AS Projekt
(štruktúra projektu)

Peter Borovanský
KAI, I-18

MS-Teams: 2sf3ph4, List, github

borovan 'at' ii.fmph.uniba.sk

Čo dostaneme zadarmo
(pokračujeme v minulej prednáške)

```kotlin
package com.fmph.kai.prednaska2020

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

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

    override fun onCreate(savedInstanceState: Bundle?) {

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

        // sem sme minule písali náš prvý kotlin kód

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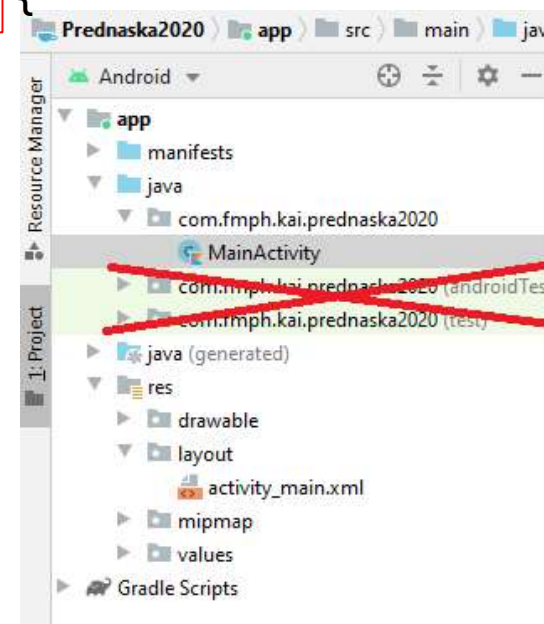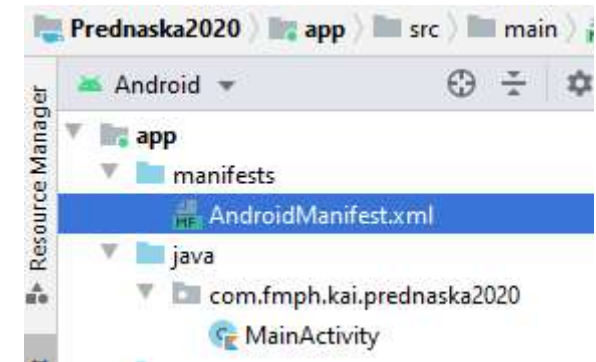
- MainActivity je inštancia triedy AppCompatActivity
- metóda onCreate sa volá v procese jej zobrazovania
- setContentView zobrazí layout podľa xml popisu v
  R.layout.*activity_main*
- argument **savedInstanceState:Bundle?** zatiaľ neriešte
- package androidTest a test môžete vymazať, pre prehľadnosť

Project: MyFirstApp2.zip

AndroidManifest.xml
(automaticky vygenerovaný súbor aplikácie)

```
Prednaska2020 > app > src > main
Android ▾
app
  manifests
    AndroidManifest.xml
  java
    com.fmph.kai.prednaska2020
      MainActivity
```

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"          ← referencia na ikonu apky
        android:label="@string/app_name"            ← referencia meno apky
        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>
```

Project: MyFirstApp2.zip

AndroidManifest.xml

Android ▾

- app
  - manifests
    - AndroidManifest.xml
  - java
    - com.fmph.kai.prednaska2020
      - MainActivity

Najhlavnejšie tagy:

- **`<application`** je jediný a popisuje ikony, logo, meno, štýl aplikácie
- **`<activity`** može ich byť viac a popisujú package definujúci aktivitu (Screen v MITI), 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 ochudobnel, mnohé veci sa presunuli do build.gradle:

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

build.gradle
(konfiguračný súbor pre gradle)

```
Android ▼

▼ app
    ▶ manifests
    ▼ java
        ▼ com.fmph.kai.prednaska2020
            MainActivity
    ▶ java (generated)
    ▶ res
▼ Gradle Scripts
    build.gradle (Project: Prednaska2020)
    build.gradle (Module: app)
    gradle-wrapper.properties (Gradle Version)
```

- build tool, podobne ako make, maven

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'
android {
    compileSdkVersion 29
    defaultConfig {
        applicationId "com.fmph.kai.prednaska2020"
        minSdkVersion 19
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    ...
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation"org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.0.2'
    ...
}
```
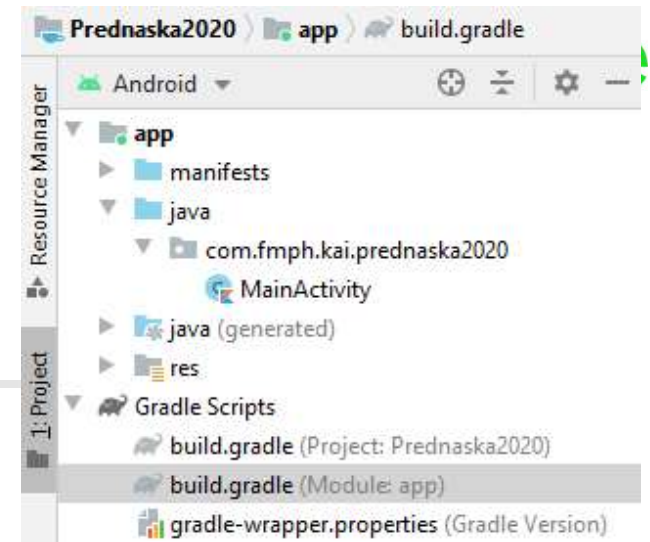
Project: MyFirstApp2.zip
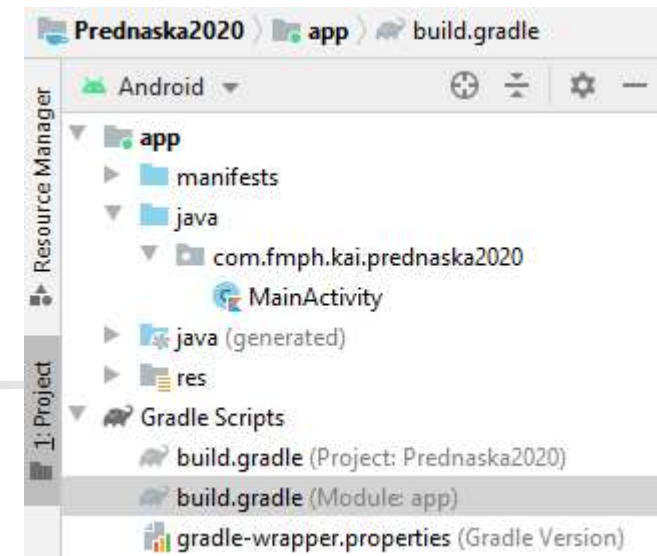
Gradle

- je plugin-based project-build/management system v AS založený na jazyku Groovy

- už existuje Kotlin Gradle Plugin pre Gradle 6+

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

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

Project: MyFirstApp2.zip

MergedManifest
(spája AndroidManifest a build.gradle)

