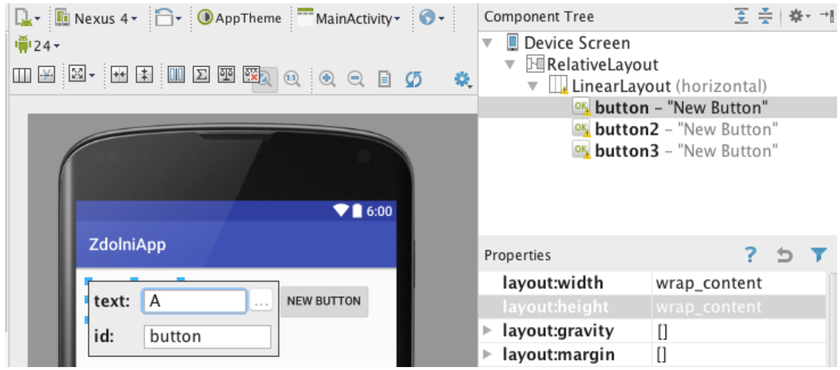
Uwaga! Sposób interpretacji wartości parametrów width i height może zależeć od rodzaju zastosowanego układu.

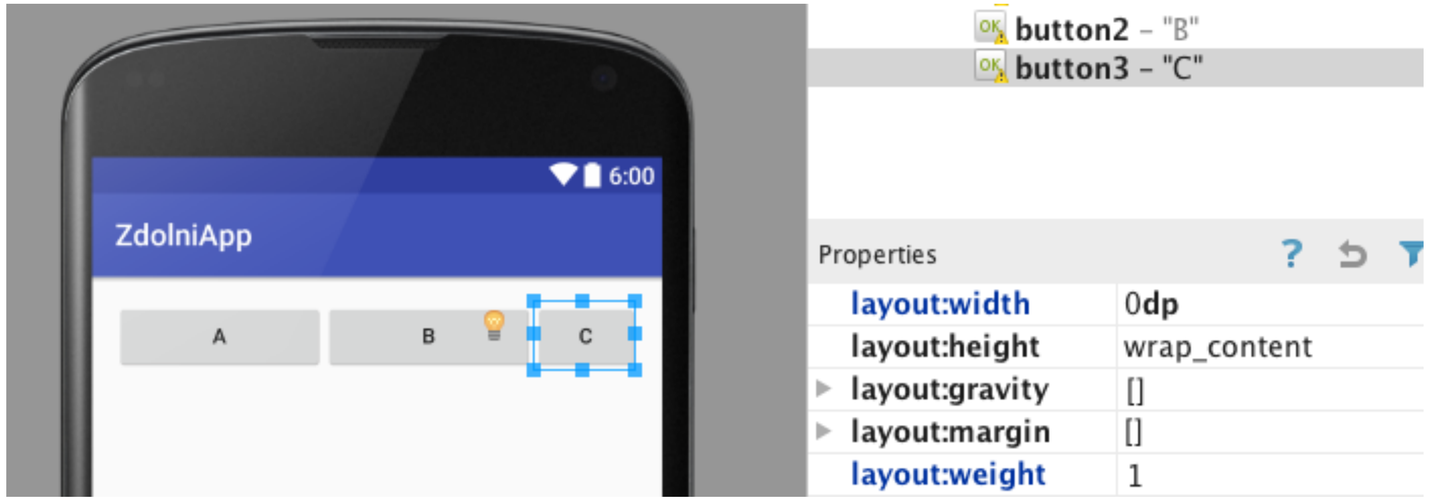
Bardzo praktycznym sposobem jest stosowaniem układu **LinearLayout** z wykorzystaniem parametru **layout:weight**. Parametr ten umożliwia nadanie wagi określającej względne jednostki szerokości/wysokości. Przykładowo:

