Hitparáda aktivít

Aktivity, View
Intent, Layout

Peter Borovanský
KAI, I-18

borovan 'at' ii.fmph.uniba.sk

Príklad jednoduchej aplikácie

(RECAP 3.prednášky)

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
- previazanie View Binding

Pikas2.zip

Logovanie

(RECAP 3.prednášky)

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()
...
})
```

Pikas2.zip

Pikas

(RECAP 3.prednášky)

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

Pikas2.zip

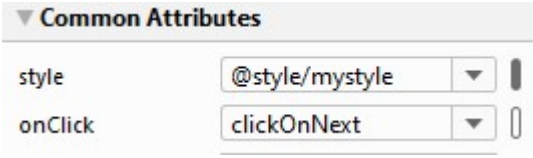
Pikas

(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)
    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])
}
```



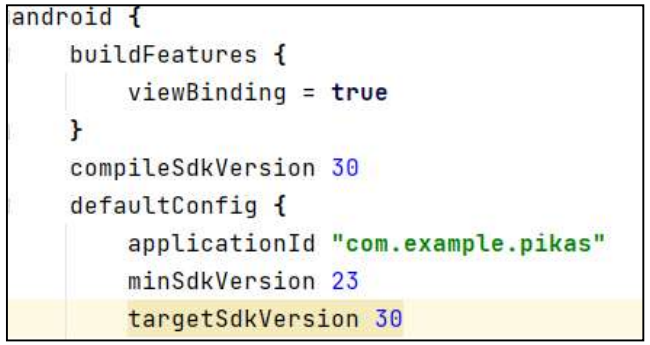
Pikas2.zip

View Binding

- findViewById()
- syntetic – kotlin-android-extensions plugin
- ďalší spôsob prepojenia komponentov (View) z .xml layoutu s kódom
- pozor: nepliesť si to s Data Binding, to príde s JetPack library, to je zložitejšie

1) do build.gradle pridajte pod

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



2) v samotnej Activity NAHRADÍTE

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

za

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

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

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

```
lateinit var binding : ActivityMainBinding
```

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

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

Pikas2.zip

View Binding

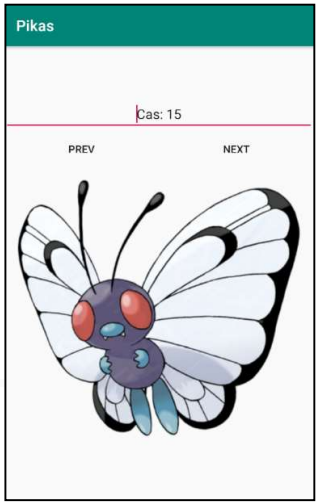
```
binding.imageView2.setImageDrawable(imgs[i])
binding.prevBtn2.setOnClickListener {
    Toast.makeText(this,
                   "prev...",
                   Toast.LENGTH_SHORT
    ).show()
    if (--i < 0) i += imgs.size
    binding.imageView2.setImageDrawable(imgs[i])
}
```

```
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])
     }
}
```

Pikas2.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

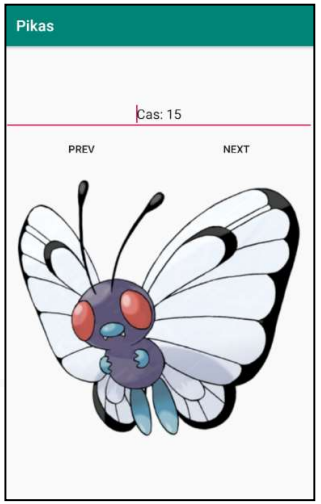
zabitie timera:

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

Pikas2.zip

Pikas

(asynchrónnosť – count down)



pomocou **android.os.CountDownTimer**

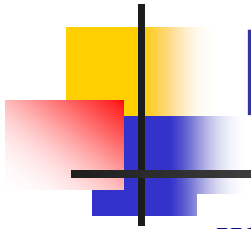
```
object:CountDownTimer(20000, 1000) { // 20sek, tik po 1sek
                                     // how long, period
    override fun onTick(millisUntilFinished: Long) {
      Log.d(TAG, "onTICK")
      runOnUiThread {
         time.setText("Cas: ${millisUntilFinished/1000}") }
    }

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

tik

game over

ukončenie appky

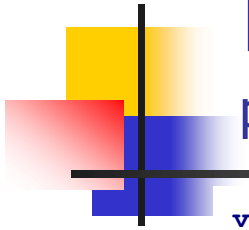
Pikas2.zip



Handler

```
val handler = Handler(Looper.getMainLooper())


override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    val runnable = Runnable {
        updateSomeGUI() // some actions
    }
    ...
    handler.postDelayed(runnable, 5000)  // +milis

    val now = SystemClock.uptimeMillis()
    val uptime = now + (15*1000-now % (15*1000)) // 15 sec
    handler.postAtTime(runnable, uptime)

    handler.post { Runnable { updateSomeGUI() }}
}
```

Pikas2.zip



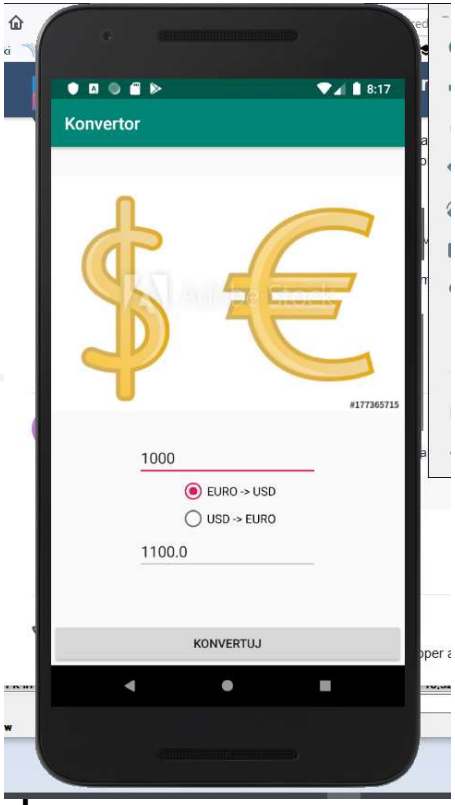
Handler

prenos parametrov

