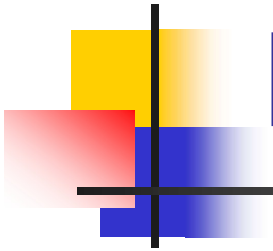
# AS Projekt
## (anatómia projektu)

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 vzorov/templates
  - prémia: Piškvorky 3x3, a ďalšie

# Čo dostaneme zadarmo

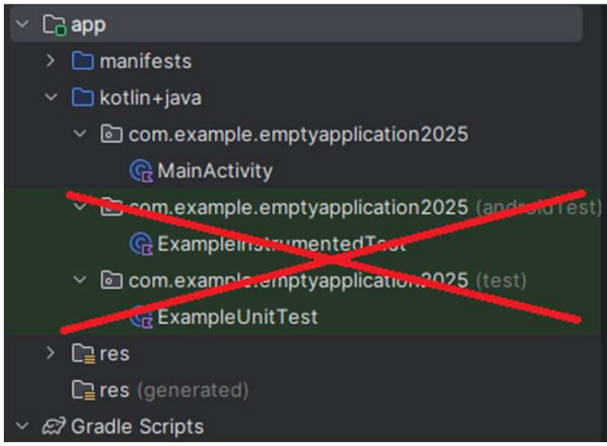
## (pokračujeme v minulej prednáške)

Chapter 6

6. A Tour of the Android Studio User Interface

```
package com.example.emptyapplication2025

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)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }
        // 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




EmptyApplication2025.zip


# AndroidManifest.xml

## (automaticky vygenerovaný súbor aplikácie)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <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

EmptyApplication2025.zip

Chapter 10

10. The Anatomy of an Android App


# 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 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

Chapter 10

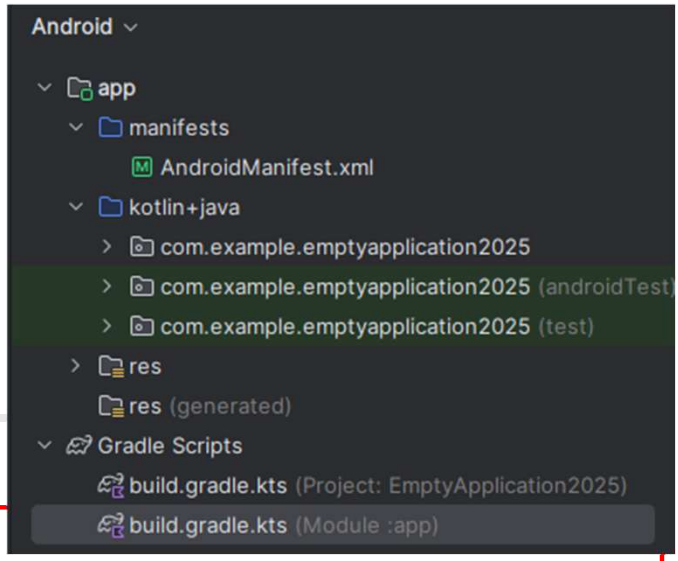
10. The Anatomy of an Android App

# 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 (podtriedy Views), hlavnou výhodou je znovu-použiteľnosť fragmentu v rôznych aktivitách. Vzťah Aktivita vs. Fragment je teda many-to-many
- **Intent** – mechanizmus ako jedna aktivita vie spustiť/zavolať inú. Intent môže obsahovať dáta. 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, SQLLite
- **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)



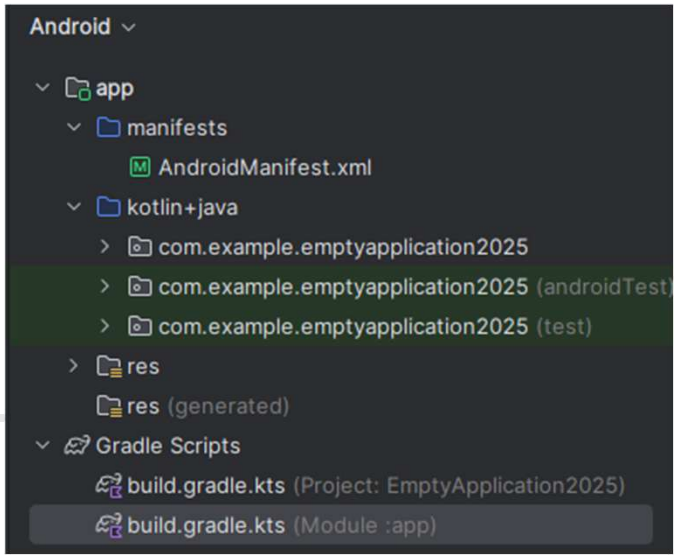

Gradle je build tool, podobne ako make, maven

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
}
android {
    namespace = "com.example.emptyapplication2025"
    compileSdk = 36
    defaultConfig {
        applicationId = "com.example.emptyapplication2025"
        minSdk = 24
        targetSdk = 36
        versionCode = 1
}
dependencies {
    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.appcompat)
    implementation(libs.material)
    implementation(libs.androidx.activity)
    implementation(libs.androidx.constraintlayout)
}
```

EmptyApplication2025.zip

# build.gradle

## (konfiguračný súbor pre gradle)




Gradle je build tool, podobne ako make, maven

...

Gradle súbory sú dva - väčšinou nás zaujíma „Module:app"

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

**OLD**

```
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
```

**NEW**

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
}
android {
    namespace =
       "com.example.emptyapplication2025"
    compileSdk = 36
    defaultConfig {
        applicationId =
        "com.example.emptyapplication2025"
        minSdk = 24
        targetSdk = 36
        versionCode = 1
        versionName = "1.0
```

EmptyApplication2025.zip

# MergedManifest

## (spája AndroidManifest a build.gradle)

```
<manifest
    android:versionCode="1"
    android:versionName="1.0"
    package="com.example.emptyapplication2025"
    xmlns:android="http://schemas.android.com/apk/res/android" >
    <uses-sdk
        android:minSdkVersion="24"
        android:targetSdkVersion="36.0" />
    <permission
        android:name="com.example.emptyapplication2025.DYNAMIC_RECEIVE
        android:protectionLevel="signature" />
    <uses-permission
        android:name="com.example.emptyapplication2025.DYNAMIC_RECEIVE
    <application
        android:allowBackup="true"
        android:appComponentFactory="androidx.core.app.CoreComponentFa
        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.EmptyApplication2025" >
        <activity
            android:exported="true"
            android:name="com.example.emptyapplication2025.MainActivity"
            <intent-filter
                <action
                    android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
```

**Manifest Sources**

- EmptyApplication2025.app main manifest (this file)
- core:1.10.1 manifest
- build.gradle.kts injection

**Other Manifest Files**

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

activity:1.8.0 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:1.10.1 manifest
core-runtime:2.2.0 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:2.6.1 manifest
lifecycle-livedata:2.6.1 manifest
lifecycle-runtime:2.6.1 manifest
lifecycle-viewmodel-savedstate:2.6.1 manifest
lifecycle-viewmodel:2.6.1 manifest
loader:1.0.0 manifest
localbroadcastmanager:1.0.0 manifest
material:1.10.0 manifest
print:1.0.0 manifest
recyclerview:1.1.0 manifest
savedstate:1.2.1 manifest
transition:1.2.0 manifest
vectordrawable-animated:1.1.0 manifest
vectordrawable:1.1.0 manifest
versionedparcelable:1.1.1 manifest
viewpager2:1.0.0 manifest
viewpager:1.0.0 manifest

EmptyApplication2025.zip

referencia meno apky


# 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">EmptyApplication2025</string>
</resources>
```

EmptyApplication2025.zip

# Buď kreatívny

(aspoň pri ic_launcher ikone)

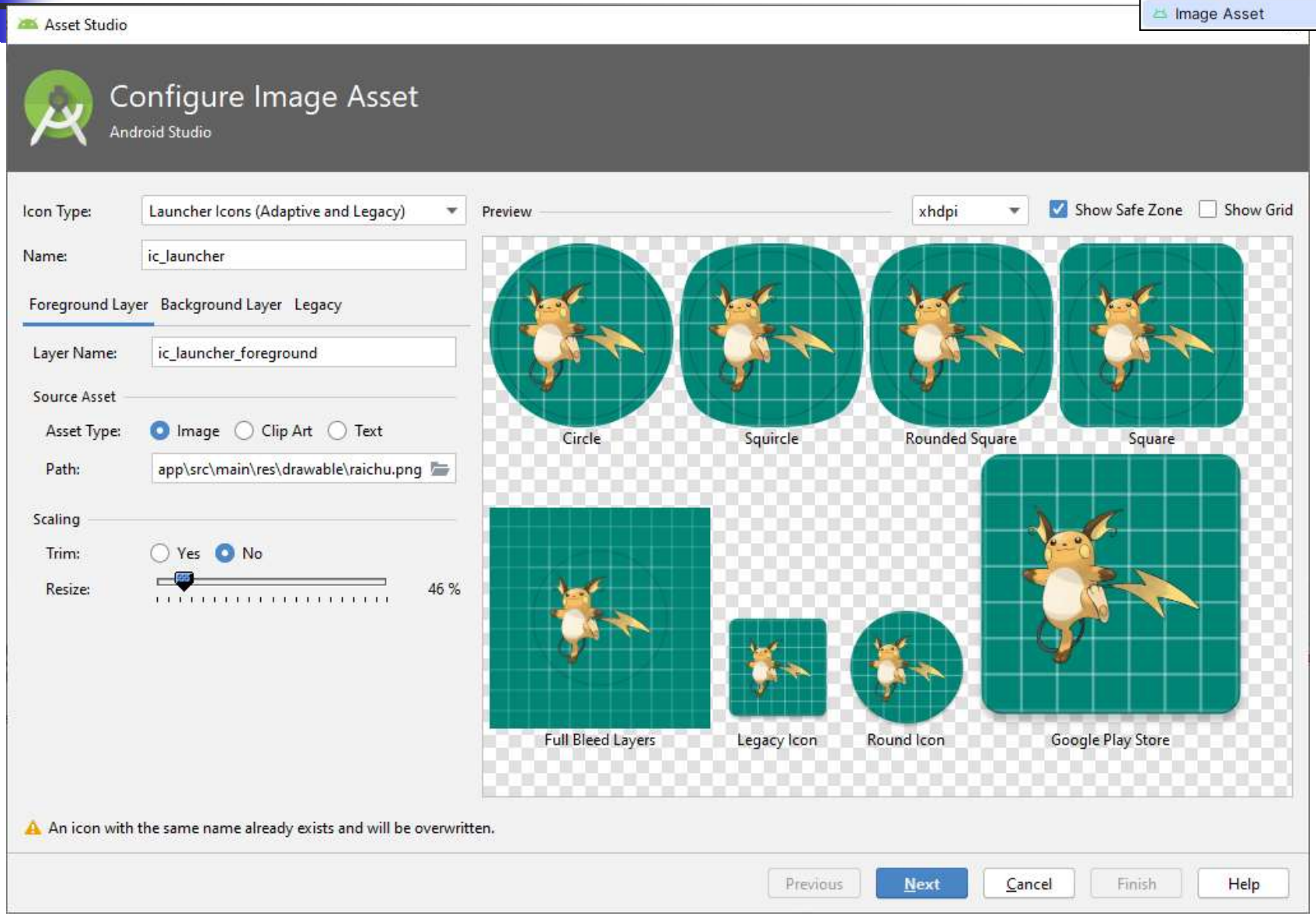
Je hrozné pri opravovaní mať v tablete/mobile viacero študentských riešení s generickými/neosobnými ikonami. Preto ak sa dá, tak sa zosobnite v posielanom riešení už v ikone vašej aplikácie.



