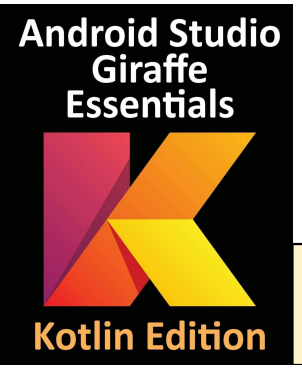
Fragment

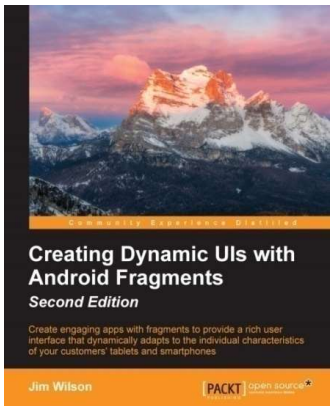
Peter Borovanský
KAI, I-18

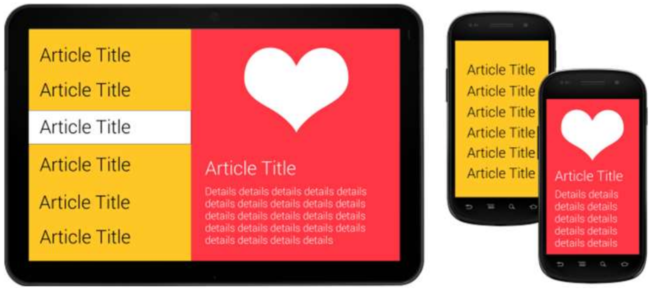
MS-Teams: 2sf3ph4, List, github

borovan 'at' ii.fmph.uniba.sk



Kap. 37 An Introduction to Android Fragments
Kap. 38 Using Fragments in Android Studio

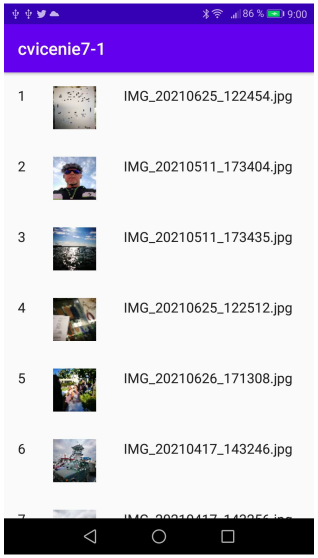




O čom to dnes bude

- **Fragment** ako základný stavebný kameň zložitejšej aplikácie
  - fragment je samostatne existujúca časť (modul) aplikácie majúca svoj layout aj správanie
  - layout má definovaný v .xml
  - princípy fungovania–fragment má tiež životný cyklus, je komplikovanejší ako ho má aktivita
  - každý fragment je podtrieda Fragment() a vkladá sa do aktivity, tzv. FragmentActivity
  - jednoduché používanie existujúcich (už hotových) Dialog Fragmentov ilustrované v závere..

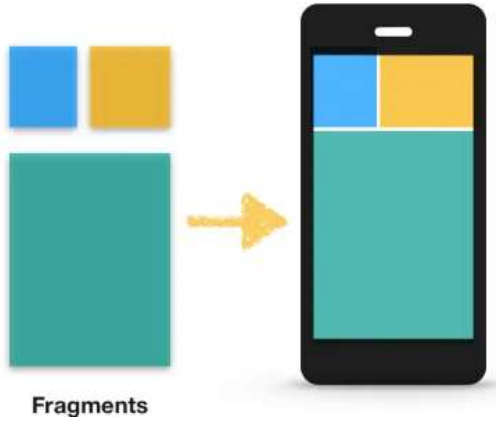
- **Master-Detail** aplikácia (Primary/Detail Flow)
  - Master je napr. zoznam všetkých objektov, Detail je detail jedného z nich
  - Cvičenie 6:



Na budúce:

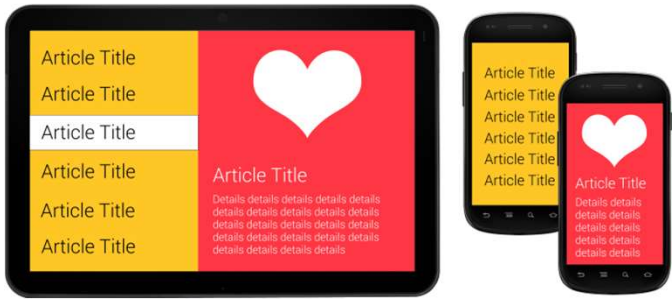
- Návrhové vzory
  - Model View Controller (MVC)
  - **Model View ViewModel** (MVVM)
  - **LiveData**
  - **JetPack v AndroidX** (androidx.* packages)

Fragmenty



- **fragment** predstavuje ucelenú časť GUI, podobne ako aktivita
- fragment má, podobne ako aktivita, životný cyklus, ale zložitejší
- hlavným cieľom fragmentu je jeho znovu-použiteľnosť (reusability)
- každý fragment má svoju aktivitu, ktorá si ho pri inicializácii pripojí (**attach**)
  - aktivita si vkladá do seba jeden, alebo viac fragmentov, ktoré navyše môžu komunikovať
- koexistencia fragmentu a aktivity je zložitejšia ako život aktivity
- vzťah fragment-aktivita je typu **many-many**
  - fragment môže byť použitý v rôznych aktivitách (o tom je reusabilita fragmentu)
  - a jedna aktivita často obsahuje viacero fragmentov, ktoré sa nejakým spôsobom prepínajú
  - pri prepínaní fragmetov často treba riešiť prenos dát medzi nimi (teda komunikáciu)
- aktivita môže obsahovať/kombinovať viacero fragmentov, dvomi spôsobmi
  - **staticky** (sú navrhnuté a staticky vložené v layout .xml-súbore aktivity)
  - **dynamicky** (vzniknú dynamicky v kóde pomocou konštruktora podtriedy Fragmentu)

Fragmenty
(história a následky)



- fragmenty sú podporované od Android 3.1 (API 11)
- ~~ak naše minSDK < 11, použijeme Support Library~~
  https://developer.android.com/topic/libraries/support-library/index.html

- historicky knižnice podporujúce Fragment sú:
  - ~~`android.app.Fragment`~~ (This class was deprecated in API level 28)
  - ~~`android.support.v4.app`~~ (od API 26-July,2017, min.API level 14)
  - a najnovšie Android Jetpack, balíky **`androidx.*`** od Android 9.0 (API level 28)

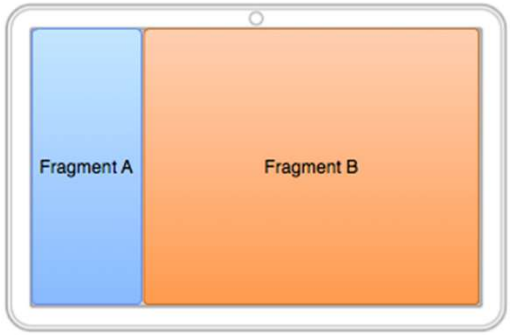
Pozor na miešanie importov z rôznych knižníc:

- `android.app.Fragment`
- `!= android.support.v4.app.Fragment`
- `!= androidx.fragment.app.Fragment`

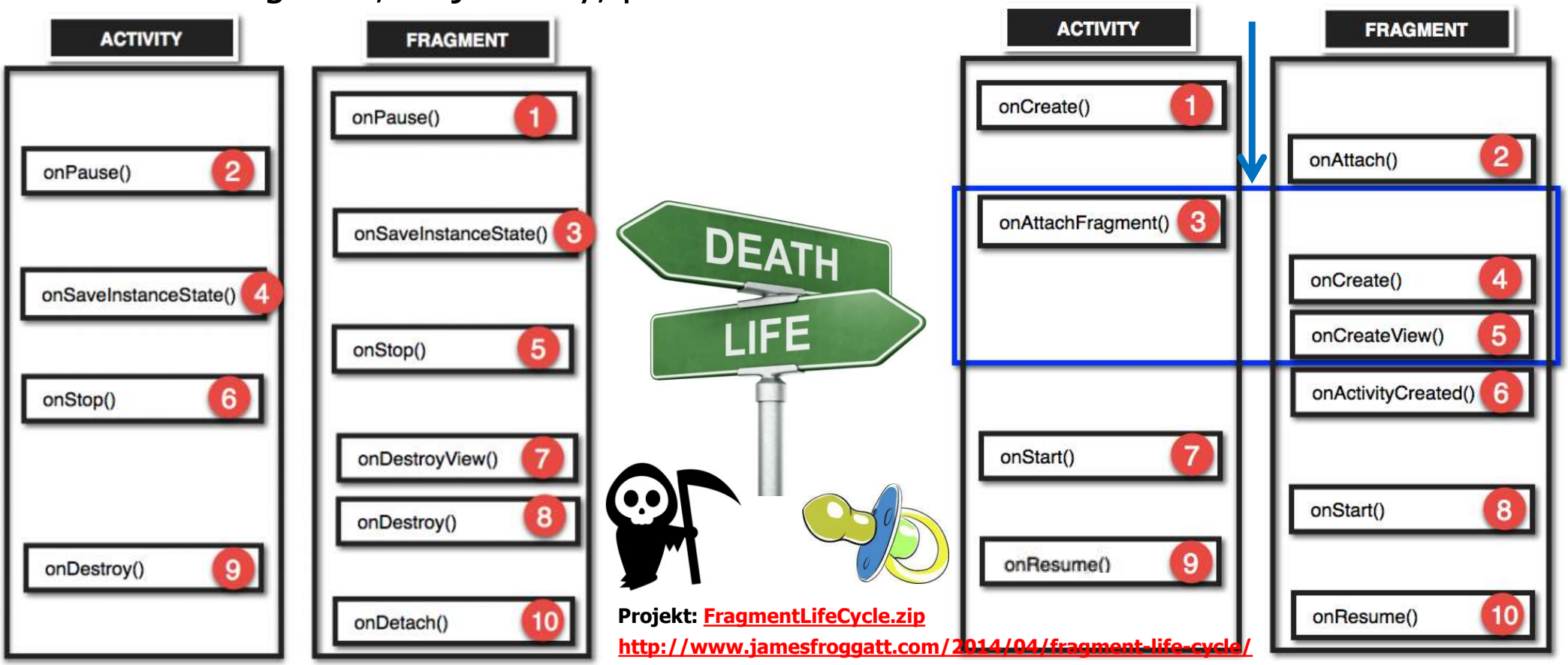
- Stavy fragmentu (životný cyklus extrémne stručne):
  - definujeme podtriedu triedy Fragment, kým nezavoláme konštruktor, tak *neexistuje nič !*
  - po FragmentSubClass(), existuje síce inštancia fragmentu ako objekt, *nevidíme nič !*
  - aktivita pripojí (*attachne*) fragment, *nevidíme nič*, ale aspoň fragment vie, že má aktivitu
  - fragment sa zobrazí na obrazovke, a *vidíme ho a existuje*