```xml
<manifest
    android:versionCode="1"
    android:versionName="1.0"
    package="com.fmph.kai.prednaska2020"
    xmlns:android="http://schemas.android.com/apk/res/android" >
    <uses-sdk
        android:minSdkVersion="19"
        android:targetSdkVersion="29" />
    <application
        android:allowBackup="true"
        android:appComponentFactory="androidx.core.app.CoreComponen
        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="com.fmph.kai.prednaska2020.MainActivity" >
            <intent-filter>
                <action
                    android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
```

Manifest Sources

core:1.3.1 manifest

app main manifest (this file)

build.gradle injection

Other Manifest Files
(Included in merge, but did not contribute a
appcompat:1.2.0 manifest, viewpager:1.0.0 r
interpolator:1.0.0 manifest, savedstate:1.0.0
vectordrawable:1.1.0 manifest, lifecycle-view
appcompat-resources:1.2.0 manifest, lifecyc
vectordrawable-animated:1.1.0 manifest, fra
manifest, constraintlayout:2.0.1 manifest, cu
versionedparcelable:1.1.0 manifest, core-ktx
loader:1.0.0 manifest, activity:1.0.0 manifest,

Merging Log
Added from the app main manifest (this file

Text   Merged Manifest

referencia meno apky →

```xml
<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)
- menus – pre aktivity (bude neskôr)
- values – pomenované konštanty (strings.xml, colors.xml, styles.xml …)

