Hitparáda aktivít

Aktivity, View
Intent, Layout

Peter Borovanský
KAI, I-18

borovan 'at' ii.fmph.uniba.sk

Hitparáda

Feedback / Hra15 / Apple Watch Tips

(Hall of Fame)

Príklad jednoduchej aplikácie

(ktorú sme si vyklikali minule)

Ilustrovali sme:

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

Project:Pikas2.zip

Nestihli sme:

- aktivitu a jej život(ný cyklus)

Logovanie
(rekapitulácia)

Tri najbežnejšie spôsoby:

- Log – loguje do okna Logcat, filtrujte podľa `TAGu` metódy `Log.d(TAG,`
- Toast – potrebuje `Context` (zjednodušene aktivita, v ktorej sa toastuje)
- Snackbar – to chce pridať závislosť do build.gradle a import snackbaru

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

prevBtn2.setOnClickListener({
    Toast.makeText(this, "prev...", Toast.LENGTH_SHORT).show()
    Log.d(TAG, "prev...")
    Snackbar.make(it, "prev...",
        Snackbar.LENGTH_SHORT).setAction("Action", null).show()
    ...
    if (--i < 0) i += imgs.size
    imageView2.setImageDrawable(imgs[i])
})
```

Project:Pikas2.zip

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

Project:Pikas2.zip

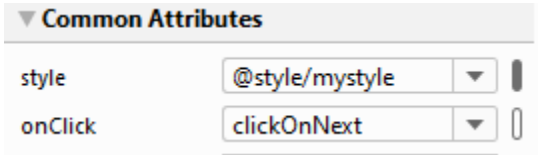
Pikas

(stav sa mieša s views a BL)

const
final

State

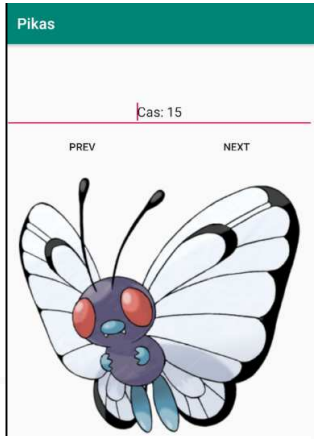
```
val TAG = "PIKAS"
var i = 0
var imgs = arrayOf<Drawable?>()
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    imgs = arrayOf(ContextCompat.getDrawable(applicationContext,
                                             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])
}
```



Project:Pikas2.zip

Pikas

(asynchrónnosť - timer)



pomocou **java.util.Timer**

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

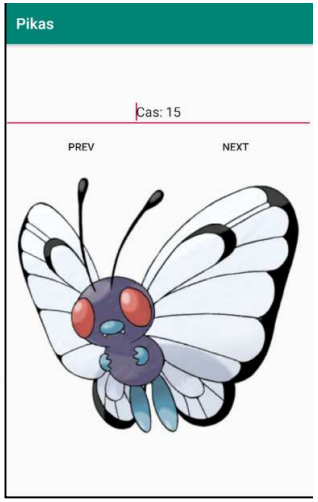
- nezabudnite na .run()
- runOnUiThread
  - má argument java.lang.Runnable, ktorý vykoná v hlavnom GUI vlákne

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

Project:Pikas2.zip

Pikas

(asynchrónnosť – count down)



pomocou **android.os.CountDownTimer**

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

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

tik

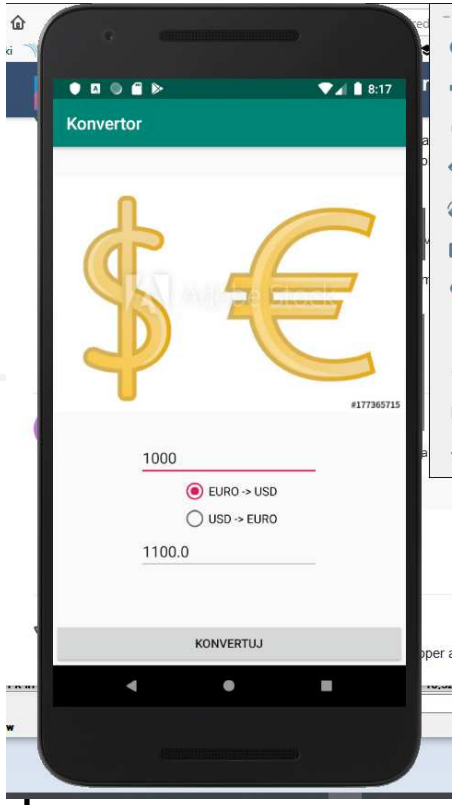
game over

ukončenie appky

Project:Pikas2.zip

Konvertor EURO USD

(logika)



Jednoduchá aplikácia na konverziu kurzov USD EURO

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

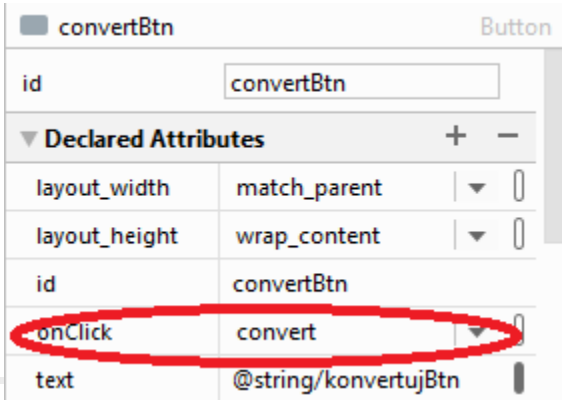
```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    convertBtn.setOnClickListener({
        Toast.makeText(this, "convert",Toast.LENGTH_SHORT).show();
        if (inputText.text.isNotEmpty()) {
            val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

a

Konvertor EURO USD

(setOnClickListener)