```
val handler = Handler()
val handler2 = object : Handler() {
    override fun handleMessage(msg: Message) {
        super.handleMessage(msg)
        Log.d("HANDLER", "msg = ${msg.arg1}")
        Log.d("HANDLER", "msg = ${msg.data.getString("time")}")
    }
}
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    val runnable = Runnable {
        updateSomeGUI() // some actions
        val msg = Message()
        msg.arg1 = 1
        msg.arg2 = 2
        val bundle = Bundle()
        val dateFormat = SimpleDateFormat("HH:mm:ss MM/dd/yyyy", Locale.US)
        bundle.putString("time", dateFormat.format(Date()))
        msg.data = bundle
        handler2.sendMessage(msg)
    }
    handler.postDelayed(runnable, 5000)  // milis
    handler.post { Runnable { updateSomeGUI() }}
}
private fun updateSomeGUI() {
    //...
}
```

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

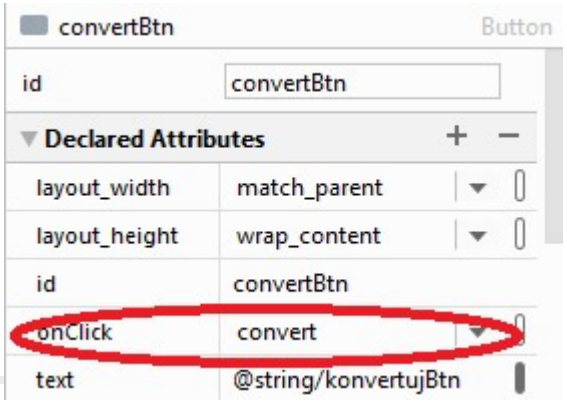
```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    convertBtn.setOnClickListener({
        Toast.makeText(this, "convert",Toast.LENGTH_SHORT).show();
        if (inputText.text.isNotEmpty()) {
            val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

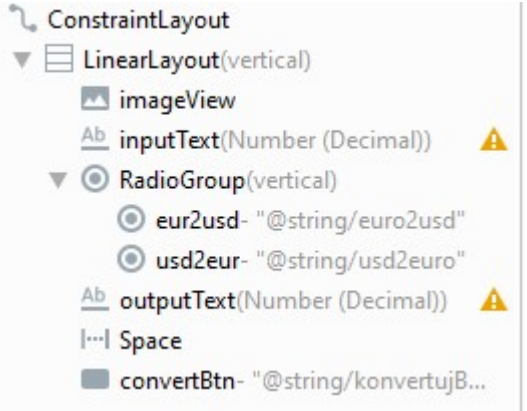
Konvertor.zip

a

Konvertor EURO USD

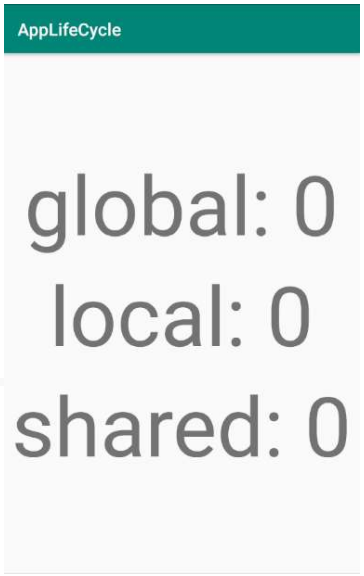
(layout)

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



Konvertor.zip

Životný cyklus apky
(prvý – zjednodušený nástrel)