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

```xml
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```
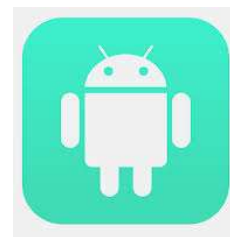
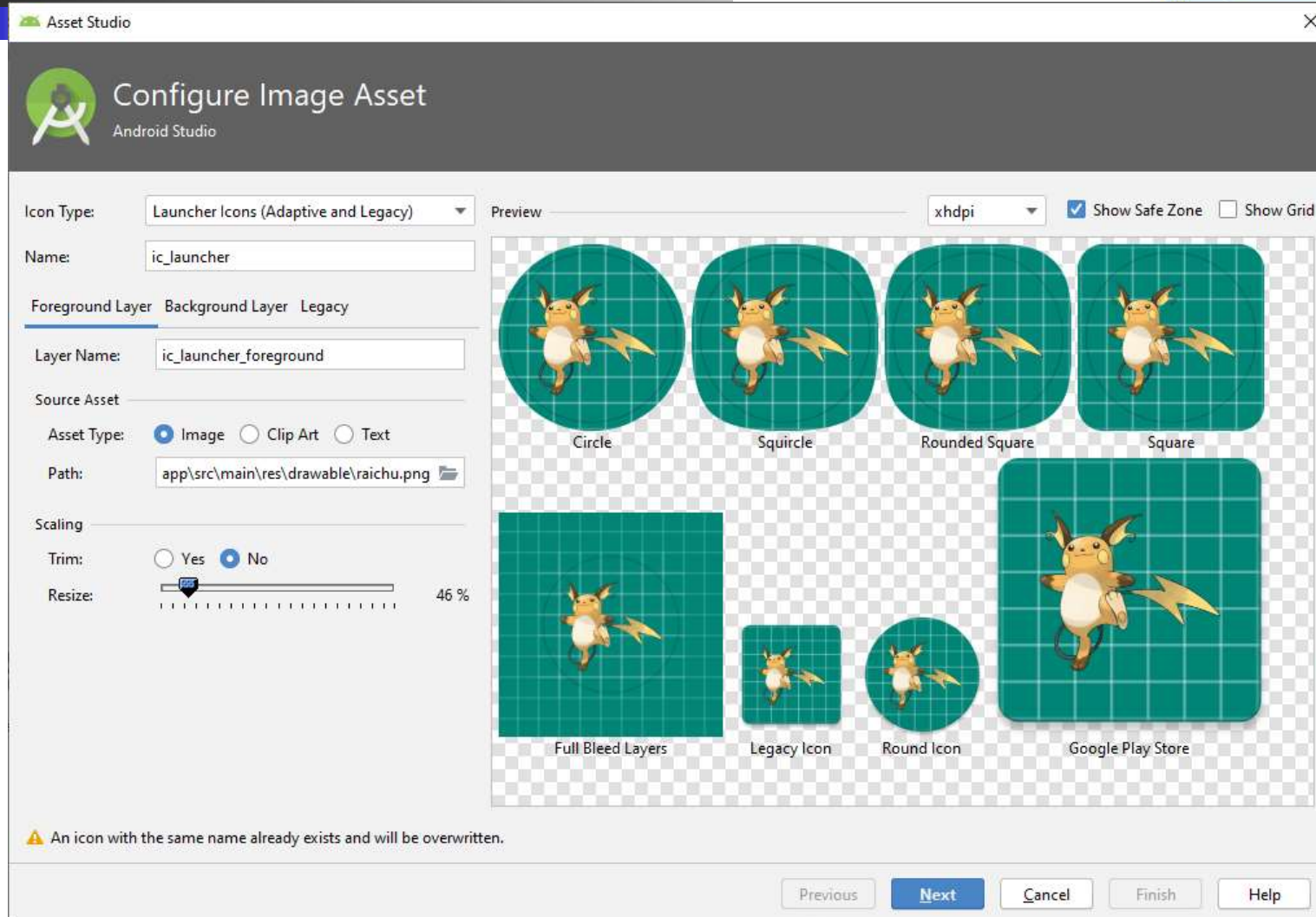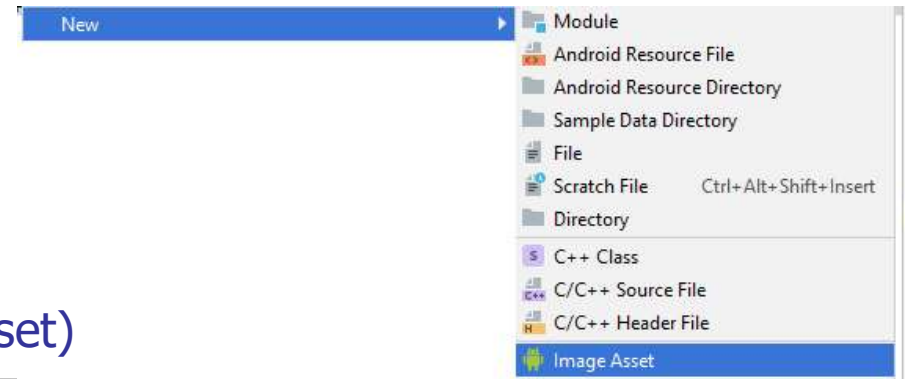Project: MyFirstApp2.zip

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.
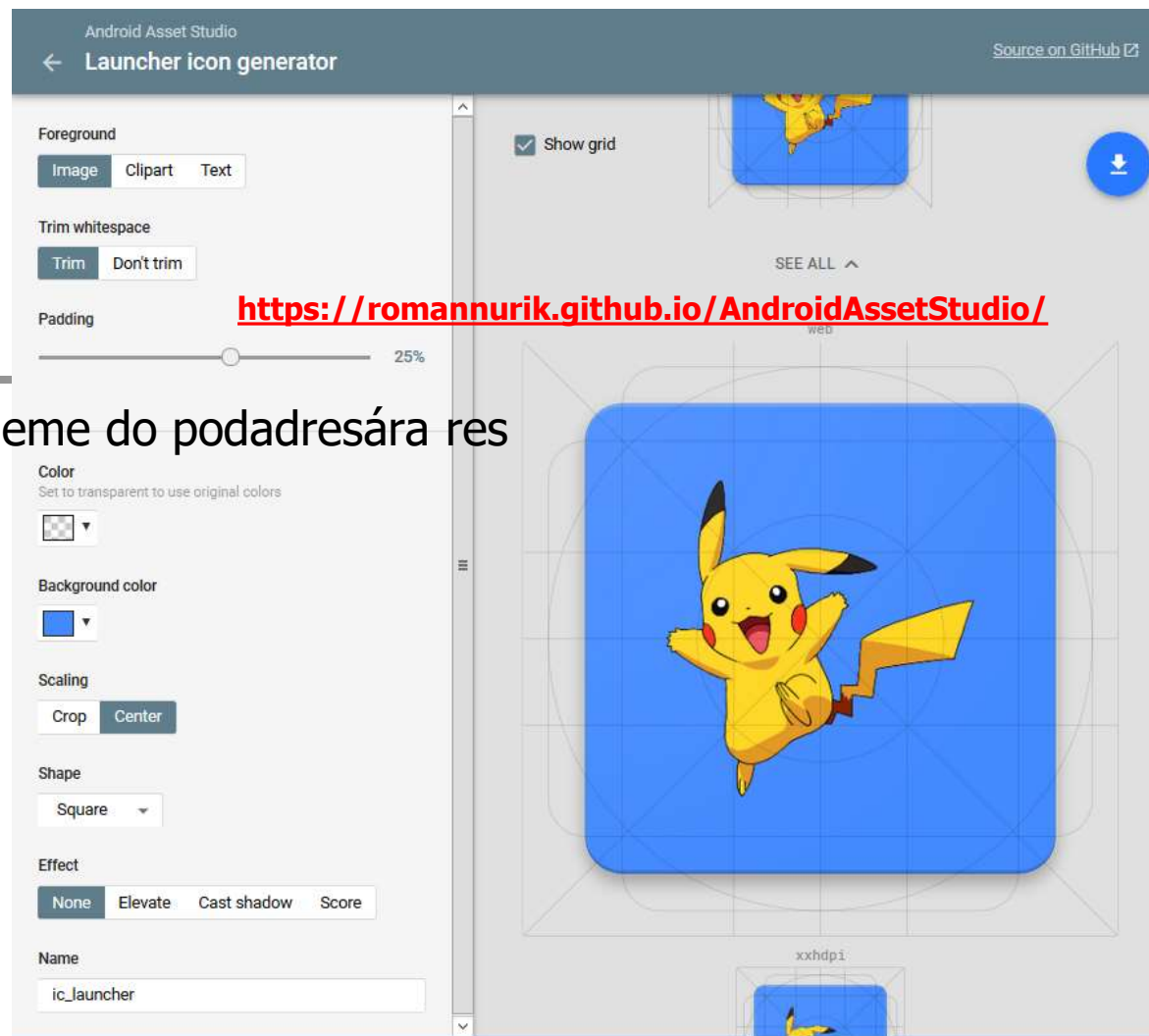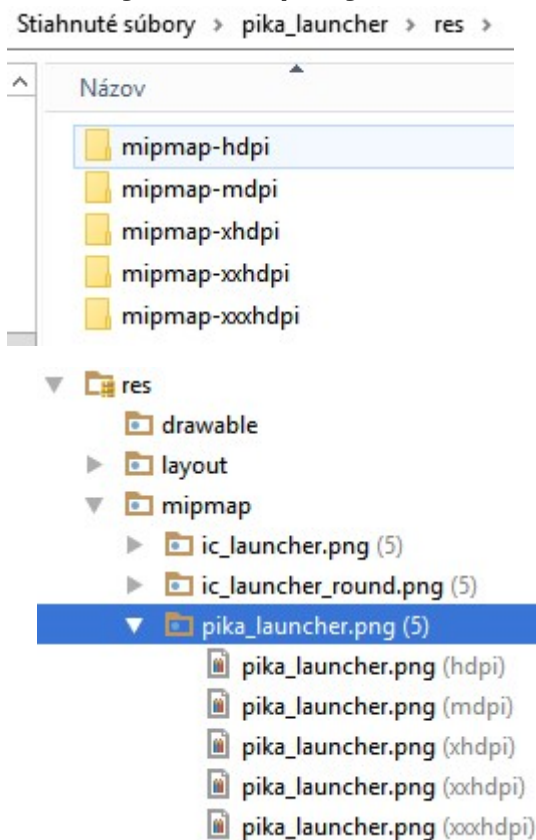
Buď kreatívny

(a použi Asset Studio - New/ImageAsset)

Android Asset Studio
Icon generator

Výsledok priamo nakopírujeme do podadresára res
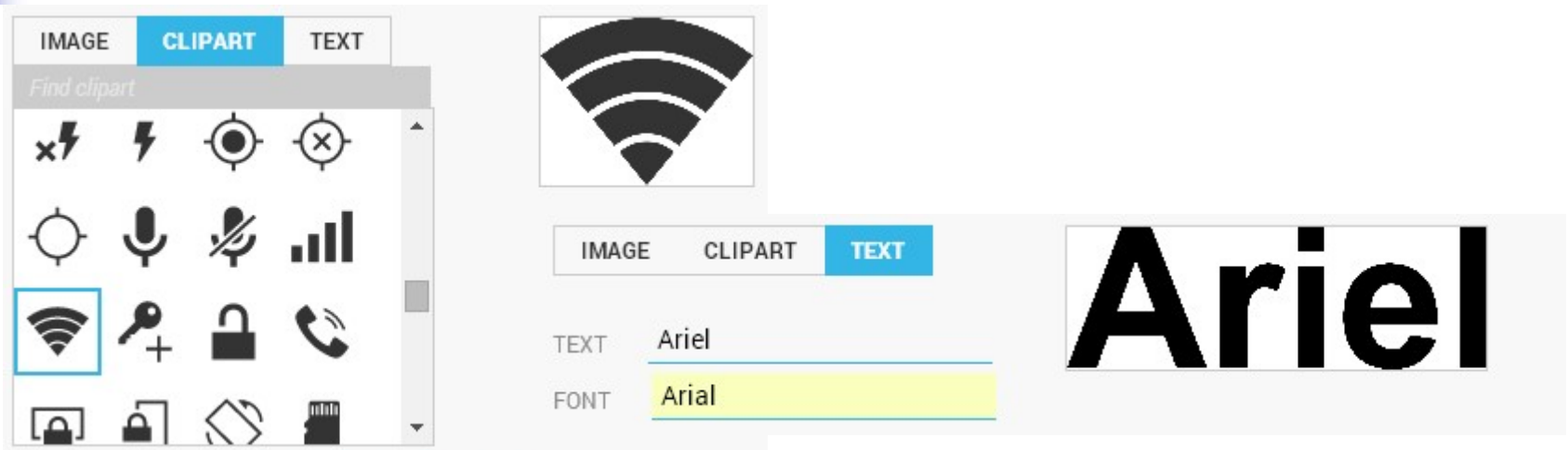
Ikony/obrázky sa

sa objavia v projekte

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

Project:Pikas.zip, Pikas2.zip

Android Asset Studio
(jedna z alternatív)