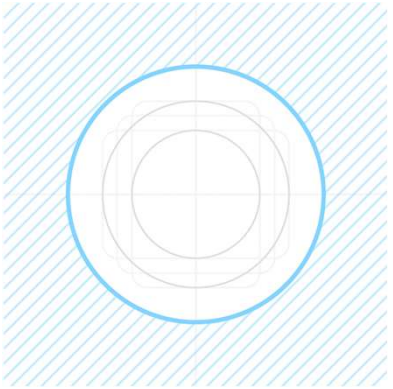


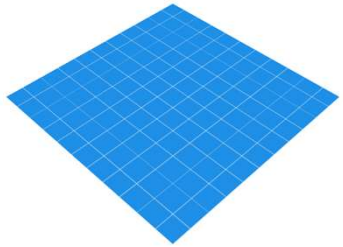

# Buď kreatívny

## (a použi Asset Studio - New/ImageAsset)

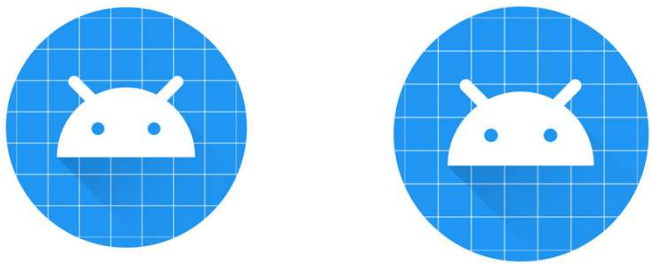


# Adaptive icon

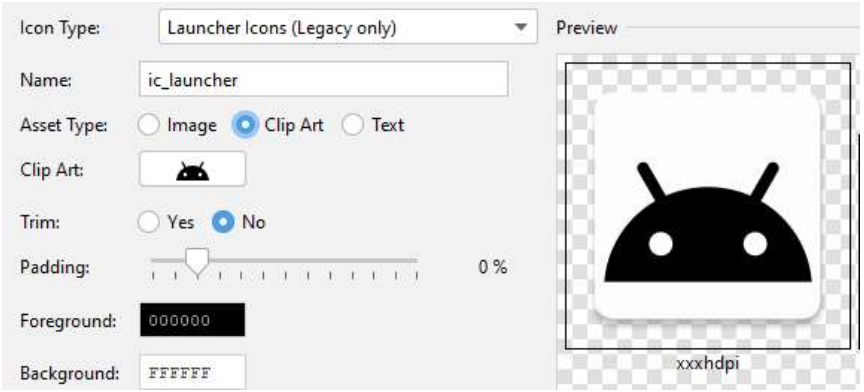




- funguje od Android-Oreo, API 26 – Android
- umožňuje zariadeniu vhodne škálovať ikonu podľa
  - zvoleného rozlíšenia 108dp, 66dp, ...
  - zvoleného orámovania
- adaptívna ikona má pozadie a popredie
- 
```
<adaptive-icon
xmlns:android="http://schemas.android.com/apk/res/android">
  <background android:drawable="@drawable/ic_launcher_background"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

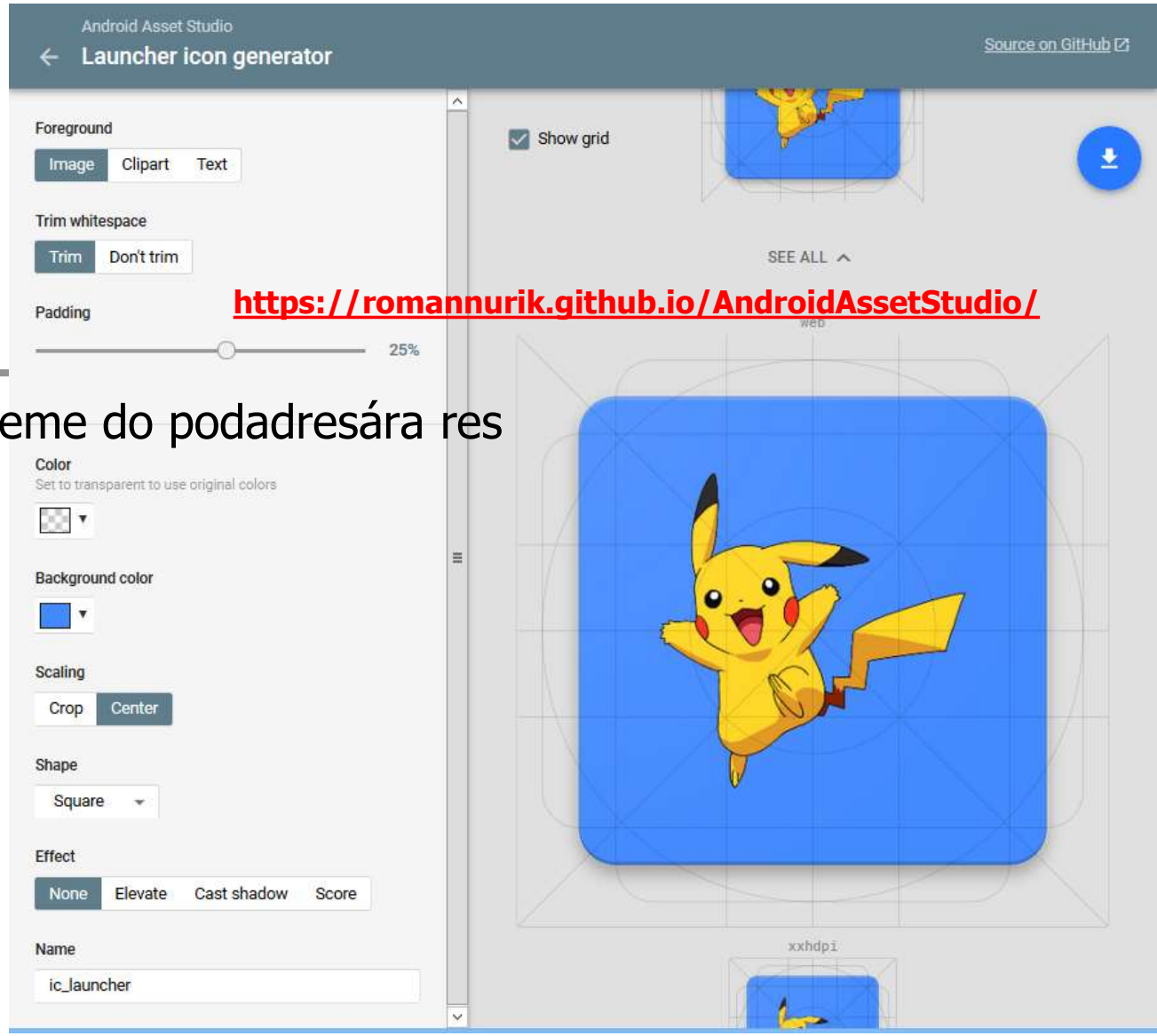
## Icon generator

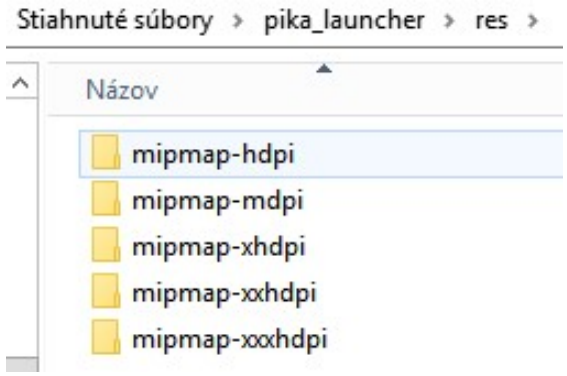
https://romannurik.github.io/AndroidAssetStudio/

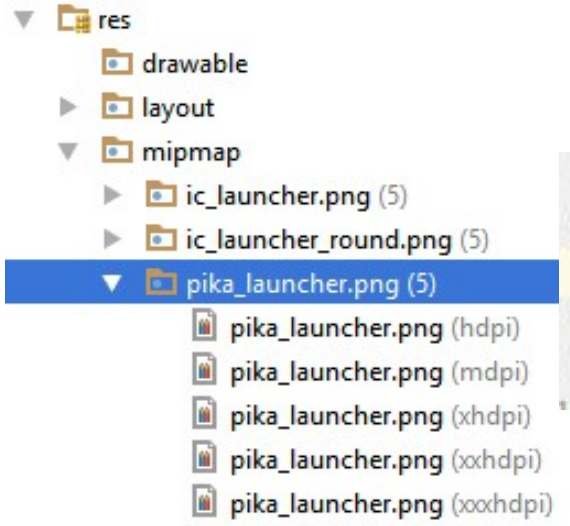
výsledok priamo nakopírujeme do podadresára res

Ikony/obrázky sa

sa objavia v projekte



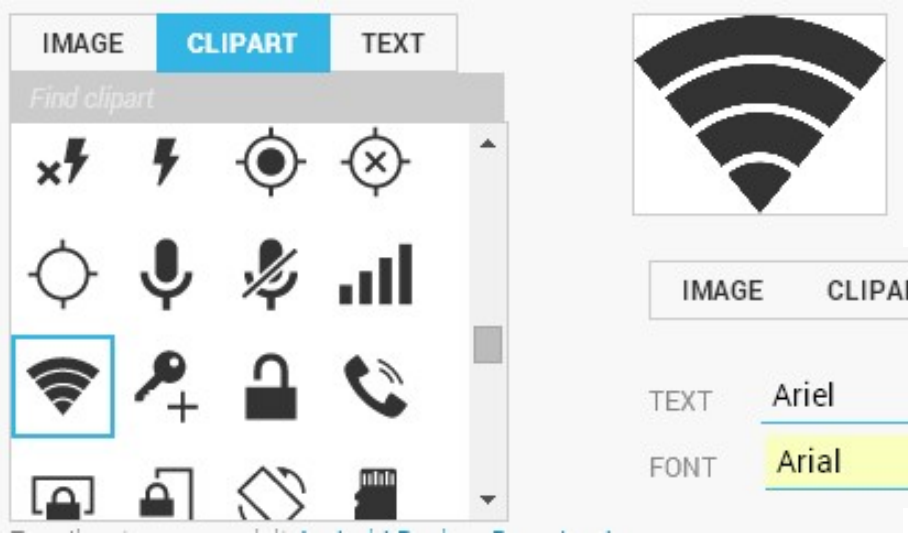

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/pika_"
    android:label @mipmap/pika_launcher
    android:roundIcon="@mipmap/ic_launcher_round"
```