Život fragmentu

(je zložitejší ako u aktivity)



- fragment predstavuje ucelenú časť GUI, podobne ako aktivita
- fragment má svoju aktivitu, ktorá ho pripojí (predpokladajme vzťah 1:1)
- …aktivita môže obsahovať/kombinovať (aj dynamicky) viacero fragmentov
- fragment, ak je dobrý, používa ho viacero aktivít (reusability)

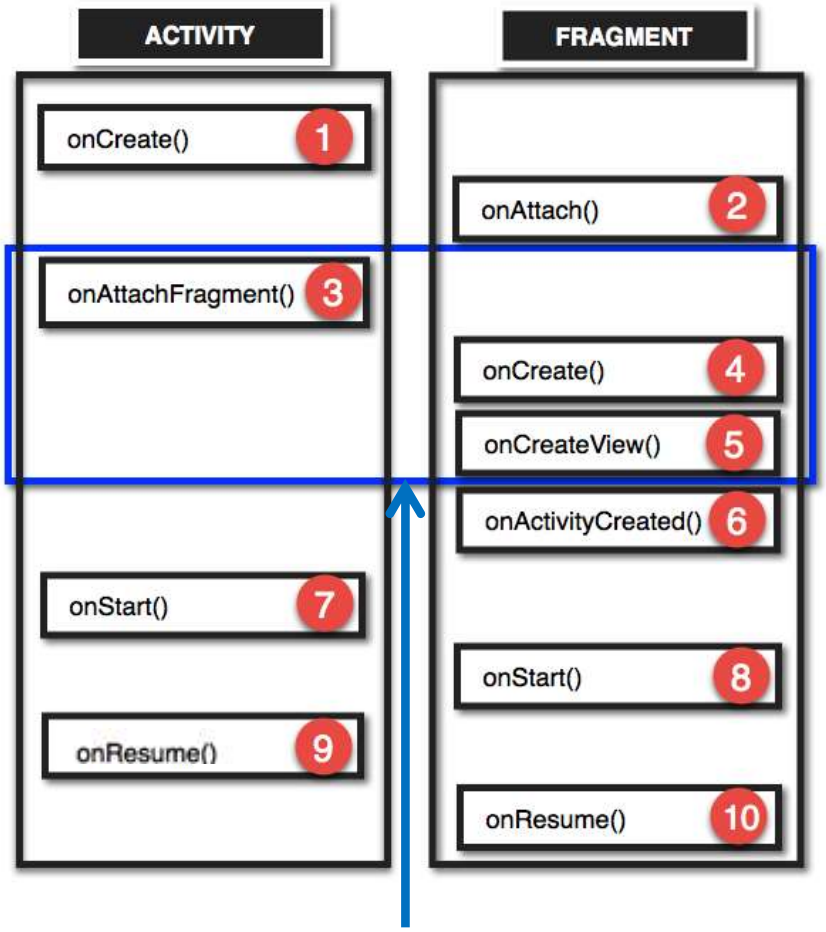


Projekt: FragmentLifeCycle.zip

http://www.jamesfroggatt.com/2014/04/fragment-life-cycle/

Vznik fragmentu

(venujme sa vzniku, nie zániku)



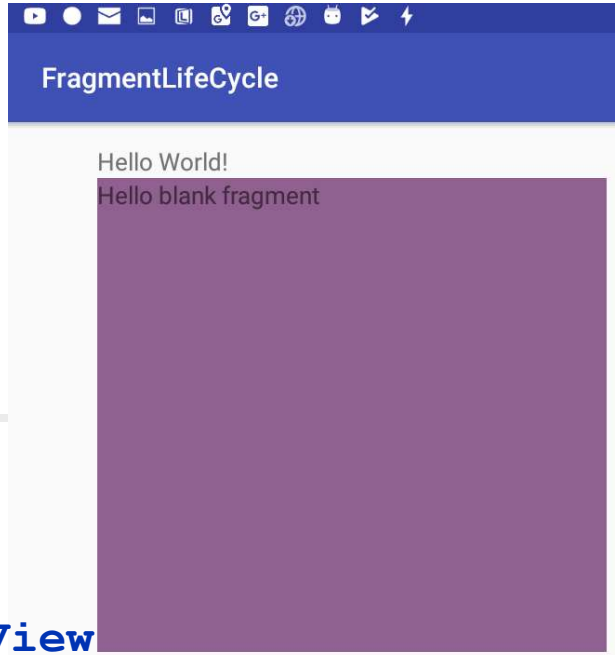
1. onCreate v activite: Najčastejšie obsahuje setContentView, ktorá definuje layout aktivity
2. onAttach vo fragmente: dostaneme pointer na aktivitu, do ktorej je vkladaný, uložíme si ho...
3. onAttachFragment v aktivite: dozvie sa, že fragment bol attach-nutý do aktivity
4. onCreate vo fragmente: aktivity onCreate nemusí byť ukončená, preto nie je dovolené adresovať UI komponenty z aktivity
5. onCreateView vo fragmente: fragmentu určíme layout, inflater (nafukovač) inflatuje
6. onActivityCreated vo fragmente: už konečne vidíme UI komponenty aj z aktivity
7. onStart v aktivite
8. onStart vo fragmente
9. onResume v aktivite
10. onResume vo fragmente

http://www.jamesfroggatt.com/2014/04/fragment-life-cycle/

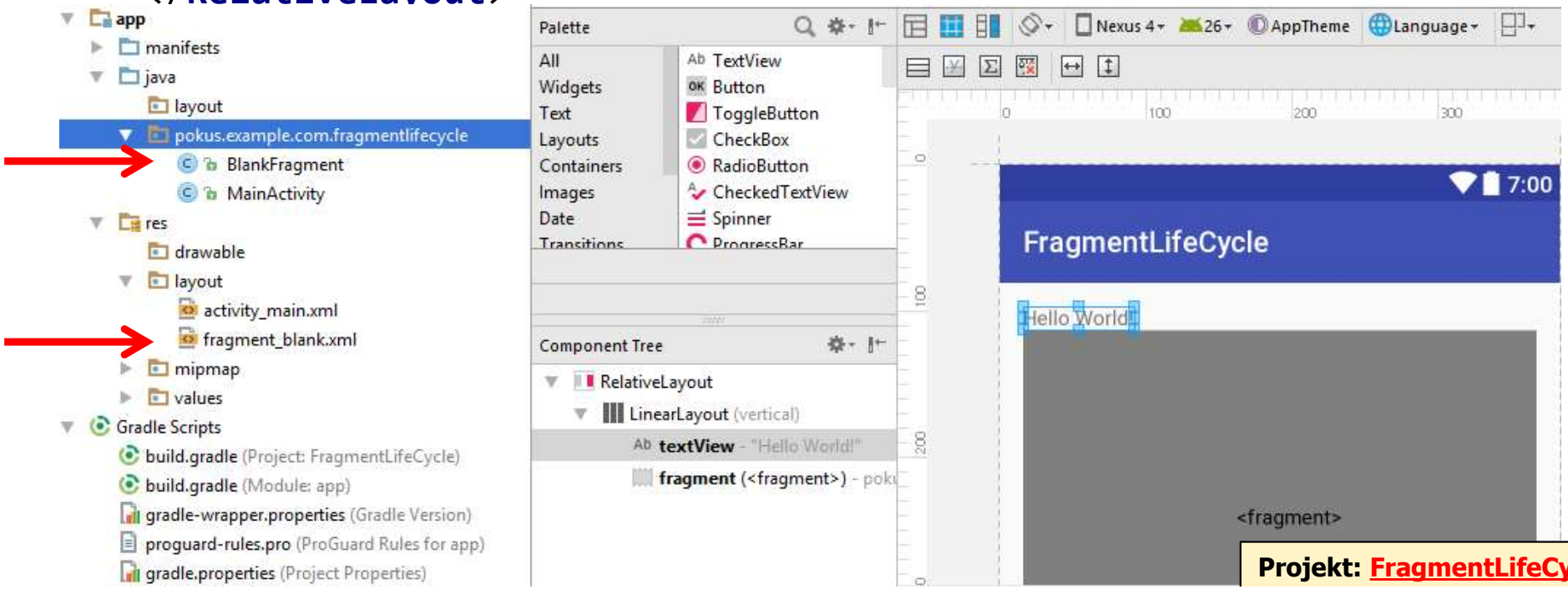
Projekt: FragmentLifeCycle.zip

Život fragmentu

(jeden fragment v aktivite)




```
<RelativeLayout
  <LinearLayout>
    <TextView ...android:text="Hello World!"/>
    <androidx.fragment.app.FragmentContainerView
      android:id="@+id/fragment"
      android:name="com.example.fragmentlifecycle.BlankFragment"/>
   </LinearLayout>
 </RelativeLayout>
```



Projekt: FragmentLifeCycle.zip

Život fragmentu
(onSaveInstance)

