



# 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`, view binding
- dobré zvyky pri návrhu layout
  - ako na warnings a errors
- Kotlin – nullables
  - operátory s tým spojené – tzv. Elvis operátor
- Cvičenie 2
  - vpisujete kódy do už pripravených templates
  - prémia: Piškvorky 3x3, a ďalšie

# Čo dostaneme zadarmo

(pokračujeme v minulej prednáške)



## Chapter 6

6. A Tour of the Android Studio User Interface

```
package com.example.emptyapplication2023
```

```
import android.os.Bundle
```

```
import androidx.appcompat.app.AppCompatActivity
```

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

```
    override fun onCreate(savedInstanceState: Bundle?) {
```

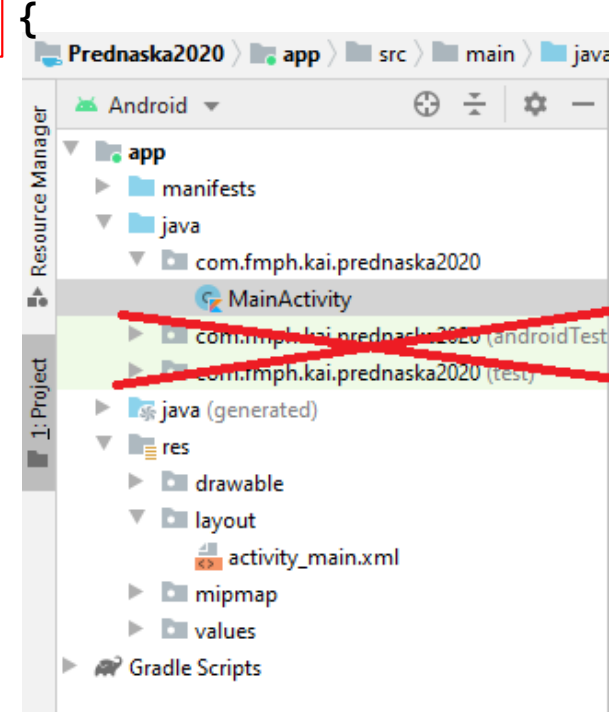
```
        super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.activity_main)
```

```
        // sem sme minule písali náš prvý kotlin kód
```

```
    }  
}
```

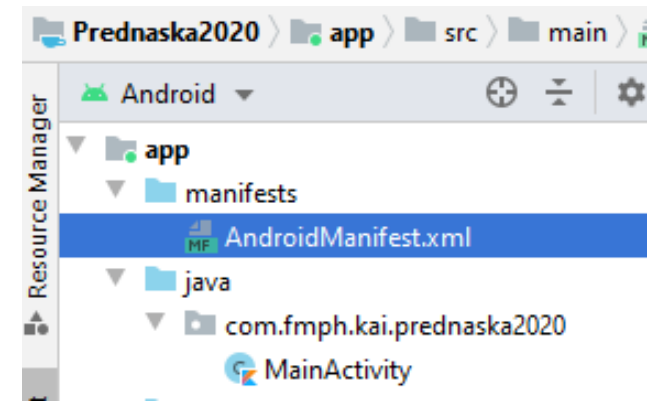
- MainActivity je inštancia triedy AppCompatActivity
- metóda onCreate() sa volá *niekde* v procese jej zobrazovania
- setContentView zobrazí layout podľa .xml popisu v  
R.layout.activity\_main
- argument savedInstanceState:Bundle? zatiaľ neriešte
- package androidTest a test môžete vymazať, pre prehľadnosť



[EmptyApplication2023.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>
```

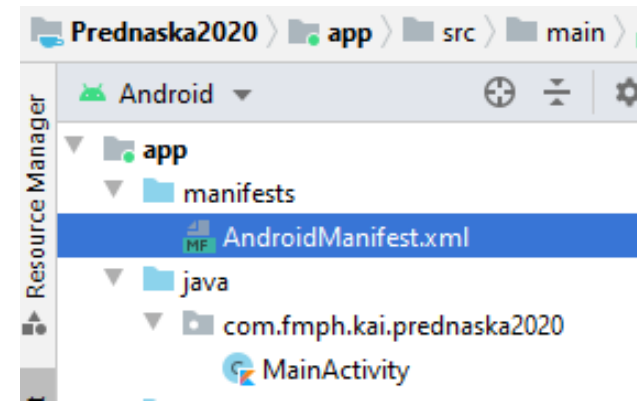
```
</manifest>
```

Alt-  
Enter

referencia na ikonu apky  
referencia meno apky



# AndroidManifest.xml



Hlavné tagy:

- **<application>** je jediný a popisuje ikony, logo, meno, štýl aplikácie
- **<activity>** môže ich byť viac a popisujú package definujúci aktivitu, intent aktivitu, 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 rozlíško 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](http://developer.android.com/guide/topics/manifest/manifest-intro.html)

# Anatómia Android aplikácie

- **Aktivita** – vizuálne komponenty, ktoré sa zobrazia na jednej obrazovke (single user interface screen)
- **Fragment** – aktivita môže byť poskladaná z viacerých fragmentov obsahujúcich vizuálne komponenty (Views), hlavnou výhodou je znovupoužiteľnosť fragmentu v rôznych aktivitách. Vzťah Aktivita vs. Fragment je teda many-to-many
- **Intent** – mechanizmus ako jedna aktivita vie spustiť inú. Explicitný intent referuje menom triedy aktivity, implicitný funkcionalitou ACTION\_VIDEO\_CAPTURE
- **Broadcast Intent-Receiver** – broadcast receiver registruje intent, na ktorý počúva-reaguje, a definuje akciu, ktorú vykoná, ak niekto vyšle intent
- **Servis** – beží na pozadí, nemá user interface
- **Content provider** – implementuje mechanizmus na zdieľanie dát aplikáciou, napr. prostredníctvom URI alebo SQL databázy, SQLite
- **Application Manifest** – xml súbor popisujúci aktivity, servisy, broadcast receivery, data providery, a práva (permissions) danej aplikácie
- **Resources** – xml reprezentácia užívateľských rozhraní, fontov, konštánt,...



# build.gradle

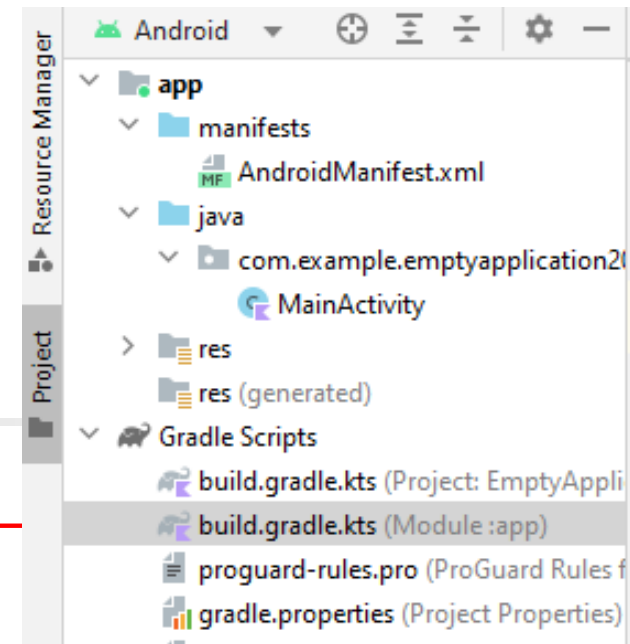
(konfiguračný súbor pre gradle)

Gradle je build tool, podobne ako make, maven

```
plugins {  
    id("com.android.application")  
    id("org.jetbrains.kotlin.android")  
}
```

```
android {  
    namespace = "com.example.emptyapplication2023"  
    compileSdk = 33  
    defaultConfig {  
        applicationId = "com.example.emptyapplication2023"  
        minSdk = 24  
        targetSdk = 33  
        versionCode = 1  
    }  
}
```

```
dependencies {  
    implementation("androidx.core:core-ktx:1.9.0")  
    implementation("androidx.appcompat:appcompat:1.6.1")  
    implementation("com.google.android.material:material:1.9.0")  
    implementation("androidx.constraintlayout:constraintlayout:2.1.1")  
}
```



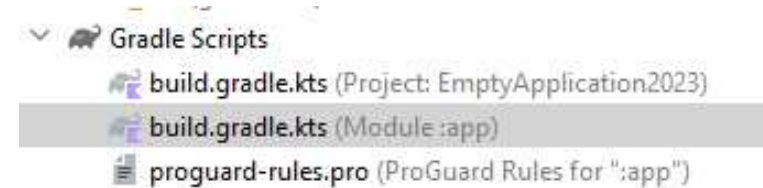
# build.gradle

(konfiguračný súbor pre gradle)

Gradle je build tool, podobne ako make, maven

...

Gradle súbory sú dva



Gradle zmenil formát z jazyka Groovy (ešte 2022) do kotlinu (poznáte príponou .kts)

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

