Haladó felhasználói felületi lehetőségek, listák kezelése

Ekler Péter

BME VIK AUT, AutSoft

peter.ekler@aut.bme.hu

Tematika

1. Android platform bemutatása, Kotlin alapok
2. Alkalmazás komponensek, Kotlin konvenciók
3. Felhasználói felület
4. Fragmentek, haladó UI
5. Listák kezelése hatékonyan
6. Perzisztens adattárolás, adatbázisok, haladó Kotlin
7. Hálózati kommunikáció
8. Felhő szolgáltatások
9. Helymeghatározás, térkép kezelés
10. Architektúra komponensek, JetPack

Tartalom

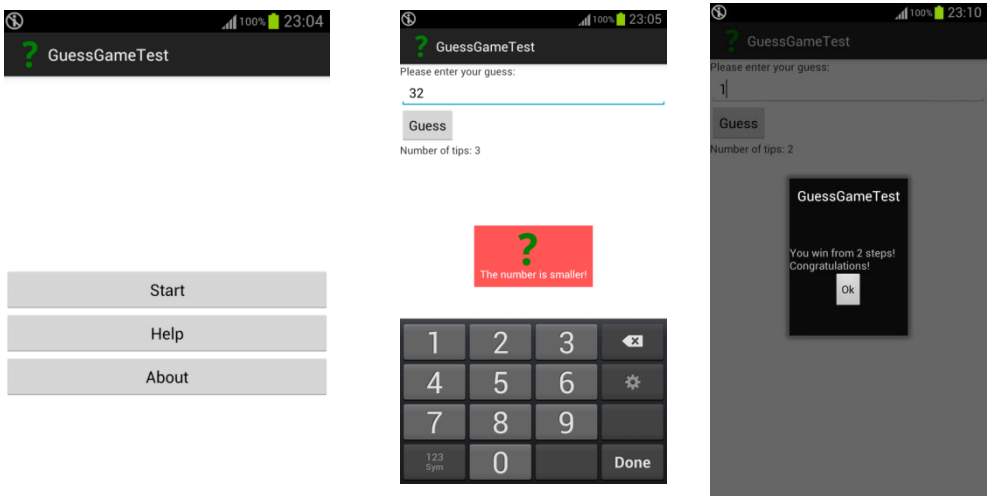
- Dialógusok
- Stílusok és témák használata
- Animációk
- Design Guideline
- Egyedi (custom) felületi elemek tervezése és megvalósítása
- Hatékony lista kezelés: RecyclerView
- CoordinatorLayout

Gyakorló alkalmazás

Barkóba

Gyakoroljunk!

- Késztsünk egy barkóba alkalmazást
- Képernyő forgatás kezelése
- *TextInputLayout* használata
- Eredmény dialógus Activity:
  android:theme="@style/Theme.AppCompat.Dialog"



Új Activity indítása

- SecondActivity indítása:

```
fun runSecondActivity() {
    val myIntent: Intent = Intent()
    myIntent.setClass(this@MainActivity,
                      SecondActivity::class.java)
    // Adat átadása
    myIntent.putExtra("KEY_DATA", "Hi there!")
    startActivity(myIntent)
}
```

Activity vezérlés, navigáció

- Manifest property-k
  - taskAffinity: task leírás
  - launchMode: indulási mód
  - allowTaskReparenting: új taskot hozzon-e létre
  - clearTaskOnLaunch: többi Activity-t törli a taskról
  - alwaysRetainTaskState: a rendszer kezelje a task állapotát
  - finishOnTaskLaunch: állítsa le az Activity ha a felhasználó kilépett
  - screenOrientation: fix landscape/portrait
- *startActivity(...)* intent Tag-ek
  - FLAG_ACTIVITY_NEW_TASK
  - FLAG_ACTIVITY_CLEAR_TOP
  - FLAG_ACTIVITY_SINGLE_TOP

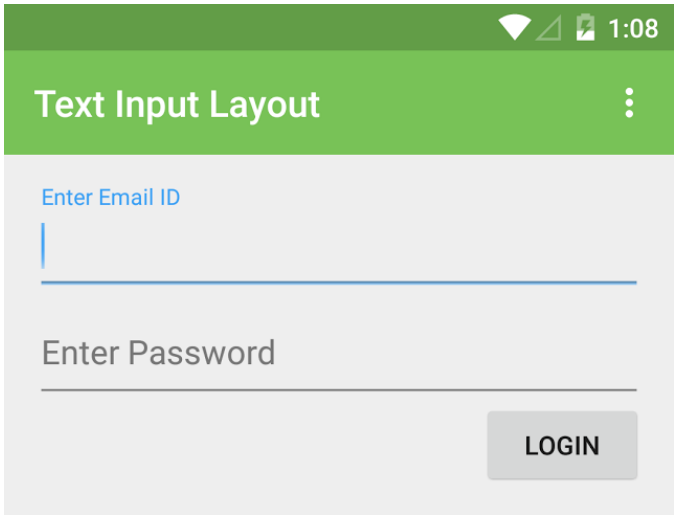
BackStack kezelése

```
val showMain = Intent(this, MainActivity::class.java)
showMain.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
startActivity(showMain)
finish()
```

TextInputLayout

- Support library: https://developer.android.com/topic/libraries/support-library/packages.html
- Design Support Library része, mint a SnackBar
- Gradle dependency:
  - implementation 'com.android.support:design:27.1.1'
- Használat:

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Enter Email ID"
      android:id="@+id/etName" />