EmptyApplication2025.zip

# Android Asset Studio

(jedna z alternatív)

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



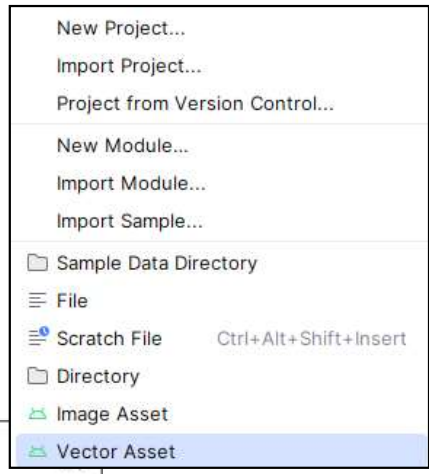


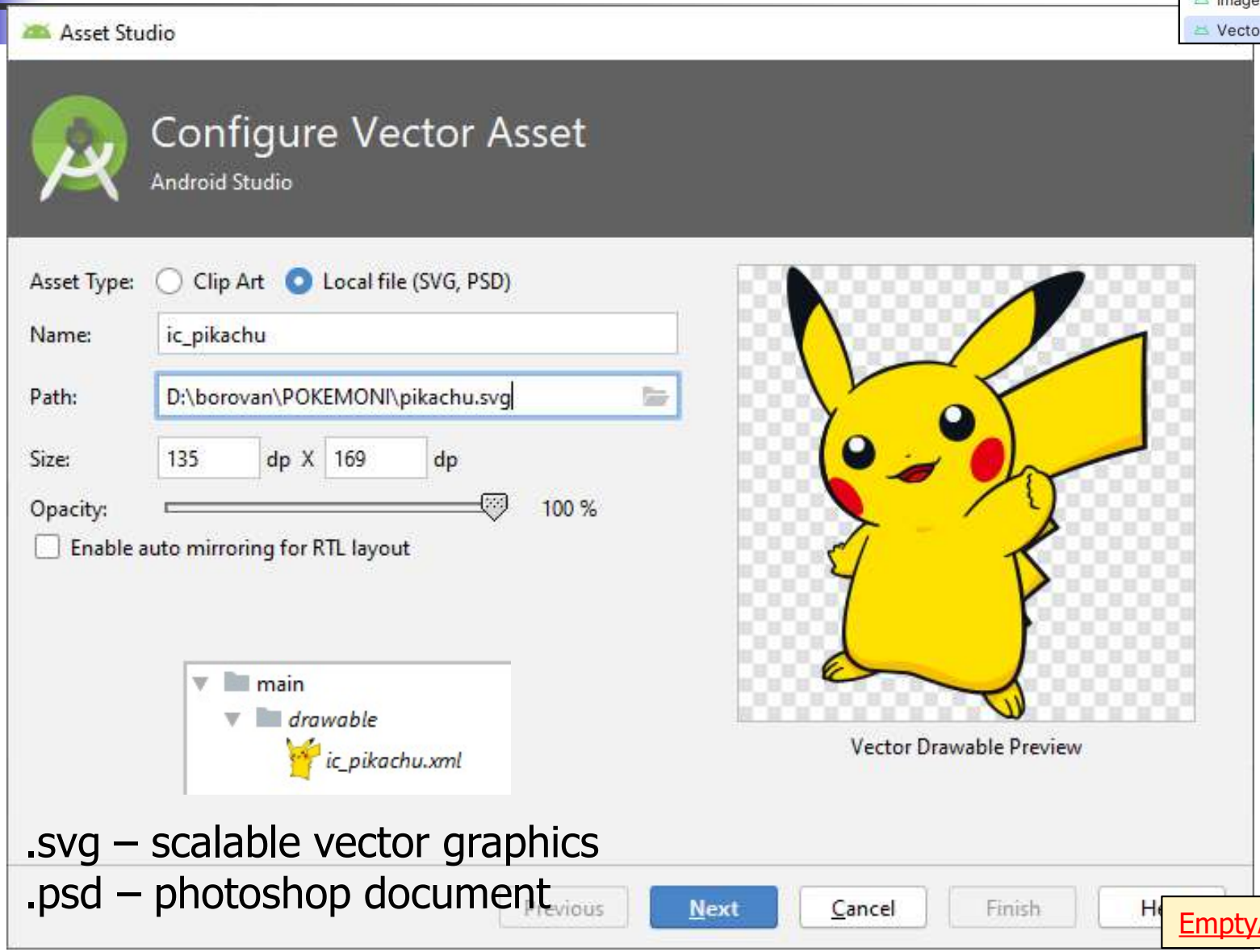



- .png,. jpg, .bmp, ...
- cliparty
- texty

# Pre .svg a .psd

(a použi Vector Asset Studio - New/VectorAsset)



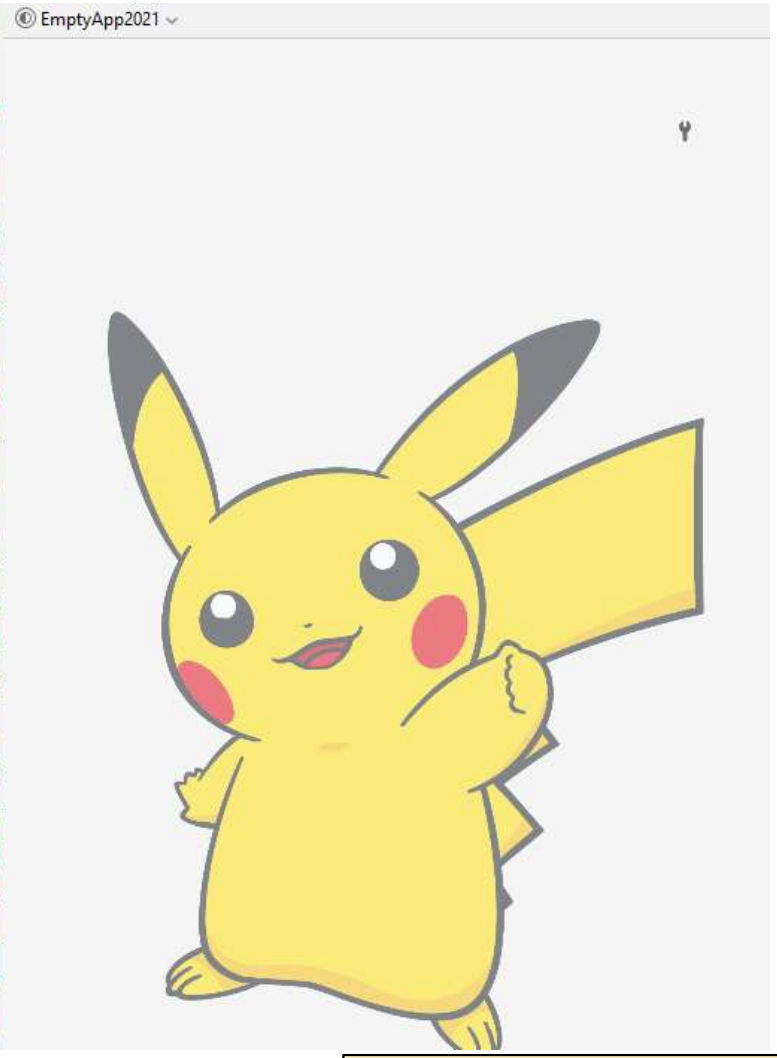

.svg – scalable vector graphics

.psd – photoshop document

EmptyApplication2025.zip

# Vektorový pikachu

```
<vector android:alpha="0.5" android:height="169dp"
    android:viewportHeight="169.1" android:viewportWidth="134.7"
    android:width="135dp" xmlns:android="http://schemas.android.c
    <path android:fillColor="#763a00" android:pathData="M79.6,140
    <path android:fillColor="#ffe100" android:pathData="M133.5,45
    <path android:fillColor="#763a00" android:pathData="M78.75,12
    <path android:fillColor="#542400" android:pathData="M79.95,14
    <path android:fillColor="#f9be00" android:pathData="M112.45,7
    <path android:fillColor="#f9be00" android:pathData="M98.35,93
    <path android:fillColor="#f9be00" android:pathData="M97.55,11
    <path android:fillColor="#542400" android:pathData="M87.95,12
    <path android:fillColor="#0d131a" android:pathData="M134.6,24
    <path android:fillColor="#0d131a" android:pathData="M13.25,12
    <path android:fillColor="#ffe100" android:pathData="M92,8.1Q93
    <path android:fillColor="#ffe100" android:pathData="M34.7,92.
    <path android:fillColor="#ffe100" android:pathData="M34.7,92.
    <path android:fillColor="#0d131a" android:pathData="M92,8.1Q9
    <path android:fillColor="#ffe100" android:pathData="M16.7,146
    <path android:fillColor="#ffe100" android:pathData="M73.55,15
    <path android:fillColor="#b50005" android:pathData="M41.7,78.
    <path android:fillColor="#e50012" android:pathData="M44.95,80
    <path android:fillColor="#f9be00" android:pathData="M17.75,11
    <path android:fillColor="#f9be00" android:pathData="M48,98.3Q
    <path android:fillColor="#f9be00" android:pathData="M22,134.8
    <path android:fillColor="#f9be00" android:pathData="M18.4,145
```