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

Resources/Drawables/Mipmap

(ikona - viacero rozlíšení)

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 extra^2 high-density (XXHDPI = ~ 480 dpi)
- 192x192 for extra^3 high-density (XXXHDPI = ~ 640 dpi)

ldpi (0.75x)

mdpi (baseline)

hdpi (1.5x)

xhdpi (2.0x)
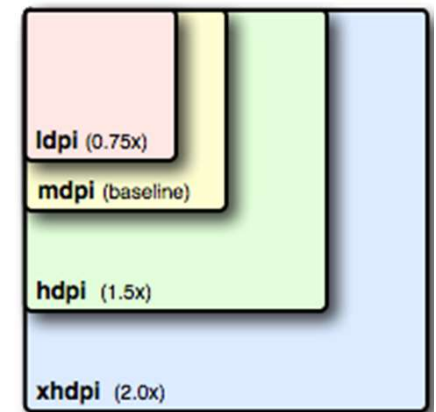
Pre .svg a .psd
(a použi Vector Asset Studio - New/VectorAsset)

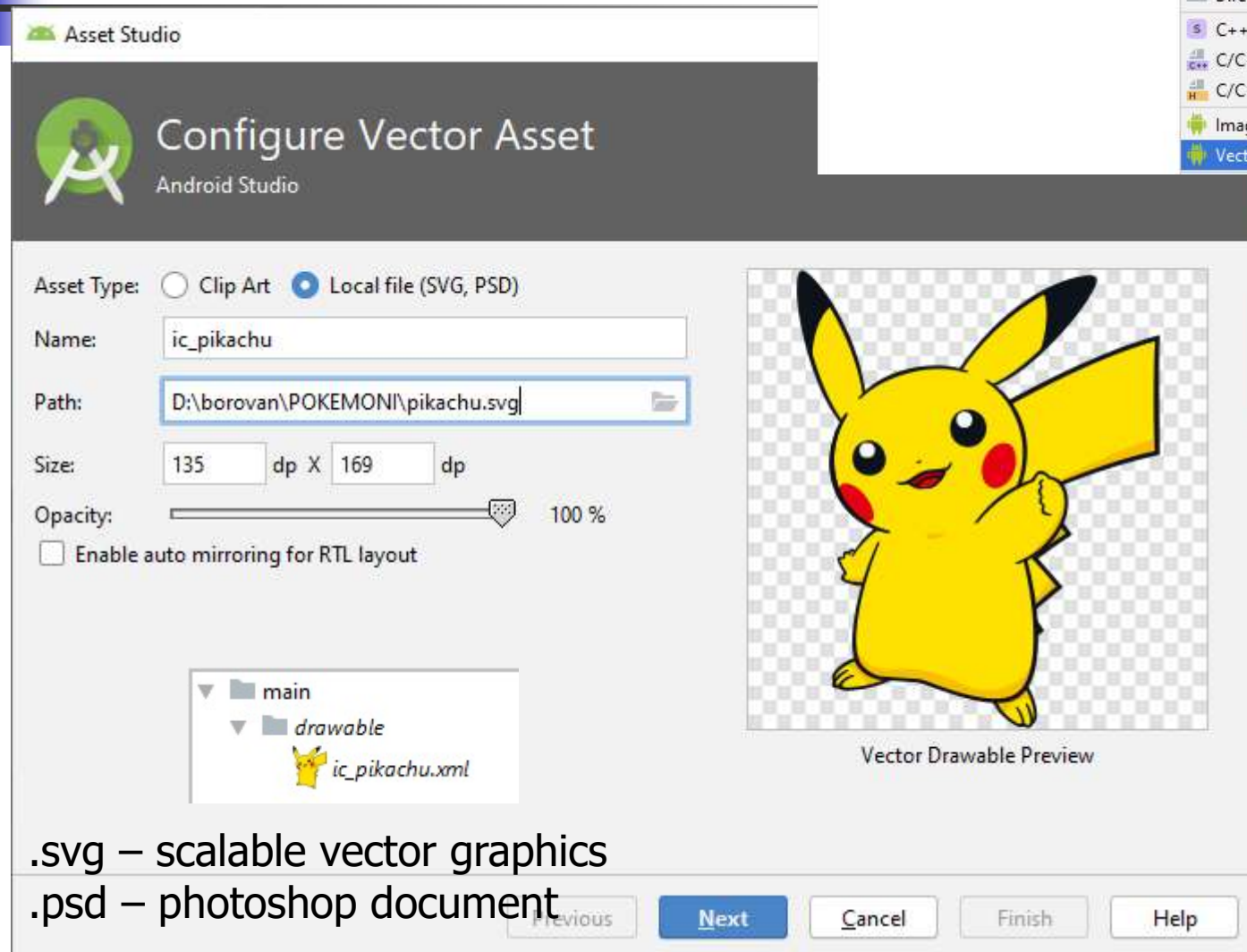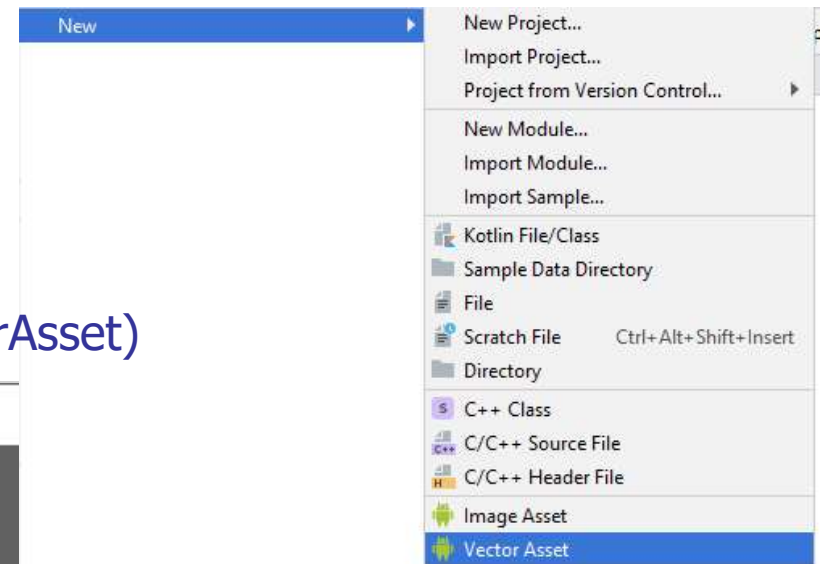| New	▶	New Project...
		Import Project...
		Project from Version Control... ▶
		New Module...
		Import Module...
		Import Sample...
		Kotlin File/Class
		Sample Data Directory
		File
		Scratch File Ctrl+Alt+Shift+Insert
		Directory
		C++ Class
		C/C++ Source File
		C/C++ Header File
		Image Asset
		Vector Asset

Asset Studio

Configure Vector Asset
Android Studio

Asset Type: ○ Clip Art   ● Local file (SVG, PSD)

Name: ic_pikachu

Path: D:\borovan\POKEMONI\pikachu.svg

Size: 135 dp X 169 dp

Opacity: ————————————●———— 100 %

☐ Enable auto mirroring for RTL layout

▼ 📁 main
　▼ 📁 drawable
　　🐭 ic_pikachu.xml

Vector Drawable Preview

.svg – scalable vector graphics
.psd – photoshop document

Previous   **Next**   Cancel   Finish   Help

Resources/Values

- string

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

- color

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

- dimentions

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

- style

