



# Pokračovanie

ListView/RecyclerView, Intent  
Canvas

Peter Borovanský  
KAI, I-18

MS-Teams: [2sf3ph4](#), [List](#), [github](#)

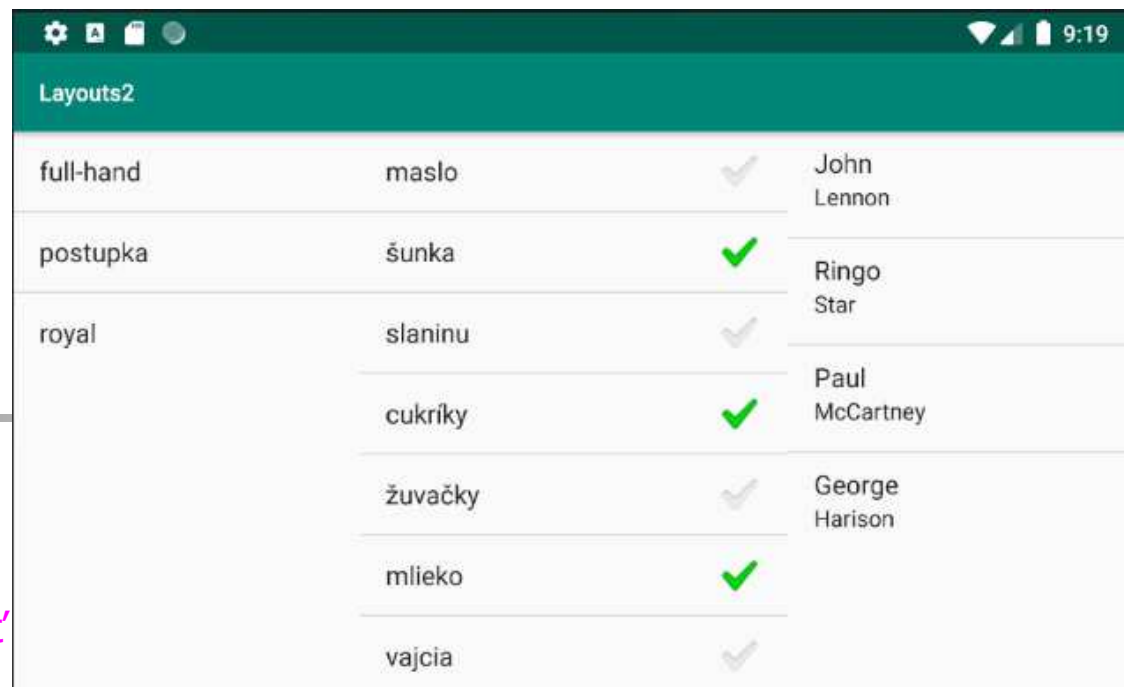
borovan 'at' ii.fmph.uniba.sk

# ListView

(variabilita)

ListView a ListActivity zobrazujú zoznam položiek a môžu mať

- preddefinovaný štýl
- môžu/nemusia sa nám páčiť
- ale sú *ready to use*
- sú jednoduché
- ale jemne zastaralé - legacy
- user defined
  - narobíme sa pri ich definícii
  - to už radšej RecyclerView...



full-hand	maslo	✓	John Lennon
postupka	šunka	✓	Ringo Star
royal	slaninu	✓	Paul McCartney
	cukríky	✓	George Harison
	žuvačky	✓	
	mlieko	✓	
	vajcia	✓	

Rôzne inštancie ListView

- `simple_list_item_1`,
- `simple_list_item_activated_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
```

# ListView

(simple\_list\_item\_1)

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

simple\_list\_item\_1

simple\_list\_item\_checked

**ListView** potrebuje inštanciu implementujúcu interface **ListAdapter**, napr. triedy **ArrayAdaper**, ktorý slúži na vykreslenie položiek **ListView**

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

```
// listView1.choiceMode =
    ListView.CHOICE_MODE_MULTIPLE
```

```
// kontext ListView
// štýl- jednoriadkový
// hodnoty
```

```
res/values/strings.xml
<string-array name="poker">
    <item>full-hand</item>
    <item>postupka</item>
    <item>royal</item>
</string-array>
```

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

# ListView

(simple\_list\_item\_checked)

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

simple\_list\_item\_1

simple\_list\_item\_checked

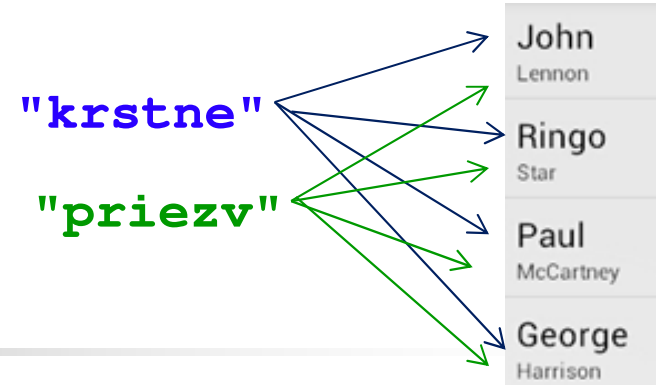
```
// nákup - checked box list view
listView2.adapter = ArrayAdapter (
    this,
    android.R.layout.simple_list_item_checked, //2-riadkový štýl
    resources.getStringArray(R.array.nakup)
)

listView2.setOnItemClickListener {
    adapterView, view, index, 1 ->
        val hodnota = adapterView.getItemAtPosition(index)
        (view as CheckedTextView).toggle() // prekresli
        Log.i(TAG, "check click: $index:$hodnota")
}
```

```
res/values/strings.xml
<string-array name="nakup">
    <item>maslo</item>
    <item>šunka</item>
    <item>slaninu</item>
    <item>cukríky</item>
    ...
    <item>kvety pre svokru</item>
</string-array>
```

# ListView 2

(simple\_list\_item\_2)



Naplniť iný, napr. dvojriadkový **ListView** je náročnejšie

```
val pairs = listOf( // hodnoty sú zoznamy máp typu 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 typu List<? extends Map<String, ?>>
    android.R.layout.simple_list_item_2, // formát/štýl ListView
    arrayOf("krstne", "priezv"), // zoznam kľúčov
    arrayOf(android.R.id.text1, android.R.id.text2) // štýl riadkov
        .toIntArray()
)

listView3.setOnItemClickListener {
    adapterView, view, index, l ->
    val hodnota = adapterView.getItemAtPosition(index)
    Log.i(TAG, "beatles click: $index:$hodnota:" +
        "${(hodnota as Map<String, String>)[\"krstne\"]}
}
```

# Rôzne preddefinované ListView

(prehľad)

The image displays four screenshots of Android applications demonstrating different ListView styles:

- SimpleListItemSingleChoice:** Shows a list with categories: Vegetables, Fruits (selected), Flower Buds, Legumes, Bulbs, and Tubers. Each item has a radio button.
- TwoLineListItem:** Shows a list with categories: Vegetables (65 items), Fruits (17 items), Flower Buds (5 items), Legumes (33 items), Bulbs (18 items), and Tubers (43 items). Each item has a two-line layout with a category name and a count.
- ActivityListItem:** Shows a list with categories: Vegetables, Fruits, Flower Buds, Legumes, Bulbs, and Tubers. Each item has a small icon and a category name.
- SimpleExpandableListItem:** Shows a list with categories: Vegetables (expanded), Squash (33 units), Zucchini (6 units), Carrots (20 units), Fruit, and Herbs. The Vegetables category is expanded, showing its sub-items.