EmptyApplication2025.zip

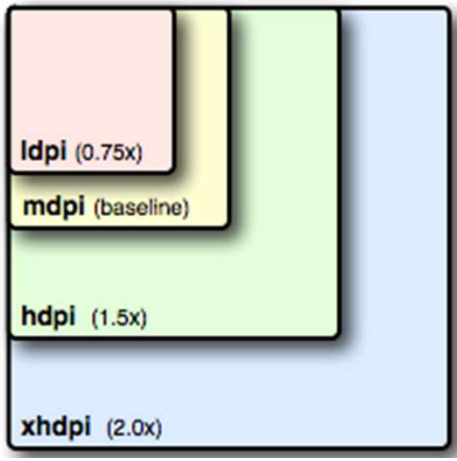
# Resources/Drawables/Mipmap

(ikona - viacero rozlíšení)

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


pomer l/m/h/xh/x²h/x³h-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 extra² high-density (XXHDPI = ~ 480 dpi)
- 192x192 for extra³ high-density (XXXHDPI = ~ 640 dpi)



# Snehulienka

## (v geometrickom rade s quocientom sqrt(2))

$\sqrt{2}$

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

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)

# 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>
```

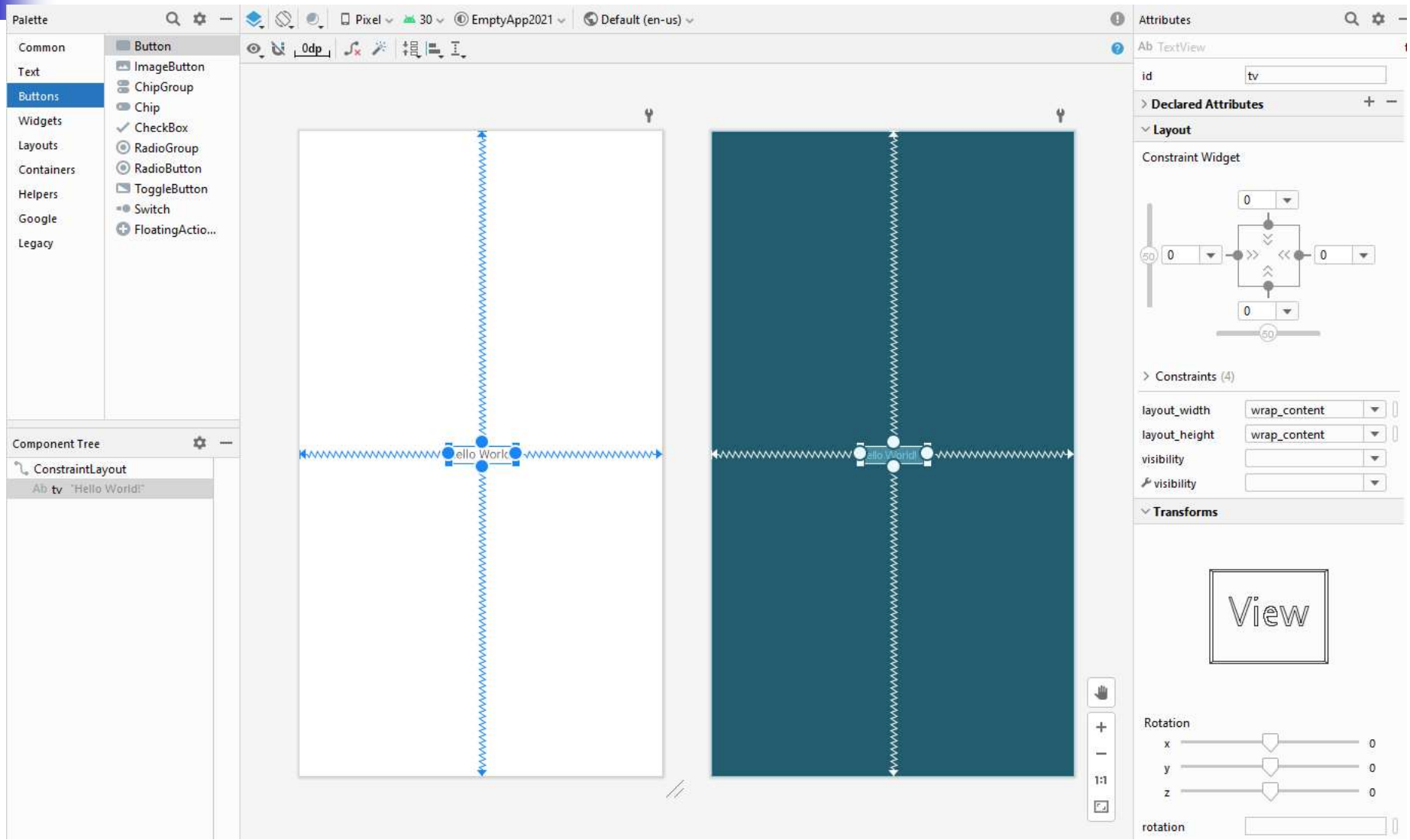
- plurals (quantity strings)

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

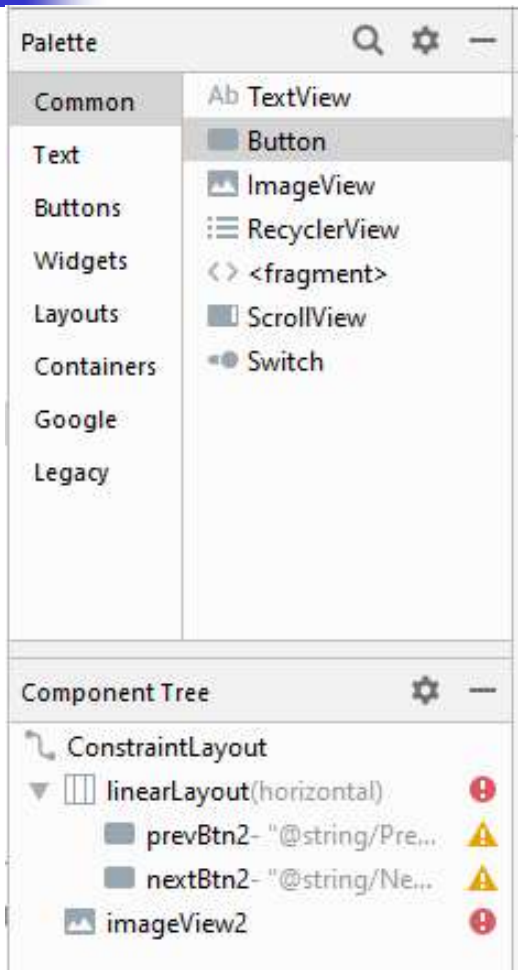
# Resources/Layout

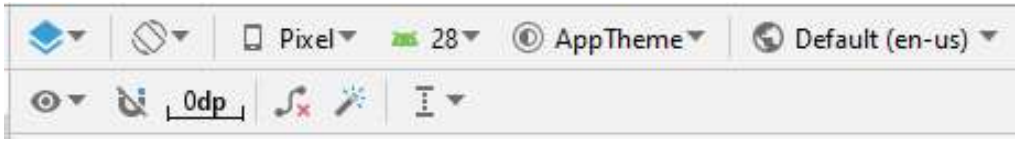
## (Design View)

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



# Layout Manager




Design/Blueprint/Design+Blueprint

Layout: Landscape/Portrait/…

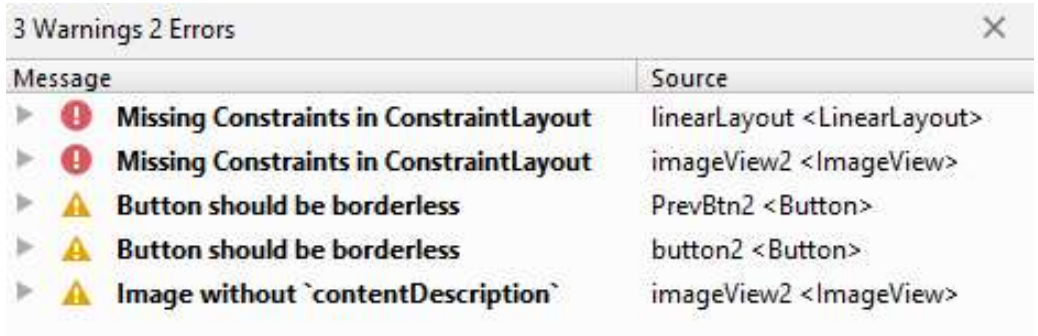
Pixel: AVD/Pixel2/Pixel#

API Level: 26/27/28/…

AppTheme :

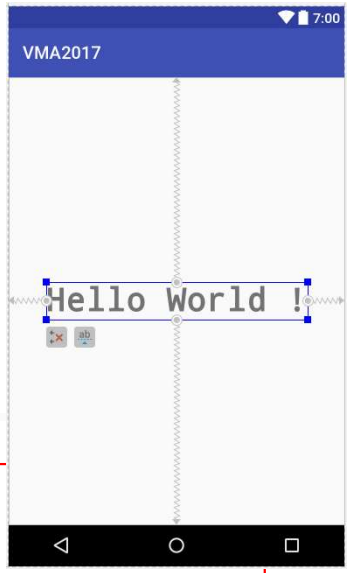
Default (en-us) : lokalizácie do rôznych jazykov

: warnings, errors



# Resources/Layout

## (Text View)




*wrap_content*
*fill_parent=*
*match_parent*

```
<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>
```

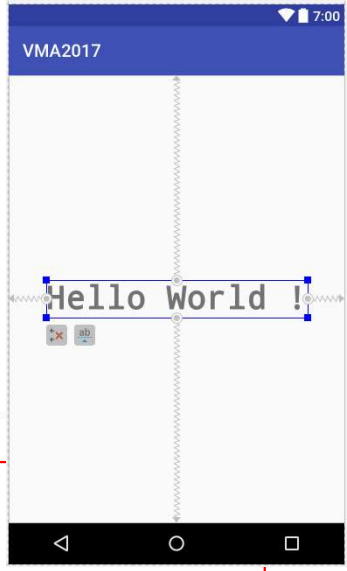
ConstraintLayout
Ab res "Hello World 1"

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

**Bad style**

# Resources/Layout

## (Text View)




*wrap_content*
*fill_parent=*
*match_parent*

