AS Projekt
(anatómia projektu)

Peter Borovanský
KAI, I-18

MS-Teams: 2sf3ph4, List, github

borovan 'at' ii.fmph.uniba.sk

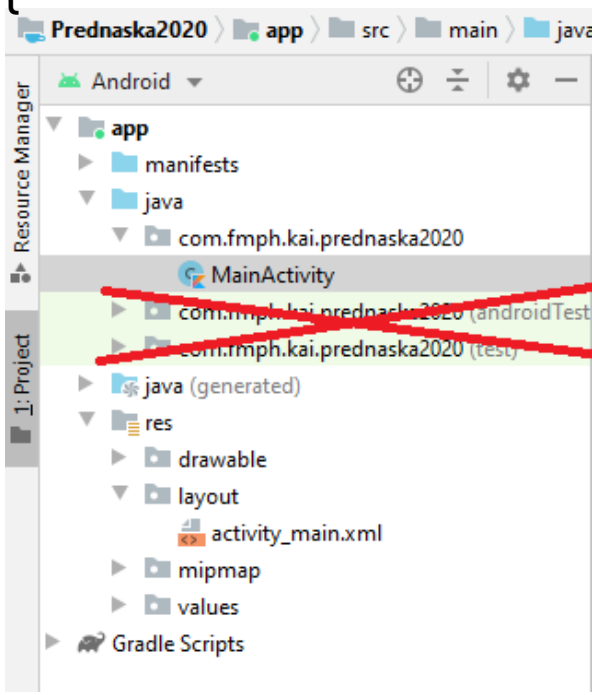
Dnes bude

- základné časti AS projektu
  - AndroidManifest, build.gradle, resources, layout, ako na obrázky či ikony, ...
- Design View
  - Design/Blueprint
- LinearLayout, TextView, Button, ...
- väzba medzi objektami z layout a kódom
  - findViewByID, plugin kotlin-android-extensions
- dobré zvyky pri návrhu layout
  - ako na warnings a errors
- Kotlin – nullables
  - operátory s tým spojené – tzv. Elvis operátor
- Cvičenie
  - vpisujete kódy do už pripravených templates
  - prémia: Piškvorky 3x3, a ďalšie

Čo dostaneme zadarmo

(pokračujeme v minulej prednáške)

```
package com.fmph.kai.prednaska2020

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {    // entry point pre App/Activity

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // sem sme minule písali náš prvý kotlin kód

    }
}
```

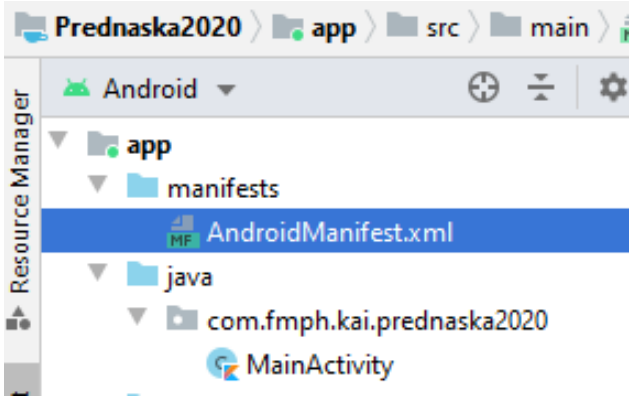


- MainActivity je inštancia triedy AppCompatActivity
- metóda onCreate() sa volá *niekde* v procese jej zobrazovania
- setContentView zobrazí layout podľa .xml popisu v R.layout.*activity_main*
- argument **`savedInstanceState:Bundle?`** zatiaľ neriešte
- package androidTest a test môžete vymazať, pre prehľadnosť

EmptyApp2021.zip

AndroidManifest.xml

(automaticky vygenerovaný súbor aplikácie)




```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.fmph.kai.prednaska2020">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

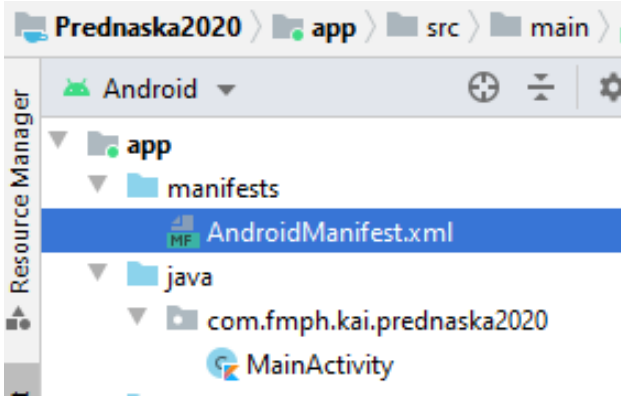
Alt-Enter

referencia na ikonu apky

referencia meno apky

EmptyApp2021.zip

AndroidManifest.xml



Hlavné tagy:

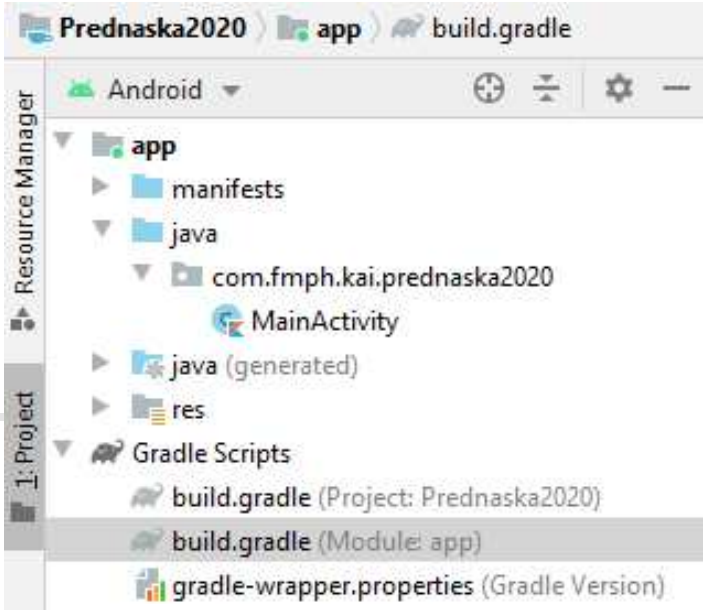
- **`<application`** je jediný a popisuje ikony, logo, meno, štýl aplikácie
- **`<activity`** može ich byť viac a popisujú package definujúci aktivitu, intent aktivity, filtre pre aktivitu, …
- **`<service`** popisujú aplikácie bežiace na pozadí, tzv. servisy
- **`<provider`** popisuje Content Provider, napr. lokálnu databázu LiteSQL
- **`<receiver`** popisuje Broadcast Receiver prijímajúci nejaké intenty

AS-manifest rokmi schudobnel, mnohé veci sa presunuli do build.gradle:

- **`<uses-configuration`** a **`<uses-feature`** popisujú HW predpoklady na spustenie apky, display, klávesnicu, senzory
- **`<uses-supportScreens`** popisuje rozliško HVGA, QVGA, QVGA, WQVGA
- **`<uses-sdk`** popisuje min./max. SDK a cieľovú verziu SDK
  http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels
- **`<uses-permissions`** popisuje práva, ktoré apka musí mať schválené
- **`<uses-library`** popisuje externé knižnice, napr. Google Maps, …
  viac na: http://developer.android.com/guide/topics/manifest/manifest-intro.html

build.gradle

(konfiguračný súbor pre gradle)

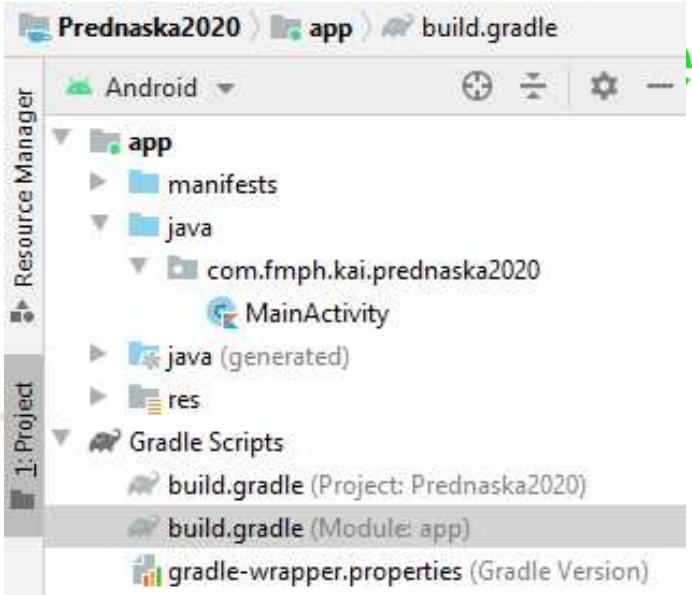


Gradle je build tool, podobne ako make, maven

```
plugins { id 'com.android.application'
          id 'kotlin-android'
          id 'kotlin-android-extensions' }
android {
    compileSdk 30
    defaultConfig {
        applicationId "com.example.emptyapp2021"
        minSdk 23
        targetSdk 30
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
...
}
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation"org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.0.2`
    ...
}
```

EmptyApp2021.zip

Gradle



- je plugin-based project-build/management system v AS založený na jazyku Groovy
- už existuje Kotlin Gradle Plugin pre Gradle 6+

```
build.gradle.kts
dependencies {
    implementation("fileTree(dir: 'libs', include: ['*.jar'])")
    implementation("org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation("androidx.appcompat:appcompat:1.0.2")
    ...
}
```

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.0.2'
    ...
}
```

EmptyApp2021.zip

MergedManifest

(spája AndroidManifest a build.gradle)

```
<manifest
  android:versionCode="1"
  android:versionName="1.0null"
  package="com.example.emptyapp2021"
  xmlns:android="http://schemas.android.com/apk/res/androic
  <uses-sdk
    android:minSdkVersion="23"
    android:targetSdkVersion="30" />
  <application
    android:allowBackup="true"
    android:appComponentFactory="androidx.core.app.CoreComp
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.EmptyApp2021" >
    <activity
      android:exported="true"
      android:name="com.example.emptyapp2021.MainActivity"
      <intent-filter
        <action
          android:name="android.intent.action.MAIN" />
        <category
          android:name="android.intent.category.LAUNCHER" />
```

Manifest Sources

core:1.6.0 manifest

EmptyApp2021.app

build.gradle manife

Other Manifest Files

(Included in merge, but c

legacy-support-core-util

manifest, customview:1.(

drawerlayout:1.0.0 manif

manifest, lifecycle-viewm

transition:1.2.0 manifest,

manifest, activity:1.2.4 m

manifest, fragment:1.3.6

lifecycle-viewmodel:2.3.1

viewpager2:1.0.0 manifes

EmptyApp2021.zip

referencia meno apky

```
<resources>
    <string name="app_name">MyFirstApp</string>
</resources>
```

