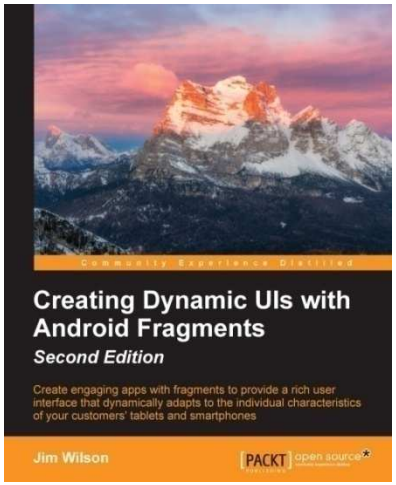
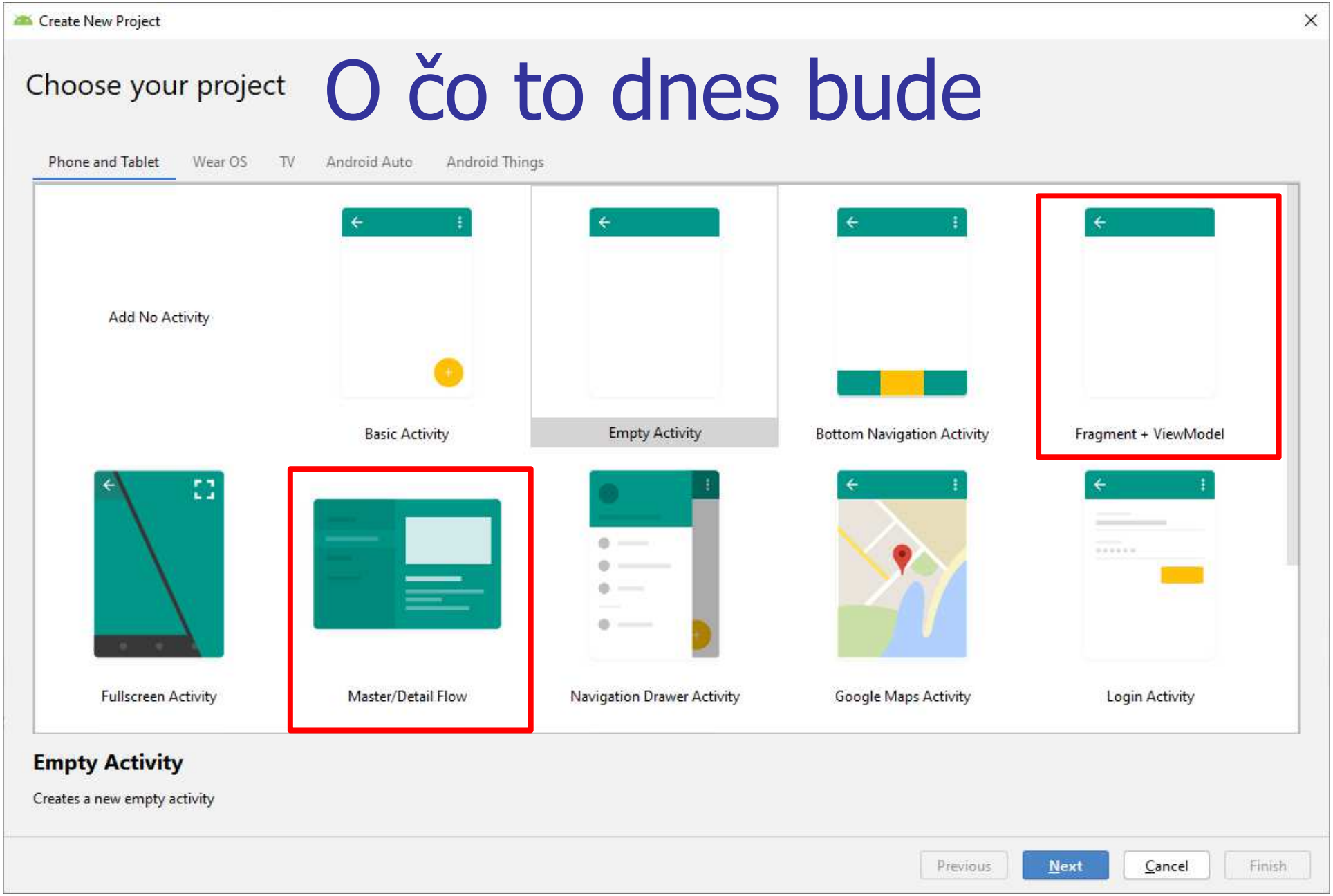
Fragment

Peter Borovanský
KAI, I-18

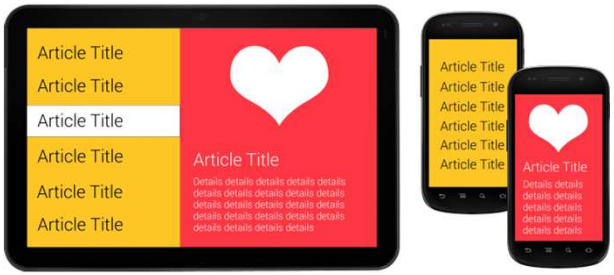
borovan 'at' ii.fmph.uniba.sk



O čo to dnes bude

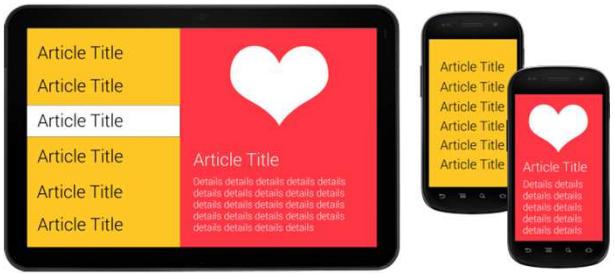


Fragmenty



- **fragment** predstavuje ucelenú časť GUI, podobne ako aktivita
- hlavným cieľom fragmentu je jeho znovu-použiteľnosť (reusability)
- každý fragment má svoju aktivitu, ktorá si ho pri inicializácii pripojí (attach)
- Koexistencia fragmentu a aktivity je zložitejšia ako život aktivity
- vzťah fragment-aktivita je typu many-many
- aktivita môže obsahovať/kombinovať viacero fragmentov, dvomi spôsobmi
  - staticky (sú navrhnuté v layout súboroch)
  - dynamicky (vzniknú dynamicky v kóde pomocou konštruktora FragmentSubClass…)

Fragmenty



- 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

- 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)
  - najnovšie Android Jetpack, balíky **`andoridx.*`** od Android 9.0 (API level 28)

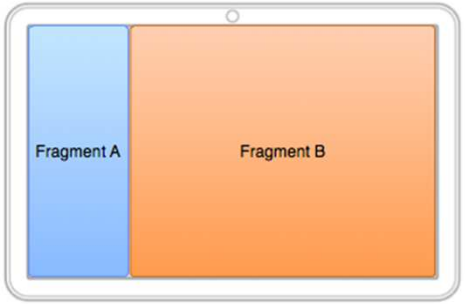
Pozor na miešanie importov z rôzných knižníc:

