

# Hitparáda aktivít

Aktivity, View  
Intent, Layout



Peter Borovanský  
KAI, I-18

borovan 'at' ii.fmph.uniba.sk



# Hitparáda

(Hall of Fame)

1.	Linda	Jurkasová	13
2.	Richard	Mészáros	12.999
3.	Zuzana	Hlávková	12



Kalkulačka / Feedback / Hra15 / Apple  
Watch Tips / Shared Canvas

Pidi-Kalkulačka, 22 riešení

Feedback, 22 riešení

MatejM, LindaJ, TomášK, IvanaN, AndreaS

Apple Watch Tips, 18 riešení

outsourcované J+L

Hra 15, 7 riešení

DominikaK, LindaJ, ZuzkaH

Shared Canvas, 4 riešenia

rád sa BT-spárujem, Iveta+Nikolaj,



# Príklad jednoduchkej aplikácie

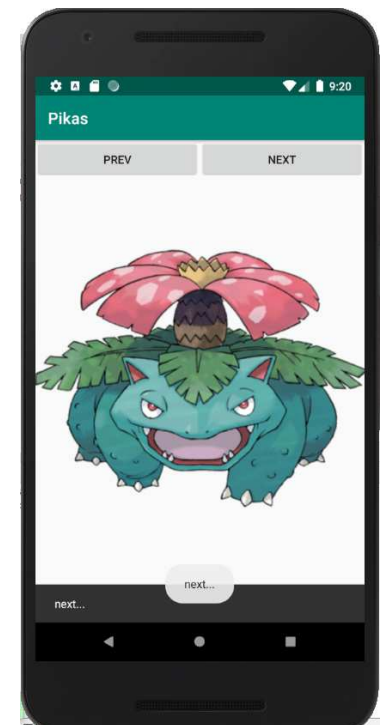
(ktorú sme si vyklikali minule)

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** (zjednodušená aktivita, v ktorej sa toastuje)
- Snackbar – to chce pridať závislosť do build.gradle a import snackbaru

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

```
prevBtn2.setOnClickListener({  
    → Toast.makeText(this, "prev...", Toast.LENGTH_SHORT).show()  
    → Log.d(TAG, "prev...")  
    → Snackbar.make(it, "next...",  
        Snackbar.LENGTH_SHORT).setAction("Action", null).show()  
    alebo .setAction(R.string.action,  
        View.OnClickListener { nextOnClickListener(it) }).show()  
    ...  
})
```

# Pikas

(rekapitulácia)

activity entry point

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

View(s)

logovanie





const  
final



```
var i = 0
```

```
var imgs = arrayOf<Drawable?>()
```

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

## super.onCreate (savedInstanceState)

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

[illegible]

```
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.setImageResource(imgs[i])
```

}



State



**▼ Common Attributes**

style	<input type="text" value="@style/mystyle"/>	▼	⋮
onClick	<input type="text" value="clickOnNext"/>	▼	⋮

@style/mystyle

clickOnNext

Project:Pikas2.zip

# Pikas

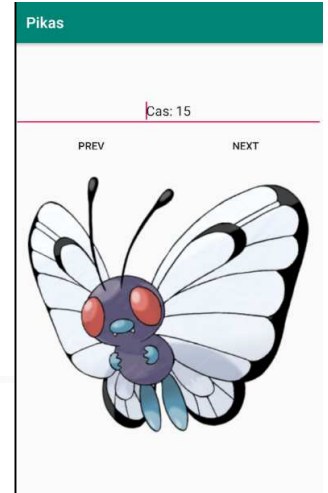
(asynchrónnosť - timer)

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

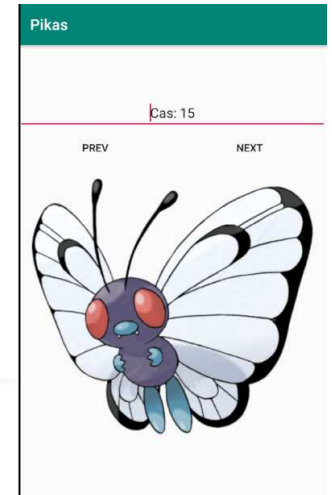
```
zabitie timera:
override fun onPause() {
    super.onPause()
    timer.cancel()
}
```