Resources/Values

- drawables - obrázky v rôznych rozlíšeniach (ldpi, mdpi, hdpi, xhdpi, xxhdpi)
- layouts – rozloženia komponentov na aktivitách (bude dnes a na budúce)
- menus – pre aktivity (bude neskôr)
- values – pomenované konštanty (strings.xml, colors.xml, styles.xml …)
- raw – obrázky
  zvuky,…

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
```

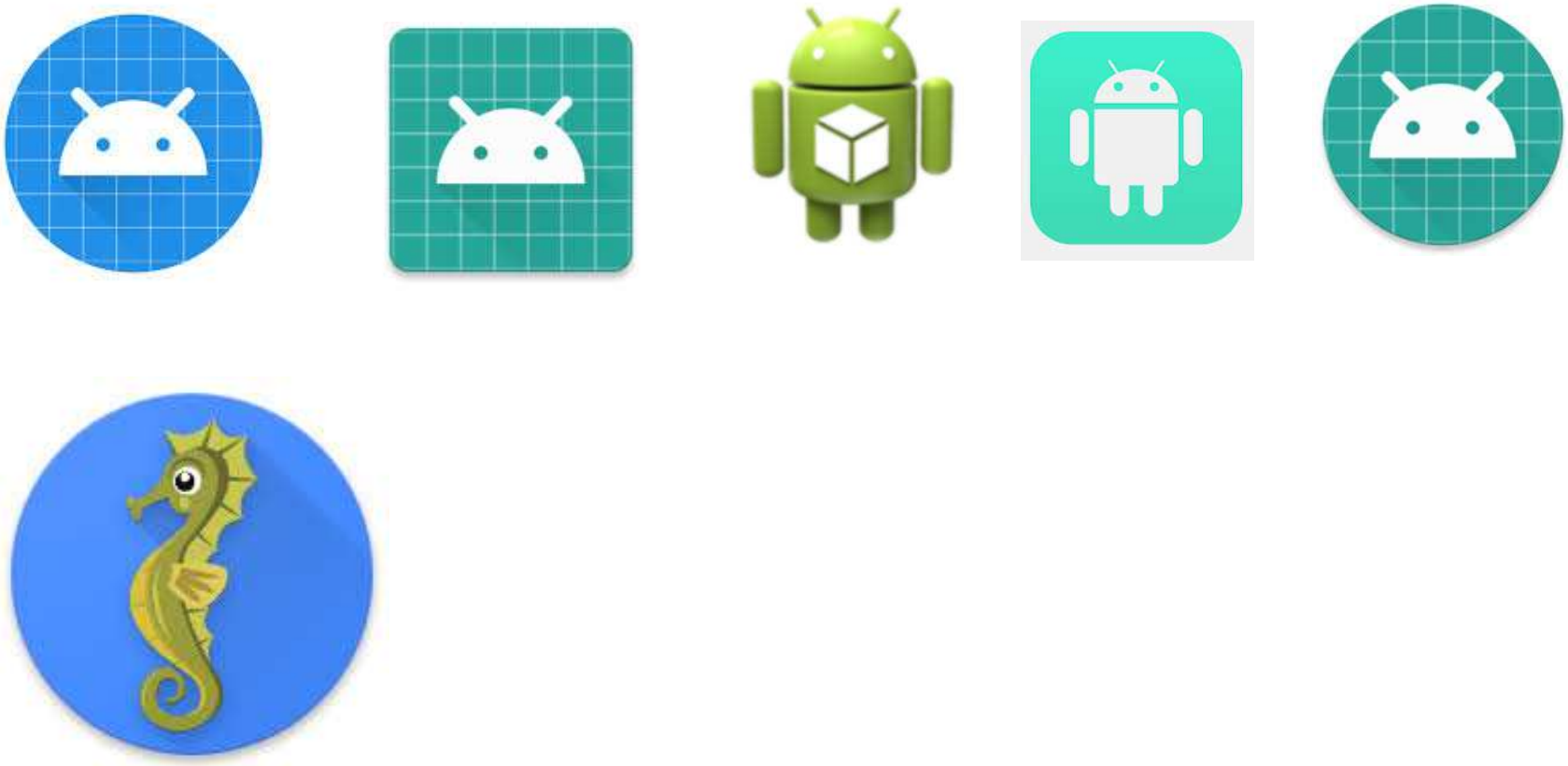
```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

EmptyApp2021.zip

Buď kreatívny

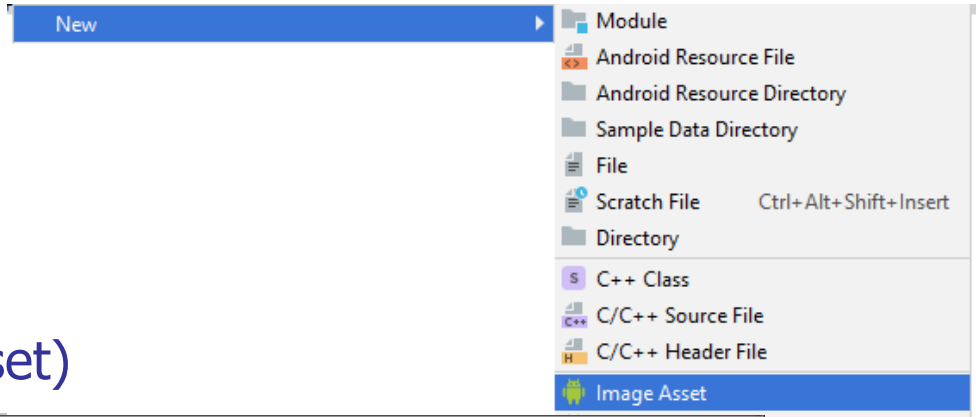
(aspoň pri ic_launcher ikone)

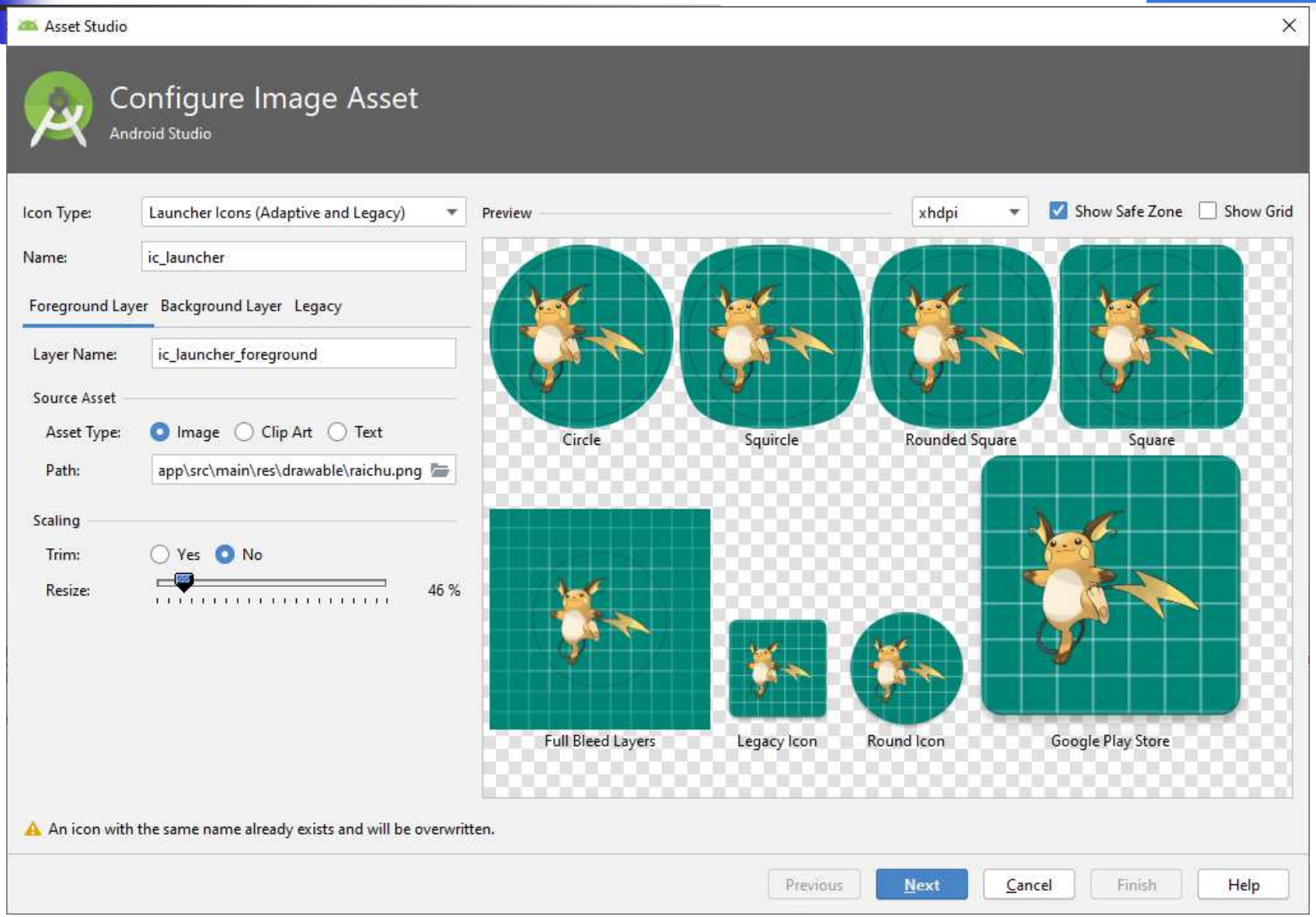
Je hrozné pri opravovaní mať v tablete/mobile viacero študentských riešení s generickými/neosobnými ikonami. Preto ak sa dá, tak sa zosobnite v posielanom riešení už v ikone vašej aplikácie.