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

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

tag vhodný na filtrovanie najlepšie definovať ako konštantu

AppLifeCycle.zip

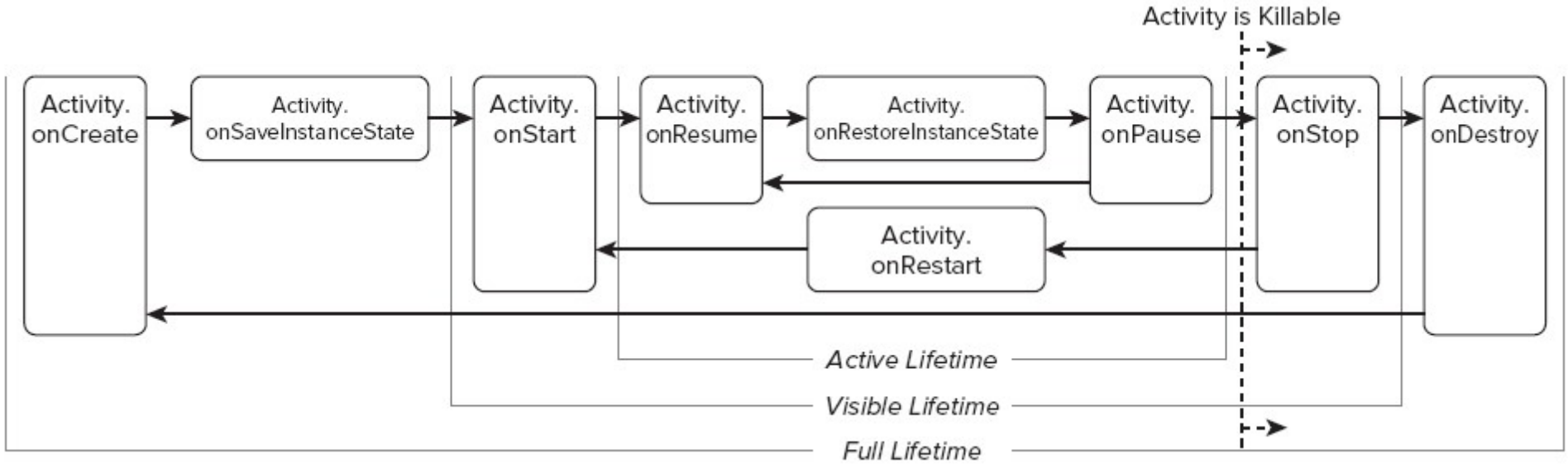
LogCat

(Filtrovanie 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

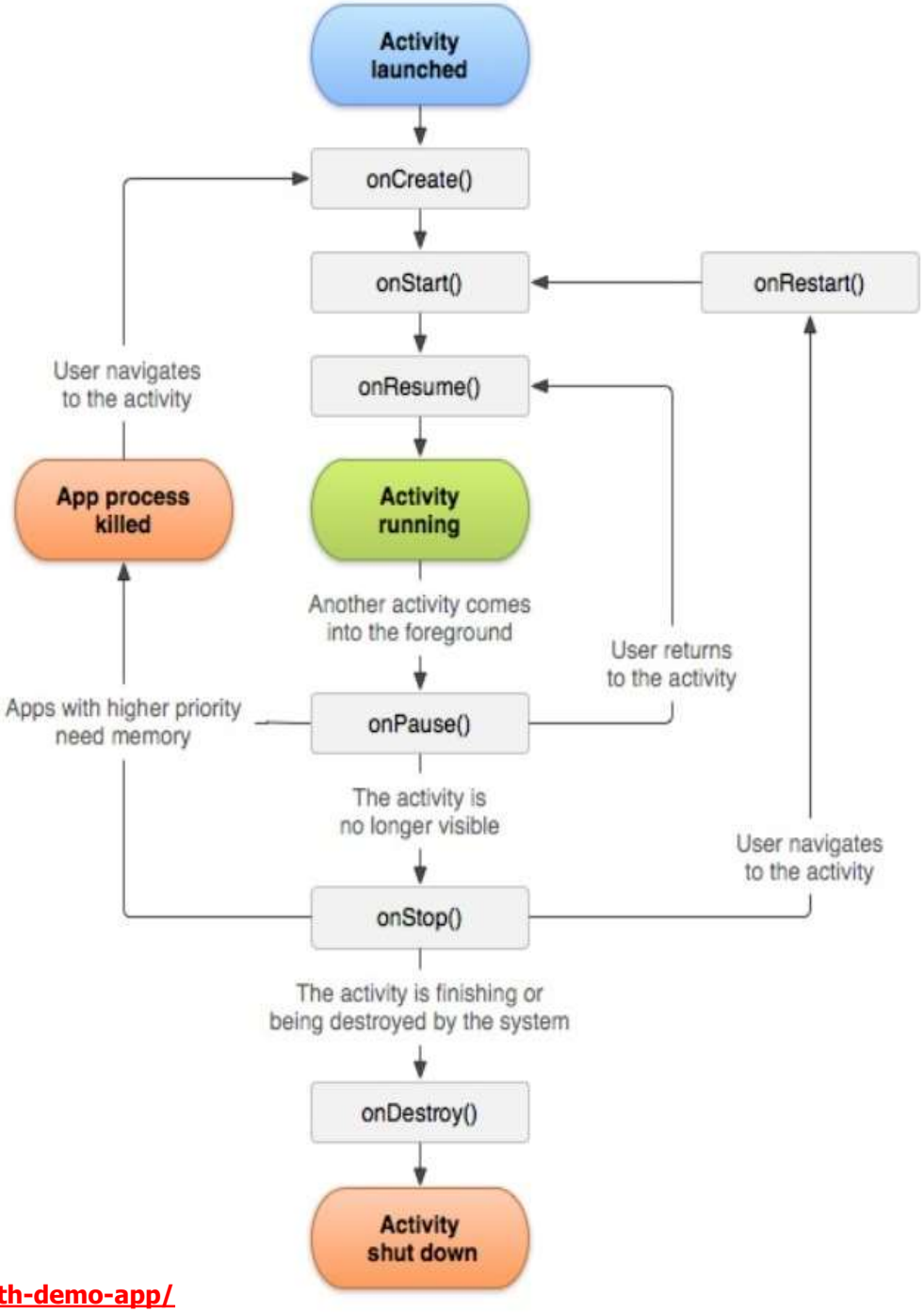
kill

- 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



zdroj: Reto Meier: PA2AD

AppLifeCycle.zip

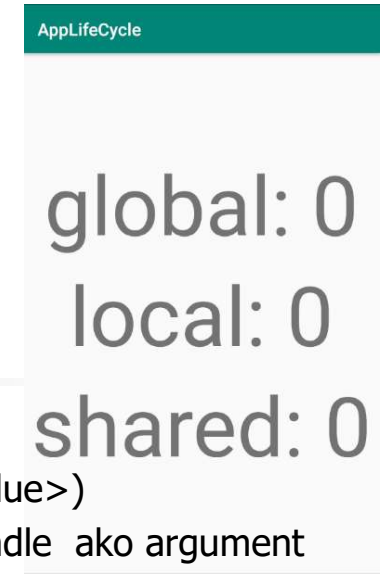


https://media.geeksforgeeks.org/wp-content/uploads/20191125171002/Activity-Lifecycle-in-Android-Demo-App.mp4

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

Persistencia

(prvý dotyk)



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

Zistíte, že:

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

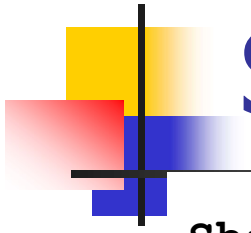
AppLifeCycle.zip

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)
  ...
}
```

AppLifeCycle.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)
      commit()
   }
}
```