</android.support.design.widget.TextInputLayout>
```



Clean Code

Forrás: https://cleancoders.com/

Mit jelent a Clean Code? Miért van rá szükség?

- Mi a software igazi értéke?
  - > Karbantarthatóság
  - > Folyamatos szállítás biztosítása

- Napjainkban:
  - > Folyamatosan változó követelmények
- Agilis fejlesztés
- Csapatmunka
- Kódminőség
- Software életciklus

Alapvető Clean Code elvek

- Elnevezés
  - Kis scopeon belül: hosszú, beszédes nevek
  - Nagy scopeban: rövid nevek
- Rövid osztályok
- Egy metódus csak egy dolgot csinál
- Rövid metódusok
  - Maximum ~4 sor!
- Kevés argumentum
  - Maximum 3
- Nincs boolean argumentum
- Nincs output argumentum

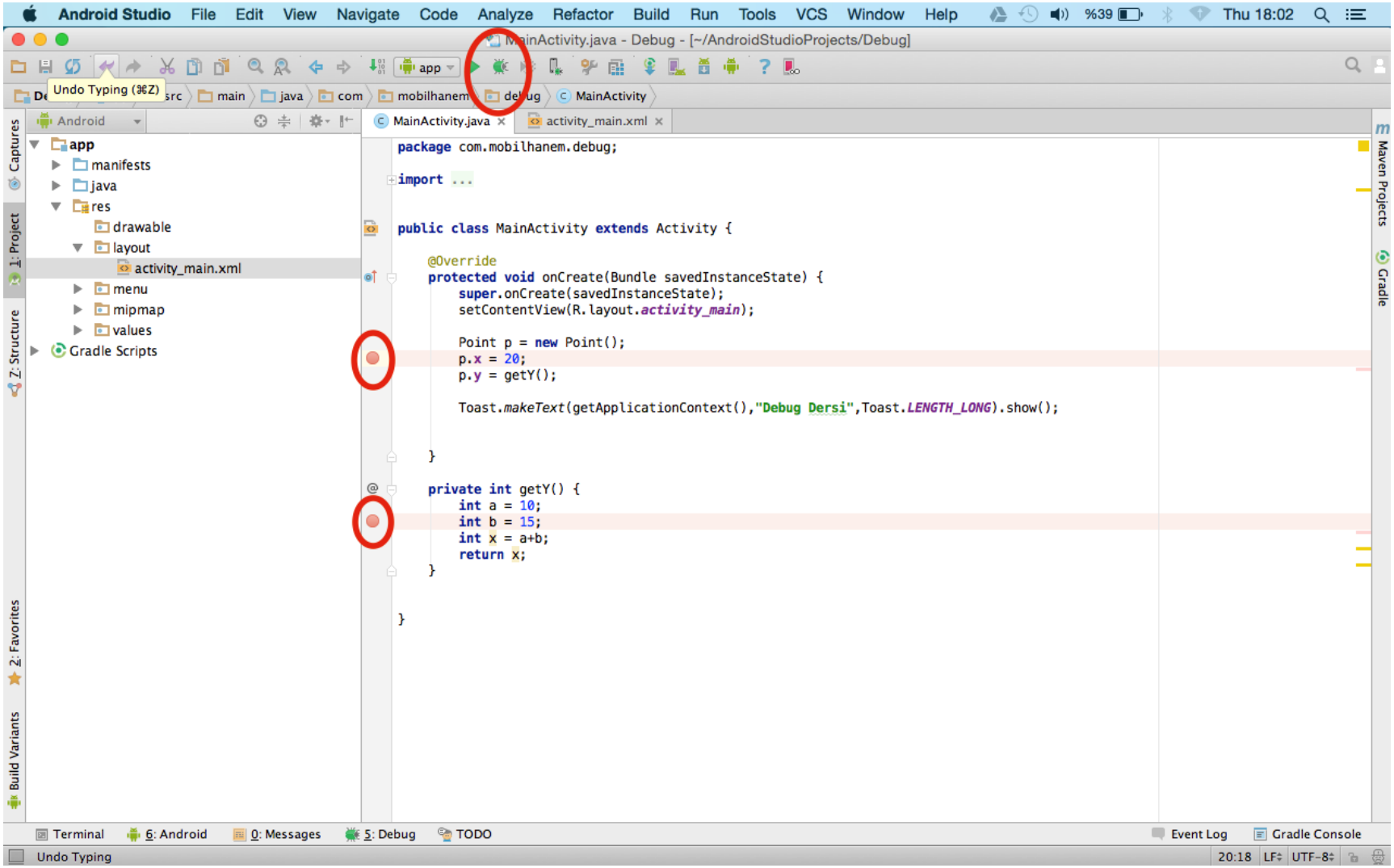
Android LogCat

- Log nézet
- Monitorozható
- Többféle log szint:
  - v(String, String) (verbose)
  - d(String, String) (debug)
  - i(String, String) (information)
  - w(String, String) (warning)
  - e(String, String) (error)
- Log.i("TAG_LOCATION", "Position: ${position}")
- File-ba is átirányítható
  - logcat -f <filename>
  - http://developer.android.com/tools/help/logcat.html

Debug lehetőségek

- SDK támogat többféle debug lehetőséget
  - Emulátor
  - On-device debug
- Debug folyamat:
  - Breakpointok elhelyzése
  - Alkalmazás indítása debug módban
  - Soronkénti végrehajtás

Debug indítása



Dialógus Fragment

DialogFragment I.

- Egy Fragment dialógusként is megjelenhet
  - Dialógus egyedi layout-tal
  - Az AlertDialog.Builder továbbra is használható