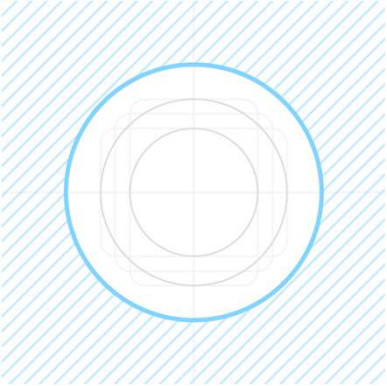
Buď kreatívny

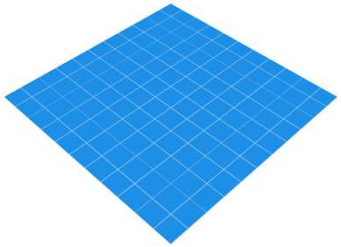
(a použi Asset Studio - New/ImageAsset)



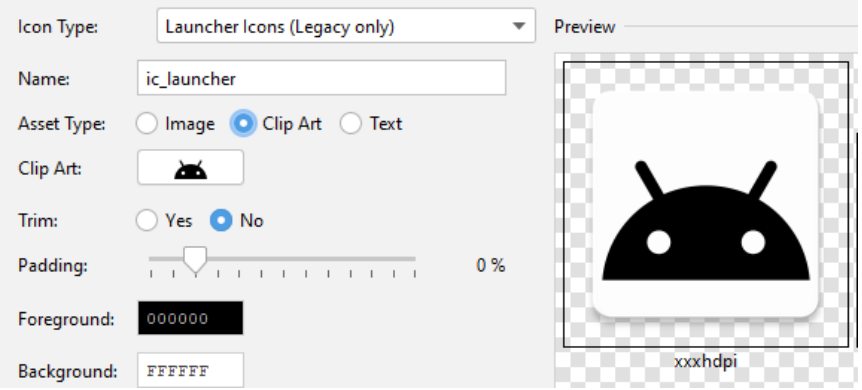


Adaptive icon





- funguje od Android-Oreo, API 26 – Android
- umožňuje zariadeniu vhodne škálovať ikonu podľa
  - zvoleného rozlíšenia 108dp, 66dp, ...
  - zvoleného orámovania
- adaptívna ikona má pozadie a popredie
-
```
<adaptive-icon
xmlns:android="http://schemas.android.com/apk/res/android">
  <background android:drawable="@drawable/ic_launcher_background" />
  <foreground android:drawable="@drawable/ic_launcher_foreground" />
</adaptive-icon>
```
- adaptívna ikona umožňuje zariadeniu robiť efekty pri zobrazovaní

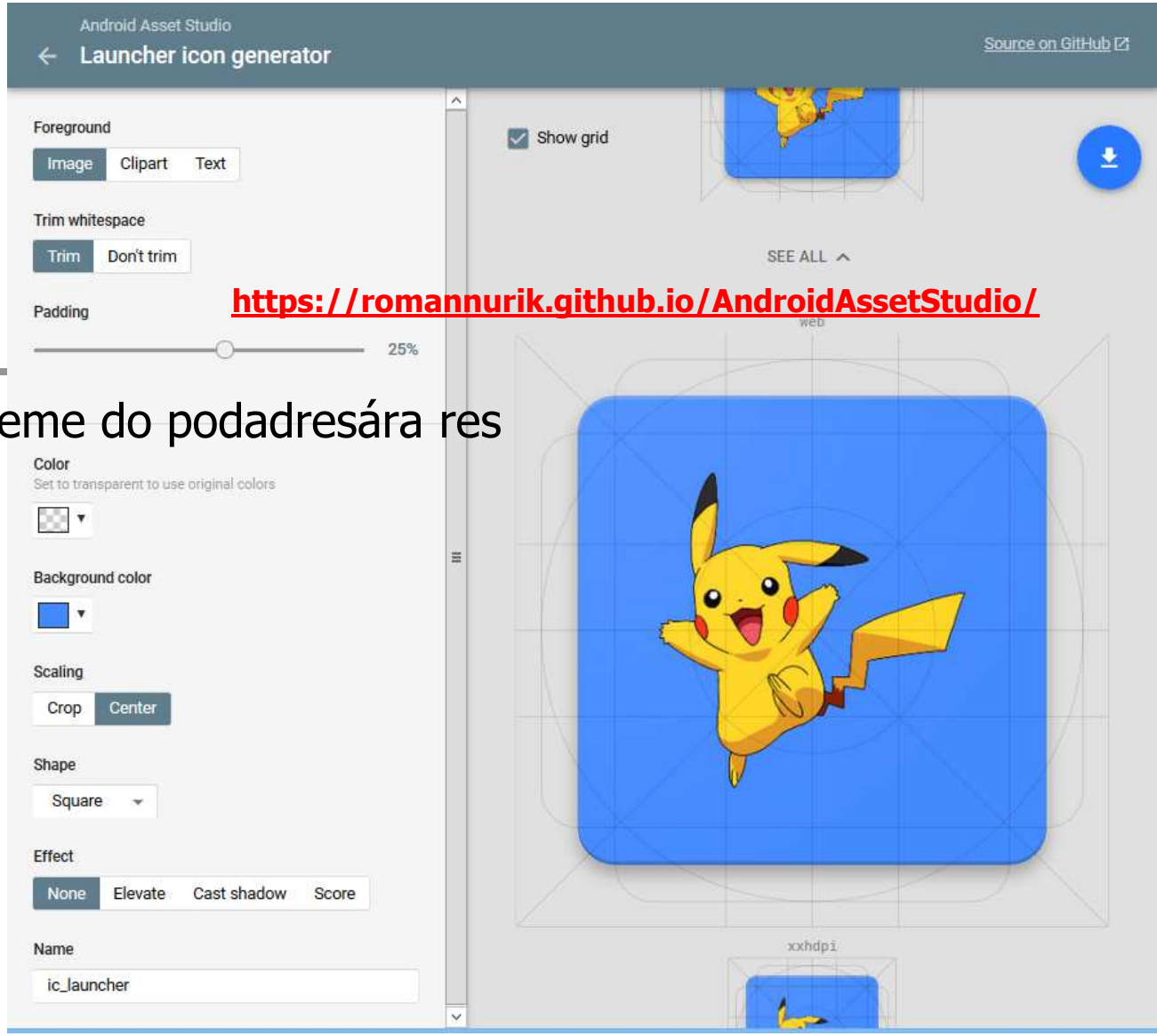


- legacy ikona je jednoduchšia

https://developer.android.com/guide/practices/ui_guidelines/icon_design_adaptive?hl=de

Android Asset Studio

Icon generator

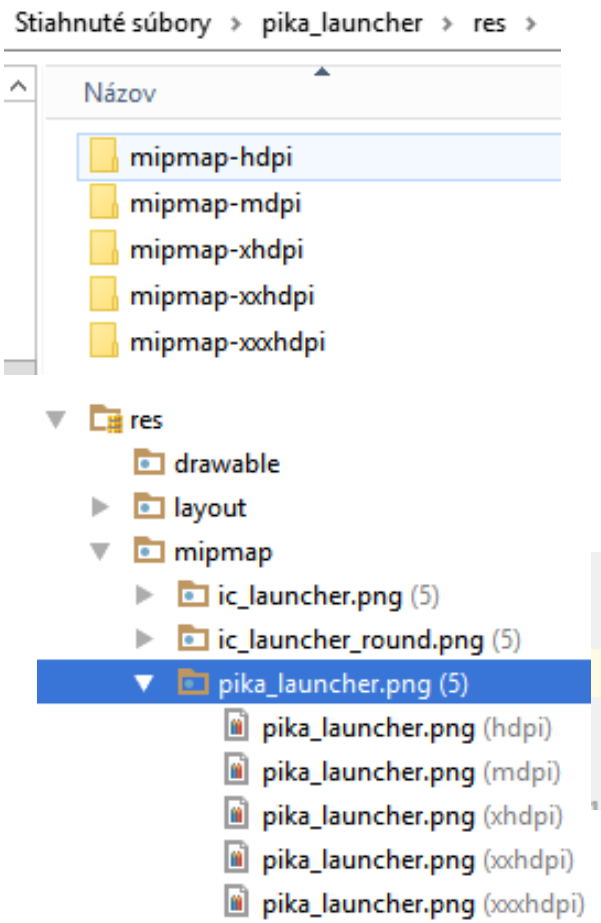


https://romannurik.github.io/AndroidAssetStudio/

výsledok priamo nakopírujeme do podadresára res

Ikony/obrázky sa

sa objavia v projekte



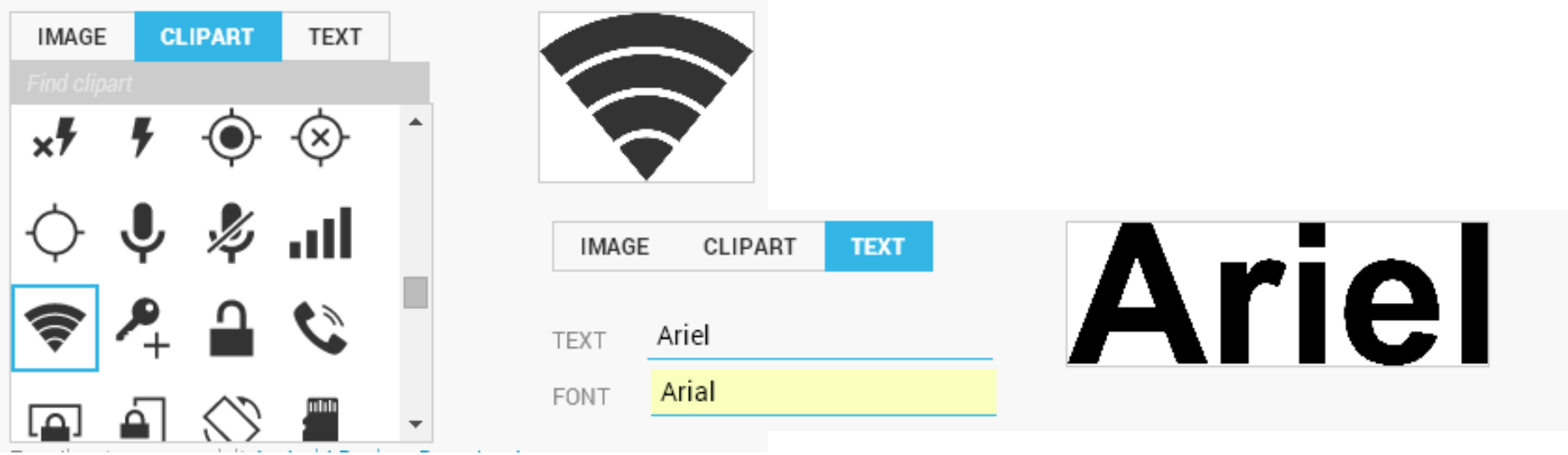

```
5   <application
6       android:allowBackup="true"
7       android:icon="@mipmap/pika_"
8       android:label @mipmap/pika_launcher
9       android:roundIcon="@mipmap/ic_launcher_round"
```

EmptyApp2021.zip

Android Asset Studio

(jedna z alternatív)

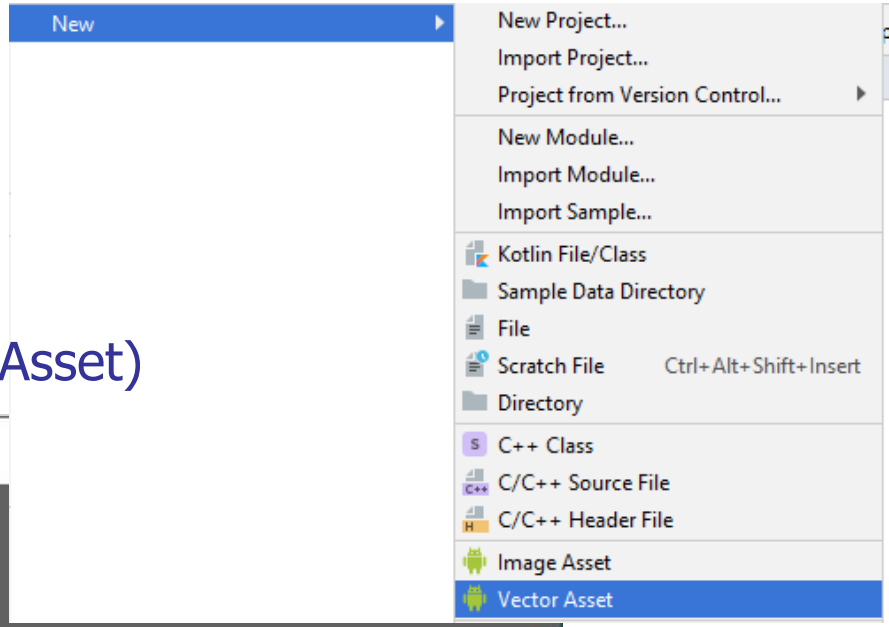
https://romannurik.github.io/AndroidAssetStudio/