# Pikas

(asynchrónnosť – count down)

pomocou `android.os.CountDownTimer`



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

tik



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

game  
over



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



ukončenie appky

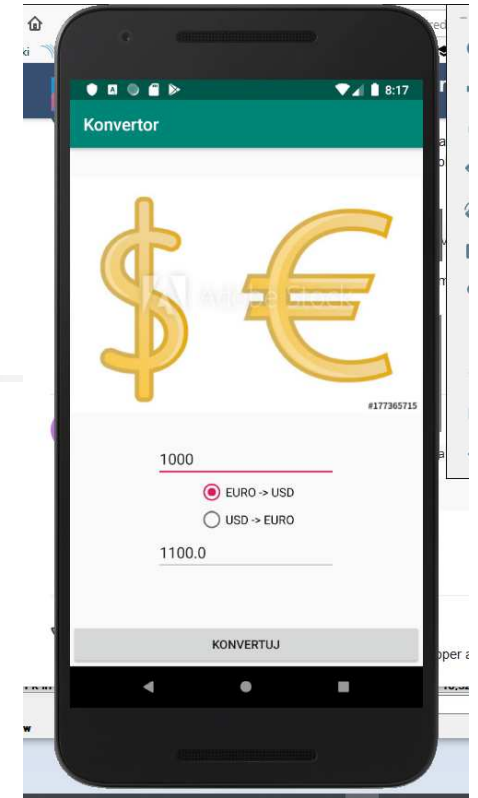


# Konvertor EURO USD

(logika)

Jednoduchá aplikácia na konverziu kurzov USD EURO

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



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



Klik na Konvertuj

# 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

*// 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)}")
    }
}
```

metóda  
Float

→ 

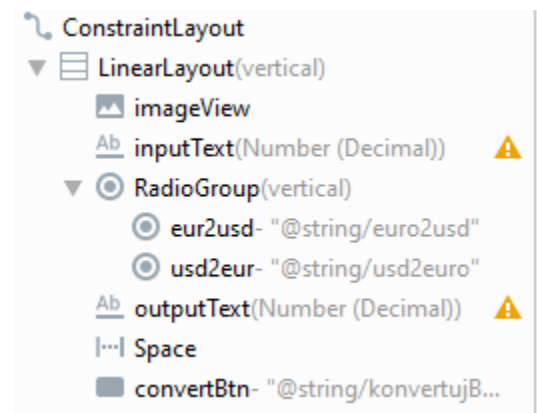
```
fun Float.format(digits: Int) =
    java.lang.String.format("%.${digits}f", this)
```

Project:Konvertor.zip

# Konvertor EURO USD

(layout)

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



# Životný cyklus apky

(prvý – zjednodušený nástrel)

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

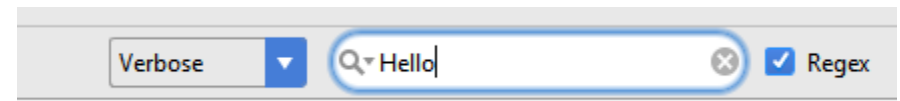
@Override

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.d("CYKLUS", "onCreate"); // LOGUJTE, LOGUJTE, LOGUJTE
}
```