- Így egy dialógus is ugyanolyan életciklussal rendelkezik, mint egy Fragment
- A FragmentDialog-ok is rákerülhetnek a BackStack-re

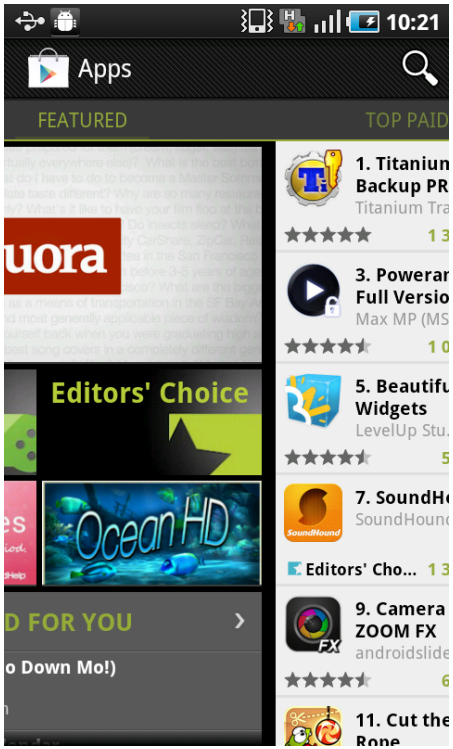
DialogFragment II.

- .onCreateDialog()
  - Ez a metódus is visszatérhet a megjelenítendő Dialog-gal
- .onCreateView()
  - Ha nem használjuk az .onCreateDialog()-ot
  - Tetszőleges megjeleníthető tartalom
- Egy DialogFragment egyben Fragment is!
  - Ha kell, akár Activity-be ágyazottan is megjeleníthető

Lapozható felületek

ViewPager

- ViewGroup, ahol az elemek közt swipe-al lehet mozogni
  - Pl.: Google Play



FragmentPagerAdapter

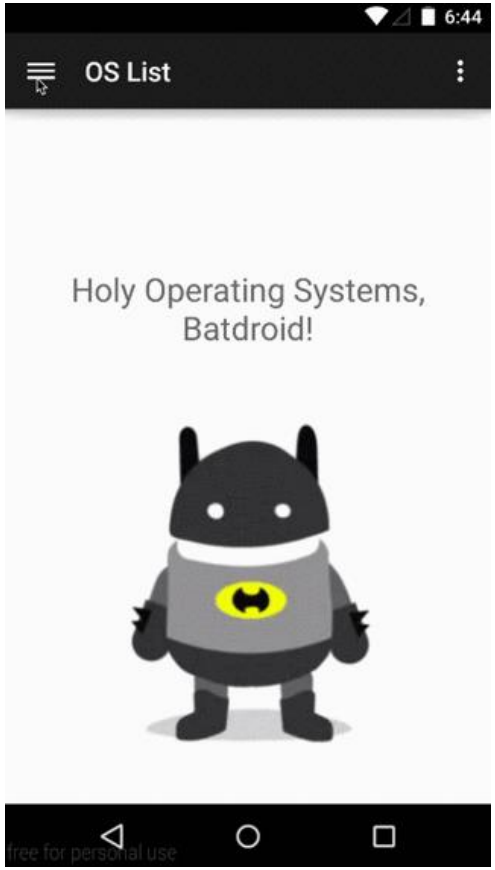
- Általában Fragment-eket adják a ViewPager oldalait
- Ekkor egy FragmentPagerAdapter példány szolgáltatja az egyes oldalakat
  - Hasonló elven működik, mint a BaseAdapter
  - getItem(position: Int): Fragment
    - Visszaadjuk a megfelelő Fragment példányt
  - getCount(): Int
    - Visszaadjuk, hogy összesen hány oldalunk van

Komplex menü típusok

Navigation Drawer

- Lényegében ugyanaz a menu XML
- DrawerLayout-al kell körül venni a belső tartalmat
- NavigationView tartalmazza a fejléc nézetet és a menü elemeket tartalmazó listát

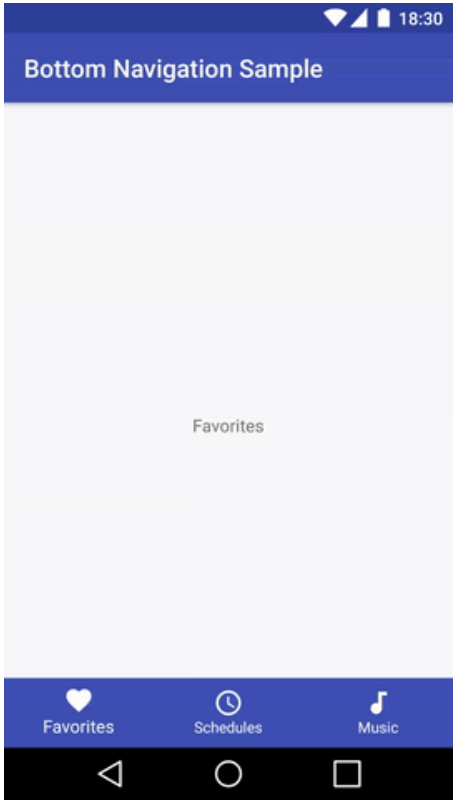
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_main"
        android:layout_width="match_parent "
        android:layout_height="match_parent" />

    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />

</android.support.v4.widget.DrawerLayout>
```