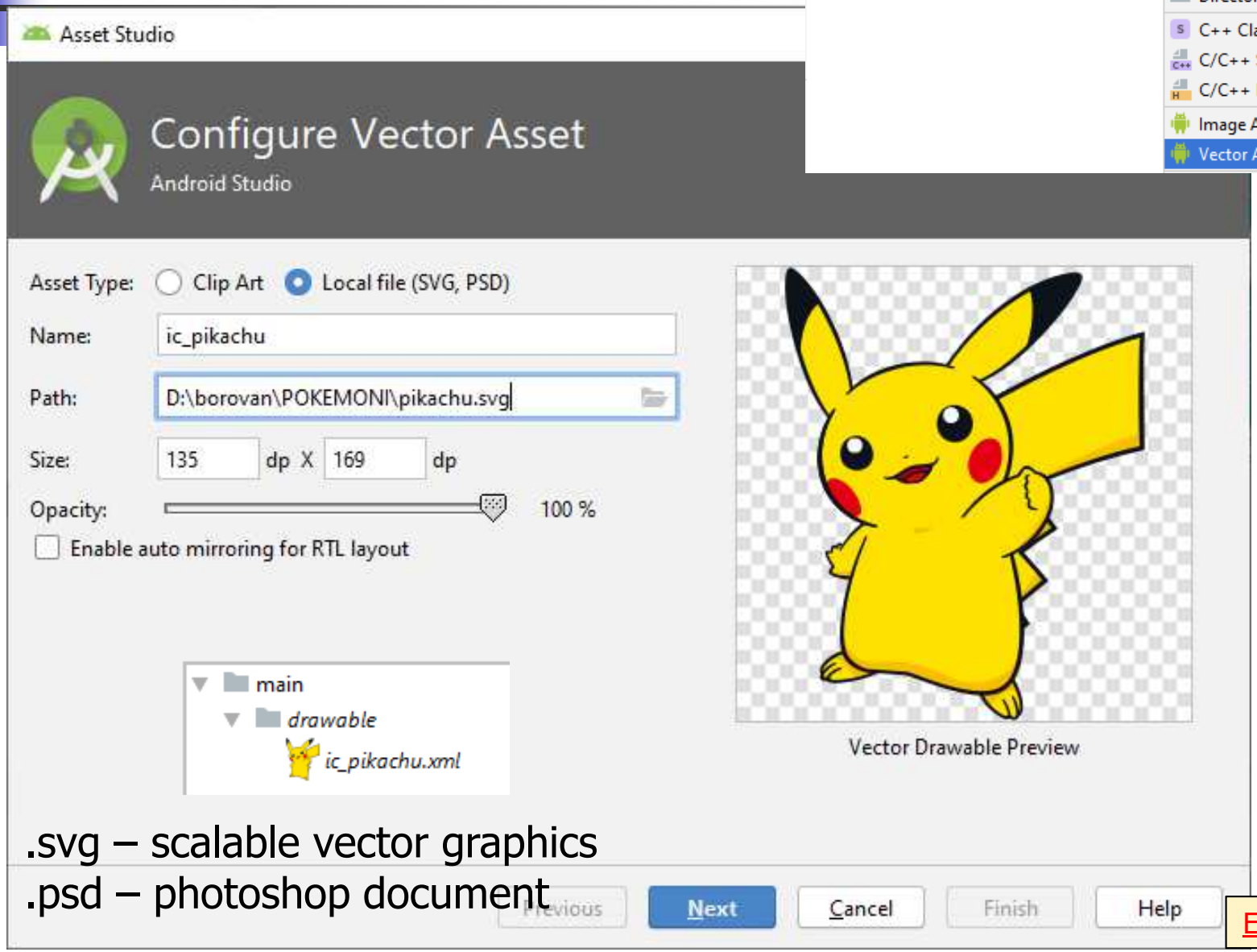


- .png,. jpg, .bmp, ...
- cliparty
- texty

Pre .svg a .psd

(a použi Vector Asset Studio - New/VectorAsset)





.svg – scalable vector graphics

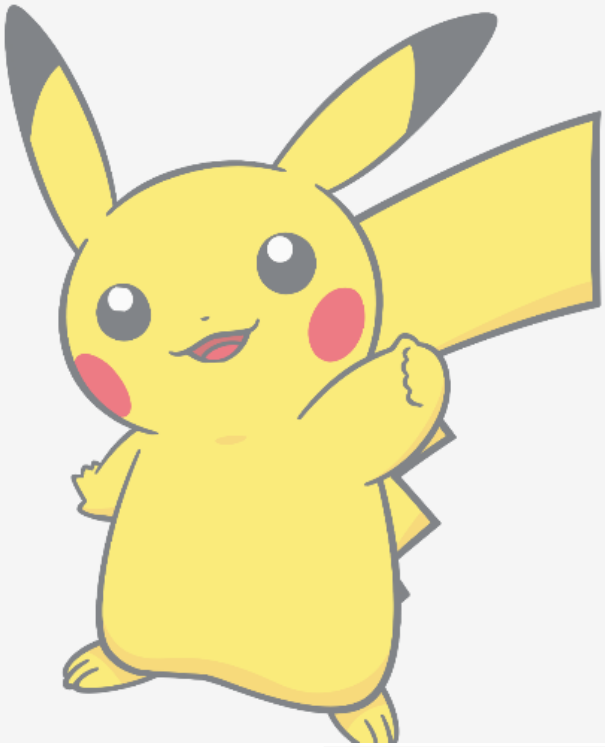
.psd – photoshop document

EmptyApp2021.zip

Vektorový pikachu

```
1   <vector android:alpha="0.5" android:height="169dp"
2       android:viewportHeight="169.1" android:viewportWidth="134.7"
3       android:width="135dp" xmlns:android="http://schemas.android.c
4       <path android:fillColor="#763a00" android:pathData="M79.6,140
5       <path android:fillColor="#ffe100" android:pathData="M133.5,45
6       <path android:fillColor="#763a00" android:pathData="M78.75,12
7       <path android:fillColor="#542400" android:pathData="M79.95,14
8       <path android:fillColor="#f9be00" android:pathData="M112.45,7
9       <path android:fillColor="#f9be00" android:pathData="M98.35,93
10      <path android:fillColor="#f9be00" android:pathData="M97.55,11
11      <path android:fillColor="#542400" android:pathData="M87.95,12
12      <path android:fillColor="#0d131a" android:pathData="M134.6,24
13      <path android:fillColor="#0d131a" android:pathData="M13.25,12
14      <path android:fillColor="#ffe100" android:pathData="M92,8.1Q9
15      <path android:fillColor="#ffe100" android:pathData="M34.7,92.
16      <path android:fillColor="#ffe100" android:pathData="M34.7,92.
17      <path android:fillColor="#0d131a" android:pathData="M92,8.1Q9
18      <path android:fillColor="#ffe100" android:pathData="M16.7,146
19      <path android:fillColor="#ffe100" android:pathData="M73.55,15
20      <path android:fillColor="#b50005" android:pathData="M41.7,78.
21      <path android:fillColor="#e50012" android:pathData="M44.95,80
22      <path android:fillColor="#f9be00" android:pathData="M17.75,11
23      <path android:fillColor="#f9be00" android:pathData="M48,98.3Q
24      <path android:fillColor="#f9be00" android:pathData="M22,134.8
25      <path android:fillColor="#f9be00" android:pathData="M18.4,145
```

EmptyApp2021



EmptyApp2021.zip

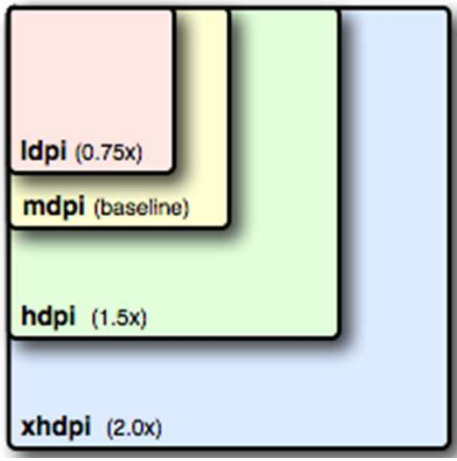
Resources/Drawables/Mipmap

(ikona - viacero rozlíšení)

http://developer.android.com/guide/practices/screens_support.html

pomer l/m/h/xh/x^{2}h/x^{3}h-dpi 3:4:6:8:12:16 - geom.postupnosť s koef. Sqrt(2) $\sqrt{2}$

- 36x36 for low-density (LDPI = ~ 120 dpi)
- 48x48 for medium-density (MDPI = ~ 160 dpi)
- 72x72 for high-density (HDPI = ~ 240 dpi)
- 96x96 for extra high-density (XHDPI = ~ 320 dpi)
- 144x144 for extra^2 high-density (XXHDPI = ~ 480 dpi)
- 192x192 for extra^3 high-density (XXXHDPI = ~ 640 dpi)



Snehulienka

(v geometrickom rade s quocientom sqrt(2))

$\sqrt{2}$

192x192 for extra³ high-density (XXXHDPI = ~ 640 dpi)

144x144 for extra² high-density (XXHDPI = ~ 480 dpi)

96x96 for extra high-density (XHDPI = ~ 320 dpi)

72x72 for high-density (HDPI = ~ 240 dpi)

48x48 for medium-density (MDPI = ~ 160 dpi)

```
imageView.setImageDrawable(
    ContextCompat.getDrawable(applicationContext,
        R.drawable.snehulienka resp.
        R.mipmap.snehulienka) )
```

Resources/Values

- string – reťazce separované z kódu, lokalizácia

```
<string name="app_name">YourFirstHello</string>
```

```
resources.getString(R.string.app_name)
```

- color - accessibility

```
<color name="transparent_green">#7700FF00</color>
```

```
resources.getColor(R.color.transparent_green)
```

- dimentions

```
<dimen name="absolutLarge">144dp</dimen>
```

```
resources.getDimension(R.dimen.absolutLarge)
```

- style – množina nastavení

```
<style name="myStyle">
      <item name="android:textSize">12sp</item>
      <item name="android:textColor">#FF00FF</item>
</style>
```

px = Pixels
in = Inches
mm = Millimeters
pt = Points, 1/72 of an inch
sp = Scale - Independent Pixels – používame pre veľkosť fontu
dp = Density - Independent Pixels – používame pre všetko ostatné