```
android {  
    compileSdk 31  
    buildFeatures {  
        viewBinding = true  
    }  
    defaultConfig {  
        applicationId "com.example.emptyapp2021"  
        minSdk 23  
        targetSdk 31  
        versionCode 1  
    }  
}
```

```
plugins {  
    id("com.android.application")  
    id("org.jetbrains.kotlin.android")  
}  
android {  
    buildFeatures {  
        viewBinding = true  
    }  
    namespace = "com.example.emptyapplication2023"  
    compileSdk = 33  
    defaultConfig {  
        applicationId = "com.example.emptyapplication2023"  
        minSdk = 24  
        targetSdk = 33  
        versionCode = 1  
    }  
}
```



# MergedManifest

(spája AndroidManifest a build.gradle)

```
<manifest
  android:versionCode="1"
  android:versionName="1.0"
  package="com.example.emptyapplication2023"
  xmlns:android="http://schemas.android.com/apk/res/android" >
  <uses-sdk
    android:minSdkVersion="24"
    android:targetSdkVersion="33" />
  <permission
    android:name="com.example.emptyapplication2023.DYNAMIC_RECEIVER_NOT_E
    android:protectionLevel="signature" />
  <uses-permission
    android:name="com.example.emptyapplication2023.DYNAMIC_RECEIVER_NOT_E
  <application
    android:allowBackup="true"
    android:appComponentFactory="androidx.core.app.CoreComponentFactory"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.EmptyApplication2023" >
    <activity
      android:exported="true"
      android:name="com.example.emptyapplication2023.MainActivity" >
      <intent-filter
        <action
          android:name="android.intent.action.MAIN" />
        <category
          android:name="android.intent.category.LAUNCHER" />
```

## Manifest Sources

- ☐ EmptyApplication2023.app main manifest (this file)
- ☒ core:1.9.0 manifest
- ☐ build.gradle.kts injection

## Other Manifest Files

(Included in merge, but did not contribute any elements)

activity:1.6.0 manifest  
annotation-experimental:1.3.0 manifest  
appcompat-resources:1.6.1 manifest  
appcompat:1.6.1 manifest  
cardview:1.0.0 manifest  
constraintlayout:2.1.4 manifest  
coordinatorlayout:1.1.0 manifest  
core-ktx:1.9.0 manifest  
core-runtime:2.1.0 manifest  
cursoradapter:1.0.0 manifest  
customview:1.1.0 manifest  
documentfile:1.0.0 manifest  
drawerlayout:1.1.1 manifest  
dynamicanimation:1.0.0 manifest  
fragment:1.3.6 manifest  
interpolator:1.0.0 manifest  
legacy-support-core-utils:1.0.0 manifest  
lifecycle-livedata-core:2.5.1 manifest  
lifecycle-livedata:2.0.0 manifest  
lifecycle-runtime:2.5.1 manifest  
lifecycle-viewmodel-savedstate:2.5.1 manifest  
lifecycle-viewmodel:2.5.1 manifest  
loader:1.0.0 manifest  
localbroadcastmanager:1.0.0 manifest  
material:1.9.0 manifest  
print:1.0.0 manifest  
recyclerview:1.1.0 manifest  
savedstate:1.2.0 manifest  
transition:1.2.0 manifest  
vectordrawable-animated:1.1.0 manifest  
vectordrawable:1.1.0 manifest  
versionedparcelable:1.1.0 manifest

[EmptyApplication2023.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="black">#FF000000</color>  
  <color name="white">#FFFFFFFF</color>  
</resources>
```

```
<resources>  
  <string name="app_name">EmptyApplication2023</string>  
</resources>
```

# Bud' 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.





# Bud' kreatívny

(a použi Asset Studio - New/ImageAsset)

New

- Module
- Android Resource File
- Android Resource Directory
- Sample Data Directory
- File
- Scratch File Ctrl+Alt+Shift+Insert
- Directory
- C++ Class
- C/C++ Source File
- C/C++ Header File
- Image Asset

Asset Studio



## Configure Image Asset

Android Studio

Icon Type: Launcher Icons (Adaptive and Legacy)

Name: ic\_launcher

Foreground Layer Background Layer Legacy

Layer Name: ic\_launcher\_foreground

Source Asset

Asset Type: ☒ Image ☐ Clip Art ☐ Text

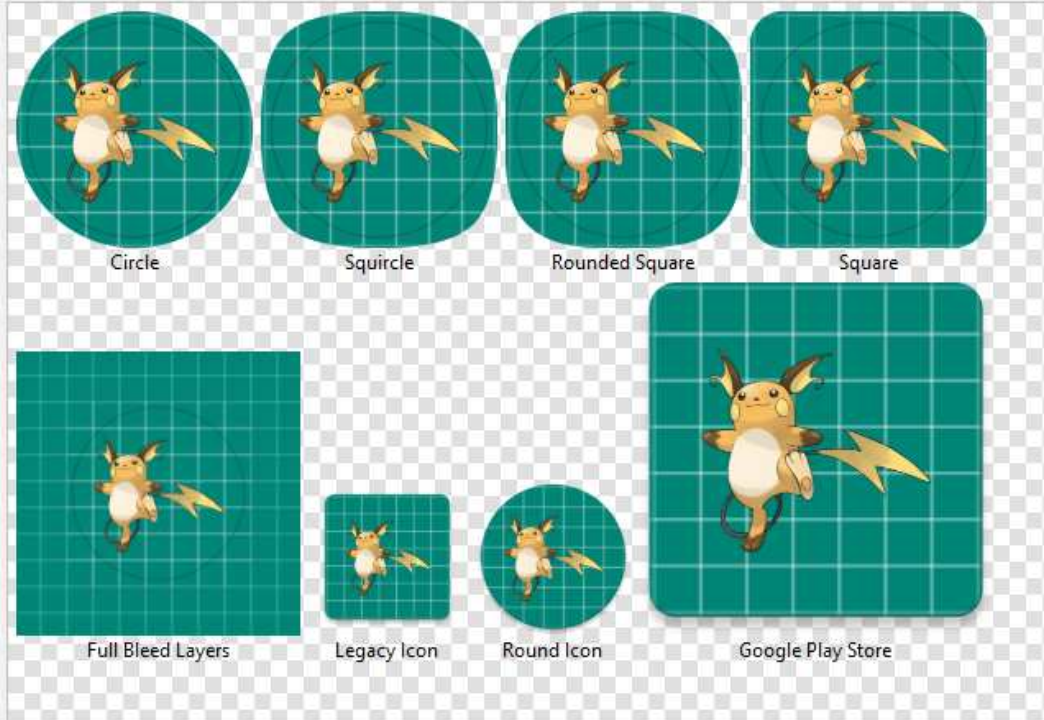
Path: app\src\main\res\drawable\raichu.png

Scaling

Trim: ☐ Yes ☒ No

Resize: 46 %

Preview xhdpi ☒ Show Safe Zone ☐ Show Grid



⚠ An icon with the same name already exists and will be overwritten.

Previous

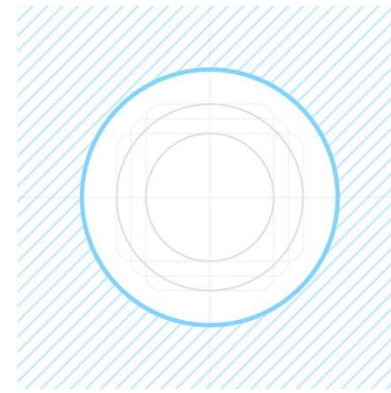
Next

Cancel

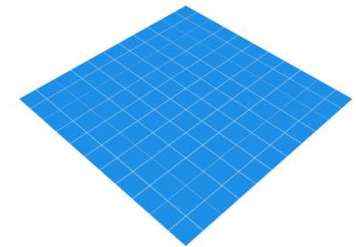
Finish

Help

# Adaptive icon



- funguje od Android-Oreo, API 26 – Android
- umožňuje zariadeniu vhodne škálovať ikonu podľa
  - zvoleného rozlíšenia 108dp, 66dp, ...
  - zvoleného orámovania



