Android

Felhasználói felület, Fragmentek

Dr. Ekler Péter
peter.ekler@aut.bme.hu

BME AUT
Department of
Automation and
Applied Informatics

Mobil alkalmazásfejlesztői verseny

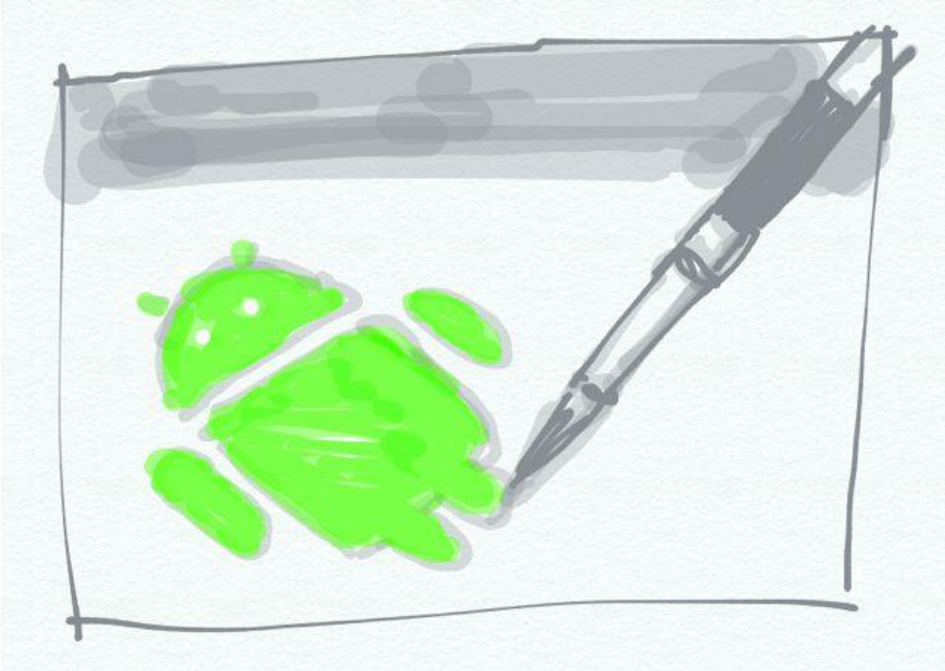
- Ütemezés:
  - Ötletek leadása: Október 8.
  - Short list kihirdetése: Október 29.
  - Prototípus béta leadása: November 23.
  - Prototípus végleges leadása: November 30.
  - Eredmény kihirdetése: December 17.
- Tetszőleges platformon megvalósítható prototíp
- **500.000 Ft összdíj!**
- Regisztráció és információk:
  - https://www.aut.bme.hu/Events/Granit2018

