Programowanie aplikacji mobilnych

dr Przemysław Juszczuk

Katedra Inżynierii Wiedzy

Wykład 6

O czym dzisiaj?

- pliki w Java;
- pliki w Android;
- obsługa dźwięków;
- animacje.

Jak przechowywać dane w Android?

- Baza SQLite – rozbudowany mechanizm bazujący na silniku bazy danych;
- plik preferencji – proste rozwiązanie; domyślnie przechowujemy tylko podstawowe informacje (np. o ustawieniach aplikacji);
- pamięć wewnętrzna – prywatny katalog pamięci wewnętrznej umożliwiający przechowywanie plików tekstowych i binarnych;
- pamięć zewnętrzna – nie jest ograniczona, ale za każdym razem musimy sprawdzić, czy jest możliwy dostęp do danych.

Shared preferences

- klasa PreferenceManager i metoda getDefaultSharedPreferences;
- metoda edit pozwala na dostęp do pliku preferences – obiekt SharedPreferences.Editor;

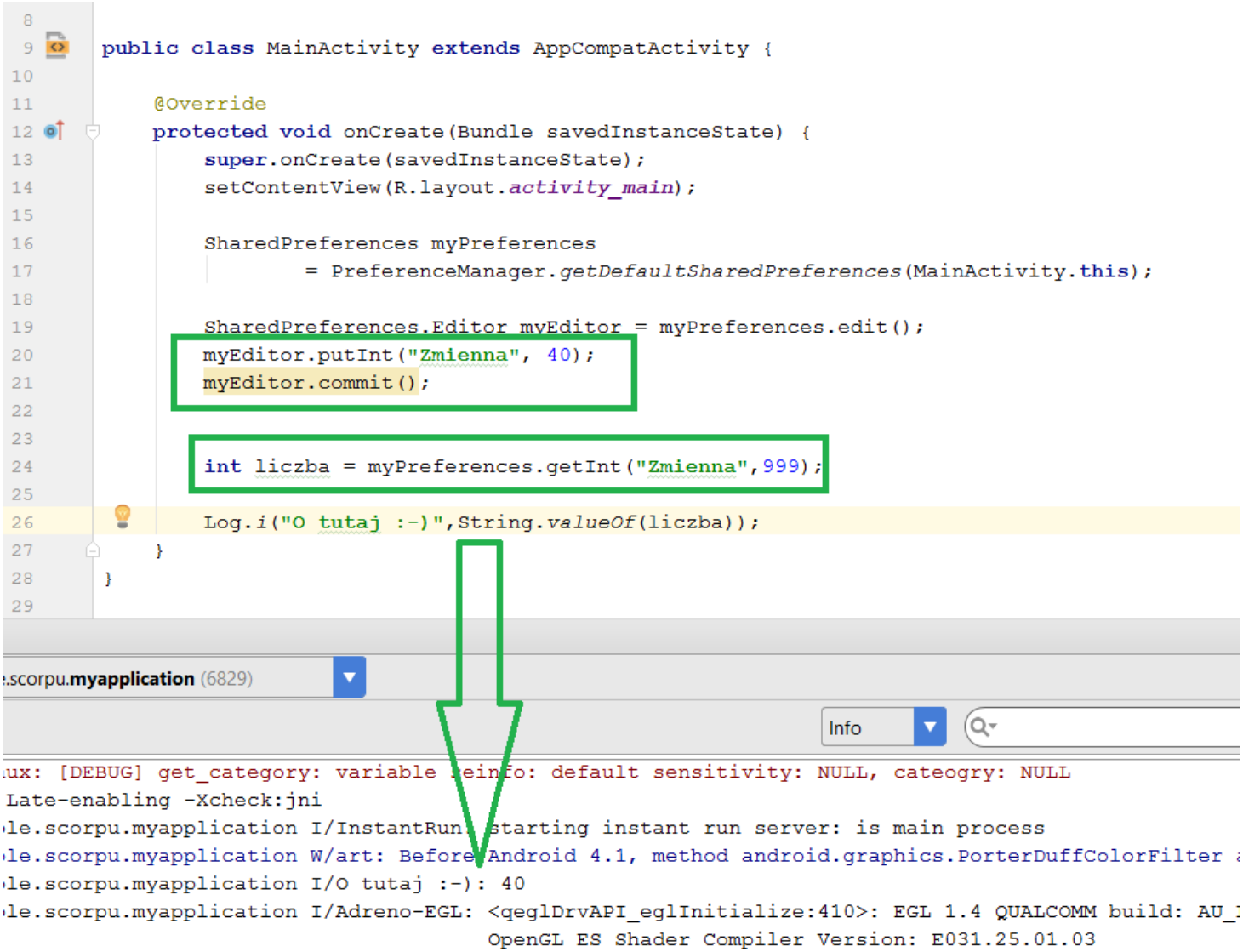
```
1  package com.example.scorpu.myapplication;
2
3  import android.content.SharedPreferences;
4  import android.preference.PreferenceManager;
5  import android.support.v7.app.AppCompatActivity;
6  import android.os.Bundle;
7
8  public class MainActivity extends AppCompatActivity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14
15         SharedPreferences myPreferences
16                 = PreferenceManager.getDefaultSharedPreferences(MainActivity.this);
17
18     }
19 }
20
```