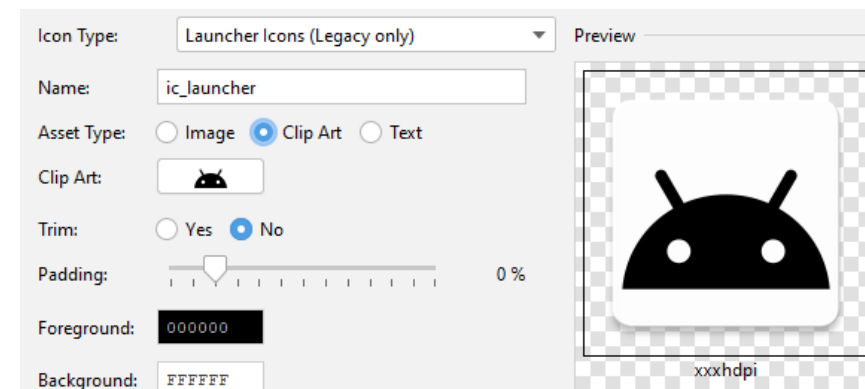
- adaptívna ikona má pozadie a popredie
- `<adaptive-icon`

```
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

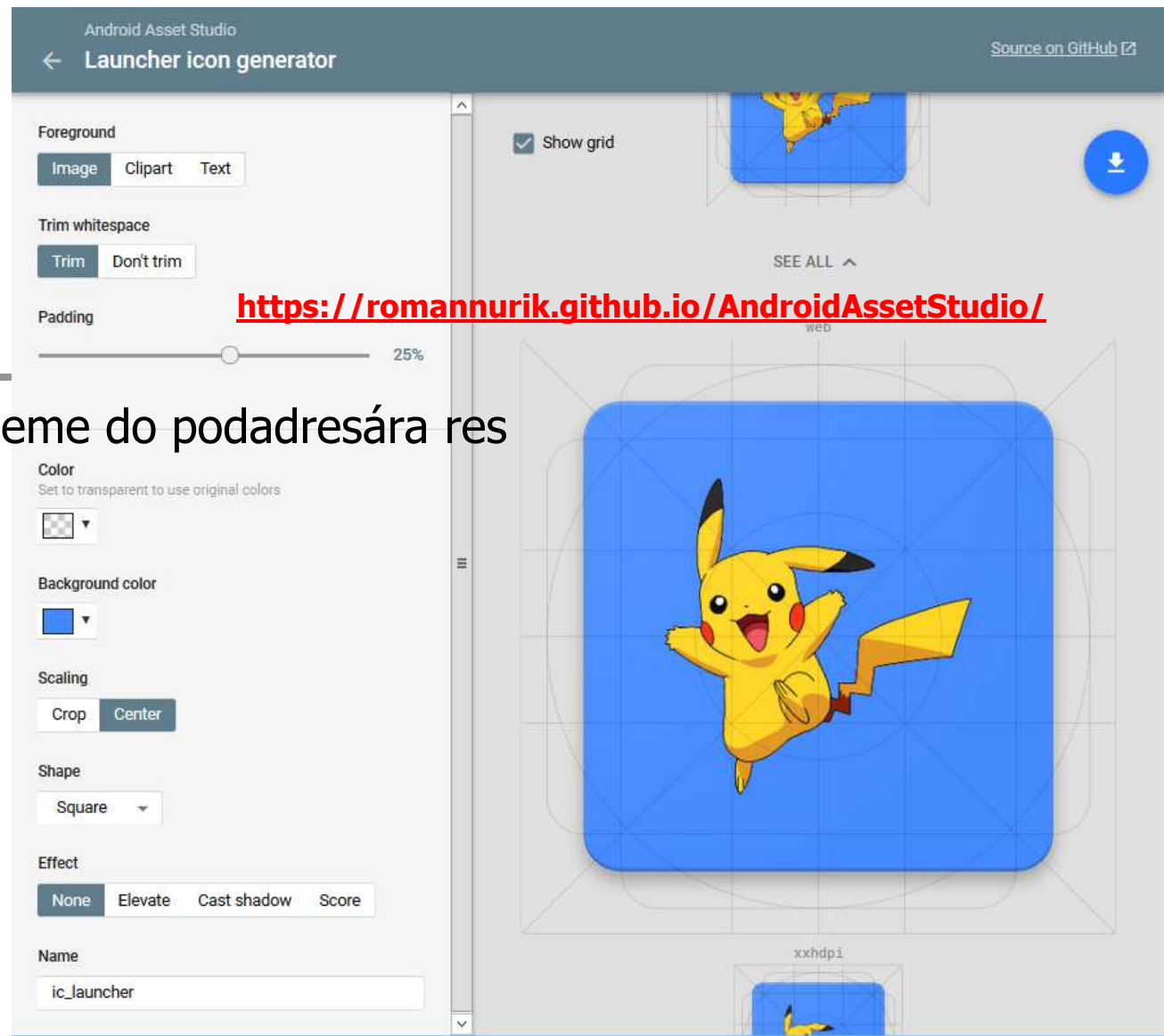
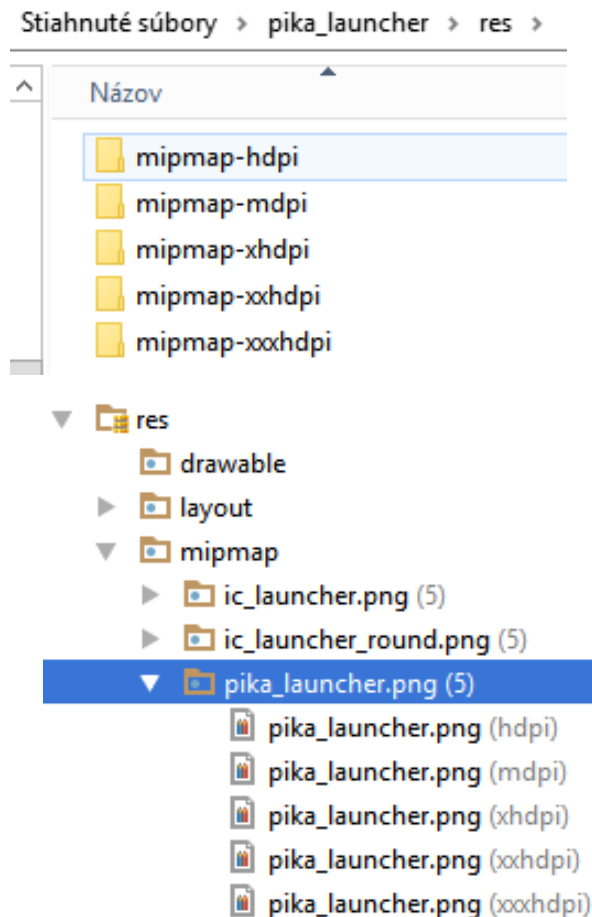


- legacy ikona je jednoduchšia



# Android Asset Studio Icon generator

výsledok priamo nakopírujeme do podadresára res  
Ikony/obrázky sa  
sa objavajú v projekte

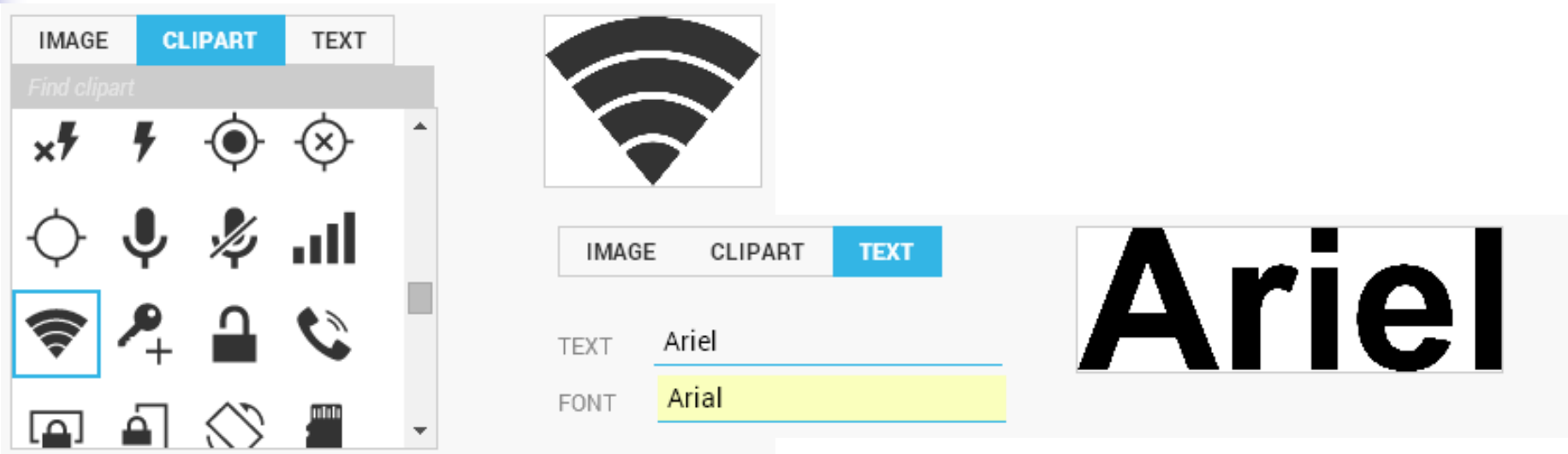