- napr. zmena orientácie displaya
- ak fragment/aktivita zaniká, môžeme si zapamäť jej stav cez **Bundle** v onSaveInstanceState a obnoviť v onCreate


```
override fun onSaveInstanceState(
                savedInstanceState? : Bundle) {
    super.onSaveInstanceState(savedInstanceState);
    savedInstanceState?.putString("key", "value")
    savedInstanceState?.putInt("score", ...)
    savedInstanceState?.putLong("time", ...)
    ... }
```

- a následne reštaurovať:

```
override fun onCreate(savedInstanceState:Bundle) {
    super.onCreate(savedInstanceState);
    savedInstanceState.getString("key")
    savedInstanceState.getInt("score")
    savedInstanceState.getLong("time")
    ... }
```

```
on Attach
on Create
on CreateView
on Activity Created
on Start
on Resume
on Pause
on Save Instance State
on Stop
on Destroy View
on Destroy
on Detach
on Attach
on Create
on CreateView
on Activity Created
on Start
on Resume
```

bez **onSaveInstanceState**

```
on Pause
on Stop
on Destroy View
on Destroy
on Detach
```

Projekt: FragmentLifeCycle.zip

Život fragmentu

(detail)

```
on Create ACTIVITY
on Attach Fragment
on Create Fragment
on CreateView Fragment
on Activity Created Fragment
on Start ACTIVITY
on Start Fragment
on Resume ACTIVITY
on Resume Fragment
```

Zmena orientácie

```
on Pause Fragment
on Pause ACTIVITY
on Save Instance State Fragment
on Save Instance State ACTIVITY
on Stop Fragment
on Stop ACTIVITY
on Destroy View Fragment
on Destroy Fragment
on Detach Fragment
on Destroy ACTIVITY
on Create ACTIVITY
on Attach Fragment
on Create Fragment
on CreateView Fragment
on Activity Created Fragment
on Start ACTIVITY
on Start Fragment
on Restore Instance State ACTIVITY
on Resume ACTIVITY
on Resume Fragment
```

home screen

```
on Pause Fragment
on Pause ACTIVITY
on Save Instance State Fragment
on Save Instance State ACTIVITY
on Stop Fragment
on Stop ACTIVITY
```

restart

```
on Restart ACTIVITY
on Start ACTIVITY
on Start Fragment
on Resume ACTIVITY
on Resume Fragment
```

keď aktivitu/fragment dáme na pozadie ▢ tak sa:

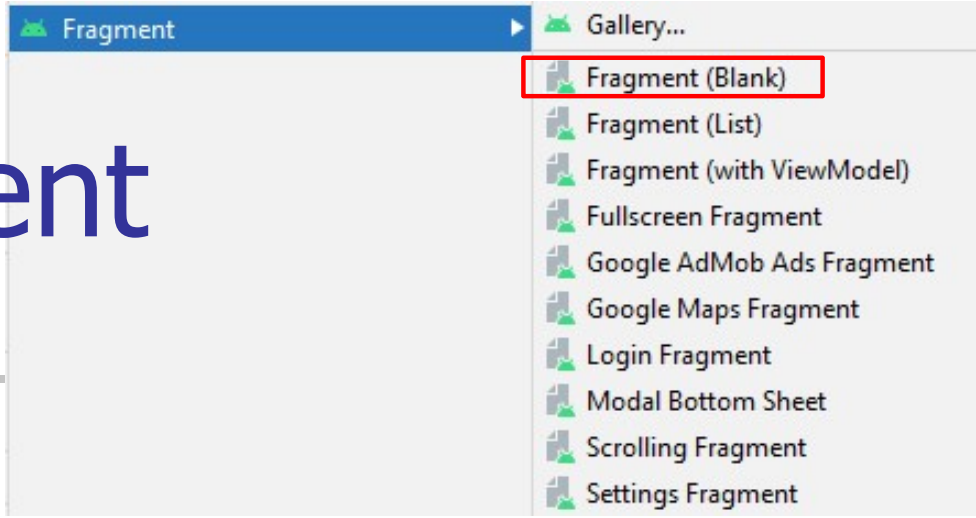
- nevolá **`onDestroy`**,
- pri opätovnom spustní sa nevolá **`onCreate`**, ale **`onRestart`**

eload=10&spfreload=10&spfreload=10#t=264.85907

Projekt: FragmentLifeCycle.zip

Statický fragment

(existuje jeho layout)



- vytvoríme podtriedu Fragment
- AS nám pomôže File/New/Fragment
- vytvoríme dva fragmenty First/Second fragment, a rôzne ofarbíme ich

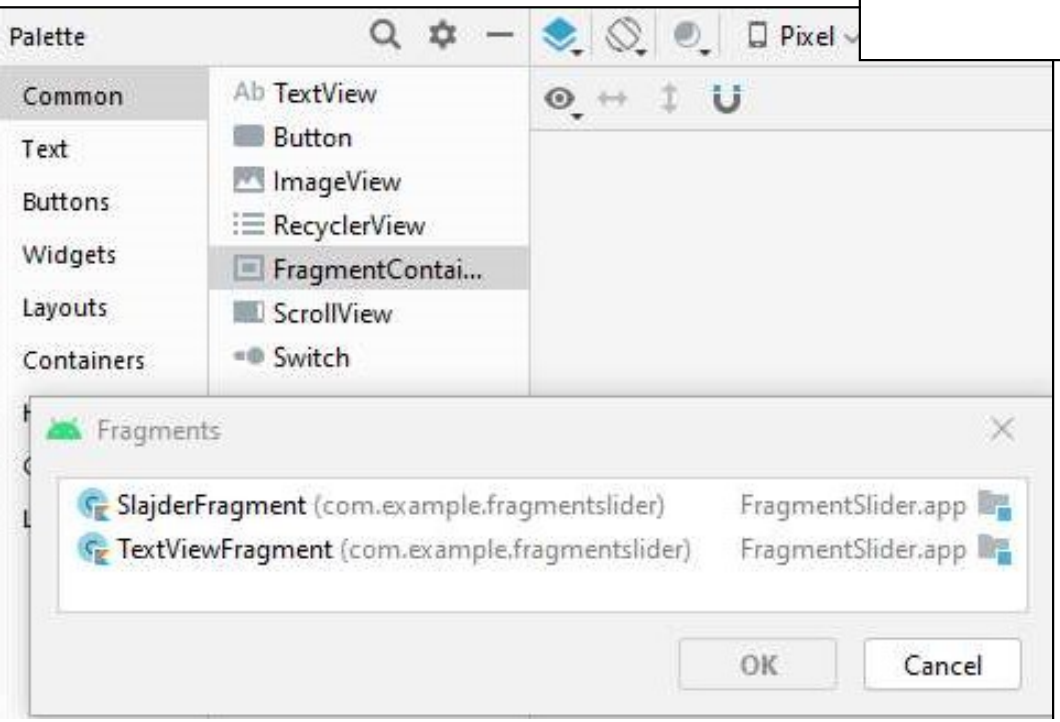
fragment_first.xml

```
<FrameLayout 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"
    tools:context="pokus.example.com.fragmentstaticky.FirstFragment">
    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/colorAccenty"
        android:text="Hello from fist fragment" />
</FrameLayout>
```

Projekt: FragmentStaticky.zip

Statický fragment

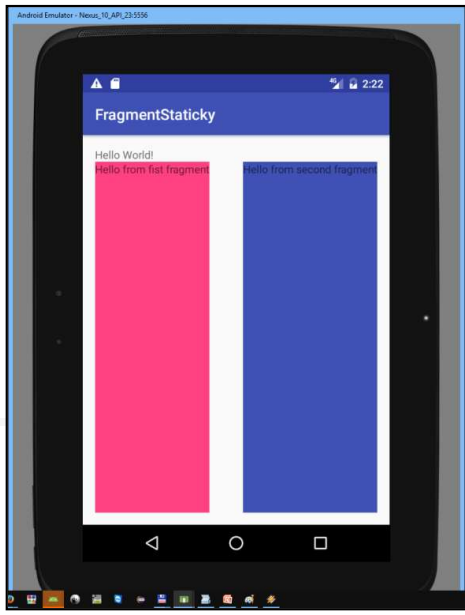
Keď potom editujeme layout aktivity, môžeme doň vložiť FragmentContainterView a v ponuke nájdeme nami vytvorené fragmenty



Projekt: FragmentStaticky.zip

Statický fragment

(jednoduchá verzia – na pochopenie)

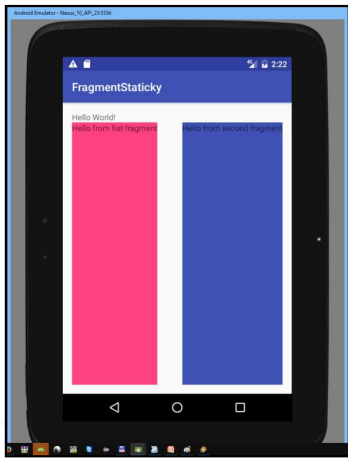



```
class FirstFragment : Fragment() {
    lateinit var mainActivity: MainActivity

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }
        // onCreateView: fragmentu určíme layout, inflater inflatuje
    override fun onCreateView(inflater: LayoutInflater,
                              container: ViewGroup?,
                              savedInstanceState: Bundle?): View {
        return inflater.inflate(R.layout.fragment_first,
                                container, false)
    }
    // onAttach vo fragmente: dostaneme pointer na aktivitu, do ktorej je vkladaný,
        uložíme si ho...
    override fun onAttach(context: Context) {
        super.onAttach(context)    // vhodné si uložiť materskú aktivitu
        mainActivity = context as MainActivity // príde v prem.context
    }
}
```

Projekt: FragmentStaticky.zip

Statický fragment

(verzia s view binding)




```
import com.example.fragmentstaticky.databinding.FragmentFirstBinding
class FirstFragment : Fragment() {
    lateinit var mainActivity: MainActivity
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }
    private lateinit var binding: FragmentFirstBinding
        // onCreateView: fragmentu určíme layout, inflater inflatuje
    override fun onCreateView(inflater: LayoutInflater,
                              container: ViewGroup?,
                              savedInstanceState: Bundle?): View {
        //return inflater.inflate(R.layout.fragment_first, container,
        binding = FragmentFirstBinding.inflate(inflater, container,
                                               false)
        return binding.root
    }
    override fun onAttach(context: Context) {
        super.onAttach(context)     // vhodné si uložiť materskú aktivitu
        mainActivity = context as MainActivity // príde v prem.context
}     }
```