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

editor.commit()
```

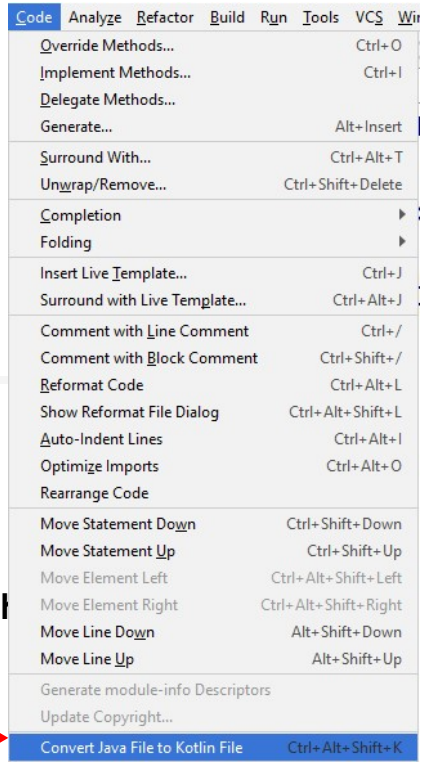
AppLifeCycle.zip

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).sh
    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 Kotline)

- **Kotlin -> JVM Byte code**

Tools/Kotlin/Show Byte Code

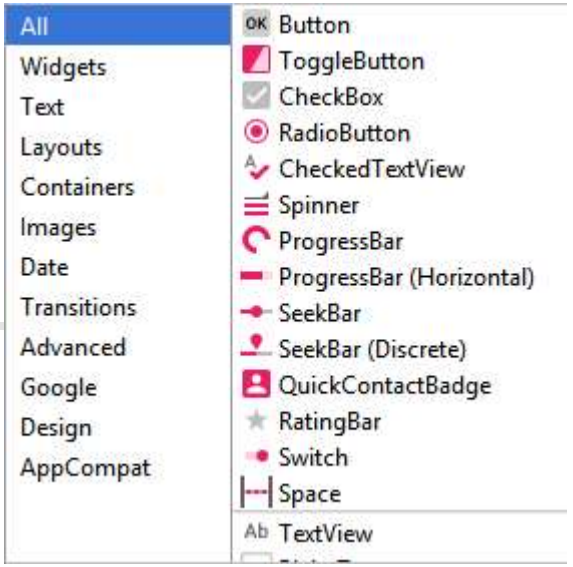
- **Decompile Byte code (to Java)**

```
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 ?

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

GUI komponenty



Layout

- LinearLayout (Verical/Horizontal)
- RelativeLayout, ConstraintLayout

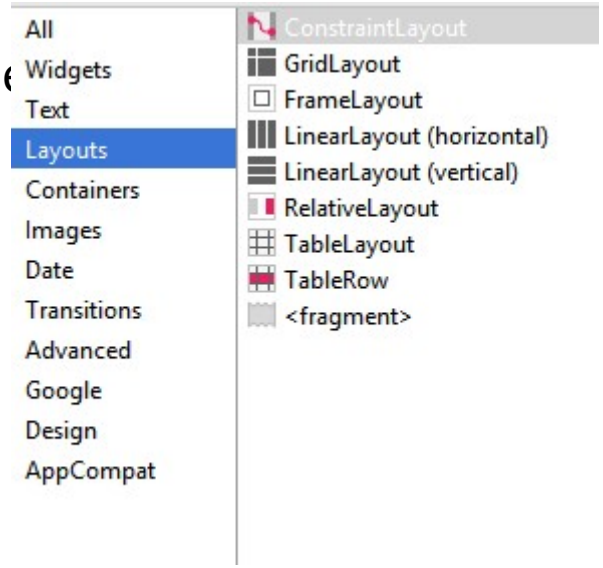
View, ViewGroup

- všetky viditeľné komponenty (widgets)

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

najznámejšie podtriedy

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



Fragment (>= 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/lbl"
              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>
```

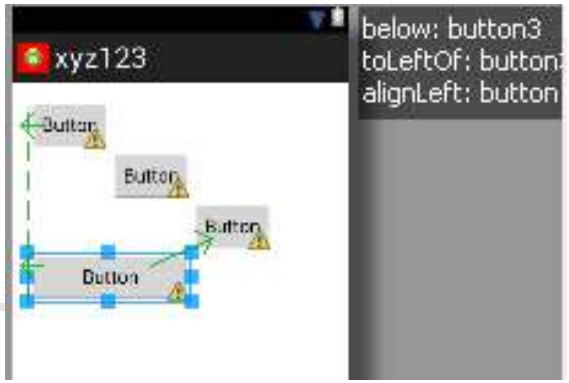
Kód na slajde je zjednodušený, originál nájdete v Layouts2.zip

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

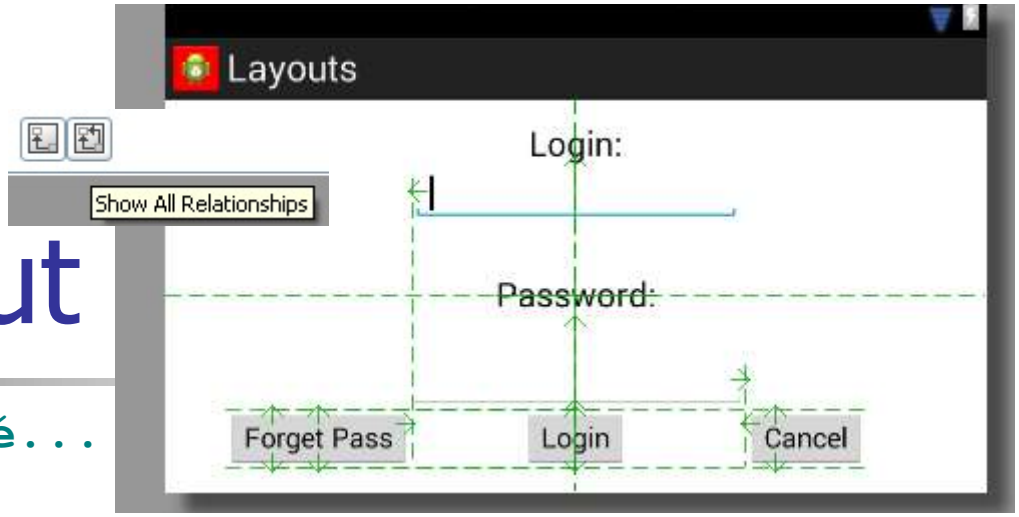
RelativeLayout