```
5
6
7 <application
8     android:allowBackup="true"
9     android:icon="@mipmap/pika_launcher"
10    android:label="@mipmap/pika_launcher"
11    android:roundIcon="@mipmap/ic_launcher_round"
12    android:supportRtl="true"
```

# Android Asset Studio

(jedna z alternatív)

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

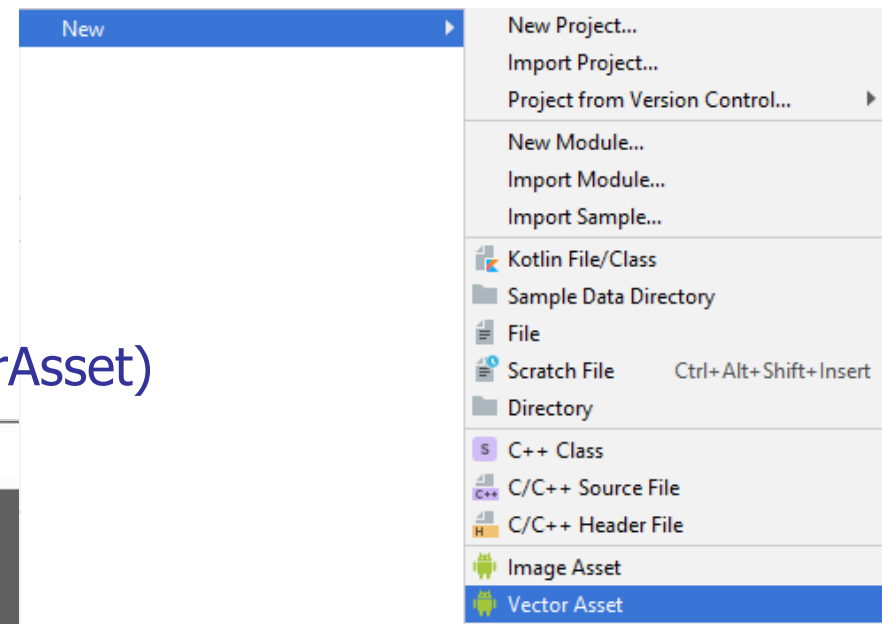
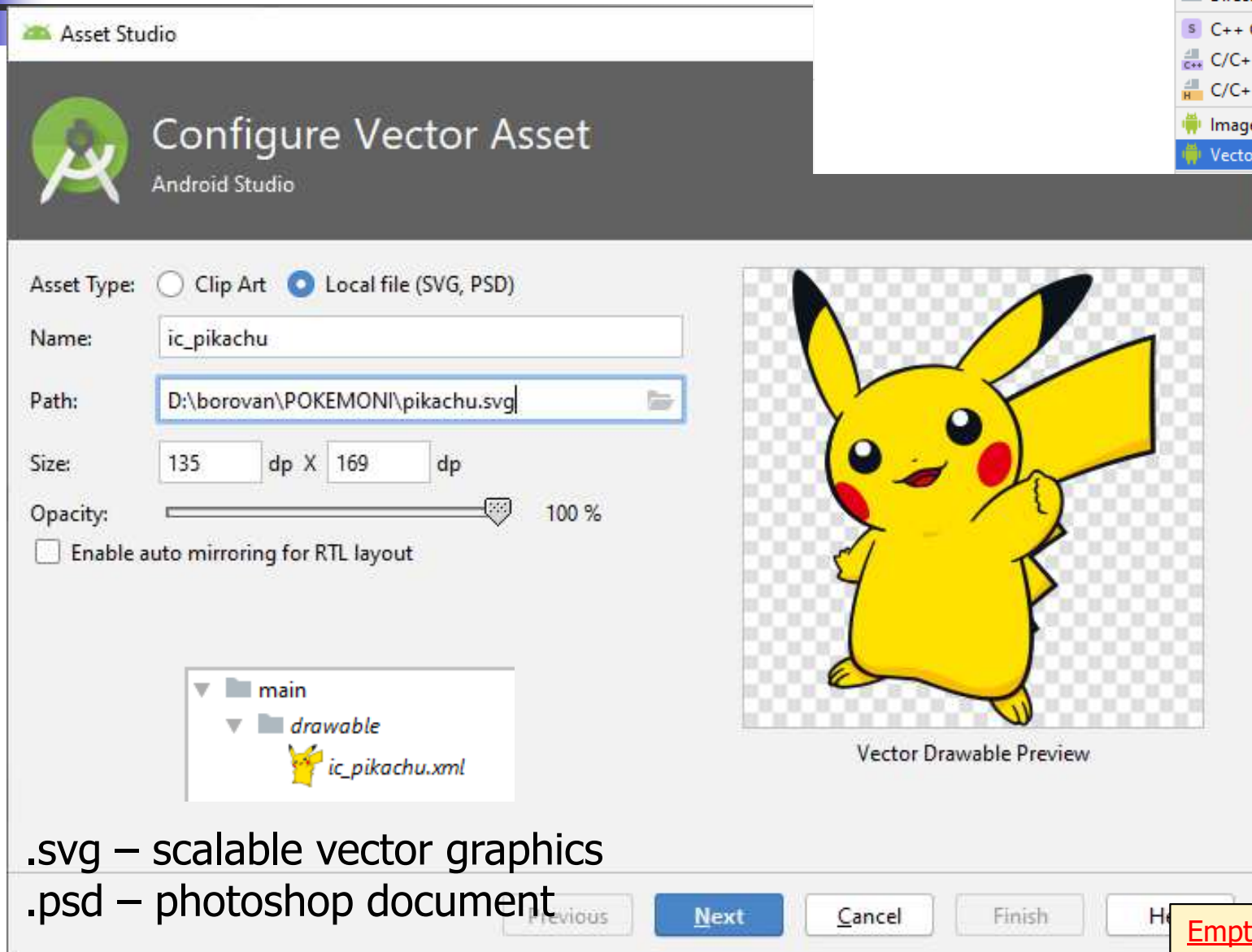


- .png, .jpg, .bmp, ...
- cliparty
- texty



# Pre .svg a .psd

(a použi Vector Asset Studio - New/VectorAsset)



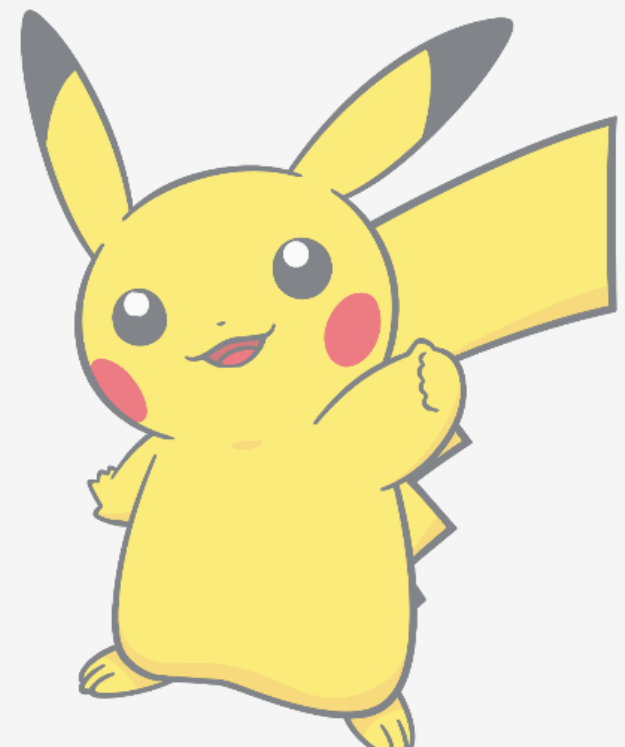
.svg – scalable vector graphics

.psd – photoshop document



# 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.com
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="M78.75,120
7   <path android:fillColor="#542400" android:pathData="M79.95,140
8   <path android:fillColor="#f9be00" android:pathData="M112.45,70
9   <path android:fillColor="#f9be00" android:pathData="M98.35,93
10  <path android:fillColor="#f9be00" android:pathData="M97.55,110
11  <path android:fillColor="#542400" android:pathData="M87.95,120
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="M92,8.109
15  <path android:fillColor="#ffe100" android:pathData="M34.7,92.4
16  <path android:fillColor="#ffe100" android:pathData="M34.7,92.4
17  <path android:fillColor="#0d131a" android:pathData="M92,8.109
18  <path android:fillColor="#ffe100" android:pathData="M16.7,146
19  <path android:fillColor="#ffe100" android:pathData="M73.55,150
20  <path android:fillColor="#b50005" android:pathData="M41.7,78.2
21  <path android:fillColor="#e50012" android:pathData="M44.95,800
22  <path android:fillColor="#f9be00" android:pathData="M17.75,110
23  <path android:fillColor="#f9be00" android:pathData="M48,98.304
24  <path android:fillColor="#f9be00" android:pathData="M22,134.8
25  <path android:fillColor="#f9be00" android:pathData="M18.4,145
```