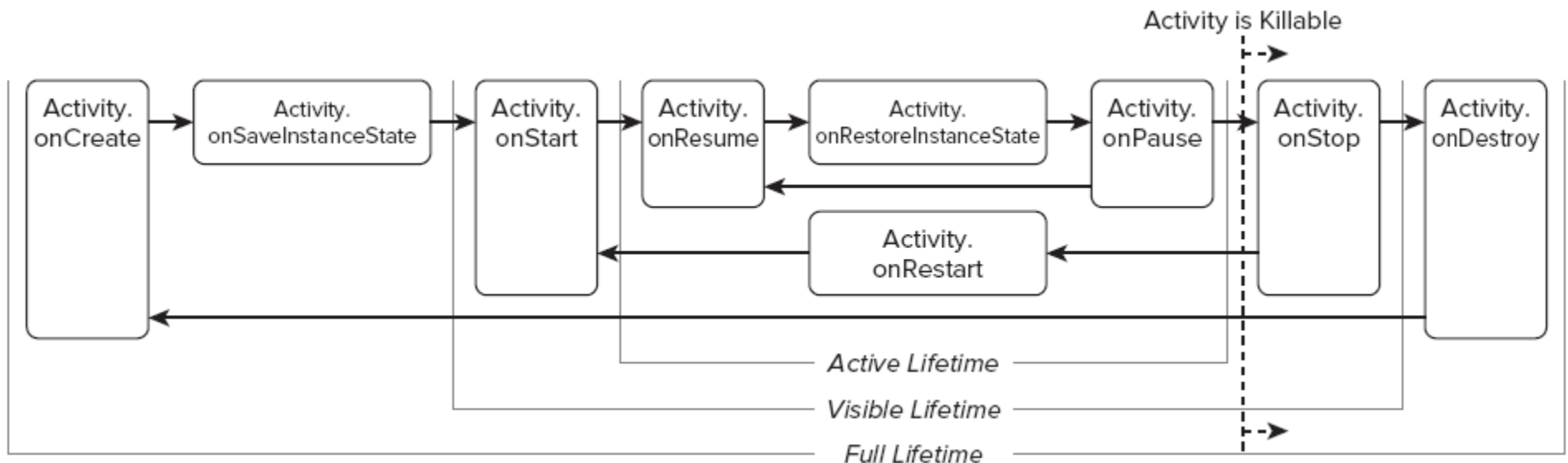
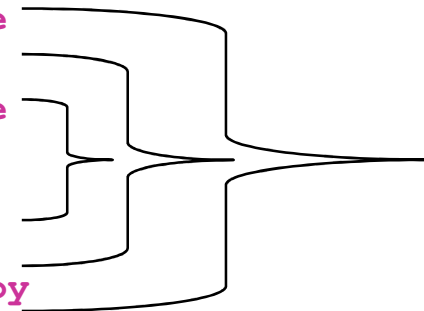
tag vhodný na filtrovanie

# LogCat

(Filtrovane logov)



- 10-13 12:55:41.091: D/Hello(405): onCreate
- 10-13 12:55:41.091: D/Hello(405): onStart
- 10-13 12:55:41.100: D/Hello(405): onResume
- kill
- 10-13 12:56:45.061: D/Hello(405): onPause
- 10-13 12:56:45.681: D/Hello(405): onStop
- 10-13 12:56:45.681: D/Hello(405): onDestroy





# Persistencia

(prvý dotyk)

global: 0  
local: 0  
shared: 0

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

Zistíte, že:

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



# Bundle?

---

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

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

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



# SharedPreferences

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

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



# Kotlin

## 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.

Kotlin Null Safety

### Sir Tony Hoare

FRS FREng



Tony Hoare in 2011

<b>Born</b>	Charles Antony Richard Hoare 11 January 1934 (age 85) Colombo, British Ceylon
<b>Residence</b>	Cambridge
<b>Other names</b>	C. A. R. Hoare
<b>Alma mater</b>	University of Oxford (BA) Moscow State University
<b>Known for</b>	Quicksort Quickselect Hoare logic Null reference Communicating Sequential Processes Structured programming
<b>Awards</b>	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 Java, resp.
- Option v Scala, resp. kde inde

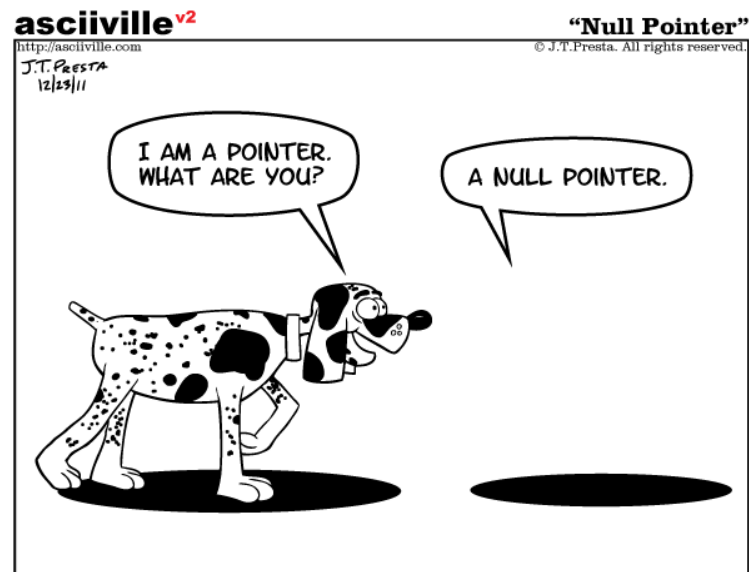
Napr. `String?` je typ pre reťazec alebo null