```
<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

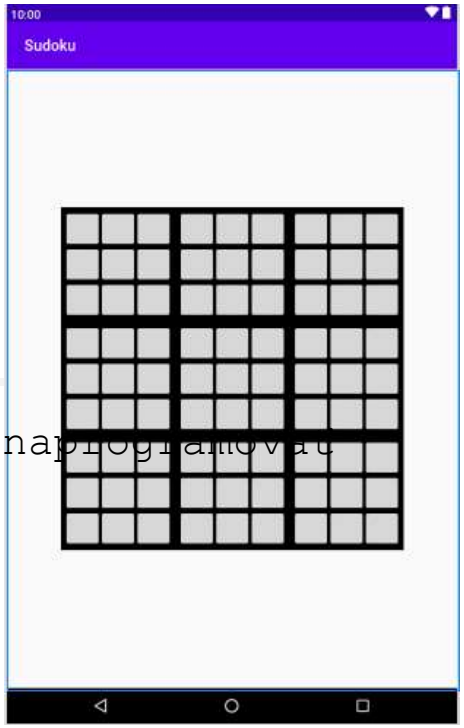
RelativeLayout




```
<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

Dynamický vs. statický layout



Layout môžete vyklikať, niekedy náročné, alebo naprogramovať

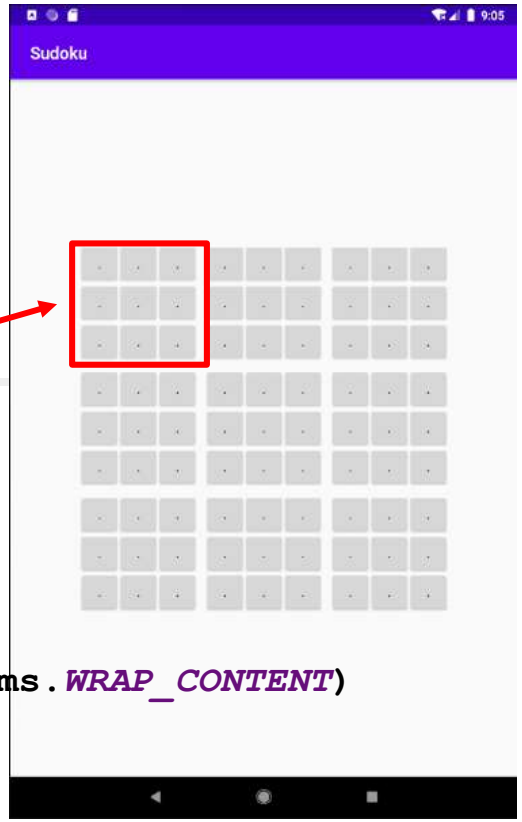
```
<GridLayout
    android:id="@+id/bigGrid"
    android:columnCount="3"
    android:rowCount="3">

    <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

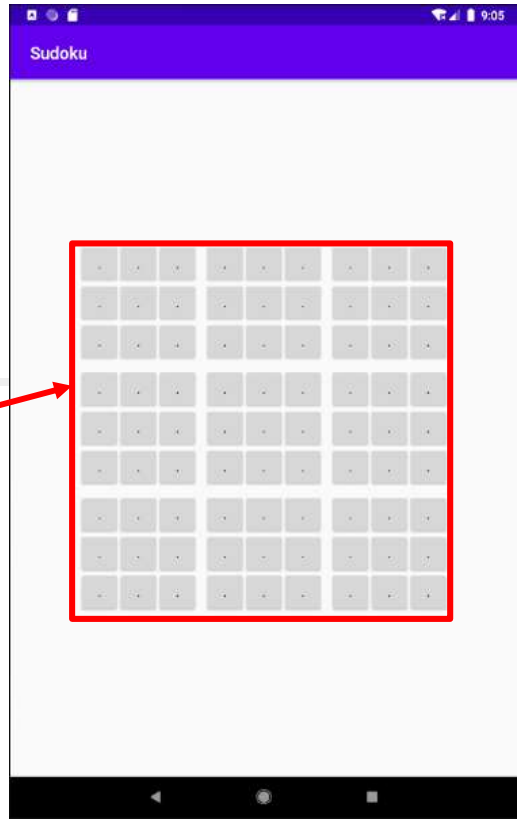



```
val smallGrid = GridLayout(this)
smallGrid.columnCount = SIZE
smallGrid.rowCount = SIZE
val smallGridParams = ViewGroup.MarginLayoutParams(

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



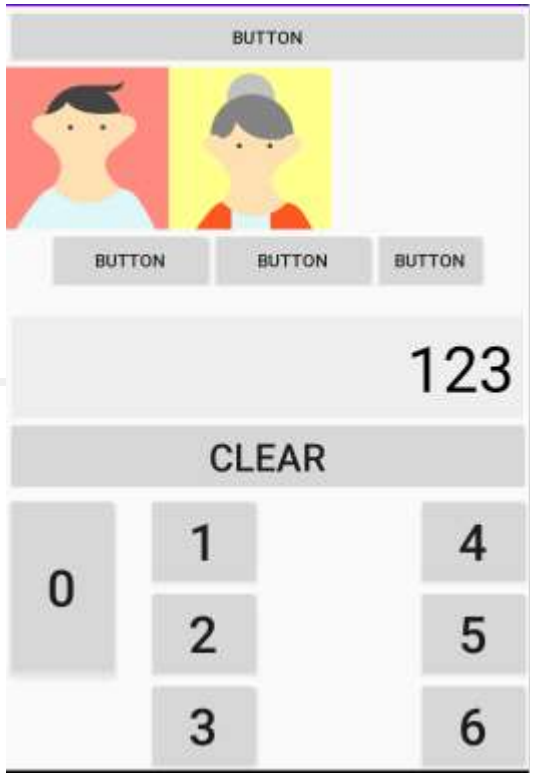