layout:weight=2 | layout:weight=2 | layout:weight=1

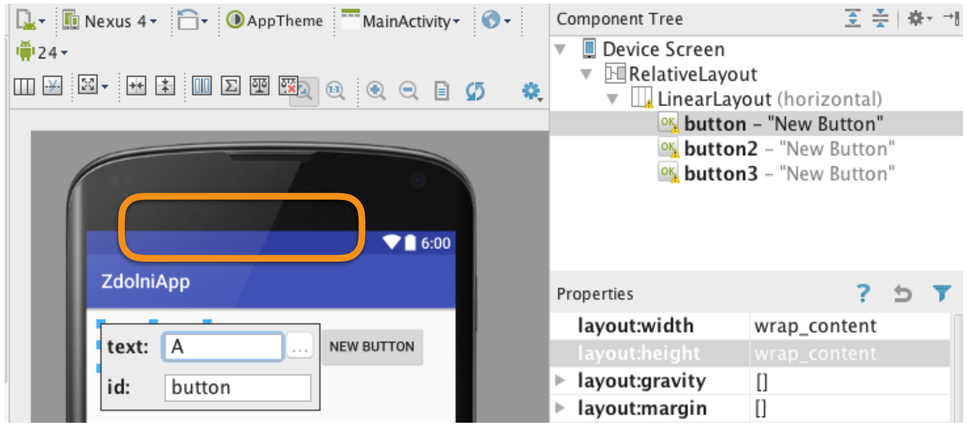
2/5 szerokości (40%) | 2/5 szerokości (40%) | 1/5 szerokości (20%)

(uwaga **layout:width** musi być ustawione na 0dp):





Każdy komponent ma określany identyfikator po to, aby można było się do niego odnieść w kodzie źródłowym.




```
<Button
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="A"
    android:id="@+id/button"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_weight="2" />
```

Button b=(Button)findViewById(R.id.button);

Reprezentuje zasoby (**R -** Resources).

1. Układy i komponenty
2. Obsługa zdarzeń
3. Operacje w systemie plików

Do kluczowych zdarzeń związanych z obsługą interfejsu graficznego można zaliczyć:

- „Wciśnięcie” przycisku - Click
- „Wciśnięcie” klawisza klawiatury - Key
- Dotknięcie ekranu - Touch
- i inne.

Z działaniami tymi związane będą metody obsługi zdarzeń oraz klasy zdarzeń:

- Click -> **onClink()** w interfejsie **OnClickListener**
- Key -> **onKey()** w interfejsie **OnKeyListener**; powiązana klasa zdarzenia **KeyEvent**,
- Touch -> **onTouch()** w interfejsie **OnTouchListener**; powiązana klasa zdarzeń **MotionEvent**
- i inne.

W dalszych przykładach będziemy wykorzystywać ciekawą formę powiadamiania użytkownika poprzez wyświetlanie powiadomienia w formie pola tekstowego.

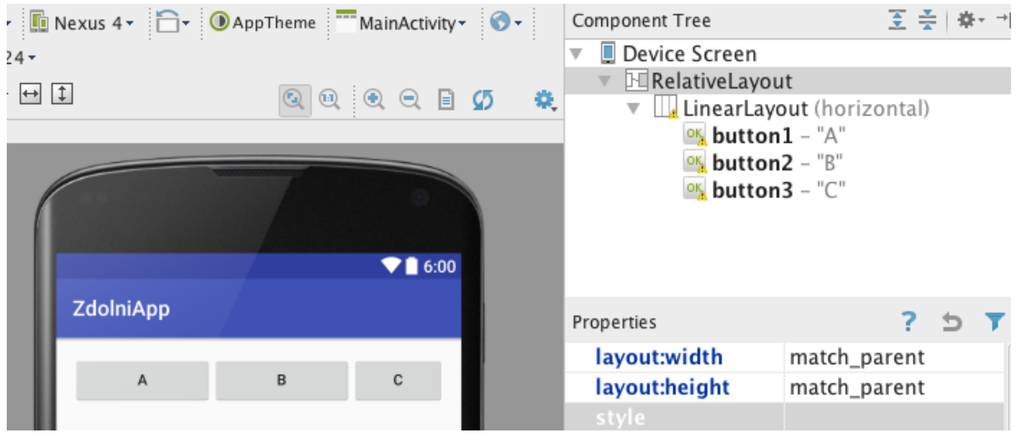
Do tego służy klasa **Toast**:

Toast.makeText(TU_OBIEKT_KONTEKSTU, TU_ZMIENNA_STRING_Z_WIADOMOŚCIĄ, TU_CZAS_WYŚWIETLANIA).show();

- TU_OBIEKT_KONTEKSTU - aktywność (**Activity**) przechowuje informacje o kontekście (jest typu/dziedziczy pośrednio po klasie **Context**) i jeśli używamy to polecenie w klasie typu **Activity** wówczas wystarczy podać: **this**.
- TU_ZMIENNA_STRING_Z_WIADOMOŚCIĄ - obiekt klasy **String**, np. "Moja wiadomość"
- TU_CZAS_WYŚWIETLANIA - okres czasu wyświetlania wiadomości, zwykle krótko: **Toast.LENGTH_SHORT** lub dłużej **Toast.LENGTH_LONG**