Projekt: FragmentStaticky.zip

Statický fragment

(verzia s view binding bez memory leak)

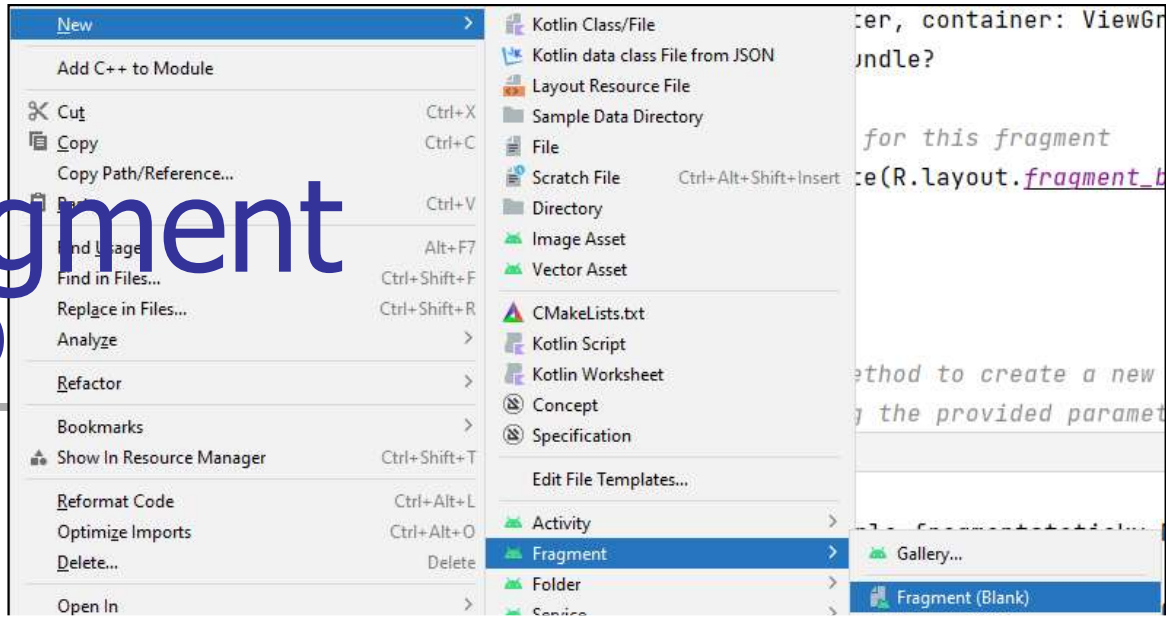
```
class FirstFragment : Fragment() {
    lateinit var mainActivity: MainActivity
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }
    private var _binding: FragmentFirstBinding? = null
    private val binding get() = _binding!!
    override fun onCreateView(inflater: LayoutInflater, container: Vi
                              savedInstanceState: Bundle?): View? {
        _binding = FragmentFirstBinding.inflate(inflater, container, f
        return binding.root
}
override fun onDestroy() {
    super.onDestroy()
    _binding = null
}
override fun onAttach(context: Context) {
    super.onAttach(context)
    mainActivity = context as MainActivity
}
```

once the fragment then invokes its onDestroyView() callback **all references to the fragment's view should be removed**, allowing the fragment's view to be garbage collected

Projekt: FragmentStaticky.zip

Statický fragment

(prázdny - vygenerovaný)




```
private const val ARG_PARAM1 = "param1"
private const val ARG_PARAM2 = "param2" // raz mená vašich parametrov

class BlankFragment1 : Fragment() {
    private var param1: String? = null // premenné, kam sa načítajú
    private var param2: String? = null // zjednotušene, nech sú String

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //arguments : Bundle?
        param1 = arguments?.getString(ARG_PARAM1) // tu sa načítajú
        param2 = arguments?.getString(ARG_PARAM2)
    }
```

Projekt: FragmentStaticky.zip

Statický fragment
(reálne dostanete)

Companion object definuje statickú metódu newInstance
dostane argumenty, ktoré nastaví do parametrov

```
companion object {
        /**
         * Use this factory method to create a new instance of this fragment using the provided parameters.
         * @param param1 Parameter 1.
         * @param param2 Parameter 2.
         * @return A new instance of fragment Frag1.
         */
    @JvmStatic
    fun newInstance(param1: String, param2: String) =
            BlankFragment1().apply {
                arguments = Bundle().apply {
                    putString(ARG_PARAM1, param1)
                    putString(ARG_PARAM2, param2)
}           }     }
```

inštanciu fragmentu by ste vyrobili:

```
val bf = BlankFragment1.newInstance("value1", "value2")
```

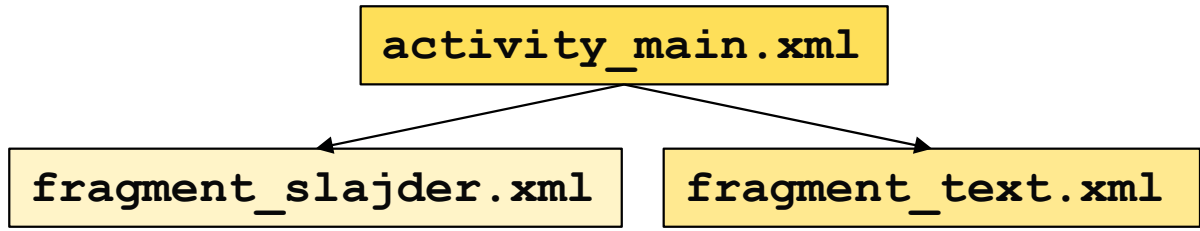
Projekt: FragmentStaticky.zip

Statický fragment
(interakcia s aktivitou)

```
class MainActivity : AppCompatActivity()
    ,
    OnFragmentInteractionListener {
    override fun
        onFragmentInteraction(uri: Uri) {
        TODO()
    }
```

definujete **akýkoľvek** listener na komunikáciu s aktitou

```
interface OnFragmentInteractionListener {
    fun onFragmentInteraction(uri: Uri)
}
// definujete premennú, kam si uložíte pointer na rodičovskú aktivitu,
// ktorá musí implementovať váš listener
private var listener: OnFragmentInteractionListener? = null
private lateinit var listener: OnFragmentInteractionListener
fun onButtonPressed(uri: Uri) {
    listener?.onFragmentInteraction(uri)
}
override fun onAttach(context: Context) {
    super.onAttach(context) // aktivita, ktorá ho attachuje, musí
    if (context is OnFragmentInteractionListener) { // spĺňať
        listener = context // interface, a uložíte si pointer na ňu
    } else {  // inak fail
        throw RuntimeException(context.toString() +
          " must implement OnFragmentInteractionListener")
    } alebo inak listener =
}     context as? OnFragmentInteractionListener
```



Slajder Fragment

fragment_slajder.xml

```
<RelativeLayout >

  <EditText
    android:id="@+id/editText"
    ...
  />

  <SeekBar
    android:id="@+id/seekBar"
    ...
  />

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

fragment_text.xml

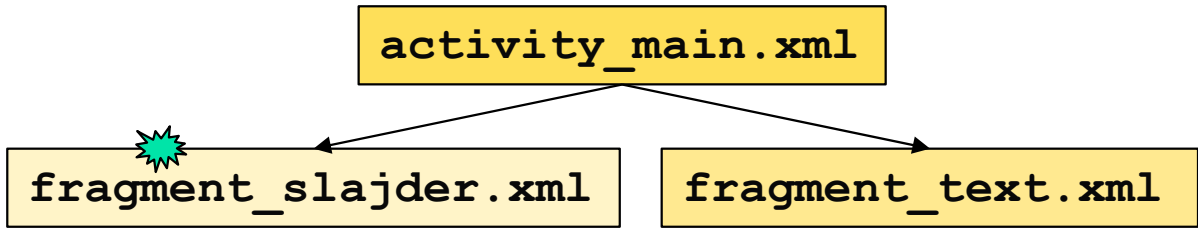
```
<RelativeLayout >

    <TextView
        android:id="@+id/textView"
        ...
    />
</RelativeLayout>
```

Statická kompozícia

activity_main.xml

