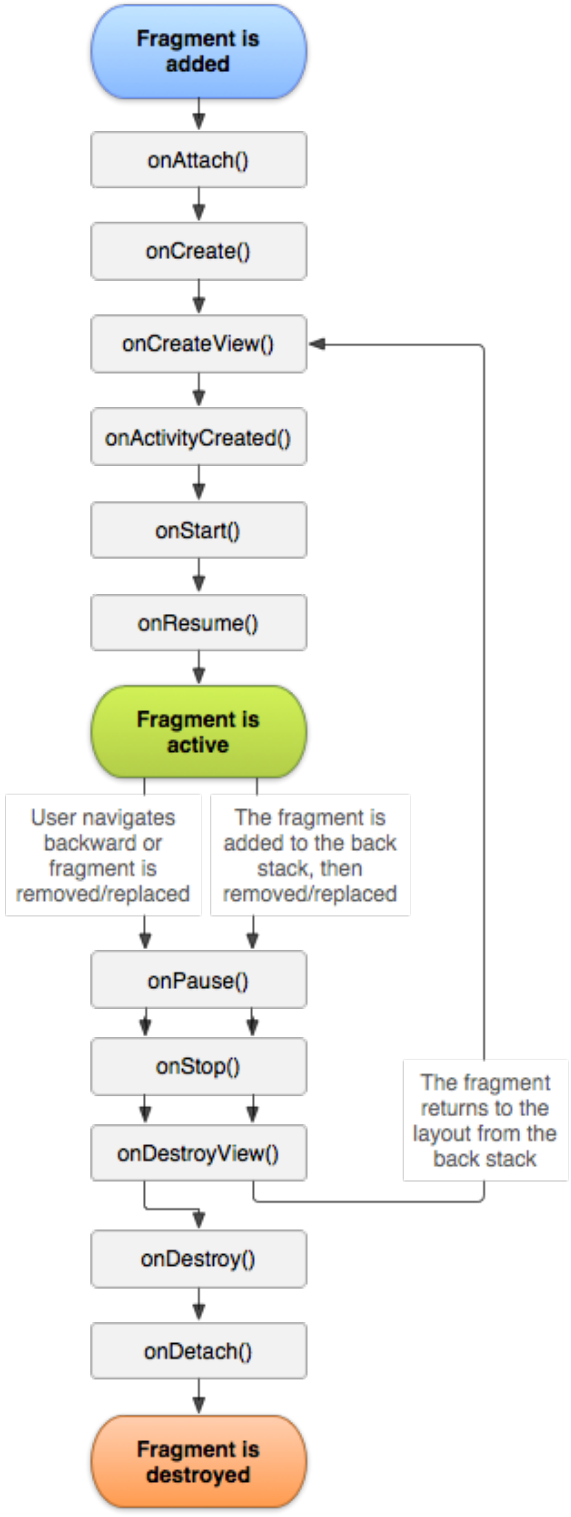
Obsługa fragmentów (fragmenty statyczne)

Fragmenty pozwalają na organizowanie komponentów interfejsu projektu dla różnych urządzeń, dając możliwość wyświetlania wielu segmentów interfejsu na dużym ekranie, np. na tablecie lub wyświetlać jeden i połączyć wszystkie ze sobą na mniejszym ekranie.