```
val SIZE = 3

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

Table vs. GridLayout

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

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

    <TableRow
       . . .
```

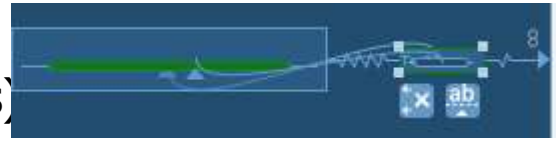



```
<GridLayout
    android:id="@+id/grid"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:columnCount="4"
    android:rowCount="6">

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

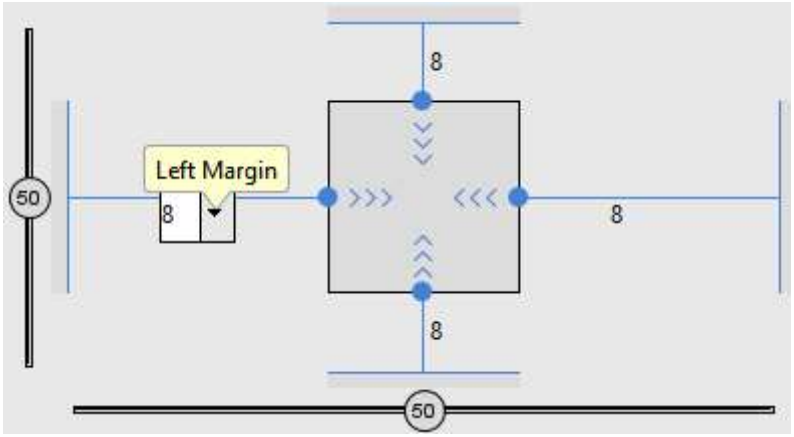
Constraint Layout

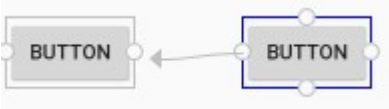
Je zovšeobecnením RelativeLayout

umožňuje nastaviť väzby, či obmedzenia (constraints)



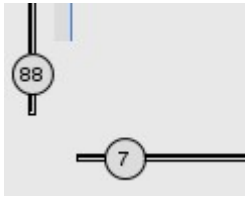
- relatívnu pozíciu
- spoločná baseline pre text
- okraje
- wrap/match content/fixná veľkosť
- vychýlenie (bias)

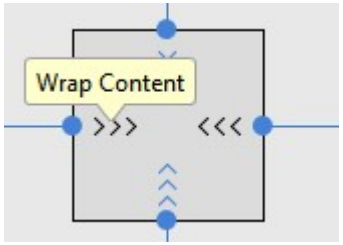




https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout

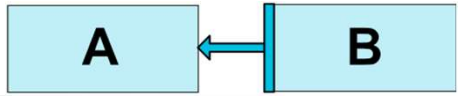
https://www.youtube.com/watch?v=z53Ed0ddxgM

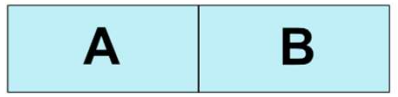




Kód na slajde je zjednodušený, originál nájdete v Layouts2.zip

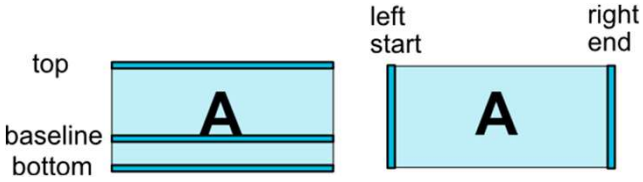
Niektoré možnosti



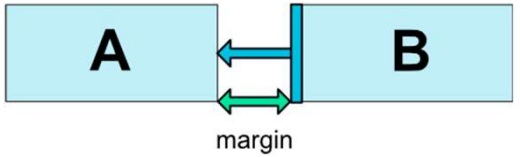



```
<Button android:id="@+id/buttonA" ... />
 <Button android:id="@+id/buttonB" ...
        app:layout_constraintLeft_toRightOf="@+id/buttonA" />
```

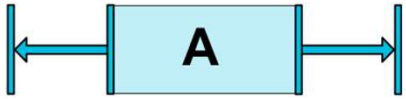
layout_constraintBaseline_toBaselineOf



android:layout_marginLeft



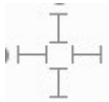
layout_constraintVertical_bias



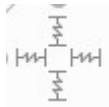
- je ich ďaleko viac, ale zaujímavejšie je to v designeri