```
<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>
```

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

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

# 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.i(…)
  - 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"`


Pikas13.zip

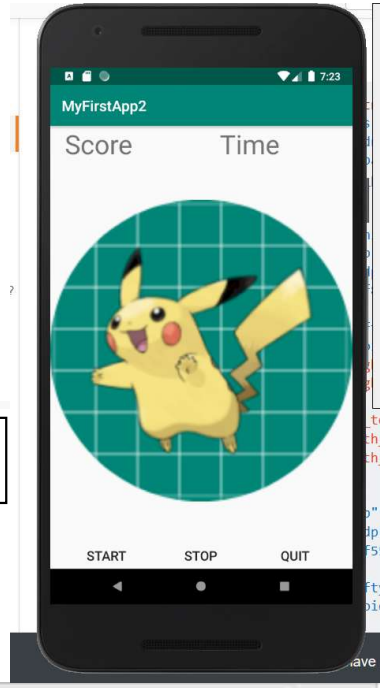
Nestihli sme:

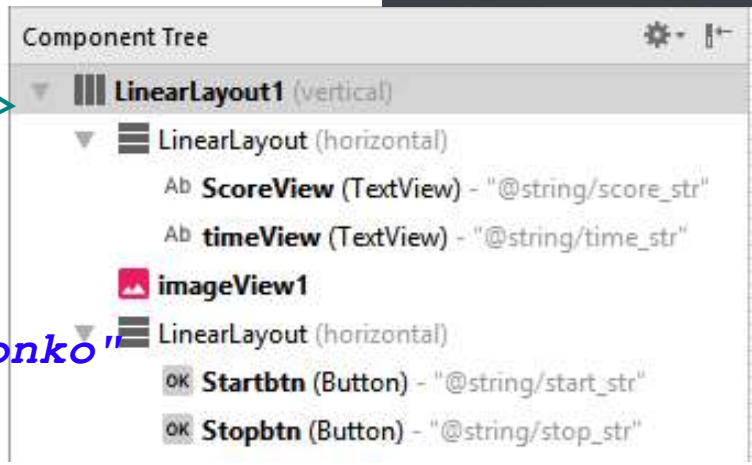
- aktivitu a jej životný cyklus

# 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**

MyFirstApp13.zip

# Väzba komponentov v kóde

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

- ~~plugin kotlin-android-extensions~~

Kotlin Android Extensions is deprecated-means that using Kotlin Synthetics for view binding is no longer supported. If your app uses Kotlin synthetics for view binding, use this guide to migrate to view binding.

```
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

## viewBindings

Konvencia: XyzActivity[.kt]
- má layout activity_xyz.xml
- **binding: ActivityXyzBinding**

*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
```

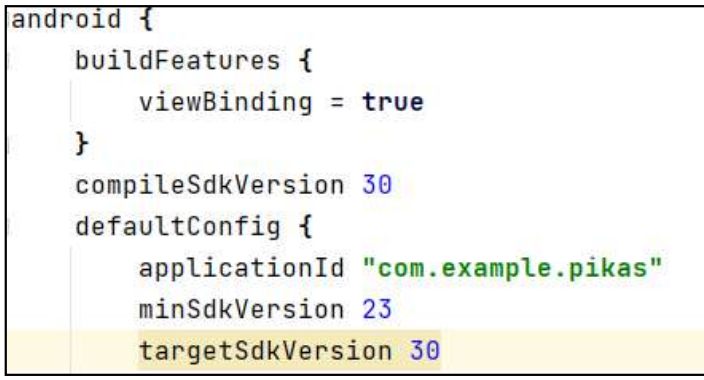
MyFirstApp13.zip

# View Binding

- findViewById() as Button, findViewById<Button>() - klasické, „javish" riešenie
- syntetic – kotlin-android-extensions plugin – deprecated od 2020
- ďalší spôsob prepojenia komponentov (View) z .xml layoutu s kódom
- pozor: nepliesť si to s Data Binding, to príde s JetPack library, to je zložitejšie

1) do build.gradle pridajte pod

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




2) v samotnej Activity NAHRADÍTE

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

za

```
val binding = ActivityMainBinding.inflate(layoutInflater)
setContentView(binding.root)
```

3) miesto referencie nejakého View, napr. `imageView2`, použijete `binding.imageView2`

4) ak mimo metódy `onCreateView` potrebujete premennú `binding`, urobte ju `lateinit var`

```
lateinit var binding : ActivityMainBinding
```

5) ak sa vaša aktivita nevolá MainA..., tak nahraďte zelené za jej meno

6) objavte, čo je `apply`, resp. iné scoping functions

Pikas13.zip

# View Binding

## príklad apply

```
binding.imageView2.setImageDrawable(imgs[i])
binding.prevBtn2.setOnClickListener {
    Toast.makeText(this,
                   "prev...",
                   Toast.LENGTH_SHORT
    ).show()
    if (--i < 0) i += imgs.size
    binding.imageView2.setImageDrawable(imgs[i])
}
```

```
binding.apply {
     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])
     }
}
```

Pikas13.zip

# Fyzické zariadenie



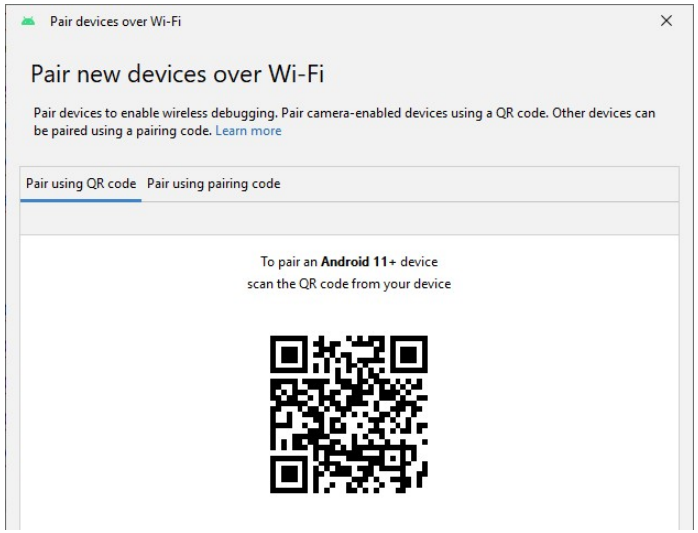
- Android Debug Bridge (ADB)

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

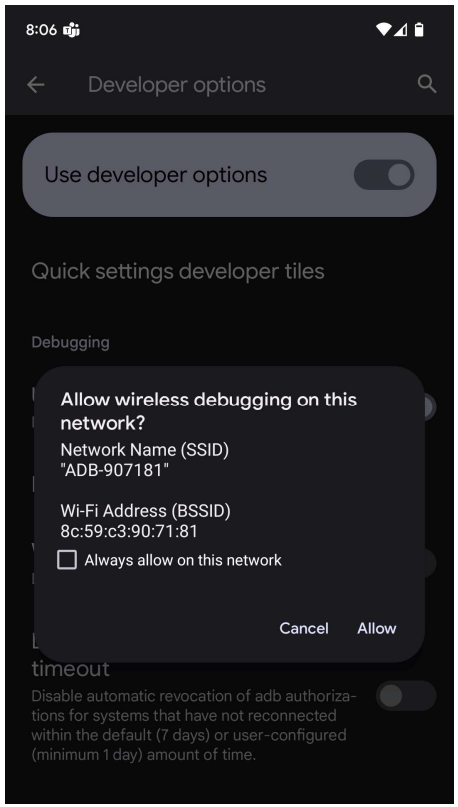
```
C:\Users\borovan>adb devices
List of devices attached
emulator-5554   device


C:\Users\borovan>adb devices
List of devices attached
XVV7N17331000103        device
```

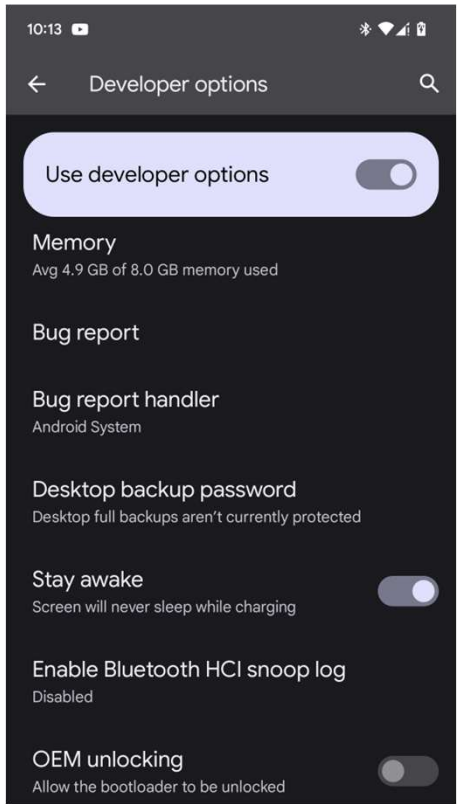
- USB Debugging on Android device, stay awake


- Wireless debugging…







# Logovanie

Tri najbežnejšie spôsoby ako (logovať, debugovať):

- trieda Log, metóda Log.i - loguje do okna Logcat, filtrujte podľa `TAGu` metódy
  - definujte si `TAG` ako konštantu
- trieda Toast, metóda Toast.make - potrebuje `Context` (zjednodušene aktivita, v ktorej sa toastuje)
  - nezabudnite na volanie .show()
- trieda Snackbar, metóda Toast.make – pridať import

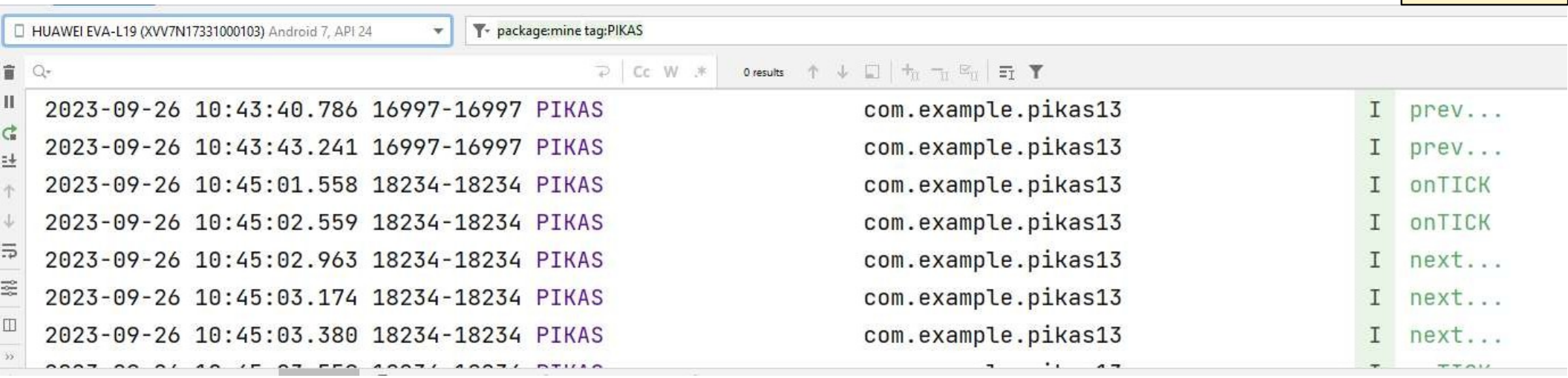
```
import com.google.android.material.snackbar.Snackbar
prevBtn2.setOnClickListener {
    Toast.makeText(this@MainActivity, "prev...", Toast.LENGTH_SHORT)
        .show()