Ale `String` je typ len pre SKUTOČNÝ REŤAZEC, not-null

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

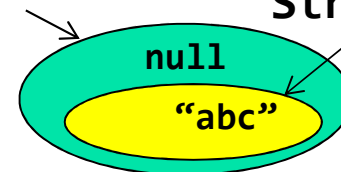
Ak ste skalo-pevne presvedčený, že hodnota `a:String?` `!= null`, môžete opatrne použiť BANG-BANG (`!!`) operátor a oklamať type-checker  
`val b:String = a!!`

Ak ale neviete, či `a:String?` `== null`, tak použijete tzv. **Elvis operátor**  
`val c:String = a?:"default, ak je prázdny reťazec"`



`String?`

`String`






# 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 Kotlinе String je **LEN skutočný reťazec** a null nepatrí do typu String

Existuje String? čo je String alebo null, vo všeobecnosti:  $T? = T \cup \text{null}$

$T?$  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.java

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

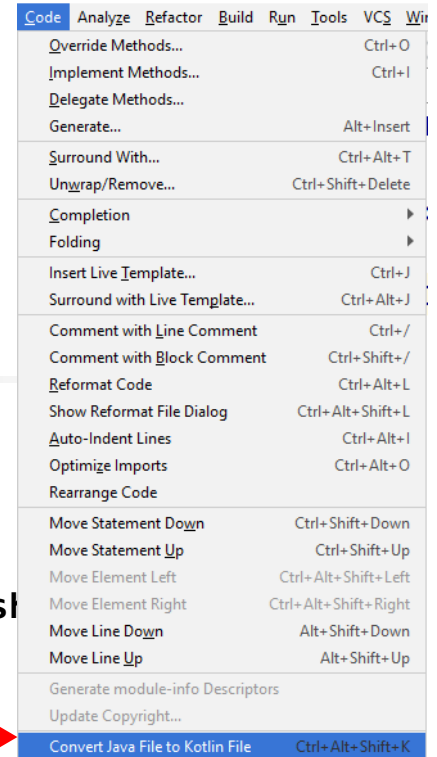
```
i = 0
iv.setImageDrawable(images[i])

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

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

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

v java  
projekte  
nájdete



Project:Pikas.zip



# Konverzie Java <-> Kotlin

---

- **Java -> Kotlin**

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

- **Kotlin -> JVM Byte code**

Tools/Kotlin/Show Byte Code

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

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



# Čo je Kotlin ?

Kotlin is the New Official Language of Android



Android

+



Kotlin



Kotlin Island

3



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



# GUI komponenty

## Layout

- LinearLayout (Verical/Horizontal)
- RelativeLayout, ConstraintLayout

## View, ViewGroup

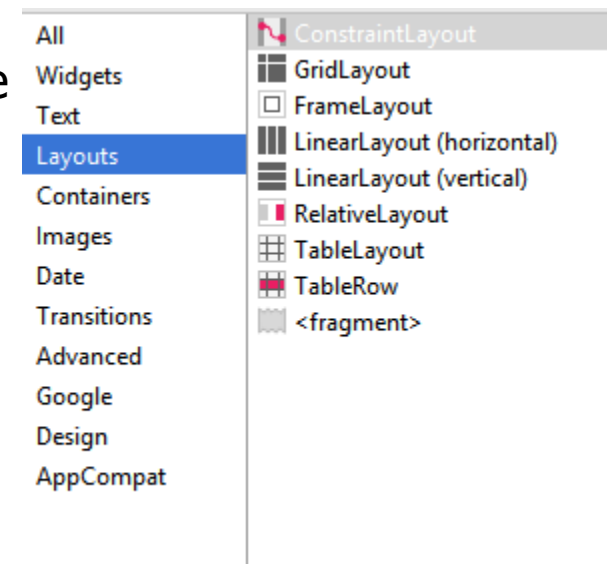
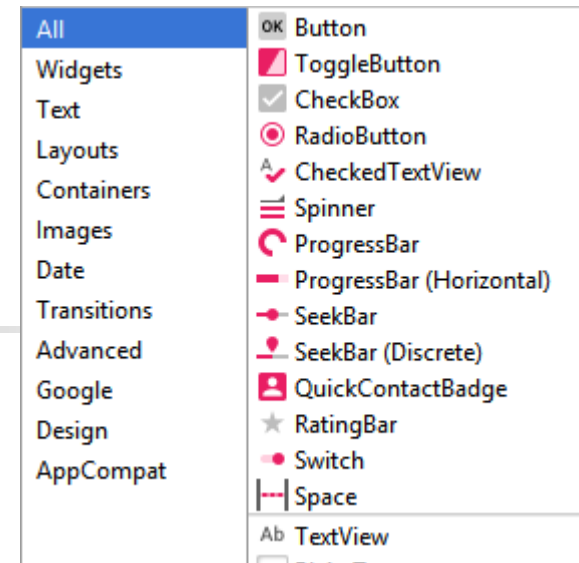
- všetky viditeľné komponenty (widgets)

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

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


Fragment ( $\geq$  API level 11)

- reusable UI components





# Layouts

- FrameLayout – objekty umiestni v ľavom hornom rohu
- LinearLayout – horizontálny/vertikálny 
- RelativeLayout – dovoľí 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"
```