Rysunek: Shared Preferences i import bibliotek

```
1  package com.example.scorpu.myapplication;
2
3  import android.content.SharedPreferences;
4  import android.preference.PreferenceManager;
5  import android.support.v7.app.AppCompatActivity;
6  import android.os.Bundle;
7
8  public class MainActivity extends AppCompatActivity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14
15         SharedPreferences myPreferences
16                 = PreferenceManager.getDefaultSharedPreferences(MainActivity.this);
17
18         SharedPreferences.Editor myEditor = myPreferences.edit();
19
20         myEditor.
21     }
22 }
23
```

```
clear()                                   Editor
commit()                                  boolean
putBoolean(String s, boolean b)           Editor
putFloat(String s, float v)               Editor
putInt(String s, int i)                   Editor
putLong(String s, long l)                 Editor
putString(String s, String s1)            Editor
putStringSet(String s, Set<String> set)   Ed
remove(String s)                          Editor
equals(Object obj)                        boolean
hashCode()                                int
```

Rysunek: Shared Preferences – zapis zmiennych



Rysunek: Shared Preferences – odczyt zmiennych

```java
import java.io.Serializable;

public class Pracownik implements Serializable {

    // prywatne pola klasy pracownik
    private String imie;
    private String nazwisko;
    private int wiek;
    private int staz_pracy;
    private double pensja;
    private final double MINIMUM = 1300.75;

    //konstruktor z 5 parametrami
     public Pracownik(String i, String n, int w, int s, double p)
     {
       this.imie = i;
       this.nazwisko = n;
       this.wiek = w;
       this.staz_pracy = s;
       this.pensja = p;
     }
     //konstruktor z 4 parametrami
     public Pracownik(String i, String n, int w, int s)
     {
         this.imie = i;
         this.nazwisko = n;
         this.wiek = w;
         this.staz_pracy = s;
         this.pensja = MINIMUM + this.staz_pracy*150.00;
     }
```

Rysunek: Szybka powtórka z plików – klasa w Java

```java
// metody do pobierania wartosci zmiennych
public String getImie() {
    return imie;
}
public String getNaziwsko()
{
    return nazwisko;
}
public int getWiek()
{
    return wiek;
}
public int getStaz_pracy()
{
    return staz_pracy;
}
public double getPensja()
{
    return pensja;
}

// metody do ustalania wartosci zmiennych
public void setImie(String i)
{
    this.imie = i;
}
public void setNazwisko(String nazwisko)
{
    this.nazwisko = nazwisko;
}
```

Rysunek: Szybka powtórka z plików – klasa w Java

```
public void setWiek(int w)
{
    this.wiek = w;
}
public void setStaz(int st)
{
    this.staz_pracy = st;
}
public void setPensja(double p)
{
    this.pensja = p;
}

// metoda wymagana do wyświetlenia danych na ekran
public String toString() {

    StringBuffer buffer = new StringBuffer();
    buffer.append(imie);
    buffer.append("\n");
    buffer.append(nazwisko);
    buffer.append("\n");
    buffer.append(wiek);
    buffer.append("\n");
    buffer.append(staz_pracy);
    buffer.append("\n");
    buffer.append(pensja);
    buffer.append("\n");


    return buffer.toString();
}
```