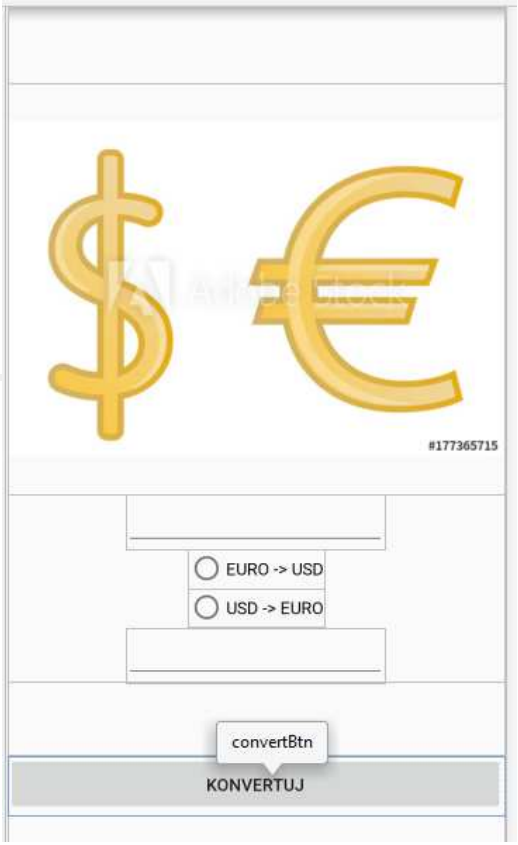

```
// very old fashion
    val cBtn = findViewById<Button>(R.id.convertBtn)
    cBtn.setOnClickListener( { v -> convert(v) } )
    cBtn.setOnClickListener { convert(it) }

// 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")
        }
    }
```

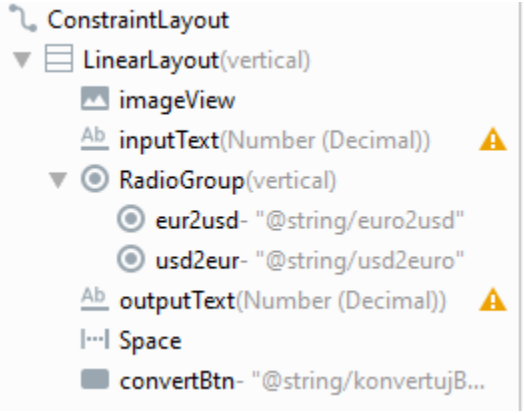
a

Konvertor EURO USD

(layout)

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

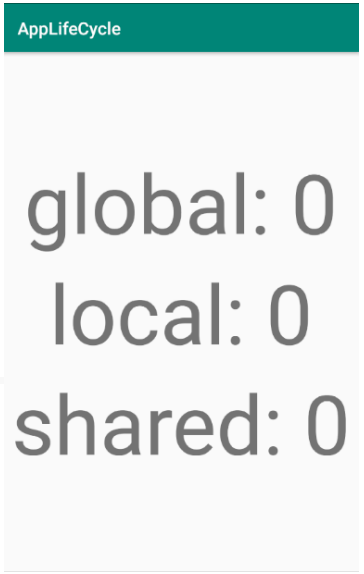




Project:Konvertor.zip

Životný cyklus apky

(prvý – zjednodušený nástrel)



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

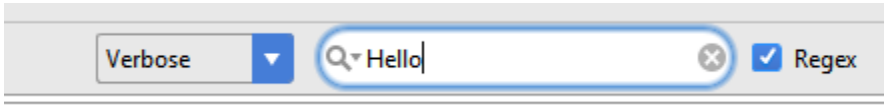
```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.d("CYKLUS","onCreate"); // LOGUJTE, LOGUJTE, LOGUJTE
}
```