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

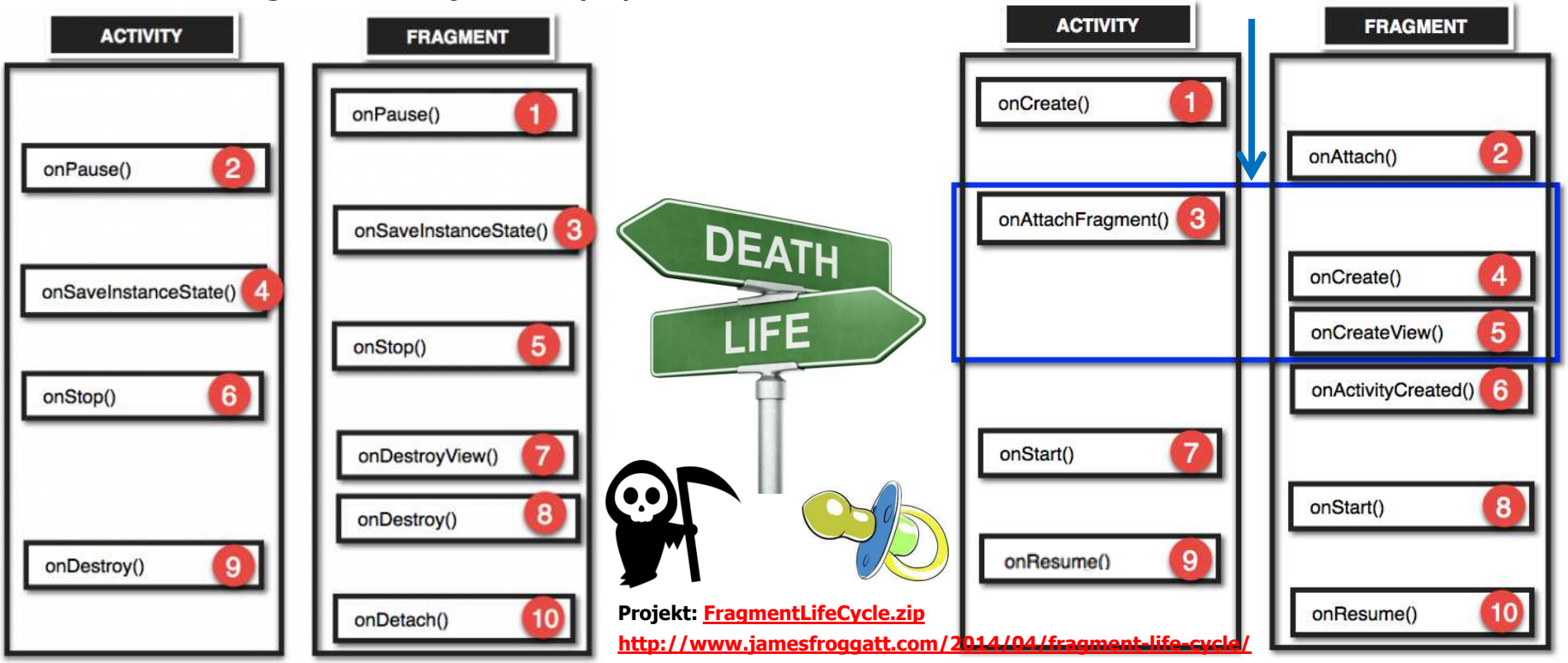
- Stavy fragmentu (životný cyklus extrémne stručne):
  - podtrieda triedy Fragment, *neexistuje nič*
  - po FragmentSubClass(), existuje inštancia fragmentu ako objekt, *nevidíme nič*
  - aktivita linkuje = attachne fragment, *nevidíme nič*, ale aspoň fragment vie, že má aktivitu
  - fragment sa zobrazí na obrazovke, a *vidíme ho*

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)

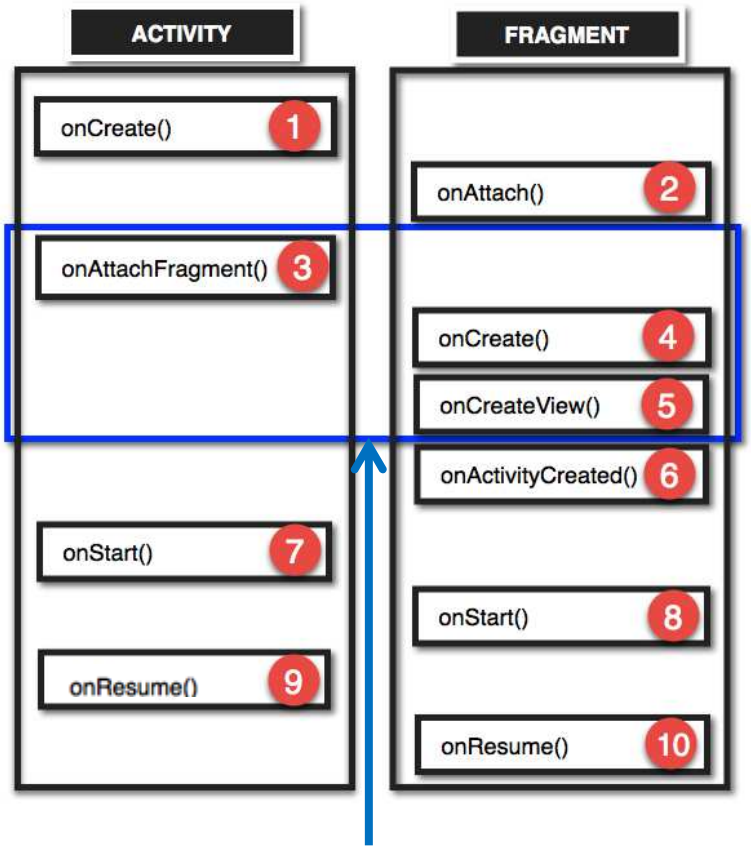


Projekt: FragmentLifeCycle.zip

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

Vznik fragmentu

(venujme sa vzniku, nie zániku)



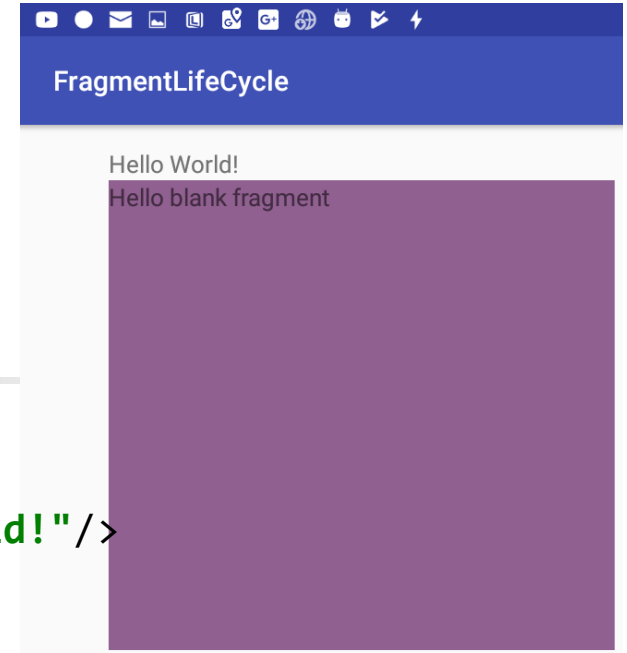
1. onCreate v activite: Najčastejšie obsahuje setContentView, ktorá definuje layout aktivity
2. onAttach vo fragmente: dostaneme pointer na aktivitu, do ktorej je vkladaný, uložíme si ho...
3. onAttachFragment v aktivite: dozvie sa, že fragment bol attach-nutý do aktivity
4. onCreate vo fragmente: aktivity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

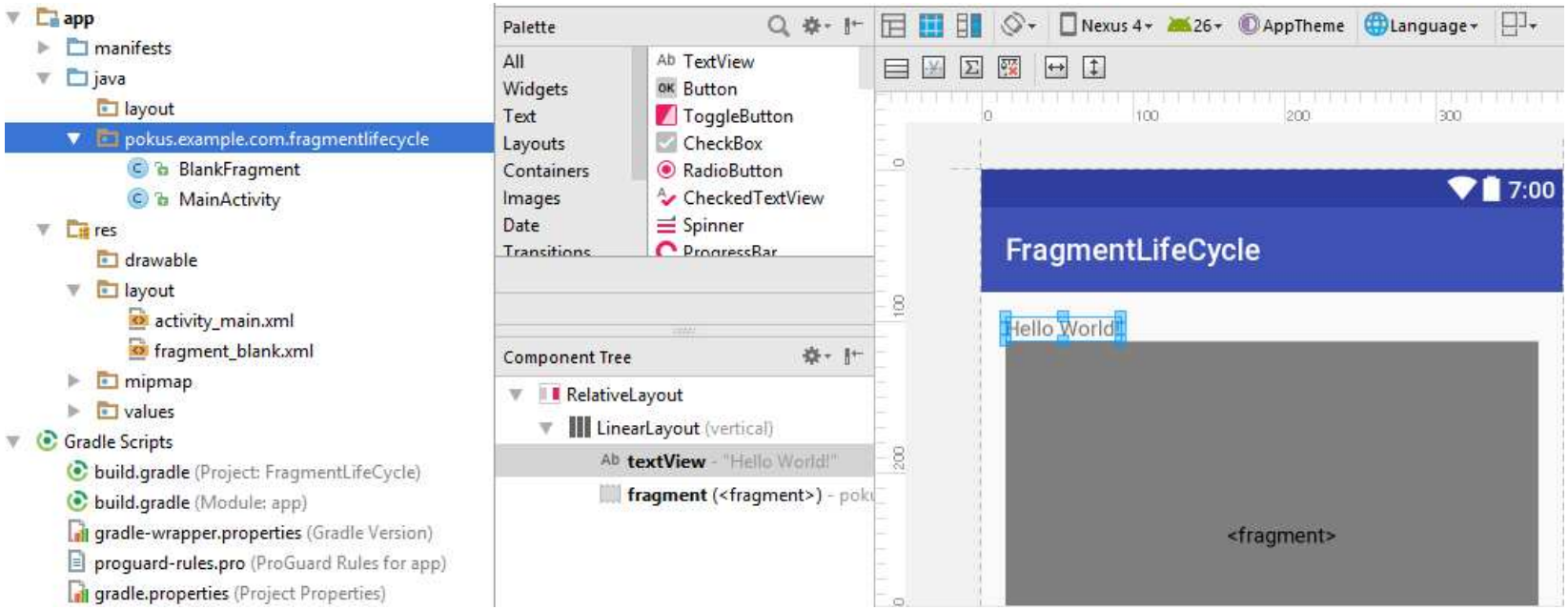
Projekt: FragmentLifeCycle.zip

Život fragmentu

(jeden fragment v aktivite)




```
<RelativeLayout
    <LinearLayout>
        <TextView ...android:text="Hello World!"/>
        <fragment ... />
    </LinearLayout>
</RelativeLayout>
```