```
java.lang.Object
└─ android.view.View
    └─ android.widget.ImageView
        └─ android.widget.ImageButton
            └─ com.google.android.material.floatingactionbutton.FloatingActionButton
```

# Material Design

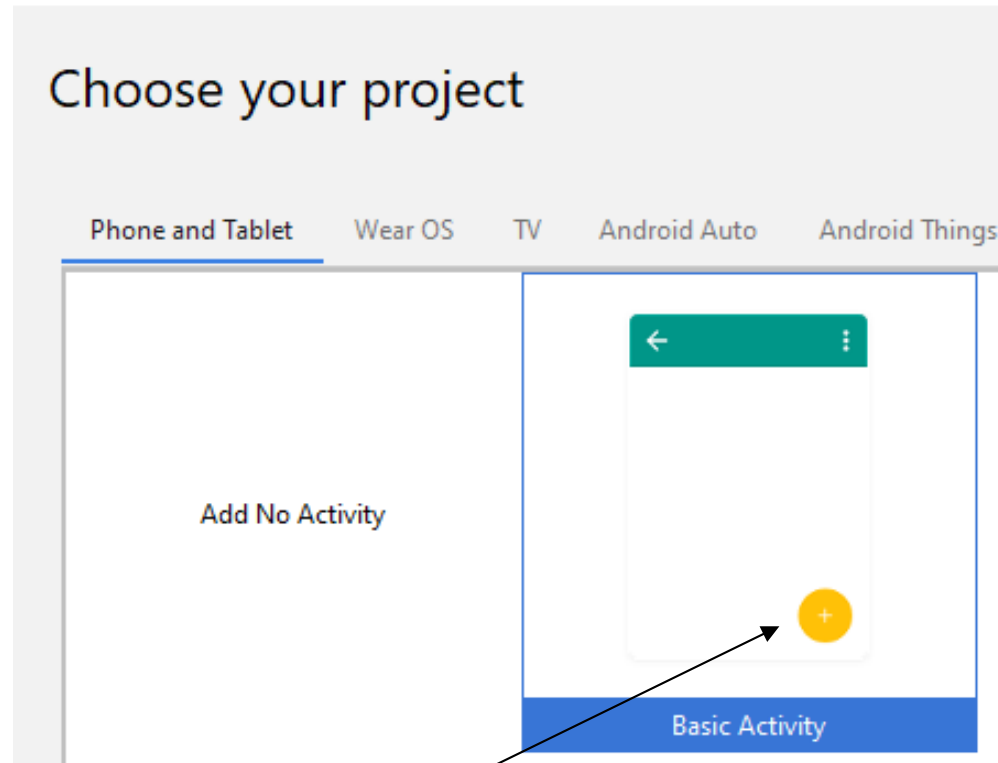
(ako a kde začať <https://material.io/>)

Kotlin Android Fundamentals: 10.2  
Material Design, dims, and colors

<https://codelabs.developers.google.com/codelabs/kotlin-android-training-material-design-dimens-colors/index.html?index=..%2F..android-kotlin-fundamentals#0>

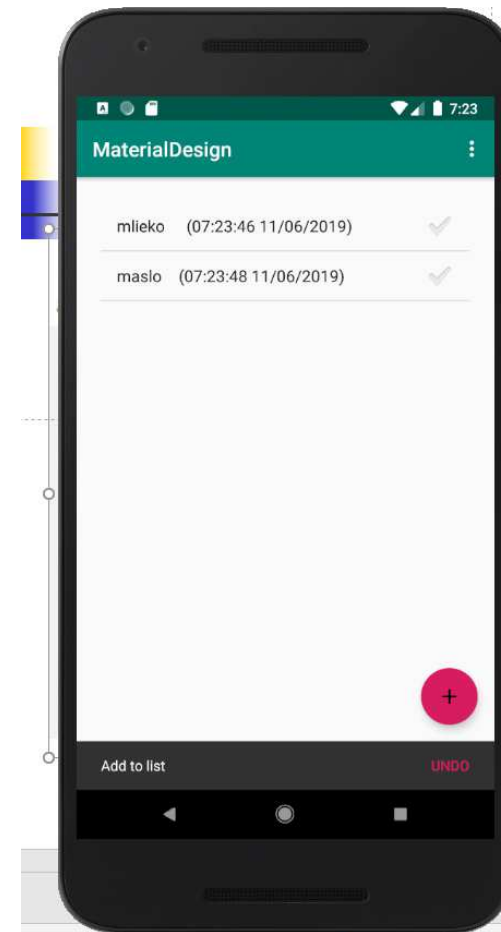
<https://developer.android.com/guide/topics/ui/look-and-feel>

Create New Project



Floating Action Button

`<com.google.android.material.floatingactionbutton.FloatingActionButton`

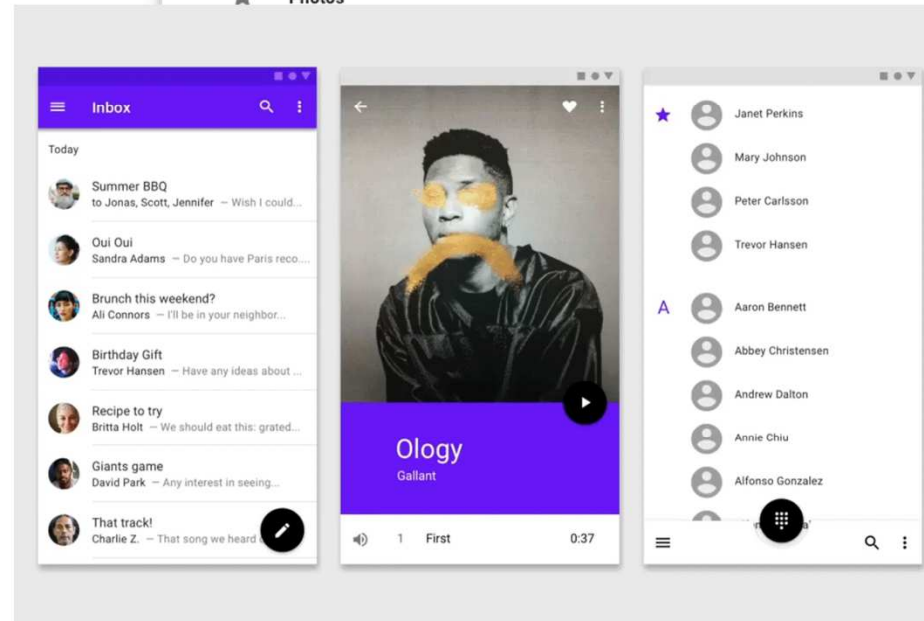
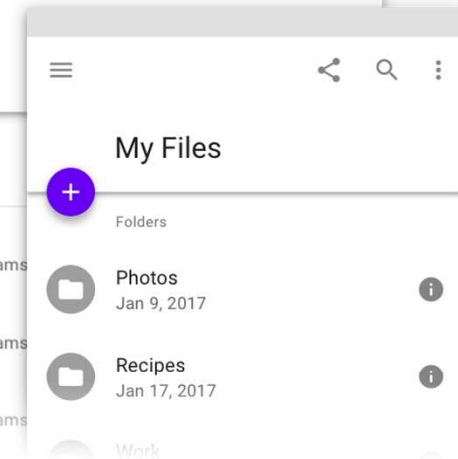
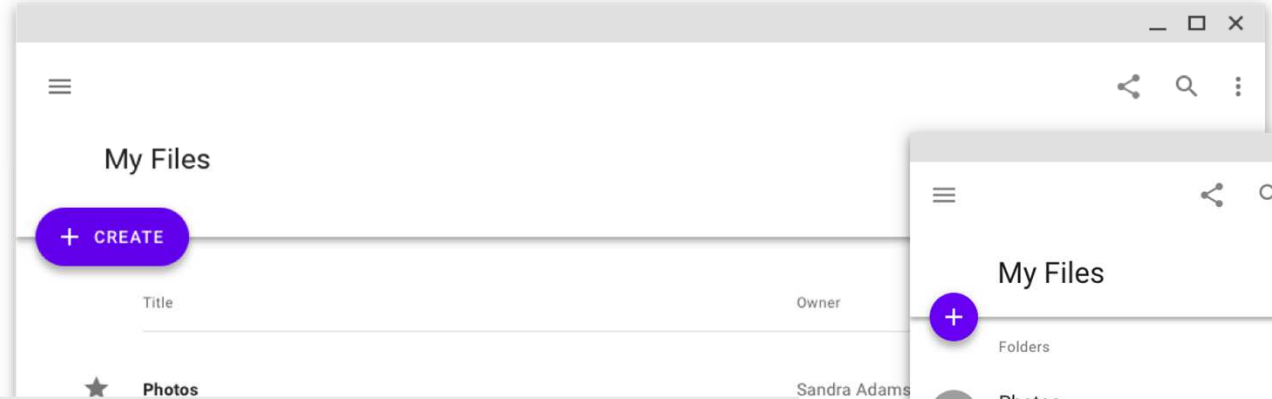
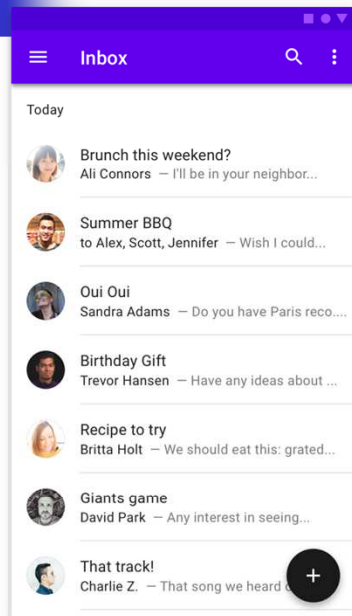


Project: MaterialDesign.zip

## 52. Working with the Floating Action Button and Snackbar

# Material Design

<https://m2.material.io/design/introduction>



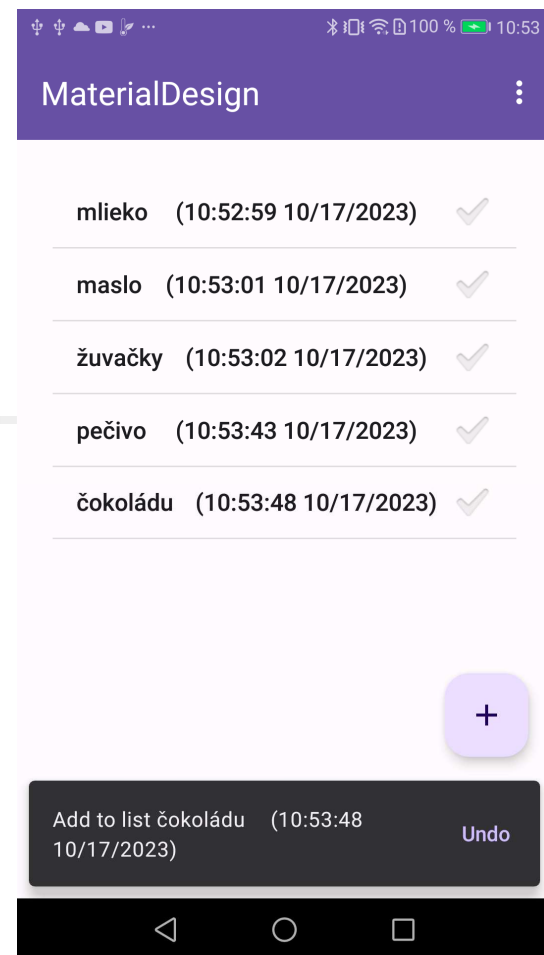


# TODO List

## ListView