Resources/Values

zložitejšie hodnoty

- array-string/integer

```
<string-array name="poker">
    <item >full-hand</item>
    <item >postupka</item>
    <item >royal</item>
</string-array>
```

```
<integer-array name="coins">
    <item>1</item>
    <item>2</item>
    <item>5</item>
    <item>10</item>
    <item>20</item>
</integer-array>
```

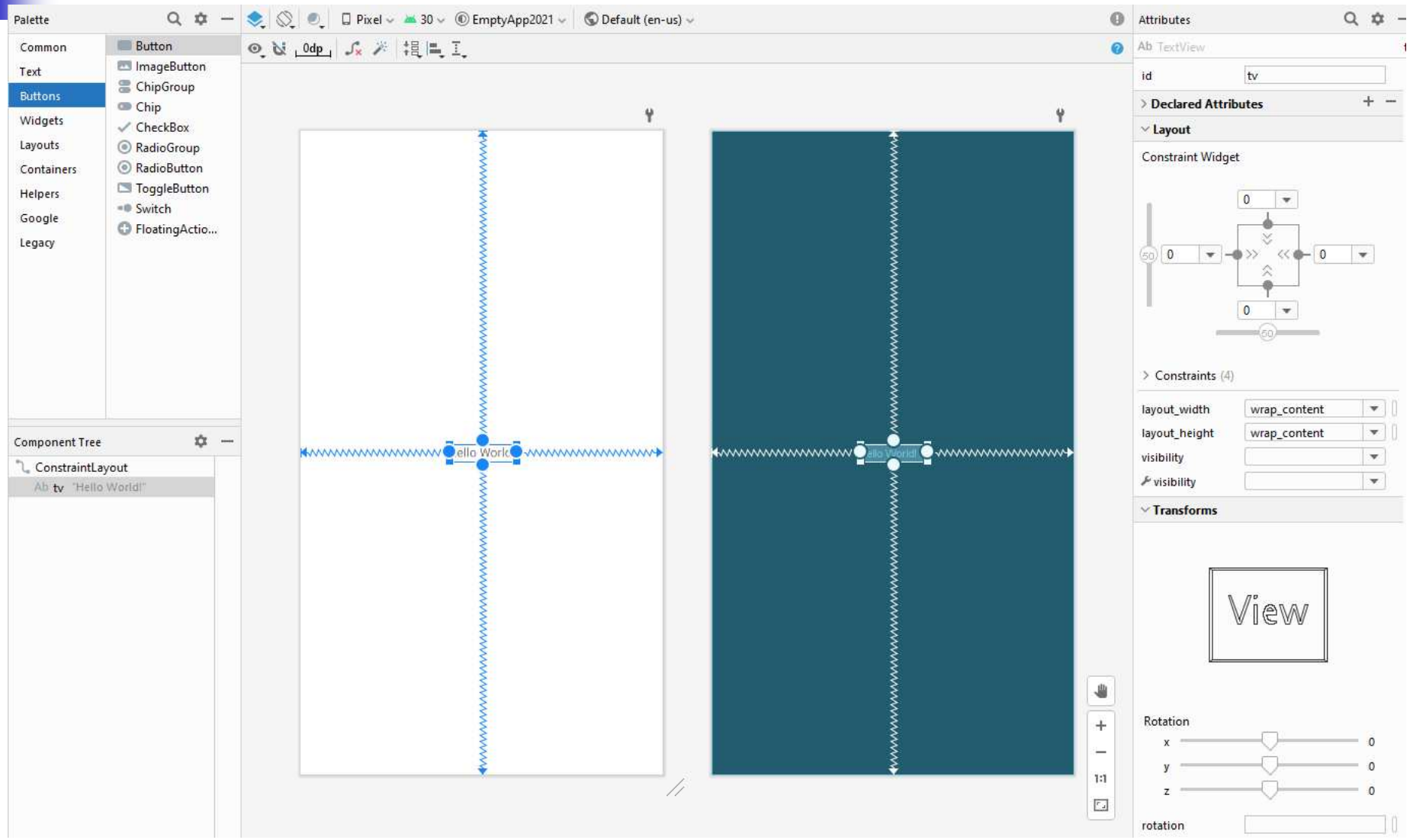
```
resources.getStringArray(R.array.otazky) :Array<String>
```

- plurals (quantity strings)

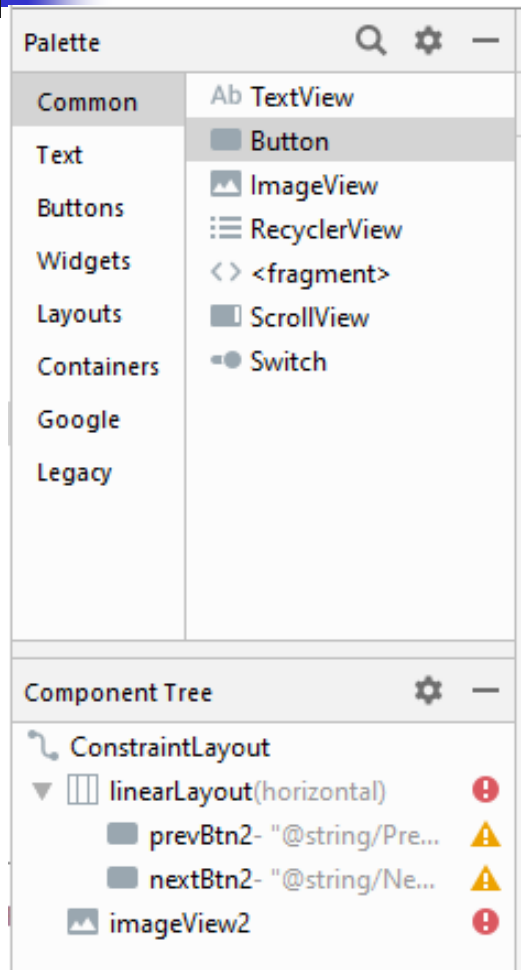
```
<plurals name="man">
    <item quantity="one">man</item>
    <item quantity="many">men</item>
    <item quantity="zero">paradis</item>
</plurals>
```

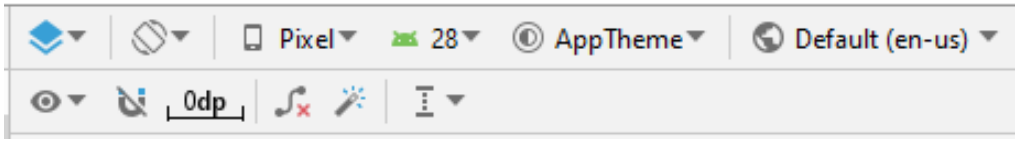
Resources/Layout
(Design View)

Konvencia:
XyzActivity[.kt/]
má layout
activity_xyz.xml



Layout Manager





Design/Blueprint/Design+Blueprint

Layout: Landscape/Portrait/…

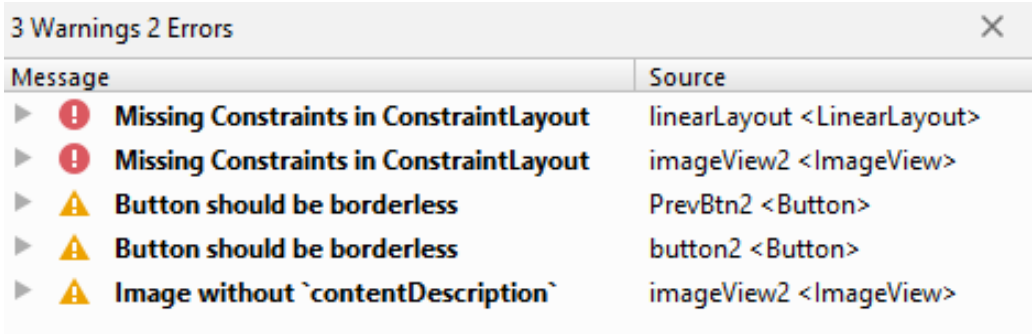
Pixel: AVD/Pixel2/Pixel#

API Level: 26/27/28/…

AppTheme :

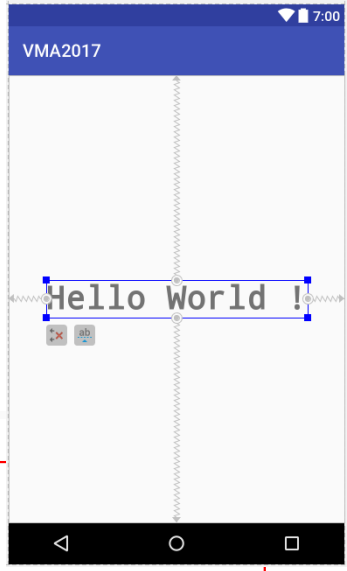
Default (en-us) : lokalizácie do rôznych jazykov

: warnings, errors



Resources/Layout

(Text View)




```
<android.support.constraint.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="pokus.example.com.vma2017.MainActivity">

         <TextView
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:fontFamily="monospace"
             android:text="Hello World!"
             android:textSize="36sp"
             android:textStyle="bold"
             app:layout_constraintBottom_toBottomOf="parent"
             app:layout_constraintLeft_toLeftOf="parent"
             app:layout_constraintRight_toRightOf="parent"
             app:layout_constraintTop_toTopOf="parent" />
    </android.support.constraint.ConstraintLayout>
```

wrap_content
fill_parent=
match_parent

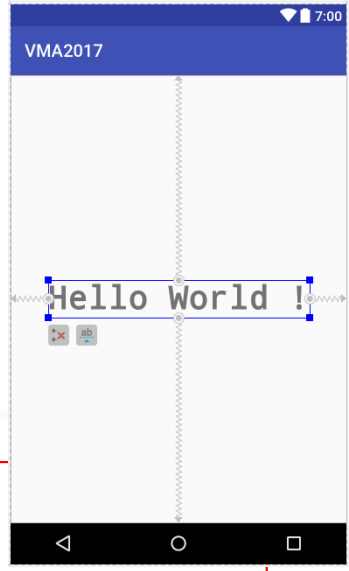
ConstraintLayout
Ab res "Hello World 1"

Hardcoded string "Hello World 1", should use `@string` resource

Bad style

Resources/Layout

(Text View)




```
<android.support.constraint.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="pokus.example.com.vma2017.MainActivity">

         <TextView
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:fontFamily="monospace"
             android:text="@string/IntroString"
             android:textSize="@dimen/reallyBigFont"
             android:textStyle="bold"
             app:layout_constraintBottom_toBottomOf="parent"
             app:layout_constraintLeft_toLeftOf="parent"
             app:layout_constraintRight_toRightOf="parent"
             app:layout_constraintTop_toTopOf="parent" />
    </android.support.constraint.ConstraintLayout>
```

wrap_content
fill_parent=
match_parent

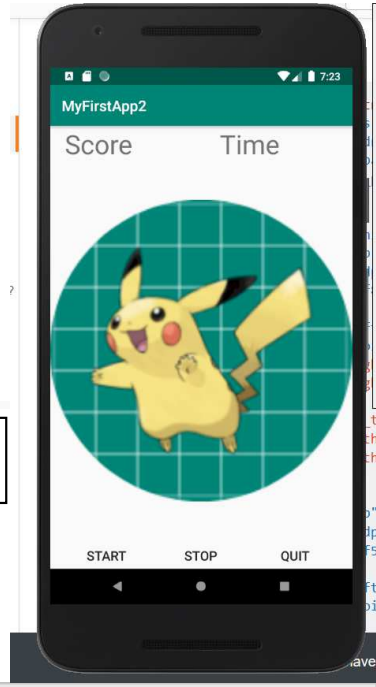
```
<resources>
    <string name="app_name">VMA2017</st
    <string name="IntroString">Hello Wo
</resources>
```

