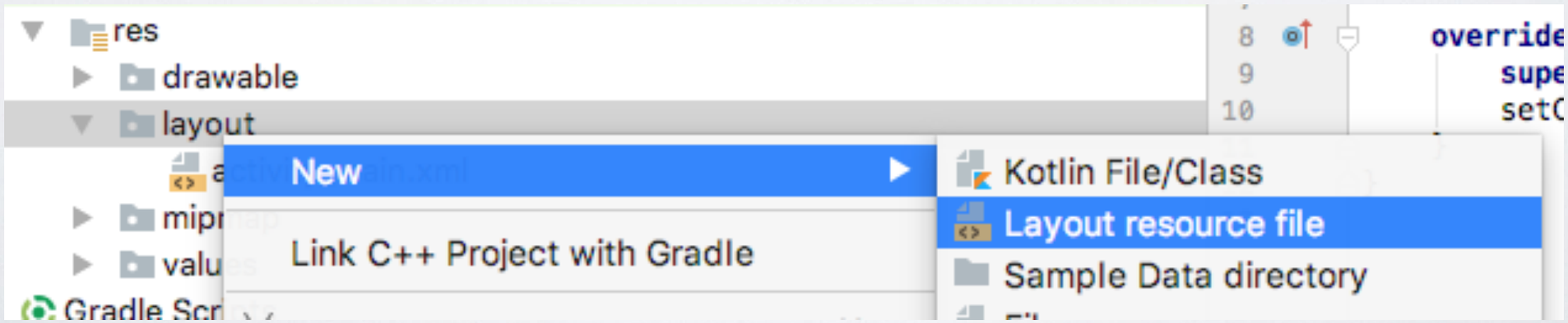
WSTĘP DO ANDROIDA

Laboratorium 4
Systemy i aplikacje bez granic

I

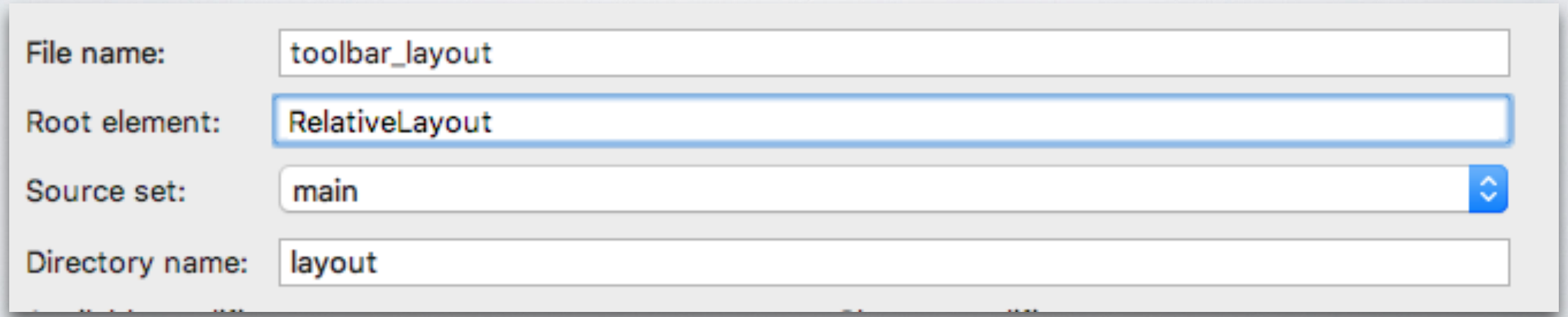
- Tworzymy nowy projekt typu Empty Acvivity o nazwie FragmentExample

- Rozwijamy gałąź app/res/layout i klikając prawym klawiszem dodajemy nowy Layout