```
<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

- 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>
```
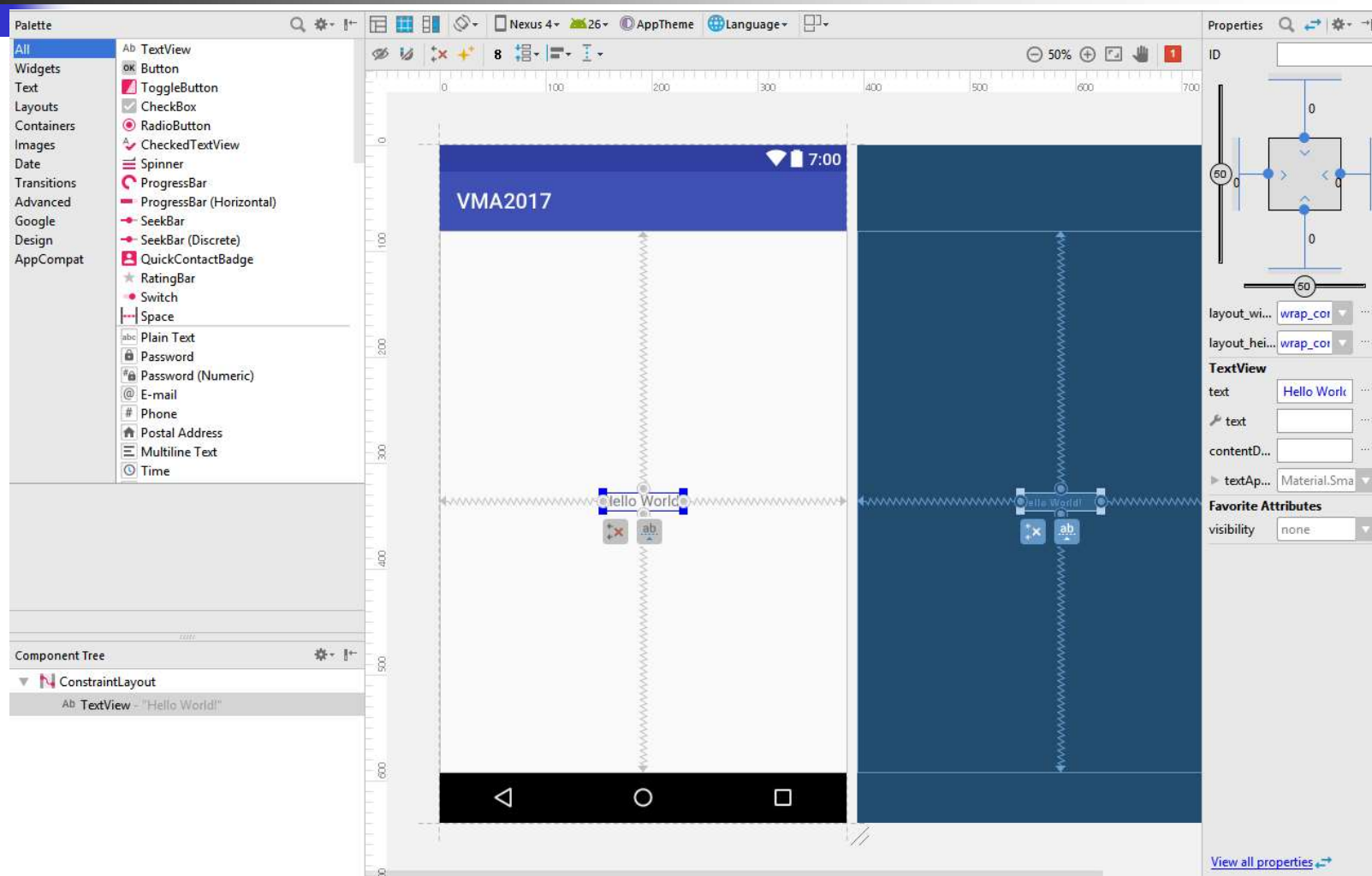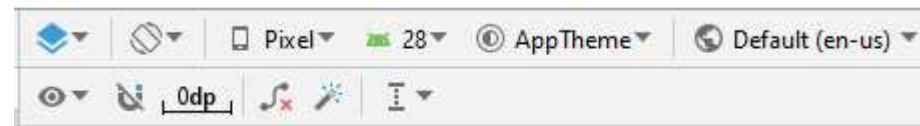
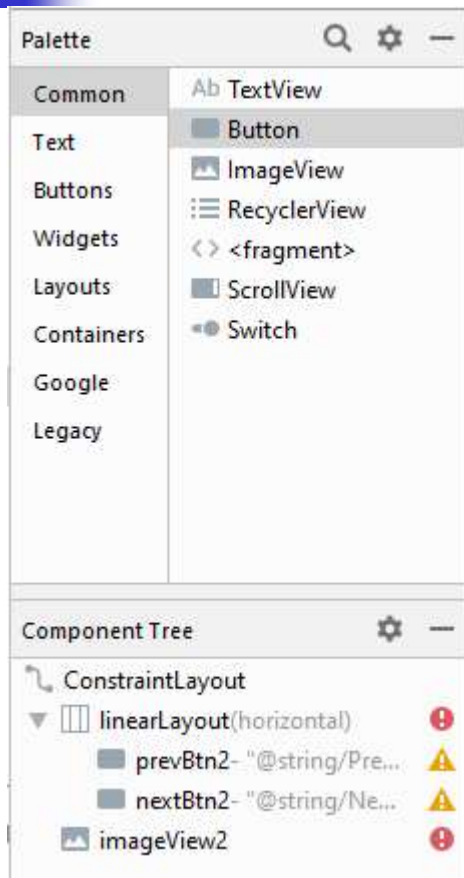- plurals (quantity strings)

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

Resources/Layout
(Design View)

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

	3 Warnings 2 Errors	
Message		**Source**
► ❶	**Missing Constraints in ConstraintLayout**	linearLayout <LinearLayout>
► ❶	**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)

VMA2017

Hello World !

```xml
<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"
```

wrap_content
fill_parent=
match_parent

ConstraintLayout

Ab res "Hello World 1"

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

```xml
        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

Resources/Layout

(Text View)

```xml
<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>
```
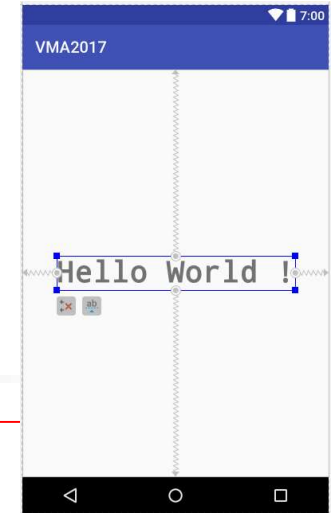
wrap_content
fill_parent=
match_parent

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

<resources>
    <dimen name="reallyBigFont">3(
</reso
```

Ako by to malo vyzerať

Žiadne warnings

```xml
<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>
```

Component Tree

- LinearLayout1 (vertical)
  - LinearLayout (horizontal)
    - Ab ScoreView (TextView) - "@string/score_str"
    - Ab timeView (TextView) - "@string/time_str"
  - imageView1
  - LinearLayout (horizontal)
    - OK Startbtn (Button) - "@string/start_str"
    - OK Stopbtn (Button) - "@string/stop_str"
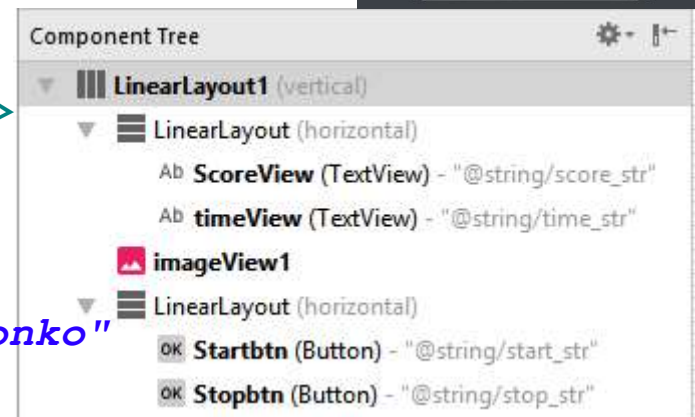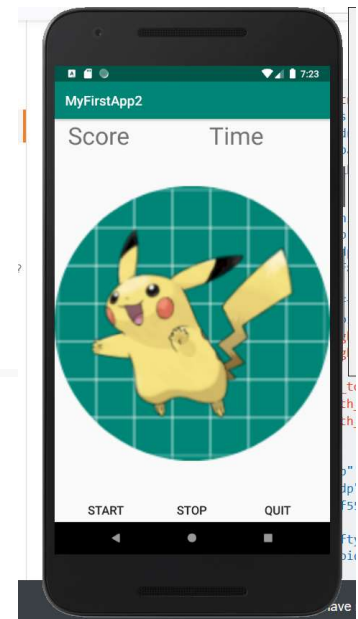
zjednodušené pre účely slajdu

Project: MyFirstApp22.zip

Logovanie

Tri najbežnejšie spôsoby:

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

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


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

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

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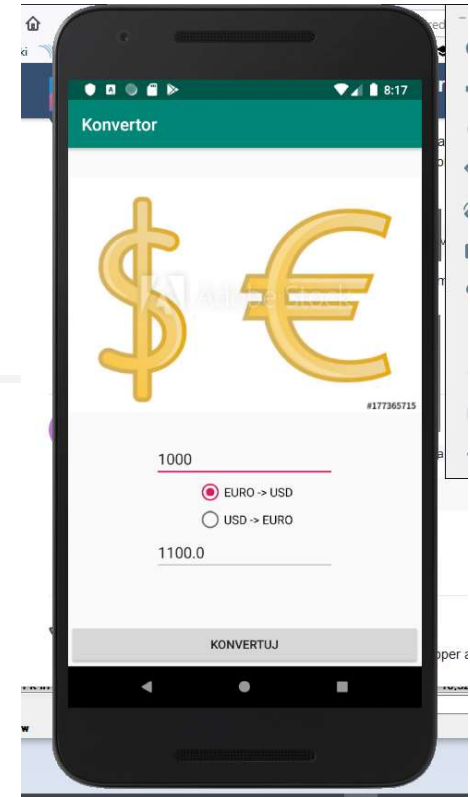
Pikas