# Resources/Drawables/Mipmap

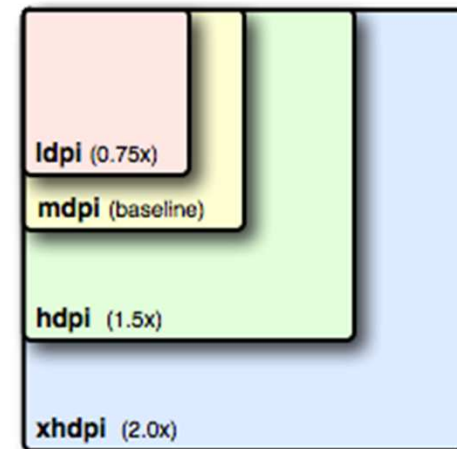
(ikona - viacero rozlíšení)

[http://developer.android.com/guide/practices/screens\\_support.html](http://developer.android.com/guide/practices/screens_support.html)



pomer l/m/h/xh/x<sup>2</sup>h/x<sup>3</sup>h-dpi 3:4:6:8:12:16 - geom.postupnosť s koef. 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<sup>2</sup> high-density (XXHDPI = ~ 480 dpi)
- 192x192 for extra<sup>3</sup> high-density (XXXHDPI = ~ 640 dpi)



$\sqrt{2}$

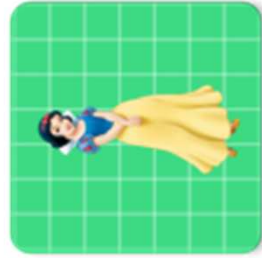
# Snehulienka

(v geometrickom rade s quociantom  $\sqrt{2}$ )

$\sqrt{2}$



192x192 for extra<sup>3</sup> high-density (XXXHDPI =  $\sim 640$  dpi)



144x144 for extra<sup>2</sup> high-density (XXHDPI =  $\sim 480$  dpi)



96x96 for extra high-density (XHDPI =  $\sim 320$  dpi)



72x72 for high-density (HDPI =  $\sim 240$  dpi)



48x48 for medium-density (MDPI =  $\sim 160$  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>
```

- color - accessibility

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

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

- dimentions

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

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

- style – množina nastavení

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

```
<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>
```

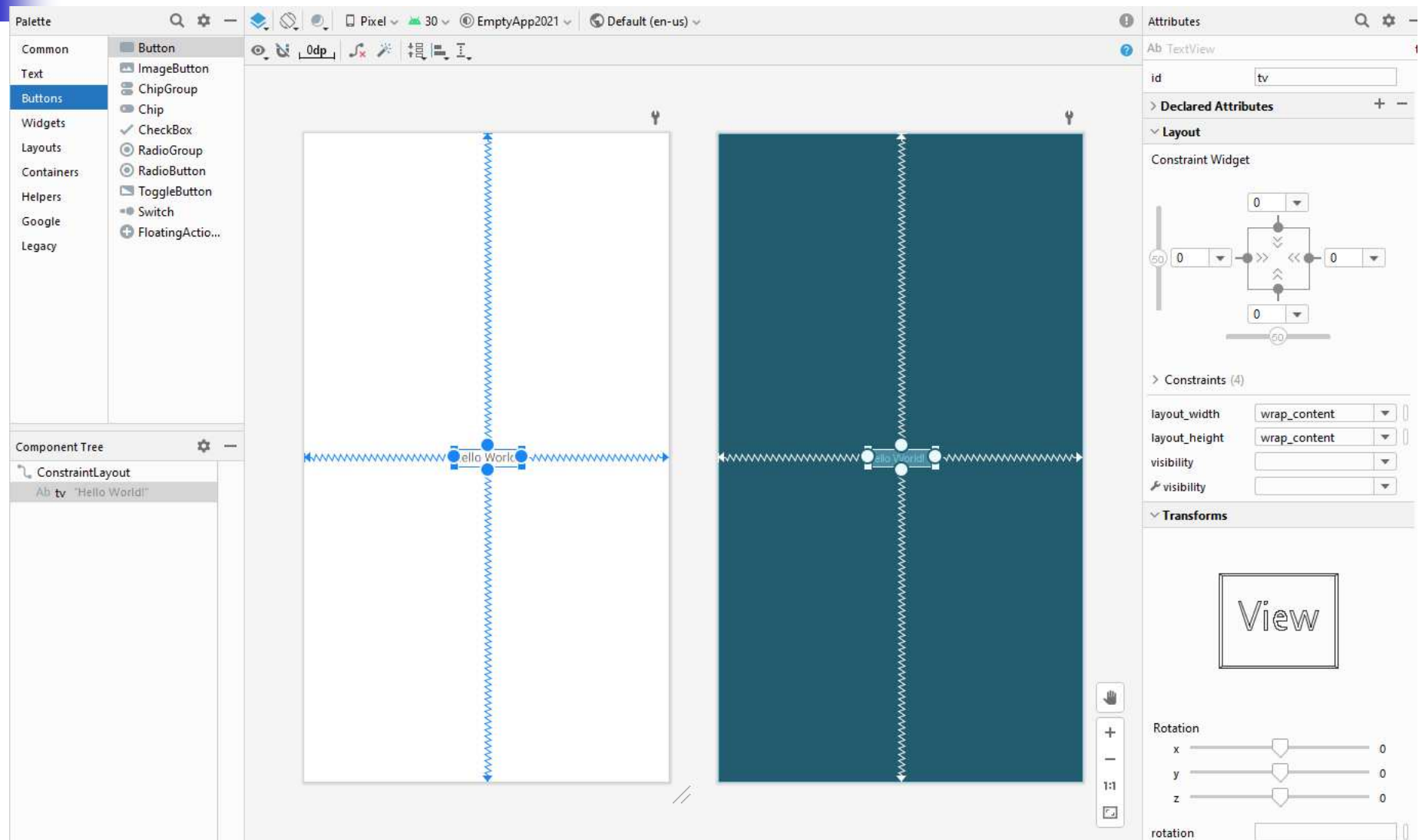
- plurals (quantity strings)

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

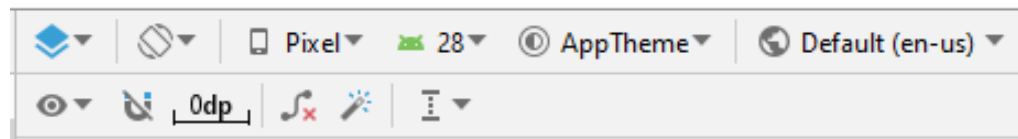
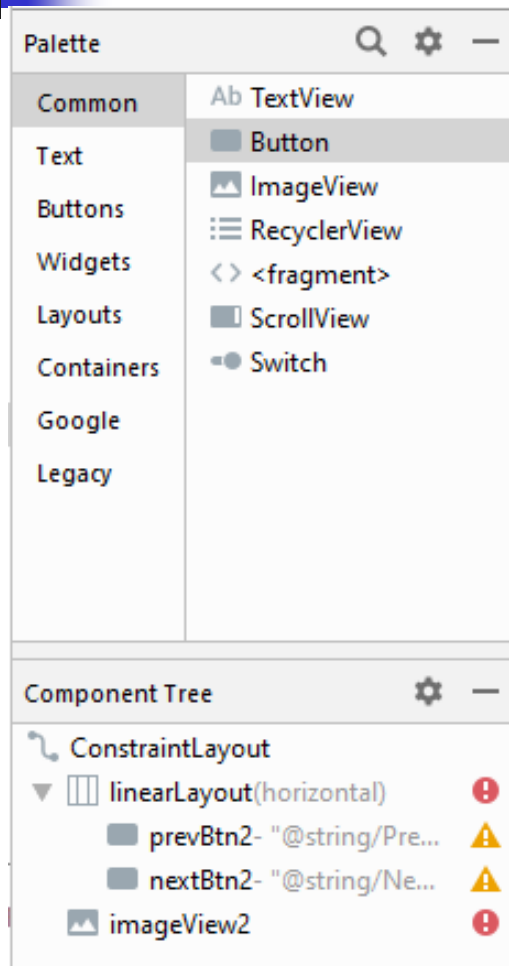
# Resources/Layout

(Design View)

Konvencia:  
`XYZActivity[.kt/]`  
má layout  
`activity_xyz.xml`



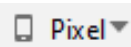
# Layout Manager



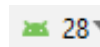
Design/Blueprint/Design+Blueprint



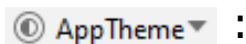
Layout: Landscape/Portrait/...



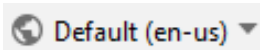
Pixel: AVD/Pixel2/Pixel#



API Level: 26/27/28/...



:



: lokalizácie do rôznych jazykov

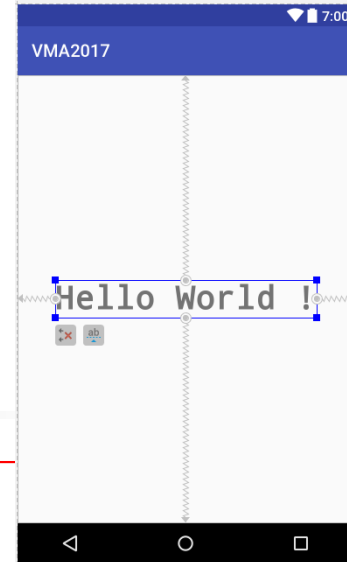


: warnings, errors

3 Warnings 2 Errors			X
Message		Source	
▶	! Missing Constraints in ConstraintLayout	linearLayout <LinearLayout>	
▶	! Missing Constraints in ConstraintLayout	imageView2 <ImageView>	
▶	! Button should be borderless	PrevBtn2 <Button>	
▶	! Button should be borderless	button2 <Button>	
▶	! Image without `contentDescription`	imageView2 <ImageView>	

# 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">
```