I

- Nazywamy plik toolbar_layout
- Jako korzeń (root) ustawiamy RelativeLayout



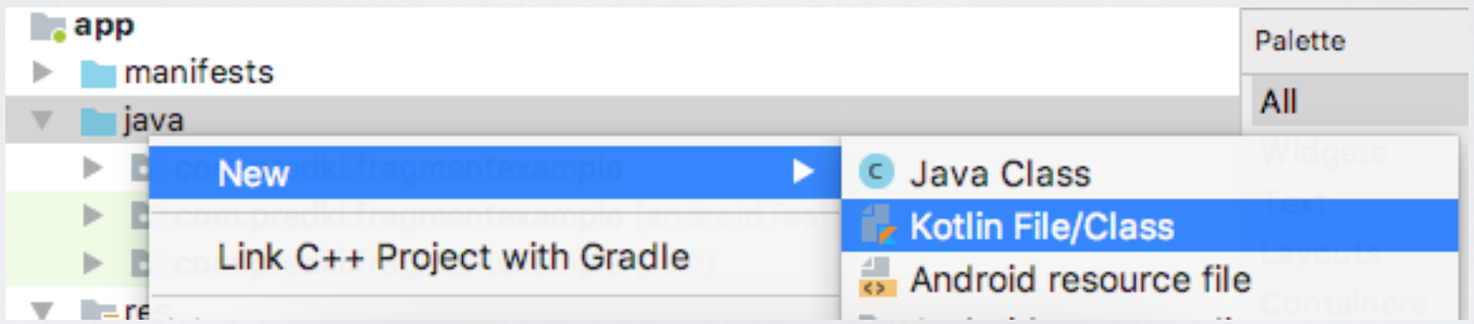
- Edytujemy projekt

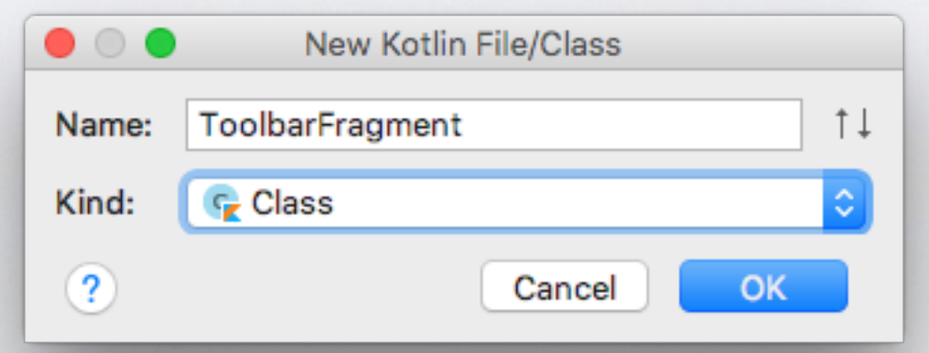
```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/seekBar1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="17dp"
        android:text="Zmień tekst" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp"
        android:ems="10"
        android:inputType="text" >
        <requestFocus />
    </EditText>

    <SeekBar
        android:id="@+id/seekBar1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/editText1"
        android:layout_marginTop="14dp"
        android:layout_alignParentLeft="true" />

</RelativeLayout>
```

- https://pastebin.com/Py6tHq7P

|

- Dodajemy klasę do app/java





- W kodzie klasy dopisujemy dziedziczenie po klasie Fragment

```
import android.app.Fragment

class ToolbarFragment: Fragment() {
}
```

I

- Przeciążamy metodę onCreateView

```
override fun onCreateView(inflater: LayoutInflater?,
                          container: ViewGroup?, savedInstanceState: Bundle?): View? {

    val view = inflater?.inflate(R.layout.toolbar_fragment,
            container, attachToRoot: false)
    val seekBar: SeekBar? = view?.findViewById(R.id.seekBar1)
    val button: Button? = view?.findViewById(R.id.button1)
    seekBar?.setOnSeekBarChangeListener(this)
    button?.setOnClickListener { v: View -> buttonClicked(v)}

    return view
}
```

- Dodajemy interfejs wewnątrz klasy

```
interface ToolbarListener {
    fun onButtonClick(position: Int, text: String)
}
```

|

- Oraz dwa pola
- Interfejs
- I metody

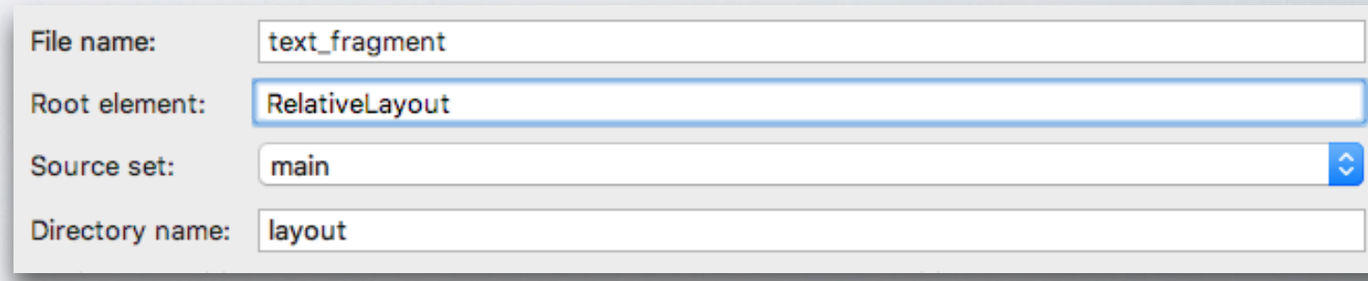
```
var seekvalue = 10
var activityCallback: ToolbarFragment.ToolbarListener? = null
```

```
class ToolbarFragment: Fragment(), SeekBar.OnSeekBarChangeListener {
```

```
override fun onAttach(context: Context?) {
    super.onAttach(context)
    try {
        activityCallback = context as ToolbarListener
    } catch (e: ClassCastException) {
        throw ClassCastException(context?.toString()
                + " musi implementować interfejs")
    }
}

private fun buttonClicked(view: View) {
    activityCallback?.onButtonClick(seekvalue,
            editText1.text.toString())
}

override fun onProgressChanged(seekBar: SeekBar, progress: Int,
                               fromUser: Boolean) {
    seekvalue = progress
}

override fun onStartTrackingTouch(arg0: SeekBar) {
}

override fun onStopTrackingTouch(arg0: SeekBar) {
}
```