Rysunek: Szybka powtórka z plików – klasa w Java

```
// metoda do zapisu danych
public void ZapiszDane(String nazwaPliku) {

      ObjectOutputStream strumien_zapis = null;  // obiekt do zapisu danych
      // jeszcze poza blokiem try, bo nie ma żadnej wartości - przypisany NULL

      try {

          // już w bloku try: ustawiamy wartość obiektu
          strumien_zapis = new ObjectOutputStream(new FileOutputStream(nazwaPliku));

          //tworzymy dwa obiekty klasy Pracownik (przy pomocy róznych konstruktorów)
          Pracownik osoba = new Pracownik("Jan","Kowalski", 30, 5, 2500.00);
          Pracownik osoba2 = new Pracownik("Jan", "Nowak", 25, 2);


          //zapisujemy obydwa obiekty do pliku
          strumien_zapis.writeObject(osoba);
          strumien_zapis.writeObject(osoba2);

      }
      catch(Exception e)
      {
          e.printStackTrace();

      }

  }
```

Rysunek: Zapis danych ObjectOutputStream

```java
// funkcja do czytania z pliku
public void czytajDane(String nazwaPliku) {


   //tworzymy obiekt do czytania danych
   ObjectInputStream strumien_odczyt = null;

   try {

       strumien_odczyt = new ObjectInputStream(new FileInputStream(nazwaPliku));

       // deklarujemy obiekt klasy Object, bo nie wiemy, jakie obiekty mamy
       // moze być cokolwiek, dlatego lepiej dać tutaj Object
       Object obj = null;

       // czytamy kolejne obiekty z pliku ( przy pomocy readObject
       // dokladnie tak samo, jak w plikach tekstowych
       Pracownik test = new Pracownik("jan", "kowalskiii", 12);
       while ((obj = strumien_odczyt.readObject()) != null) {

           //sprawdzamy, czy odczytany obiekt należy do klasy pracownik
           if (obj instanceof Pracownik) {

               // jezeli tak, to wyswietlamy dane na ekran
               test = (Pracownik)obj;
               System.out.println(test.getNaziwsko());
               if(test.getNaziwsko() == "Nowak")
               {
                test.setNazwisko("jakies nowe nazwisko");
               }

               System.out.println(test.getNaziwsko());
               //System.out.println(((Pracownik)obj).innyToString());
           }

       }
   }
```

Rysunek: Odczyt danych ObjectInputStream

```
 // obsługa konca pliku - bez tego wysypie nam się odczyt
 catch (EOFException ex)
    {
        System.out.println("Koniec pliku.");
    }
  // na wszelki wypadek obsługa innych wyjątków
 catch (Exception e)
 {
     e.printStackTrace();
 }
}

public static void main(String[] args) {
    // wywolujemy metodę klasy Main (tak nazwałem główną klasę programu)
    new Main().ZapiszDane("plik2.txt");
    // czytamy dane z pliku
    new Main().czytajDane("plik2.txt");
}
```

Rysunek: Wywołanie metod

```
1  package com.example.scorpu.lab03;
2
3  import android.content.Context;
4  import android.media.MediaPlayer;
5  import android.os.Build;
6  import android.support.v7.app.AppCompatActivity;
7  import android.os.Bundle;
8  import android.view.View;
9  import android.widget.Button;
10 import android.widget.Toast;
11
12 import java.io.FileOutputStream;
13 import java.io.OutputStreamWriter;
14
15 public class MainActivity extends AppCompatActivity {
16
17     public static final int sound1 = R.raw.fx1;
18     public static final int sound2 = R.raw.fx2;
19     public static final int sound3 = R.raw.fx3;
20
21     Button dzwiek1;
22     Button dzwiek2;
23     Button dzwiek3;
24
25     public void playSound(Context context, int soundID)
26     {
27         MediaPlayer mp = MediaPlayer.create(context,soundID);
28         mp.start();
29     }
```