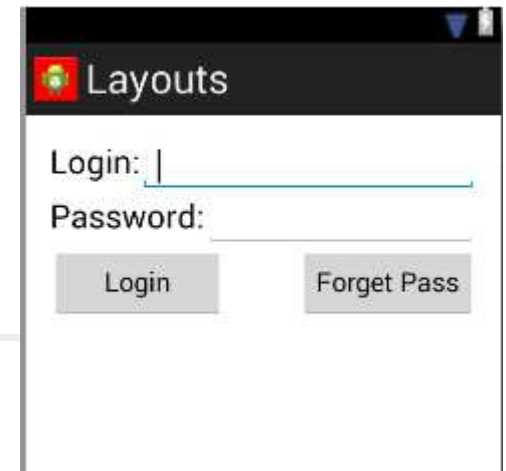
```
            android:layout_width="match_parent" --roztiahni
```

```
            android:layout_height="wrap_content" --na výšku fontu
```

```
            android:inputType="textEmailAddress" /> -- filter
```

```
    </LinearLayout>
```

... podobne pre password



# LinearLayout

<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

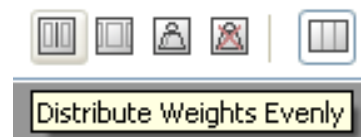
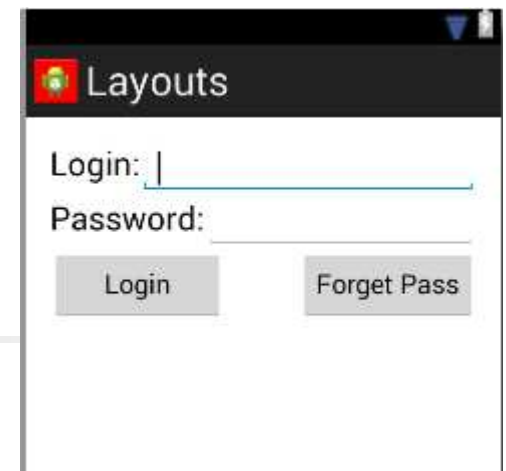
android:id="@+id/forgetPass"

android:layout\_weight="50"

android:text="@string/forget" />

</LinearLayout>

</LinearLayout>



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

# 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" />
```



# 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
```

