Prvé aplikácie

ViewBinding, Nullables, Layouts, životný cyklus

Peter Borovanský
KAI, I-18

borovan 'at' ii.fmph.uniba.sk

Odovzdávanie riešení zadaní

do Listu

Odovzdávajte .zip, ktorý v koreni obsahuje

- **README** s popisom ovládania, resp. vami implementované detaily nespomenuté v zadaní,
- projekt AS s **vymazaným build adresárom** (app/build) – výrazne schudne, a potom nie je problém s limitom 30MB pre List
- ak sa inak nedá, link na alternatívne úložisko, ale tiež bez app/build
- **vytiahnutým .apk** súborom z app/build/intermediares/apk/debug do koreňa .zipu, aby ho opravujúci nemuseli hľadať v útrobách projektu

Za nedodržanie tohoto formátu sa strhávajú body, ktoré sú úmerné času cvičiaceho, aby:

- zo zdrojáku pochopil ovládanie vašej apky (ak ste zabudli README)
- kompilovaním vášho projektu, ak ste neposkytli .apk
- resp. uviedli minSDK=33, leby vy máte zariadenie s Android 13

Code review robíme pre vás, tak nám pomôžte. Ďakujeme za pochopenie.

Príklad jednoduchej aplikácie

Ilustruje:

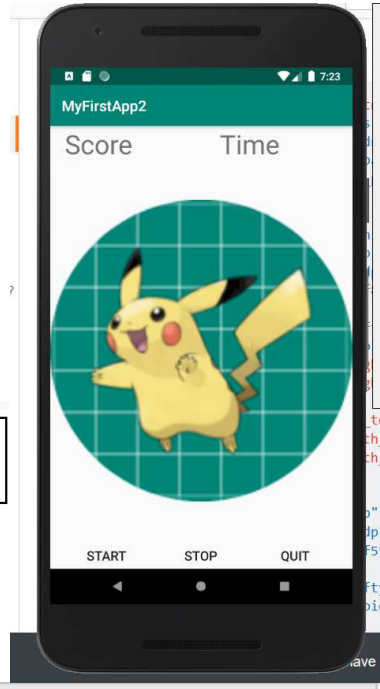
- príklad návrhu (vyklikania) jednoduchého GUI (single activity app)
- logovanie udalostí ako efektívny prostriedok ladenia pomocou
  - Log.i(…)
  - Toast.make(…)
  - Snackbar.make(…)
- používanie Image/Vector Asset (drawable/mipmap)
- používanie resource editora (pri definovaní strings.xml)
- používanie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

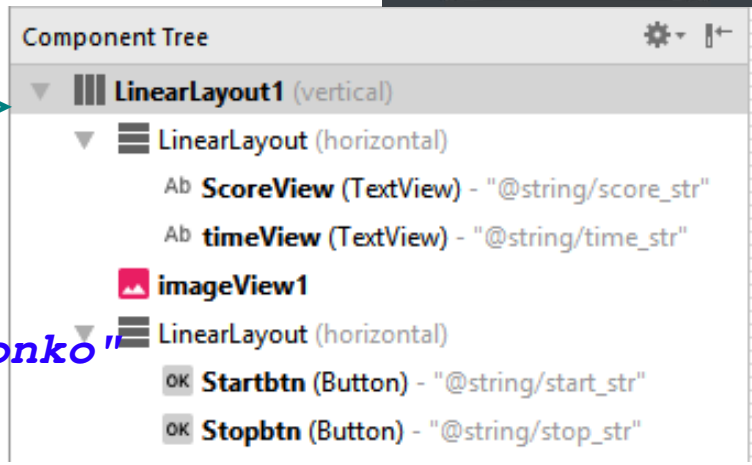
Pikas13.zip

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
- ~~syntetics – 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
        }
```

```
android {
    buildFeatures {
        viewBinding = true
    }
    compileSdkVersion 30
    defaultConfig {
        applicationId "com.example.pikas"
        minSdkVersion 23
        targetSdkVersion 30
```

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

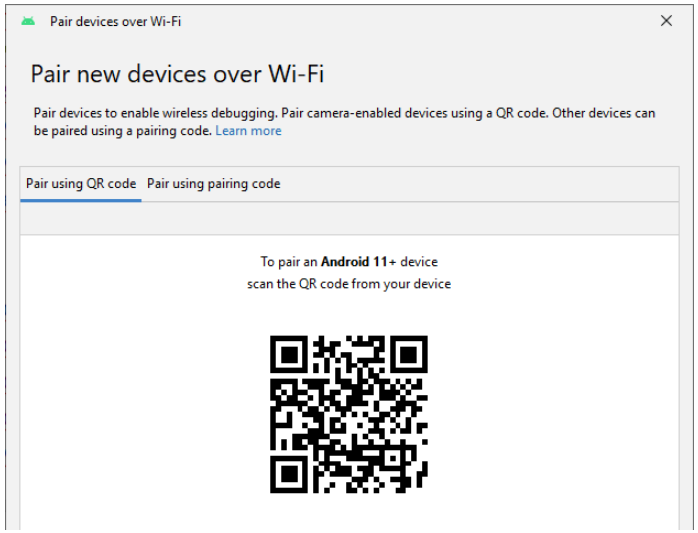
- Android Debug Bridge (ADB)

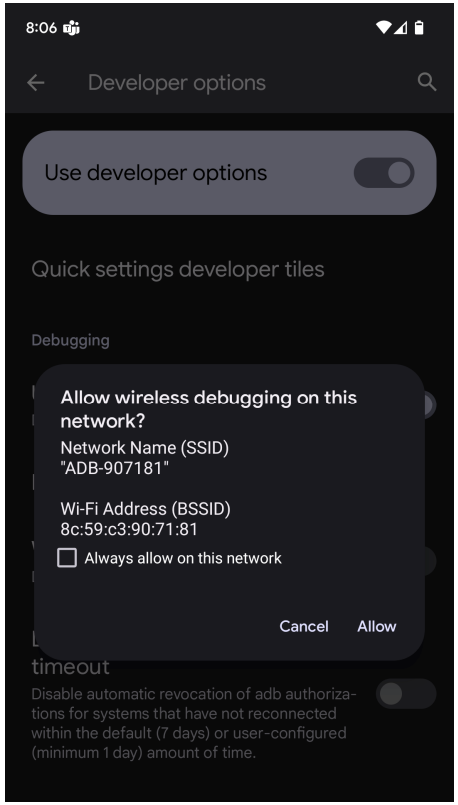
```
C:\Users\borovan>adb -s emulator-5554 emu kill
OK: killing emulator, bye bye
OK
```

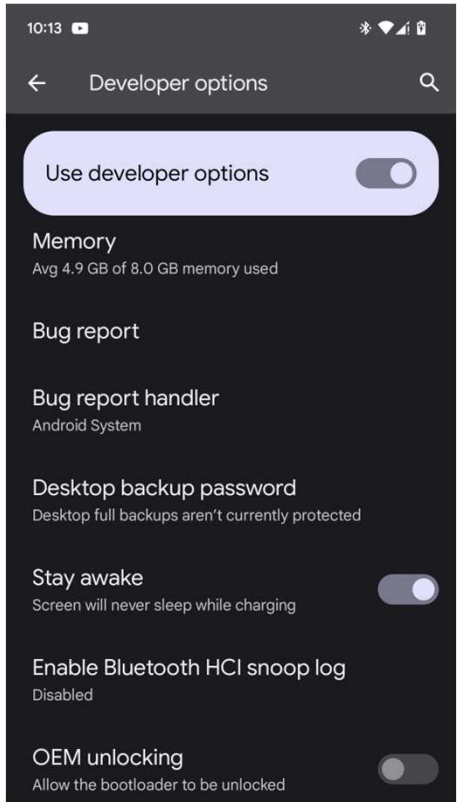
```
C:\Users\borovan>adb devices
List of devices attached
emulator-5554   device


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

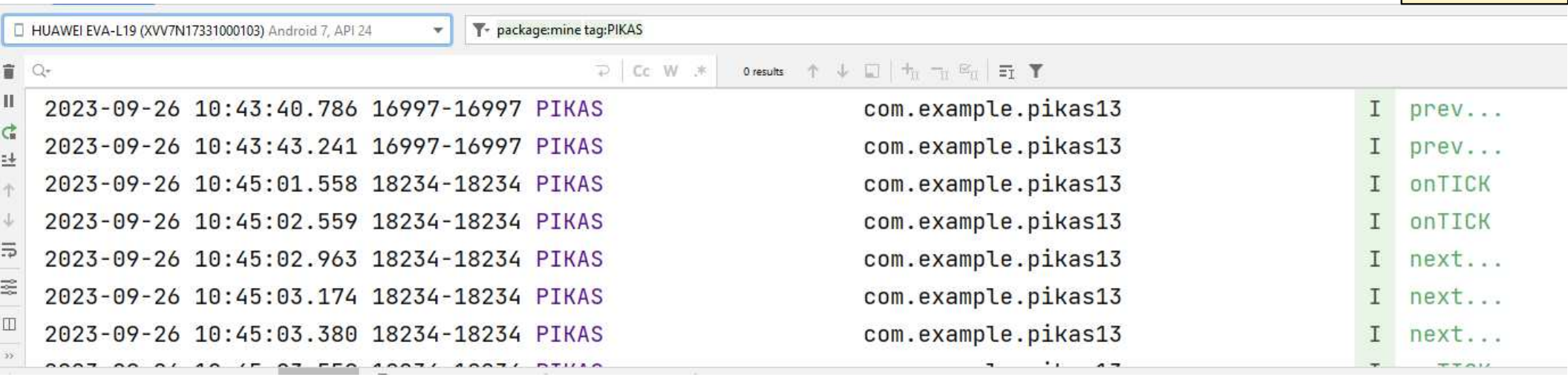
```
import com.google.android.material.snackbar.Snackbar
prevBtn2.setOnClickListener {
    Toast.makeText(this@MainActivity, "prev...", Toast.LENGTH_SHORT)
        .show()

    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

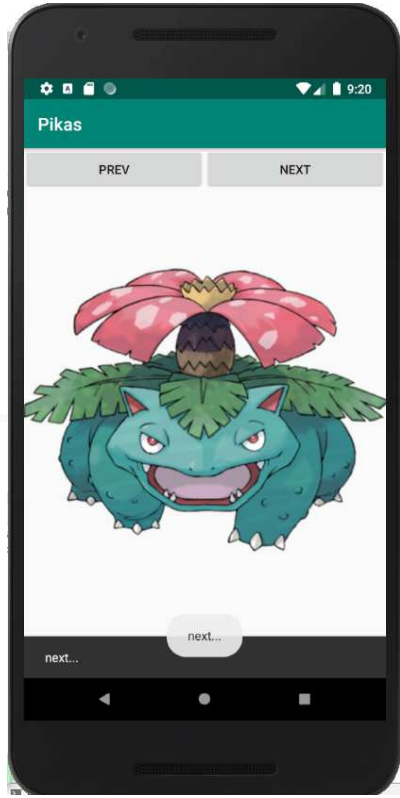
```
val TAG = "PIKAS"
Log.i(TAG, "prev...")
```

Pikas13.zip



AppLifeCycle13.zip

Pikas

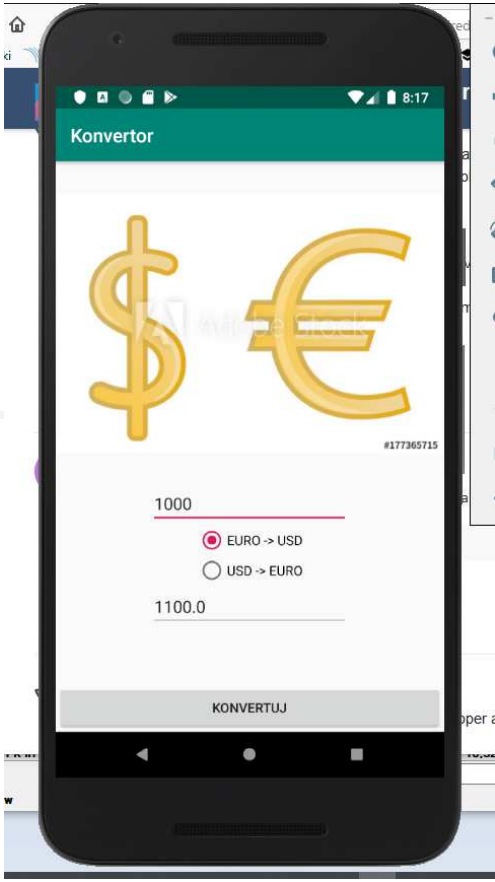