```
adapter = ArrayAdapter(  
    this@MainActivity,  
    android.R.layout.simple_list_item_checked,  
    listItems  
)  
contentMain.listView.adapter = adapter  
contentMain.listView.setOnItemClickListener {  
    adapterView, view, index, l ->  
        val hodnota = adapterView.getItemAtPosition(index)  
        (view as CheckedTextView).toggle()  
    }  
}
```

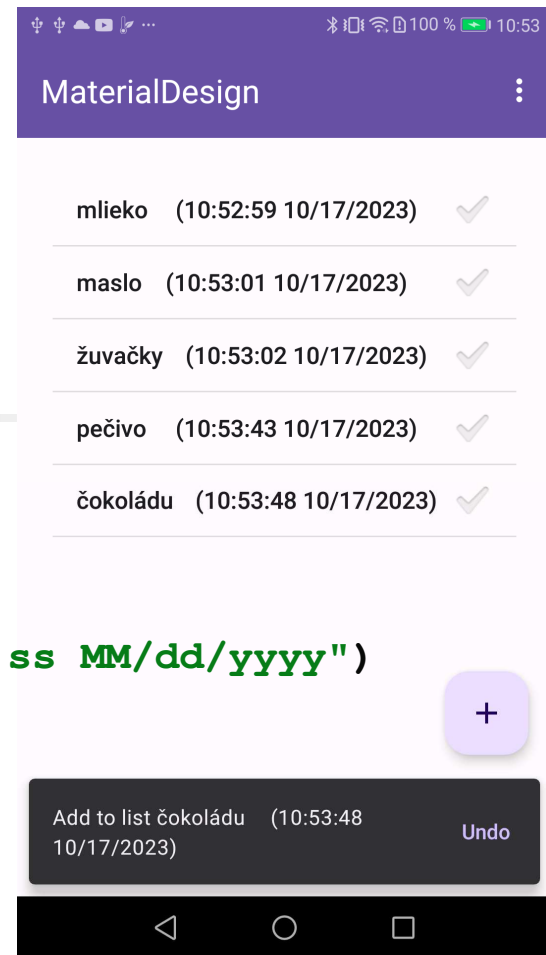
```
<com.google.android.material.floatingactionbutton.FloatingActionButton  
    android:id="@+id/fab"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="bottom|end"  
    android:layout_margin="@dimen/fab_margin"  
    app:srcCompat="@drawable/ic_add_entry"  
    android:contentDescription="pridaj polozku" />
```



# TODO List

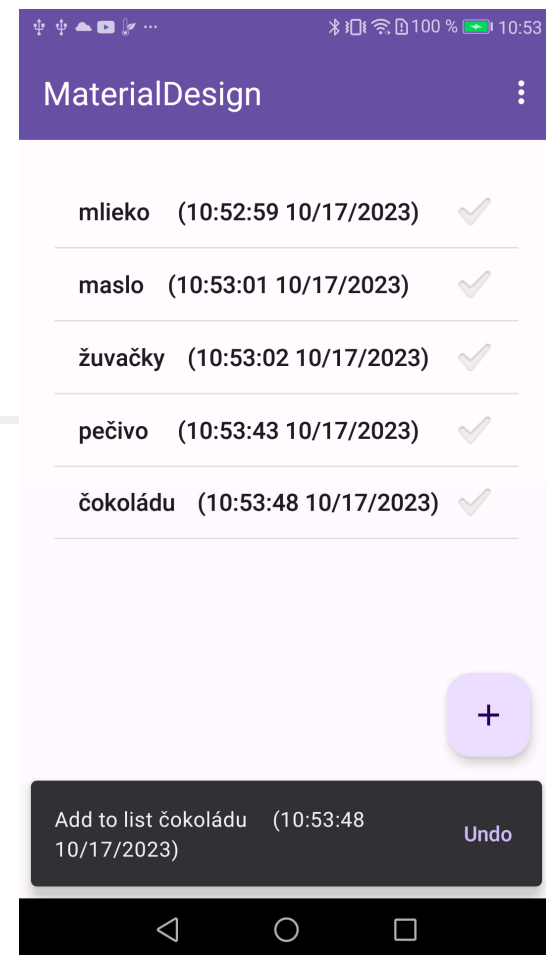
setOnClickListener

```
fab.setOnClickListener {  
    val format: SimpleDateFormat = SimpleDateFormat("HH:mm:ss MM/dd/yyyy")  
    var itemText = ""  
    try {  
        itemText =  
            resources.getStringArray(R.array.potraviny)  
                [listItems.size] + "\t\t\t({{  
                    format.format(  
                        Date()  
                    })  
                }})"  
    } catch (e: Exception) {  
        itemText = "no more items"  
    }  
    listItems.add(itemText)  
    adapter.notifyDataSetChanged()  
}
```



# TODO List

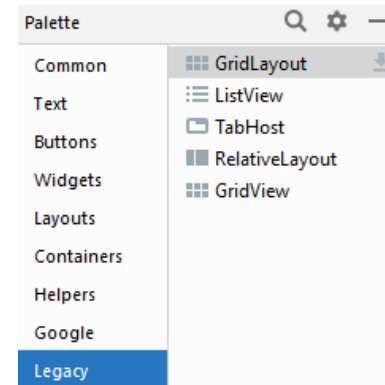
SnackBar



```
SnackBar.make(it, "Add to list $itemText", SnackBar.LENGTH_LONG)
    .setAction("Undo",
        {
            if (listItems.size > 0) {
                listItems.removeAt((listItems.size - 1))
            }
            adapter.notifyDataSetChanged()
            SnackBar.make(it, "last removed", SnackBar.LENGTH_SHORT)
                .setAction("Action", null).show()
        }
    ).show()
```



# RecyclerView



- podlieha MVVM návrhovému vzoru
- efektívnejšia forma ako **ListView**, renderuje len políčka zobrazené, scroluje, ...

ViewModel – dáta, ktoré sa majú zobrazovať v jednom políčku

```
data class ItemsViewModel(val text: String) { }
```

RecyclerView.Adapter – boilerplate code

```
class CustomAdapter(private val mList: List<ItemsViewModel>) :
    RecyclerView.Adapter<CustomAdapter.ViewHolder>() {
    // vytvorí view zodpovedajúce jednému riadku zoznamu, bez hodnôt, template riadku
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder { ... }
    // previaže prázdny holder s dátami adaptéra, pomocou vstupu mList
    override fun onBindViewHolder(holder: ViewHolder, position: Int) { ... }
    // počet riadkov zobrazených v zozname
    override fun getItemCount(): Int { ... }
    // jeden riadok zoznamu
    class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val textView = itemView.findViewById<TextView>(R.id.textView)
        init {
            itemView.setOnClickListener {
                Log.i("TAG", "item selected ${textView.text}")
            }
        }
    }
}
```

# RecyclerView

detail adaptéra

```
res/layouts/R.layout.row
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <TextView
        android:id="@+id/textView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="TextView" />
</LinearLayout>
```

```
data class ItemsViewModel(val text: String) { }
```

```
class CustomAdapter(private val mList: List<ItemsViewModel>) :
    RecyclerView.Adapter<CustomAdapter.ViewHolder>() {
    // vytvorí view zodpovedajúce jednému riadku zoznamu, bez hodnôt, template riadku
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        val view = LayoutInflater.from(parent.context)
            .inflate(R.layout.row, parent, false)
        return ViewHolder(view)
    }
    // previaže prázdny holder s dátami adaptéra, mList
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        holder.textView.text = mList[position].text
    }
    // počet riadkov
    override fun getItemCount(): Int {
        return mList.size
    }

    class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val textView = itemView.findViewById<TextView>(R.id.textView)
        init {
            itemView.setOnClickListener { Log.i("TAG", "selected ${textView.text}")
                Toast.makeText(itemView.context, "item selected ${textView.text}",
                    Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```



55. An Android RecyclerView and  
CardView Tutorial

# CardView

detail riadku

```
res/layouts/R.layout.row
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    app:cardCornerRadius="12dp"
    app:cardElevation="3dp"
    app:contentPadding="4dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="1dp">

        <TextView
            android:id="@+id/textView"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textSize="32sp"
            android:background="@color/black"
            android:textColor="@color/white"
            android:text="TextView" />

    </LinearLayout>
</androidx.cardview.widget.CardView>
```

RecycleV

peter

pavel

anna

# RecyclerView

```
layout_main.xml
<LinearLayout
    android:layout_width="match_parent"
    android:orientation="horizontal"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
```

```
//-- recyclerView inicializácia
```

```
binding.rv.layoutManager = LinearLayoutManager(this)
```

```
binding.rv.adapter = CustomAdapter(
```

```
    listOf(
```

```
        ItemsViewModel("peter"),
```

```
        ItemsViewModel("pavel"),
```

```
        ItemsViewModel("anna")
```

```
    ))
```

LayoutManager

Item 0

Item 1

Item 2

LinearLayoutManager

0 1

2 3

4 5

GridLayoutManager

0 1

2 3

4 5

StaggeredGridLayoutManager

<https://magdamiu.com/2020/12/29/recyclerview-android/>

## 57. An Overview of Android Intents

# Intent

(filter)

vid' AndroidManifest: <intent-filter> hovorí, na aký intent aktivita reaguje

```
<activity android:name=".MainActivity">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

Spustí sa ako prvá

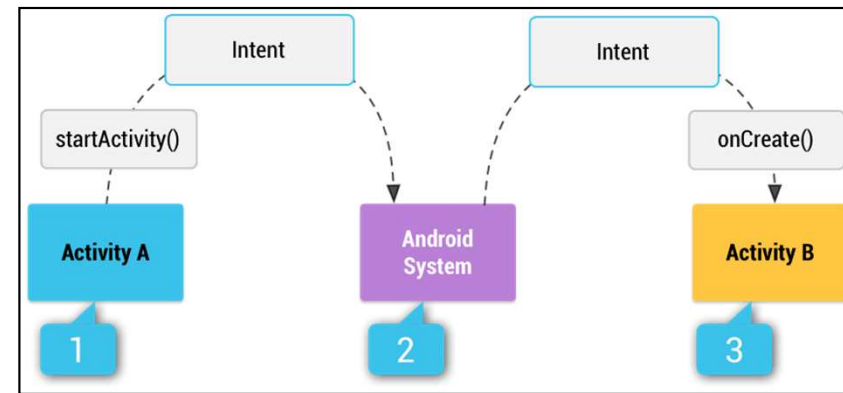
Aktivita je vlastne jedno-obrazovková aplikácia

```
<activity android:name=".ListActivity">
  <intent-filter>
    <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</activity>
```

nespustí sa



# Intent



## Scenár:

- aktivita A vytvorí objekt Intent, na ktorý má Aktivita B nastavený intent-filter
- zavolá ho cez `startActivity(intent..)`, resp. `startActivityForResult(...)`
- systém hľadá aplikáciu, ktorá má daný nastavený intent-filter
- ak ich existuje viacero, vyskočí dialóg na výber správnej aplikácie
- zavolá sa `onCreate` aktivity v nájdennej aplikácii (B)

**Explicitné** intenty popisujú plne kvalifikované meno triedy, napr. `com.example.Pokus`

**<intent-filter>**