Bottom Navigation View

- Ugyanaz a menu XML
- Elhelyezhető a layouton:

```
<android.support.design.widget.BottomNavigationView
    android:id="@+id/bottom_navigation"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    app:itemBackground="@color/colorPrimary"
    app:itemIconTint="@drawable/nav_item_color_state"
    app:itemTextColor="@drawable/nav_item_color_state"
    app:menu="@menu/bottom_navigation_main" />
```



STÍLUSOK ÉS TÉMÁK

Stílusok létrehozása

- Style file: res/values/styles.xml

```
<style name="ExampleStyle">
    <item name="android:textSize">22sp</item>
    <item name="android:textColor">#0000EE</item>
</style>
```

- Alkalmazás:

```
<TextView
    android:id="@+id/tvHello"
    android:text="@string/hello_world"
    style="@style/ExampleStyle" />
```

Témák használata

- Téma definíció:

```
<style name="CustomTheme" parent="android:Theme">
  <item name="android:windowTitleSize">50dip</item>
  <item name="android:textColor">#000000</item>
  <item name=
    "android:windowBackground">@color/white_color</item>
</style>
```

- Témák öröklődhetnek egymásból
- Témák alkalmazása Manifest-ben:

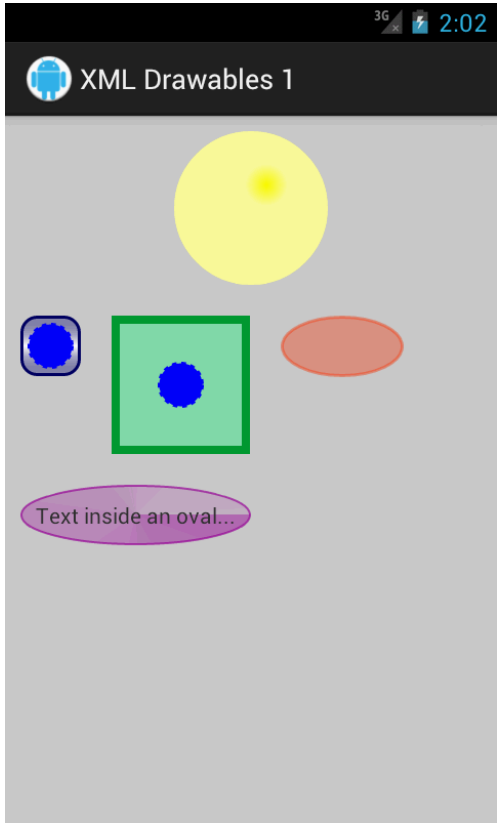
```
<activity android:label="" android:name=".MainActivity"
    android:screenOrientation="portrait"
    android:theme="@style/CustomTheme"/>
```

GRAFIKAI ERŐFORRÁSOK

Grafikai erőforrások 1/2

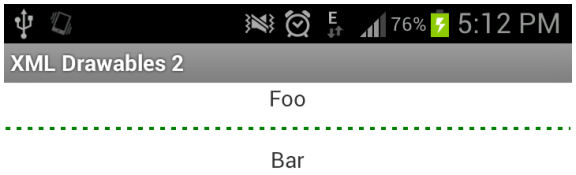
- XML definíció

```
<?xml version="1.0" encoding="utf-8"?>
<shape
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:shape="oval">

  <gradient
    android:type="radial"
    android:gradientRadius="20"
    android:centerX=".6"
    android:centerY=".35"
    android:startColor="#FFFF00"
    android:endColor="#FFFF99" />

  <size
    android:width="100dp"
    android:height="100dp"/>
</shape>
```



Grafikai erőforrások 2/2

- Green dashed line:




```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android=
  "http://schemas.android.com/apk/res/android"
    android:shape="line">
    <stroke
        android:width="2dp"
        android:color="#008000"
        android:dashWidth="3dp"
        android:dashGap="4dp"/>

    <size android:height="20dp" />
</shape>
```

Állapot leírás XML-ben

- Gomb szövegszín állapottól függően:
  - res/color/button_text.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
        android:color="#ffff0000"/> <!-- pressed -->
    <item android:state_focused="true"
        android:color="#ff0000ff"/> <!-- focused -->
    <item android:color="#ff000000"/> <!-- default -->
</selector>
```

- Alkalmazás:

```
<!-- use  android:layerType="software" only if needed!>
<Button
    android:layout_width=„match_parent"
    android:layout_height="wrap_content"
    android:text="@string/button_text„
    android:layerType="software"
    android:textColor="@color/button_text" />
```

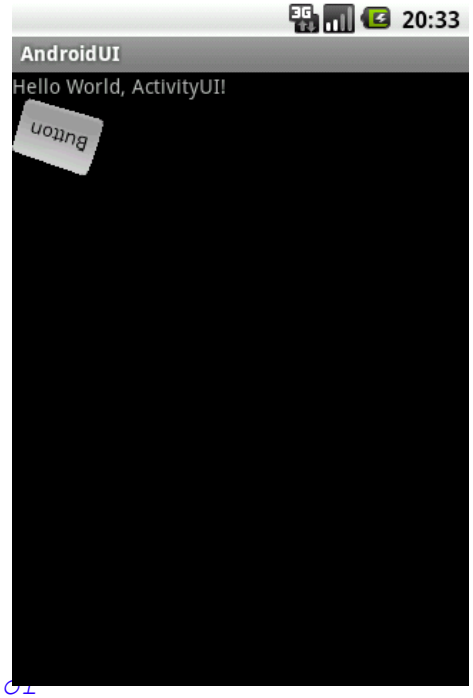
ANIMÁCIÓK

Animációk

- Animációk támogatása
  - XML erőforrás (*res/anim*)
  - Programkód
- Layout animáció
  - *Scale*
  - *Rotate*
  - *Translate*
  - *Alpha*
- Három fő típus:
  - Tween animáció
  - Frame animáció
  - Property animator

Tween animáció erőforrás

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <scale
        android:interpolator=
          "@android:anim/accelerate_interpolator"
        android:fromXScale="0.0"
        android:toXScale="1.0"
        android:fromYScale="0.0"
        android:toYScale="1.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="1000" />

    <alpha android:fromAlpha="0.0"
        android:toAlpha="1.0"
        android:duration="5000"/>

    <rotate
        android:interpolator="@android:anim/accelerate_interpolator"
        android:fromDegrees="0.0„
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="5000" />
</set>
```



Tween animáció lejátszása

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

    val showAnim = AnimationUtils.loadAnimation(this,
      R.anim.btnanim)
    btnAnim.startAnimation(showAnim)
}
```

Frame animáció

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="
    http://schemas.android.com/apk/res/android"
     id="selected" android:oneshot="false">
    <item android:drawable=
      "@drawable/monster1" android:duration="200" />
    <item android:drawable=
      "@drawable/monster2" android:duration="200" />
    <item android:drawable=
      "@drawable/monster3" android:duration="200" />
    <item android:drawable=
      "@drawable/monster4" android:duration="200" />
    <item android:drawable=
      "@drawable/monster5" android:duration="200" />
</animation-list>
```

Frame animáció lejátszása

```
ivMonster.setBackgroundResource(
  R.anim.monstermove)
val frameAnimation: AnimationDrawable =
  ivMonster.getBackground()
frameAnimation.start()
```

OBJECTANIMATOR

Mostoha Roland (roland.mostoha@autsoft.hu)

Android animációk

- Objektumok, View-k attribútumait módosíthatjuk az idő függvényében
- Animator ősosztály
- ValueAnimator – Int, Float, Object értékek animációja egy adott időintervallum alatt

```
val animation = ValueAnimator.ofFloat(0f, 100f)
animation.duration = 1000
animation.start()
```