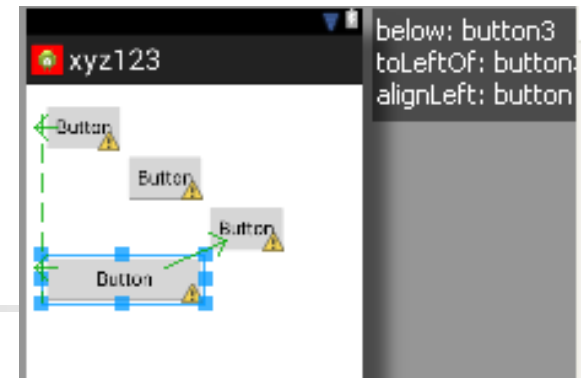
```
        android:id="@+id/button4"
```

```
        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> ... skrátene...

<EditText

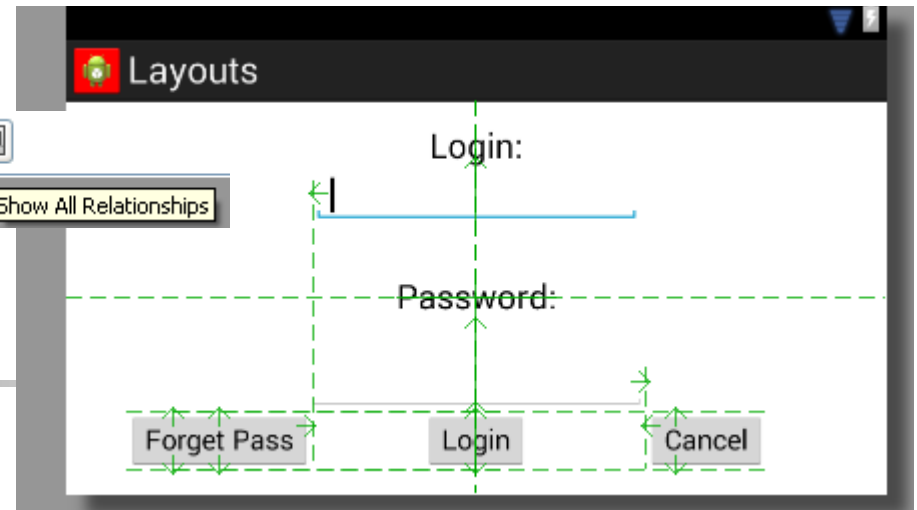
```
    android:id="@+id/passwdtv"  
    android:layout_below="@+id/pass"  
    android:layout_centerHorizontal="true"/>
```

<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" />
```

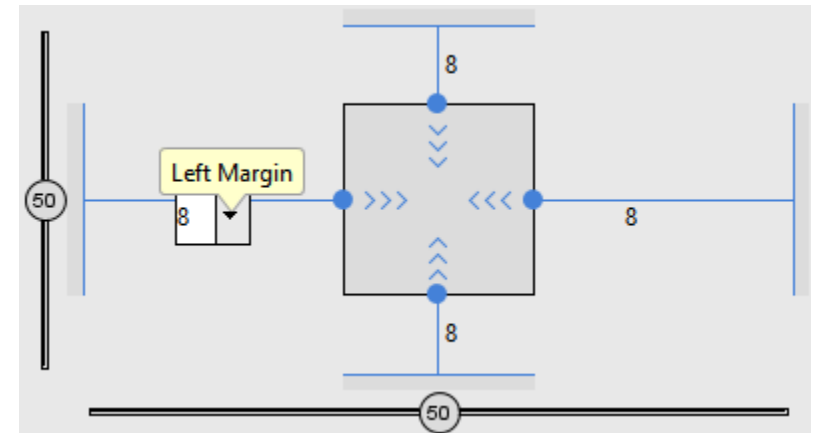
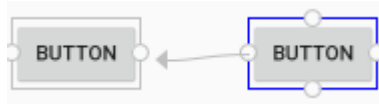


originál nájdete v Layouts2.zip

# Constraint Layout

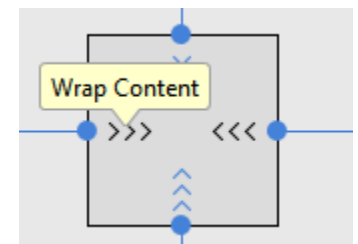
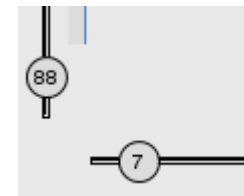
Umožňuje nastaviť väzby