*wrap\_content  
fill\_parent  
match\_parent*

```
<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>
```

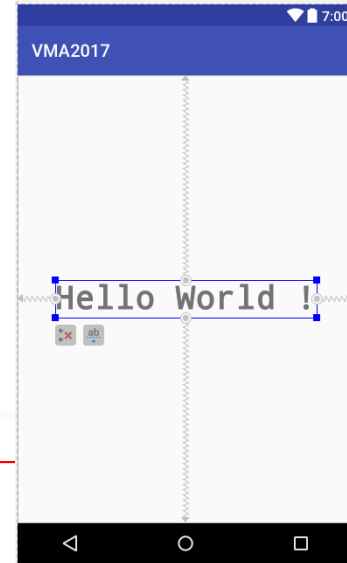
**Bad style**

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



# 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</string>
```

```
    <string name="IntroString">Hello World</string>
```

```
</resources>
```

```
<resources>
```

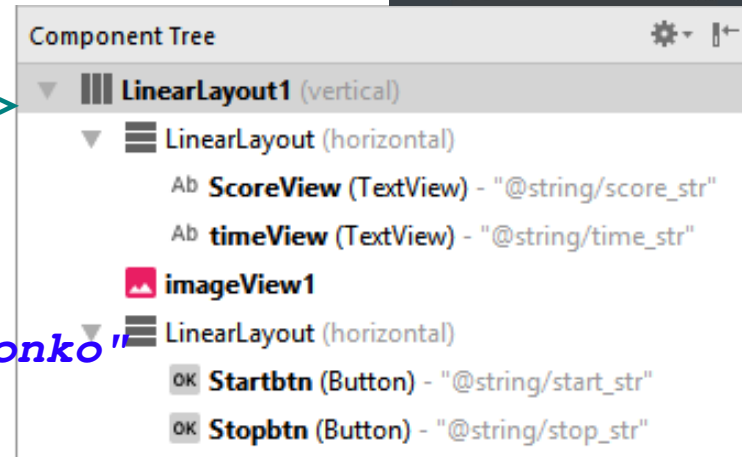
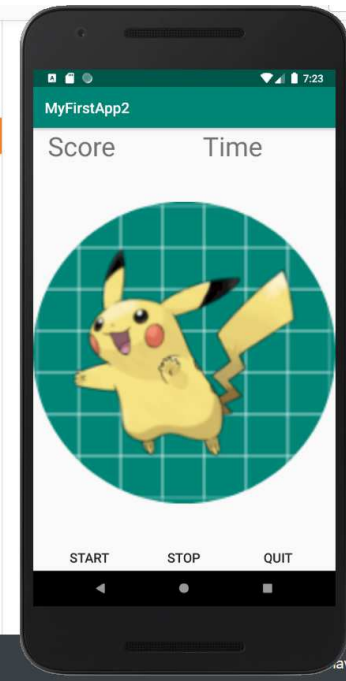
```
    <dimen name="reallyBigFont">30dp</dimen>
```

```
</resources>
```

# Ako by to malo vyzerat'

```
<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

# 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



# Väzba komponentov v kóde

```
build.gradle.kts
```

```
android {  
    buildFeatures {  
        viewBinding = true  
    }  
}
```

```
class MainActivity : AppCompatActivity() {  
    private lateinit var binding: ActivityMainBinding  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        binding = ActivityMainBinding.inflate(LayoutInflater)  
        setContentView(binding.root)  
//setContentView(R.layout.activity_main)  
val startBtn = findViewById<Button>(R.id.startBtn)  
val iv = findViewById<ImageView>(R.id.imageView)  
        val startBtn = binding.startBtn  
        val iv = binding.imageView  
    }  
}
```



# Fyzické zariadenie

## 7. Testing Android Studio Apps on a Physical Android Device

### Android Debug Bridge (ADB)

```
C:\Users\borovan>adb -s emulator-5554 emu kill
OK: killing emulator, bye bye
OK
```

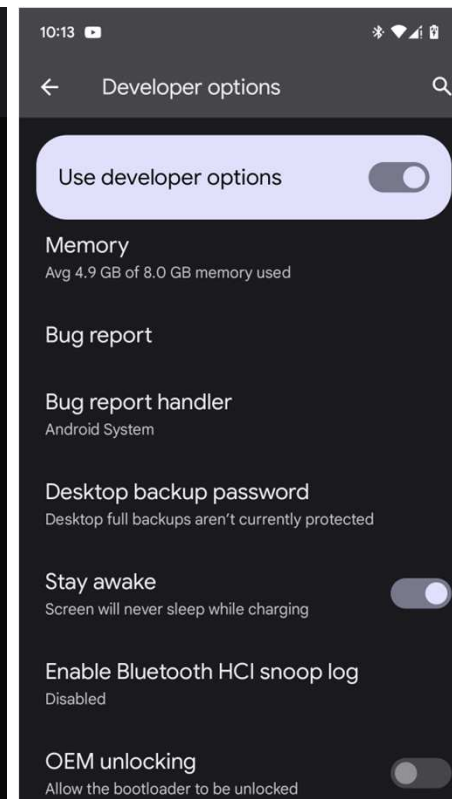
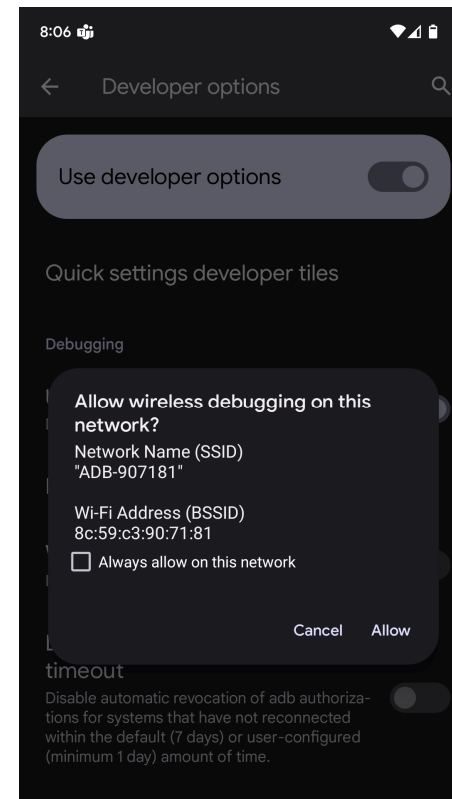
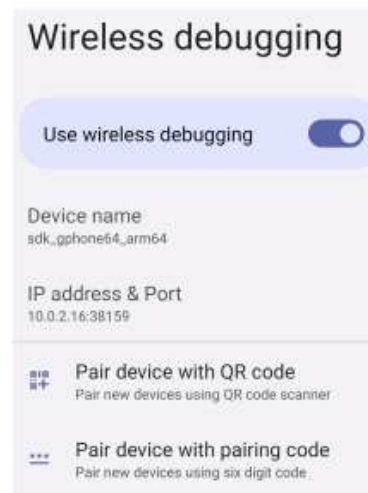
```
C:\Users\borovan>adb devices
List of devices attached
emulator-5554    device

C:\Users\borovan>adb devices
List of devices attached
XVW7N17331000103    device
```

### USB Debugging on Android device, stay awake



### Wireless debugging...





# Logovanie

---

Tri najbežnejšie spôsoby ako (logovať, debugovať):

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

```
import com.google.android.material.snackbar.Snackbar

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

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

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



# Logovanie