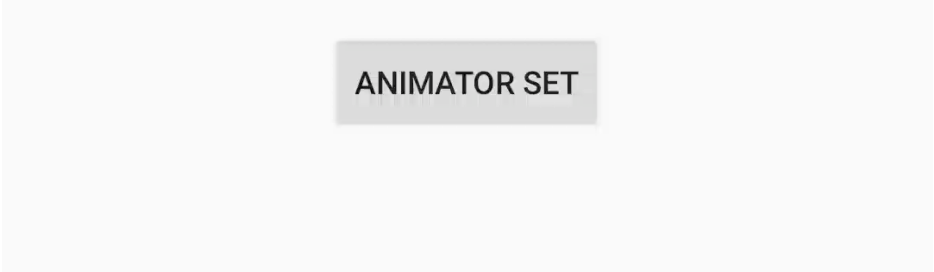
ObjectAnimator

- ValueAnimator leszármazottja, alap animáció típusok használhatóak
  - rotation
  - translation
  - scale
  - alpha
  - backgroundColor

```
ObjectAnimator animation = ObjectAnimator.ofFloat(view, "translationX", 100f);
animation.setDuration(1000);
animation.start();
```

ObjectAnimator

- AnimatorSet használatával összefűzhetünk animációkat



ObjectAnimator példa

```
<Button
    android:id="@+id/rotateButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="startAnimation"
    android:text="Rotate" />
```

```
fun startAnimation(view: View) {
    when (view.id) {
        R.id.rotateButton -> {
            val animation = ObjectAnimator.ofFloat(view, "rotation", 360f)
            val listener = object : Animator.AnimatorListener {
                override fun onAnimationRepeat(animator: Animator) {
                }

                override fun onAnimationEnd(animator: Animator) {
                    view.rotation = 0f
                }

                override fun onAnimationCancel(animator: Animator) {
                }

                override fun onAnimationStart(animator: Animator) {
                }
            }
            animation.addListener(listener)
            animation.apply {
                duration = 500
                start()
            }
        }
    }
}
```

Kotlin kiegészítések

- Apply használata animációk attribútumainak beállításához

```
colorAnimation.apply {
    duration = 500
    interpolator = AccelerateDecelerateInterpolator()
    start()
}
```

Kotlin kiegészítések

- AnimationListener hozzáadása lambda paraméterekkel

Előtte:

```
addListener(object: Animator.AnimatorListener {
    override fun onAnimationRepeat(p0: Animator?) {
    }
    override fun onAnimationEnd(p0: Animator?) {
        view.rotation = 0f
    }
    override fun onAnimationCancel(p0: Animator?) {
    }
    override fun onAnimationStart(p0: Animator?) {
    }
})
```

Utána:

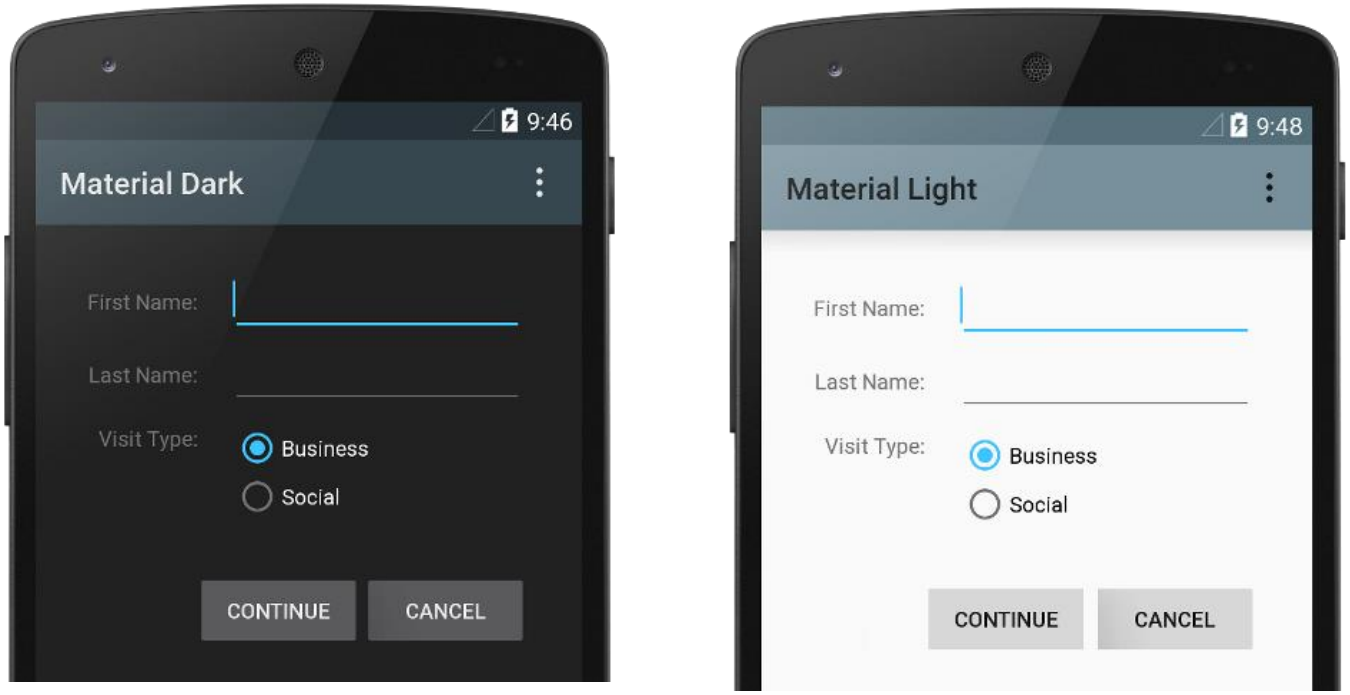
```
doOnEnd {
    view.rotation = 0f
}
```

ObjectAnimator - erőforrás

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- Before rotating, immediately set the alpha to 0. -->
    <objectAnimator
        android:valueFrom="1.0"
        android:valueTo="0.0"
        android:propertyName="alpha"
        android:duration="0" />

    <!-- Rotate. -->
    <objectAnimator
        android:valueFrom="-180"
        android:valueTo="0"
        android:propertyName="rotationY"
        android:interpolator="@android:interpolator/accelerate_decelerate"
        android:duration="@integer/card_flip_time_full" />