```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    var i = 0
    var imgs = arrayOf(
      ContextCompat.getDrawable(applicationContext,
                                R.drawable.butterfree),
               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

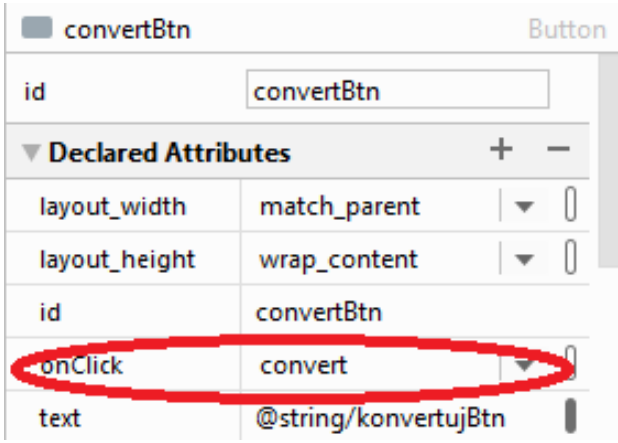
```
override fun onCreate(savedInstanceState: Bundle?)
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    convertBtn.setOnClickListener({
        Toast.makeText(this, "convert",Toast.LENGTH_SHORT).show();
        if (binding.inputText.text.isNotEmpty()) {
            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

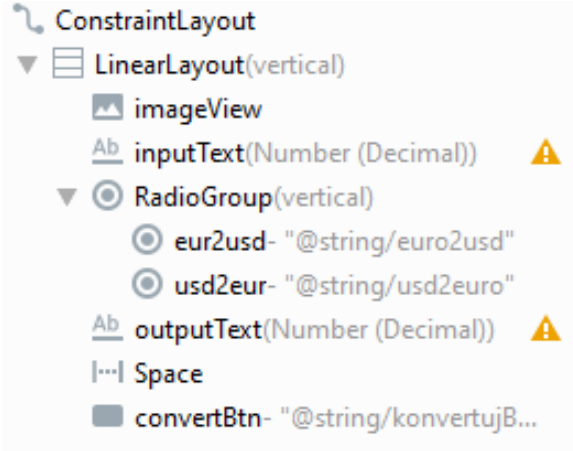
Konvertor13.zip

a

Konvertor EURO USD

(layout)

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



Konvertor13.zip

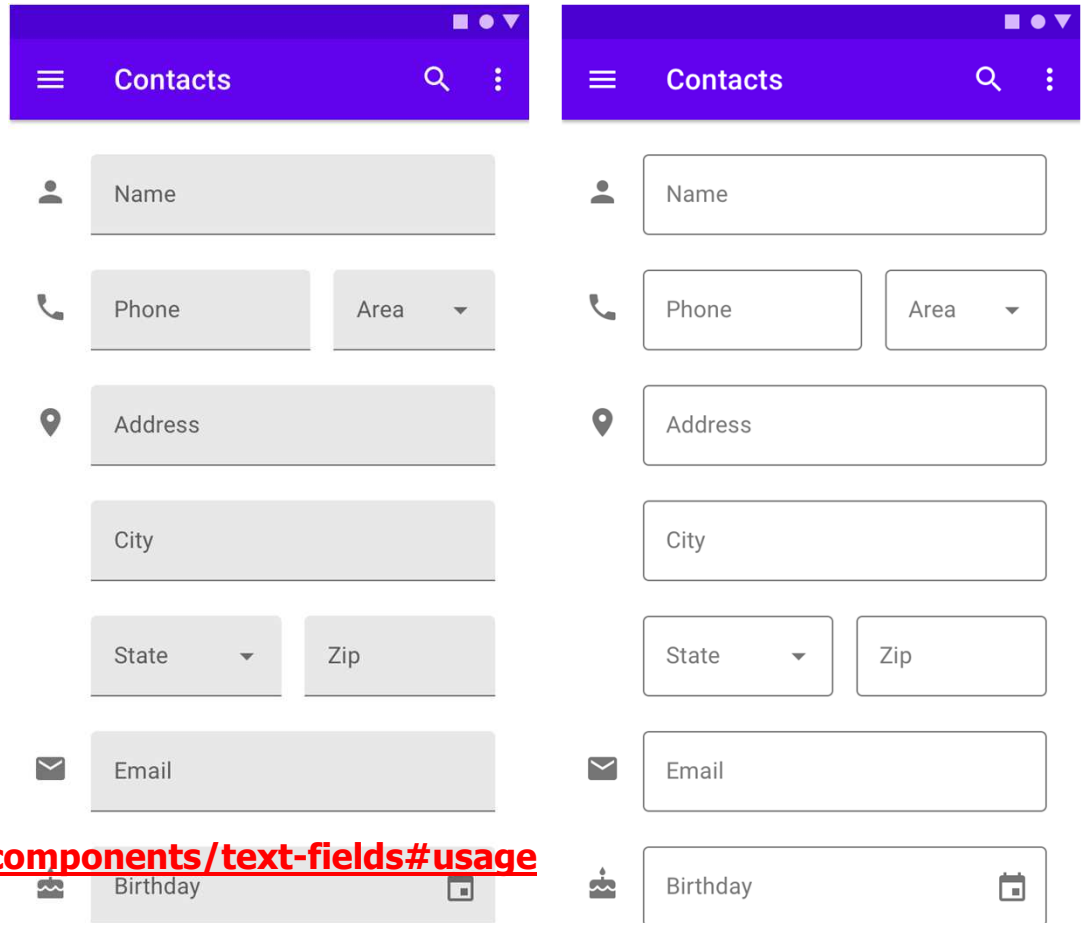
Text Fields

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

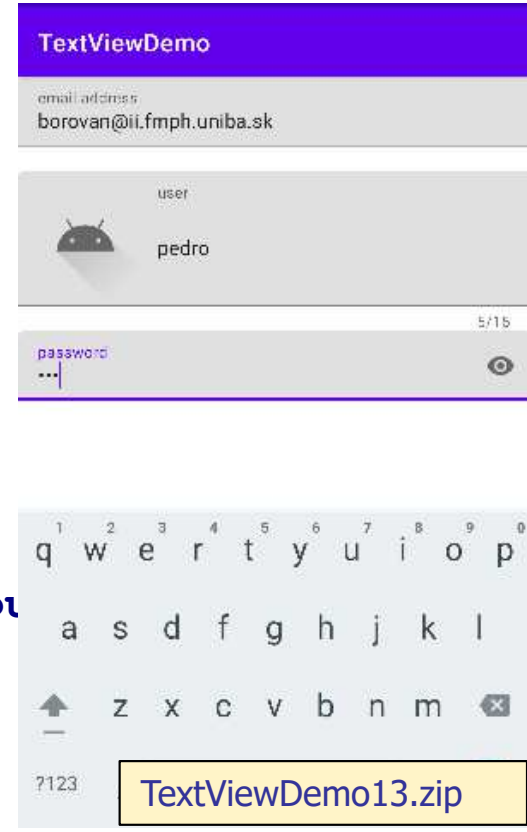


https://material.io/components/text-fields#usage

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

Chapter 12

12. Kotlin Data Types, Variables, and Nullability

Kotlin – pokračovanie

Cheat sheets

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

The billion-dollar mistake

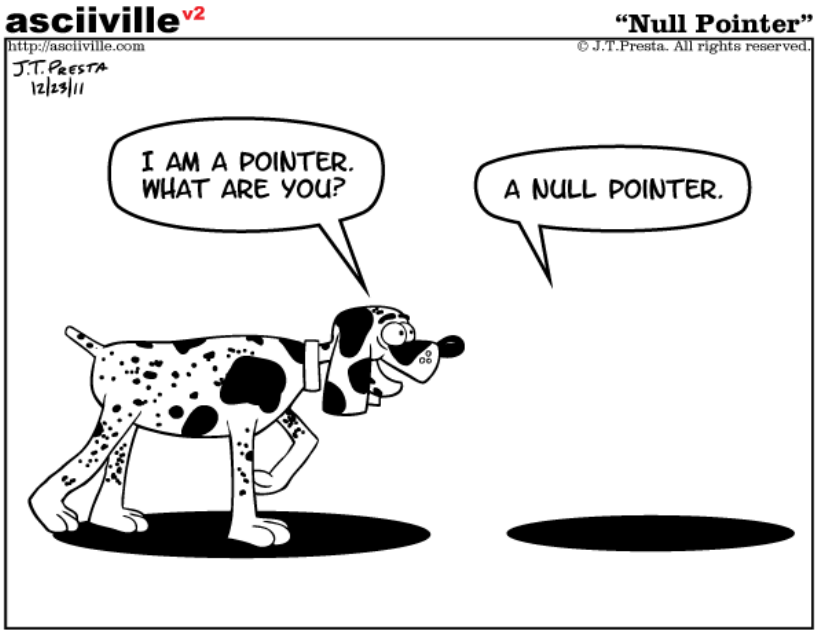
I call it my billion-dollar mistake. It was the invention of the **null** reference in 1965…This has led to innumerable errors, vulnerabilities, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years.

Sir Tony Hoare
FRS FREng

Tony Hoare in 2011

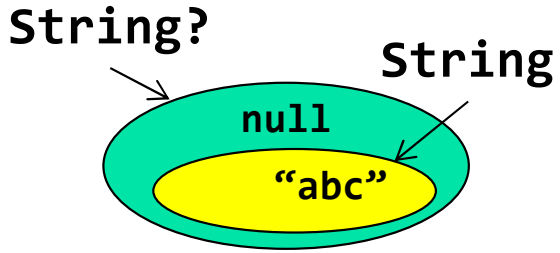
Born	Charles Antony Richard Hoare 11 January 1934 (age 85) Colombo, British Ceylon
Residence	Cambridge
Other names	C. A. R. Hoare
Alma mater	University of Oxford (BA) Moscow State University
Known for	Quicksort Quickselect Hoare logic Null reference Communicating Sequential Processes Structured programming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

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
}
```