Mobil alkalmazásfejlesztői verseny

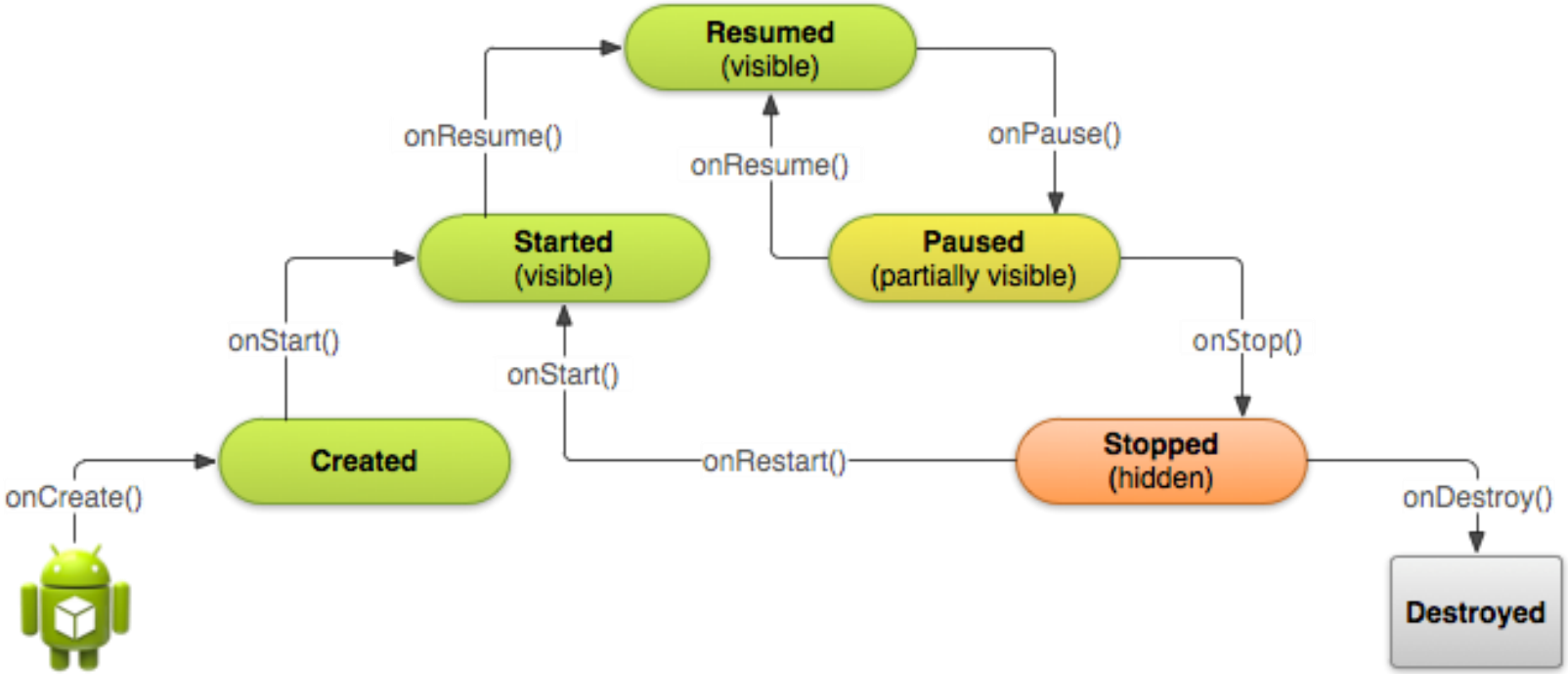
- Tájékoztató – új banki szolgáltatások (PSD2)
- Szeptember 19, szerda, 17 óra, QBF14
- Jelentkezés:
  - > https://goo.gl/forms/5gUmoVPwagRN35fD3

Activity életciklus

- Rajzoljuk le az Activity életciklust!



Activity életciklus



Hogy is volt?

- Egy Android alkalmazás milyen komponensekből épülhet fel?
- Mi a Service komponens?
- Miket kell tartalmaznia a manifest állománynak?
- Az Activity callback életciklus-függvények felüldefiniálásakor meg kell-e hívni kötelezően az ős osztály implementációját?
- Ha A Activity-ből átváltunk B Activity-re, milyen sorrendben hívódnak meg az életciklus függvények?
- Magyarázza el az Activity Back Stack működési elvét!