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

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

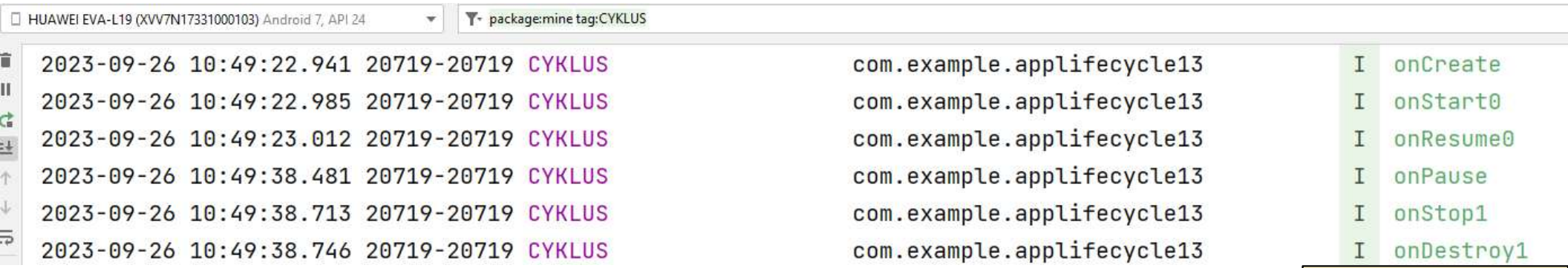
Pikas13.zip

# Logovanie

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

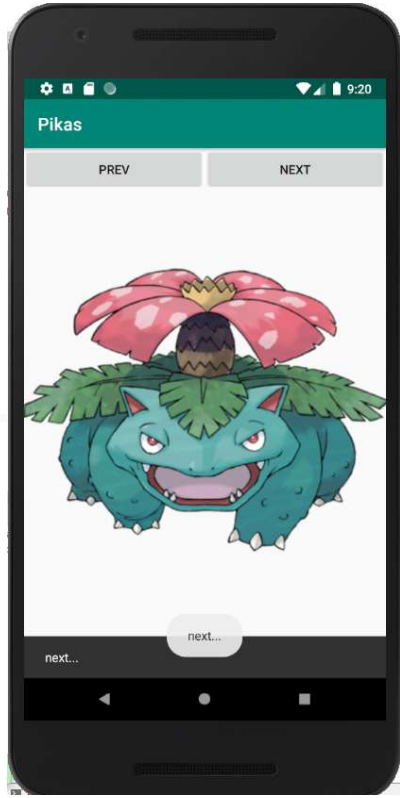
Pikas13.zip




AppLifeCycle13.zip

# Pikas

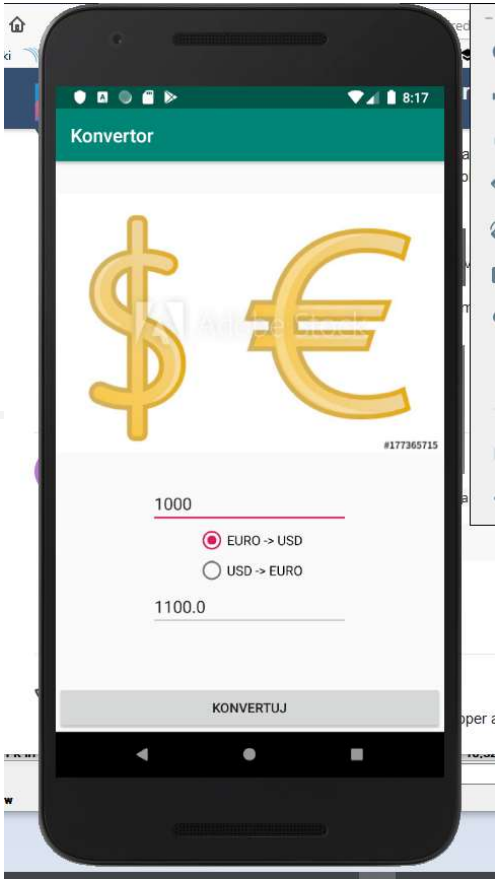



```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    var i = 0
    var imgs = arrayOf(
      ContextCompat.getDrawable(applicationContext,
                                R.drawable.butterfree),
                               ..., ..., ...    )
      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])
```

Pikas13.zip

# Konvertor EURO USD

## (logika)




Jednoduchá aplikácia na konverziu kurzov USD EURO

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

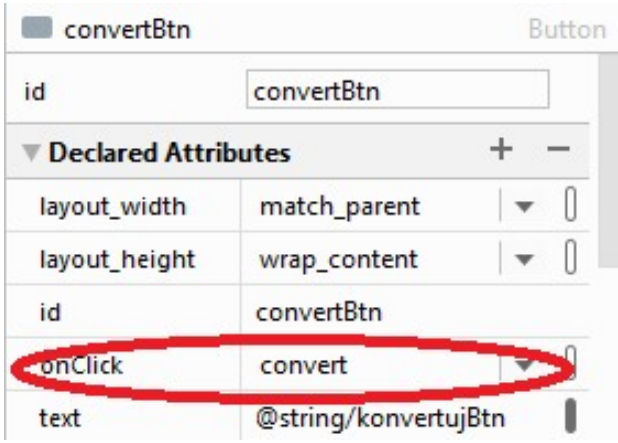
```
override fun onCreate(savedInstanceState: Bundle?)
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    convertBtn.setOnClickListener({
        Toast.makeText(this, "convert",Toast.LENGTH_SHORT).show();
        if (binding.inputText.text.isNotEmpty()) {
            val input = binding.inputText.text.toString().toFloat()
            var output = input
            val exchangeRate = 1.07f
            if (eur2usd.isChecked) output = exchangeRate * output
            if (usd2eur.isChecked) output = output / exchangeRate
            binding.outputText.setText("$output")        //
```

Klik na Konvertuj

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)
```

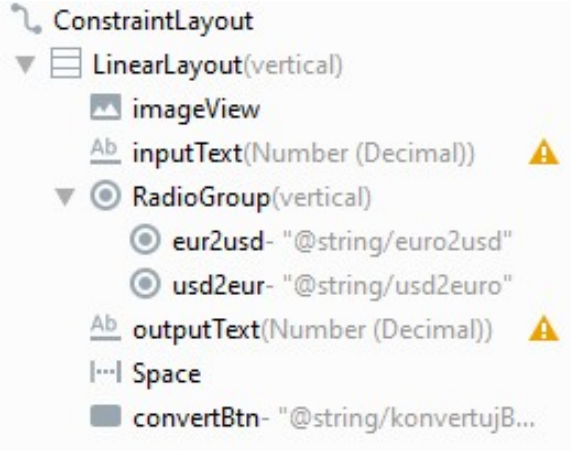
extension metóda Float

Konvertor13.zip

a

# Konvertor EURO USD
## (layout)

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


Konvertor13.zip

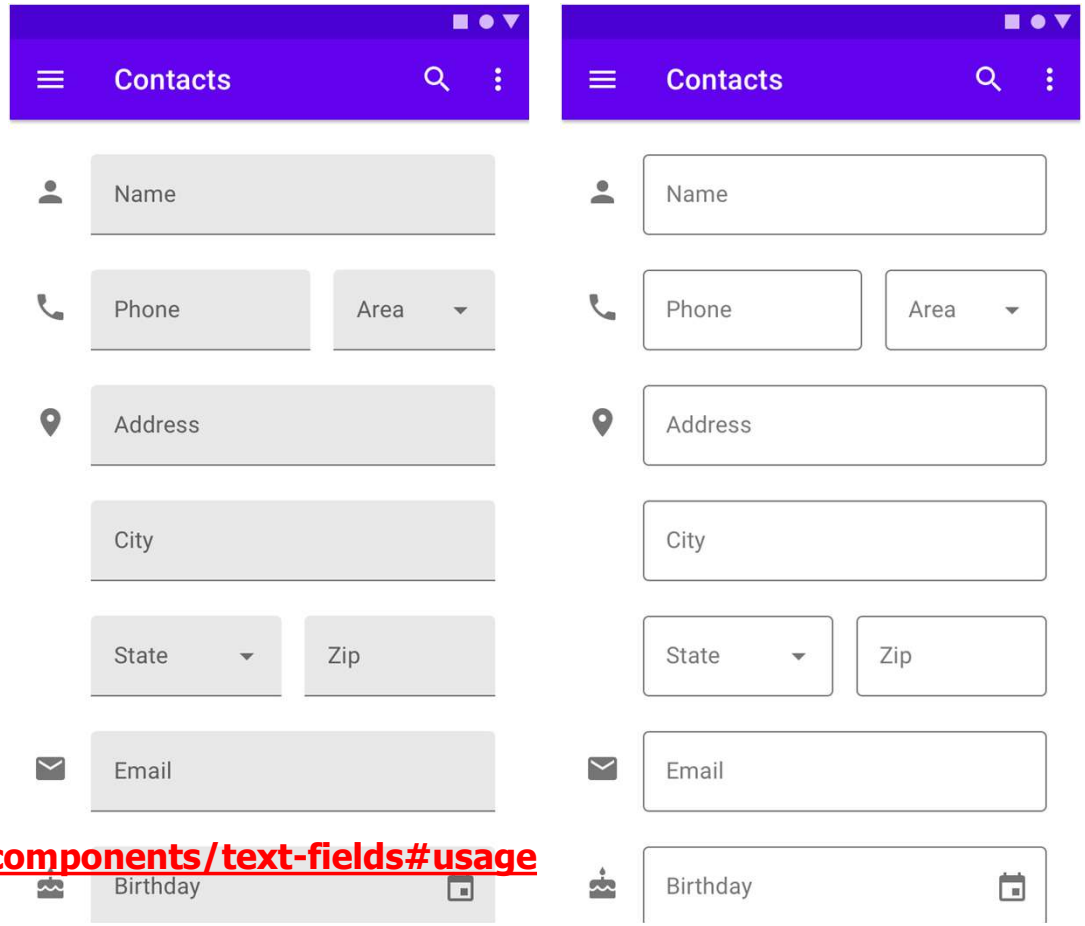
# Text Fields

## 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:1.9.0'
```