    <!-- Half-way through the rotation (see startOffset), set the alpha to 1. -->
    <objectAnimator
        android:valueFrom="0.0"
        android:valueTo="1.0"
        android:propertyName="alpha"
        android:startOffset="@integer/card_flip_time_half"
        android:duration="1" />
</set>
```

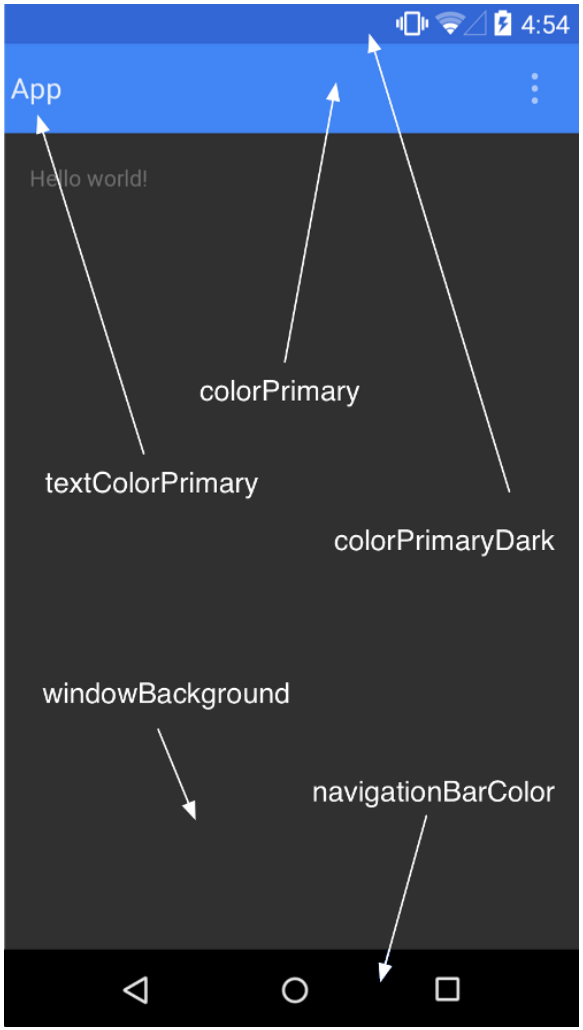
MATERIAL DESIGN

Material Design

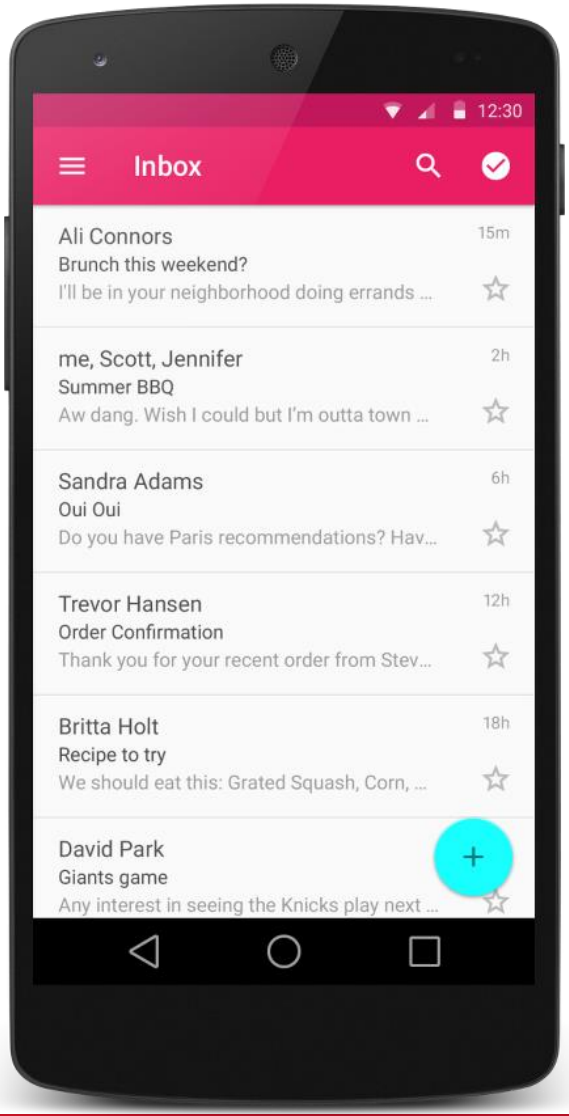


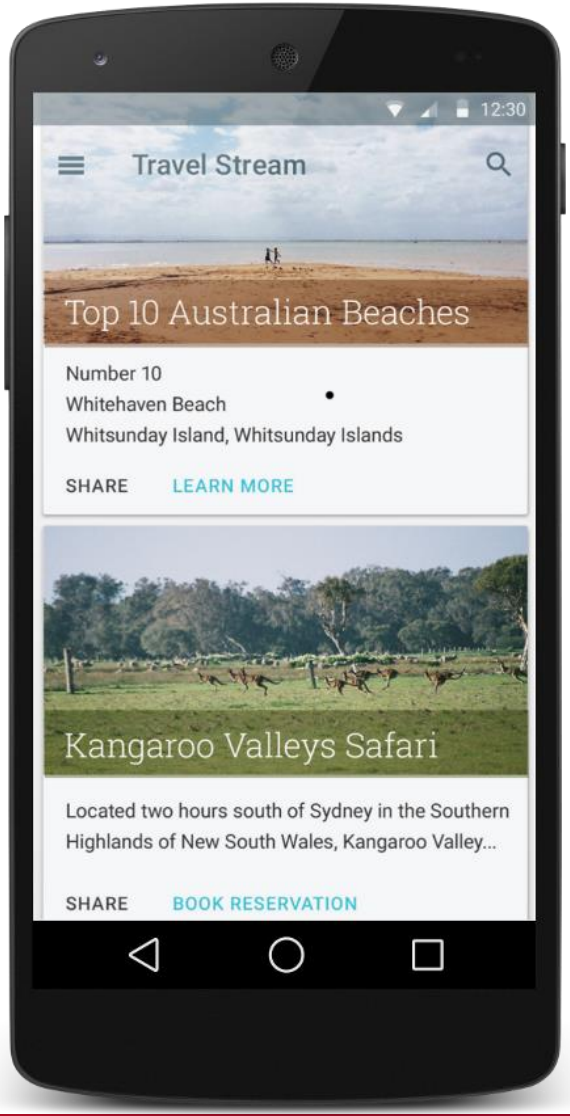
https://www.google.com/design/spec/material-design/introduction.html

Material téma



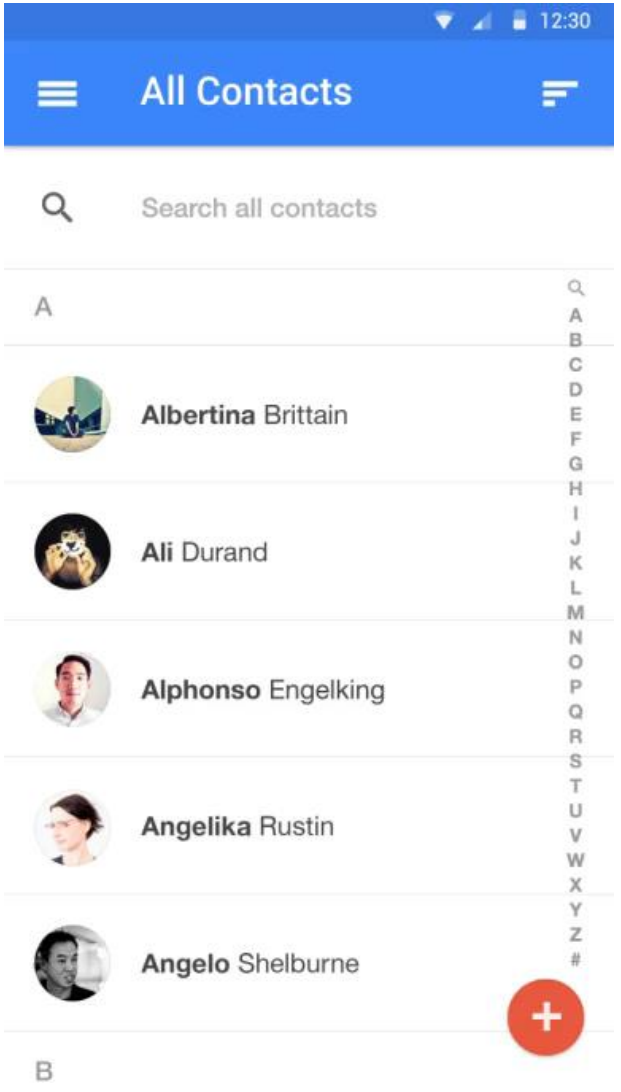
Listák (RecyclerView) és kártyák (CardView)





További Material elemek

- Árnyékok (Z property)
- Animációk
  - touch feedback
  - circular reveal
  - activity transition
  - curved motion
  - view state change
  - state list drawables
- Grafikák
  - Vector drawables
  - Drawable tinting
  - Color extraction



Színek

- Palette API
  - Optimális betű, és egyéb színek lekérdezése kép alapján
- Például: adott egy háttérkép, kérdés, hogy milyen színnel írjunk rá