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



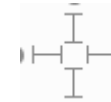
<https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html>

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

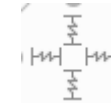


Kód na slajde je zjednodušený, originál nájdete v Layouts2.zip

# Constraint Layout



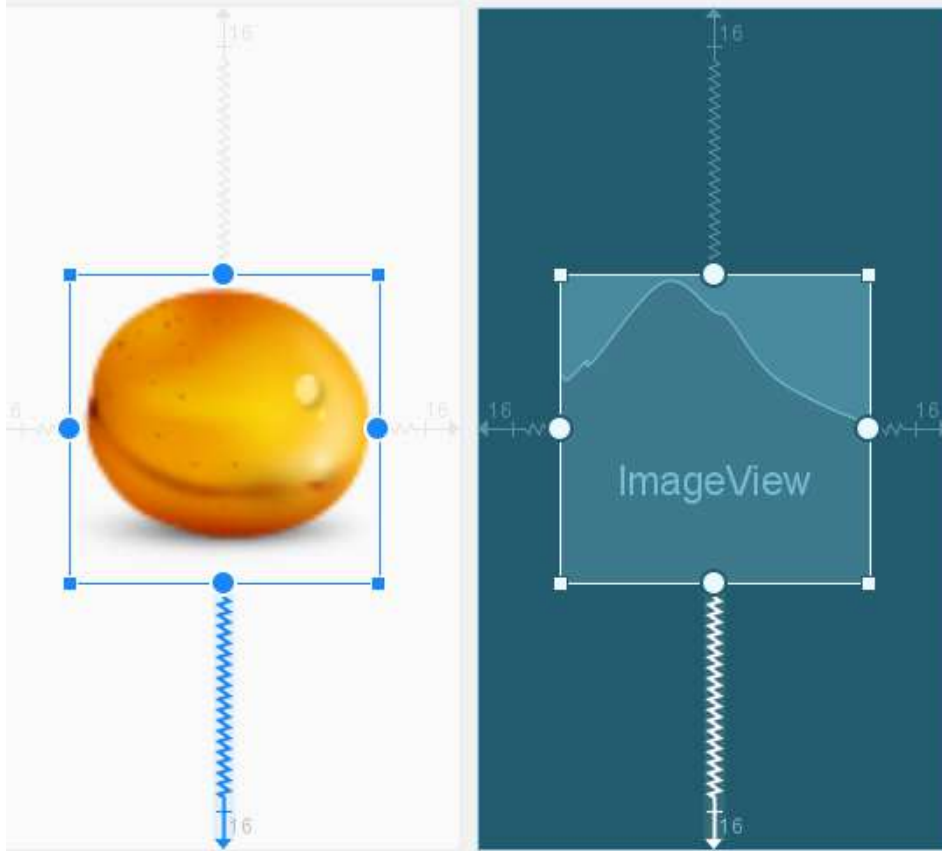
fixed size in dp ☹️



match parent



wrap content



weights

margins

size

id: imageView2  
srcCompat: @drawable/apricot

Layout

Constraint Widget

Weights: 50

Constraints:

- Start → StartOf parent (16dp)
- End → EndOf parent (16dp)
- Top → TopOf parent (16dp)
- Bottom → BottomOf parent (16dp)

layout\_width: 269dp

layout\_height: 268dp

visibility: [dropdown]