```
<RelativeLayout >
 <androidx.fragment.app.FragmentContainerView
   android:id="@+id/fragmentSlajder"
 />
 <androidx.fragment.app.FragmentContainerView
   android:id="@+id/fragmentTextView"
 />
</RelativeLayout>
```

Projekt: FragmentSlider.zip



Slajder Fragment

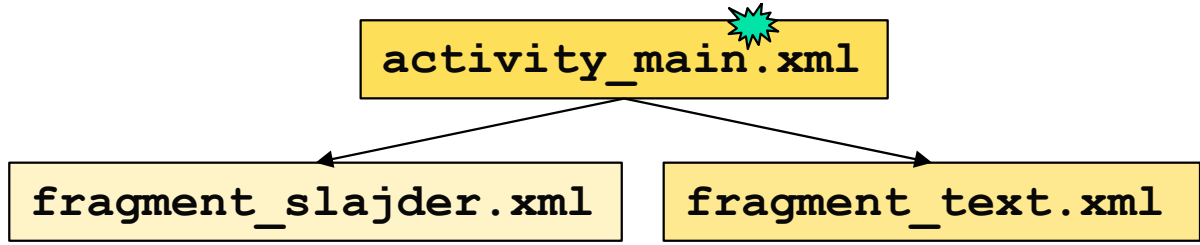
```
class SlajderFragment : Fragment() {
    var slajder = 50
    private lateinit var binding: FragmentSlajderBinding
    interface Listener {
        fun onButtonClick(position: Int, text : String)
    }
    lateinit var activityCallBack : Listener
    override fun onAttach(context: Context) {
        super.onAttach(context)
        try { activityCallBack = context as Listener
        } catch (e : ClassCastException) {
            throw ClassCastException(context.toString() + " does not implement Listener")
        }
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
      super.onViewCreated(view, savedInstanceState)
      binding.apply {
        seekBar.setProgress(slajder)
        seekBar.setOnSeekBarChangeListener (
          object : SeekBar.OnSeekBarChangeListener {
              override fun onProgressChanged(sb: SeekBar, progress: Int, fromUser: Boolean) {
                  slajder = progress
              }
         })
         button.setOnClickListener{ v ->
            activityCallBack.onButtonClick(slajder, editText.text.toStrin
}    } }
```

... vnorený interface **požiadavky na attachera** ... požiadavky na aktivitu

attacher musí implementovať interface Listener

attacher musí implementovať onButtonClick

Projekt: FragmentSlider.zip



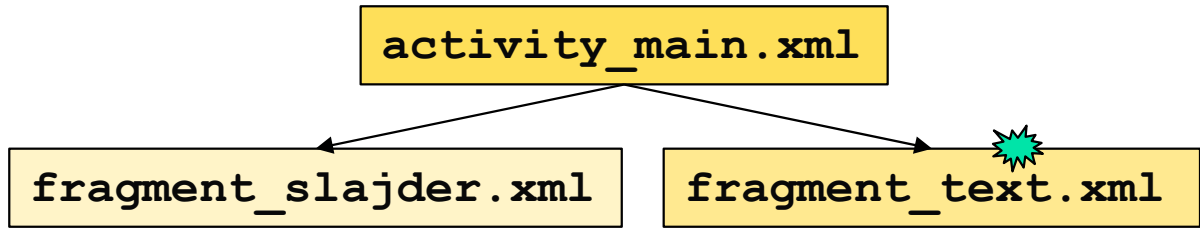
Slajder Fragment

MainActivity staticky obsahuje SlajderFramgent, aj TextViewFragment

```
class MainActivity : FragmentActivity(), SlajderFragment.Listener {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    override fun onButtonClick(fontSize: Int, text: String) {
        val textViewFragment =
            supportFragmentManager.findFragmentById(
                R.id.fragmentTextView) as TextViewFragment
        textViewFragment.changeText(fontSize, text)
    }
}
```

attacher spĺňa požiadavky, implementuje interface

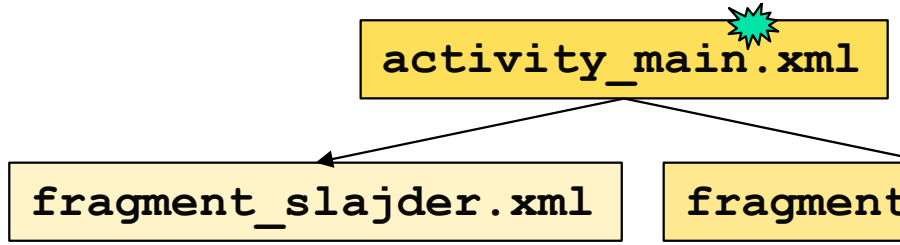
Projekt: FragmentSlider.zip



TextView Fragment

```
class TextViewFragment : Fragment() {
    private lateinit var binding: FragmentTextBinding

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.fragment_text,
                                container, false)
        binding = FragmentTextBinding.inflate(inflater,
                                container, false)
        return binding.root
    }
    fun changeText(fontsize : Int, text : String) {
        binding.apply {
          textView.textSize = fontsize.toFloat()
          textView.text = text
}   }   }
```

Projekt: FragmentSlider.zip



Flow

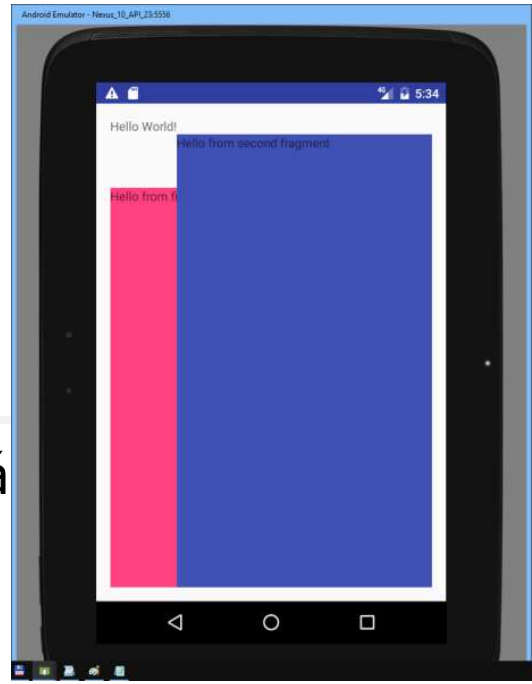
```
class MainActivity : FragmentActivity(), SlajderFragment.Listener {

    override fun onButtonClick(fontSize: Int, text: String) {
        val textViewFragment =
            supportFragmentManager.findFragmentById(
                    R.id.fragmentTextView) as TextViewFragment
        textViewFragment.changeText(fontSize, text)
}}
```

```
class TextViewFragment : Fragment() {
  fun changeText(fontsize : Int, text
      binding.textView.textSize = fontsize.toFloat()
      binding.textView.text = text
} }
```

```
class SlajderFragment : Fragment() {
  interface Listener {
    fun onButtonClick(postion: Int, text : String)
  }
  lateinit var activityCallBack : Listener
  binding.button.setOnClickListener{ v ->
    activityCallBack.onButtonClick(slajder, editText.text.toString())}
} }
```

Projekt: FragmentSlider.zip

Dynamický fragment



- dynamická práca s fragmentami je častejšia ako statická
- adresovanie fragmentu používame:
  - *supportFragmentManager* ~~*(nie fragmentManager)*~~
  - findFragmentById()
  - findFragmentByTag()

```
val sfr = supportFragmentManager // nie FragmentManager android.app.*
        .findFragmentById(R.id.frameLayout2) as SecondFragment
alebo
        .findFragmentByTag("tag2") as SecondFragment
sfr.setFText(s)
```

```
<RelativeLayout   // layout activity je zjednodušený, pozri kód
   <TextView android:text="Hello World!" android:id="@+id/textView" />
   <FrameLayout android:id="@+id/frameLayout1"  </FrameLayout>
   <FrameLayout android:id="@+id/frameLayout2"
                android:tag="tag2“ </FrameLayout>
</RelativeLayout>
```

placeholdery

Projekt: FragmentDynamicky.zip

Dynamický fragment

- dynamická práca s fragmentami je častejšia ako statická

ukážeme si:

- vytvorenie inštancie podtriedy Fragment
- poslanie argumentov fragmentu cez položku ***`arguments`***

- získanie referencie na fragment layout cez ***`supportFragmentManager`***
- vytvorenie FragmentTransaction
  - .beginTransaction()
  - .add()
  - .commit()

- vo fragmente získame context aktivity
- ten obsahuje poslané argumenty v položke ***`arguments`***

Projekt: FragmentDynamicky.zip

Dynamický fragment

aktivita môže mať viac fragmentov, ktoré spravuje ***supportFragmentManager***

- pridávanie/rušenie/modifikácia fragmentu je vždy cez FragmentTransaction:

```
val firstFragment = FirstFragment() // vytvorenie inštancie Fragment
val bundle = Bundle()
   bundle.putInt("init", 10)  // posielanie argumentu/ov do fragmentu
firstFragment.arguments = bundle
val ft = supportFragmentManager.beginTransaction()
ft.apply {
  ft.add(R.id.frameLayout1,firstFragment, "tag1")  // renderovanie
  ft.add(R.id.frameLayout2,SecondFragment(),"tag2")//podľa xml layout
  ft.commit()
}
```

vo fragmente získame context activity a hodnotu poslaných argumentov

```
override fun onAttach(context: Context) {
    super.onAttach(context)
    state=arguments?.getInt("init",0)?:0 // získanie hodnôt argumentov
    mainActivity = context as Updater
}
```

Projekt: FragmentDynamicky.zip

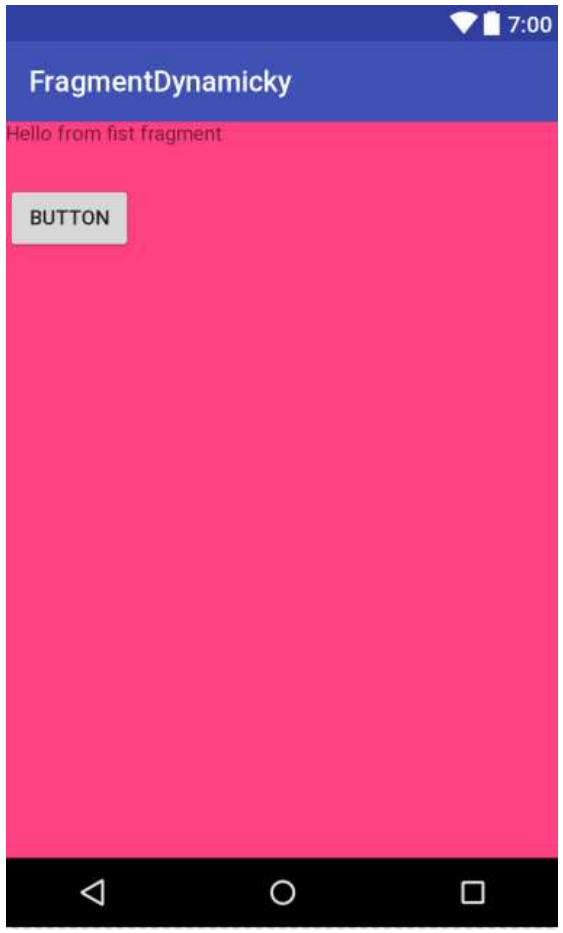
Dynamický fragment