Projekt: FragmentLifeCycle.zip

Život fragmentu
(onSaveInstance)

- napr. zmena orientácie displaya
- fragment/aktivita zaniká, môžeme si zapamäť stav:

```
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 onSaveInstance

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

Projekt: FragmentLifeCycle.zip

Život fragmentu

(detail)

Zmena orientácie

```
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
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:

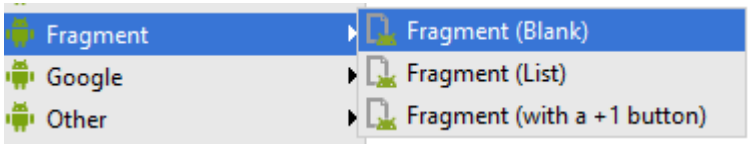
- nevolá onDestroy,
- pri opätovnom spustní sa nevolá onCreate, ale onRestart

load=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

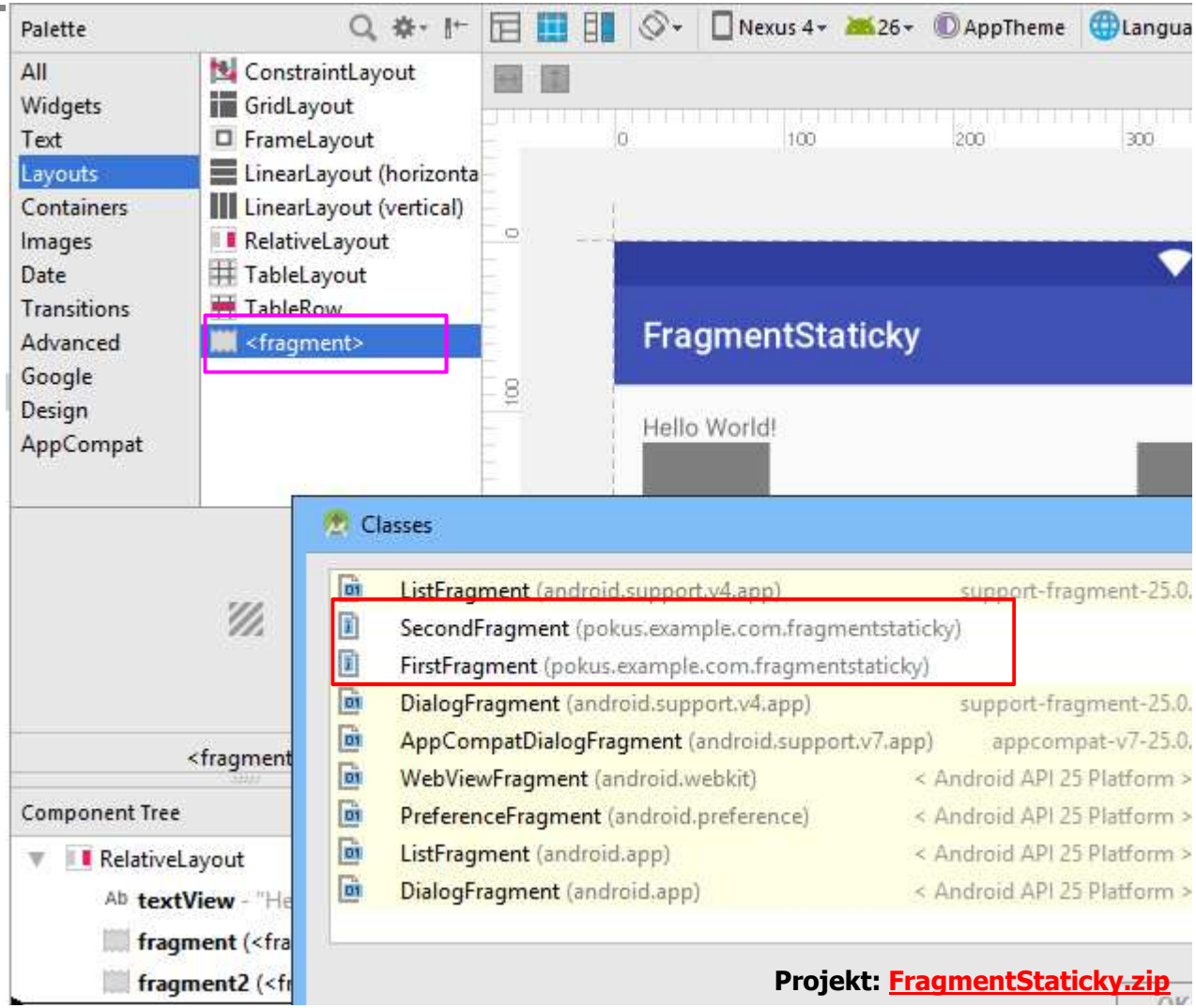
fragment_first.xml

```
<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

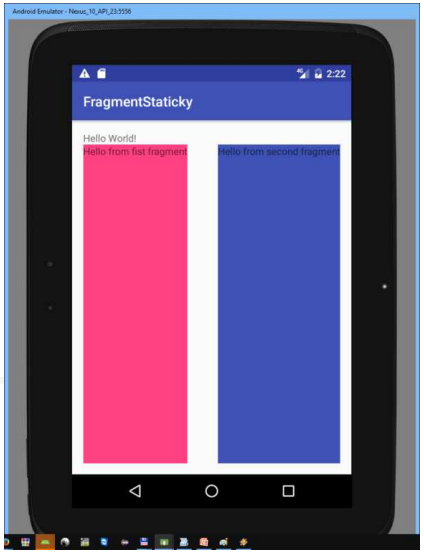
Statický fragment

Keď potom editujeme layout aktivity, tak môžeme doň vložiť <fragment> a v detailnejšej ponuke nájdeme nami vytvorené fragmenty



Projekt: FragmentStaticky.zip

Statický fragment

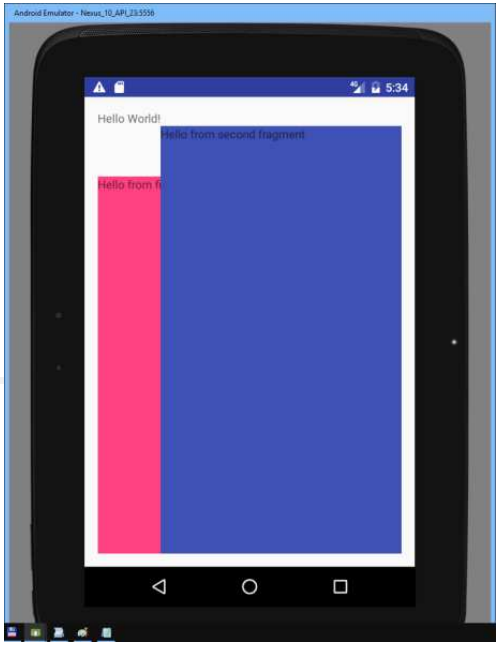



```
class FirstFragment : Fragment() {
    lateinit var mainActivity: MainActivity

    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)
    }
    override fun onAttach(context: Context) {
        super.onAttach(context)
        mainActivity = context as MainActivity
    }
}
```

Projekt: FragmentStaticky.zip

Dynamický fragment



- dynamická práca s fragmentmi je častejšia ako statická
- adresovanie fragmentu používame:
  - `findFragmentById()`
  - `findFragmentByTag()`