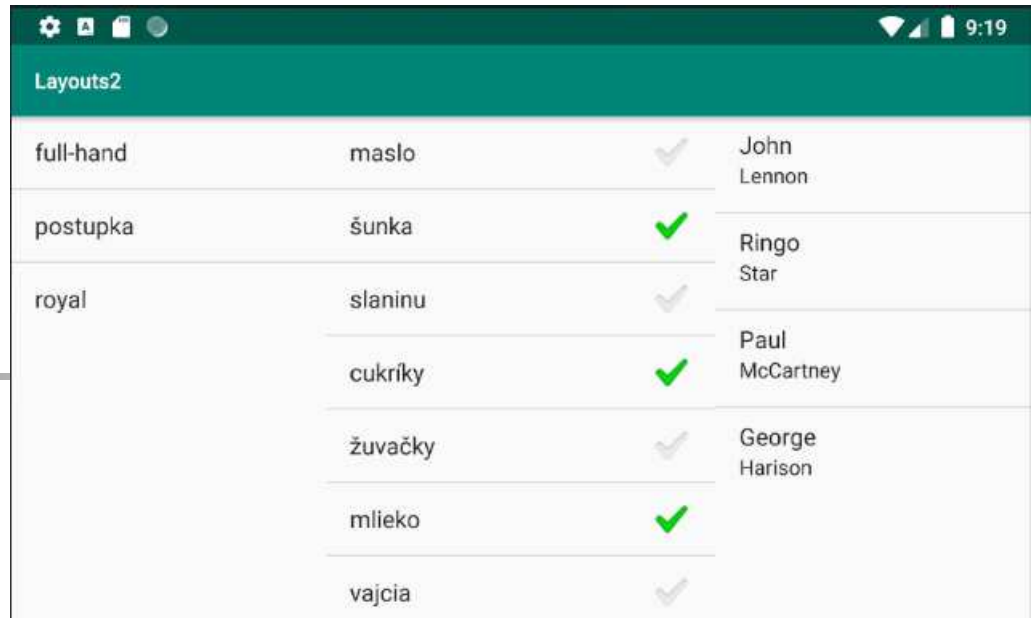
visibility: [dropdown]



# ListView

(variabilita)

- preddefinovaný štýl
  - môžu/nemusia sa nám páčiť
- user defined
  - narobíme sa pri ich definícii



Category	Description	Checkbox	Name
full-hand	maslo	<input type="checkbox"/>	John Lennon
postupka	šunka	<input checked="" type="checkbox"/>	Ringo Star
royal	slaninu	<input type="checkbox"/>	Paul McCartney
royal	cukríky	<input checked="" type="checkbox"/>	George Harison
royal	žuvačky	<input type="checkbox"/>	
royal	mlieko	<input checked="" type="checkbox"/>	
royal	vajcia	<input type="checkbox"/>	

Tri rôzne inštancie ListView  
`simple_list_item_1`,  
`simple_list_item_checked`  
`simple_list_item_2`

## Odchyťavanie udalostí v ListView

```
com.example.layouts2 D/ZOZNAM: beatles click: 2:{krstne=Paul, priezvv=McCartney}
com.example.layouts2 D/ZOZNAM: beatles click: 1:{krstne=Ringo, priezvv=Star}
com.example.layouts2 D/ZOZNAM: beatles click: 3:{krstne=George, priezvv=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)

full-hand	žuvačky	✓
postupka	mlieko	✓
royal	vajcia	✓
	pečivo	✓

simple\_list\_item\_1

simple\_list\_item\_checked

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

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

# ListView

(simple\_list\_item\_checked)

full-hand	žuvačky	<input type="checkbox"/>
postupka	mlieko	<input type="checkbox"/>
royal	vajcia	<input type="checkbox"/>
	pečivo	<input checked="" type="checkbox"/>

simple\_list\_item\_1

simple\_list\_item\_checked

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

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

# ListView 2

(simple\_list\_item\_2)

"krstne"

"priezv"

John
Lennon
Ringo
Star
Paul
McCartney
George
Harrison

Naplniť iný, napr. dvojriadkový ListView je náročnejšie

```
// beatles list view
```

```
val pairs = listOf(  
    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,  
    android.R.layout.simple_list_item_2,  
    arrayOf("krstne", "priezv"),  
    arrayOf(android.R.id.text1, android.R.id.text2)  
        .toIntArray()  
)  
  
listView3.setOnItemClickListener {  
    adapterView, view, index, l ->  
    val hodnota = adapterView.getItemAtPosition(index)  
    Log.d("onItemClick", "beatles click: $index:$hodnota")  
}
```

# Rôzne preddefinované ListView

(prehľad)