Rysunek: Wywołanie metod

```
25      public void playSound(Context context, int soundID)
26      {...}
30
31      @Override
32      protected void onCreate(Bundle savedInstanceState) {
33          super.onCreate(savedInstanceState);
34          setContentView(R.layout.activity_main);
35          dzwiek1 = (Button) findViewById(R.id.buttonSound1);
36          dzwiek2 = (Button) findViewById(R.id.buttonSound2);
37          dzwiek3 = (Button) findViewById(R.id.buttonSound3);
38      }
39
40      public void GrajSound_1(View v)
41      {...}
58
59      public void WriteBtn(View v) {
60          try {
61              FileOutputStream fileout=openFileOutput("datafile.txt", MODE_PRIVATE);
62              OutputStreamWriter outputWriter=new OutputStreamWriter(fileout);
63              outputWriter.write("Ala ma kota");
64              outputWriter.close();
65
66              Toast.makeText(getBaseContext(), "Zapisano!",
67                      Toast.LENGTH_SHORT).show();
68
69          } catch (Exception e) {
70              e.printStackTrace();
71          }
72      }
```

Rysunek: Wywołanie metod

```
3   import android.content.Context;
4   import android.media.MediaPlayer;
5   import android.os.Build;
6   import android.support.v7.app.AppCompatActivity;
7   import android.os.Bundle;
8   import android.view.View;
9   import android.widget.Button;
10  import android.widget.Toast;
11
12  import java.io.FileInputStream;
13  import java.io.FileOutputStream;
14  import java.io.InputStreamReader;
15  import java.io.OutputStreamWriter;
16
17  public class MainActivity extends AppCompatActivity {
18
19      public static final int sound1 = R.raw.fx1;
20      public static final int sound2 = R.raw.fx2;
21      public static final int sound3 = R.raw.fx3;
22
23      Button dzwiek1;
24      Button dzwiek2;
25      Button dzwiek3;
26
27      public void playSound(Context context, int soundID)
28      {...}
32
```

Rysunek: Wywołanie metod

```
12  import java.io.FileInputStream;
13  import java.io.FileOutputStream;
14  import java.io.InputStreamReader;
15  import java.io.OutputStreamWriter;
16
17  public class MainActivity extends AppCompatActivity {
18
19      public static final int sound1 = R.raw.fx1;
20      public static final int sound2 = R.raw.fx2;
21      public static final int sound3 = R.raw.fx3;
22
23      Button dzwiek1;
24      Button dzwiek2;
25      Button dzwiek3;
26
27      static final int READ_BLOCK_SIZE = 100;
28
29      public void playSound(Context context, int soundID)
30      {...}
34
35      @Override
36      protected void onCreate(Bundle savedInstanceState) {
37          super.onCreate(savedInstanceState);
38          setContentView(R.layout.activity_main);
39          dzwiek1 = (Button) findViewById(R.id.buttonSound1);
40          dzwiek2 = (Button) findViewById(R.id.buttonSound2);
41          dzwiek3 = (Button) findViewById(R.id.buttonSound3);
42      }
```

Rysunek: Wywołanie metod

```
44      public void GrajSound_1(View v)
45      {...}
62
63      public void WriteBtn(View v) {...}
77
78      public void ReadBtn(View v) {
79          try {
80              FileInputStream fileIn=openFileInput("datafile.txt");
81              InputStreamReader InputRead= new InputStreamReader(fileIn);
82
83              char[] inputBuffer= new char[READ_BLOCK_SIZE];
84              String s="";
85              int charRead;
86              while ((charRead=InputRead.read(inputBuffer))>0) {
87                  String readstring=String.copyValueOf(inputBuffer,0,charRead);
88                  s +=readstring;
89              }
90              InputRead.close();
91              Toast.makeText(getBaseContext(), s,Toast.LENGTH_SHORT).show();
92
93          } catch (Exception e) {
94              e.printStackTrace();
95          }
96      }
```

Rysunek: Wywołanie metod

Trochę o dźwiękach

Wersja Lollipop (5.1) – i wyżej

- różnice związane z obsługą dźwieku w przypadku wersji Lollipop i wyżej oraz wcześniejszych;
- zmienna statyczna do sprawdzenia aktualnej wersji systemu;
- porównanie wersji z wersją docelową (Lollipop).