```
val firstFragment = FirstFragment()
    val bundle = Bundle()
    bundle.putInt("init", 10)  // posielanie argumentu/ov do fragmentu
    firstFragment.arguments = bundle

supportFragmentManager
    .beginTransaction()
        .add(R.id.frameLayout1, firstFragment, "tag1") //pridanie
        .addToBackStack(null) // fragment nie je zničený, ale objaví
                              sa opätovne po stlačení Back tlačidla
        .remove(firstFragment)                    // odstráenie
        .replace(R.id.frameLayout1, firstFragment) // nahradenie
    .commit()
```

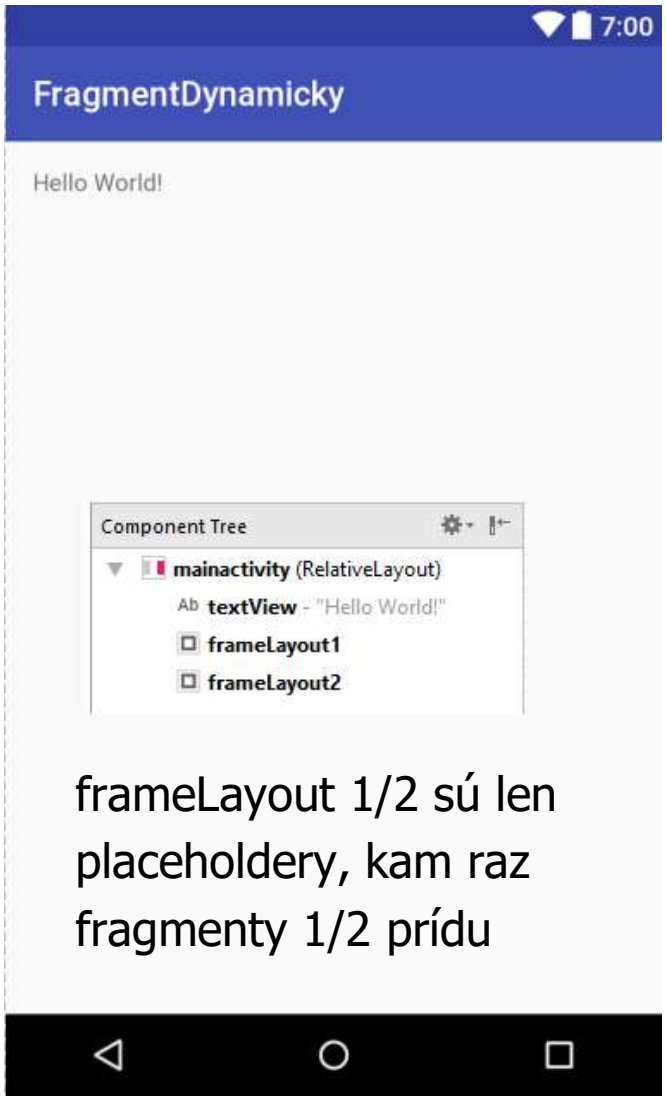
Projekt: FragmentDynamicky.zip

Dynamický fragment

`fragment_first.xml`



`activity_main.xml`



frameLayout 1/2 sú len placeholdery, kam raz fragmenty 1/2 prídu

`fragment_second.xml`

Projekt: FragmentDynamicky.zip

Komunikácia medzi fragmentami

Nikdy nie fragment<->fragment, ale nepriamo cez ich spoločnú aktivitu !

MainActivity implementuje náš Update interface

```
interface Updater {
   fun update(s:String)   // medzi aktivitami chceme posielať string
}
class MainActivity : FragmentActivity(), Updater {
  override fun update(s:String) {
     textView.text = s              // TextView v bielej aktivite
     val sfr =
        supportFragmentManager // nájdi druhý/modrý fragment
              .findFragmentById(R.id.frameLayout2) as SecondFragment
   alebo
        supportFragmentManager
              .findFragmentByTag("tag2") as SecondFragment
     sfr.setFText(s)
   }
```

Projekt: FragmentDynamicky.zip

Komunikácia medzi fragmentmi

Nikdy nie fragment<->fragment, ale nepriamo cez ich spoločnú aktivitu

FirstFragment volá náš update do main activity

```
class FirstFragment : Fragment() {
        lateinit var mainActivity: Updater
        private var state = 0

    override fun onAttach(context: Context) {
        super.onAttach(context)
        state = arguments?.getInt("init", 0)?:0
        mainActivity = context as Updater
    }

    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super.onActivityCreated(savedInstanceState)
        button.setOnClickListener {
                    mainActivity.update("state:" + state++) }
    }
```

Projekt: FragmentDynamicky.zip

Komunikácia medzi fragmentmi

- Nikdy nie fragment<->framgment, ale nepriamo cez ich spoločnú aktivitu

SecondFragment

```
class SecondFragment : Fragment() {

    fun setFText(s: String) {
        largeTextView.text = s
    }
```

Projekt: FragmentDynamicky.zip

Komunikácia medzi fragmentmi
(sumarizácia)

```
class FirstFragment {
var ma : Updater
var state …
// API < 23
onAttach(Activity a) {
  ma = a as Updater
}
// API >= 23
onAttach(Context ctx) {
  ma = ctx as Updater
}
onActivityCreated(...){
  Button = ...
  ..onClick() {
   ...ma.update(state)
  }
}
```

```
class
  MainActivity : Updater {

fun update(state){
  f=supportFragmentManager().
  findFragmentById/Tag()
 f.setFText(state)
}
```

```
interface Updater {
  fun update(state)
}
```

```
class
 SecondFragment {

 setFText(state){
   ...
 }
}
```

Ak by chceli **komunikovať obojsmerne**, tak `SecondF` tiež si musí odložiť referenciu na aktivitu a komunikovať cez ňu, referencia z fragmentu na jeho aktivitu je **`getActivity()`**

Projekt: FragmentDynamicky.zip

Komunikácia medzi fragmentmi

(nech zostane skryté, čo môže zostať skyté)

```
class FirstFragment {
  interface Updater {
    fun update(state)
  }
var ma : Updater
var state …

onAttach(Activity a) {
onAttach(Context a) {
  ma = a as Updater
}
onActivityCreated(...){
  Button =...
  ..onClick() {
   ...ma.update(state)
  }
}
```

```
class MainActivity :
 FirstFragment.Updater {

void update(state){
 f=supportFragmentManager().
  findFragmentById/Tag()
 f.setFText(state)
}
```

```
class
 SecondFragment {

 setFText(state){
   ...
 }
}
```

Interface Updater súvisí len s **FirstFragment** a **MainActivity**, takže v niektorej z nich by mal byť ukrytý

Projekt: FragmentDynamicky.zip

Aktivita fragmentu

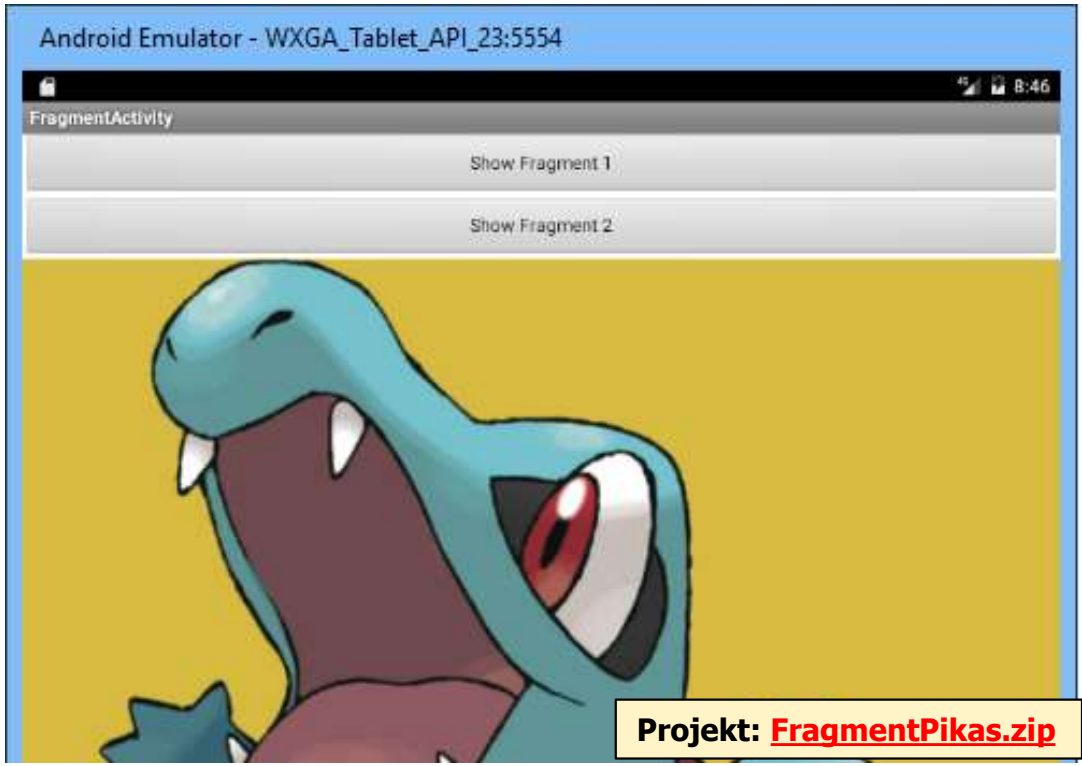

```
<LinearLayout
    android:orientation="vertical" >

    <Button
        android:id="@+id/fragment1"
        android:text="Show Fragment 1" />

    <Button
        android:id="@+id/fragment2"
        android:text="Show Fragment 2" />

    <FrameLayout  // sem dynamicky vložíme jeden z fragmentov
        android:id="@+id/fragment_place"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

Projekt: FragmentPikas.zip

Fragmenty

```
<LinearLayout  ...FragmentButtons
    android:orientation="horizontal"
    <Button
      android:text="Previous"
      android:id="@+id/prevBtn"/>
    <Button
      android:text="Next"
      android:id="@+id/nextBtn"
    />
    <Button
      android:text="Quit"
      android:id="@+id/quitBtn"
```


```
<LinearLayout ...FragmentImage
    android:orientation="vertical">
    <ImageView
        android:id="@+id/imageView"
    />
</LinearLayout>
```