|

- Analogicznie tworzymy drugi fragment o nazwie text_fragment
- Projekt drugiego fragmentu




```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="Fragment Two"
        android:textAppearance="?android:attr/textAppearanceLarge" />
</RelativeLayout>
```

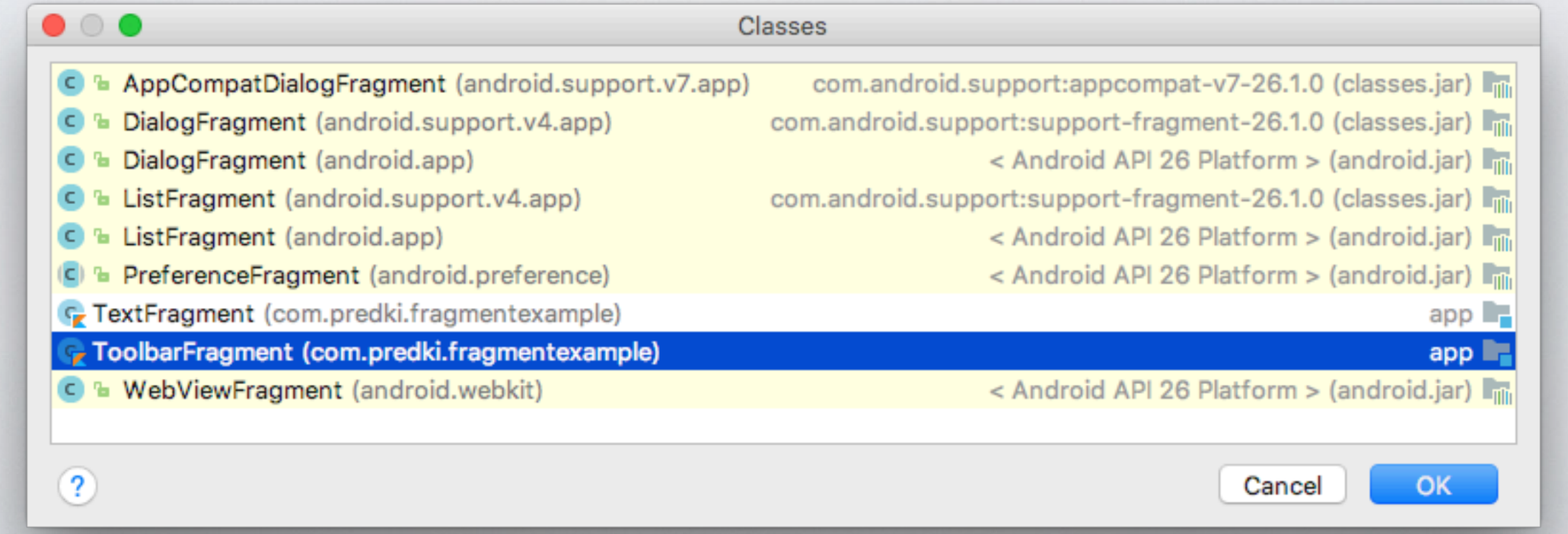
|

- Kod drugiego fragmentu

```
import android.os.Bundle
import android.support.v4.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import kotlinx.android.synthetic.main.text_fragment.*

class TextFragment : Fragment() {

    override fun onCreateView(inflater: LayoutInflater?,
                              container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {

        return inflater?.inflate(R.layout.text_fragment,
                container, attachToRoot: false)
    }

    fun changeTextProperties(fontsize: Int, text: String)
    {
        textView1.textSize = fontsize.toFloat()
        textView1.text = text
    }
}
```