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

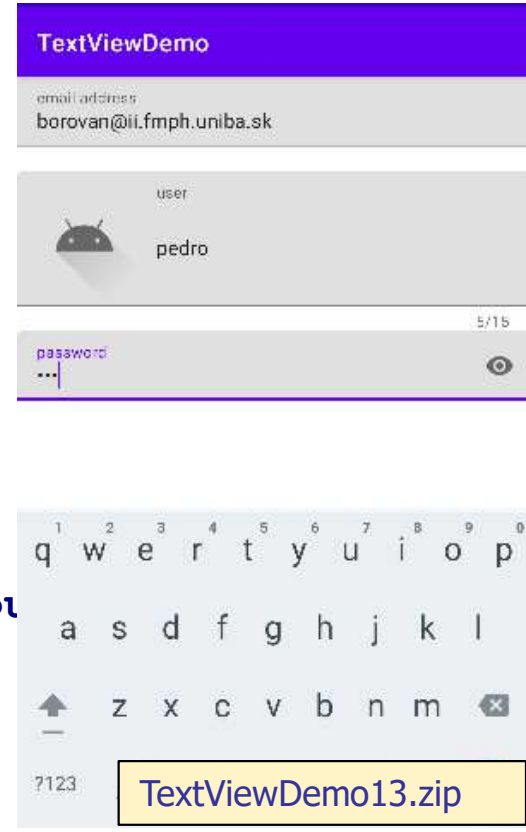



https://material.io/components/text-fields#usage

TextViewDemo13.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.TextInpu
```




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

TextViewDemo13.zip

# 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)
```

TextViewDemo13.zip

# Kotlin – pokračovanie

Cheat sheets

- https://www.programming-idioms.org/cheatsheet/Kotlin
- https://github.com/vmandro/Prednasky/tree/master/Kotlin

**The billion-dollar mistake**

I call it my billion-dollar mistake. It was the invention of the **null** reference in 1965…This has led to innumerable errors, vulnerabilities, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years.


**Sir Tony Hoare**

FRS FREng

Tony Hoare in 2011

| | |
|---|---|
| **Born** | Charles Antony Richard Hoare<br>11 January 1934 (age 85)<br>Colombo, British Ceylon |
| **Residence** | Cambridge |
| **Other names** | C. A. R. Hoare |
| **Alma mater** | University of Oxford (BA)<br>Moscow State University |
| **Known for** | Quicksort<br>Quickselect<br>Hoare logic<br>Null reference<br>Communicating Sequential Processes<br>Structured programming |
| **Awards** | Turing Award (1980)<br>Harry H. Goode Memorial Award (1981)<br>Faraday Medal (1985)<br>Computer Pioneer Award (1990)<br>Kyoto Prize (2000)<br>IEEE John von Neumann Medal (2011) |

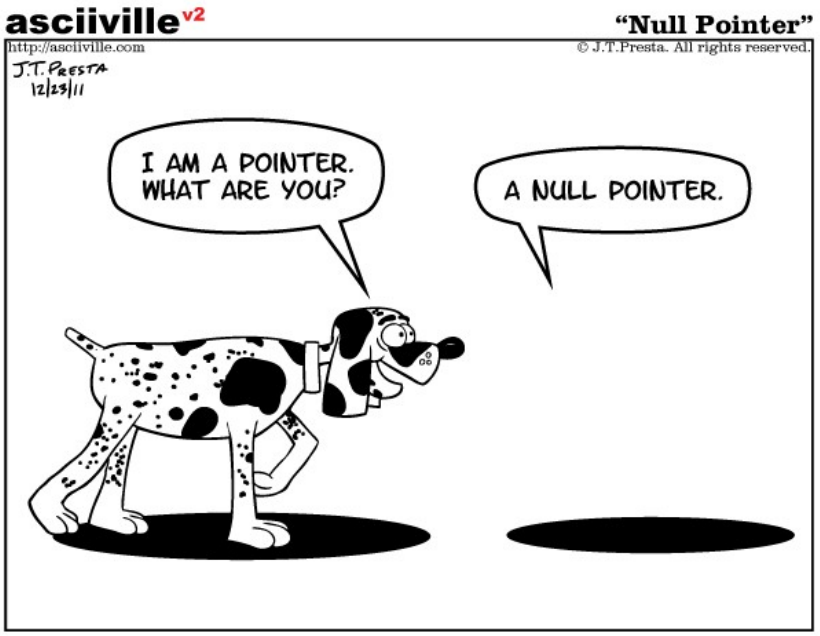
# Kotlin null safety

https://blog.kotlin-academy.com/null-safety-bb20b0cbf1a4

"Kotlin null safety" image created using OpenAI's DALL·E.

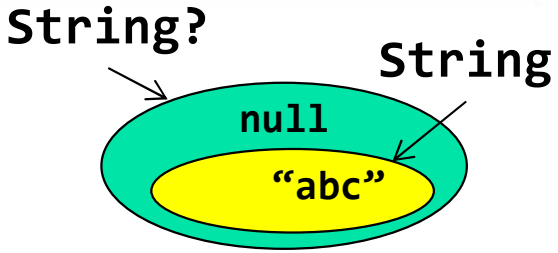
"Java's lack of inherent null safety" image created using OpenAI's DALL·E.

# Nullables




To, čo je

- Optional v Jave,  resp.
- Option v Scale, resp. kdekade iné inde




Napr. String? je typ pre reťazec alebo null
Ale String je typ len pre SKUTOČNÝ REŤAZEC, not-null

Preto `a:String?` nemôžete priradiť do `b:String`, lebo čo, ak by `a == null`

Ak ste skalo-pevne presvedčený, že hodnota `a:String? != null`,
môžete opatrne použiť BANG-BANG (**`!!`**) operátor a oklamať type-checker

```
val b:String = a!!
```

Ak ale neviete, či `a:String? =?= null`, tak použijete tzv. **Elvis operátor**

```
val c:String = a?:"default, ak je prázdny reťazec"
```

# Nullables

## (ďalšie operátory na konverziu medzi type a type?)

- Elvis operátor

  **obj?:default** = if (obj == null) default else obj

- Safe call operátor (Elvis na Žižku)

  **obj?.m()** = if (obj == null) null else obj.m()

- Not-null assertion (bang-bang !!)

  **obj!!** = if (obj != null) obj else N.P.E. – null pointer Ex.

- Safe cast

  **obj as? T** = if (obj typeof T) obj else null

  **obj as T** = if (!obj typeof T) cast exception

- let

  **obj?.let {…it…}** = if (obj != null) {…it <- obj…}

# Nullables

## (ešte raz, podrobnejšie)


V Jave je typ String skutočný reťazec alebo null

V Kotline String je **LEN skutočný reťazec** a null nepatrí do typu String

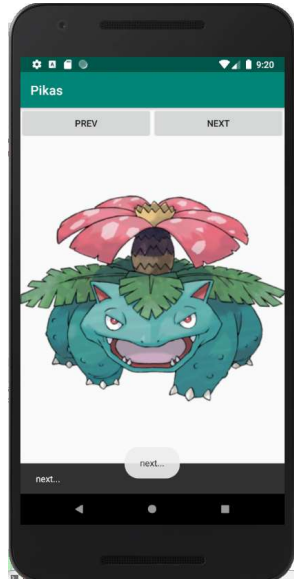
Existuje String? čo je String alebo null, vo všobecnosti: T? = T ∪ null

T? Podobne vo Swingu, Java Optional[T] =, Scala Option[T]

```
fun foo(str : String?) {
    println(str)
    if (str != null) println(str.toUpperCase())
    println(str?.toUpperCase())  // safe call operátor
                                 // x?.m == if (x != null) x.m else null
}
fun stringLen(s: String?): Int = s?.length?:0 // Elvis operátor
if (if (s == null) then null else s.length) == null then 0 else s.length
fun nonEmptystringLen(s: String?): Int {
    val sNotNull: String = s!!          // určite nebude null,
                // ak bude tak exception kotlin.KotlinNullPointerException
    return sNotNull.length
}
```

# Pikas

(rekapitulácia)




activity entry point

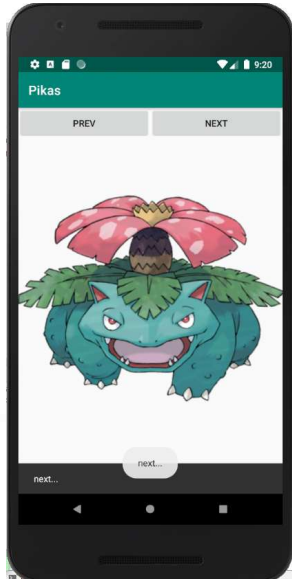
View(s)