```
val palette = Palette.Builder(myBitmap).generate()
val vibrant = palette.getVibrantColor(0x00000000)
```

- További infok:
  - http://willowtreeapps.com/blog/palette-the-new-api-for-android/
  - https://www.bignerdranch.com/blog/extracting-colors-to-a-palette-with-android-lollipop/

EGYEDI NÉZETEK

Egyedi nézetek

- View leszármazott
- Beépített nézetek és *LayoutGroup*-ok is felüldefiniálhatók, pl. saját nézet *RelativeLayout*-ból leszármaztatva
- *<merge>* XML elem
- XML-ek egymásba ágyazhatósága: *<include>*

Egyedi felületi nézet

- Teljesen egyedi felületi elemek definiálása
- Meglévő felületi elemek kiegészítése
- Érintés események kezelése
- Dinamikus rajzolás
  - Színek, rajzolási stílus
  - Gyakori alakzatok: vonal, négyzet, kör stb.
  - Szöveg rajzolása
  - Képek megjelenítése
- Megjelenítési mérethez való igazodás
- XML-ből is használható!

CustomView feladatai

- Törekedjünk az Android szabványok betartására
- Támogassuk a szükséges attribútumok XML-ben való megadását is
- Eseménykezelés, paraméterekhez való hozzáférés biztosítása
- Képernyőméret és eszköz függetlenség biztosítása

CustomView példa

- Forráskód:

```
class MyView(context: Context, attrs: AttributeSet?)
        : View(context, attrs)
```

- XML:

```
<hu.bme.aut.amorg.examples.customview.MyView
    android:id="@+id/myView"
    android:layout_width="match_parent"
    android:layout_height="100dp"/>
```

XML tulajdonságok megadása

- values/attrs.xml:

```
<resources>
    <declare-styleable name="MyButton">
        <attr name= "counter_color" format="color" />
    </declare-styleable>
</resources>
```

Paraméter beállítása:

```
<hu.bme.aut.amorg.examples.customview.MyButton
    ...
    app:counter_color="#ff00ff00" />
```

XML tulajdonságok lekérdezése

```
class MyButton(context: Context, attrs: AttributeSet?) : Button(context, attrs) {
    private var counter = 0
    private var paintText = Paint()

    init {
        paintText.color = Color.RED
        paintText.textSize = 40f

        val typedArray = context.theme.obtainStyledAttributes(attrs, R.styleable.MyButton, 0, 0)
        try {
            paintText.color = typedArray.getColor(R.styleable.MyButton_counter_color, Color.RED)
        } finally {
            typedArray.recycle()
        }
    }

    override fun onDraw(canvas: Canvas) {
        super.onDraw(canvas)
        canvas.drawText("" + counter, 15f, 45f, paintText);
    }

    override fun onTouchEvent(event: MotionEvent): Boolean {
        if (event.action == MotionEvent.ACTION_DOWN) {
            counter++
            invalidate()
        }
        return super.onTouchEvent(event)
    }
}
```

Kép megjelenítése