```
<action android:name="com.example.Pokus" />
```

```
<category android:name="android.intent.category.DEFAULT" />
```

**</intent-filter>**

**Implicitné** intenty popisujú všeobecnú akciu, napr. `ACTION_IMAGE_CAPTURE` alebo `ACTION_VIDEO_CAPTURE`

Ich zoznam, popis je tu: <https://developer.android.com/guide/components/intents-common>

<https://developer.android.com/guide/components/intents-filters>

# Intent

(startActivity)

## Layouts

Grid layout

Frame layout

Relative layout

Constraint layout

Linear layout

List layout

Simple List layout

```
listViewID.adapter = ArrayAdapter<String>(
    this, android.R.layout.simple_list_item_1,
    arrayListOf("Grid layout", "Frame layout", "Relative layout",
        "Constraint layout")) // dáta nepatria do kódu, ale .xml
resources.getStringArray(R.array.activities)
```

```
listViewID.setOnItemClickListener {
    adapterView, view, index, l ->
    Log.i("LISTPICK",
        "click: $index:${adapterView.getItemAtPosition(index)}")
    if (index < klasy.size)
        startActivity(Intent(this@MainActivity, klasy[index]))
} // čo je this@MainActivity - this z vonkajšieho scope
```

```
val klasy:Array<Class<out AppCompatActivity>>= arrayOf(
    GridLayoutActivity::class.java,
    FrameLayoutActivity::class.java,
    ...
    MainActivity::class.java
)
```

```
<activity>
    android:name=".GridLayoutActivity"
    android:label="@string/title_activity_grid_layout"
    <intent-filter>
        <action android:name="com.example.layouts2.GridLayoutActivity" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

# Reflection & Intent

ACTION\_VIEW

Added in API level 1

```
public static final String ACTION_VIEW
```

Activity Action: Display the data to the user. This is the most common action performed on data -- it is the generic action you can use on a piece of data to get the most reasonable thing to occur. For example, when used on a contacts entry it will view the entry; when used on a mailto: URI it will bring up a compose window filled with the information supplied by the URI; when used with a tel: URI it will invoke the dialer.

Input: `getData()` is URI from which to retrieve data.

Output: nothing.

- intent vytvoríte
  - `Intent(context, Class<*>)`
    - `Intent(this@MainActivity, GridLayoutActivity::class.java)`
  - `Intent(reťazec, uri)`,
    - `Intent(MediaStore.ACTION_IMAGE_CAPTURE)`
    - `Intent(Intent.ACTION_VIEW, Uri.parse("https://google.com"));`
    - `Intent(Intent.ACTION_SEND, Uri.parse("mailto:"));`
- Reflection model v Kotlin je iný ako v Java,
  - Kotlin má triedu `KClass<>`, Java má `Class<>`
- `val kotlinClass: KClass<GridLayoutActivity> = GridLayoutActivity::class`
- na získanie Java class referencie, treba použiť property `.java`
- `val javaClass: Class<GridLayoutActivity> = GridLayoutActivity::class.java`
- `val i = Intent(this@MainActivity, javaClass)`
- viac o Kotlin reflection  
<https://kotlinlang.org/docs/reference/reflection.html>



# List project

---

V ďalšom uvidíme sériu rôznych nezávislých aktivít, ktoré ilustrujú:

- intro\_activity
  - logo, intent, Countdown/Timer, MediaPlayer
- email\_activity
  - listView, intent.putExtra, startActivityForResult
- canvas\_activity
  - canvas/view – Draw, MultiTouch, onTouch, Option & Context Menu
- pisky\_activity
  - piškvorky, začiatok aj koniec jednoduchej hry
- login\_activity
  - ukladanie informácie pomocou SharedPreferences



# Intent - filter

- CATEGORY\_BROWSABLE - ovláda web browser
- CATEGORY\_LAUNCHER - ovláda spúšťač aplikácie
- *android.intent.action.MAIN - vstupný bod programu*

```
<intent-filter>  
    <action android:name="android.intent.action.MAIN" />  
    <category android:name="android.intent.category.LAUNCHER" />  
</intent-filter>
```

- CATEGORY\_DEFAULT – startActivity/startActivityForResults

```
<intent-filter>  
    <action android:name="com.example.list.CanvasActivity" />  
    <category android:name="android.intent.category.DEFAULT" />  
</intent-filter>
```

spustenie:

```
startActivity(
```

```
    Intent(this@IntroActivity, MainActivity::class.java))
```

ak máme:

```
class MainActivity : AppCompatActivity() {
```



# Reflexivita

---

Aby sme nemuseli mať konštantu ako pole všetkých tried, trieda sa dá vyrobiť z mena triedy pomocou reflexívneho volania statickej metódy `Class.forName`

```
class MainActivity : AppCompatActivity() {
    ...
    listView1.setOnItemClickListener {
        adapterView, view, index, l ->
            val hodnota = adapterView.getItemAtPosition(index)
            Log.i(TAG, "list item click: $index:$hodnota")
            val klasa: KClass<Any>
                = Class.forName("com.example.list.$hodnota").kotlin
            //val intent = Intent(this, IntroActivity::class.java)
            val qname: String? = klasa.qualifiedName
            Log.i(TAG, "class name: $qname")
            val intent = Intent(qname)
            startActivity(intent)
    }
}
```



# IntroActivity

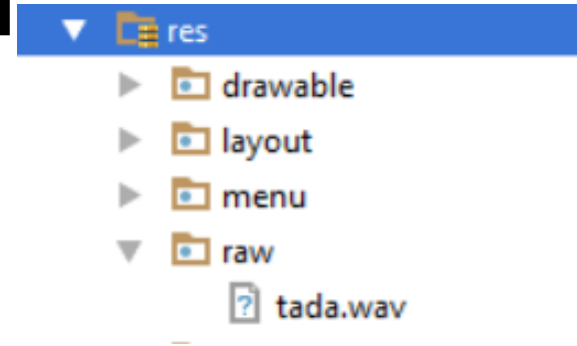
(Intent, timer)

IntroActivity – CountdownTimer odpočítavajúci čas pre úvodné logo+.mp3

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_intro)  
  
    // timer odpočítavajúci 4s.  
    object: CountdownTimer(4000, 1000) {  
        override fun onTick(millisUntilFinished: Long) {}  
        override fun onFinish() {  
            Log.i(TAG, "go back to mainActivity")  
            startActivity(  
                Intent(this@IntroActivity, MainActivity::class.java)  
            )  
        }  
    }.start()
```

# MediaPlayer

(lokálne – adresár raw)



tada.mp3 [.wav] uložíme do project/res/raw  
 ... a bude zakompilovaná do apky, a zazipovaná do zipky ☺

```
lateinit var mp : MediaPlayer           // globálna premenná
// ak je muzička lokálna, v res/raw/tada.wav
mp = MediaPlayer.create(this, R.raw.tada)
mp.isLooping = false
mp.start()
```

```
override fun onPause() {                // ak je IntroActivity pauzovaná, keď
    super.onPause() // odštartujeme com.example.actilist.MainActivity
    mp.release()    // uvoľníme MediaPlayer objekt, mp
    finish()        // akonáhle sa rozbehne MainActivity,
                    // IntroActivity zanikne
                    // to rieši aj navigáciu, problém s back buttonom
}
```

Media player má problémy v niektorých emulátoroch

<https://stackoverflow.com/questions/59073433/building-a-simple-mediaplayer-with-kotlin-in-android-studio-c>

Project: List.zip





# MediaPlayer

(onPreparedListener)

iná možnosť, tada.mp3 je prístupná niekde na sieti, dotiahneme ju a zahráme

*// problém:apka musí deklarovať, že chce prístup na internet*

```
lateinit var mp : MediaPlayer
try { // ak je muzička na webe, jej dotiahnutie môže niečo trvať
    val uri = Uri.parse("https://dai.fmph.uniba.sk/courses/VMA/wave.mp3")
    mp.setAudioStreamType(AudioManager.STREAM_MUSIC)

    mp.setOnPreparedListener { mp.start() }

    mp.setDataSource(getApplicationContext(), uri)
    mp.prepare() // tu sa spustí doťahovanie súboru
} catch (e:IOException) {
    e.printStackTrace()
    Toast.makeText(this, "file error", Toast.LENGTH_SHORT).show()
}
```

do AndroidManifest.xml

treba deklarovať povolenie aplikácie prístupu na internet

```
<uses-permission android:name="android.permission.INTERNET" />
```

<https://dai.fmph.uniba.sk/courses/VMA/wave.mp3>

Project: List.zip

# MediaPlayer

(na SD-karte, resp. v internej pamäti)

## AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Môžeme sa skúšať triať do správnej cesty muziky, obrázku, či súboru:

```
mp.setDataSource("/mnt/sdcard/Music/tada.wav")
mp.setDataSource("/mnt/sdcard/Music/wave.mp3")
mp.setDataSource("/storage/sdcard0/Music/wave.mp3")
mp.setDataSource("/Removable/SD/Music/wave.mp3")
```

// ale správna cesta k prístupu k Music je cez root external storage

```
val filePath = Environment.getExternalStorageDirectory().toString()
    + "/Music/tada.wav"
```

```
Log.i(TAG, filePath) // vždy si zalogujte cestu,
                     // aby ste vedeli, kde súbor hľadá
```

```
mp = MediaPlayer()
mp.setDataSource(filePath) // hneď viete, prečo to nehrá..
mp.setOnPreparedListener { mp.start() }
mp.prepare()
```



# SoundPool

```
var audioAttrib = AudioAttributes.Builder()
    .setUsage(AudioAttributes.USAGE_GAME)
    .setContentType(AudioAttributes.CONTENT_TYPE_SONIFICATION)
    .build()
val soundPool = SoundPool.Builder()
    .setAudioAttributes(audioAttrib)
    .setMaxStreams(10)
    .build()
```