```
val sfr = supportFragmentManager // nie FragmentManager android.app.*
                .findFragmentById(R.id.frameLayout2)
                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" </FrameLayout>
</RelativeLayout>
```

placeholdery

Projekt: FragmentDynamicky.zip

Dynamický fragment

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

- pridávanie/rušenie/modifikácia fragmentu vždy cez FragmentTransaction:

```
val ft = supportFragmentManager.beginTransaction()
val firstFragment = FirstFragment()
   val bundle = Bundle()
   bundle.putInt("init", 10)  // posielanie argumentu/ov do fragmentu
   firstFragment.arguments = bundle
ft.add(R.id.frameLayout1, firstFragment, "tag1")
ft.add(R.id.frameLayout2, SecondFragment(),"tag2")
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 argumentu
    mainActivity = context as Updater
}
```

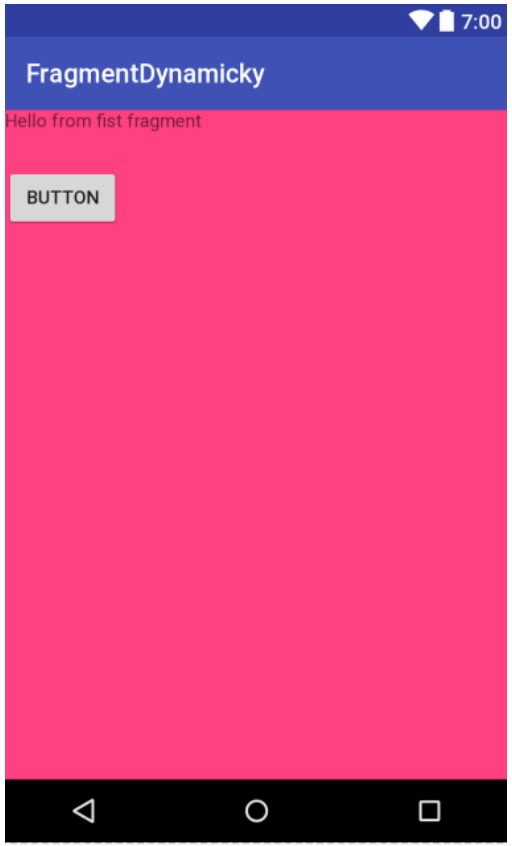
Projekt: FragmentDynamicky.zip

Dynamický fragment

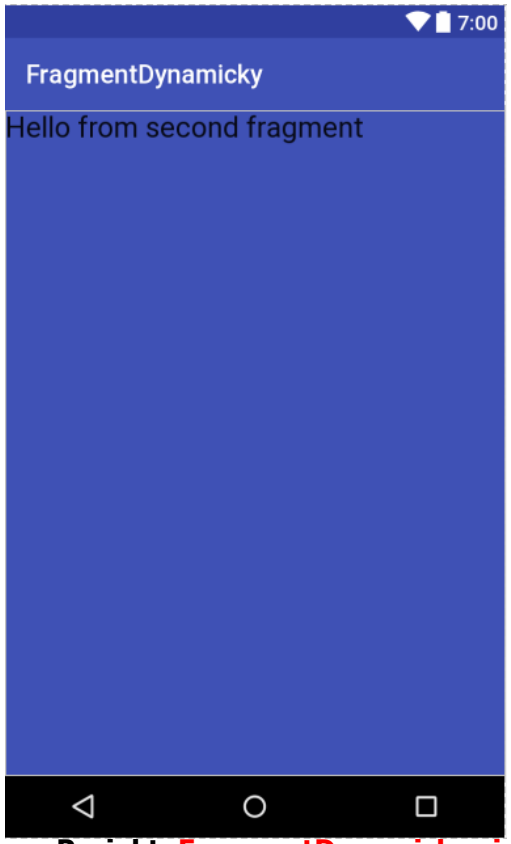
activity_main.xml

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

fragment_first.xml



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
     sfr.setFText(s)
     alebo
        supportFragmentManager
        .findFragmentByTag("tag2") as SecondFragment
   }
```

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 {
Updater ma;
XXX state;
// API < 23
onAttach(Activity a) {
  ma = (Updater)a;
}
// API >= 23
onAttach(Context ctx) {
  ma = (Updater)ctx;
}
onActivityCreated(...){
  Button =...
  ..onClick() {
   ...ma.update(state);
  }
}
```

```
class MainActivity :
   Updater {

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

```
interface Updater {
  void 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);
  }
Updater ma;
XXX state;

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

```
class MainActivity :
 FirstFragment.Updater {

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

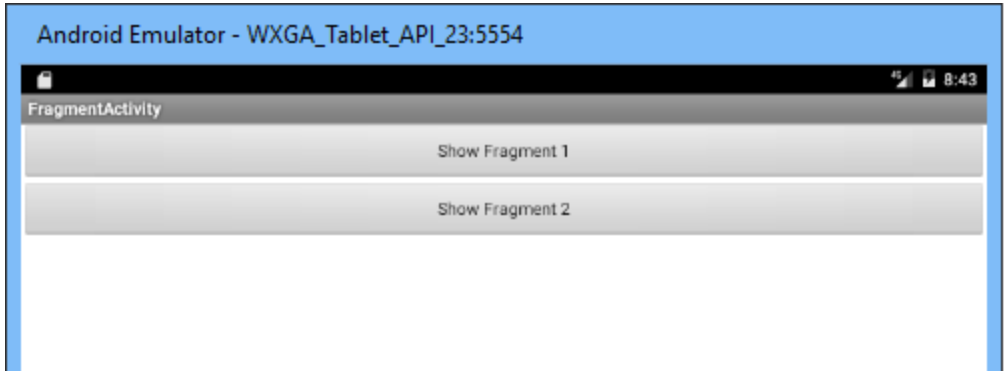
```
class
 SecondFragment {

 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>
```