```kotlin
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])
    })
}
```
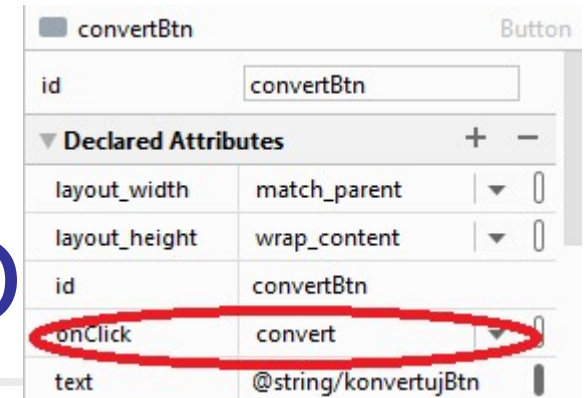
Project:Pikas2.zip

Konvertor EURO USD


```kotlin
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
            if (eur2usd.isChecked) output = 1.1F * output
            if (usd2eur.isChecked) output = output / 1.1F
            outputText.setText("$output")
        }}
    )
}
```

Project:Konvertor.zip

Konvertor EURO USD

convertBtn	Button
id	convertBtn

▼ Declared Attributes   + −

layout_width	match_parent ▼
layout_height	wrap_content ▼
id	convertBtn
onClick	**convert** ▼
text	@string/konvertujBtn

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

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


  fun convert(v: View) {
      Toast.makeText(this, "convert", Toast.LENGTH_SHORT).show()
      if (inputText.text.isNotEmpty()) {
          val input = inputText.text.toString().toFloat();
          var output = input
          if (eur2usd.isChecked) output = 1.1F * output
          if (usd2eur.isChecked) output = output / 1.1F
          outputText.setText("$output")
      }
  }
```

Príklad jednoduchej aplikácie

(ktorú sme si vyklikali minule)

Ilustrovali sme:

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

Nestihli sme:

- aktivitu a jej životný cyklus
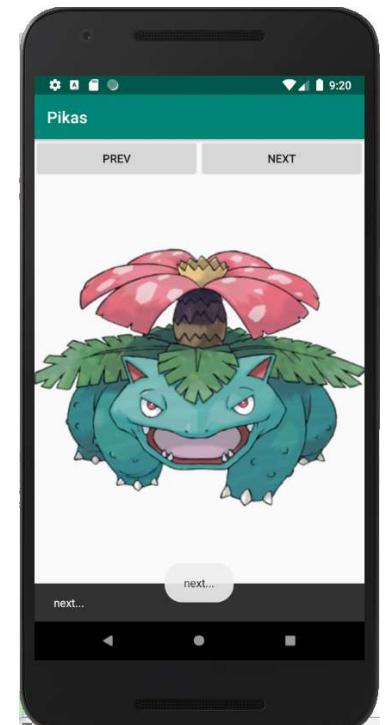
Project:Pikas2.zip

Logovanie
(rekapitulácia)

Tri najbežnejšie spôsoby:

- Log – loguje do okna Logcat, filtrujte podľa `TAGu` metódy `Log.d(TAG,`

- Toast – potrebuje `Context` (zjednodušene aktivita, v ktorej sa toastuje)

- Snackbar – to chce pridať závislosť do build.gradle a import snackbaru

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

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

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

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

})
```

Pikas
(rekapitulácia)

activity entry point

logovanie

View(s)

```kotlin
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])
    })
}
```
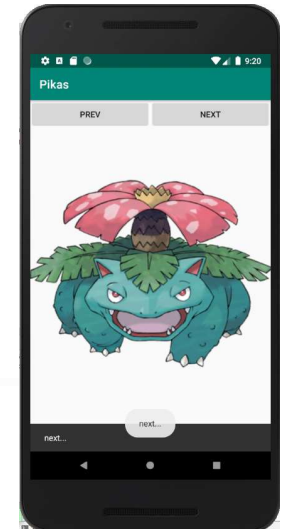
Project:Pikas2.zip

Pikas

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

const final

```kotlin
val TAG = "PIKAS"
var i = 0
var imgs = arrayOf<Drawable?>()
```

State

```kotlin
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])
}
```

▼ Common Attributes	
style	@style/mystyle
onClick	clickOnNext

Project:Pikas2.zip
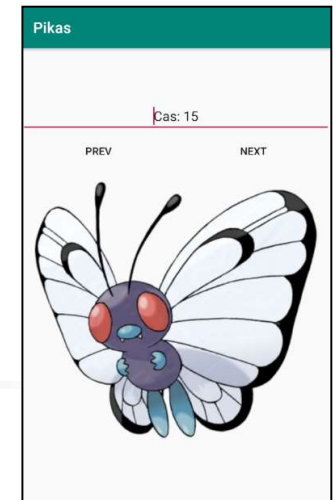
Pikas

(asynchrónnosť - timer)

pomocou `java.util.Timer`