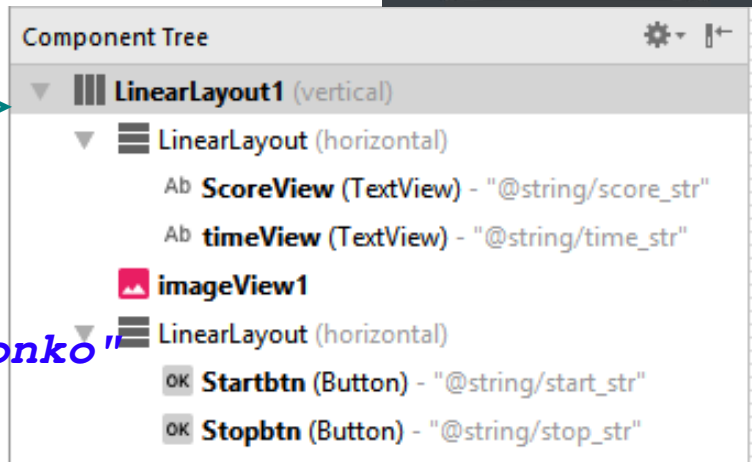
```
<resources>
    <dimen name="reallyBigFont">30
</reso
```

Ako by to malo vyzerať

```
<LinearLayout
    <TextView
        android:id="@+id/ScoreView"
        android:text="@string/score_str"/>
    <TextView
        android:id="@+id/timeView"
        android:text="@string/time_str" />
</LinearLayout>
<ImageView
    android:id="@+id/imageView1"
    android:contentDescription="@string/dronko"
    android:src="@drawable/ic_launcher" />
<LinearLayout
    <Button
        android:id="@+id/Startbtn"
        android:text="@string/start_str" />
    <Button
        android:id="@+id/Stopbtn"
        android:text="@string/stop_str" />
</LinearLayout>
```

Žiadne warnings





zjednodušené pre účely slajdu

MyFirstApp22.zip

Väzba komponentov v kóde

- `val btn = findViewById<Button>(R.id.button)`
- `val iv = findViewById<ImageView>(R.id.imageView1)`

- plugin kotlin-android-extensions

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'kotlin-android-extensions'
}
```

- import syntetic pomocou Alt-Enter
- `import kotlinx.android.synthetic.main.activity_main.*`

Old school, java style

Deprecated 2017-2020

@Parcelize od 2020

```
    val s = findViewById<Button>(R.id.startBtn)
    val iv = findViewById<ImageView>(R.id.imageView)

    startBtn.setText("Start")
}
```

Unresolved reference: startBtn

Create local variable 'startBtn' Alt+Shift+Enter More actions... Alt+Enter

Logovanie

Tri najbežnejšie spôsoby ako (logovať, debugovať):

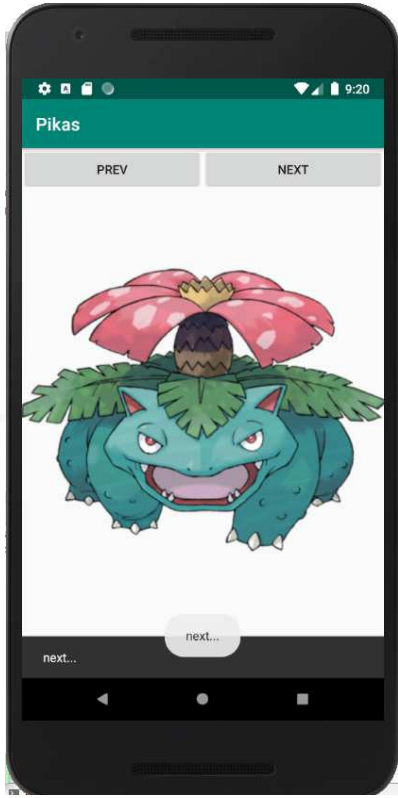
- Log
- Toast
- Snackbar – to chce pridať závislosť do build.gradle

```
dependencies {
    implementation 'com.android.support:design:28.0.0'
import com.google.android.material.snackbar.Snackbar
```

```
prevBtn2.setOnClickListener({
    Toast.makeText(this, "prev...", Toast.LENGTH_SHORT).show()

    Log.d(TAG, "prev...")

    Snackbar.make(view, "prev...",
        Snackbar.LENGTH_SHORT).setAction("Action", null).show()
    ...
    if (--i < 0) i += imgs.size
    imageView2.setImageDrawable(imgs[i])
})
```

Pikas2.zip

Pikas

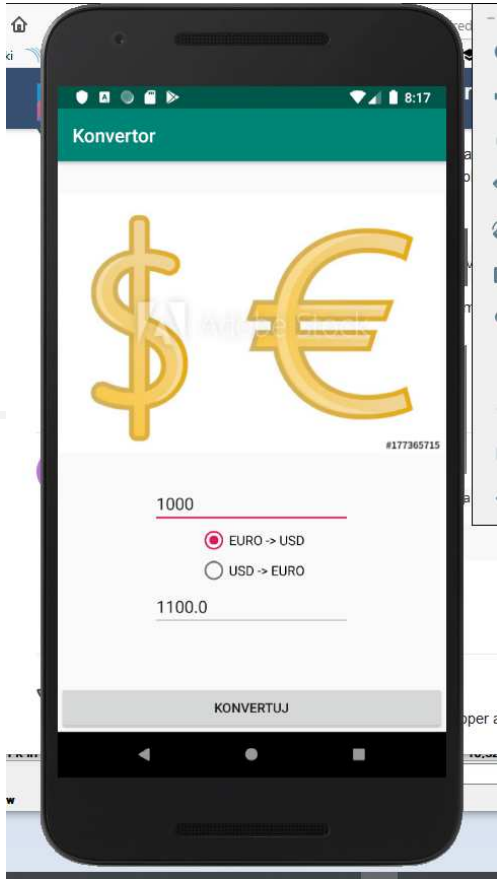



```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    var i = 0
    var imgs = arrayOf(
      ContextCompat.getDrawable(applicationContext,
                                R.drawable.butterfree),
                               ..., ..., ...     )
      imageView2.setImageDrawable(imgs[i])
      prevBtn2.setOnClickListener({
          Toast.makeText(this, "prev...", Toast.LENGTH_SHORT).show()
          if (--i < 0) i += imgs.size
          imageView2.setImageDrawable(imgs[i])
      })
      nextBtn2.setOnClickListener({
          Toast.makeText(this, "next...", Toast.LENGTH_LONG).show()
          i = (++i)%imgs.size
          imageView2.setImageDrawable(imgs[i])
     })
}
```

Pikas2.zip

Konvertor EURO USD

(logika)



Jednoduchá aplikácia na konverziu kurzov USD EURO

- s modifikovateľným TextView pre zadanie sumy, reálneho čísla
- RadioButtonom pre výber smeru konverzie
- s nemodifikovateľným poľom pre výsledok
- Button Konvertuj pre vykonanie akcie

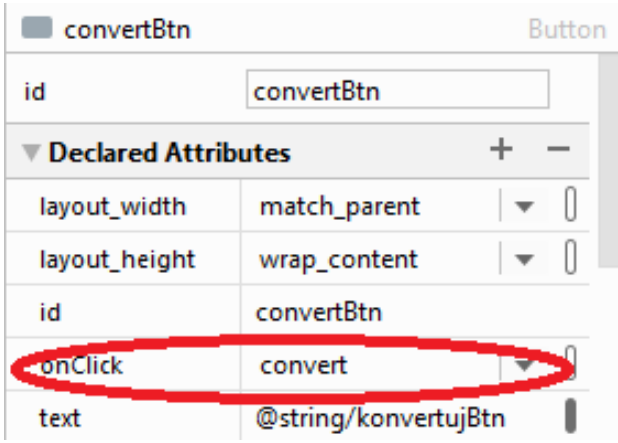
```
override fun onCreate(savedInstanceState: Bundle?)
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    convertBtn.setOnClickListener({
        Toast.makeText(this, "convert",Toast.LENGTH_SHORT).show();
        if (inputText.text.isNotEmpty()) {
            val input = inputText.text.toString().toFloat();
            var output = input
            val exchangeRate = 0.96f
            if (eur2usd.isChecked) output = exchangeRate * output
            if (usd2eur.isChecked) output = output / exchangeRate
            outputText.setText("$output")        // set
```

Klik na Konvertuj

Konvertor.zip

Konvertor EURO USD

(setOnClickListener)




```
// very old fashion
    val cBtn = findViewById<Button>(R.id.convertBtn)
    cBtn.setOnClickListener( { v -> convert(v) } )
    cBtn.setOnClickListener { convert(it) }

// old fashion
    convertBtn.setOnClickListener { v -> convert(v) }
    convertBtn.setOnClickListener { convert(it) }


    fun convert(v: View) {
        Toast.makeText(this, "convert", Toast.LENGTH_SHORT).show()
        if (inputText.text.isNotEmpty()) {
            val input = inputText.text.toString().toFloat();
            var output = input
             val exchangeRate = 0.96f
             if (eur2usd.isChecked) output = exchangeRate * output
             if (usd2eur.isChecked) output = output / exchangeRate
           outputText.setText("${output.format(2)}")     }}
    fun Float.format(digits: Int) =
        java.lang.String.format("%.${digits}f", this)
```

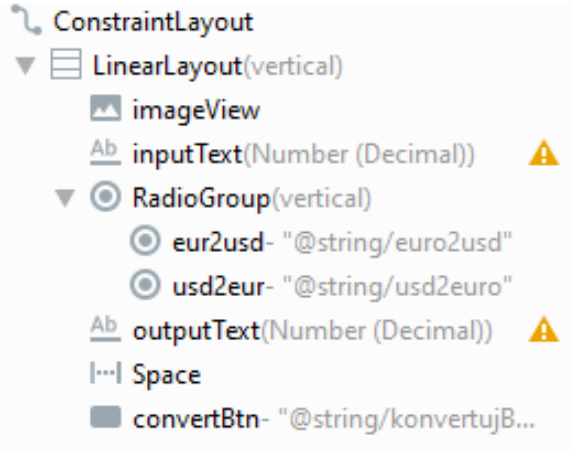
extension metóda Float

Konvertor.zip

a

Konvertor EURO USD
(layout)