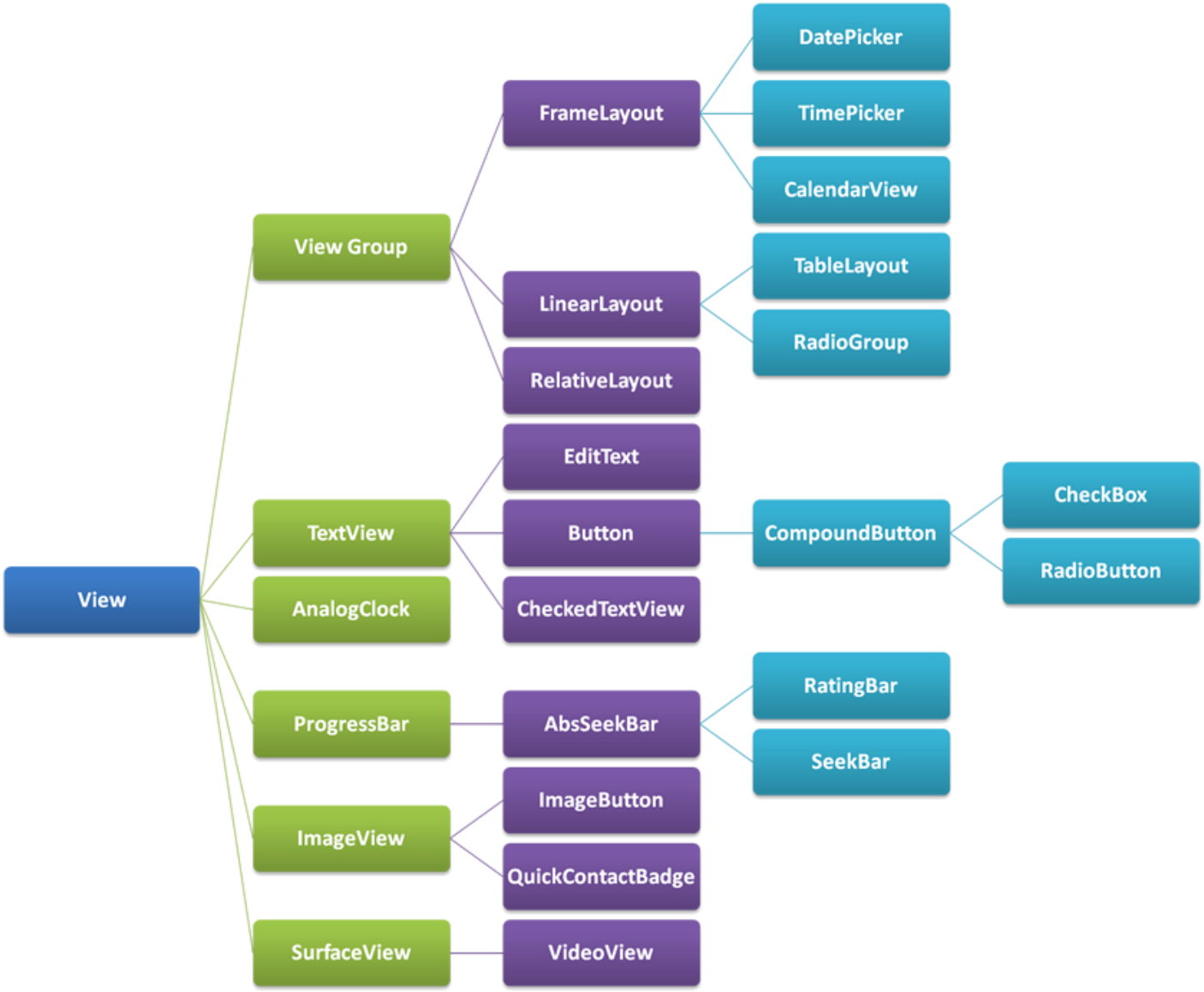
Hogy is volt?

- Mit értünk a sűrűségfüggetlen pixel fogalom alatt?
- Egy 320 dpi-s képernyőn, 1 dp mennyi fizikai pixelnek felel meg ?
- Vázolja fel egy Android alkalmazás kódját, mely egy gombot jelenít meg és a gombot lenyomva a „Clicked!” szöveg jelenik meg egy Toast-ban!
- Hogy biztosítja az Android a lokalizáció támogatását?

Tartalom

- UI építő elemek
  - Layout (ViewGroup)
  - View-k
- Android Fragment framework
  - Mik azok a Fragment-ek, hogyan használjuk?
- Support library
  - Fragment backport, ViewPager, TabStrip, egyebek
- RecyclerView és egyedi felület elem tervezés
- Felugró ablakok, animációk, rajz erőforrások
- Android felület tervezési javaslatok
- Összefoglalás