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP)
{
  Toast.makeText(this, "Ok", Toast.LENGTH_SHORT).show();
}
else
{
  Toast.makeText(this, "stary", Toast.LENGTH_SHORT).show();
}
```

Co potrzebujemy?

- klasa Soundpool - służąca do przechowywania i operowania kolekcją dźwięków (pool of sounds). W tej klasie mamy nie tylko obsługę i odtwarzanie dźwięku, ale także dekompresję;
- dodanie folderu z dźwiękami - do assets;
- obsługa dźwięków wave, ogg;
- nowe podejście to wykorzystanie obiektu AudioAttributes do ustalenia konkretnych atrybutów;
- nowszy, nieco dłuższy sposób;

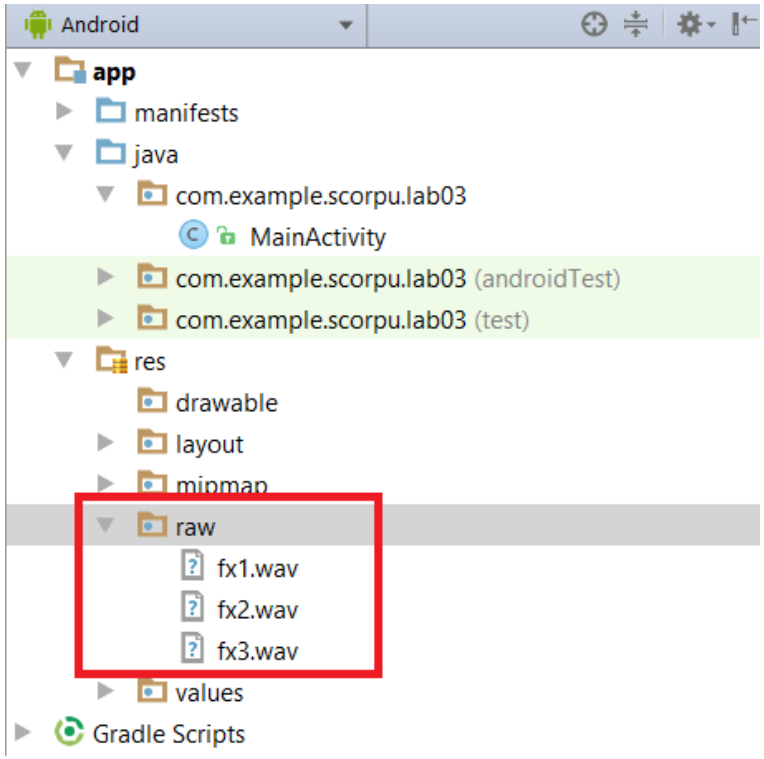
Zacznijmy starszego sposobu

```
SoundPool sp; // główny obiekt
int nowPlaying =-1; // flaga - odgrywanie dźwięku true/false
int idFX1 = -1;  // jw.
sp = new SoundPool(5, AudioManager.STREAM_MUSIC, 0);
// czyli lista parametrów do obiektu klasy SoundPool.
```

Dalej SoundPool

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP)
{
AudioAttributes audioAttributes = new
     AudioAttributes.Builder().setUsage(AudioAttributes.
     USAGE_ASSISTANCE_SONIFICATION)
     .setContentType(AudioAttributes.CONTENT_TYPE_SONIFICATION)
     .build();

sp = new SoundPool.Builder()
     .setMaxStreams(5)
     .setAudioAttributes(audioAttributes)
     .build();
}
```