Čo sú Scope functions

Marek Žitnik

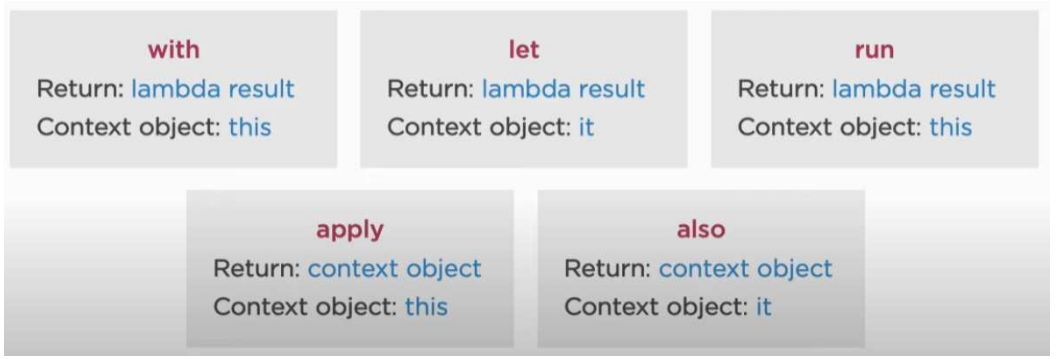
V skratke: Scope funkcie sú funkcie ktoré vám umožňujú spusitiť blok kódu v kontexte objektu. Scope funkcie vytvárajú dočasný „scope“, v ktorom môžte v objektu pristupovať cez **it** a **this**

Máme 5:

- **let**
- **run**
- **also**
- **apply**
- **with**

Sú medzi nimi 2 typy rozdielov a to :

- **čo vracajú**
- **v akom kontexte referujeme na objekt**



Čo to znamená a ako sa to používa

Najlepšie je to vidieť na príklade: Scope functions nám umožňujú spraviť z niečoho takéhoto

```
val cisla = mutableListOf("1" ,"11","100","101","110","111","1111")

val resturnList = cisla.map {it.length}.filter { it < 3 }.let {
    println(it)
}
```

→

Niečo takéto

```
val cisla = mutableListOf("1" ,"11","100","101","110","111","1111")

val resturnList = cisla.map {it.length}.filter { it < 3 }

println(resturnList)
```

Niečo podobné môžeme vidieť aj u

```
class s {
    var meno: String? = null
    var vek: Int? = null
}
val clovek1 = s()
clovek1.meno = "Jozko Mrkvicka"
clovek1.vek = 42
```

→

```
val clovek2 = s().apply{
    meno = "Jozko Mrkvicka"
    vek = 42
}
```

Príklad pre

```
class osoba {
    var meno: String = "Jozko Mrkvicka"
    var vek: Int = 42
}

val clovek : osoba? = osoba()

val bio = clovek?.run{
    println(meno)
    print(vek)
    vek + 2
    "Je to Jozko"
}
println(bio)
```

→

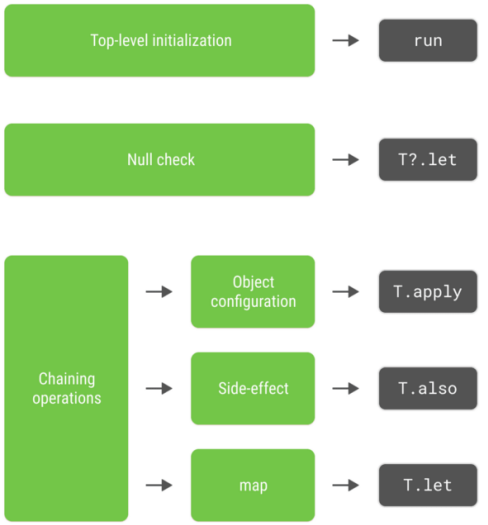
```
class osoba {
    var meno: String = "Jozko Mrkvicka"
    var vek: Int = 42
}

val clovek : osoba? = osoba()

val bio = clovek?.run{
    println(meno)
    print(vek)
    vek + 2
    "Je to Jozko"
}
println(bio)
```

Sumár

Marek Žitnik

Kedy použiť akú funkciu:

- **with :** keď potrebuješ operovať nad non-nullovím objektom
- **let : :** keď potrebuješ použiť iba lambda výraz na nullable objektom a vyhnúť sa NullPointerException
- **run :** keď potrebuješ operovať na nullable objekte, použiť lambda výraz vyhnúť sa NullPointerEception
- **apply :** keď potrebuješ inicializovať alebo konfigurovať objekt
- **also :** keď potrebuješ dodatočné objektové konfigurácie alebo operácie



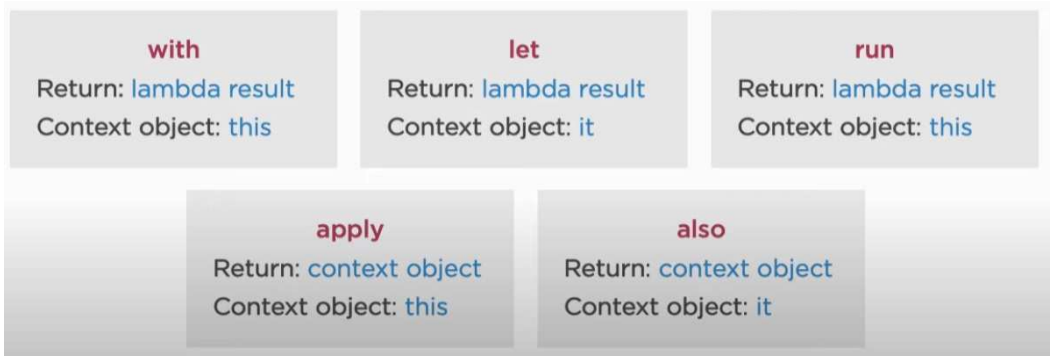

```
val myObj = run {
    val generator = DateStringGenerator(2008, 9, 23)
    generator.locale = "US"
    generator.localizedDate
}
```

```
class MyClass {

    fun printExceptionMessage(exception: Exception?) {
        exception?.let {
            println(exception.message)
        }
    }
```

```
    val lastWeeksDate: String = Calendar.getInstance().apply {
        add(Calendar.DAY_OF_YEAR, -7)
    }
```

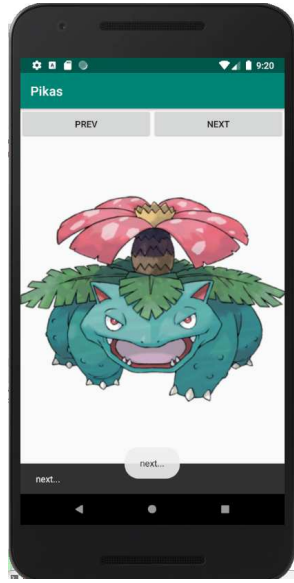
```
        .also {
            println("Created calendar of type: ${it.calendarType}")
        }
```

```
        .let { calendar ->
            val format = SimpleDateFormat("dd/M/yyyy hh:mm:ss", Locale.US)
            format.format(calendar.time)
        }
}
```

Pikas

(rekapitulácia)



activity entry point

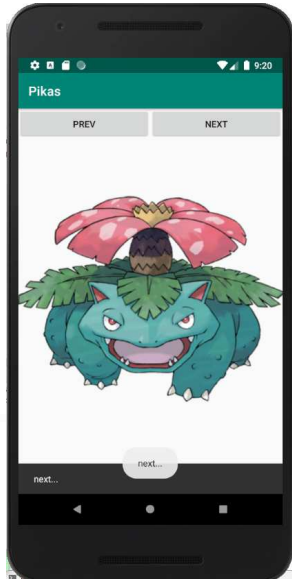
View(s)

```
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

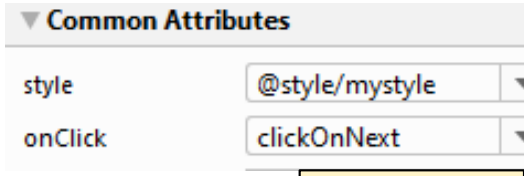
(stav sa mieša s views a logikou – riešenie príde)



const
final

State

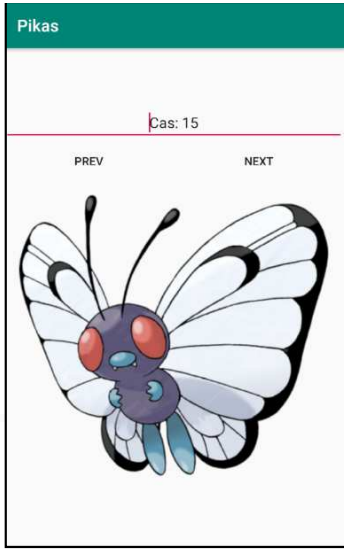
```
val TAG = "PIKAS"
var i = 0
var imgs = arrayOf<Drawable?>()
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)
    imgs = arrayOf(ContextCompat.getDrawable(applicationContext,
                                             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()
```

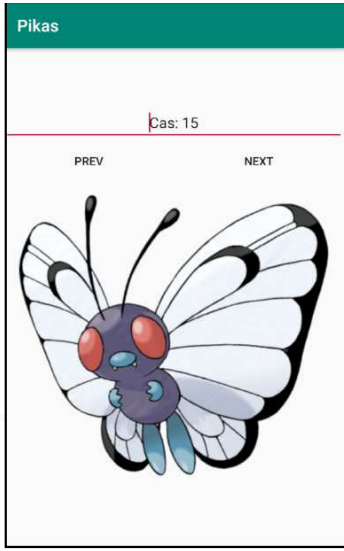
- nezabudnite na .run()
- **runOnUiThread**
  - má argument java.lang.Runnable, ktorý vykoná v hlavnom GUI vlákne