Android UI architektúra



Layout-ok

Android felhasználói felület felépítése

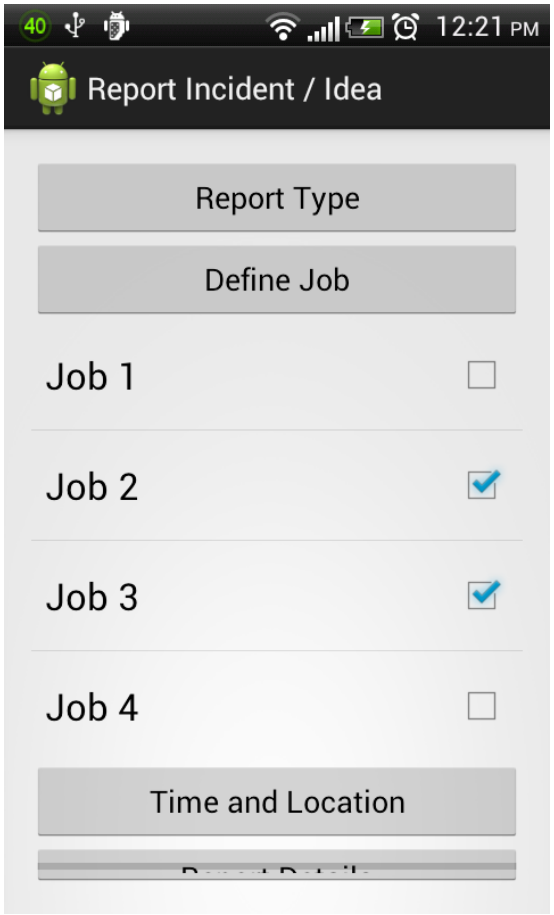
- Minden elem a View-ból származik le
- Layout-ok (elrendezések):
  - ViewGroup leszármazottak
  - ViewGroup is a View-ból származik le!
- ViewGroup-ok egymásba ágyazhatók
- Saját View és ViewGroup is készíthető, illetve a meglevők is kiterjeszthetők

Layout-ok (ViewGroup)

- LinearLayout
- RelativeLayout
- ConstraintLayout
- AbsoluteLayout (NEM használjuk!)
- GridLayout
- RecyclerView
- Teljes lista:
  - http://developer.android.com/reference/android/view/ViewGroup.html

LinearLayout

- LinearLayout != Lista

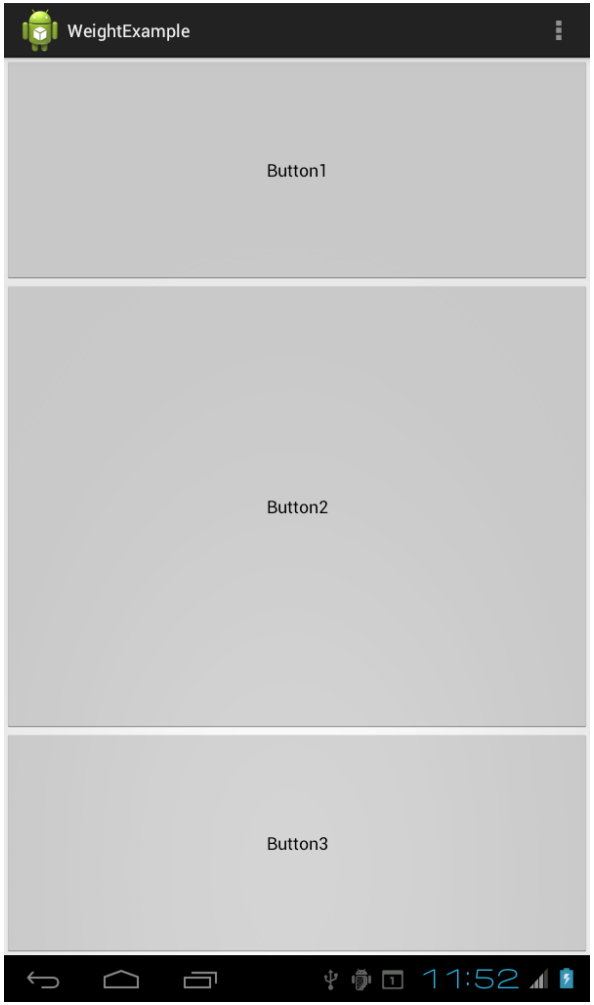


Súlyozás Layout tervezéskor

- Megadható egy layout teljes súly értéke (weightSum)
- Elemek súly értéke megadható és az alapján töltődik ki a layout
  - layout_weight érték
  - A megfelelő width/height ilyenkor 0dp legyen!
- Hasonló, mint HTML-ben a %-os méret megadás