```
Timer("tik-tak").schedule(1000,1000) { // delay, period
    Log.d(TAG, "onTICK")
    cas++
    runOnUiThread { time.setText("Cas: $cas ") }
}.run()
```
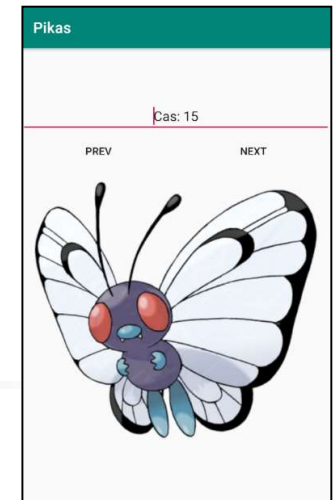
- nezabudnite na .run()
- runOnUiThread
  - má argument java.lang.Runnable, ktorý vykoná v hlavnom GUI vlákne

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

Project:Pikas2.zip

Pikas

(asynchrónnosť – count down)

pomocou `android.os.CountDownTimer`

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

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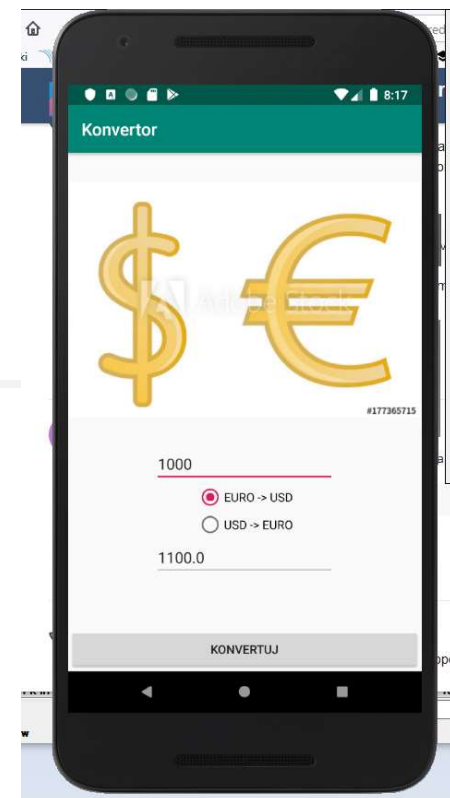
tik

game over

ukončenie appky

Project:Pikas2.zip

Konvertor EURO USD

(logika)

Jednoduchá aplikácia na konverziu kurzov USD EURO
- s modifikovateľným TextView pre zadanie sumy, reálneho čísla
- RadioButtonom pre výber smeru konverzie
- s nemodifikovateľným poľom pre výsledok
- Button Konvertuj pre vykonanie akcie

```kotlin
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    convertBtn.setOnClickListener({                    // Klik na Konvertuj
        Toast.makeText(this, "convert",Toast.LENGTH_SHORT).show();
        if (inputText.text.isNotEmpty()) {
            val input = inputText.text.toString().toFloat();// get
            var output = input
            if (eur2usd.isChecked) output = 1.1F * output
            if (usd2eur.isChecked) output = output / 1.1F
            outputText.setText("$output")                  // set
        }
    }
```

Konvertor EURO USD

(setOnClickListener)

convertBtn		Button
id	convertBtn	
▼ **Declared Attributes**		**+** **−**
layout_width	match_parent	▼ 〔
layout_height	wrap_content	▼ 〔
id	convertBtn	
onClick	convert	▼
text	@string/konvertujBtn	

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


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



  fun convert(v: View) {
      Toast.makeText(this, "convert", Toast.LENGTH_SHORT).show()
      if (inputText.text.isNotEmpty()) {
          val input = inputText.text.toString().toFloat();
          var output = input
          if (eur2usd.isChecked) output = 1.1F * output
          if (usd2eur.isChecked) output = output / 1.1F
          outputText.setText("${output.format(2)}")     }}
  fun Float.format(digits: Int) =
      java.lang.String.format("%.${digits}f", this)
```

metóda
Float

Project:Konvertor.zip

Konvertor EURO USD
(layout)


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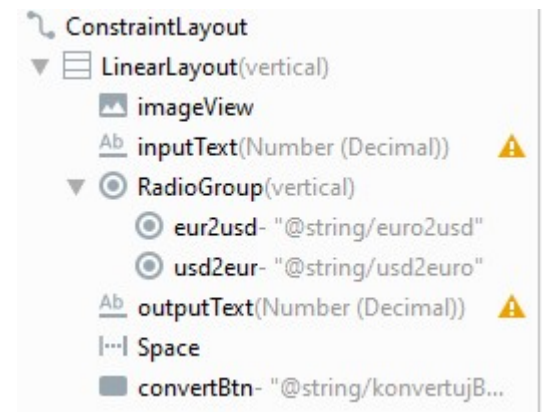
Životný cyklus apky

(prvý – zjednodušený nástrel)

AppLifeCycle

global: 0
local: 0
shared: 0

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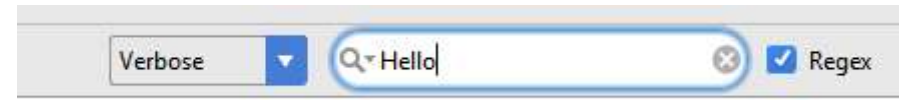- do každej metódy dáme kontrolný výpis, aby sme pochopili životný cyklus

```
@Override

public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);

    Log.d("CYKLUS","onCreate"); // LOGUJTE, LOGUJTE, LOGUJTE

}
```

tag vhodný na filtrovanie

Project:AppLifeCycle.zip

LogCat
(Filtrovanie logov)

| Verbose | ▼ | Q▾ Hello | ⊗ | ☑ Regex |

- **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**

Activity is Killable

Activity.onCreate → Activity.onSaveInstanceState → Activity.onStart → Activity.onResume → Activity.onRestoreInstanceState → Activity.onPause → Activity.onStop → Activity.onDestroy

Activity.onRestart

Active Lifetime

Visible Lifetime

Full Lifetime

Project:AppLifeCycle.zip