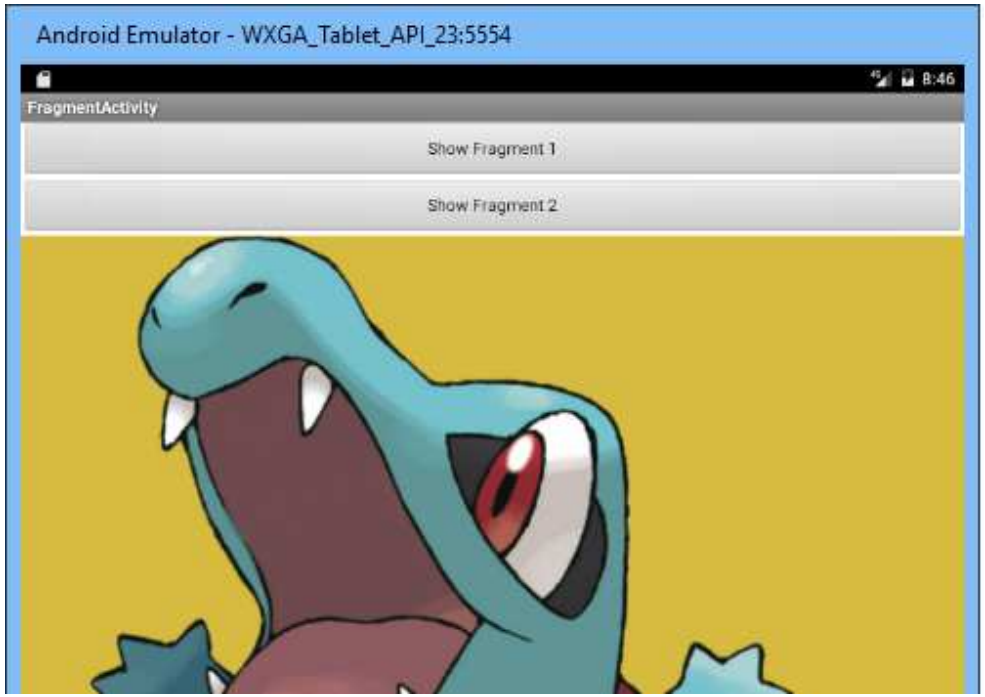
Projekt: FragmentPikas.zip

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"
    />
```

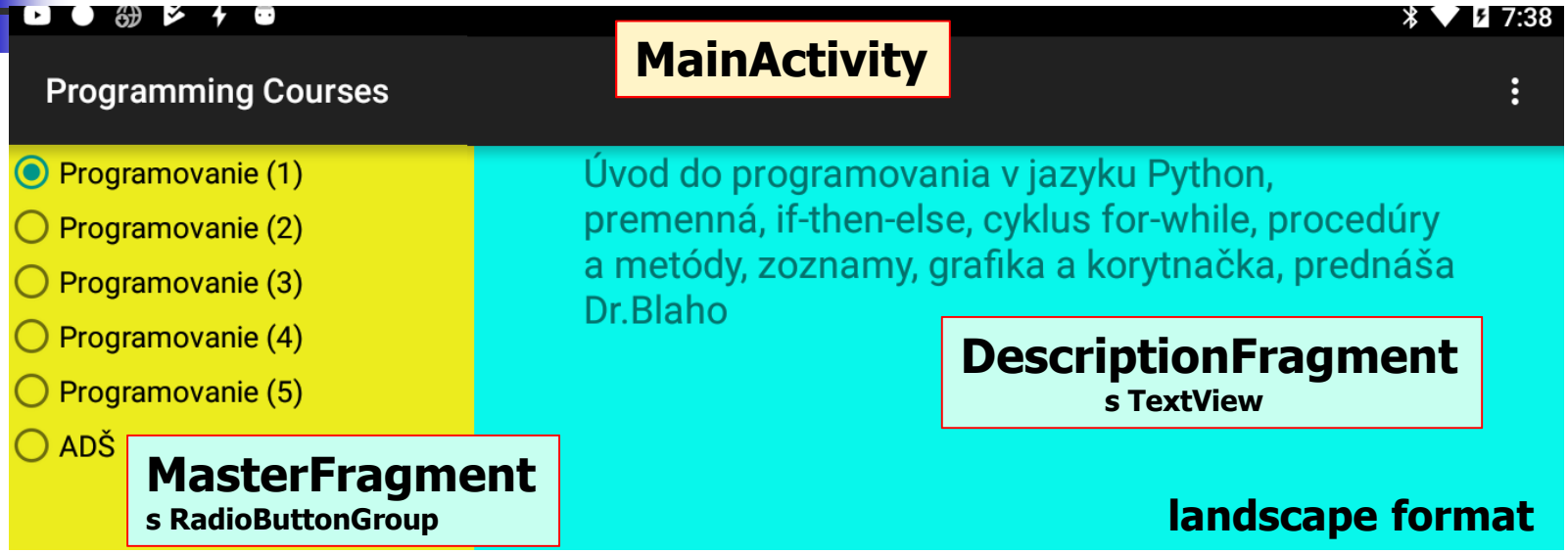
Projekt: FragmentPikas.zip

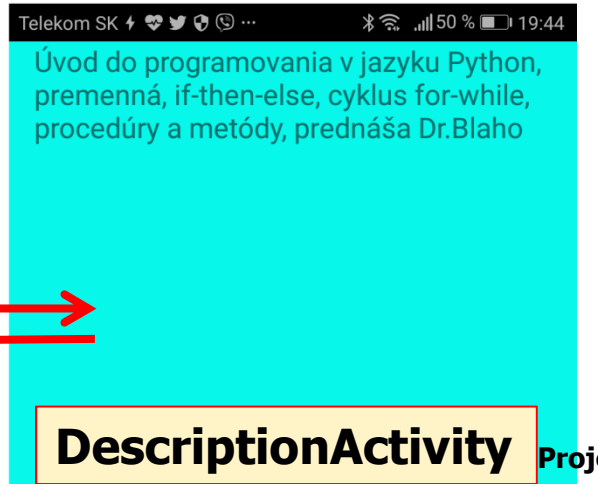
```
<LinearLayout ...FragmentImage
    android:orientation="vertical">
    <ImageView
        android:id="@+id/imageView"
    />
</LinearLayout>
```



Master Detail

(MainActivity)

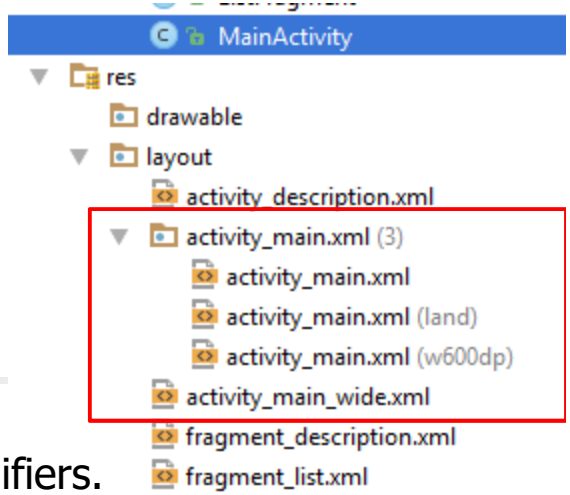




Projekt: FragementCourses.zip

Master Detail

(MainActivity)

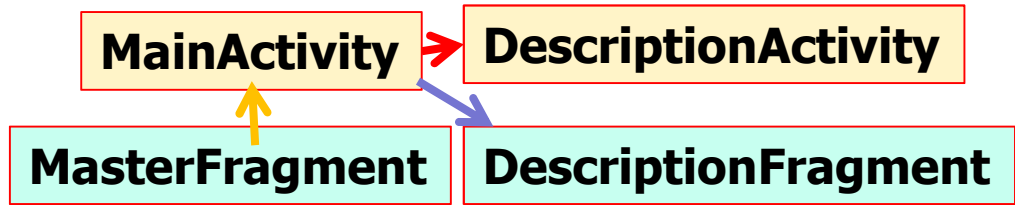


- aktivita/fragment môžu mať rôzne zobrazenia/layouts, napr. podľa orientácie, resp. rozlíšenia displaya, tzv.qualifiers.
- Kľúčom je Android Resource Directory, ak na zdrojáku aktivity klikneme pravým, pomôže vám vygenerovať špecializované layouts aktivity podľa zobraz. parametrov

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

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

Projekt: FragementCourses.zip



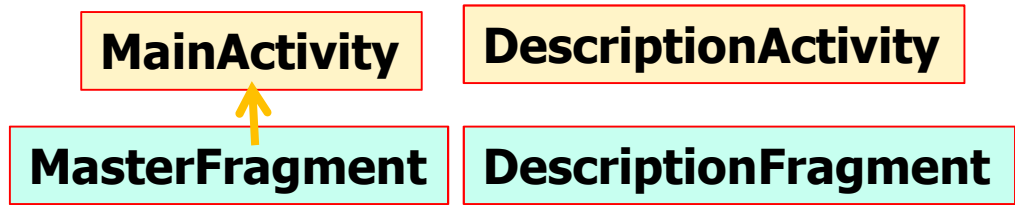
Master Detail

(MainActivity)

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

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

Projekt: FragementCourses.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: FragementCourses.zip

MainActivity | DescriptionActivity

MasterFragment | DescriptionFragment

Master Detail

(DescriptionFragment)

```
class DescriptionFragment : Fragment() {
    lateinit var tv: TextView
    override fun onCreateView(inflater: LayoutInflater,
                              container: ViewGroup?,
                              savedInstanceState:Bundle?):View? {
        val view = inflater.inflate(R.layout.fragment_description,
                              container, false)
        tv = view.findViewById(R.id.descriptionID) as TextView
        return view
    }

    fun setDetail(index: Int) {
        val descriptions = resources.getStringArray(
                        R.array.course_full_
        val course = descriptions[index]
        tv.text = course
    }
}
```

```
<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: FragementCourses.zip

MainActivity | DescriptionActivity

MasterFragment | DescriptionFragment

Master Detail

(DescriptionActivity)