```
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)

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

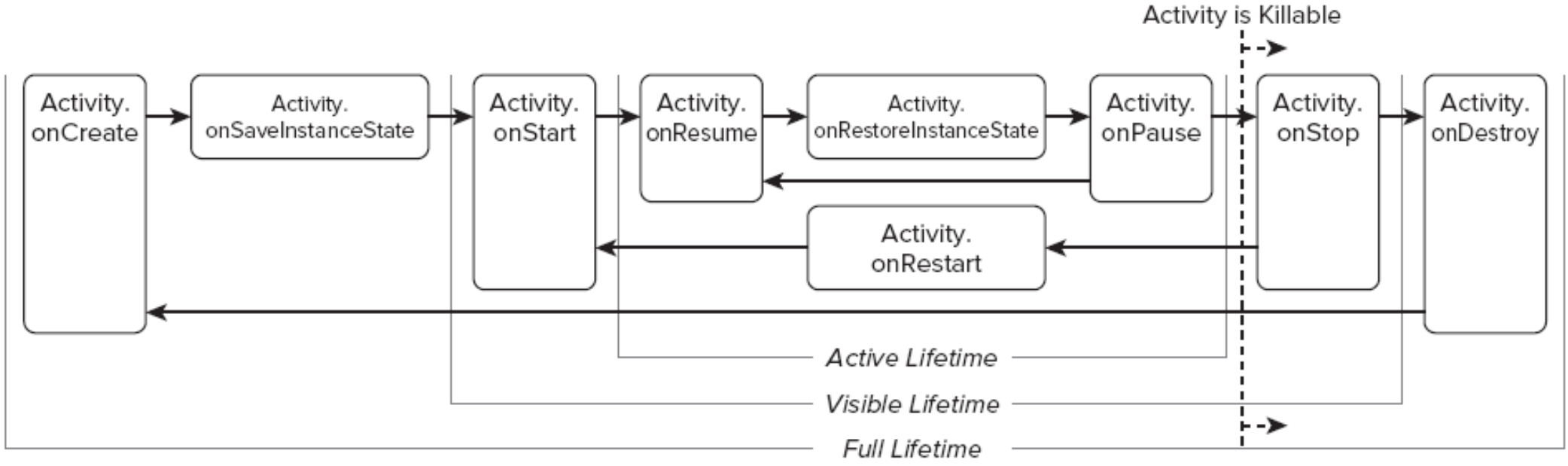
AppLifeCycle13.zip

21. Android Activity State Changes by Example

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
```

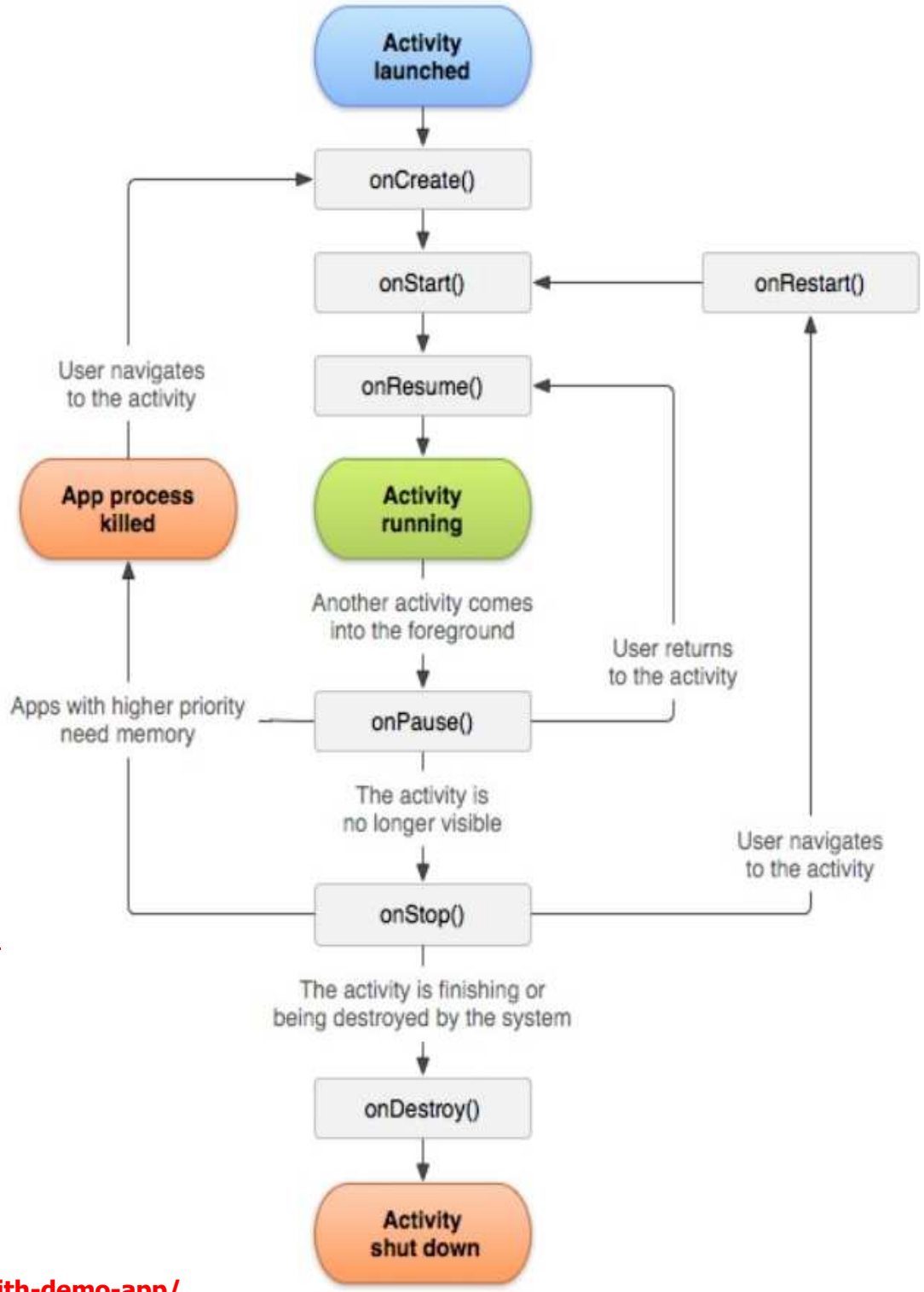


zdroj: Reto Meier: PA2AD

AppLifeCycle13.zip

Chapter 20

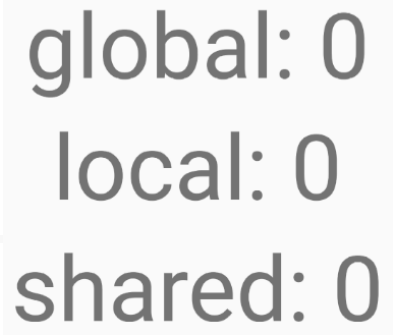
20. Handling Android Activity State Changes



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

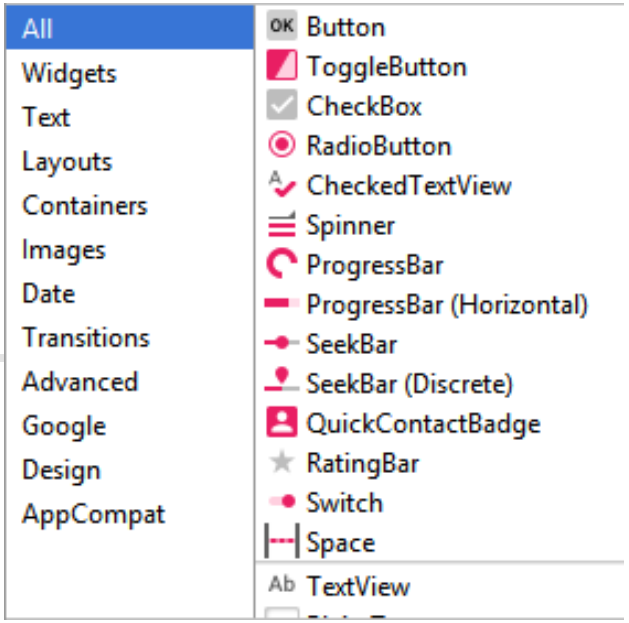
Čo je Kotlin ?

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

Chapter 23

23. Understanding Android Views, View Groups and Layouts

GUI komponenty



Layout

- LinearLayout (Vertical/Horizontal)
- RelativeLayout, ConstraintLayout

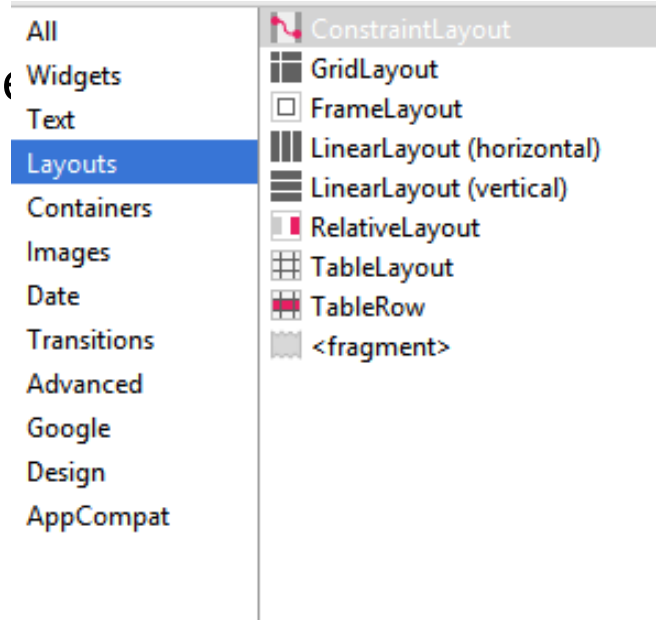
View, ViewGroup

- všetky viditeľné komponenty (widgets)

Activity - analógia Screenu (MIT), resp. Form/Frame

najznámejšie podtriedy

- ListActivity pre ListView, zobrazenie zoznamu
- MapActivity pre MapView (zobrazenie mapy)



Fragment (>= API level 11)

- reusable UI components

Layouts

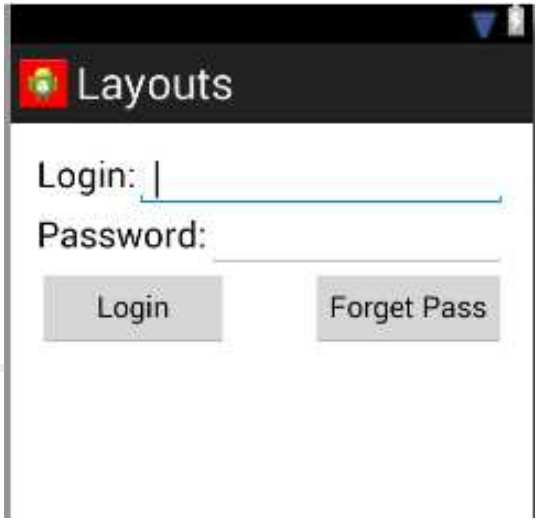
(match_parent, wrap_content)

- FrameLayout – objeky umiestni v ľavom hornom rohu
- LinearLayout – horizontálny/vertikálny
- RelativeLayout – dovolí umiestniť objekty relatívne k pozíciám iných objektov
- ConstraintLayout (support library, API 9, od Android Studio 2.2)
- GridLayout (od API Level 14)