```
<LinearLayout
    <ImageView …/>
    <EditText …/>
    <RadioGroup
        <RadioButton …/>
        <RadioButton …/>
    </RadioGroup>
    <EditText …/>
    <Space …/>
    <Button …/>
</LinearLayout>
```



Konvertor.zip

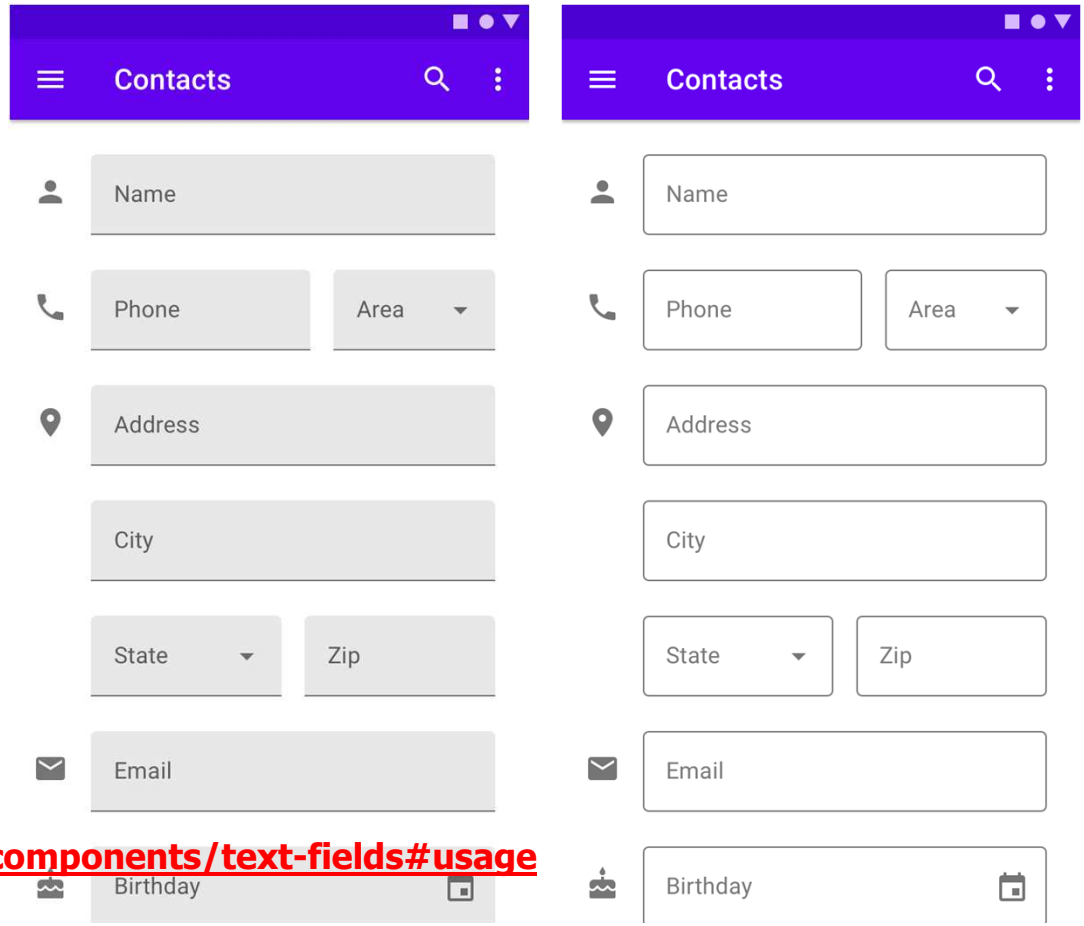
Text Fields

prvý dotyk s Material Design

Material Design je Google knižnica GUI komponentov unifikovaná pre Android, iOS, Flutter, web, ...

```
dependencies {
implementation 'com.google.android.material:material:1.4.0`
```

- zahŕňa Button, Text fields, SnackBars, Sliders, a mnoho ďalších vizuálnych komponentov Views

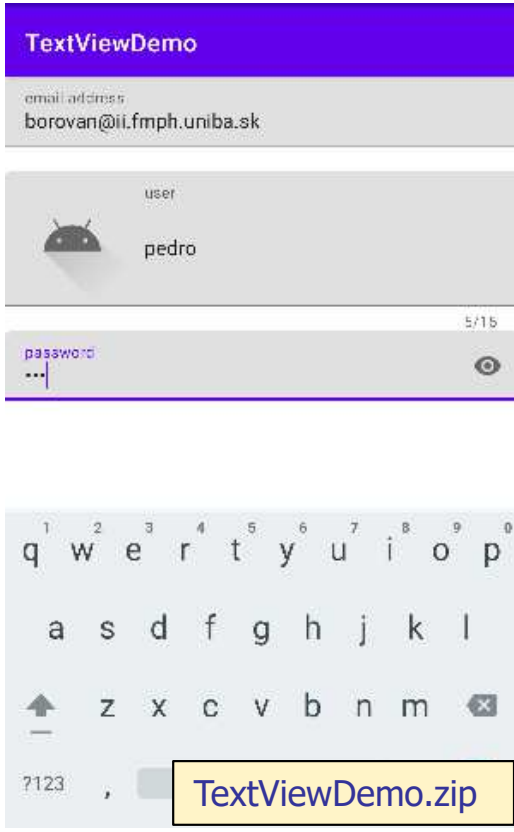


https://material.io/components/text-fields#usage

TextViewDemo.zip

TextInput[Layout/EditText]

```
<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:startIconDrawable="@drawable/ic_launcher_foreground"
    app:startIconContentDescription="@string/iconDescription"
    app:startIconCheckable="true"
    app:endIconMode="clear_text"
    app:counterEnabled="true"
    app:counterMaxLength="15"
    app:errorEnabled="true">
    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/userTV"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/userHint"
        android:maxLength="15"
        android:inputType="textPersonName" />
</com.google.android.material.textfield.TextInputLayout>
```



https://material.io/components/text-fields#usage

TextWatcher

```
val textWatcher = object : TextWatcher {     // singleton
    override fun beforeTextChanged(s: CharSequence, ...) { }
    override fun afterTextChanged(s: Editable?) { }
    override fun onTextChanged(s: CharSequence?, ...) {
        button.isEnabled =
                  emailTV.text?.isNotEmpty()?:false &&
                  userTV.text?.isNotEmpty()?:false &&
                  passwordTV.text?.isNotEmpty()?:false
        button.isEnabled =
            if (emailTV.text != null && userTV.text != null &&
                passwordTV.text != null)
              emailTV.text!!.isNotEmpty() &&
              userTV.text!!.isNotEmpty() &&
              passwordTV.text!!.isNotEmpty()
            else
                false
}   }
emailTV.addTextChangedListener(textWatcher)
userTV.addTextChangedListener(textWatcher)
passwordTV.addTextChangedListener(textWatcher)
```

TextViewDemo.zip

Príklad jednoduchej aplikácie

(ktorú sme si vyklikali minule)

Ilustrovali sme:

- príklad návrhu (vyklikania) jednoduchého GUI (single activity app)
- logovanie udalostí ako efektívny prostriedok ladenia pomocou
  - Log.d(…)
  - Toast.make(…)
  - Snackbar.make(…)
- používanie Image/Vector Asset (drawable/mipmap)
- používanie resource editora (pri definovaní strings.xml)
- používanie layout editora pri tvorbe rozhrania (ešte bude)
- eventhandler (**`.setOnClickListener`**) previazané cez
  - **`findViewById<Button>(R.id.`*`quitBtn`*`)`**
  - **`prevBtn.setOnClickListener({ })`**
  - property `android:onClick="nextOnClickListener"`

Pikas2.zip

Nestihli sme:

- aktivitu a jej životný cyklus

Logovanie
(rekapitulácia)

Tri najbežnejšie spôsoby:

- Log – loguje do okna Logcat, filtrujte podľa `TAGu` metódy `Log.d(TAG,`
- Toast – potrebuje `Context` (zjednodušene aktivita, v ktorej sa toastuje)
- Snackbar – to chce pridať závislosť do build.gradle a import snackbaru

```
dependencies {
    implementation 'com.android.support:design:28.0.0'}
import com.google.android.material.snackbar.Snackbar
```

```
prevBtn2.setOnClickListener({
    Toast.makeText(this, "prev...", Toast.LENGTH_SHORT).show()

    Log.d(TAG, "prev...")

    Snackbar.make(it, "next...",
        Snackbar.LENGTH_SHORT).setAction("Action", null).show()
    alebo .setAction(R.string.action,
         View.OnClickListener { nextOnClickListener(it) }).show()
...
})
```

Project:Pikas2.zip

Pikas

(rekapitulácia)

activity entry point

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    var i = 0
    var imgs = arrayOf(
      ContextCompat.getDrawable(applicationContext,
                                R.drawable.butterfree),
      ...          )
      imageView2.setImageDrawable(imgs[i])
      prevBtn2.setOnClickListener({
          Toast.makeText(this, "prev...", Toast.LENGTH_SHORT).show()
          if (--i < 0) i += imgs.size
          imageView2.setImageDrawable(imgs[i])
      })
      nextBtn2.setOnClickListener({
          Toast.makeText(this, "next...", Toast.LENGTH_LONG).show()
          i = (++i)%imgs.size
          imageView2.setImageDrawable(imgs[i])
    })
}
```

logovanie

View(s)

Pikas2.zip

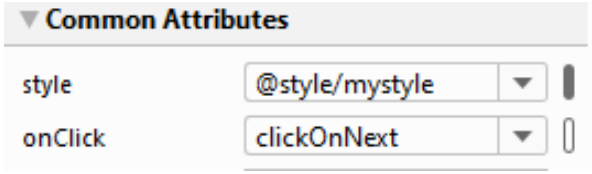
Pikas

(stav sa mieša s views a logikou – riešenie príde)

const final

State

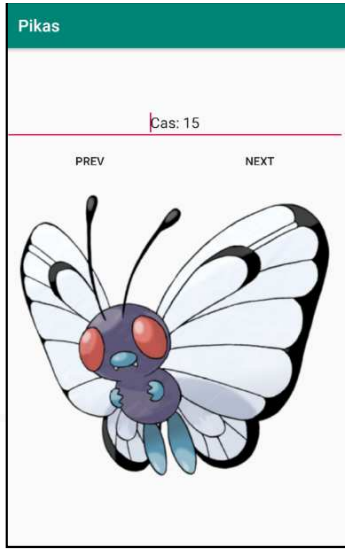
```
val TAG = "PIKAS"
var i = 0
var imgs = arrayOf<Drawable?>()
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    imgs = arrayOf(ContextCompat.getDrawable(applicationContext,
                                             R.drawable.butterfree), … )
    imageView2.setImageDrawable(imgs[i])
    prevBtn2.setOnClickListener {          // it:View -> { ... }
        if (--i < 0) i += imgs.size
         imageView2.setImageDrawable(imgs[i])
    }
}
// prepojene cez property android:onClick="nextOnClickListener"
fun nextOnClickListener(v: View) {
    i = (++i) % imgs.size
    imageView2.setImageDrawable(imgs[i])
}
```