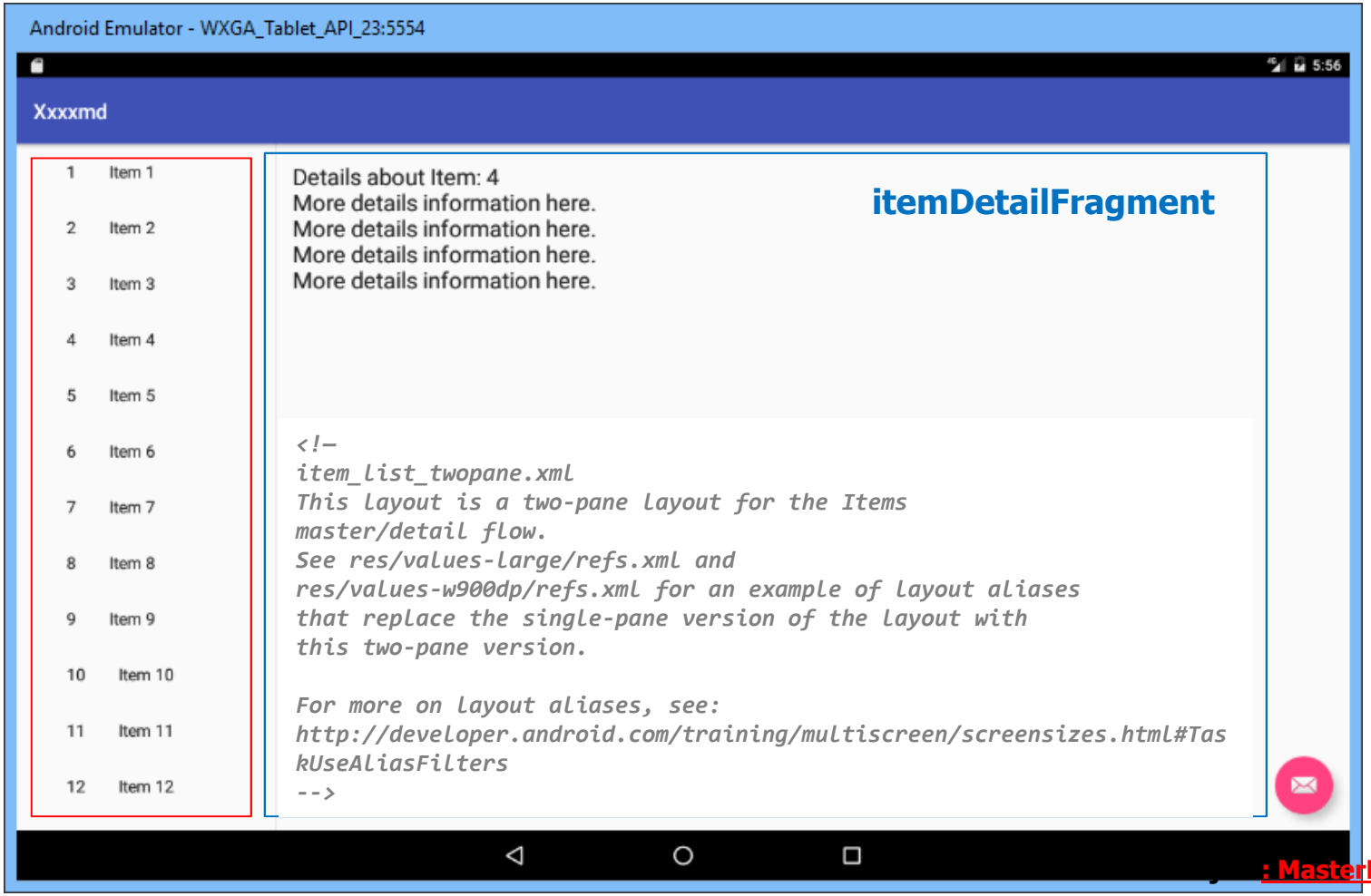
```
class DescriptionActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_description)
        val intent = intent
        val selectedIndex = intent.getIntExtra("selectedIndex",-1)
        if (selectedIndex != -1) {
            val descriptionFragment = supportFragmentManager
                        .findFragmentById(R.id.fragmentDescription)
                        as DescriptionFragment
            descriptionFragment.setDetail(selectedIndex)
        }
    }
```

Projekt: FragementCourses.zip

MasterDetail

(veľké rozlíšenie)

- nechajte AS vygenerovať M/D projekt, a pokúste sa pochopiť kód

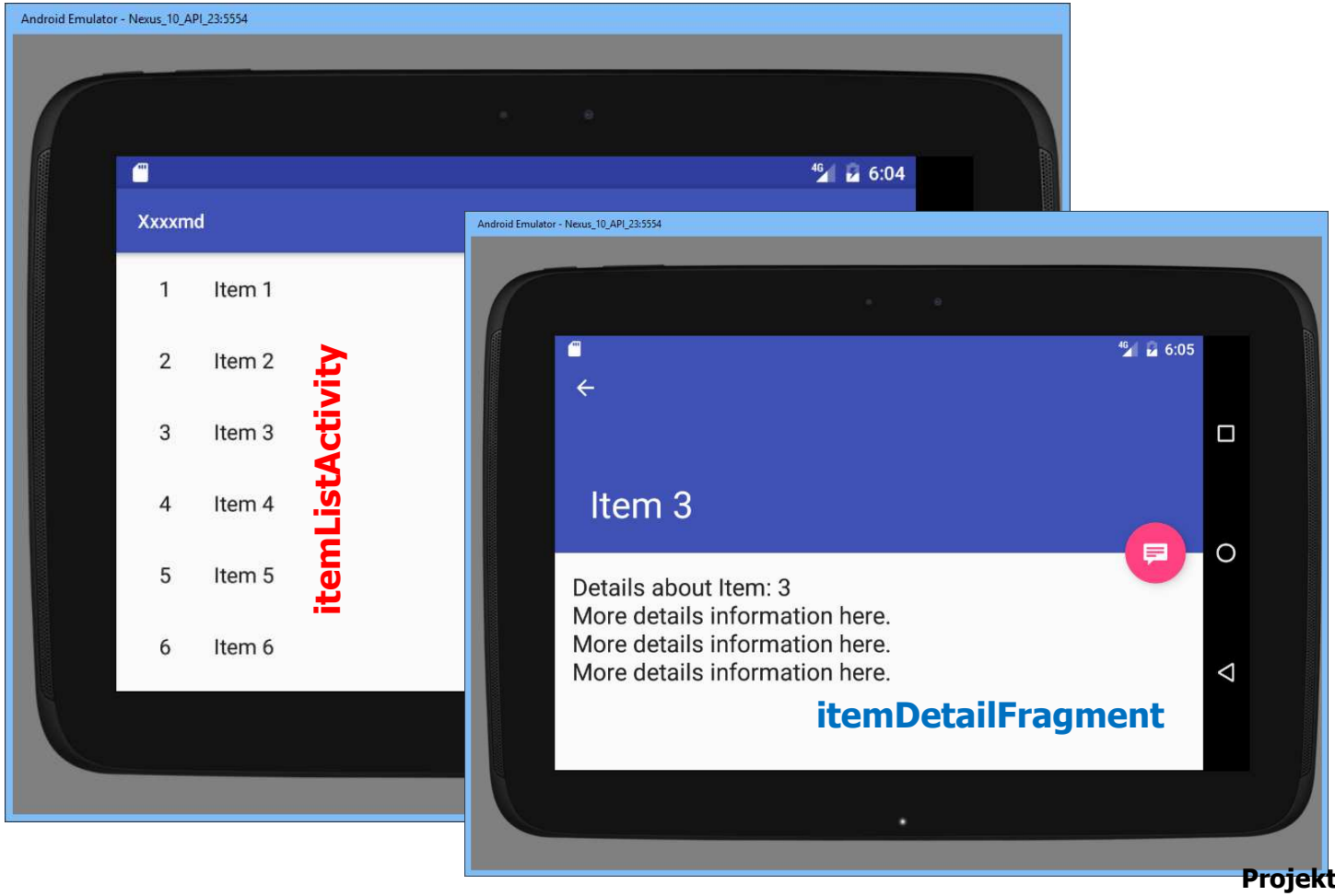


: MasterDetail.zip

MasterDetail

(malé rozlíšenie)

- pre iné rozlíšenie dostanete iný look



Projekt: MasterDetail.zip

MasterDetails

(veľké rozlíšenie)