```
class SoundPlayer(context: Context) {
    private val soundPool: SoundPool =
        SoundPool(10, AudioManager.STREAM_MUSIC, 0)
    init {
        try {
            val assetManager = context.assets
            var descriptor: AssetFileDescriptor // adresár assets
                = assetManager.openFd("shoot.ogg") // .mp3, .wav,
            shootID = soundPool.load(descriptor, 0) // toto trvá...
        } catch (e: IOException) {
            Log.e("SoundPlayer", "sound file not found")
        }

        soundPool.setOnLoadCompleteListener ({ // keď load skončil
            soundPool, sampleID, status -> // môžeme zahrat'
                soundPool.play(sampleID, 1f, 1f, 1, 1, 1f) })
    }
}
```



# EmailActivity

(data do intentu, startActivityForResult s callbackom)

```
val emailString = edtEmail.text.toString() // dáta z formulára
val subjectString = edtSubject.text.toString()
val bodyString = edtBody.text.toString()

Toast.makeText(this@EmailActivity, "posielam mail",
    Toast.LENGTH_LONG).show()

val intent = Intent(android.content.Intent.ACTION_SEND) // SEND

intent.type = "text/plain"
intent.putExtra(android.content.Intent.EXTRA_SUBJECT, subjectString)
intent.putExtra(android.content.Intent.EXTRA_EMAIL,
    arrayOf(emailString) ) // pole adresátov
intent.putExtra(android.content.Intent.EXTRA_TEXT, bodyString)

// startActivity(intent)
startActivityForResult(intent, REQUEST_SEND_EMAIL)

private val REQUEST_SEND_EMAIL = 777
```



# EmailActivity

(onActivityResult = callback)

---

```
private val REQUEST_SEND_EMAIL = 777

override fun onActivityResult( // CALLBACK pre startActivityForResult
    requestCode: Int,
    resultCode: Int,
    data: Intent?) {

    // Check which request we're responding to
    if (requestCode == REQUEST_SEND_EMAIL) {
        Toast.makeText(
            this@EmailActivity, "email poslany, alebo aj nie..",
            Toast.LENGTH_SHORT
        ).show();
    }
}
```



# PhotoActivity

(data z intentu)

Princíp intent-`startActivityForResult` spolu s `onActivityResult` ešte raz:

```
val takePictureIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
    // test, či existuje komponent, ktorý spracuje intent  
if (takePictureIntent.resolveActivity(getPackageManager()) != null) {  
    Toast.makeText(this@PhotoActivity,  
        "smile ... taking picture", Toast.LENGTH_LONG).show();  
    startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);  
} else {  
    Toast.makeText(this@PhotoActivity,  
        "sorry ... no picture", Toast.LENGTH_LONG).show();  
}  
  
private val REQUEST_IMAGE_CAPTURE = 690
```



# PhotoActivity

## (data z intentu)

Standard Intent action that can be sent to have the camera application capture an image and return it.

The caller may pass an extra EXTRA\_OUTPUT to control where this image will be written.

If the EXTRA\_OUTPUT is not present, then a small sized image is returned as a Bitmap object in the extra field. This is useful for applications that only need a small image.

If the EXTRA\_OUTPUT is present, then the full-sized image will be written to the Uri value of EXTRA\_OUTPUT.

V callback onActivityResult získavame z indentu data/odfotený obrázok ako Bitmap:

```
private val REQUEST_IMAGE_CAPTURE = 690

override fun onActivityResult(requestCode: Int,
    resultCode: Int, data: Intent?) {
    // Check which request we're responding to
    if (requestCode == REQUEST_IMAGE_CAPTURE &&
        resultCode == RESULT_OK) {
        Toast.makeText(this@PhotoActivity, "thanks ...",
            Toast.LENGTH_LONG).show()
        val extras = data?.getExtras()
        val imageBitmap = extras?.get("data") as Bitmap
        pictureImageView.setImageBitmap(imageBitmap)
    }
}
```

## 75. Making Runtime Permission Requests in Android

# Permissions

Povolenia (permissions) aplikácie slúžia na zabezpečenie:

- vašich privátnych dát (cez INTERNET, BLUETOOTH, ACCESS\_WIFI)
- ochranu súkromia (ACCESS\_FINE\_LOCATION, [READ/WRITE]\_CONTACTS)

Ak máte (Android  $\leq$  5.1 || target SDK  $<$  23), tak

`<uses-permissions />` sú staticky v AndroidManifest.xml,

Povolenia sa získavajú staticky pri inštalácii, ak ich užívateľ odmietne, aplikácia sa neinštaluje.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Inak (Android  $\geq$  6.0 && target SDK  $\geq$  23) aplikácia môže povolenia žiadať počas behu, podľa toho o akú službu práve ide (tzv. Runtime permissions). Ak užívateľ odmietne, aplikácia beží ďalej.

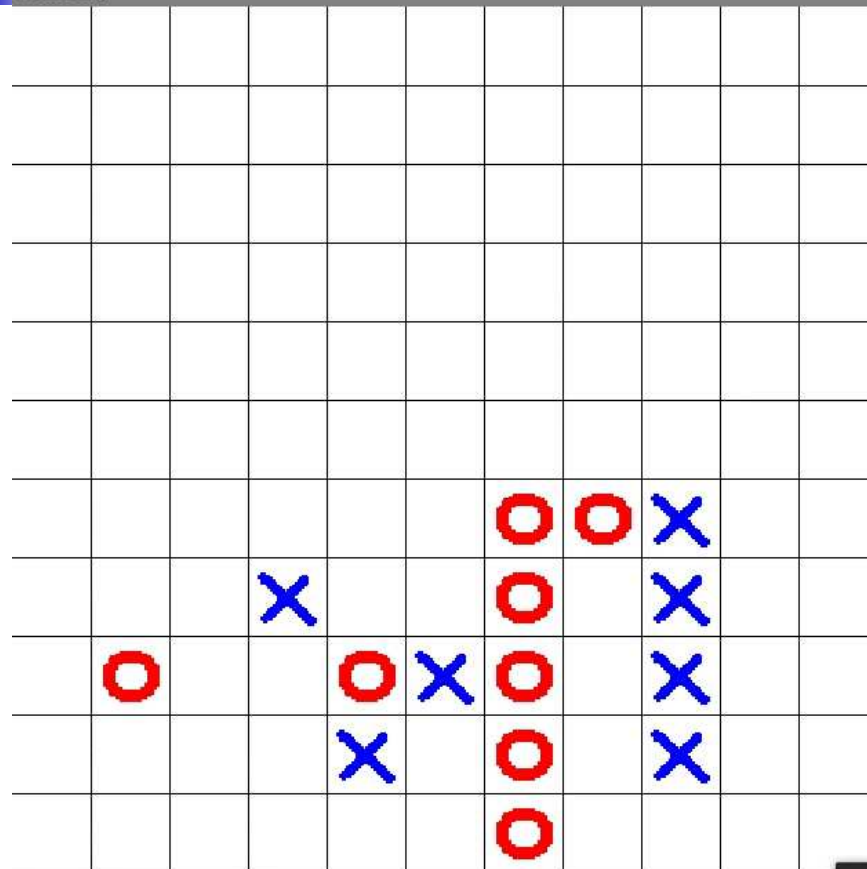
<https://developer.android.com/guide/topics/permissions/overview>



# Piškvorky

(logická hra v canvase)

Piškvorky



o vyhrali

Príklad logickej hry (hlavolamu), kde

- v pozadí nebeží žiadne vlákno,
- nehýbu sa postavičky,
- nemusíme pravidelne prekreslovať plochu, aby sme navodili ilúziu

☐ hry

☐ simulácie

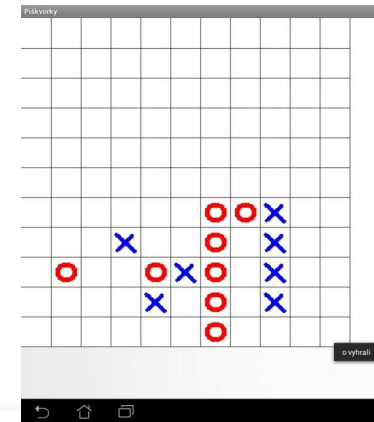
Na príklade Piškvorky ilustrujeme, že View má metódy:

- onTouch
- onDraw



# onTouch vo View

(onTouchEvent)



```
class PiskyView(context: Context, attrs: AttributeSet) :  
    View(context, attrs) {    // Piškvorky sú podtrieda View  
                                // načítanie bitmapy obrázkov, postavičiek  
  
    o_img = resources.getDrawable(R.drawable.o).toBitmap()  
    x_img = ContextCompat.getDrawable(context, R.drawable.x)!!.toBitmap()  
  
    override fun onTouchEvent(e: MotionEvent): Boolean {  
        if (e.action == MotionEvent.ACTION_DOWN) {  
            val iX = (e.x / cellSize).toInt()    // transformácia  
            val iY = (e.y / cellSize).toInt()    // pixlov na bunku  
            if (iX >= SIZE || iY >= SIZE) return true    // mimo hraciu dosku  
            if (playGround[iY][iX] == -1) {    // voľné políčko ?  
                playGround[iY][iX] = onTurn    // polož značku hráča  
                onTurn = 1 - onTurn    // na ťahu, a ide súper  
                invalidate()    // toto nakoniec prekreslí view  
                val winner = check(iX, iY)    // vyhodnotenie víťazov...  
                if (winner != -1)  
                    Toast.makeText(getContext(), "x vyhrali", Toast.LENGTH_LONG).show()  
            } else
```



# View/Activity

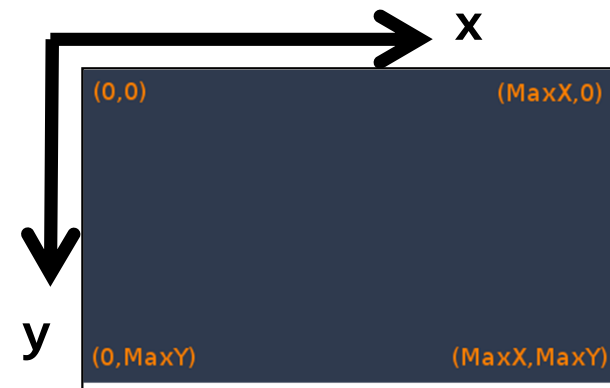
---

Každá podtrieda View má tieto metódy, ktoré môžeme predefinovať