tag vhodný na filtrovanie

Project:AppLifeCycle.zip

LogCat

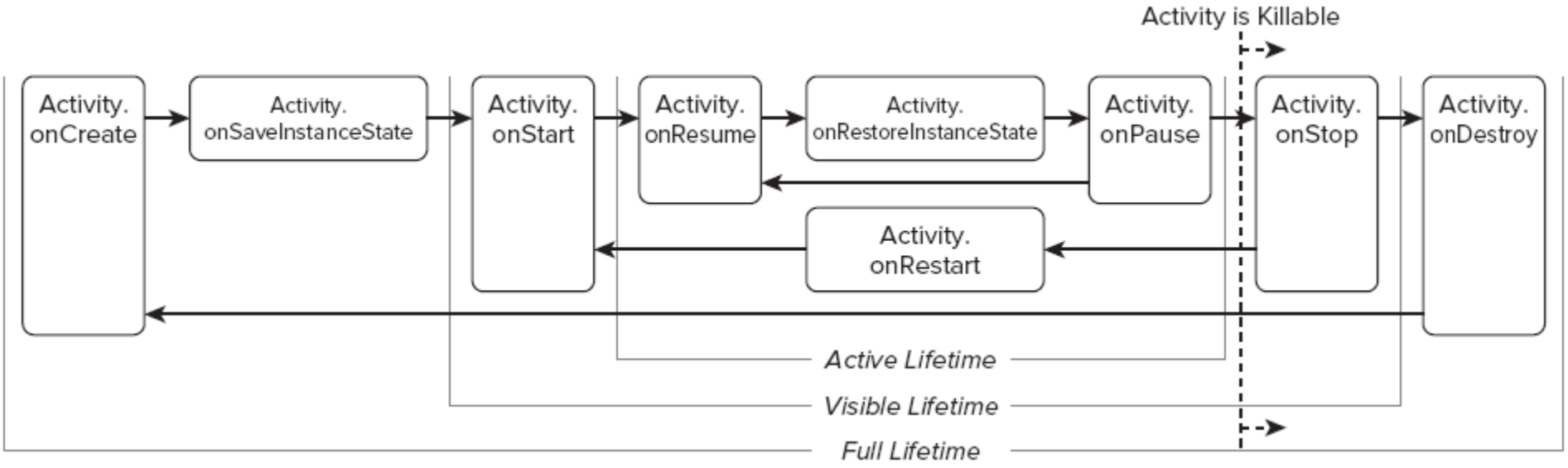
(Filtrovanie logov)



- `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`

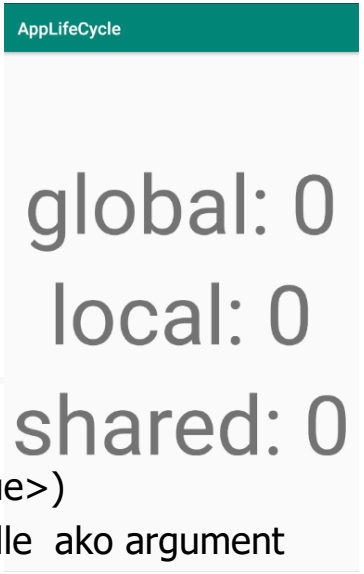
- `10-13 12:56:45.061: D/Hello(405): onPause`
- `10-13 12:56:45.681: D/Hello(405): onStop`
- `10-13 12:56:45.681: D/Hello(405): onDestroy`



zdroj: Reto Meier: PA2AD

Project:AppLifeCycle.zip

Persistencia
(prvý dotyk)

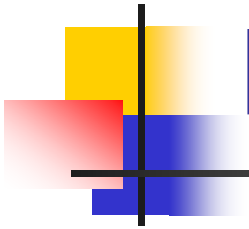


- **`globalCounter`** je premenná, ktorá sa
  - pri **`onSaveInstanceState`** uloží do Bundle (HashMap<String, Value>)
  - pri **`onCreate(savedInstanceState: Bundle?)`** príde táto Bundle ako argument
- **`localCounter`** je bežná lokálna triedna premená v MainActivity
- **`sharedCounter`** je premenná,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