```
<FrameLayout
        android:id="@+id/FrameLayout1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        <ImageView
          android:id="@+id/imageView1"
          android:layout_width="match_parent" --roztiahni podľa
          android:layout_height="match_parent" -- rodičovského
          android:src="@drawable/ic_launcher" />

</FrameLayout>
```

Kód na slajde je zjednodušený, originál nájdete v Layouts2.zip

LinearLayout

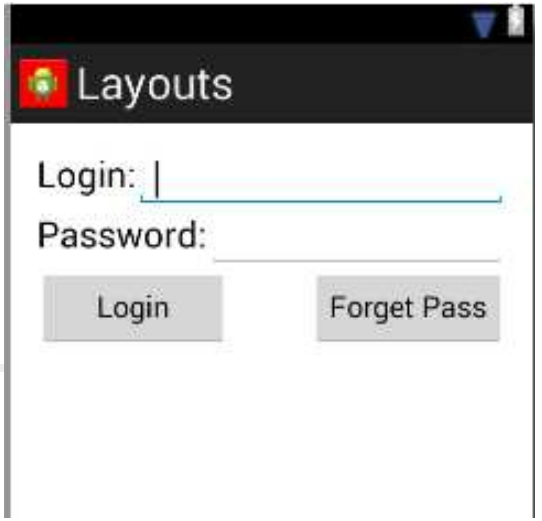



```
<LinearLayout
    android:orientation="vertical"
    <LinearLayout
       android:orientation="horizontal"
        <TextView
              android:id="@+id/lb1"
              android:text="@string/login"/>
       <EditText
              android:id="@+id/logintv"
              android:layout_width="match_parent" --roztiahni
              android:layout_height="wrap_content"-na výšku fontu
              android:inputType="textEmailAddress" /> -- filter
    </LinearLayout>
  ... podobne  pre password
```

Kód na slajde je zjednodušený, originál nájdete v Layouts2.zip

LinearLayout

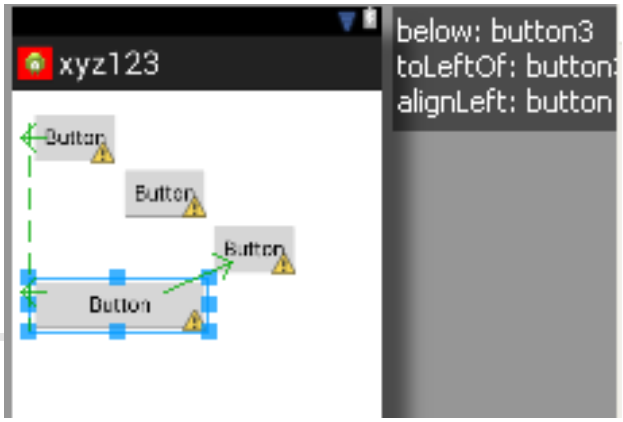
(weight, gravity, align with the base line)




```
<LinearLayout  ... Pokračovanie
        <LinearLayout
        android:orientation="horizontal"
        <Button
                android:id="@+id/logBtn"
                android:layout_weight="50"
                android:text="@string/Login"/>
        <Space
                android:layout_weight="50" />
        <Button
                android:id="@+id/forgetPass"
                android:layout_weight="50"
                android:text="@string/forget" />
        </LinearLayout>
</LinearLayout>
```

Kód na slajde je zjednodušený, originál nájdete v Layouts2.zip

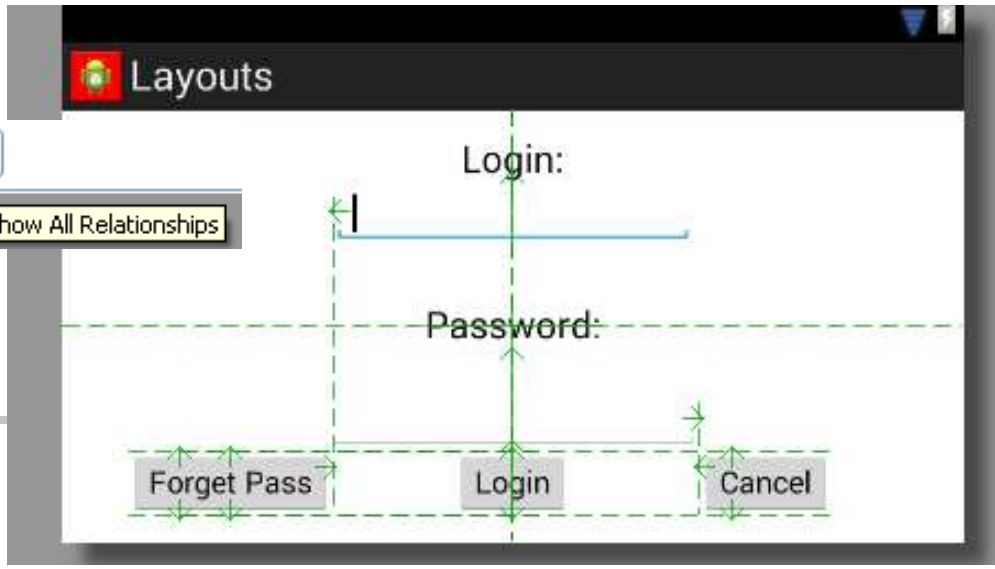
RelativeLayout




```
<RelativeLayout
    <Button
        android:id="@+id/button1"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"/>
    <Button
        android:id="@+id/button2"
        android:layout_below="@+id/button1"
        android:layout_toRightOf="@+id/button1"/>
... <Button
        android:id="@+id/button4"
        android:layout_alignLeft="@+id/button1"
        android:layout_below="@+id/button3"
        android:layout_toLeftOf="@+id/button3" />
</RelativeLayout>
```

Kód na slajde je zjednodušený, originál nájdete v Layouts2.zip

RelativeLayout




```
<RelativeLayout> ... skrátené...
 <EditText
          android:id="@+id/passwdtv"
          android:layout_below="@+id/pass"
          android:layout_centerHorizontal="true"/>
 <Button
          android:id="@+id/loginBtn"
          android:layout_below="@+id/passwdtv"
          android:text="@string/Login" />
 <Button
          android:id="@+id/forgetBtn"
          android:layout_alignBottom="@+id/loginBtn"
          android:layout_alignTop="@+id/loginBtn"
          android:layout_toLeftOf="@+id/passwdtv"
          android:text="@string/forget" />
```

originál nájdete v Layouts2.zip

GridLayout


```
<GridLayout
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:columnCount="4"
    android:rowCount="4">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="1"
        android:id="@+id/button1"
        android:layout_row="0"
        android:layout_column="0" />

    <Button ...
          android:layout_row="0"
        android:layout_column="1" />
```

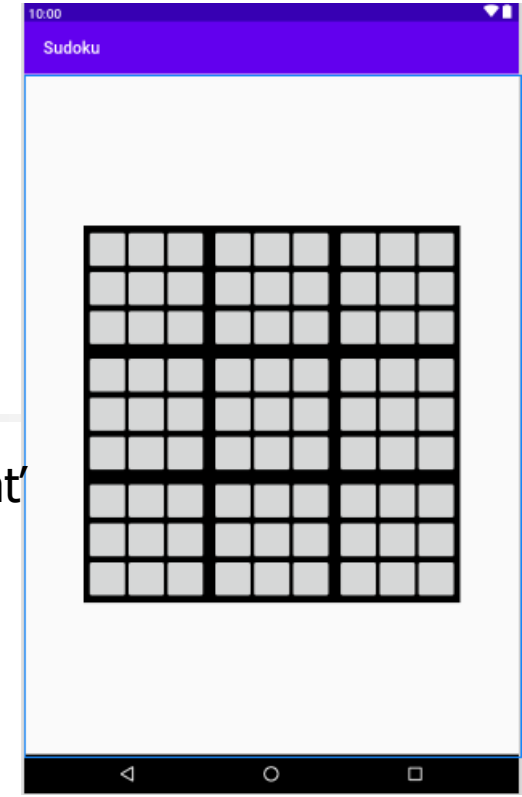
Kód na slajde je zjednodušený, originál nájdete v Layouts2.zip

Table vs. GridLayout


```
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <Button
         android:id="@+id/button7"
         android:text="Button“ />

    <TableRow
        . . .
```

```
<GridLayout
    android:id="@+id/grid"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:columnCount="4"
    android:rowCount="6">