```
class XXX(var context: Context, attrs: AttributeSet) :  
    View(context, attrs) {  
  
    override fun onSizeChanged(w: Int, h: Int, oldw: Int, oldh: Int){  
        super.onSizeChanged(w, h, oldw, oldh)  
    }  
  
    override fun onTouchEvent(event: MotionEvent?): Boolean {  
        return super.onTouchEvent(event)  
    }  
  
    override fun onKeyDown(keyCode: Int, event: KeyEvent?): Boolean{  
        return super.onKeyDown(keyCode, event)  
    }  
  
    override fun onDraw(canvas: Canvas?) {  
        super.onDraw(canvas)  
    }  
}
```

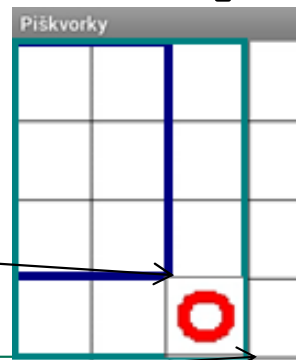
# onDraw vo View

(kreslenie do Canvas)



```
override protected fun onDraw(canvas: Canvas) { // paint()
    minSize = Math.min(getWidth(), getHeight()) - 2
    cellSize = minSize / SIZE // min.zrozmerov canvas/SIZE=10

    canvas.drawColor(Color.WHITE)
    val p = Paint()
    p.setColor(Color.BLACK)
    p.setStrokeWidth(1f)
    for (i in 1..SIZE) {
        canvas.drawLine(i*cellSize, 0f, i*cellSize, minSize, p)
        canvas.drawLine(0f, i*cellSize, minSize, i*cellSize, p)
    }
    for (y in 0 until SIZE) {
        for (x in 0 until SIZE) {
            canvas.drawBitmap(o_img, null,
                Rect(x1, y1, x2, y2),
                p)
        }
    }
}
```



# Maľovátko

(MotionEvent actions)

*Finger paint*

`event: MotionEvent` vám detailnejšie prezradí, aký event skutočne nastal

```
private val mPath: Path // android.graphics.Path
```

```
override protected fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)
    canvas.drawPath(mPath, mPaint)
}
```

Path je sekvencia

- úsečiek
- kvadratických a
- kubických kriviek

```
override fun onTouchEvent(event: MotionEvent): Boolean {
```

```
    val x = event.x
```

```
    val y = event.y
```

```
    when (event.action) {
```

```
        MotionEvent.ACTION_DOWN -> {
            startTouch(x, y) invalidate() }
        MotionEvent.ACTION_MOVE -> {
```

```
            moveTouch(x, y) invalidate() }
```

```
        MotionEvent.ACTION_UP -> {
            upTouch() invalidate() }
```

```
    }
```

```
    return true // event bol spracovaný, nešíri sa ďalej
```

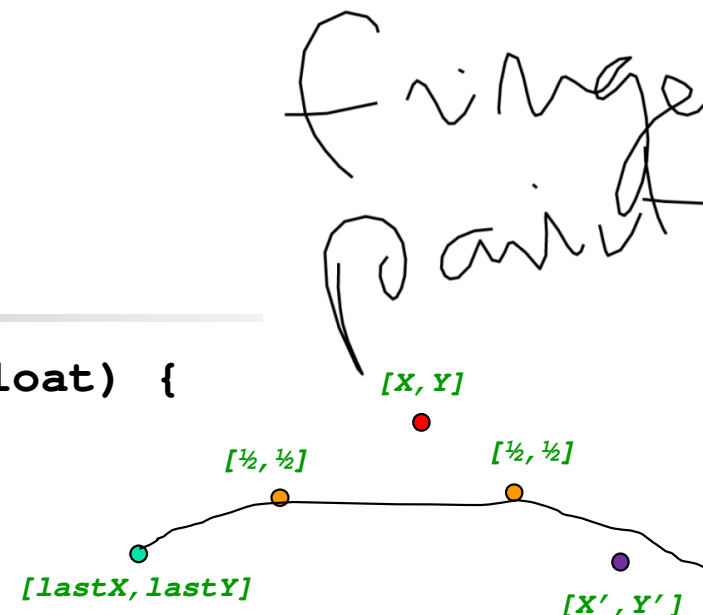
# Maľovátko

(bezier vs. linear - nebezier)

```
private fun startTouch(x: Float, y: Float) {
    mPath.moveTo(x, y)
    lastX = x
    lastY = y
}

private val TOLERANCE = 5f // minimálne epsilon pre pohyb
private fun moveTouch(x: Float, y: Float) {
    val dx = Math.abs(x - lastX)
    val dy = Math.abs(y - lastY)
    if (dx >= TOLERANCE || dy >= TOLERANCE) { // ak zmena bola >=
        mPath.lineTo(x, y); // kreslíme úsečku, dostaneme kostrbaté
        // úsečka z posledného bodu path do x,y
        // alebo
        mPath.quadTo(lastX, lastY, (x+lastX)/2, (y+lastY)/2)
        // kreslíme časť paraboly, aproximácia
        // kvadratickou krivkou...
        // to ale potrebujeme 3 body