Project: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
  ...
}

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

Project: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 {
          this.putInt("kluc", sharedCounter)
          this.commit()
     }
}
```

Project:AppLifeCycle.zip

Kotlin

Cheat sheets

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

The billion-dollar mistake

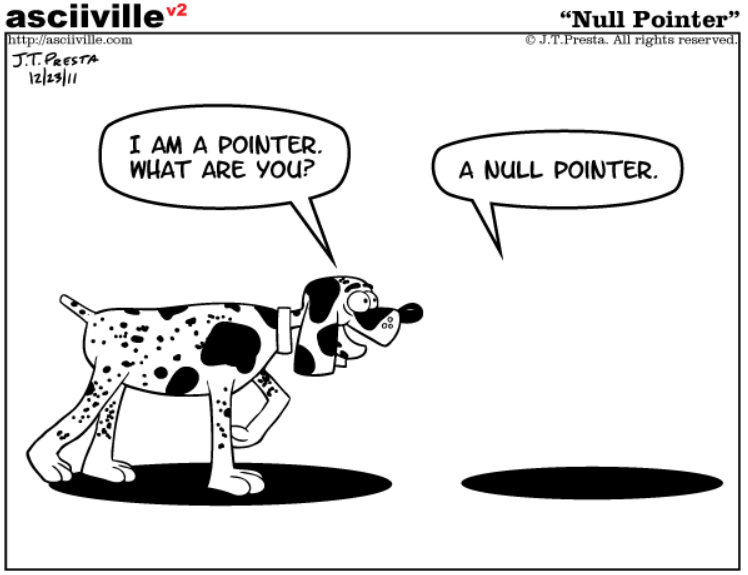
I call it my billion-dollar mistake. It was the invention of the **null** reference in 1965…This has led to innumerable errors, vulnerabilities, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years.

Sir Tony Hoare
FRS FREng

Tony Hoare in 2011

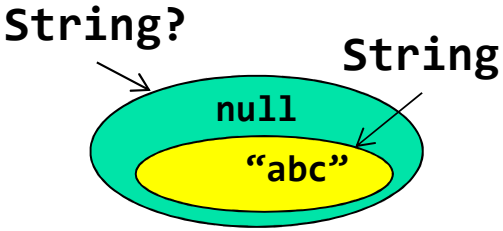
Born	Charles Antony Richard Hoare 11 January 1934 (age 85) Colombo, British Ceylon
Residence	Cambridge
Other names	C. A. R. Hoare
Alma mater	University of Oxford (BA) Moscow State University
Known for	Quicksort Quickselect Hoare logic Null reference Communicating Sequential Processes Structured programming
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]

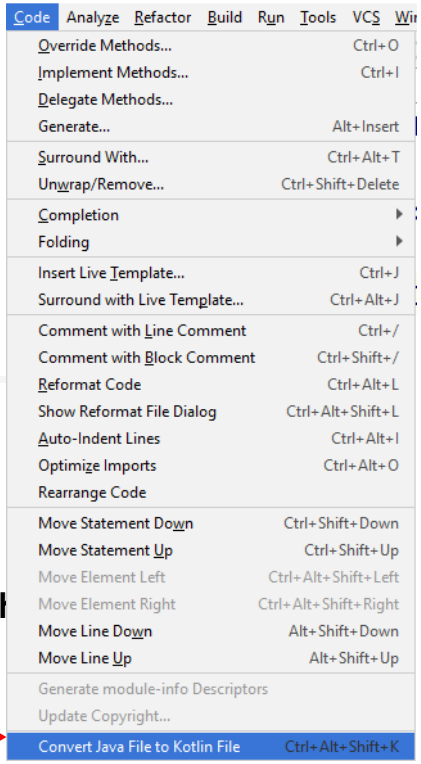
```
fun foo(str : String?) {
   println(str)
   if (str != null) println(str.toUpperCase())
   println(str?.toUpperCase())  // safe call operátor
                        // x?.m == if (x != null) x.m else null
}
fun stringLen(s: String?): Int = s?.length?:0 // Elvis operátor
if (if (s == null) then null 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
}
```

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
    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 Kotline)

- **Kotlin -> JVM Byte code**

Tools/Kotlin/Show Byte Code

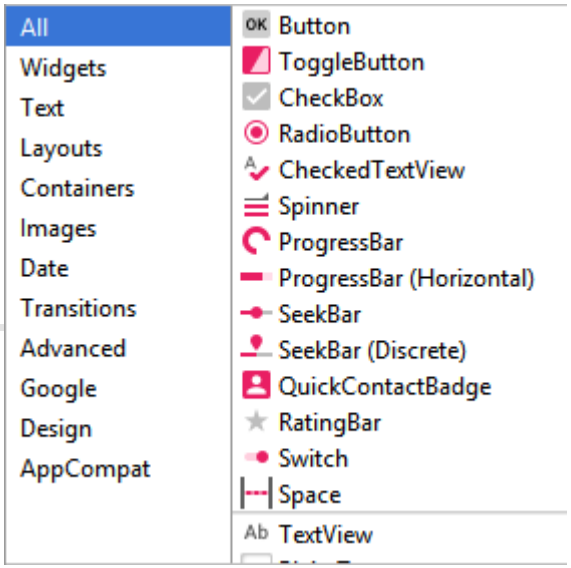
- Decompile Byte code (to Java)

```
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

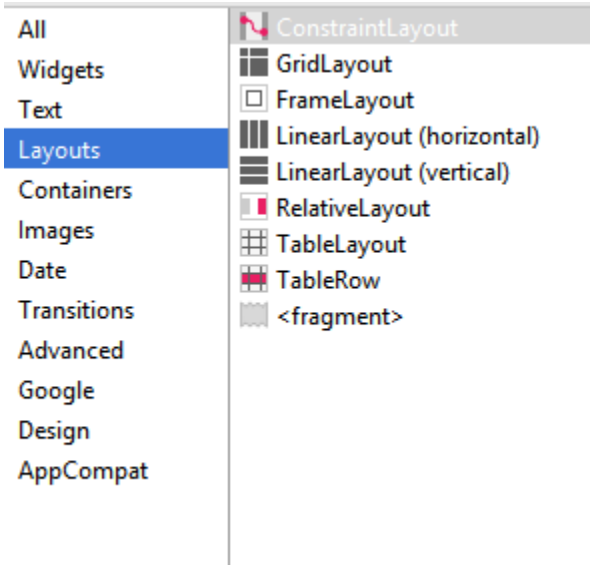
View, ViewGroup

- 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

- 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

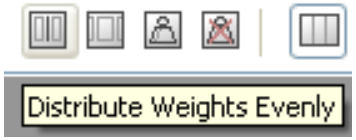
LinearLayout


```
<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