```
img: Bitmap =
  BitmapFactory.decodeResource(
    getResources(), R.drawable.icon);
//...
override fun onDraw(canvas: Canvas?)  {
  canvas.drawColor(Color.BLACK)
  canvas.drawBitmap(img , 10, 10, null);
}
```

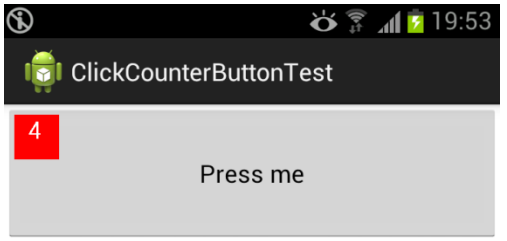
Méret beállítása

- XML-ben megadható a méret
- Dinamikusan is módosítható
- Például négyzetes megjelenítés szélességhez igazítva:

```
override fun onMeasure(widthMeasureSpec: Int, heightMeasureSpec: Int)
{
    val w = MeasureSpec.getSize(widthMeasureSpec)
    val h = MeasureSpec.getSize(heightMeasureSpec)
    val d = when {
        (w == 0) -> h
        (h == 0) -> w
        (w < h) -> w
        else -> h
    }
    setMeasuredDimension(d, d)
}
```

Gyakoroljunk!

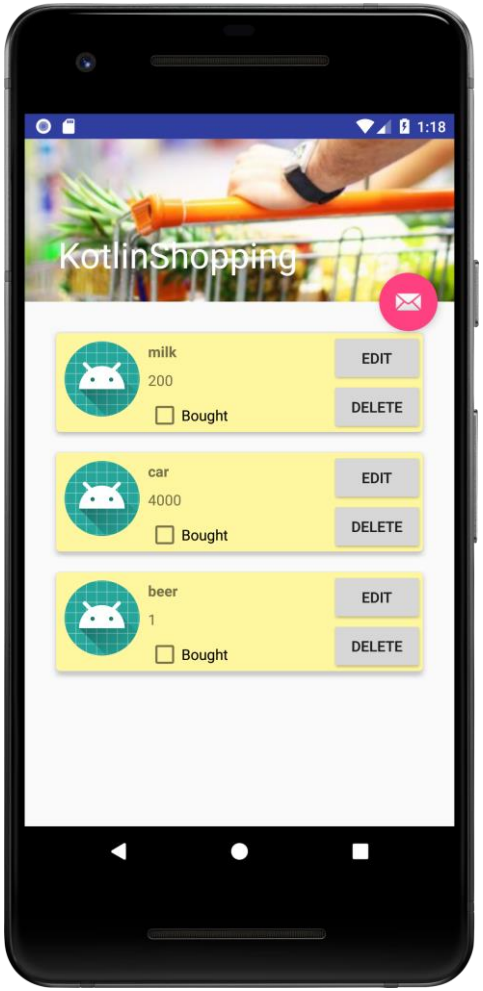
- Készítsünk egy alkalmazást, amely egy saját gomb vezérlőt jelenít meg!
- Valósítsuk meg, hogy a gombra kattintáskor egy számláló a gomb bal felső sarkában mutassa hányszor kattintottunk a gombra!
- A számláló mérete és láthatósága legyen XML paraméterből állítható!



LISTÁK KEZELÉSE

RecyclerView

- Listák hatékony kezelése
- Gyors scrollozás
- Általános érintés gesztusok támogatása (swipe, move, stb.)
- *ViewHolde*r minta a gyors működés érdekében
- Hatékony elem újrafelhasználás
- *Flexibilis*



RecyclerView.Adapter<ViewHolder> 1/3

- Inicializálás, konstruktor

```
private val context: Context
private val items: MutableList<ShoppingItem> = mutableListOf<ShoppingItem>(
        ShoppingItem("milk", 200, false),
        ShoppingItem("car", 4000, false),
        ShoppingItem("beer", 1, false)
)

constructor(context: Context) : super() {
    this.context = context
}
```

- Egy sor nézetének beállítása: onCreateViewHolder

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
    val view = LayoutInflater.from(parent.context).inflate(
            R.layout.row_item, parent, false
    )
    return ViewHolder(view)
}
```

ViewHolder implementáció

```
class ViewHolder(itemView: View?) : RecyclerView.ViewHolder(itemView) {
    val tvName = itemView.tvName
    val tvPrice = itemView.tvPrice
    val cbBought = itemView.cbBought
    val btnEdit = itemView.btnEdit
}
```

RecyclerView.Adapter<ViewHolder> 2/3

- Sorban levő elemek értékeinek beállítása
- Eseménykezelők beállítása
- ViewHolder binding

```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    val (name, price, bought) = items[holder.adapterPosition]
    holder.tvName.text = name
    holder.tvPrice.text = price.toString()
    holder.cbBought.isChecked = bought

    holder.btnEdit.setOnClickListener{
        (context as MainActivity).showEditTodoDialog(items[holder.adapterPosition])
    }
}
```

RecyclerView.Adapter<ViewHolder> 3/3

- Elemek száma, hozzáadás, törlés

```
override fun getItemCount() = items.size

fun addItem(item: ShoppingItem) {
    items += item
    notifyItemInserted(items.lastIndex)
}

private fun deleteItemBasedOnPosition(position: Int) {
    items.removeAt(position)
    notifyItemRemoved(position)
}
```

Bevásárlólista - terv

1. Data class
2. Egy sor layout-ja
3. RecyclerView – lista hol legyen
4. Adapter – megmonda hogy mi legyen a RecyclerView-ba

Összefoglalás

- Barkóba alkalmazás
- Dialógusok
- Stílusok és témák használata
- Animációk
- Egyedi (custom) felületi elemek tervezése és megvalósítása
- Hatékony lista kezelés: RecyclerView

Köszönöm a figyelmet!

peter.ekler@aut.bme.hu

AutSoft