Przykład: `Toast.makeText(this, "Moja wiadomość",Toast.LENGTH_SHORT).show();`

Zróbmy przykład.




```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="pg.eti.zdolni.zdolniapp.MainActivity">

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true">

        <Button
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="A"
            android:id="@+id/button1"
            android:layout_alignParentTop="true"
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true"
            android:layout_weight="2" />

        <Button
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="B"
            android:id="@+id/button2"
            android:layout_weight="2" />

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

Zróbmy przykład.

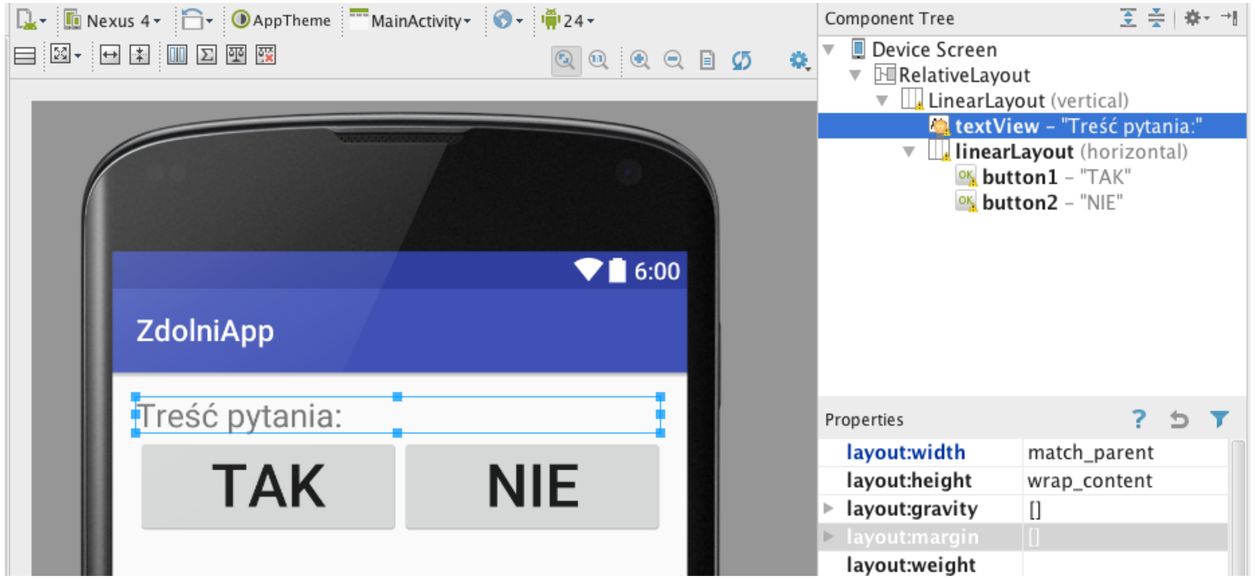
activity_main.xml × MainActivity.java ×

```java
package pg.eti.zdolni.zdolniapp;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

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

        Button b1=(Button)findViewById(R.id.button1);
        Button b2=(Button)findViewById(R.id.button2);
        Button b3=(Button)findViewById(R.id.button3);

        b1.setOnClickListener(this);
        b2.setOnClickListener(this);
        b3.setOnClickListener(this);

    }//onCreate()

    @Override
    public void onClick(View v) {
        switch(v.getId()) {
            case R.id.button1:{
                info("Wciśnięto przycisk 1");
                break;
            }
            case R.id.button2:{
                info("Wciśnięto przycisk 2");
                break;
            }
            case R.id.button3:{
                info("Wciśnięto przycisk 3");
                break;
            }
        }
    }//onClick()

    private void info(String info){
        Toast.makeText(this,info,Toast.LENGTH_SHORT).show();
    }//info()