```
<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>
```



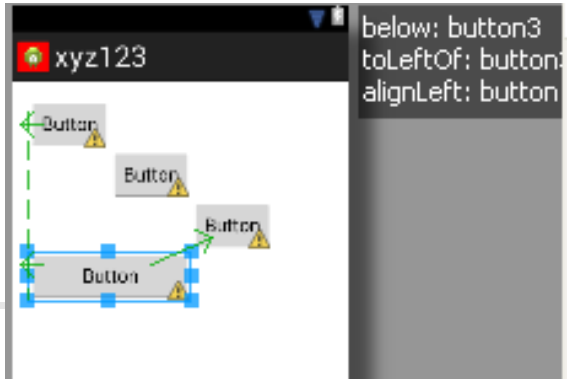
Kód na slajde je zjednodušený, originál nájdete v Layouts2.zip

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

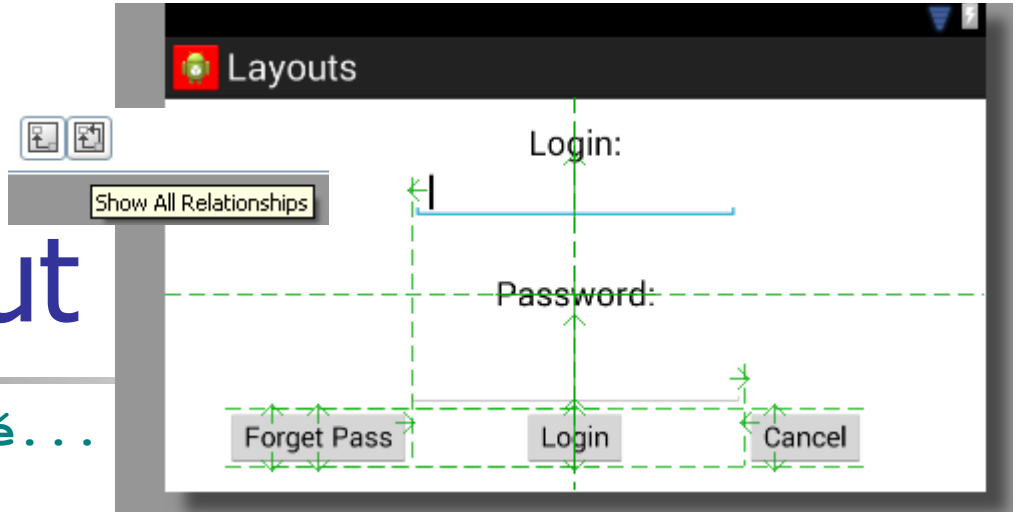
RelativeLayout