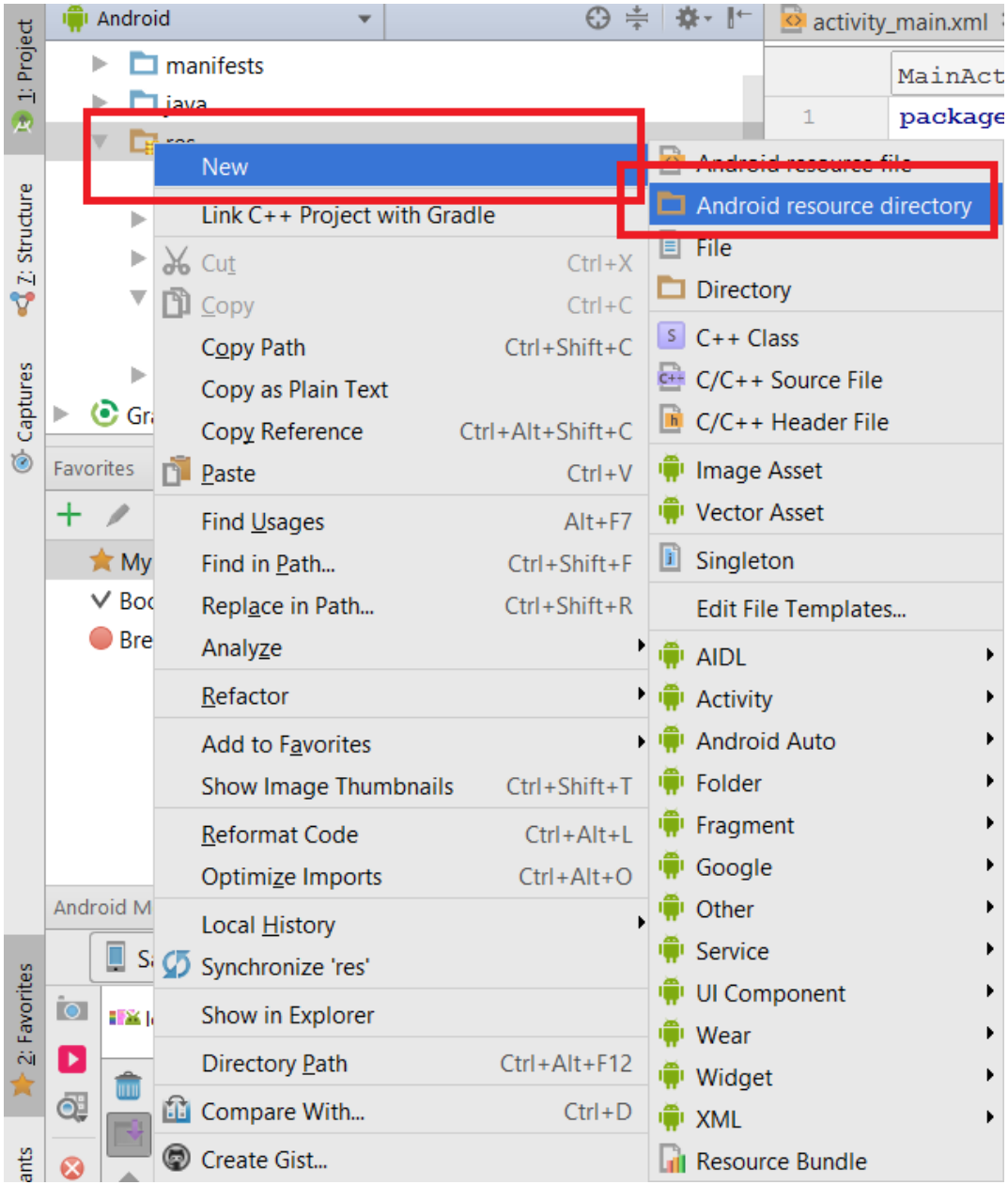


Rysunek: Katalog z dźwiękiem

Try catch finally

```
try
{
  AssetManager assetManager = this.getAssets();
  AssetFileDescriptor descriptor;

  descriptor = assetManager.openFd("fx1.ogg");
  idFX1 = sp.load(descriptor, 0);
}
catch(IOException e)
{
  Log.e("error", "Problem z plikiem");
}
```



Rysunek: Katalog z dźwiękiem

```
public class MainActivity extends AppCompatActivity {

    public static final int S1 = R.raw.fx1;

    public static void playSound(Context context, int soundID){
        MediaPlayer mp = MediaPlayer.create(context, soundID);
        mp.start();
    }


public void Graj(View v)
{
    Toast.makeText(this, "Mamy jakiś dźwięk", Toast.LENGTH_SHORT).show();
    playSound(this,S1);

}
```

Rysunek: Klasa MediaPlayer

Dziękuję za uwagę