Projekt: FragmentPikas.zip

addToBackStack

```
fragment1Btn.setOnClickListener {
    Log.d(TAG, "open Fragment 1")
    val fm = supportFragmentManager
    val fragmentTransaction = fm.beginTransaction()
    fragmentTransaction.replace(R.id.fragment_place, fr1)
                       // .add(R.id.fragment_place, fr1);
                       // .addToBackStack(null)
                       .commit()
}
```

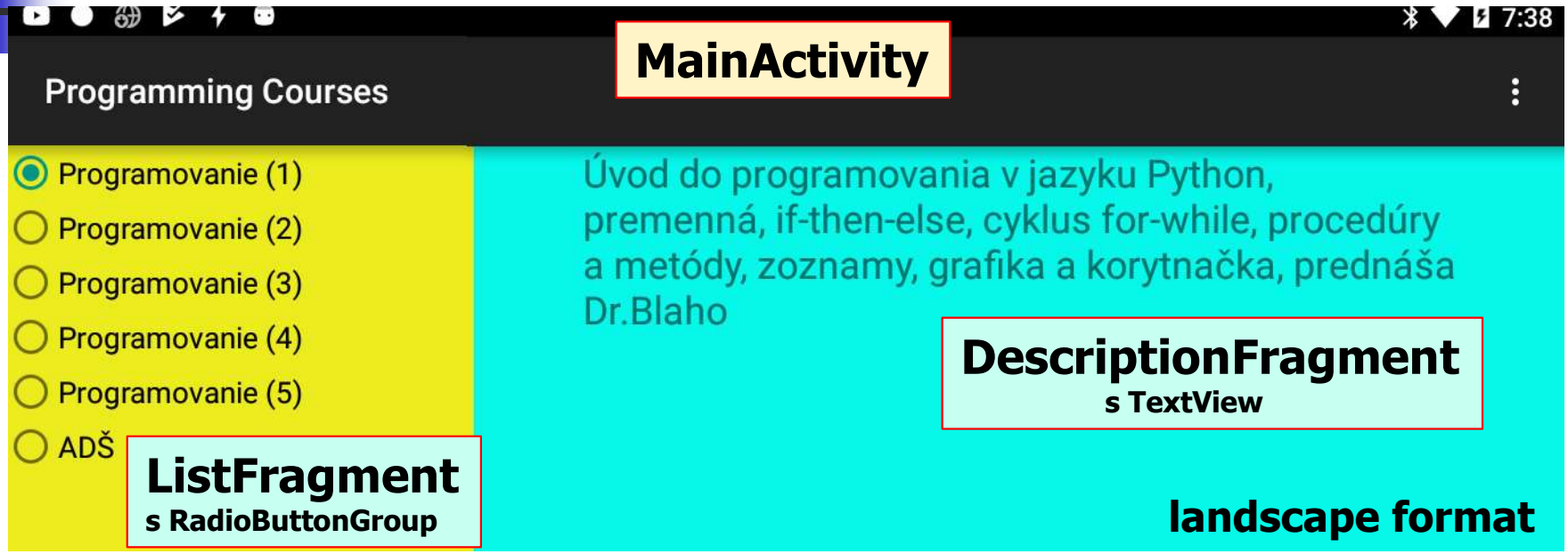
```
fragment1Btn.setOnClickListener {
    Log.d(TAG, "open Fragment 1")
    val fm = supportFragmentManager
    val fragmentTransaction = fm.beginTransaction()
    fragmentTransaction.replace(R.id.fragment_place, fr1)
                       .addToBackStack(null)
                       .commit()
}
```

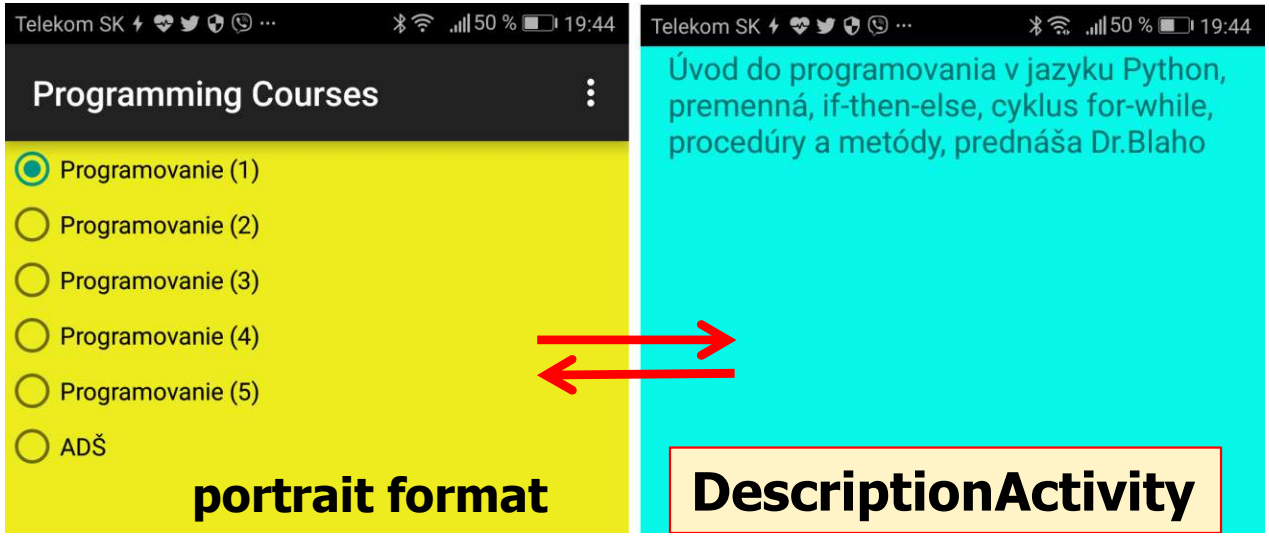
```
.add: Fragment already added: FragmentImage
```

Projekt: FragmentPikas.zip

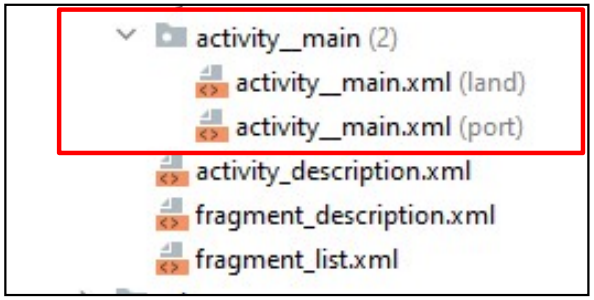
Master Detail

(MainActivity)





FragmentCourses.zip



Master Detail

(MainActivity)

- aktivita/fragment môžu mať rôzne zobrazenia/layouts,
  napr. podľa orientácie, resp. rozlíšenia displaya, tzv.qualifiers.
- Kľúčom je Android Resource Directory, ak na zdrojáku aktivity klikneme pravým, pomôže vám vygenerovať špecializované layouts aktivity podľa zobraz. parametrov

activity_main.xml (land)

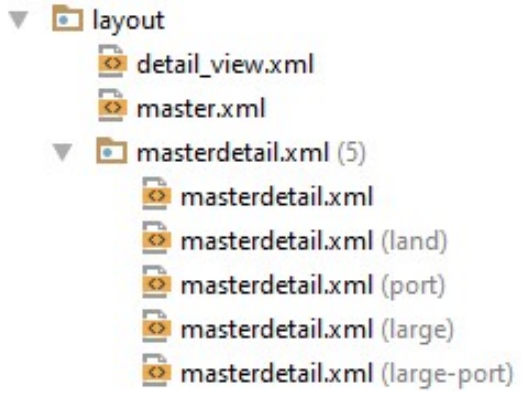
```
<LinearLayout ...
    android:orientation="horizontal“
    <fragment ...
     tools:layout="@layout/fragment_list"/>
    <fragment ...
     tools:layout="@layout/fragment_description"/>
</LinearLayout>
```

activity_main.xml (port)

```
<LinearLayout ...
    android:orientation="vertical"
    <fragment
          android:layout_width="match_parent"
          android:id="@+id/fragmentTitles"/>
</LinearLayout>
```

Projekt: FragmentCourses.zip

Adaptívny layout



Ak pre rôzne rozlíšenia a orientácie display (...qualifiers) chceme iné layouty

qualifiers

New Resource File

File name: masterdetail
Root element: LinearLayout
Source set: main
Directory name: layout-land

Available qualifiers:
Layout Direction
Smallest Screen V
Screen Width
Screen Height
Size
Ratio
UI Mode
Night Mode
Density
Touch Screen
Keyboard
Text Input
Navigation State

>>
<<

Chosen qualifiers:
Landscape

Screen orientation:
Landscape

New Resource File

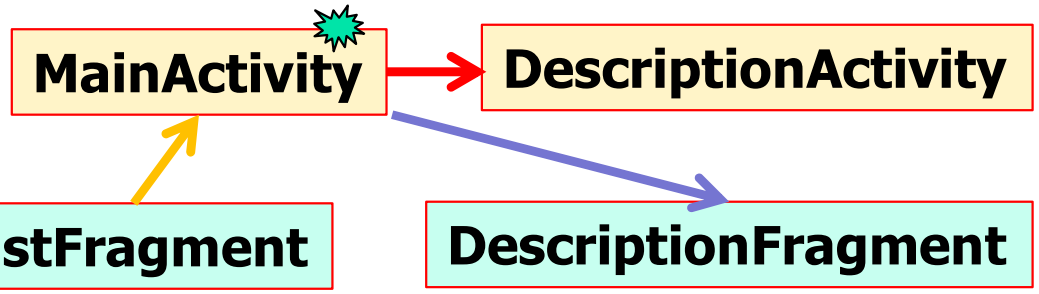
File name: masterdetail
Root element: LinearLayout
Source set: main
Directory name: layout-port