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

Constraint Layout

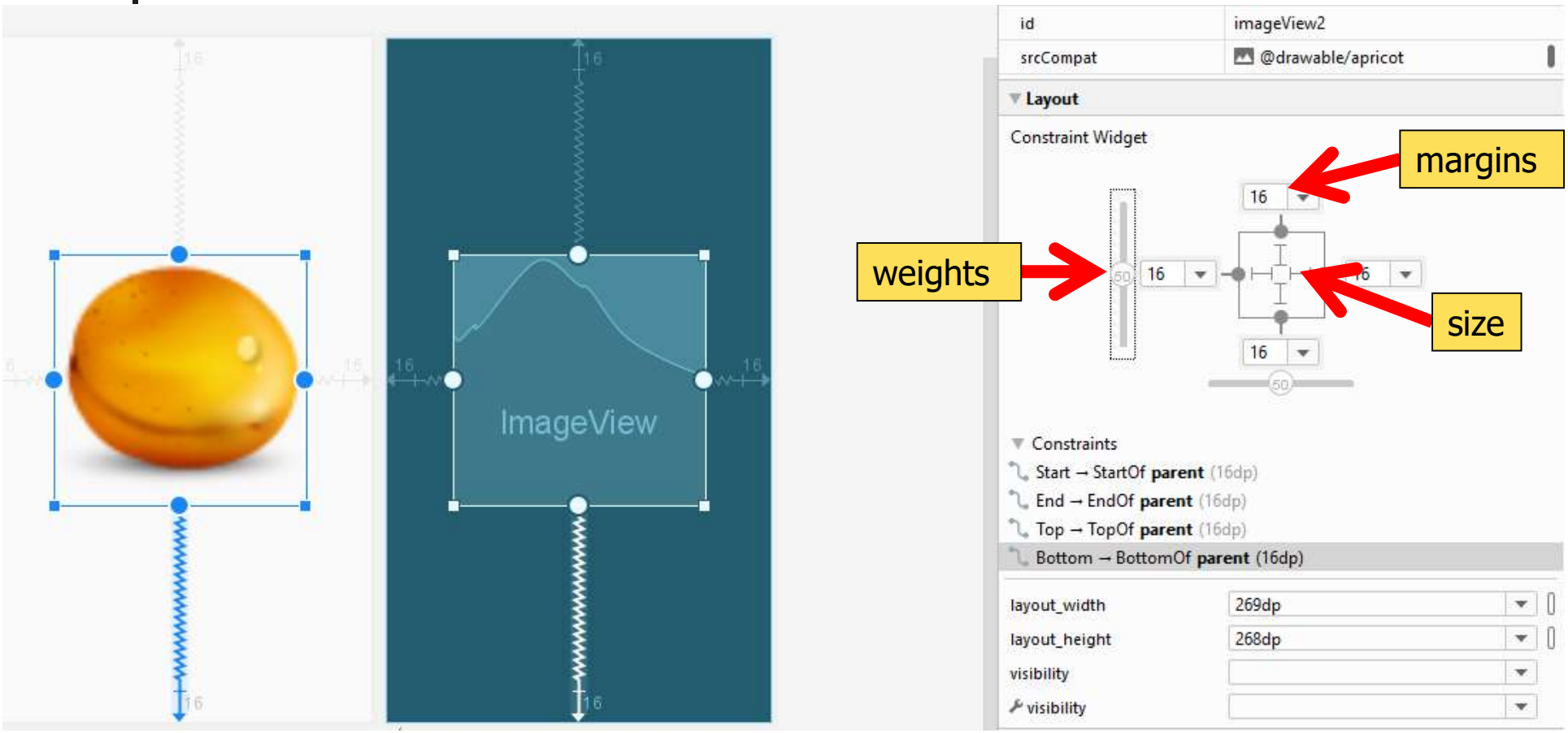


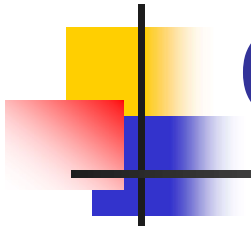
fixed size in dp ☹



match parent

wrap content





Constraint Layout

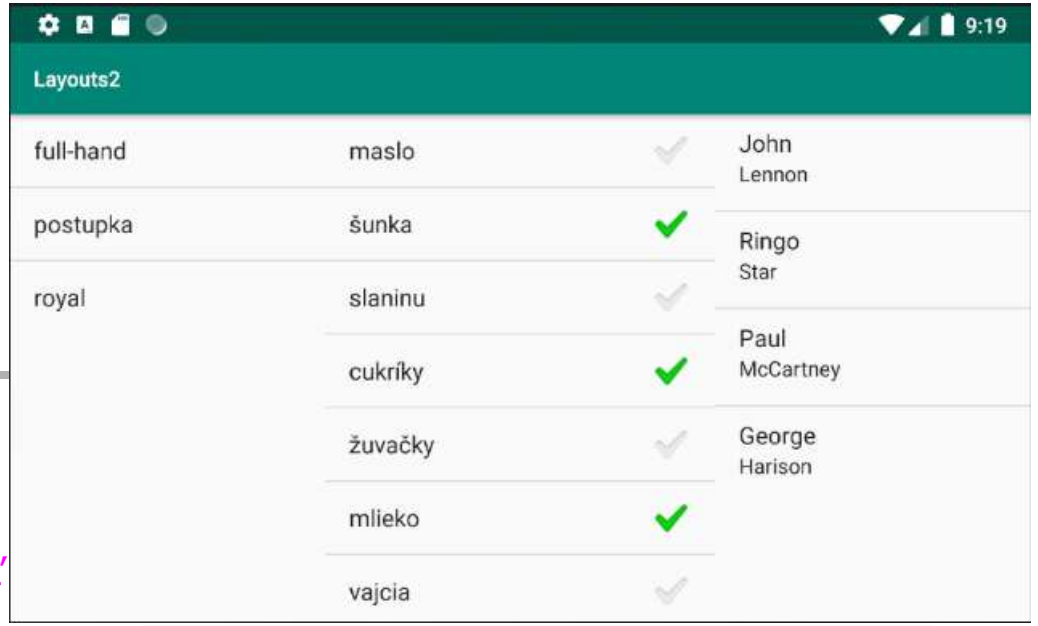
- Convert View
- BluePrint vs. Design móde
- Show constraints – zobrazí constraints v BluePrint resp. Design móde
- remove constraint (ctrl-)
- Horizontálne vertikálne constraints nemožno miešať
- Komponent musí mať aspoň jeden horizontálny a verikálny constraint
- Clear All Constraints – zmaže všetky
- Okraje - margins
- Infer Constraints –
- match_constraint "0dp "(deprecated match_parent)/wrap_content
- base line
- Imageview - ratio