Layout súlyozás példa

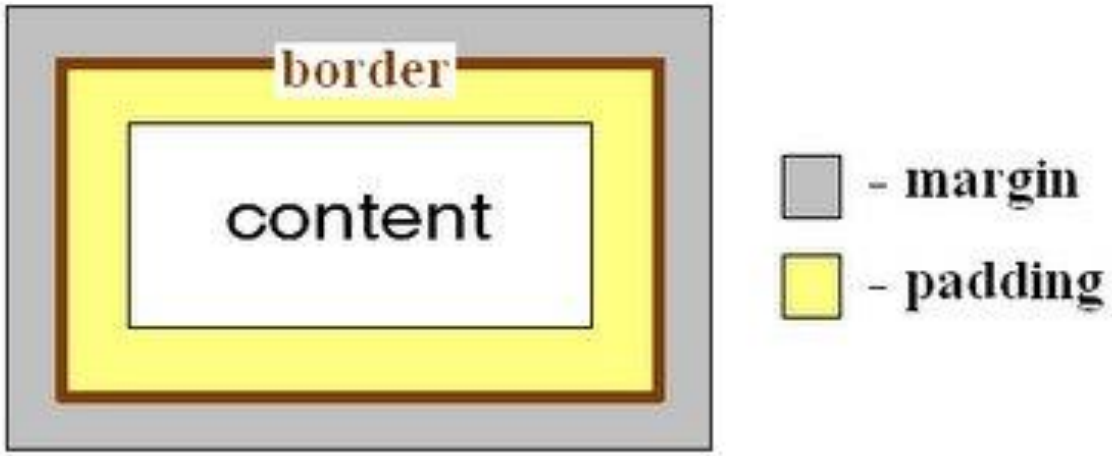
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:weightSum="4"
    android:orientation="vertical">

    <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Button1" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:text="Button2" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Button3" />

</LinearLayout>
```



LinearLayout példák

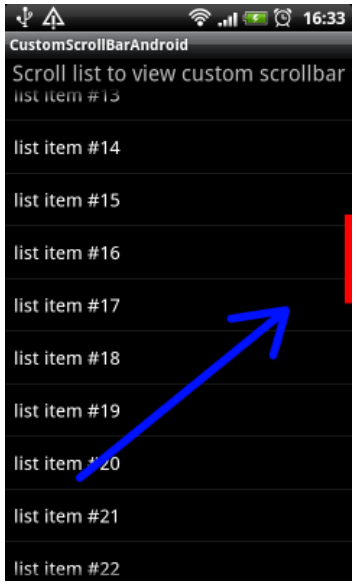
- Jellemző paraméterek:
  - Margin, padding
  - Gravity
  - ScrollView
  - Weight

Padding és Margin



ScrollView

- ScrollView és HorizontalScrollView
- Layout container, amely scrollozást tesz lehetővé, ha a benne levő tartalom „nagyobb”
- Nem kötelező a teljes képernyőt kitöltenie
- Egy layout/képernyő több ScrollView-t is tartalmazhat

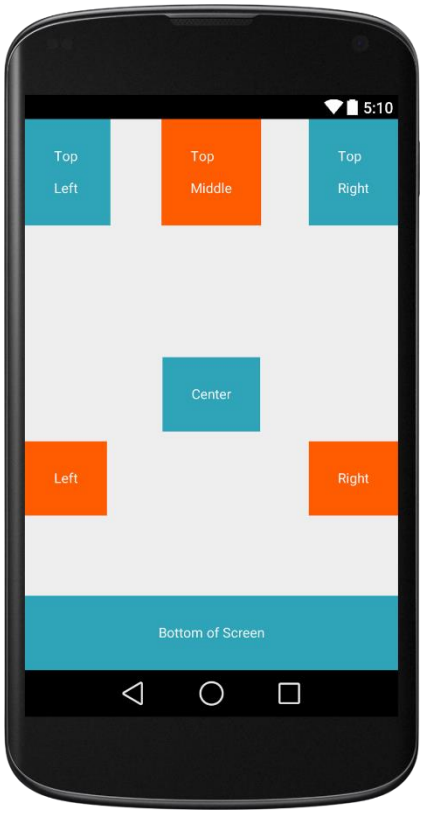


ScrollView példa

```
<ScrollView xmlns:android=
      "http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    android:fillViewport="false">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <ImageView
            android:id="@+id/imageView"
            android:layout_width="wrap_content"
            android:layout_height="200dp"
            android:scaleType="centerCrop"
            android:src="@drawable/image" />