- Projekt MasterDetails je zjednodušená verzia

: MasterDetails.zip

MasterDetails

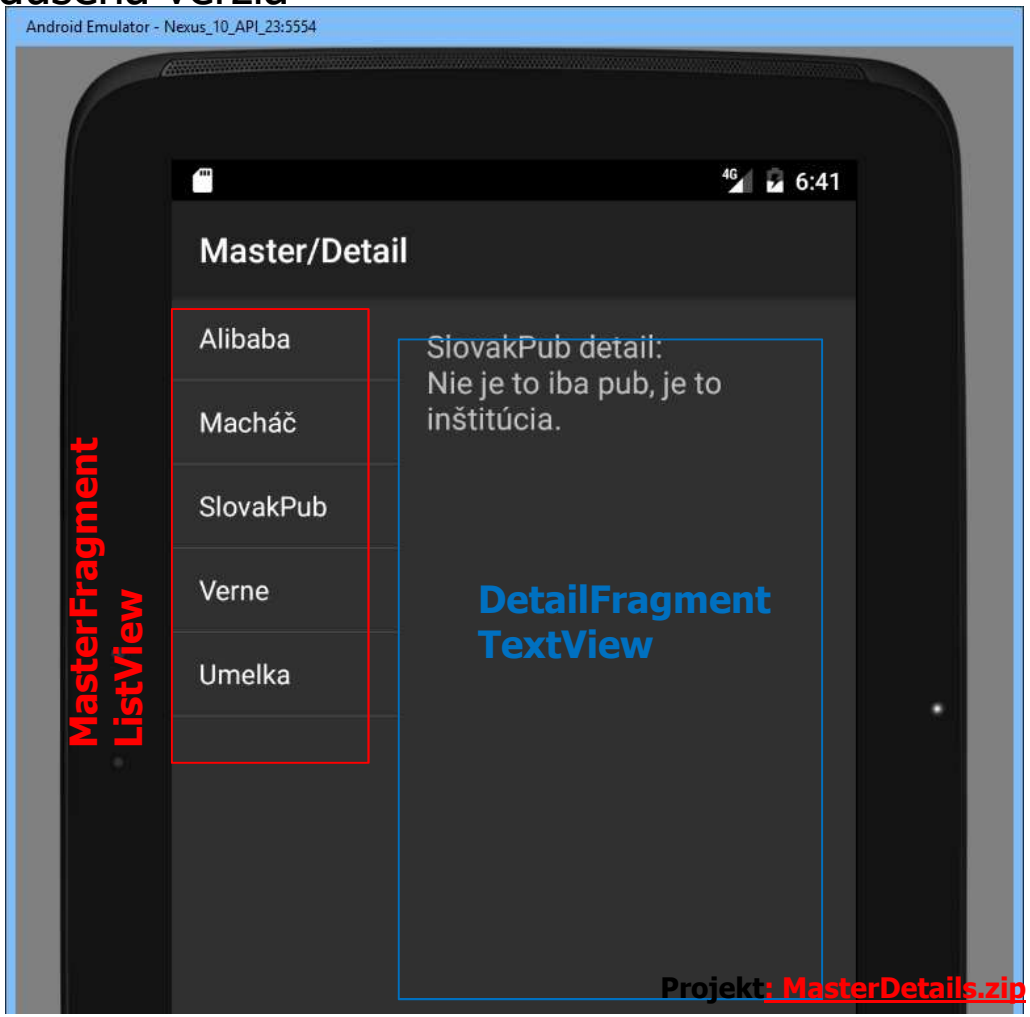
(malé rozlíšenie)

- Projekt MasterDetails je zjednodušená verzia

Problémy:

- pri zmene orientácie aktivity/ fragmentu príde k strate dát/ nastavení aktivity/fragmentu

- pri menšom rozlíšení by sme privítali iný layout fragmentov v móde landscape/portrait



Projekt: MasterDetails.zip

Perzistencia dát fragmentu

- potrebujeme uložiť index v `ListView`, na ktorom sme stáli do `Bundle savedInstance`
- pri `onCreateView` fragementu opätovne obnovíme index zo `savedInstance`

```
public class DetailFragment extends Fragment {
   @Override
   // toto sa zavolá pred restartom aktivity/fragementu
   public void onSaveInstanceState(Bundle outState) {
     super.onSaveInstanceState(outState);
     outState.putInt("INDEX", index); // uloženie hodnoty

   }
   // bundle outstate sa odpamätá až do event.volania/reštartu a/f
   @Override
   public View onCreateView(LayoutInflater inflater,
        ViewGroup container, Bundle savedInstanceState) {
     if (savedInstanceState != null) {// načítanie hodnoty
       index = savedInstanceState.getInt("INDEX");
     } ...
   // bundle je dictionary resp. HashMap<String, Object>
```

Projekt: MasterDetails.zip

Argumenty fragmentu

(fragment môže dostať argumenty od aktivity – tiež Bundle)

```
public class DetailFragment extends Fragment {
    // fragment môže dostať bungle argumentov aj od aktivity
    @Override
    public void onStart() {
      super.onStart();
      Bundle args = getArguments();
      if (args != null) {
        updateDetailView(args.getInt("INDEX"));
      } else if (index != -1) {
        updateDetailView(index);
      }
```

Bundle je HashMap<String, Object>

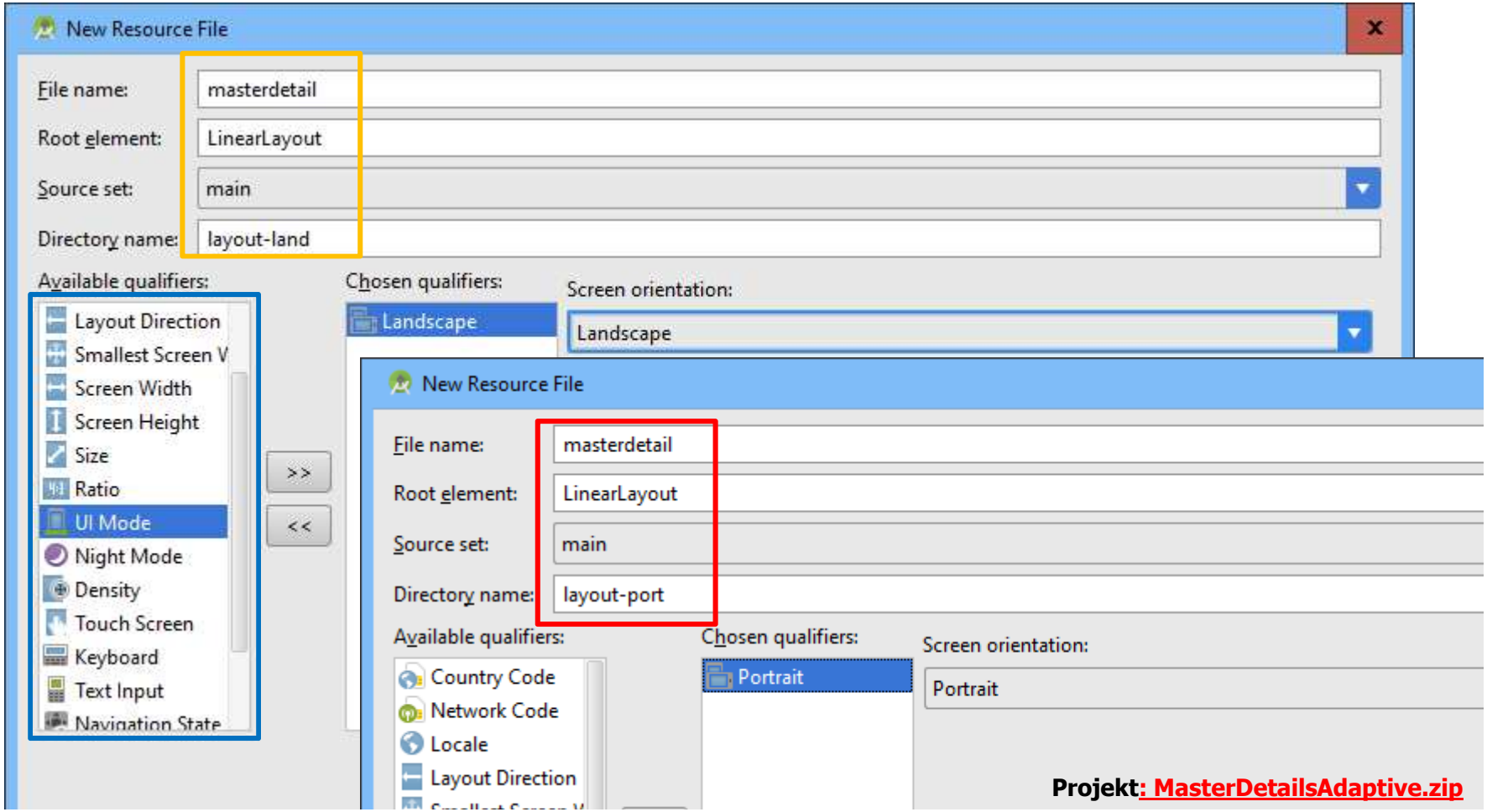
```
// Pri vytvorení fragmentu, ak aktivita chce odovzdať bungle
   argumentov vznikajúcemu fragmentu
   DetailFragment newFragment = new DetailFragment();
   Bundle args = new Bundle();
   args.putInt("INDEX", index);
   newFragment.setArguments(args);
```