```
<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



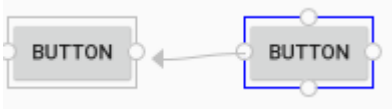

```
<RelativeLayout> ... skrátené...
 <EditText
            android:id="@+id/passwdtv"
            android:layout_below="@+id/pass"
            android:layout_centerHorizontal="true"/>
  <Button
            android:id="@+id/loginBtn"
            android:layout_below="@+id/passwdtv"
            android:text="@string/Login"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

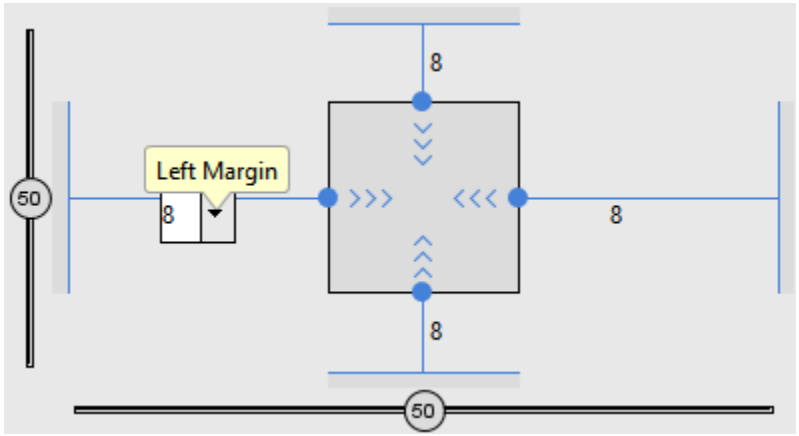
originál nájdete v Layouts2.zip

Constraint Layout

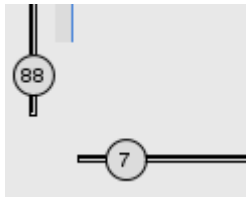
Umožňuje nastaviť väzby

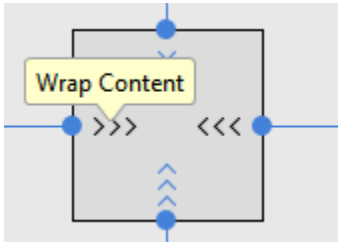
- relatívnu pozíciu
- spoločná baseline pre text
- okraje
- wrap/match content/fixná veľkosť
- vychýlenie (bias)





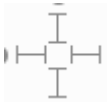
https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html
https://www.youtube.com/watch?v=z53Ed0ddxgM



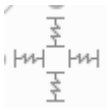


Kód na slajde je zjednodušený, originál nájdete v Layouts2.zip

Constraint Layout

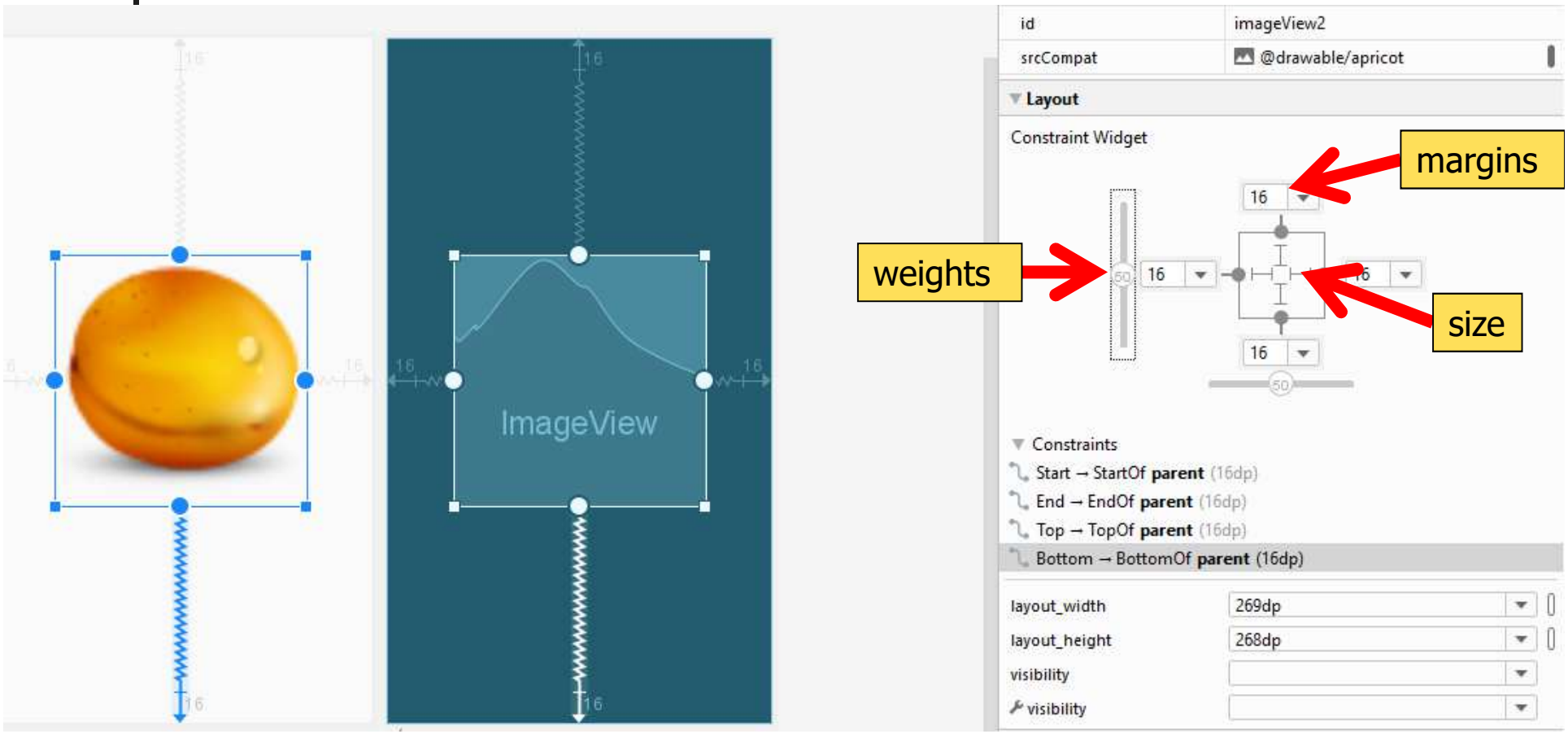


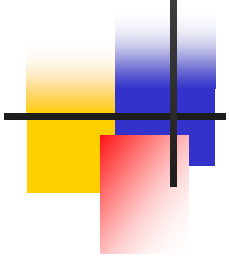
fixed size in dp ☹



match parent

wrap content





ListActivity vs. ListView

(setListAdapter – pre jednoduchý typ ListView)

Všetky aktivity zoradíme do zoznamu, z ktorého sa vybratím daná aktivita spustí

```
... ako naplniť zoznam/list ...
public class MainList extends ListActivity {
// konštanta v programe, ktorou naplníme ListView
String[] myActivities = {"Intro“,"MainActivity","EmailActivity”,…};
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
// alternatíva: zoznam aktivít je v
// res/values/strings
myActivities = getResources().getStringArray(R.array.listofallact);
setListAdapter(
   new ArrayAdapter<String>(MainList.this,        // kde je ListView
         android.R.layout.simple_list_item_1,     // typ ListView
         myActivities));                  // hodnoty do ListView
}
```