        ...
    </LinearLayout>
</ScrollView>
```

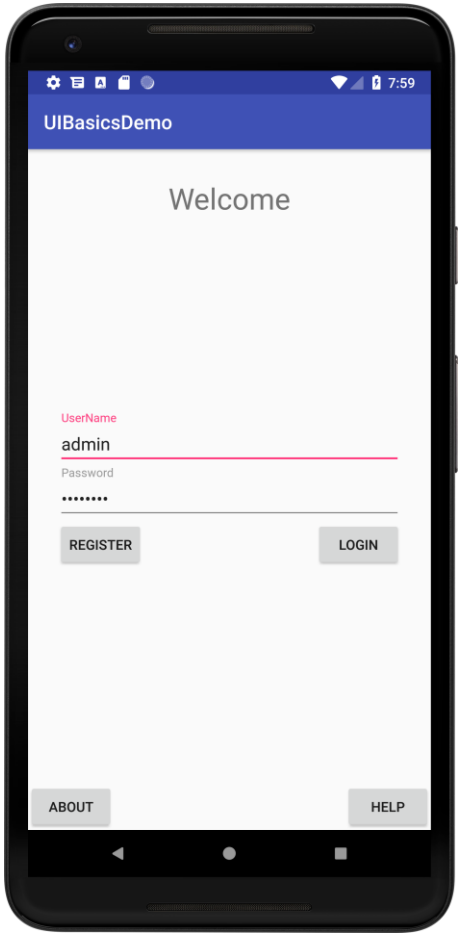
RelativeLayout

- Elemek egymáshoz való viszonya definiálható
- Demo



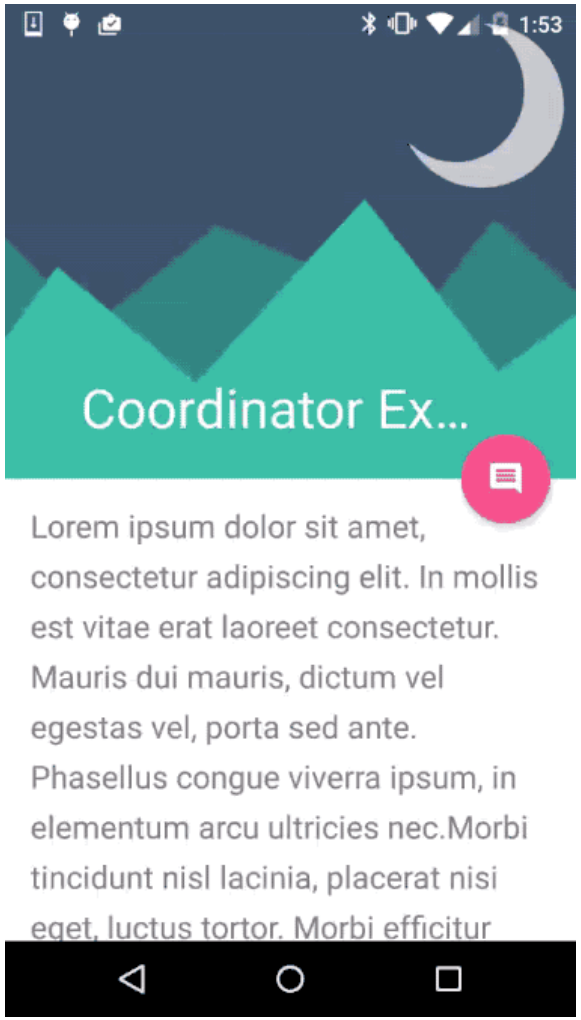
Gyakoroljunk

- Készítsünk egy Login képernyőt



CoordinatorLayout, AppBarLayout

- CoordinatorLayout: továbbfejlesztett FrameLayout
- CoordinatorLayout fő feladatai:
  - Felső szintű alkalmazás UI irányelv
  - Konténer, mely támogatja a beépített elemek material stílushoz igazodó elhelyezkedését
- Behavior paraméterekkel meghatározható a kapcsolódó elemek elhelyezése
- AppBarLayout csatolható hozzá, mely a material design-hez illeszkedő scrollozást támogatja



Nézetek (Widgetek/”View”-k)

View-k 1/2

- *Button, EditText, CheckBox, RadioButton, ToggleButton*

ImageButton

ListView

GridView

Spinner

AutoCompleteTextView