        lastX = x
        lastY = y
    }
}
```



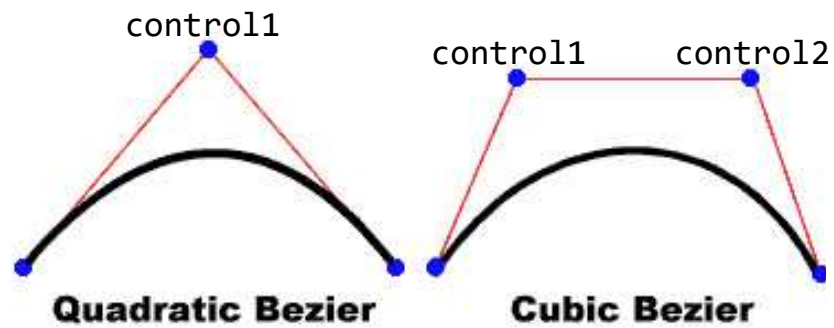
## Snímka 38

---

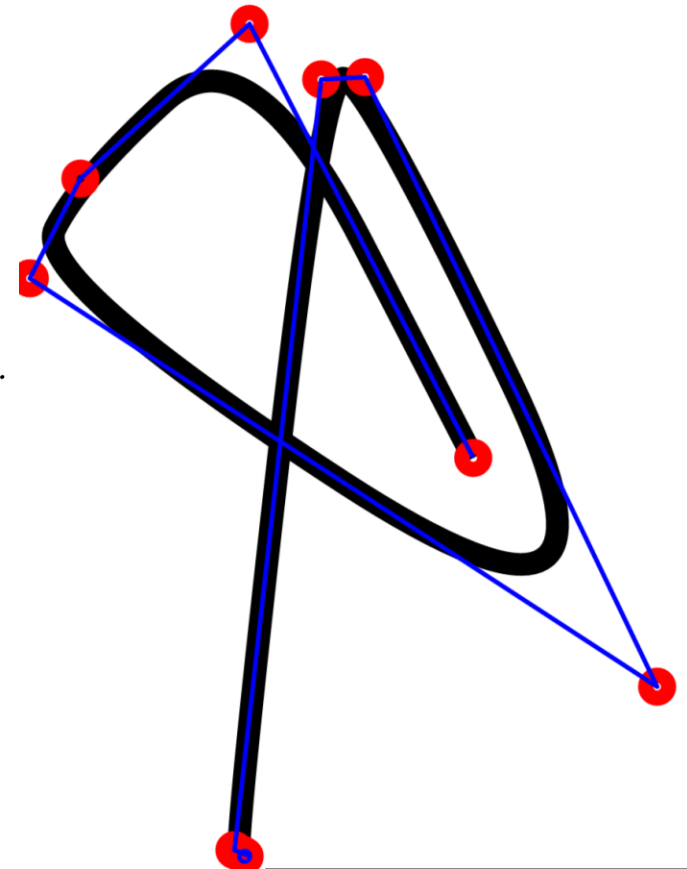
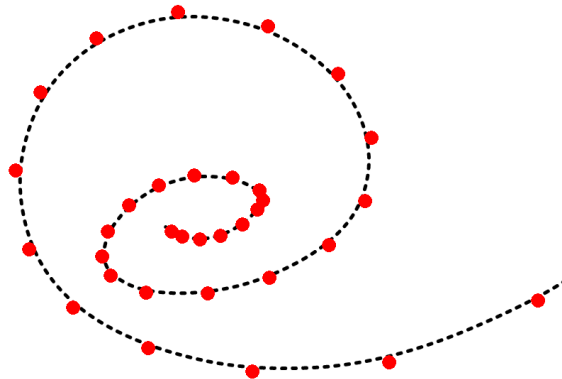
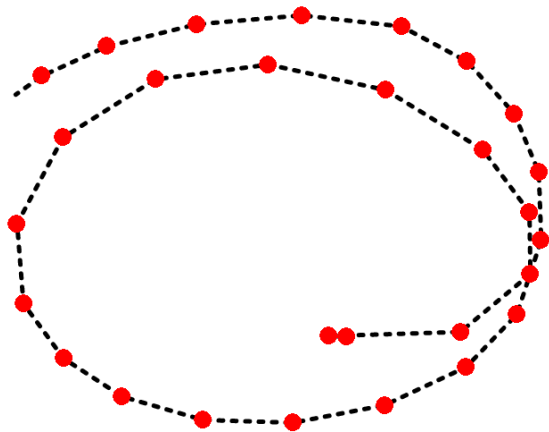
**PB1**

Peter Borovansky; 6. 11. 2019

# Bézier



- `lineTo(x,y)`
- `quadTo(controlX, controlY, x, y)`
- `cubeTo(controlX1, controlY1, controlX2, controlY2, x, y)`





# Hierarchia

(je dôležitá pre poriadok)

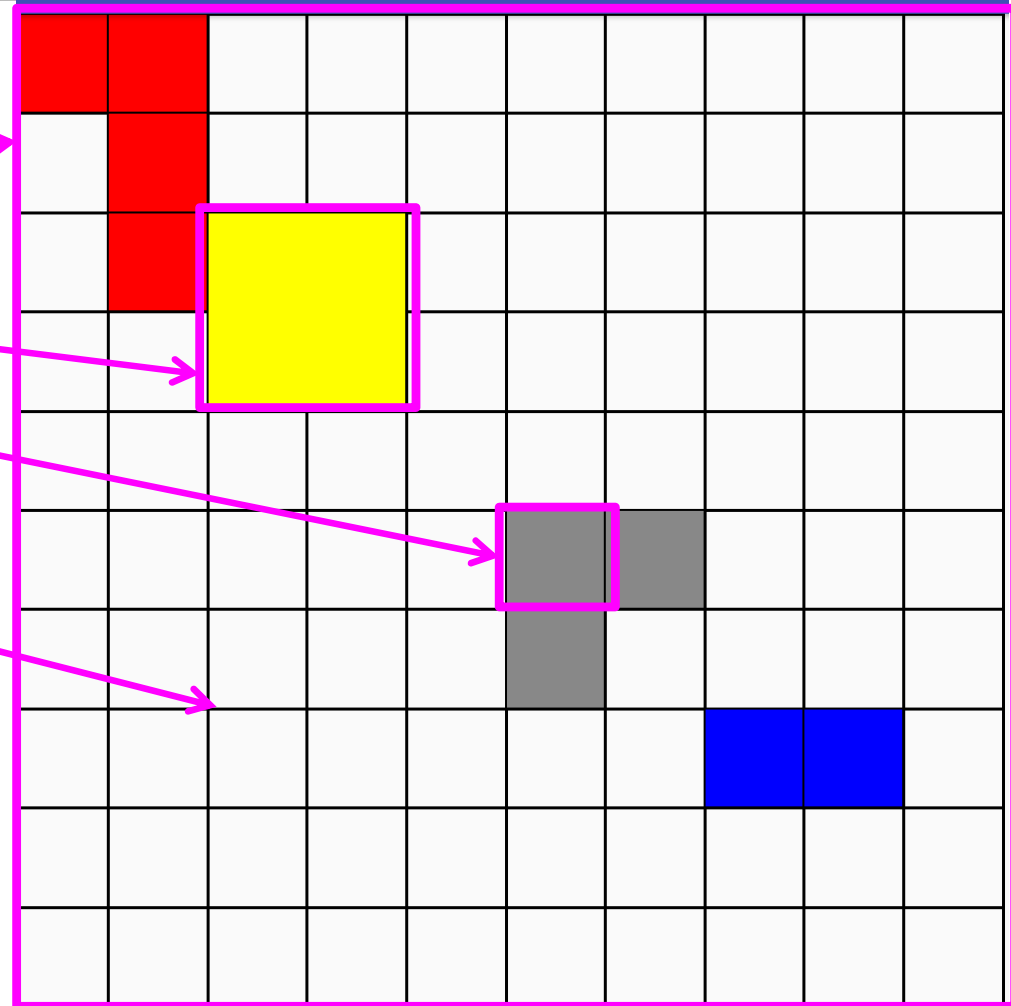
Objektov/tried:

- **Canvas**
- **Scena**
- **Tvar**
- **Stvorcek**
- **Mreza**

Každý reaguje na:

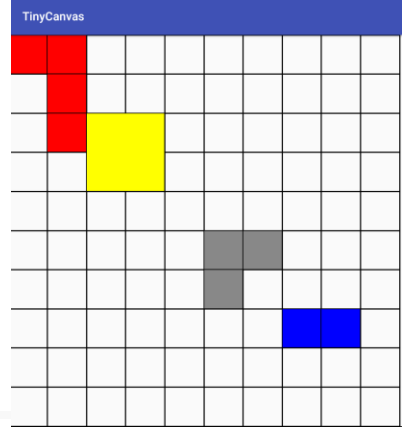
- **onTouch ( )**
- **onDraw ( )**

TinyCanvas



# Hierarchia

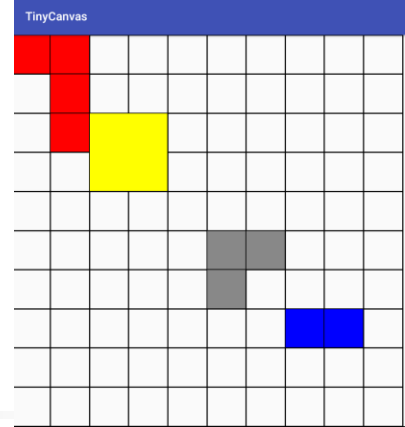
(Tvar - Shape)



```
class Tvar(val stvorceky:List<Stvorcek>) {  
    // Tvar je zoznam štvorčekov  
    fun onDraw() {  
        // vykresli Tvar - vykresli každý jeho štvorček  
        for (stvorcek in stvorceky) stvorcek.onDraw()  
    }  
    fun onTouched(motionEvent: MotionEvent): Boolean {  
        if (isIn(motionEvent)) { // bol tvar zasiahnutý eventom ?  
            var reDraw = false // oznám všetkým prekresli sa  
            for (stvorcek in stvorceky)  
                reDraw = reDraw or stvorcek.onTouched(motionEvent)  
            return reDraw // true, ak treba invalidate()  
        } else  
            return false  
    }  
    private fun isIn(motionEvent: MotionEvent) =  
        stvorceky.any { it.isIn(motionEvent) }  
    var isIn = false // ak niektorý zo štvorčekov bol  
    for (stvorcek in stvorceky) // zasiahnutý, tak Tvar bol tiež  
    isIn = isIn or stvorcek.isIn(motionEvent)  
    return isIn  
}
```

# Hierarchia

(Stvorecek)



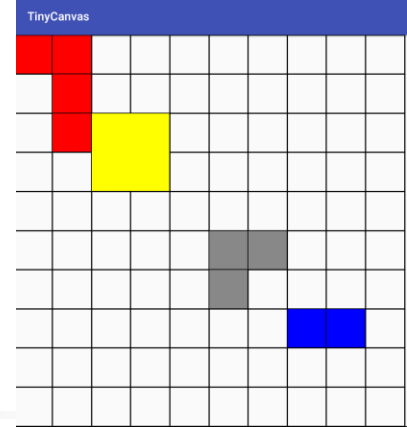
```
fun onDraw() {
    val r = Rect(x+1, y+1, x+sizeX-1, y+sizeY-1)
    CanvasView.c!!.drawRect(r, p) // chýbajú detaily, ako farba...
}

fun onTouched(event: MotionEvent): Boolean {
    int action = event.getAction()
    if (action == MotionEvent.ACTION_DOWN) {
        ... START MOVE ... } // začiatok dragovania štvorčeka
    else if (action == MotionEvent.ACTION_UP ||
        action == MotionEvent.ACTION_CANCEL) {
        ... END MOVE ... // koniec dragovania, urobí 'hop'
    } else if (action == MotionEvent.ACTION_MOVE) {
        ... MOVE ... } // počas dragovania, prekresľujeme
    }
}

fun isIn(event: MotionEvent): Boolean { // event je v obdĺžniku
    return x <= event.getX() && event.getX() <= x + sizeX
        &&
        y <= event.getY() && event.getY() <= y + sizeY
}
```

# Hierarchia

(top level Canvas)



```
override fun onTouch(view: View,  
                      motionEvent: MotionEvent): Boolean {  
    if (scena.onTouched(motionEvent))  
        invalidate()  
    return true  
}
```

```
override fun onDraw(canvas: Canvas) {  
    super.onDraw(canvas)  
    mreza.onDraw() // prekreslí mrežu  
    scena.onDraw() // prekreslí scénu,  
                  // scéna je List<Tvar>  
                  // a každý Tvar je List<Stvorcek>  
}
```

Objektov/tried:

- Canvas
- Scena
- Tvar
- Stvorcek
- Mreza

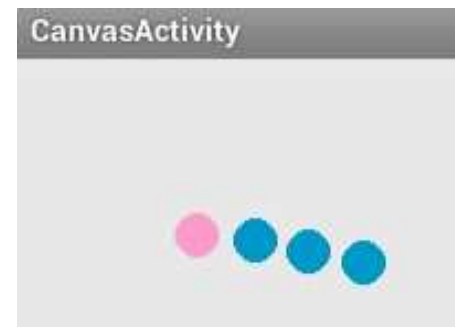
reagujú na:

- onTouch(event)
  - onDraw()
- príp.
- isIn(event)

# Vlákno (Thread) vo View

(dynamická hra v canvase, simulácia cez thread)

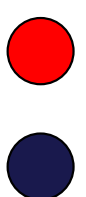
```
class CanvasView(context: Context, attrs: AttributeSet) :  
    View(context, attrs), View.OnTouchListener, View.OnKeyListener {  
    var touchX = 100f; var touchY = 100f           // interface  
    var ballX = 200f; var ballY = 200f  
  
    init {  
        setOnTouchListener(this) setOnKeyListener(this)  
        val th = object : Thread() { // SAM - single abstract method  
            override fun run() { // život vlákna  
                while (!stopped) {  
                    if (!paused) { // simulácia  
                        ballX += (touchX-ballX)/touches/50  
                        ballY += (touchY-ballY)/touches/50  
                        touchX = (ballX+50*touchX[i])/51  
                        touchY = (ballY+50*touchY[i])/51  
                        try { // pozdržanie  
                            Thread.sleep(100)  
                            postInvalidate() // prekreslenie v GUI vlákne  
                        } catch (e: InterruptedException) {  
                        }  
                    }  
                }  
            }  
        }  
        th.start() // spustenie vlákna  
    }  
}
```



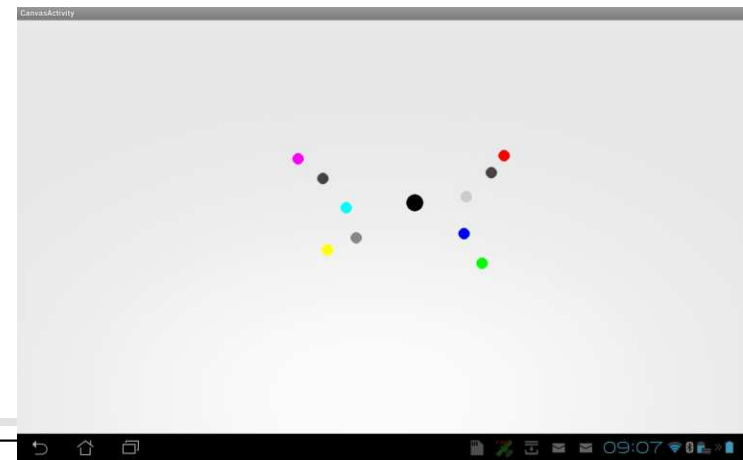
# onDraw, onTouch vo View