Projekt: MasterDetails.zip

Adaptívny layout

- layout
  - detail_view.xml
  - master.xml
  - masterdetail.xml (5)
    - masterdetail.xml
    - masterdetail.xml (land)
    - masterdetail.xml (port)
    - masterdetail.xml (large)
    - masterdetail.xml (large-port)

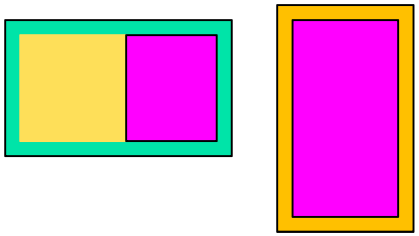
Ak pre rôzne rozlíšenia a orientácie display (...qualifiers) chceme iné layouty



qualifiers

Projekt: MasterDetailsAdaptive.zip

Flexibilný layout

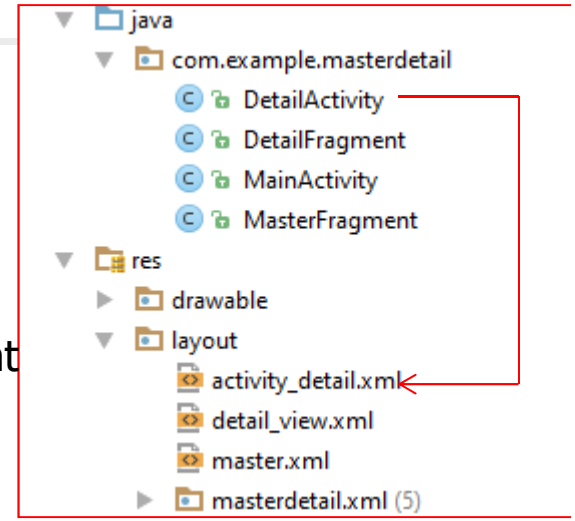


Landscape

- MainActivity
  - First/MasterFragment
  - Second/DetailFragment

Portrait

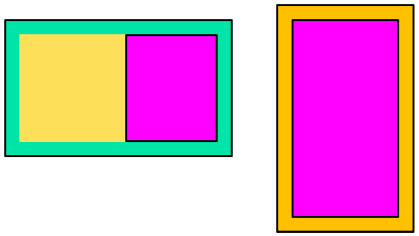
- MainActivity
  - First/MasterFragment
- DetailActivity
  - Second/DetailFragment




```
public void update(int index) {
    int orientation=getResources().getConfiguration().orientation;
    if (orientation== Configuration.ORIENTATION_LANDSCAPE) {
            ... to, čo sme robili predtým
    } else { // Configuration.ORIENTATION_PORTRAIT
        Intent in = new Intent(this, DetailActivity.class);
        in.putExtra("YNDEX",index);
        startActivity(in);
    }
}
```

Projekt: MasterDetailsFlexible.zip

Flexibilný layout

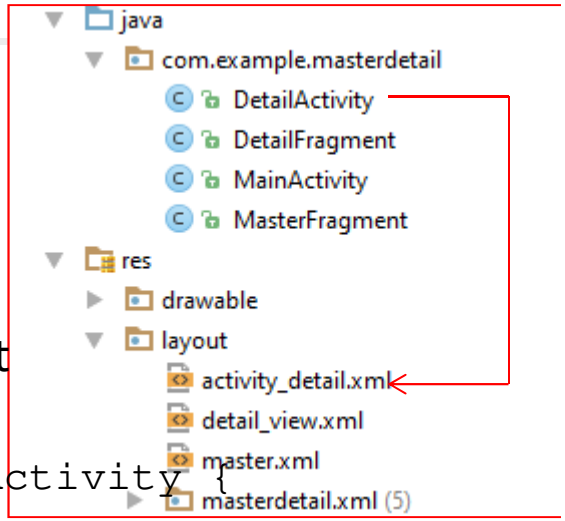


Landscape

- MainActivity
  - First/MasterFragment
  - Second/DetailFragment

Portrait

- MainActivity
  - First/MasterFragment
- DetailActivity
  - Second/DetailFragment




```
public class DetailActivity extends FragmentActivity {
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_detail);
    Intent in = getIntent();
    int yndex = in.getIntExtra("YNDEX",0);
    FragmentManager fm = getSupportFragmentManager();
    DetailFragment detailfr =
        (DetailFragment)fm.findFragmentById(R.id.detail_fragment);
    if (detailfr != null) {
      detailfr.updateDetailView(yndex);
    }
```

Projekt: MasterDetailsFlexible.zip

R.layout.*yes_no_layout*

Dialog Fragment

(podtrieda Fragment)

```
public class YesNoDialog extends DialogFragment implements OnClickListener{
    Button yes, no;
    Updater updater;
    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        updater = (Updater)activity;
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        setCancelable(false);   // dialog sa nedá zrušiť
        View view = inflater.inflate(R.layout.yes_no_layout,container,false);
        yes = (Button)view.findViewById(R.id.yesBtn);
        yes.setOnClickListener(this);
        no = (Button)view.findViewById(R.id.yesBtn);
        no.setOnClickListener(this);
        return view;
    }
```

Projekt: FragmentDialog.zip

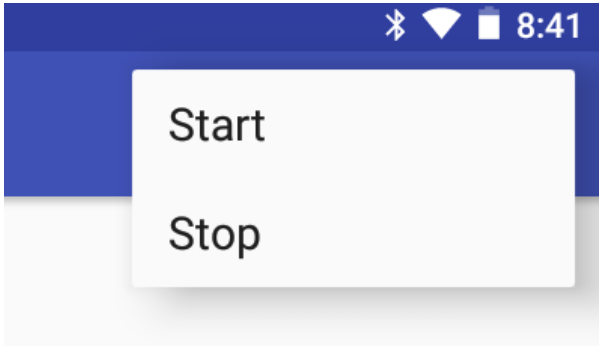
Dialog Fragment
(pokračovanie)


```
public class YesNoDialog extends DialogFragment implements OnClickListener
    ...
    @Override
    public void onClick(View view) {
            if (view.getId() == R.id.yesBtn) {
                updater.sendMessage("yes pressed");
                dismiss();  // zmizne dialog
            } if (view.getId() == R.id.NoBtn) {
                updater.sendMessage("no pressed");
                dismiss();  // zmizne dialog
            }  else {
                //...
            }
        }
        interface Updater {
            void sendMessage(String msg);
        }
    }
```

Projekt: FragmentDialog.zip

Dialog Fragment

(volanie v MainActivity)

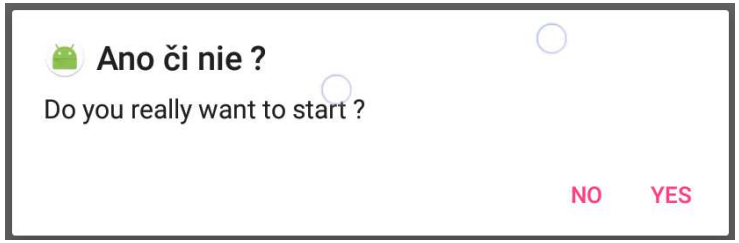



```
public class MainActivity extends AppCompatActivity
                        implements YesNoDialog.Updater {
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            ...
            case R.id.StopID:
                FragmentManager fm = getFragmentManager();
                YesNoDialog ynd = new YesNoDialog();
                ynd.show(fm, "Yes or No ?");
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    @Override
    public void sendMessage(String msg) {
        if (msg.equals("yes pressed"))
            MainActivity.this.finish();
    }
```

Ak bolo Yes na really want?

Projekt: FragmentDialog.zip



Alert Dialog

(musí to ísť aj jednoduchšie – varenie z polotovarov)

```
case R.id.StartID:
    AlertDialog.Builder builder=new AlertDialog.Builder(MainActivity.this)
    builder
        .setTitle("Ano či nie ?")
        .setMessage("Do you really want to start ?")
        .setIcon(R.mipmap.ic_launcher_round)
        .setCancelable(false)
        .setPositiveButton(R.string.yesText,
            new DialogInterface.OnClickListener() {
              @Override
              public void onClick(DialogInterface dialogInterface, int i) {
                 Toast.makeText(MainActivity.this, "Start it",
                            Toast.LENGTH_SHORT).show();
              }
            })
         .setNegativeButton(R.string.noText,  ...
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
    return true;
```

Projekt: FragmentDialog.zip

Fragment transactions

Čo nebolo...

- atomická operácia, podobne ako databázach
- beginTransaction; add(where, what, tag); commitTransaction
- beginTransaction; remove(tag/ID); endTransaction