    <TextView
        android:layout_columnSpan="4"
        android:layout_gravity="right"/>
```

Dynamický vs. statický layout



Layout môžete vyklikať (niekedy náročné) alebo naprogramovať

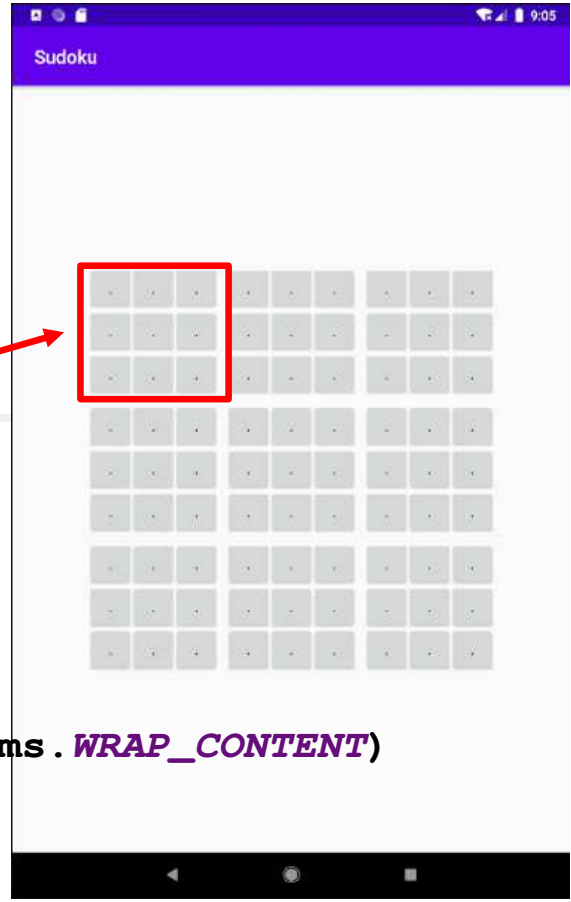
```
<GridLayout
    android:id="@+id/bigGrid"
    android:columnCount="3"
    android:rowCount="3">

    <GridLayout
        android:columnCount="3"
        android:rowCount="3">

        <Button
            android:id="@+id/button00"
            android:layout_width="@dimen/buttonSize"
            android:layout_height="@dimen/buttonSize"
            android:layout_margin="@dimen/buttonmargin"
            android:onClick="onClick"
            android:text=""/>

    </GridLayout>
</GridLayout>
```

9x

9x

activity_mainstatic.xml originál nájdete v Sudoku.zip

Dynamický vs. statický layout



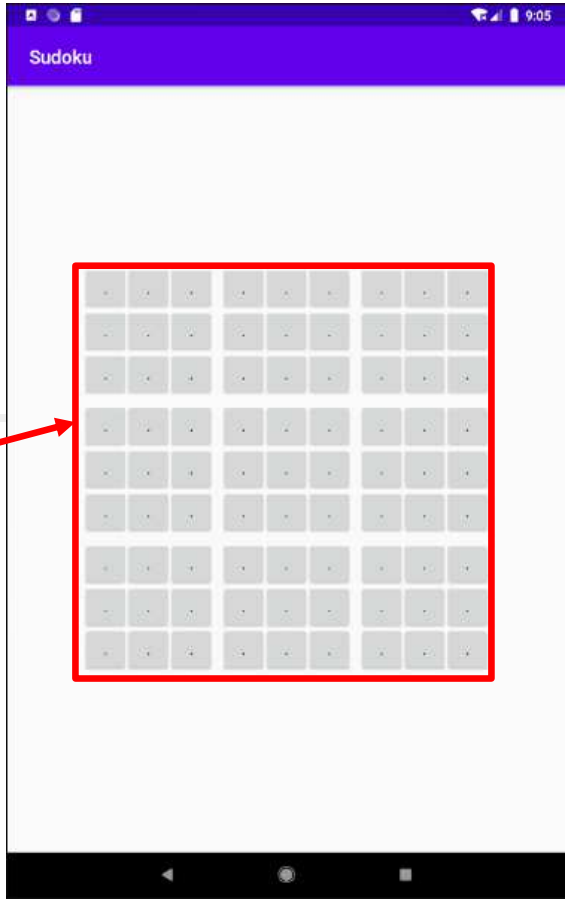

```
val smallGrid = GridLayout(this)
smallGrid.columnCount = SIZE
smallGrid.rowCount = SIZE
val smallGridParams = ViewGroup.MarginLayoutParams(

ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP_CONTENT)
smallGrid.layoutParams = smallGridParams

for (row in 0 until smallGrid.rowCount) {
    for (col in 0 until smallGrid.columnCount) {
        val button = Button(this)
        button.id = ... + 3*3*3*rowb + 3*3*row + 3*colb + col
        button.text = "."
        val buttonSize = resources.getDimension(R.dimen.buttonSize).toInt()
        val buttonParams = ViewGroup.MarginLayoutParams(buttonSize,buttonSize)
        button.layoutParams = buttonParams
        button.setOnClickListener { v -> onClick(v) }
        smallGrid.addView(button)
    }
}
bigGrid.addView(smallGrid)
```

```
fun onClick(v: View) {
    Log.d("SUDOKU", "clicked on ${v.id}")
}
```

originál nájdete v Sudoku.zip

Dynamický vs. statický layout

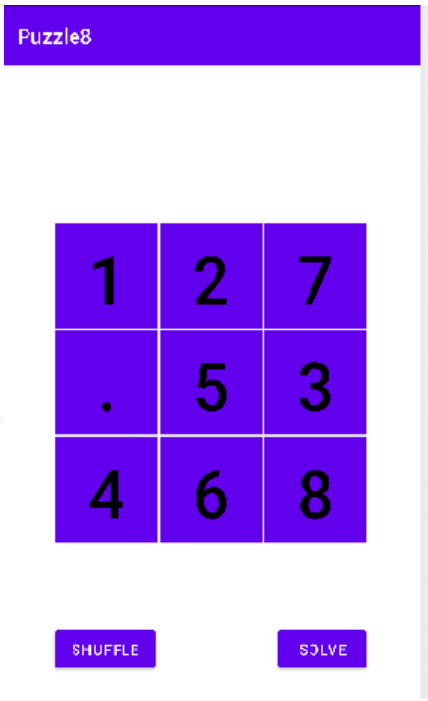
```
val SIZE = 3

val bigGrid = GridLayout(this)
bigGrid.columnCount = SIZE
bigGrid.rowCount = SIZE
bigGrid.layoutParams = ViewGroup.LayoutParams(
    ViewGroup.LayoutParams.WRAP_CONTENT,
    ViewGroup.LayoutParams.WRAP_CONTENT)
for (rowb in 0 until bigGrid.rowCount) {
    for (colb in 0 until bigGrid.columnCount) {
        val smallGrid = GridLayout(this)

        ... celý kód z predošlého slajdu

        bigGrid.addView(smallGrid)
    }
}
ll.addView(bigGrid)
```




```
<LinearLayout
    android:id="@+id/ll"
    android:orientation="horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>
```

activity_maindynamic.xml originál nájdete v Sudoku.zip

Puzzle 8



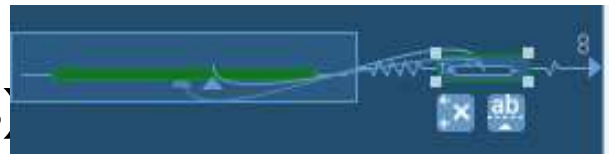

```
buttons = arrayOf(
    arrayOf(bindings.button00,bindings.button01,bindings.button02),
    arrayOf(bindings.button10,bindings.button11,bindings.button12),
    arrayOf(bindings.button20,bindings.button21,bindings.button22)
)
```

```
fun shuffleBtn(view: View) {
    bindings.shuffleBtn.isEnabled = false
    bindings.solveBtn.isEnabled = false

    object : CountDownTimer(10000, 100) {
        override fun onTick(p0: Long) {
          swap one pair
        }
        override fun onFinish() {
            bindings.shuffleBtn.isEnabled = true
            bindings.solveBtn.isEnabled = true
        }
    }.start()
}
```

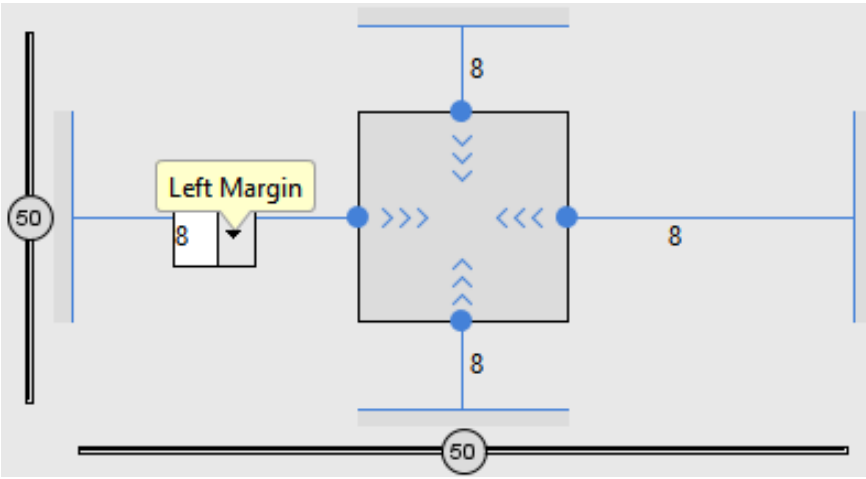
Constraint Layout

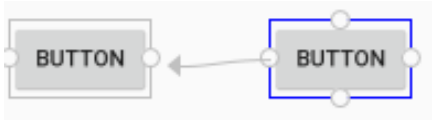
Je zovšeobecnením RelativeLayout

umožňuje nastaviť väzby, či obmedzenia (constraints)



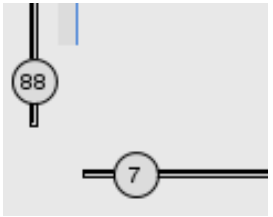
- relatívnu pozíciu
- spoločná baseline pre text
- okraje
- wrap/match content/fixná veľkosť
- vychýlenie (bias)

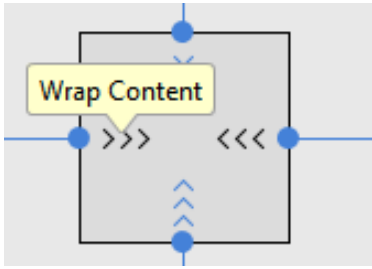




https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout

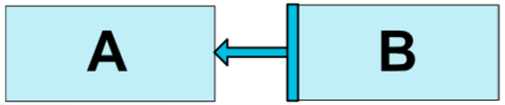
https://www.youtube.com/watch?v=z53Ed0ddxgM





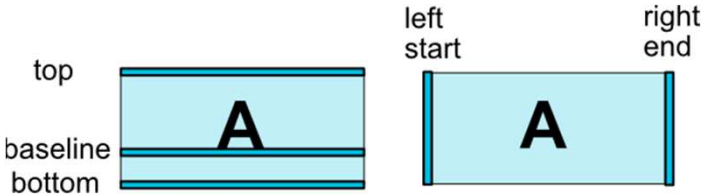
Kód na slajde je zjednodušený, originál nájdete v Layouts2.zip

Niektoré možnosti

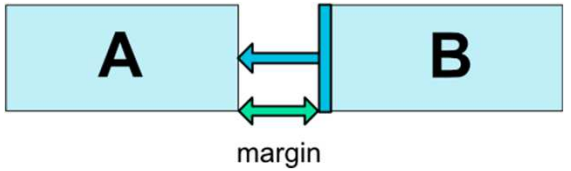