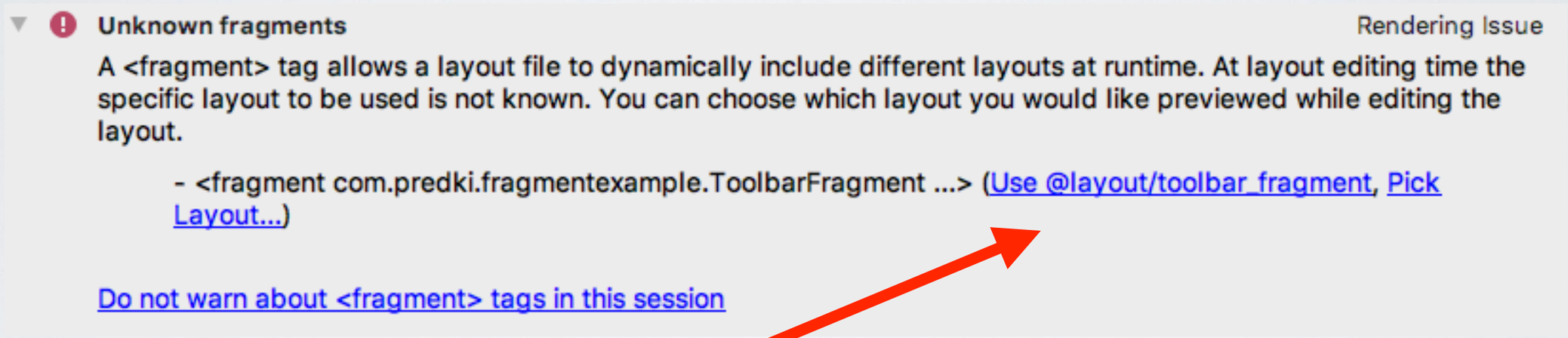
|

- Wracamy do naszej głównej aktywności
- Z grupy Layouts wrzucamy na nasz projekt <fragment>
- Wybieramy z okienka ToolbarFragment



|

- Po kliknięciu w przycisk błędów zobaczymy komunikat:



- Wybieramy to

- Podobnie dodajemy drugi fragment

|

- Nadajemy fragmentom id odpowiednio toolbarFragment i textFragment
- Nasz klasa musi zaimplementować interfejs ToolbarListener i mieć metodę onButtonClick
- Uruchamiamy

```
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.support.v4.app.FragmentActivity
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : FragmentActivity(),
        ToolbarFragment.ToolbarListener {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    override fun onButtonClick(fontsize: Int, text: String) {
        val tf=textFragment as TextFragment
        tf.changeTextProperties(fontsize, text)
    }
}
```

II

- Tworzymy nowy projekt typu Empty Activity i nazywamy go FileAccess

- Do manifestu dodajemy linijkę

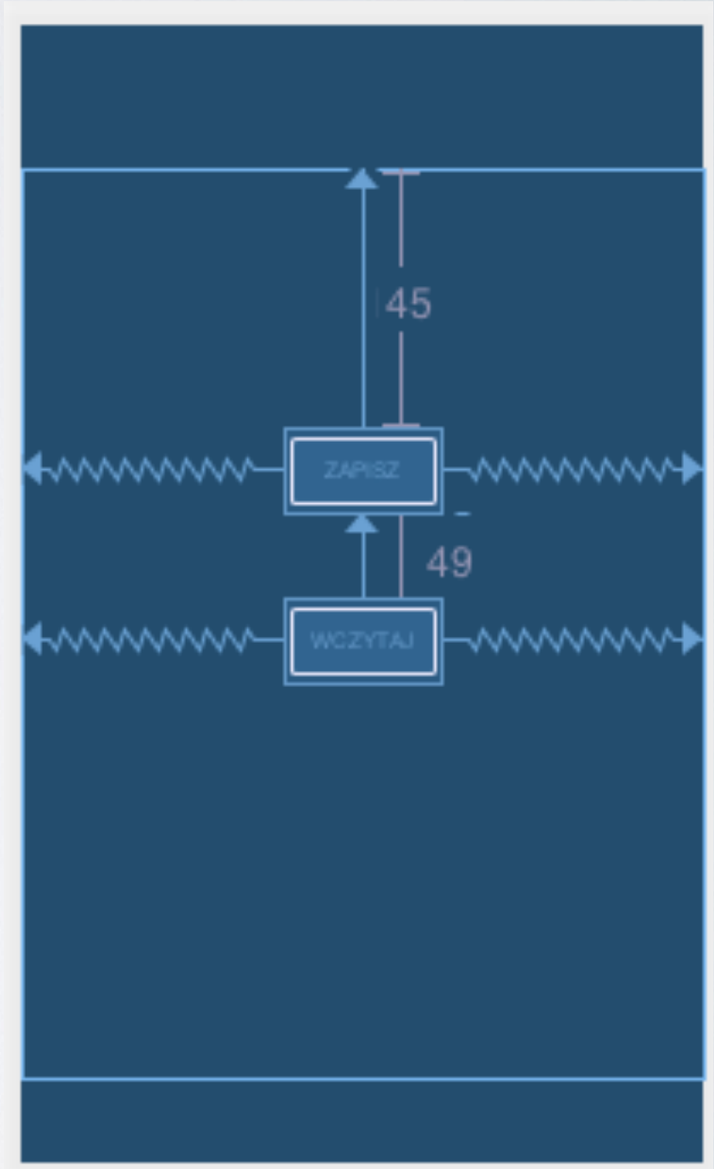
```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

- Do proj act