```
<string-array name="listofallact">
    <item>Intro</item>
    <item>MainActivity</item>
    <item>EmailActivity</item>
</string-array>
```

01Intro/ActiList.zip

ListActivity

(onListItemClick, reflexivita)

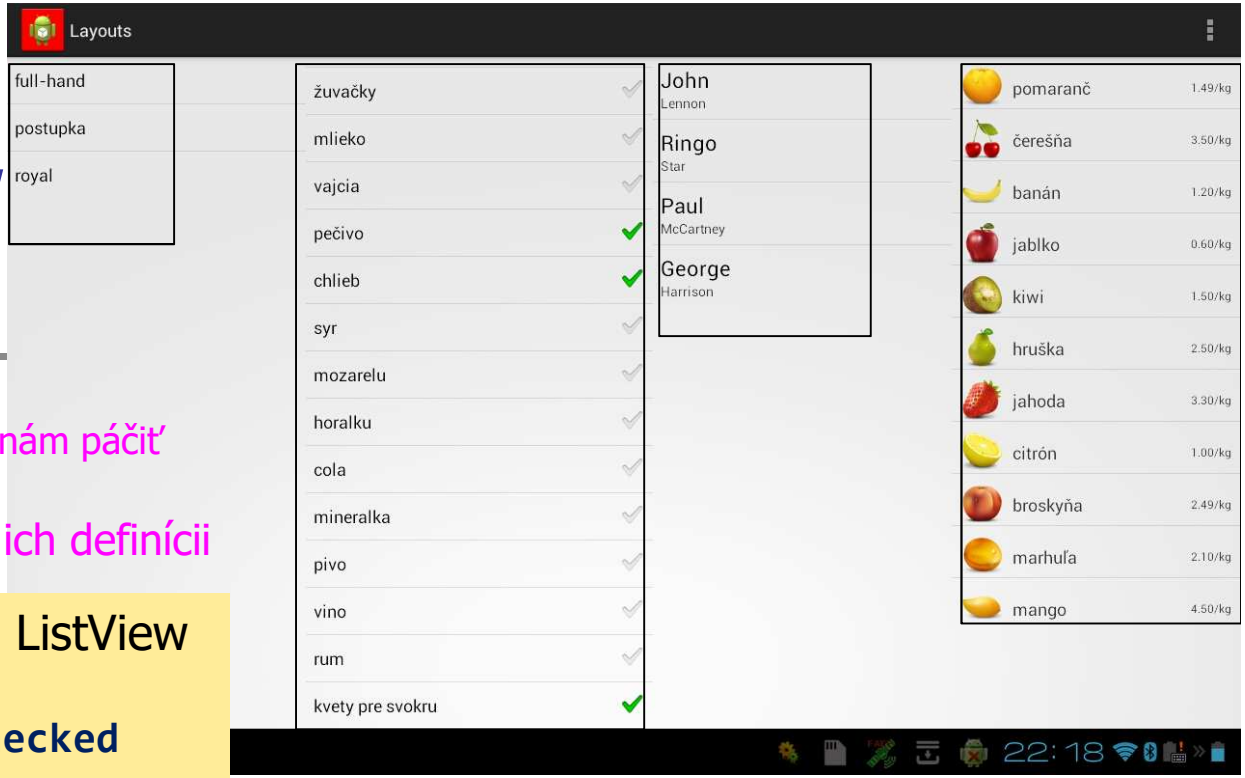
Ako z mena aktivity vyrobiť intent, ktorý aktivitu spustí,
resp. meno balíka (package), v ktorom je aktivita definovaná

```
protected void onListItemClick(ListView l, View v,
                               int position, long id) {
   String className = myActivities[position]; // napr. “Intro”
   try {
        Class mainClass =        // wow !! Reflexivita v praxi ;-)
                Class.forName("com.example.actilist."+ className);
        Intent in = new Intent(MainList.this, mainClass);
        startActivity(in);
   } catch (Exception E) {
        E.printStackTrace();
   }
}
```

01Intro/ActiList.zip

ListView
(variabilita)

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

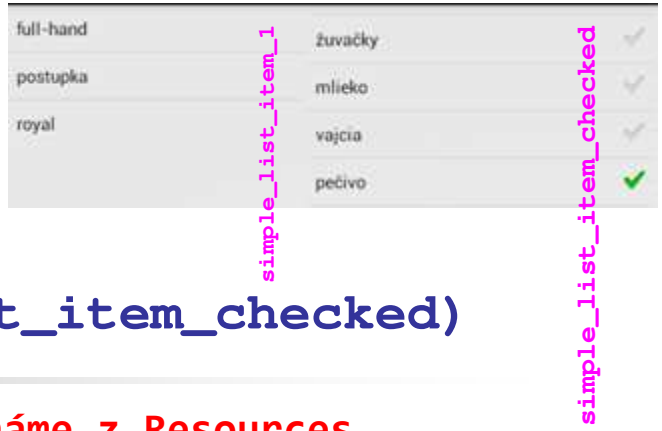


Štyri rôzne inštancie ListView
`simple_list_item_1`,
`simple_list_item_checked`
`simple_list_item_2`
a vlastný row layout

Sústredili sme sa na layout, a neriešili sme odchytávanie udalostí v ListView