```
val TAG = "PIKAS"  
Log.i(TAG, "prev...")
```

Pikas13.zip

HUAWEI EVA-L19 (XVV7N17331000103) Android 7, API 24

package:mine tag:PIKAS

0 results

2023-09-26 10:43:40.786	16997-16997	PIKAS	com.example.pikas13	I	prev...
2023-09-26 10:43:43.241	16997-16997	PIKAS	com.example.pikas13	I	prev...
2023-09-26 10:45:01.558	18234-18234	PIKAS	com.example.pikas13	I	onTICK
2023-09-26 10:45:02.559	18234-18234	PIKAS	com.example.pikas13	I	onTICK
2023-09-26 10:45:02.963	18234-18234	PIKAS	com.example.pikas13	I	next...
2023-09-26 10:45:03.174	18234-18234	PIKAS	com.example.pikas13	I	next...
2023-09-26 10:45:03.380	18234-18234	PIKAS	com.example.pikas13	I	next...

HUAWEI EVA-L19 (XVV7N17331000103) Android 7, API 24

package:mine tag:CYKLUS

2023-09-26 10:49:22.941	20719-20719	CYKLUS	com.example.applifecycle13	I	onCreate
2023-09-26 10:49:22.985	20719-20719	CYKLUS	com.example.applifecycle13	I	onStart0
2023-09-26 10:49:23.012	20719-20719	CYKLUS	com.example.applifecycle13	I	onResume0
2023-09-26 10:49:38.481	20719-20719	CYKLUS	com.example.applifecycle13	I	onPause
2023-09-26 10:49:38.713	20719-20719	CYKLUS	com.example.applifecycle13	I	onStop1
2023-09-26 10:49:38.746	20719-20719	CYKLUS	com.example.applifecycle13	I	onDestroy1

AppLifeCycle13.zip

# Pikas

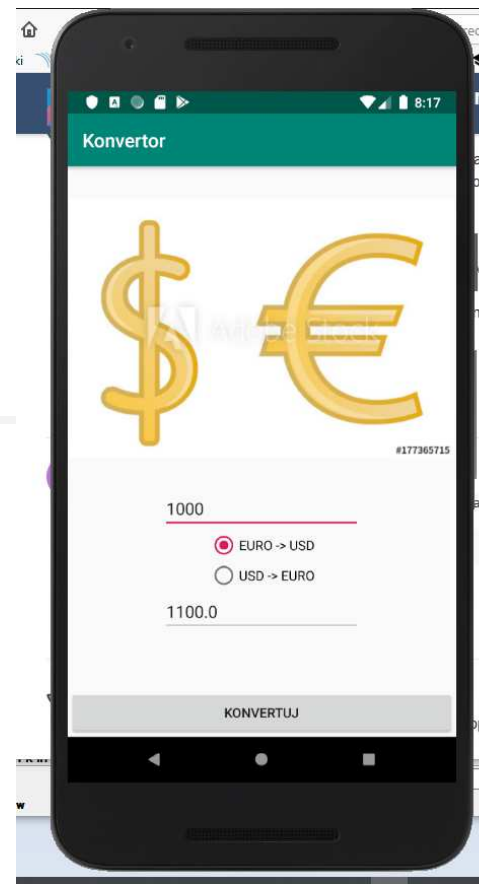
```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    var i = 0
    var imgs = arrayOf(
        ContextCompat.getDrawable(applicationContext,
                                R.drawable.butterfree),
        ..., ...,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





# Konvertor EURO USD

(logika)



Klik na Konvertuj

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 = 1.07f  
            if (eur2usd.isChecked) output = exchangeRate * output  
            if (usd2eur.isChecked) output = output / exchangeRate  
            outputText.setText("$output") // set
```

Konvertor13.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()
```

```
    binding.apply {
```

```
        if (inputText.text.isNotEmpty()) {
```

```
            val input = inputText.text.toString().toFloat();
```

```
            var output = input
```

```
            val exchangeRate = 1.07f
```

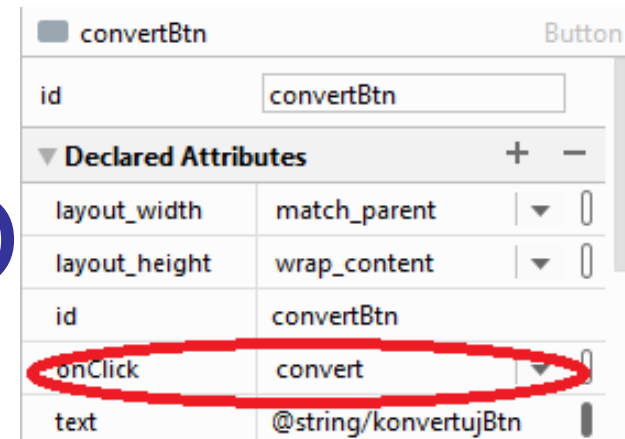
```
            if (eur2usd.isChecked) output = exchangeRate * output
```

```
            if (usd2eur.isChecked) output = output / exchangeRate
```

```
            outputText.setText("$${output.format(2)}")    } } }
```

```
fun Float.format(digits: Int) =
```

```
    java.lang.String.format("%.${digits}f", this)
```

convertBtn

Button

id

convertBtn

▼ Declared Attributes

+

-

layout\_width

match\_parent

▼

0

layout\_height

wrap\_content

▼

0

id

convertBtn

onClick

convert

▼

0

text

@string/konvertujBtn

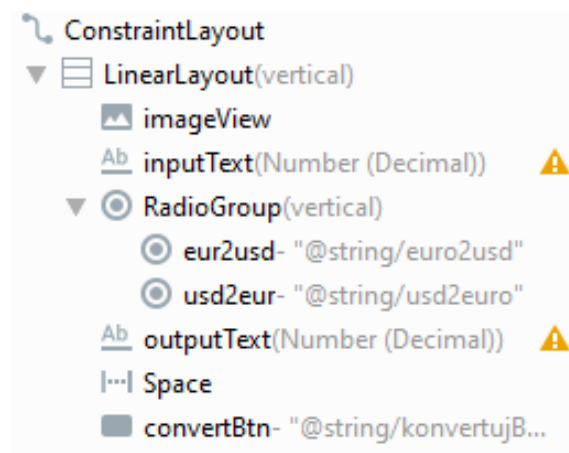
extension  
metóda  
Float



# Konvertor EURO USD

(layout)

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



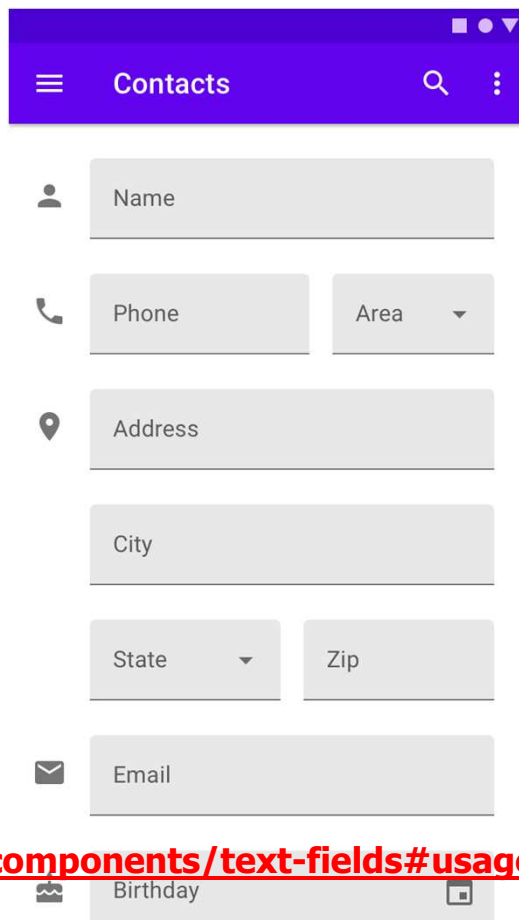
# Text Fields

## 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



Contacts

Name

Phone Area

Address

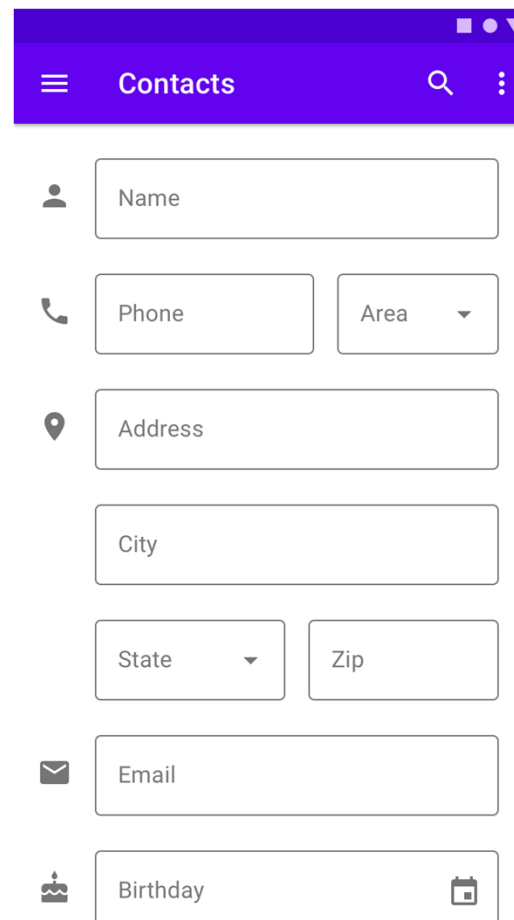
City

State Zip

Email

Birthday

The image shows a mobile app interface for 'Contacts' using the Material Design text field style. The fields are light gray with a subtle shadow and a thin border. The 'Phone' field is split into 'Phone' and 'Area' with a dropdown arrow. The 'State' field is also a dropdown. The 'Birthday' field has a calendar icon on the right.