}//class
```

Zróbmy przykład.

Toast

I jeszcze jeden przykład - proste rozszerzenie poprzedniego stanowiące praktyczny program.



RelativeLayout

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:paddingBottom="@dimen/activity_vertical_margin"
7      android:paddingLeft="@dimen/activity_horizontal_margin"
8      android:paddingRight="@dimen/activity_horizontal_margin"
9      android:paddingTop="@dimen/activity_vertical_margin"
10     tools:context="pg.eti.zdolni.zdolniapp.MainActivity">
11
12     <LinearLayout
13         android:orientation="vertical"
14         android:layout_width="match_parent"
15         android:layout_height="match_parent"
16         android:layout_alignParentBottom="true"
17         android:layout_alignParentLeft="true"
18         android:layout_alignParentStart="true">
19
20         <TextView
21             android:layout_width="match_parent"
22             android:layout_height="wrap_content"
23             android:text="Treść pytania:"
24             android:id="@+id/textView"
25             android:textSize="22dp" />
26
27         <LinearLayout
28             android:orientation="horizontal"
29             android:layout_width="match_parent"
30             android:layout_height="wrap_content"
31             android:layout_alignParentLeft="true"
32             android:layout_alignParentStart="true"
33             android:id="@+id/linearLayout">
34
35             <Button
36                 android:layout_width="0dp"
37                 android:layout_height="wrap_content"
38                 android:text="TAK"
39                 android:id="@+id/button1"
40                 android:layout_alignParentTop="true"
41                 android:layout_alignParentLeft="true"
42                 android:layout_alignParentStart="true"
43                 android:layout_weight="2"
44                 android:textSize="40dp" />
45
46             <Button
47                 android:layout_width="0dp"
48                 android:layout_height="wrap_content"
49                 android:text="NIE"
50                 android:id="@+id/button2"
51                 android:layout_weight="2"
52                 android:textSize="40dp" />
53         </LinearLayout>
54     </LinearLayout>
55 </RelativeLayout>
56
```

Przykład c.d.

```
10  public class MainActivity extends AppCompatActivity implements View.OnClickListener {
11
12      String [] questions={
13        "Czy dziedziczenie w programowaniu obiektowym oznacza przejęcie przez jedną klasę cech i zachowania innej klasy (bazowej)? ",
14        "Czy hermetyzacja w programowaniu obiektowym oznacza sterowanie dostępem do elementów klas za pomocą specyfikatorów public, private, protected?",
15        "Czy polimorfizm w programowaniu obiektowym oznacza metodą zapisywania plików?",
16         "Czy interfejs w programowaniu obiektowym w Javie oznacza klasę w pełni abstrakcyjną?"
17      };
18      boolean []answers={true,true,false,true};
19      int index=0;
20      TextView tv;
21
22      @Override
23      protected void onCreate(Bundle savedInstanceState) {
24          super.onCreate(savedInstanceState);
25          setContentView(R.layout.activity_main);
26
27          Button b1=(Button)findViewById(R.id.button1);
28          Button b2=(Button)findViewById(R.id.button2);
29          tv=(TextView)findViewById(R.id.textView);
30          tv.setText(questions[index]);
31          b1.setOnClickListener(this);
32          b2.setOnClickListener(this);
33      }//onCreate()
34
35      @Override
36      public void onClick(View v) {
37          switch(v.getId()) {
38              case R.id.button1:{
39                  if(answers[index]){
40                      info("Odpowiedź prawidłowa");
41                      if(index<(questions.length-1)) {
42                          index++;
43                          tv.setText(questions[index]);
44                      }
45                  }else
46                      info("Zła odpowiedź");
47                  break;
48              }
49              case R.id.button2:{
50                  if(!answers[index]){
51                      info("Odpowiedź prawidłowa");
52                      if(index<(questions.length-1)) {
53                          index++;
54                          tv.setText(questions[index]);
55                      }
56                  }else
57                      info("Zła odpowiedź");
58                  break;
59              }
60          }
61      }//onClick()
62
63      private void info(String info){
64          Toast.makeText(this,info,Toast.LENGTH_SHORT).show();
65      }//info()
66
67  }//class
```

Przykład c.d.


11:21
ZdolniApp
Czy hermetyzacja w programowaniu obiektowym oznacza sterowanie dostępem do elementów klas za pomocą specyfikatorów public, private, protected?
TAK
NIE
Odpowiedź prawidłowa