```
override fun onDraw(canvas: Canvas?) { // prekreslenie
    super.onDraw(canvas)
    if (canvas != null) {
        Paint p = Paint() // kreslenie guľičiek
        p.setColor(getResources().getColor(R.color.red))
        canvas.drawCircle(touchX, touchY, 10, p)
        p.setColor(getResources().getColor(R.color.blue))
        canvas.drawCircle(ballX, ballY, 10, p)
    } else
        Log.i("Canvas", "null")
}

override fun boolean onTouch(v:View, event:MotionEvent):Boolean {
    touchX = event.getX() // netestujeme typ eventu
    touchY = event.getY() // zoberieme len X,Y súradnice
    return true
}
```



# MultiTouch



```
override fun onTouch(v: View, event: MotionEvent): Boolean {  
    Log.i("Canvas", "counts:" + event.pointerCount)  
    val maskedAction = event.actionMasked
```

```
    if (maskedAction == MotionEvent.ACTION_DOWN ||  
        maskedAction == MotionEvent.ACTION_POINTER_DOWN) {
```

```
        touches = event.pointerCount  
        for (i in 0 until event.pointerCount) {  
            Log.i("Canvas", "X:" + event.getX(i))  
            Log.i("Canvas", "Y:" + event.getY(i))  
            touchX[i] = event.getX(i)  
            touchY[i] = event.getY(i)  
        }  
        return true
```

Žiadne dva  
prsty sa  
nedotknú  
naraz

```
class CanvasView(...) : View(...),  
    View.OnTouchListener,  
    View.OnKeyListener {  
  
    override fun onTouch(...) : Boolean  
    override fun onKey(...) : Boolean
```



# onKey vo View

```
class CanvasView(...) : View(...),
    View.OnTouchListener,
    View.OnKeyListener {

    override fun onTouch(...) : Boolean
    override fun onKey(...) : Boolean
}
```

```
override fun onKey(arg0: View, arg1: Int, arg2: KeyEvent) :
    Boolean {
    val rnd = Random()
    when (arg1) {
        KeyEvent.KEYCODE_DPAD_LEFT -> ballX -= rnd.nextInt(50)
        KeyEvent.KEYCODE_DPAD_RIGHT -> ballX += rnd.nextInt(50)
        KeyEvent.KEYCODE_DPAD_UP -> ballY -= rnd.nextInt(50)
        KeyEvent.KEYCODE_DPAD_DOWN -> ballY += rnd.nextInt(50)
        KeyEvent.KEYCODE_SPACE -> {
            ballX += rnd.nextInt(100) - 50
            ballY += rnd.nextInt(100) - 50
        }
        else -> return false
    }
    invalidate()
    return true // event handled
}
```



# Option Menu

(onCreateOptionsMenu)

```
<menu
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/pause" android:icon="@drawable/pause"
        android:title="Pause">

    </item>
    <item android:id="@+id/play" android:icon="@drawable/play"
        android:title="Play">

    </item>
    <item android:id="@+id/stop" android:icon="@drawable/stop"
        android:title="Stop">

    </item>
</menu>
```

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {
    val inflater = menuInflater
    inflater.inflate(R.menu.activity_canvas, menu)
    (menu as? MenuBuilder)?.setOptionalIconsVisible(true)
    return super.onCreateOptionsMenu(menu)
```



Pause



Play



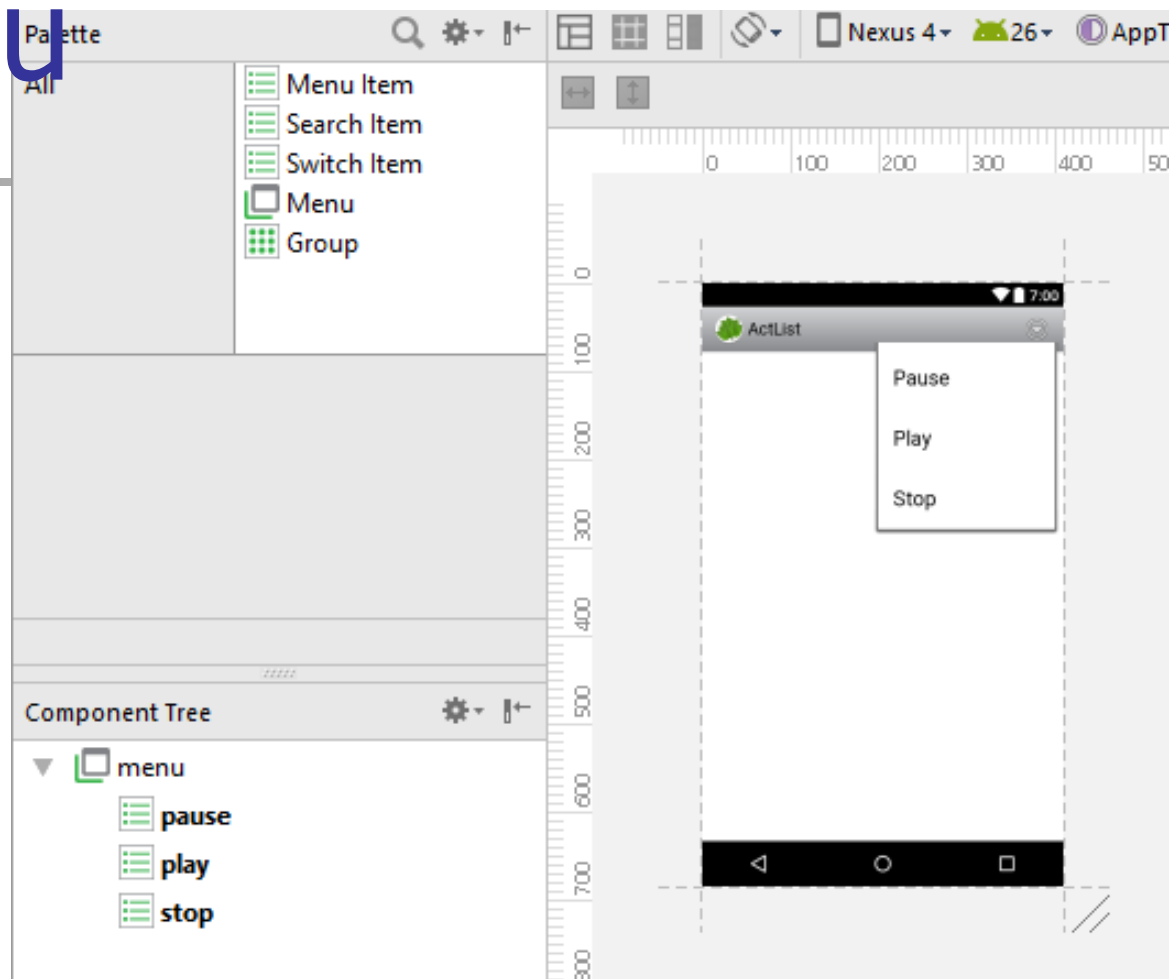
Stop

# Option Menu

(onOptionsItemSelected)

Rovnako dobre to môžete menu navrhovať v design view editore, ako priamo v .xml

Spôsob zobrazenia a renderovania závisí na API level zariadenia



```

<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/pause" android:icon="@drawable/pause" android:title="Pause"> </item>
    <item android:id="@+id/play" android:icon="@drawable/play" android:title="Play"> </item>
    <item android:id="@+id/stop" android:icon="@drawable/stop" android:title="Stop"> </item>
</menu>

```

# Option Menu

```
Thread th = new Thread() {  
    fun run() {  
        while (!stopped) {  
            if (!paused) {  
                ...  
            }  
        }  
    }  
}
```

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    binding.apply {  
        when (item.getItemId()) {  
            R.id.pause -> {  
                canvasView1.paused = true  
                return true  
            }  
            R.id.play -> {  
                canvasView1.paused = false  
                return true  
            }  
            R.id.stop -> {  
                canvasView1.stopped = true  
                return true  
            }  
            else -> return super.onOptionsItemSelected(item)  
        }  
    }  
}
```

# Context Menu

```
override fun onCreate(  
    savedInstanceState: Bundle?) { ...  
    registerForContextMenu(listView1) // rozdiel od OptionMenu  
} ContextMenu (oproti OptionMenu) treba registrovať k príslušnému view
```

```
override fun onCreateContextMenu(menu: ContextMenu?, v: View?,  
    menuInfo: ContextMenu.ContextMenuInfo? ) {  
    getMenuInflater().inflate(R.menu.list_menu, menu)  
} // v je View, na ktoré bolo spáchané ContextMenu Action
```

```
override fun onContextItemSelected(item: MenuItem): Boolean {  
    val info = item.getMenuInfo() as AdapterContextMenuInfo  
    val className = actList.get(info.id.toInt())  
    when (item.getItemId()) {  
        R.id.remove -> {  
            actList.removeAt(info.id.toInt())  
            la.notifyDataSetChanged()  
            return true  
        }  
    }  
}
```