```
<Button android:id="@+id/buttonA" ... />
 <Button android:id="@+id/buttonB" ...
        app:layout_constraintLeft_toRightOf="@+id/buttonA" />
```

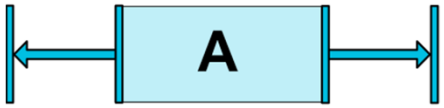
layout_constraintBaseline_toBaselineOf



android:layout_marginLeft



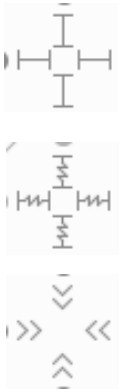
layout_constraintVertical_bias



- je ich ďaleko viac, ale zaujímavejšie je to v designeri

https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout

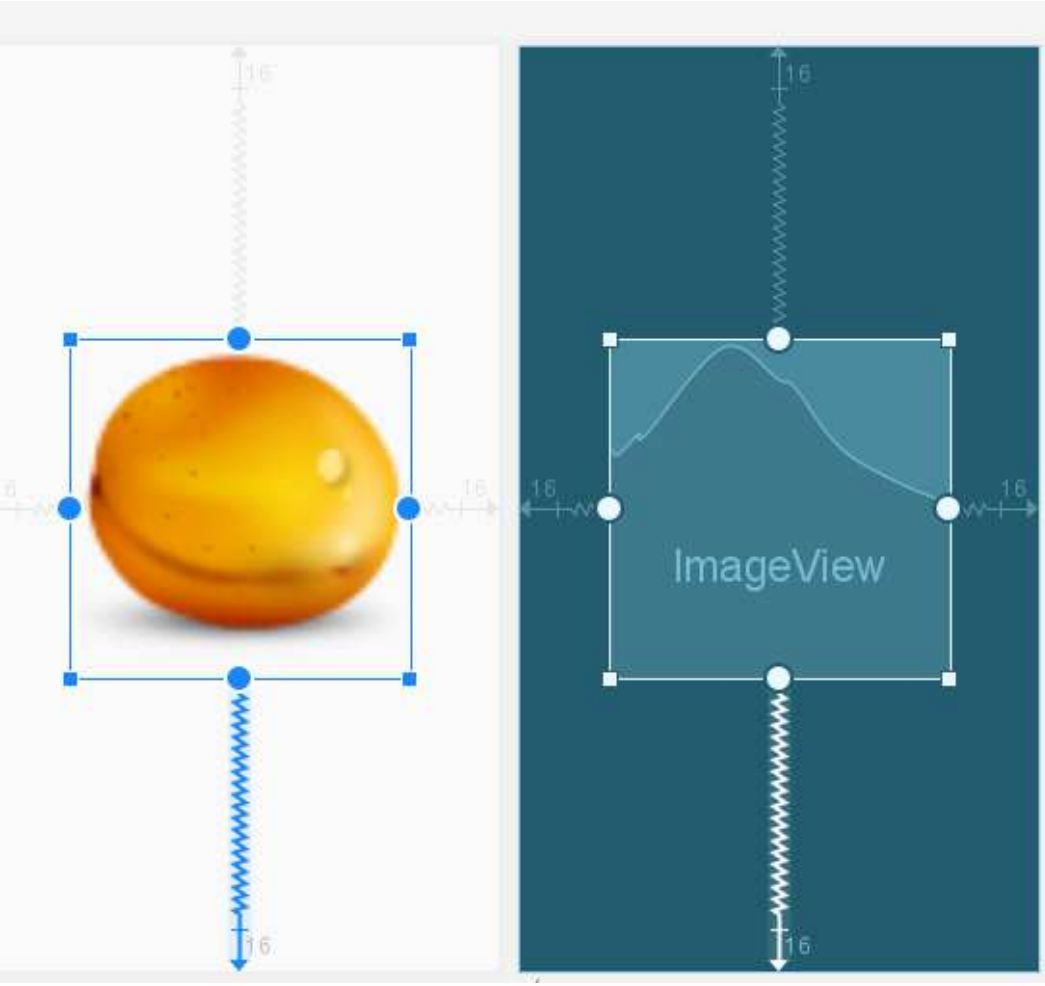
Constraint Layout

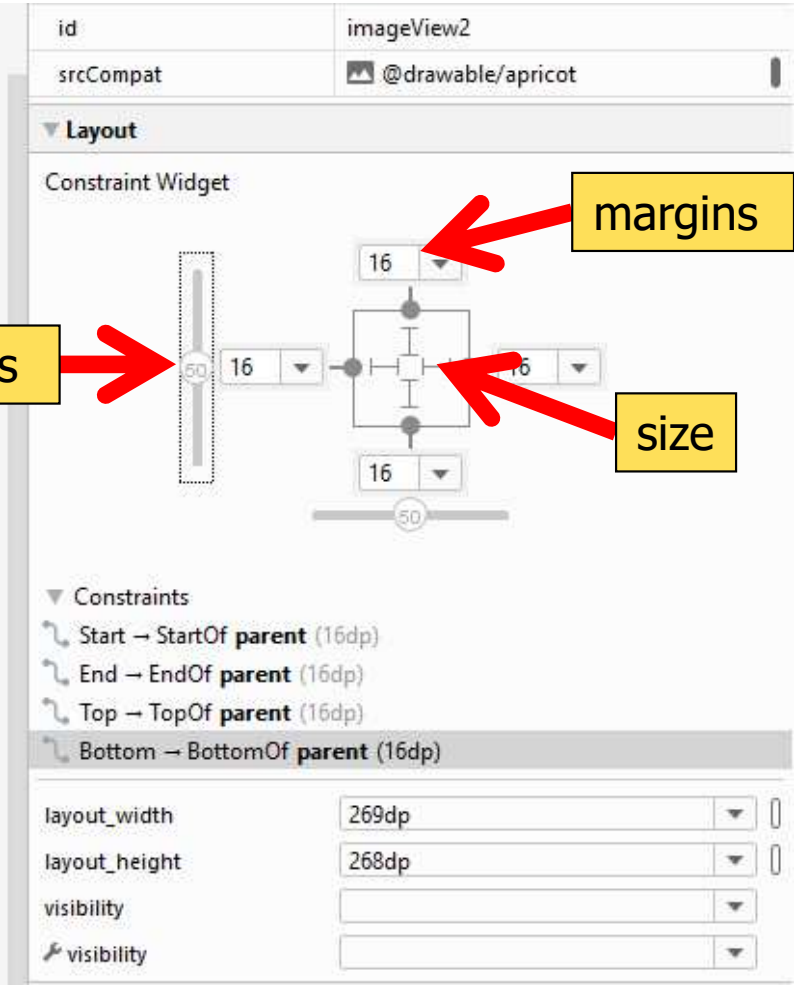


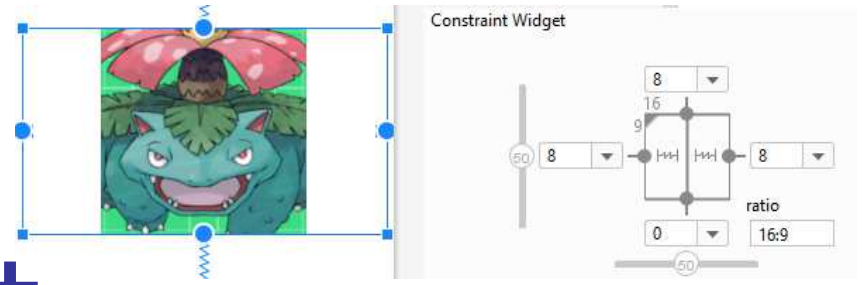
fixed size in dp ☹

match parent

wrap content

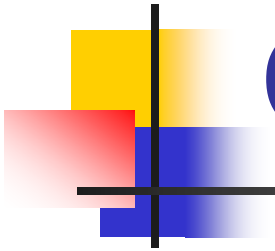






Constraint Layout

- Convert View
- BluePrint vs. Design móde
- Show constraints – zobrazí constraints v BluePrint resp. Design móde
- remove constraint (ctrl-)
- Horizontálne vertikálne constraints nemožno miešať
- Komponent musí mať aspoň jeden horizontálny a verikálny constraint
- Clear All Constraints – zmaže všetky
- Okraje - `layout_marginStart`
- Infer Constraints –
- `wrap_content`/`match_constraint`=`0dp` (deprecated `match_parent`)
- `layout_constraintWidth_min`, `layout_constraintWidth_max`
- base line
- ImageView – ratio - `layout_constraintDimensionRatio`
- `rotation[X,Y,Z]`, `scale[X,Y,Z]`, `translation[X,Y,Z]`

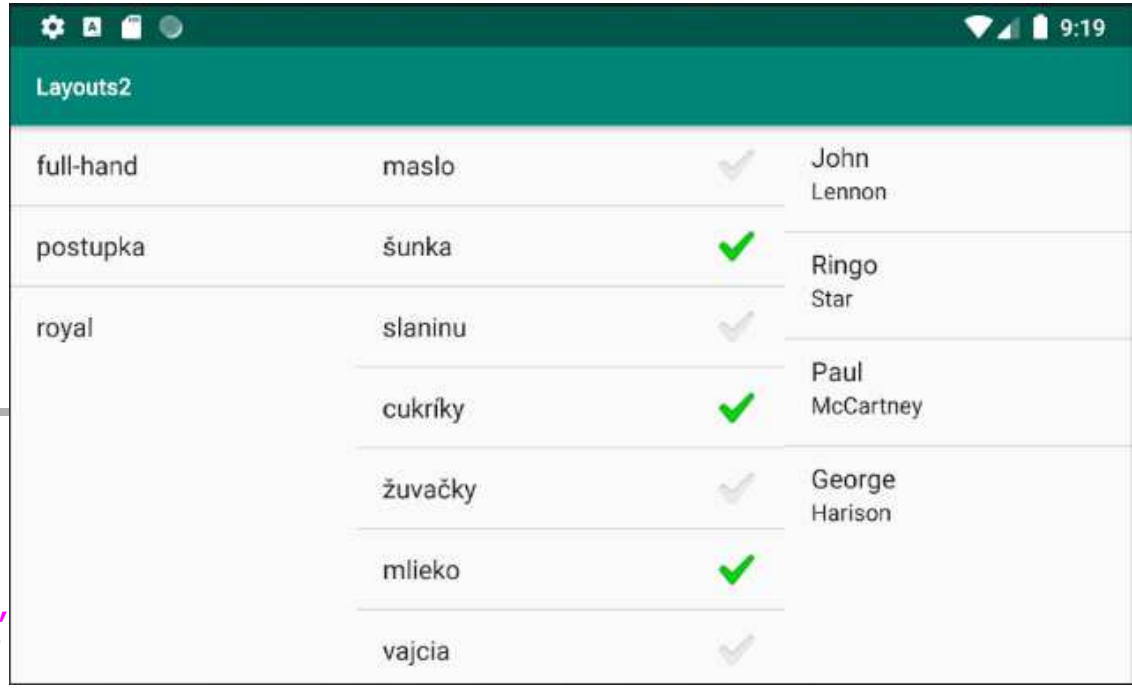


Constraint Layout

- zret'azenie – chain
- zarovnanie - alignment
- centrovanie
- guidelines
- Barriers
- Groups

ListView

(variabilita)



ListView a ListActivity zobrazujú zoznam položiek a môžu mať

- preddefinovaný štýl
  - môžu/nemusia sa nám páčiť
  - ale sú ready to use
- user defined
  - narobíme sa pri ich definícii

Rôzne inštancie ListView

simple_list_item_1,
simple_list_item_activated_1
simple_list_item_checked
simple_list_item_2
....

Odchytávanie udalostí v ListView