Pikas2.zip

Pikas

(asynchrónnosť - timer)



pomocou **`java.util.Timer`**

```
Timer("tik-tak").schedule(1000,1000) { // delay, period
    Log.d(TAG, "onTICK")
    cas++
    runOnUiThread { time.setText("Cas: $cas ") }
}.run()
```

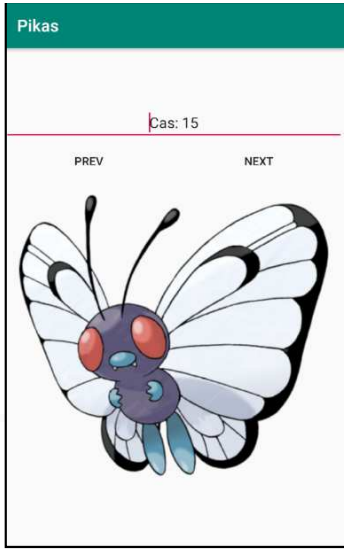
- nezabudnite na .run()
- runOnUiThread
  - má argument java.lang.Runnable, ktorý vykoná v hlavnom GUI vlákne

```
zabitie timera:
override fun onPause() {
    super.onPause()
    timer.cancel()
}
```

Pikas2.zip

Pikas

(asynchrónnosť – count down)



pomocou **android.os.CountDownTimer**

```
object:CountDownTimer(20000, 1000) { // 20sek, tik po 1sek
                                     // how long, period
    override fun onTick(millisUntilFinished: Long) {
        Log.d(TAG, "onTICK")
        runOnUiThread {
            time.setText("Cas: ${millisUntilFinished/1000}") }
    }

    override fun onFinish() {
        Log.d(TAG, "onFinish")
        exitProcess(-1)
    }
}.start()
```

tik

game over

ukončenie appky

Pikas2.zip

Životný cyklus apky

(prvý – zjednodušený nástrel)

global: 0
local: 0
shared: 0

- Alt-Insert = Generate Override Implemented Methods:
  - `override fun onDestroy()`
  - `override fun onPause()`
  - `override fun onRestart()`
  - `override fun onRestoreInstanceState(Bundle savedInstanceState)`
  - `override fun onResume()`
  - `override fun onSaveInstanceState(Bundle outState)`
  - `override fun onStart()`
  - `override fun onStop()`
- do každej metódy dáme kontrolný výpis, aby sme pochopili životný cyklus

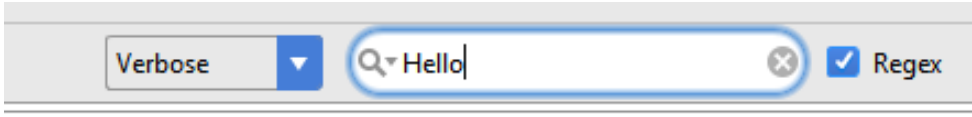
```
override fun onCreate(Bundle savedInstanceState?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    Log.d("CYKLUS","onCreate") // LOGUJTE, LOGUJTE, LOGUJTE
}
```

tag vhodný na filtrovanie

AppLifeCycle.zip

LogCat

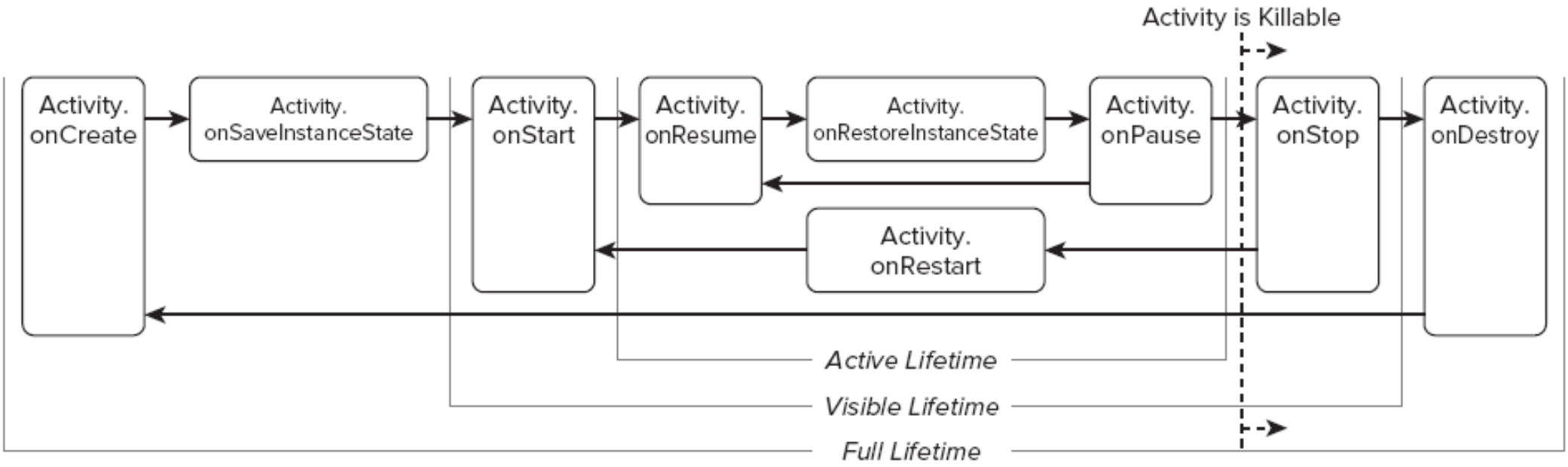
(Filtrovanie logov)



- 10-13 12:55:41.091: D/Hello(405): onCreate
- 10-13 12:55:41.091: D/Hello(405): onStart
- 10-13 12:55:41.100: D/Hello(405): onResume

kill

- 10-13 12:56:45.061: D/Hello(405): onPause
- 10-13 12:56:45.681: D/Hello(405): onStop
- 10-13 12:56:45.681: D/Hello(405): onDestroy



zdroj: Reto Meier: PA2AD

AppLifeCycle.zip

Persistencia
(prvý dotyk)

global: 0
local: 0
shared: 0

- **`globalCounter`** je premenná, ktorá sa
  - pri **`onSaveInstanceState`** uloží do Bundle (HashMap<String, Value>)
  - pri **`onCreate(savedInstanceState: Bundle?)`** príde táto Bundle ako argument
- **`localCounter`** je bežná lokálna triedna premená v MainActivity
- **`sharedCounter`** je premenná, ktorá sa ukladá
  - pri **`onPause`** sa uloží do **`SharedPreferences`** (HashMap<String, Value>)
  - pri **`onResume`** sa prečíta zo **`SharedPreferences`**
- všetky tri premenné sa inkrementujú pri onPause

Zistíte, že:

- aktivita, ak zmení orientáciu, tak sa reštartne, vytvorí sa nová inštancia a zavolá sa **`onCreate`**. Preto premenná **`localCounter`** sa vynuluje.
- ak si chcete niečo uchovať aj po zmene orientácie aktivity, treba to uložiť do bundle, zapíšete to tam v **`onSaveInstanceState`** a prečítate v **`onCreate`**
- ak si chcete niečo uchovať aj po reštarte aplikácie, treba to uložiť do **`SharedPreferences`**

AppLifeCycle.zip

Bundle?

Bundle má metódy [put/get][Int/Boolean/Char/Float/Any/...]

```
override fun onRestoreInstanceState(
            savedInstanceState: Bundle?) {
  super.onRestoreInstanceState(savedInstanceState)
  globalCounter = savedInstanceState?.getInt("COUNTER")?:0
  ...
}

override fun onSaveInstanceState(outState: Bundle?,
                  outPersistentState: PersistableBundle?) {
  super.onSaveInstanceState(outState, outPersistentState)
  outState?.putInt("COUNTER", globalCounter)
  ...
}
```

AppLifeCycle.zip

SharedPreferences

SharedPreferences má metódy get[Int/Boolean/Char/Float/Any/...]

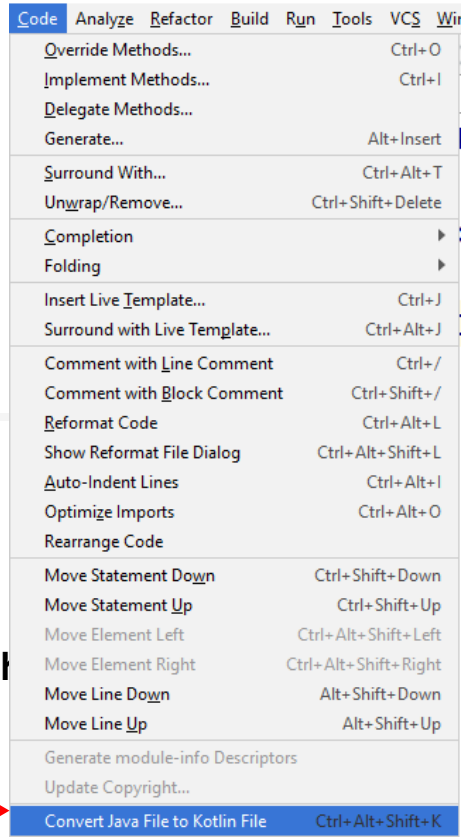
```
private lateinit var preferences: SharedPreferences
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    preferences = getSharedPreferences("lifecycle",
                                       Context.MODE_PRIVATE)
}
override fun onResume() {
    sharedCounter = preferences.getInt("kluc",0)
}
override fun onPause() {
    preferences.edit {
          this.putInt("kluc", sharedCounter)
          this.commit()
      }
}
```

AppLifeCycle.zip

Pikas.java

(auto-generovaný Code/Convert Java->Kotlin)

```
i = 0
iv.setImageDrawable(images[i])

quit.setOnClickListener { v ->
    Toast.makeText(this, "BYE BYE", Toast.LENGTH_LONG).sh
    this.finishAffinity()
}

prev.setOnClickListener {
    Log.d("PIKA", "onPREV")
    Toast.makeText(this@MainActivity, "PREV", Toast.LENGTH_SHORT).show()
    i--
    if (i < 0) i = images.size - 1
    iv.setImageDrawable(images[i])
}
next.setOnClickListener { v ->
    i++
    Log.d("PIKA", "onNEXT")
    Toast.makeText(this@MainActivity, "NEXT", Toast.LENGTH_SHORT).show()
    i = i % images.size
    iv.setImageDrawable(images[i])
}
```



v java projekte nájdete

Pikas.zip

Konverzie Java <-> Kotlin

- **Java -> Kotlin**

Code/Convert Java File to Kotlin File (neuzná sa to ako DÚ v Kotline)

- **Kotlin -> JVM Byte code**

Tools/Kotlin/Show Byte Code

- **Decompile Byte code (to Java)**

```
protected void onCreate(@Nullable Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   this.setContentView(2131296283);
   final ObjectRef images = new ObjectRef();
   final IntRef i = new IntRef();
   View var10000 = this.findViewById(2131165189);
   if (var10000 == null) {
     throw new TypeCastException("null cannot be cast to non-null type android.widget.Button");
   } else {
```

Čo je Kotlin ?

https://proandroiddev.com/modern-android-development-with-kotlin-september-2017-part-1-f976483f7bd6