```
10-19 16:11:00.179: D/onItemClick(17189): item click: 0:full-hand
10-19 16:11:01.569: D/onItemClick(17189): item click: 1:postupka
10-19 16:11:02.729: D/onItemClick(17189): item click: 2:royal
10-19 16:11:04.659: D/onItemClick(17189): checked: true:maslo
10-19 16:11:06.839: D/onItemClick(17189): checked: true:mlieko
10-19 16:11:07.769: D/onItemClick(17189): checked: true:kvety pre svokru
10-19 16:11:10.409: D/onItemClick(17189): item click: 1
```

01Intro/ActiList.zip



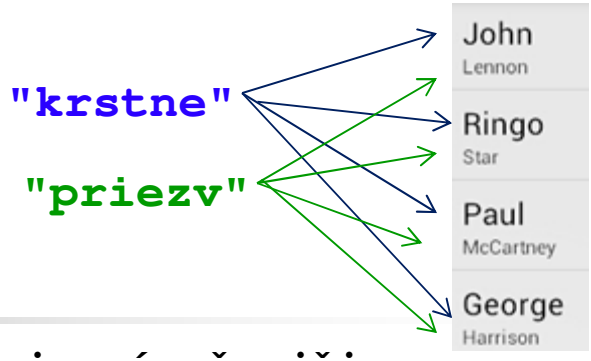
ListView 1

(simple_list_item_1, simple_list_item_checked)

```
String[] pList =          // ak dáta/zoznam ťaháme z Resources
        getResources().getStringArray(R.array.poker);
ListView lv1 = (ListView) findViewById(R.id.listView1);
// pre lv1 vytvoríme ArrayAdapter, poskytneme dáta pre ListView
// ListView Adapter prepojí data v zozname s ich view
lv1.setAdapter(new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1,
        pList));// to isté aj pre simple_list_item_checked
lv1.setOnItemClickListener(this);          // priviaž listener k lv1

-------- onItemClick interface vyžaduje onItemClick metódu
public void onItemClick(AdapterView<?> la, View v, int indx...
final String item = (String) la.getItemAtPosition(indx);
   Log.d("onItemClick", "item click: " + indx + ":" + item);
   CheckedTextView tv = (CheckedTextView) v;
   tv.toggle();  // zmení checked na unchecked, a vice-versa
   Log.d("onItemClick","checked:"+tv.isChecked()+":"+ item1);
```

01Intro/ActiList.zip

ListView 2
(simple_list_item_2)



Naplniť iný, napr. dvojriadkový ListView je náročnejšie

```
ArrayList<Map<String, String>> pairs = new ArrayList<Map<String, String>>();
HashMap<String, String> item1 = new HashMap<String, String>();
    item1.put("krstne", "John"); // 1.riadok 1.položky
    item1.put("priezv", "Lennon");        // 2.riadok 1.položky
    pairs.add(item1);                     // 1.položka zoznamu pairs
   .....                          // a ďalší „bítlsáci“ ...
ListView lv3 = (ListView) findViewById(R.id.listView3);
String[] from = {"krstne", "priezv" };// symbolické mená riadkov
int[] to = { android.R.id.text1, android.R.id.text2 };
//čo je text1,2 http://developer.android.com/reference/android/R.id.html

lv3.setAdapter(
        new SimpleAdapter(this, pairs, // context a zoznam riadkov
        android.R.layout.simple_list_item_2, from, to)); // šablóna
lv3.setOnItemClickListener(this);
```

Kód na slajde je zjednodušený, originál najdete v Layouts1.zip

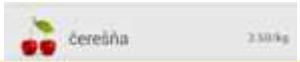
Rôzne preddefinované ListView
(prehľad)

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

Vlastný ListView

Myšlienka: musíme definovať náš vlastný ArrayAdapter

```
FruitArrayAdapter fruitArrayAdapter = new FruitArrayAdapter(
                ActivityList3.this,
                R.layout.listview_row_layout);
lv4.setAdapter(fruitArrayAdapter);  // ktorý podhodíme ListView
lv4.setOnItemClickListener(this);

fruitArrayAdapter.add(            // ktorý naplníme vlastnými objektami
   new Fruit(R.drawable.apple,"jablko","0.60/kg"));

public class Fruit {
   private int fruitImg;
   private String fruitName;
   private String price;
...                              // musí poskytovať getView:index->View
}                                                    1 ->
```



Kód na slajde je zjednodušený, originál najdete v Layouts.zip

ArrayAdapter

```
public class FruitArrayAdapter extends ArrayAdapter<Fruit> {
  private List<Fruit> fruitList = new ArrayList<Fruit>();
  static class FruitViewHolder {
        ImageView fruitImg;
        TextView fruitName, price;
  }
  public void add(Fruit object) {
        fruitList.add(object);
  }
  public View getView(int position, ...) {
      View row = ... R.layout.listview_row_layout ...
      row.setTag(viewHolder);
      Fruit fruit = getItem(position);
      viewHolder.fruitImg.setImageResource(fruit.getFruitImg());
      viewHolder.fruitName.setText(fruit.getFruitName());
      viewHolder.price.setText(fruit.getPrice());
      return row;
  }
```

1 ->

Kód na slajde je zjednodušený, originál najdete v Layouts1.zip