Contacts

Name

Phone Area

Address

City

State Zip

Email

Birthday

The image shows the same mobile app interface but with a flat text field style. The fields are white with a thin gray border. The 'Phone' and 'State' fields still have their respective dropdown arrows and the 'Birthday' field has its calendar icon.

# 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?.isEmpty()?:false &&
            userTV.text?.isEmpty()?:false &&
            passwordTV.text?.isEmpty()?:false
        button.isEnabled =
            if (emailTV.text != null && userTV.text != null &&
                passwordTV.text != null)
                emailTV.text!!.isEmpty() &&
                userTV.text!!.isEmpty() &&
                passwordTV.text!!.isEmpty()
            else
                false
    }
}

emailTV.addTextChangedListener(textWatcher)
userTV.addTextChangedListener(textWatcher)
passwordTV.addTextChangedListener(textWatcher)
```

# Príklad jednoduchéj 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"`

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** (zjednodušená 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()
```

```
    ...
```

```
})
```



# 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])  
    })  
}
```

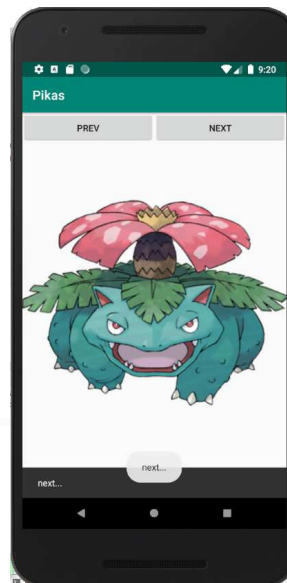
View(s)

logovanie



# Pikas

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



```
const final 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])
}
```

State

## ▼ Common Attributes

style	@style/mystyle	▼
onClick	clickOnNext	▼

# 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()
}
```

Pikas

Cas: 15

PREV

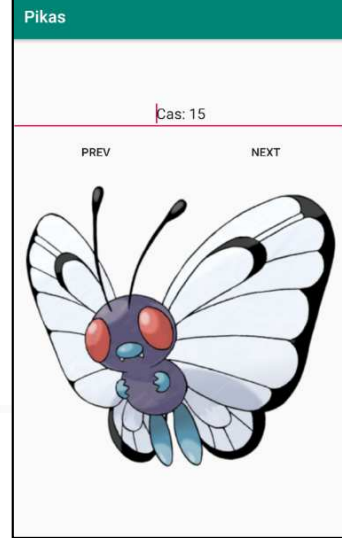
NEXT



# Pikas

(asynchrónnosť – count down)

pomocou `android.os.CountDownTimer`



```
object:CountDownTimer(20000, 1000) { // 20sek, tik po 1sek  
    // how long, period
```

tik

```
    override fun onTick(millisUntilFinished: Long) {  
        Log.d(TAG, "onTICK")  
        runOnUiThread {  
            time.setText("Cas: ${millisUntilFinished/1000}") }  
        }  
    }
```

game  
over

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

```
}.start()
```

ukončenie appky

# 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()`

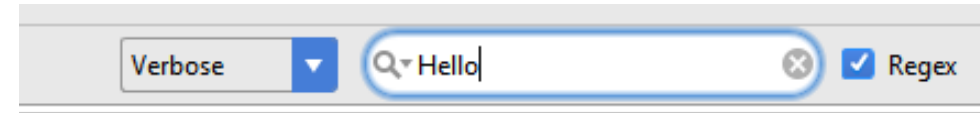
- do každej metódy dáme kontrolný výpis, aby sme pochopili životný cyklus

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

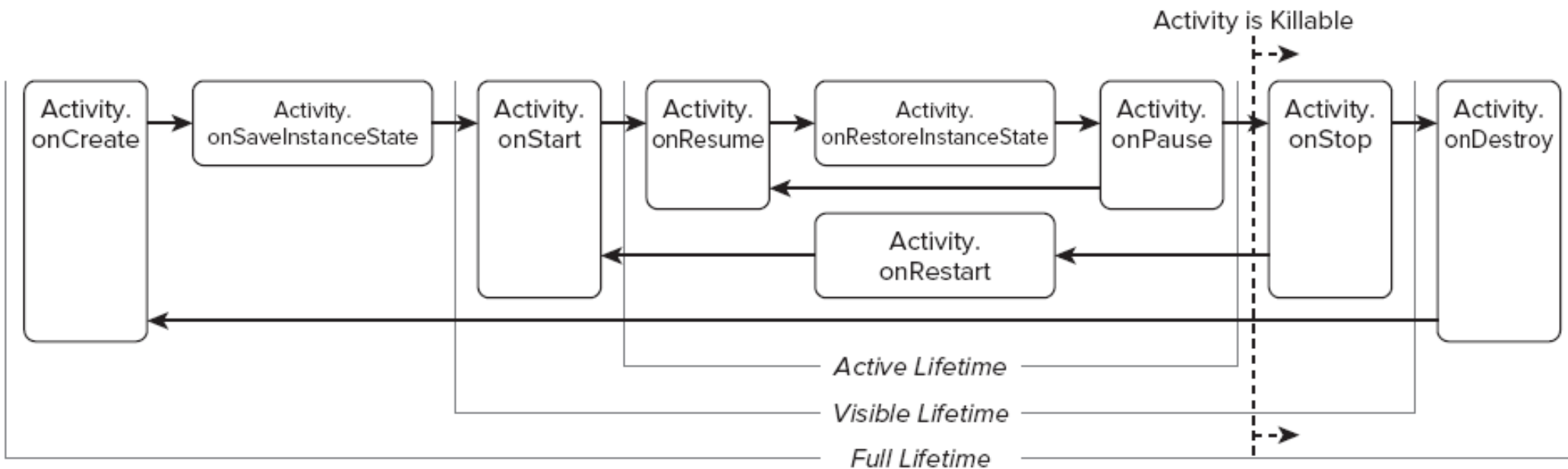
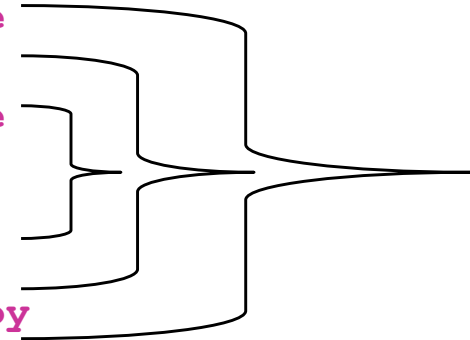
tag vhodný na filtrovanie

# LogCat

(Filtrovane 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





# 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**



# 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)  
    ...  
}
```



# SharedPreferences

SharedPreferences má metody 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()
    }
}
```

# Pikas.java

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

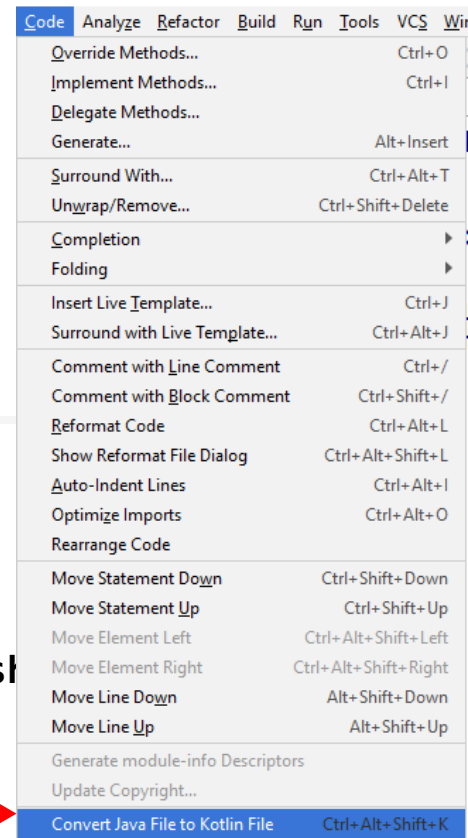
```
i = 0
iv.setImageDrawable(images[i])

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



# Čo je Kotlin ?

Kotlin is the New Official Language of Android



Android

+



Kotlin



Kotlin Island

3