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

Pikas13.zip

# Pikas

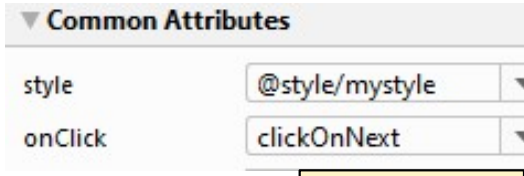
## (stav sa mieša s views a logikou – riešenie príde)




const
final

State

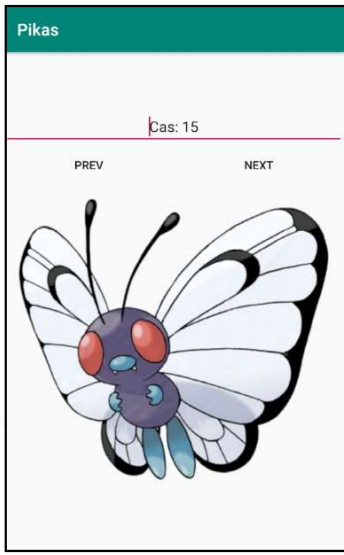
```
val TAG = "PIKAS"
var i = 0
var imgs = arrayOf<Drawable?>()
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)
    imgs = arrayOf(ContextCompat.getDrawable(applicationContext,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




Pikas13.zip

# Pikas

## (asynchrónnosť - timer)




pomocou **java.util.Timer**

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

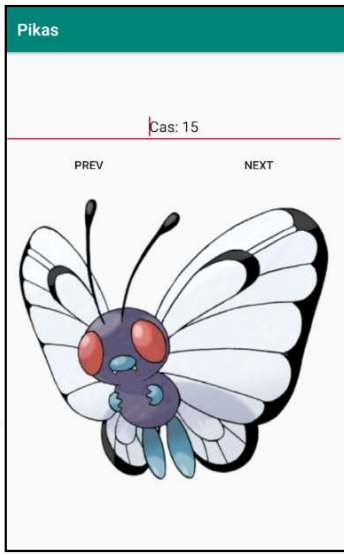
- nezabudnite na .run()
- **runOnUiThread**
  - má argument java.lang.Runnable, ktorý vykoná v hlavnom GUI vlákne

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

Pikas13.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 {
         binding.time.setText(
                        "Cas: ${millisUntilFinished/1000}") }
    }

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

tik

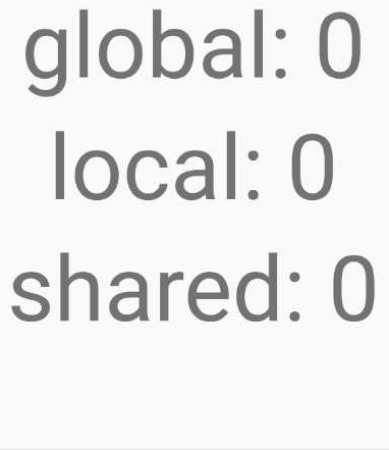
game over

ukončenie appky

Pikas13.zip

# Životný cyklus apky

## (prvý – zjednodušený nástrel)




- 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
}
```

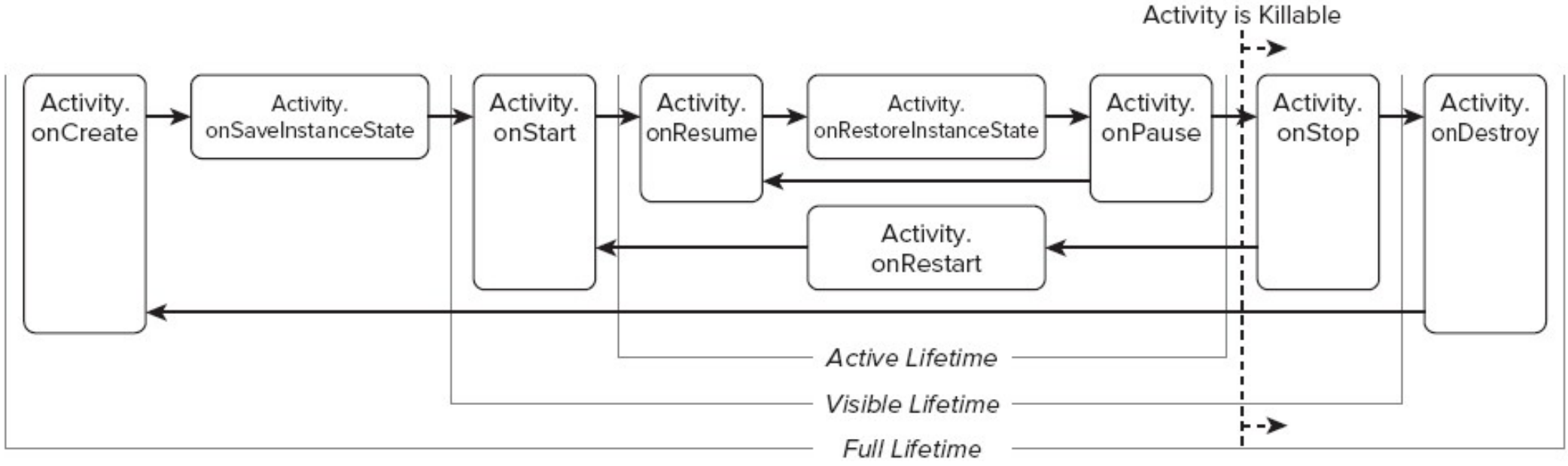
tag vhodný na filtrovanie

AppLifeCycle13.zip

# LogCat
## (Filtrovanie logov)

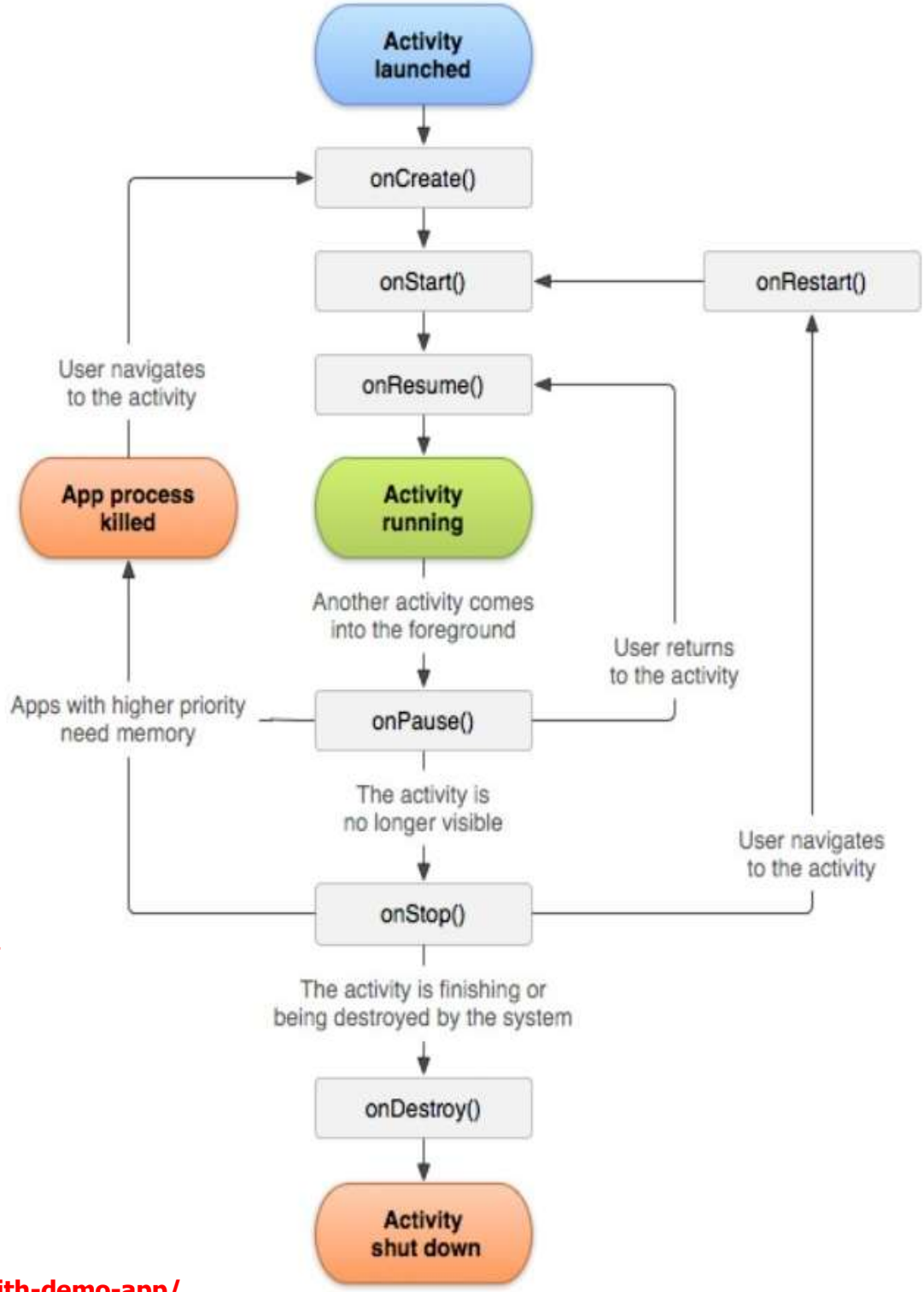
package:mine tag:CYKLUS

```
20719-20719 CYKLUS  com.example.applifecycle13  I  onCreate
20719-20719 CYKLUS  com.example.applifecycle13  I  onStart0
20719-20719 CYKLUS  com.example.applifecycle13  I  onResume0
20719-20719 CYKLUS  com.example.applifecycle13  I  onPause
20719-20719 CYKLUS  com.example.applifecycle13  I  onStop1
20719-20719 CYKLUS  com.example.applifecycle13  I  onDestroy1
```




zdroj: Reto Meier: PA2AD

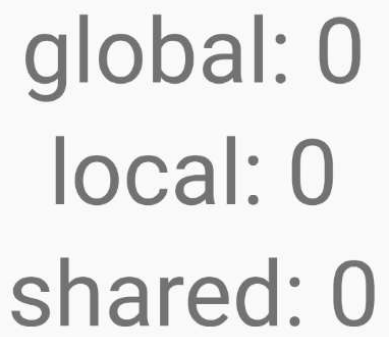
AppLifeCycle13.zip




https://media.geeksforgeeks.org/wp-content/uploads/20191125171002/Activity-Lifecycle-in-Android-Demo-App.mp4

https://www.geeksforgeeks.org/activity-lifecycle-in-android-with-demo-app/

# Persistencia
## (prvý dotyk)

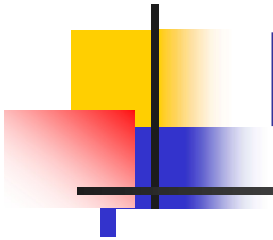



- **`globalCounter`** je premenná, ktorá sa
  - pri **`onSaveInstanceState`** uloží do Bundle (HashMap<String, Value>)
  - pri **`onCreate(savedInstanceState: Bundle?)`** príde táto Bundle ako argument
- **`localCounter`** je bežná lokálna triedna premená v MainActivity
- **`sharedCounter`** je premenná, 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`**

AppLifeCycle13.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
  ... OLD SCHOOL:
  if (savedInstanceState != null &&
     savedInstanceState.getInt("COUNTER") != null) {
     globalCounter = savedInstanceState!!.getInt("COUNTER")!!
  } else
     globalCounter = 0
}

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

AppLifeCycle13.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 {
       putInt("kluc",
            sharedCounter)
       apply()
    }
}
```

```
val editor = preferences.edit()
editor.putInt("kluc",
        sharedCounter)

editor.apply()
```

AppLifeCycle13.zip

# Čo je Kotlin ?



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