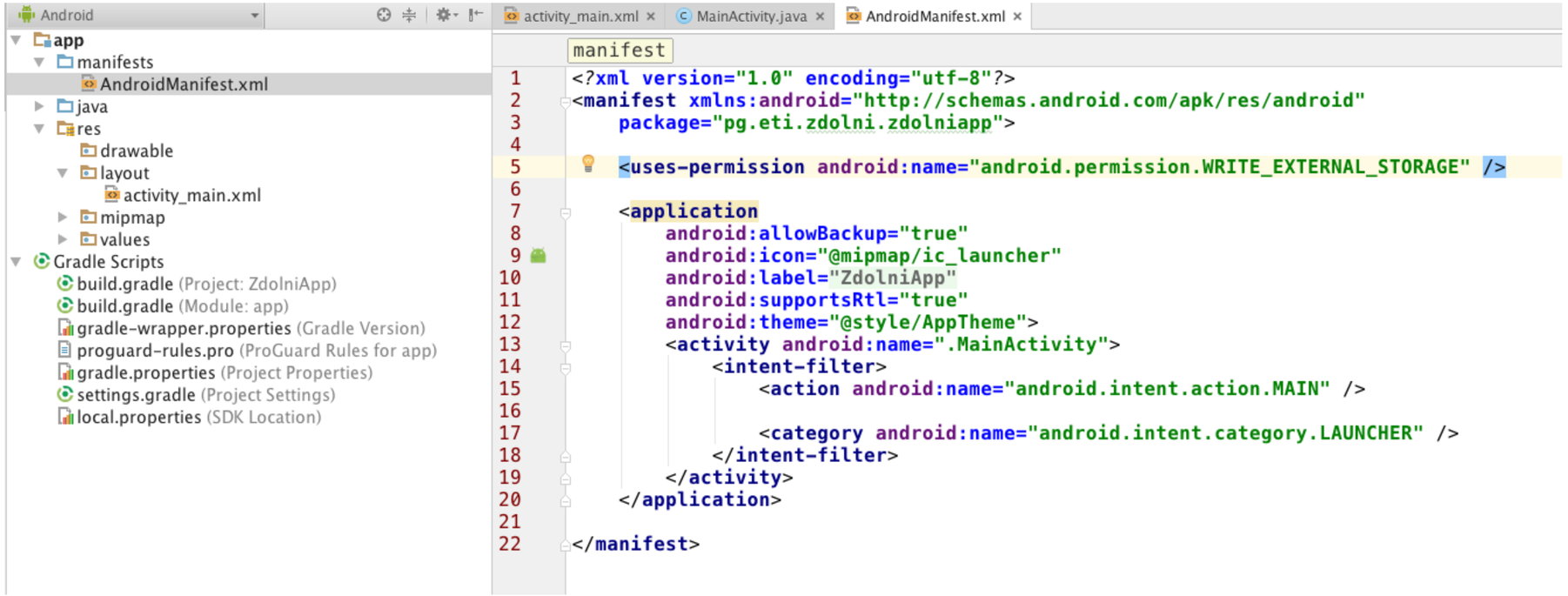
Język Java i Android - podstawy
Plan prezentacji

1. Układy i komponenty
2. Obsługa zdarzeń
3. Operacje w systemie plików

Na koniec tego materiału kilka informacji o nadawaniu uprawnień i operacjach na plikach.

Ze względu na to, że niektóre operacje mogą powodować niebezpieczne efekty dlatego wprowadzono kontrolę operacji z możliwością nadawania aplikacji pewnych uprawnień (przez programistę).

Uprawnienia te zapisuje się w pliku konfiguracyjnym AndroidManifest.xml:



Uprawnienia mają swoje nazwy i są wymagane aby pewne operacje były możliwe (docelowy użytkownik otrzymuje informację o żądaniu uprawnień i wyraża zgodę).

UWAGA! Wraz z kolejnymi wersjami systemu Android, uprawnienia i możliwości związane z powiązanymi operacjami mogą ulegać zmianom. (Np. w Androidzie w wersji 6 i nowszych zmieniono uprawnienia związane z zapisem plików).

Przykładowe uprawnienia:

WRITE_EXTERNAL_STORAGE - pozwól na zapis do plików na karcie SD
SEND_SMS - pozwól aplikacji wysyłać SMSy
INTERNET - pozwól aplikacji otwierać połączenia korzystając z gniazd TCP/IP
BLUETOOTH - pozwól aplikacji na połączenie z parowanym urządzeniem Bluetooth
ACCESS_NETWORK_STATE - pozwól aplikacji pobierać informacje o stanie sieci

i kilkadziesiąt innych

Operacje na strumieniach są realizowane za pomocą klas znanych już z podstaw języka Java.

W tej części zajmiemy się operacjami na plikach z zastosowaniem klas:

- **FileInputStream** - odczyt z pliku
- **FileOutputStream** - zapis do pliku
- **File** - Informacje i operacje w systemie plików

Ważne jest jednak w jaki sposób uzyskamy dostęp do karty pamięci (SD lub emulowanej SD), na której możemy wykonywać operacje. W tym celu posłużymy się poleceniem:

```
String root = Environment.getExternalStorageDirectory().toString();
```

wówczas root reprezentuje ścieżkę do katalogu karty SD. Inny przykład:

```
String root = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES).toString();
```

ilustruje pobranie adresu ścieżki do katalogu „Pictures” na karcie SD.

DEMONSTRACJA...

I to tyle na razie...

Jacek Rumiński