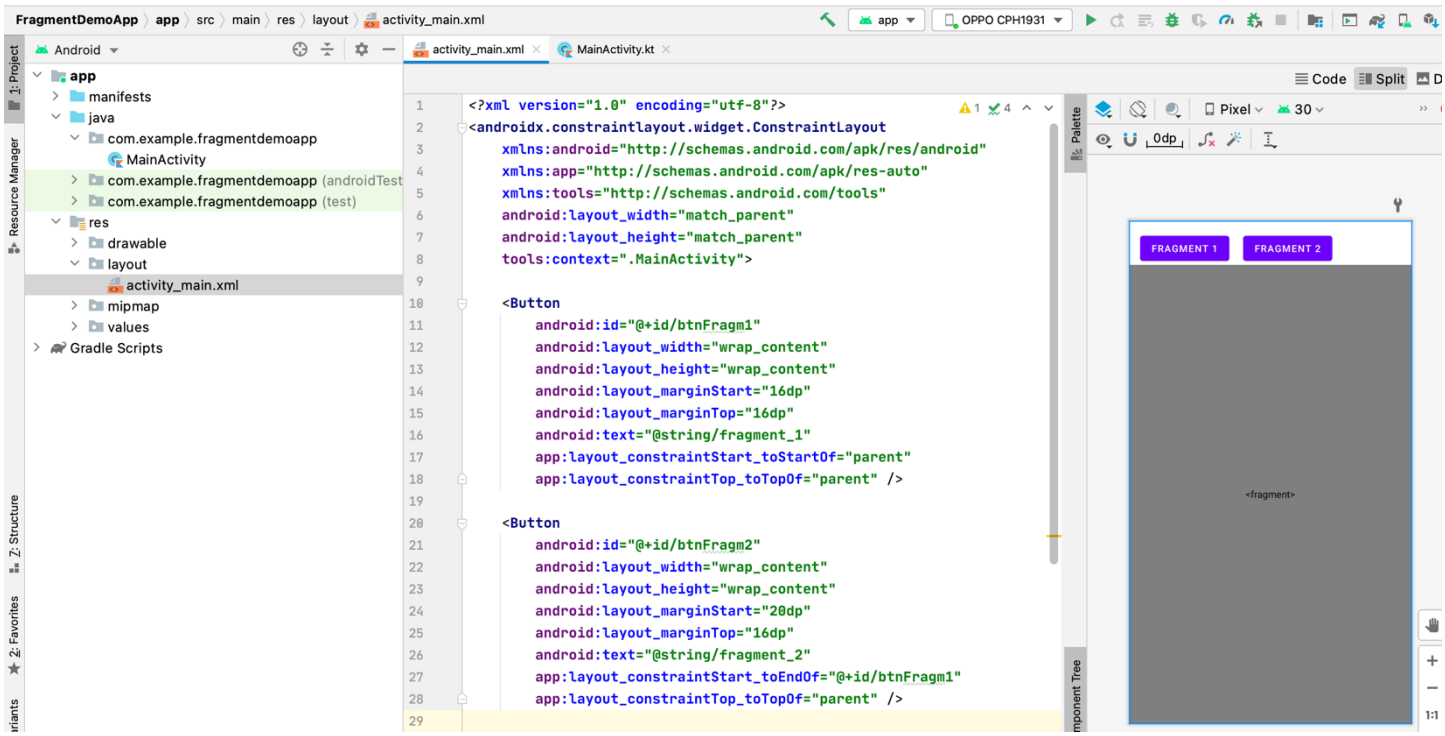
Pomagają również podzielić kod na łatwe do zarządzania kawałki, bez potrzeby polegania na dużych i skomplikowanych klasach Activity. Jedną z ostatnich i prawdopodobnie najbardziej wartościową funkcją jest to, że fragmenty pozwalają na łatwe poruszanie się w aplikacji i umożliwiają proste komunikowanie się między różnymi sekcjami aplikacji.

Podobieństwo fragmentów do aktywności:



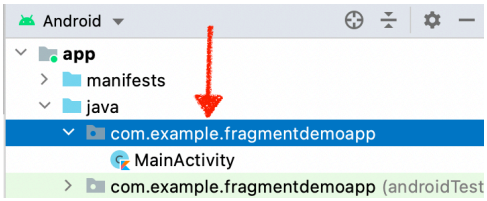
Przykładem zastosowania fragmentów jest kalkulator, który na pionowym ekranie wyświetla podstawowe funkcje, a na dużym ekranie tabletów lub w pozycji poziomej wyświetla rozbudowane menu. Porównanie właściwości ajktywności i fragmentów:

Trzeba jeszcze dodać, że każdy fragment musi być umieszczany w swoim kontenerze. Utwórzmy aplikację FragmentDemoApp z pustej aktywności. Wstawmy do niej dwam przyciski oraz kontener na fragmenty:

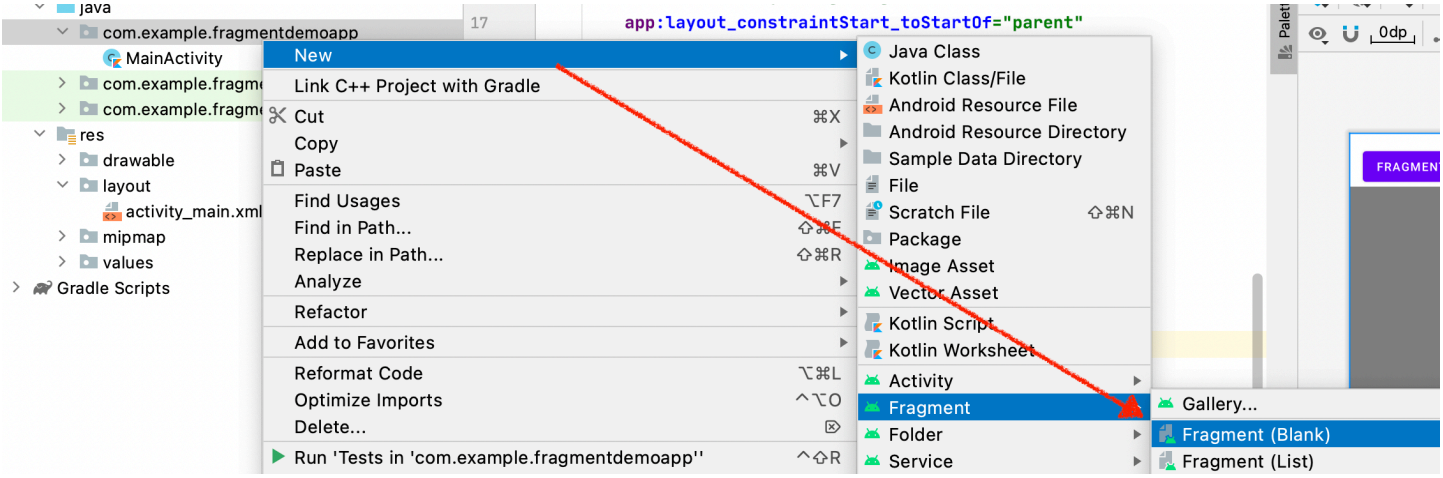



```
30     <fragment
31         android:id="@+id/fragment"
32         android:layout_width="match_parent"
33         android:layout_height="0dp"
34         app:layout_constraintBottom_toBottomOf="parent"
35         app:layout_constraintEnd_toEndOf="parent"
36         app:layout_constraintHorizontal_bias="0.5"
37         app:layout_constraintStart_toStartOf="parent"
38         app:layout_constraintTop_toBottomOf="@+id/btnFragm1">
39
40     </fragment>
41 </androidx.constraintlayout.widget.ConstraintLayout>
```

Utwórzmy teraz fragment klikając prawym przyciskiem myszy w:

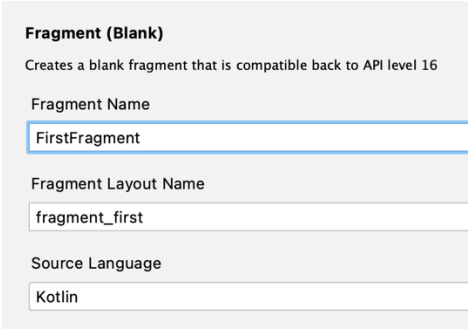


i wybierając pusty fragment:



Oprócz pliku FirstFragment.kt zostanie utworzony polik fragment_first.xml .

Nadajemy mu nazwę:



Modyfikujemy kod XML według wzoru:

activity_main.xml | fragment_first.xml | MainActivity.kt

```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".FirstFragment">

    <TextView
        android:id="@+id/fragment1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/fragment_1"
        android:textAlignment="center"
        android:textSize="30sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Utworzony plik kt

FirstFragment.kt

```kotlin
package com.example.fragmentdemoapp

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup

// TODO: Rename parameter arguments, choose names that match
// the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
private const val ARG_PARAM1 = "param1"
private const val ARG_PARAM2 = "param2"

class FirstFragment : Fragment() {
    // TODO: Rename and change types of parameters
    private var param1: String? = null
    private var param2: String? = null

    override fun onCreate(savedInstanceState: Bundle?) {...}

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

    companion object {...}
}
```

należy dostosować do swoich potrzeb:

FirstFragment.kt | SecondFragment.kt | fragment_second.xml | fragment_fir