Available qualifiers:
Country Code
Network Code
Locale
Layout Direction

Chosen qualifiers:
Portrait

Screen orientation:
Portrait

Projekt: MasterDetailsAdaptive.zip

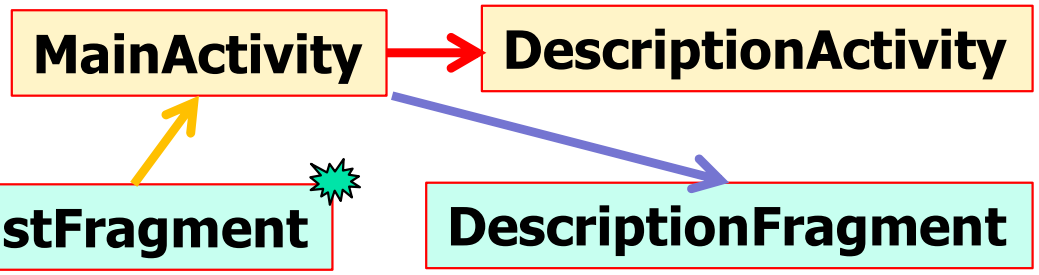


Master Detail

(MainActivity)

```
class MainActivity : AppCompatActivity(), ListFragment.Updater {

    override fun update(selectedIndex: Int) {
        val descriptionFragment = supportFragmentManager.
                findFragmentById(R.id.fragmentDescription)
                        as? DescriptionFragment
        if (descriptionFragment == null || // zatiaľ neexistuje,
            !descriptionFragment.isVisible) { // alebo ho nevidno
            if (!mCreating) {
                val intent = Intent(this,
                        DescriptionActivity::class.java)
                intent.putExtra("selectedIndex", selectedIndex)
                startActivity(intent)
            }
        } else {
            descriptionFragment.setDetail(selectedIndex)
        }
    }
```

Projekt: FragmentCourses.zip



Master Detail

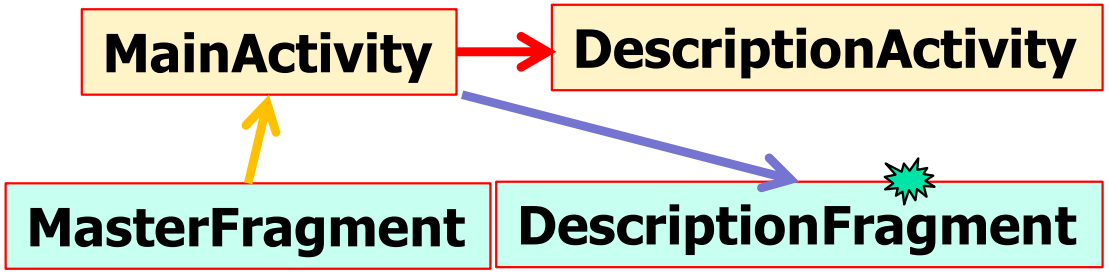
(MasterFragment)

```
class ListFragment:Fragment(),RadioGroup.OnCheckedChangeListener {

    internal interface Updater {
        fun update(selectedIndex: Int)
    }
    override fun onCheckedChanged(group:RadioGroup,checkedId:Int) {
            var selectedIndex = -1
            when (checkedId) {
                R.id.prog1ID -> selectedIndex = 0
                R.id.prog2ID -> selectedIndex = 1
                R.id.prog3ID -> selectedIndex = 2
                R.id.prog4ID -> selectedIndex = 3
                R.id.prog5ID -> selectedIndex = 4
                R.id.adsID   -> selectedIndex = 5
            }
            val listener = activity as Updater
            listener.update(selectedIndex)
        }
```

Projekt: FragmentCourses.zip

Master Detail

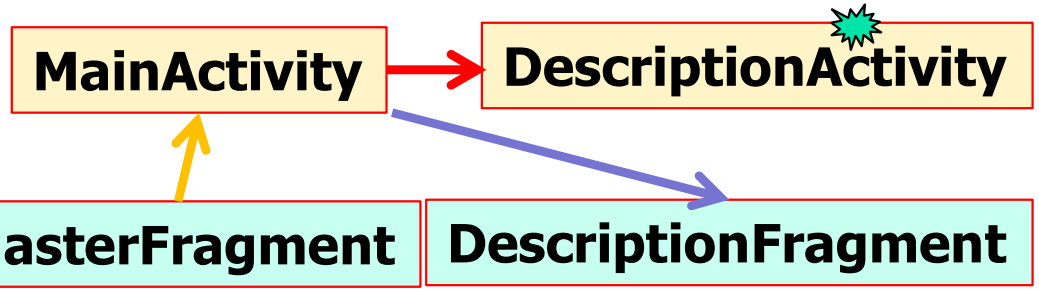
(DescriptionFragment)




```
class DescriptionFragment : Fragment() {
    private var _binding : FragmentDescriptionBinding? = null
    private val binding get() = _binding!!

    override fun onCreateView(inflater: LayoutInflater,
                              container: ViewGroup?,
                              savedInstanceState:Bundle?):View? {
        _binding = FragmentDescriptionBinding.inflate(inflater,
                                  container, false)
        return binding.root
    }

    fun setDetail(index: Int) {
        val descriptions =
        resources.getStringArray(R.array.course_full_descriptions)
        binding.descriptionID.text = descriptions[index]
    }
}
```

```
<string-array name="course_full_descriptions">
  <item>@string/prog1Detail</item>
  <item>@string/prog2Detail</item>
  <item>@string/prog3Detail</item>
  <item>@string/prog4Detail</item>
  <item>@string/prog5Detail</item>
  <item>@string/adsDetail</item>
</string-array>
```

Projekt: FragmentCourses.zip



Master Detail

(DescriptionActivity)

```
private lateinit var binding : ActivityDescriptionBinding
private var index = -1
override fun onCreate(savedInstanceState: Bundle?) {
    Log.i(TAG, "onCreate in DescriptionActivity")
    super.onCreate(savedInstanceState)
    binding = ActivityDescriptionBinding.inflate(layoutInflater)
    setContentView(binding.root)
    index = intent.getIntExtra("selectedIndex", -1)
}
override fun onResume() {
    super.onResume()
    val descriptionFragment =
        supportFragmentManager.findFragmentById
                (R.id.fragmentDescription) as DescriptionFragment
    descriptionFragment.setDetail(index)
}
```

Projekt: FragmentCourses.zip

R.layout.*yes_no_layout*

Dialog Fragment

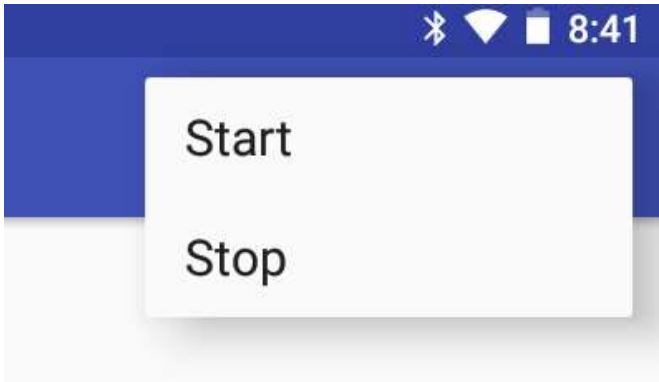
(podtrieda Fragment)

```
class YesNoDialog : DialogFragment() {
    lateinit var updater : Updater
    override fun onAttach(activity: Activity) {
        super.onAttach(activity)
        updater = activity as Updater
    }
    override fun onCreateView(inflater: LayoutInflater,
                              container: ViewGroup?,
                              savedInstanceState: Bundle?): View?
        isCancelable = false   // neda sa zrusit dialog
        val view = inflater.inflate(R.layout.yes_no_layout,
                                    container, false)
        yesBtn.setOnClickListener {
                    updater.sendMessage("yes pressed")
                    dismiss()  // zmizne dialog
                }
         return view
    }
```

Projekt: FragmentDialog.zip

Dialog Fragment

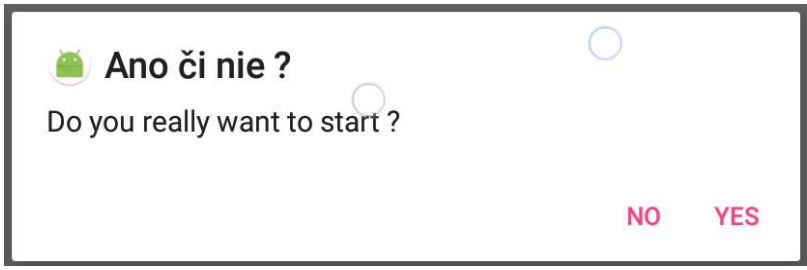
(volanie v MainActivity)




```
class MainActivity : AppCompatActivity(), YesNoDialog.Updater {

  override fun onOptionsItemSelected(item: MenuItem): Boolean {
    when (item.itemId) {
            ...
      R.id.StopID -> {
         YesNoDialog().show(supportFragmentManager, "Yes or No ?")
           return true
      }
    }
    return super.onOptionsItemSelected(item)
  }

override fun sendMessage(msg: String) {
    if (msg == "yes pressed")
        this@MainActivity.finish()
}
```

Ak bolo Yes na really want?

Projekt: FragmentDialog.zip



Alert Dialog

(musí to ist' aj jednoduchšie – varenie z polotovarov)

```
R.id.StartID -> {
   val builder = AlertDialog.Builder(this@MainActivity)
   builder.setTitle("Ano či nie ?")
      .setMessage("Do you really want to start ?")
      .setIcon(R.mipmap.ic_launcher_round)
      .setCancelable(false)
      .setPositiveButton(R.string.yesText)
         { dialogInterface, i -> Toast.makeText(this@MainActivity,
                "Start it", Toast.LENGTH_SHORT).show() }
      .setNegativeButton(R.string.noText)
         { dialogInterface, i -> Toast.makeText(this@MainActivity,
                "DO NOT Start it", Toast.LENGTH_SHORT).show() }
      .setNeutralButton(R.string.whoKnowsText)
         { dialogInterface, i -> Toast.makeText(this@MainActivity,
                "DO NOTHING", Toast.LENGTH_SHORT).show() }
      val alertDialog = builder.create()
      alertDialog.show()
      return true
}
```

Projekt: FragmentDialog.zip