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
```

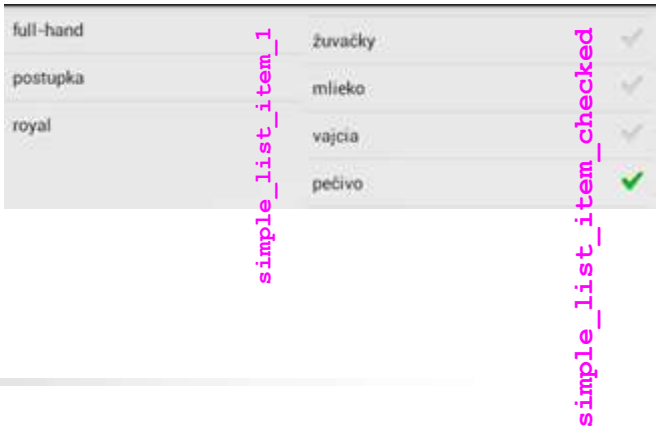
Project: Layouts2.zip

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

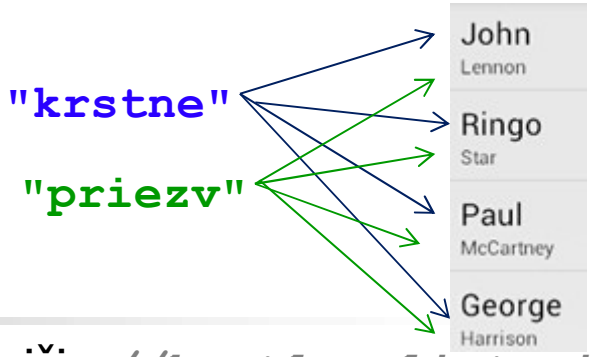
ListView
(simple_list_item_checked)




```
// 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*

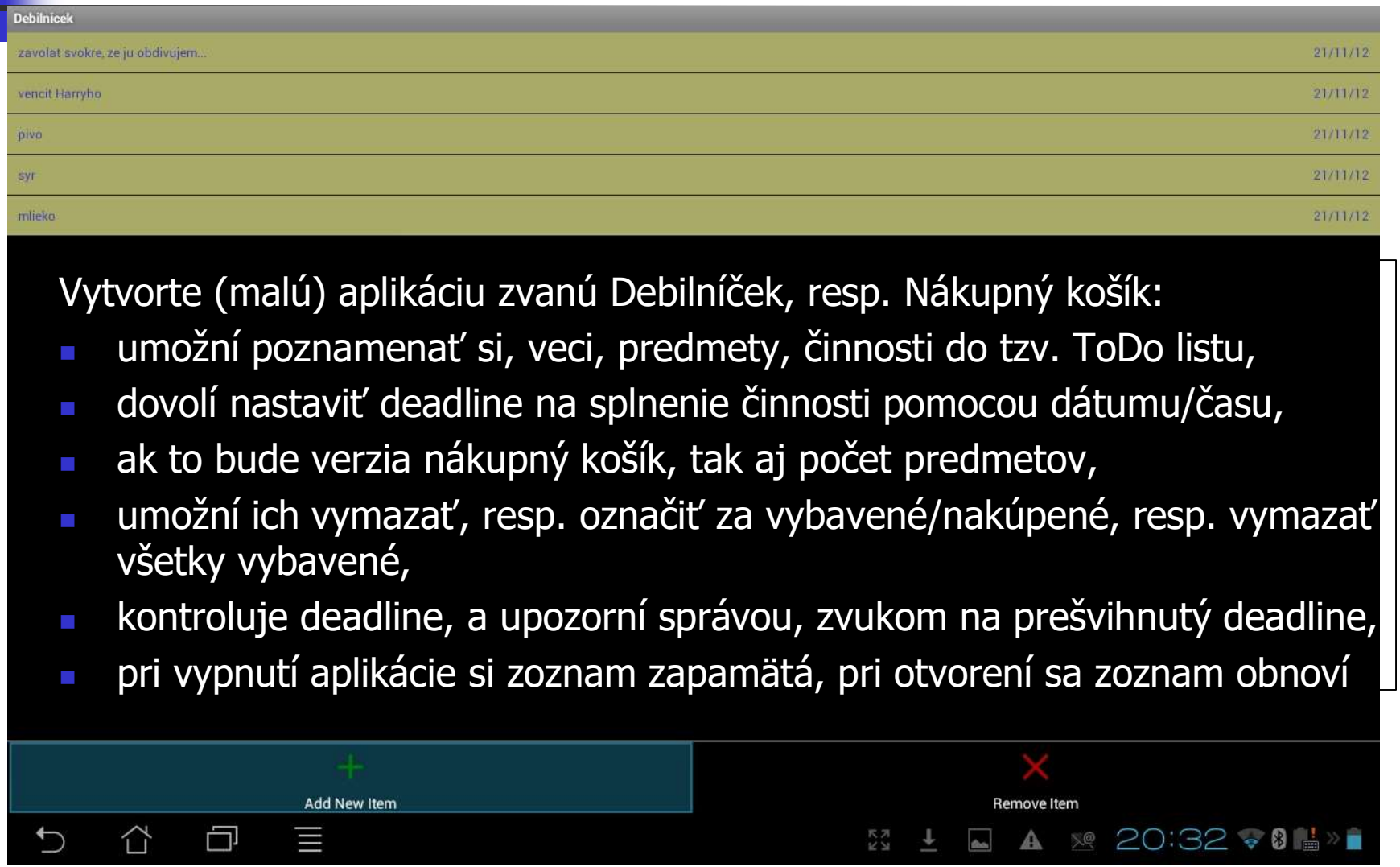
```
val pairs = listOf( // hodnoty sú zoznam máp kľúč->hodnota
    mapOf("krstne" to "John","priezv" to "Lennon"),mapOf("krstne" to "Ringo","priezv" to "Star"),
    mapOf("krstne" to "Paul","priezv" to "McCartney"),mapOf("krstne" to "George","priezv" to "Harison")
)
listView3.adapter = SimpleAdapter(this,
      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