```kotlin
package com.example.fragmentdemoapp

import androidx.fragment.app.Fragment

class FirstFragment : Fragment(R.layout.fragment_first) {

}
```

W identyczny sposób tworzymy fragment 2 (SecondFragment.kt):

FirstFragment.kt | SecondFragment.kt | fragment_second.xml | fragment_first

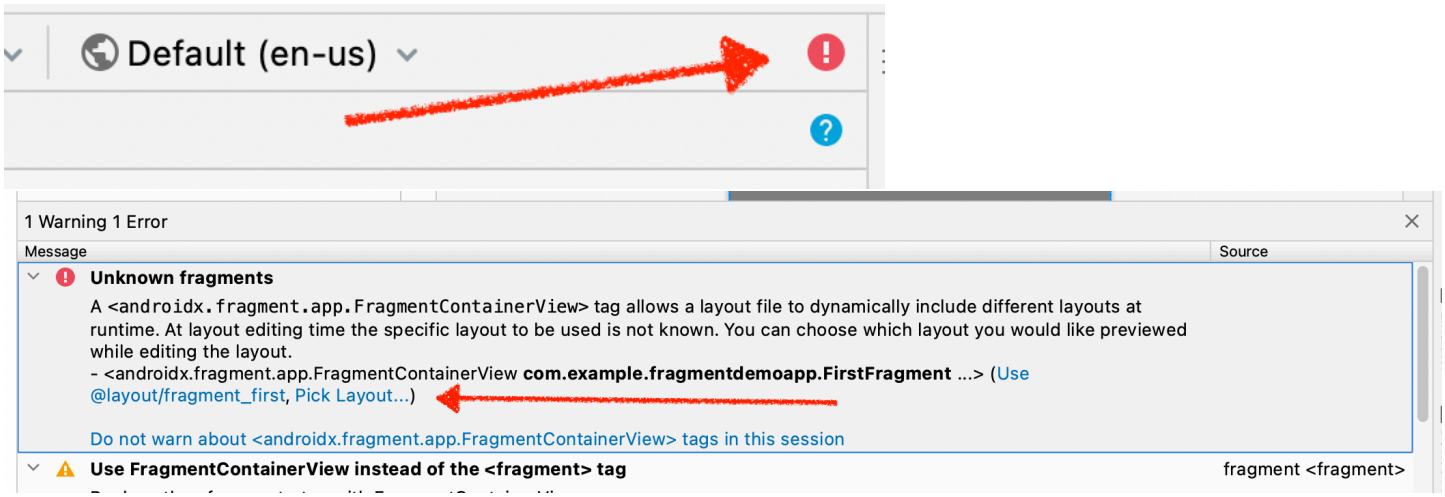
```kotlin
package com.example.fragmentdemoapp

import androidx.fragment.app.Fragment

class SecondFragment : Fragment(R.layout.fragment_second) {

}
```

Aby. Fragmenty były rozpoznawane należy je uaktywnić wybierając:



Teraz możemy zdecydować, który fragment ma być uruchamiany jako startowy: dopisujemy linię:

activity_main.xml

```
31      <fragment
32          android:name="com.example.fragmentdemoapp.FirstFragment"
33          android:id="@+id/fragment"
```

Oraz modyfikujemy znacznik na FrameLayout:

activity_main.xml

~/AndroidStudioProjects/FragmentDemoApp/app/src/main/res/layout/activity_main.xml

```
31      <FrameLayout
32          android:name="com.example.fragmentdemoapp.FirstFragment"
33          android:id="@+id/fragment"
34          android:layout_width="match_parent"
35          android:layout_height="0dp"
36          app:layout_constraintBottom_toBottomOf="parent"
37          app:layout_constraintEnd_toEndOf="parent"
38          app:layout_constraintHorizontal_bias="0.5"
39          app:layout_constraintStart_toStartOf="parent"
40          app:layout_constraintTop_toBottomOf="@+id/btnFragm1"
41          tools:layout="@layout/fragment_first"/>
42
```

Teraz zmieniamy zawartość pliku głównej aktywności MainActivity.kt wg. wzoru:

MainActivity.kt

```kotlin
package com.example.fragmentdemoapp

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import com.example.fragmentdemoapp.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        //setContentView(R.layout.activity_main)
        setContentView(binding.root)
        val firstFragment:FirstFragment = FirstFragment()
        val secondFragment:SecondFragment = SecondFragment()
        supportFragmentManager.beginTransaction().apply { this: FragmentTransaction
            replace(R.id.fragment,firstFragment).commit()
        }
        binding.btnFragm1.setOnClickListener{ it: View!
            supportFragmentManager.beginTransaction().apply { this: FragmentTransaction
                replace(R.id.fragment,firstFragment)
                addToBackStack( name: null)
                commit()
            }
        }
        binding.btnFragm2.setOnClickListener{ it: View!
            supportFragmentManager.beginTransaction().apply { this: FragmentTransaction
                replace(R.id.fragment,secondFragment)
                addToBackStack( name: null)
                commit()
            }
        }
    }
}
```

Oraz uruchomić aplikację

Rozbudujemy zachowanie aplikacji w taki sposób aby reagowała na obrót ekranu:

W tym celu do budujemy funkcję IF do MainActivity.kt wg. Wzoru:

```
17        val firstFragment: FirstFragment = FirstFragment()
18        val secondFragment: SecondFragment = SecondFragment()
19        //reakcja statyczna na obrót ekranu
20        if (ActivityInfo.SCREEN_ORIENTATION_SENSOR_PORTRAIT == Configuration.ORIENTATION_PORTRAIT) {
21                supportFragmentManager.beginTransaction().apply { this: FragmentTransaction
22                    replace(R.id.fragment, firstFragment)
23                    addToBackStack( name: null)
24                    commit()
25                }
26        } else {
27                supportFragmentManager.beginTransaction().apply { this: FragmentTransaction
28                    replace(R.id.fragment, secondFragment)
29                    addToBackStack( name: null)
30                    commit()
31                }
32        }
33        //-- zachowanie przycisków
34        supportFragmentManager.beginTransaction().apply { this: FragmentTransaction
```