```
com.example.layouts2 D/ZOZNAM: beatles click: 2:{krstne=Paul, priezv=McCartney}
com.example.layouts2 D/ZOZNAM: beatles click: 1:{krstne=Ringo, priezv=Star}
com.example.layouts2 D/ZOZNAM: beatles click: 3:{krstne=George, priezv=Harison}
com.example.layouts2 D/ZOZNAM: check click: 3:cukríky
com.example.layouts2 D/ZOZNAM: check click: 4:žuvačky
com.example.layouts2 D/ZOZNAM: item click: 1:postupka
com.example.layouts2 D/ZOZNAM: item click: 2:royal
com.example.layouts2 D/ZOZNAM: item click: 0:full-hand
com.example.layouts2 D/ZOZNAM: check click: 2:slaninu
```

Project: Layouts2.zip

ListView
(simple_list_item_1)

```
// poker - simple_list_item1 view
listView1.adapter = ArrayAdapter<String>(
    this,
    android.R.layout.simple_list_item_1, // jednoriadkový
       // simple_list_item_activated_1
    resources.getStringArray(R.array.poker) // hodnoty
)

// listView1.choiceMode = ListView.CHOICE_MODE_MULTIPLE


listView1.setOnItemClickListener {
   adapterView, view, index, l -> // View.OnItemClickListener
      val hodnota = adapterView.getItemAtPosition(index)
      Log.d(TAG, "item click: $index:$hodnota")
}
```

Project: Layouts2.zip

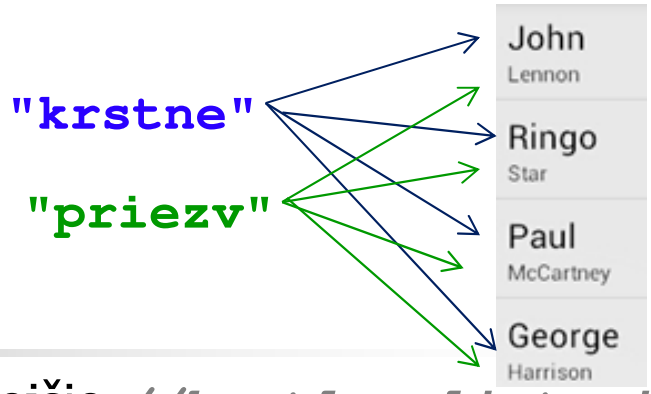
ListView

(simple_list_item_checked)

```
// nákup - checked box list view
listView2.adapter = ArrayAdapter<String>(
     this,
     android.R.layout.simple_list_item_checked, //2riadkový
     resources.getStringArray(R.array.nakup)
)

listView2.setOnItemClickListener {
     adapterView, view, index, l ->
        val hodnota = adapterView.getItemAtPosition(index)
        (view as CheckedTextView).toggle() // prekresli
        Log.d(TAG, "check click: $index:$hodnota")
}
```

Project: Layouts2.zip

ListView 2
(simple_list_item_2)



Naplniť iný, napr. dvojriadkový ListView je náročnejšie *//beatles list view*

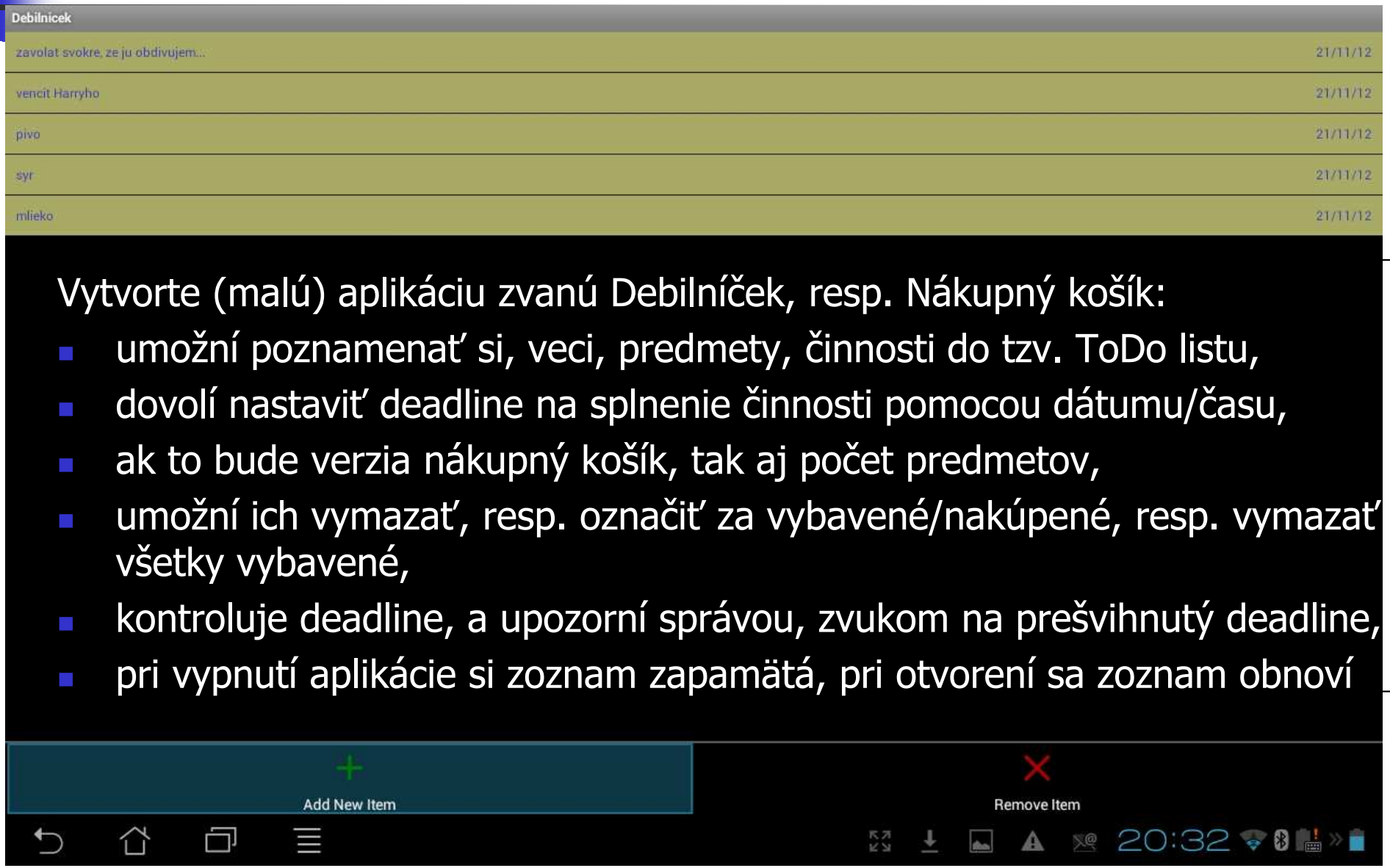
```
val pairs = listOf( // hodnoty sú zoznam máp kľúč->hodnota
    mapOf("krstne" to "John","priezv" to "Lennon"),mapOf("krstne" to "Ringo","priezv" to "Star"),
    mapOf("krstne" to "Paul","priezv" to "McCartney"),mapOf("krstne" to "George","priezv" to "Harison")
)
listView3.adapter = SimpleAdapter(this,
      pairs,                                    // hodnoty
      android.R.layout.simple_list_item_2, // format ListView
      arrayOf("krstne", "priezv"),          // zoznam kľúčov
      arrayOf(android.R.id.text1,android.R.id.text2) // riadky
         .toIntArray()
   )
   listView3.setOnItemClickListener {
         adapterView, view, index, l ->
        val hodnota = adapterView.getItemAtPosition(index)
        Log.d(TAG, "beatles click: $index:$hodnota:"+
                  "${(hodnota as Map<String, String>)["krstne"]
}
```

Rôzne preddefinované ListView
(prehľad)

zdroj: https://developer.xamarin.com/guides/android/user_interface/working_with_listviews_and_adapters/part_3_-_customizing_a_listview's_appearance/

Domáca úloha 2

(jedna z 3 alternatív, na budúcu prednášku bude Menus)



Vytvorte (malú) aplikáciu zvanú Debilníček, resp. Nákupný košík:

- umožní poznamenať si, veci, predmety, činnosti do tzv. ToDo listu,
- dovolí nastaviť deadline na splnenie činnosti pomocou dátumu/času,
- ak to bude verzia nákupný košík, tak aj počet predmetov,
- umožní ich vymazať, resp. označiť za vybavené/nakúpené, resp. vymazať všetky vybavené,
- kontroluje deadline, a upozorní správou, zvukom na prešvihnutý deadline,
- pri vypnutí aplikácie si zoznam zapamätá, pri otvorení sa zoznam obnoví