```
class contact {
    var name: String? = null
    var phone: String? = null

    constructor() {}

    constructor(name:String?, phone:String?)  {
        this.name=name
        this.phone=phone
    }

    constructor(line:String?) {
        if (line!=null) {
            val tokens=line.split( ...delimiters: ";")
            if (tokens.size==2) {
                name = tokens[0]
                phone = tokens[1]
            }
        }
    }

    override fun toString(): String {
        return "Contact $name @ $phone"
    }

    fun toCSV():String {
        return "$name;$phone\n"
    }

}
```

II

- W naszej aktywności definiujemy 2 przyciski Button: Zapisz i Wczytaj
- Definiujemy metody do ich obsługi saveClick i readClick



II

- Dodajemy kod w klasie

```
fun saveClick(v:View) {
    val contacts = Arrays.asList(
            contact( name: "Darth Vader",  phone: "555328431"),
            contact( name: "Luke Skywalker",  phone: "5558946766"))

    val filename = "contacts.csv"
    val file = OutputStreamWriter(openFileOutput(filename, Context.MODE_PRIVATE))


    for (c in contacts) {
        file.write(c.toCSV())
    }
    file.flush()
    file.close()
    Toast.makeText( context: this,  text: "Plik został zapisany", Toast.LENGTH_LONG).show()
}
```

```
private fun FileExists(path:String):Boolean {
    val file = baseContext.getFileStreamPath(path)
    return file.exists()
}
```

||

```
fun readClick(v: View) {
    val contacts = ArrayList<contact>()
    try {
        val filename = "contacts.csv"
        if (FileExists(filename)) {
            val file = InputStreamReader(openFileInput(filename))
            val br = BufferedReader(file)

            var line = br.readLine()
            while (line != null) {
                contacts.add(contact(line))
                line = br.readLine()
            }
            file.close()
            val count = contacts.size
            Toast.makeText( context: this,  text: "Wczytano $count kontaktów", Toast.LENGTH_LONG).show()
        } else
        {
            Toast.makeText( context: this,  text: "Nie znaleziono pliku", Toast.LENGTH_LONG).show()
        }

        } catch (e: Exception) {

    }
}
```

II

- Uruchamiamy
- Po operacji zapis/odczyt możemy podejrzeć plik w AndroidStudio za pomocą Device File Explorer

II

- Przerabiamy kod na External Storage

```
fun saveClick(v:View) {
    val contacts = Arrays.asList(
            contact( name: "Darth Vader",  phone: "555328431"),
            contact( name: "Luke Skywalker",  phone: "5558946766"))

    val filename = "contacts.csv"
    val path = this.getExternalFilesDir( type: null)


    val contactDirectory = File(path,  child: "ContactData")
    contactDirectory.mkdirs()
    val file=File(contactDirectory,filename)

    for (c in contacts) file.appendText(c.toCSV())

    Toast.makeText( context: this,  text: "Plik został zapisany", Toast.LENGTH_LONG).show()
}
```

```
fun readClick(v: View) {
    val contacts = ArrayList<contact>()
    try {
        val filename = "contacts.csv"
        val path = this.getExternalFilesDir( type: null)
        val contactDirectory = File(path,  child: "ContactData")

        if (contactDirectory.exists()) {
            val file=File(contactDirectory,filename)

            if (file.exists()) {
                val br = BufferedReader(file.reader())

                var line = br.readLine()
                while (line != null) {
                    contacts.add(contact(line))
                    line = br.readLine()
                }
                val count = contacts.size
                Toast.makeText( context: this,  text: "Wczytano $count kontaktów", Toast.LENGTH_LONG).show()
            } else {
                Toast.makeText( context: this,  text: "Nie znaleziono pliku", Toast.LENGTH_LONG).show()
            }
        }

        } catch (e: Exception) {

    }
}
```