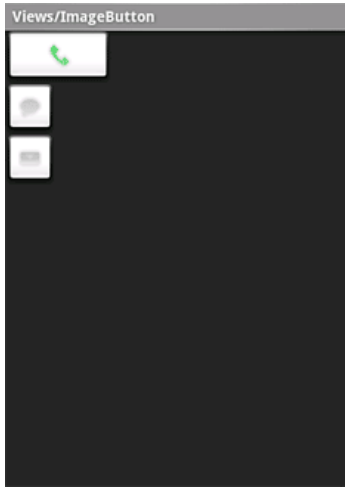
Gallery

ImageSwitcher

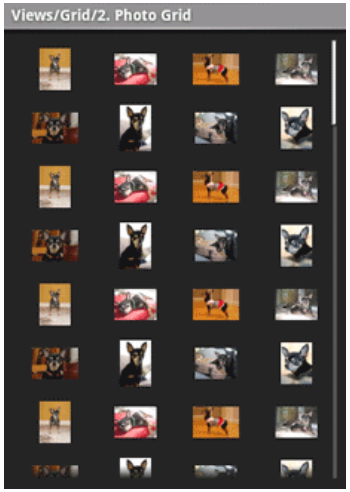
DatePicker, TimePicker

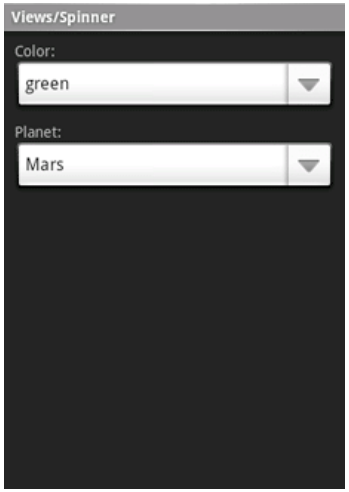
View-k 2/2

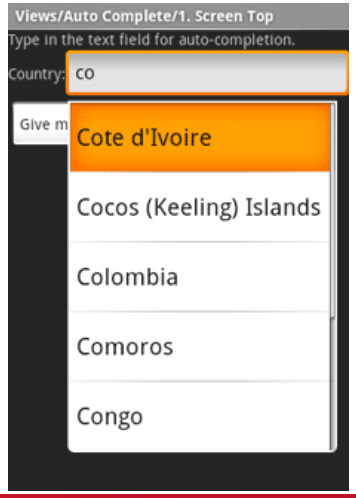

Views/Controls/1. Light Theme
Save
Checkbox 1
Checkbox 2
RadioButton 1
RadioButton 2
Star
OFF
OFF

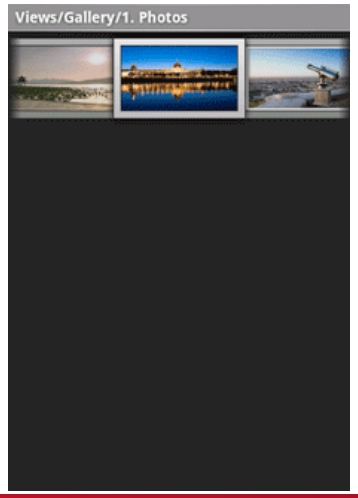

Views/ImageButton

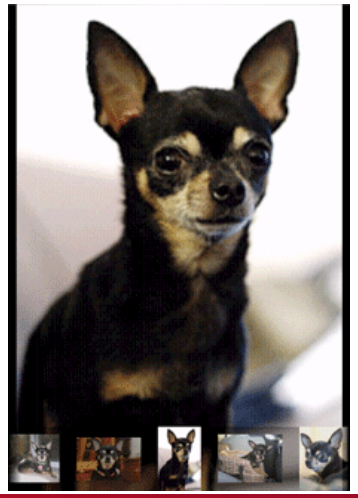

Views/Lists/1. Array
Abbaye de Belloc
Abbaye du Mont des Cats
Abertam
Abondance
Ackawi
Acorn
Adelost


Views/Grid/2. Photo Grid


Views/Spinner
Color:
green
Planet:
Mars


Views/Auto Complete/1. Screen Top
Type in the text field for auto-completion.
Country:
co
Give m
Cote d'Ivoire
Cocos (Keeling) Islands
Colombia
Comoros
Congo


Views/Gallery/1. Photos




Views/Date Widgets/1. Dialog
8-12-2008 20:49
change the date
Thu, August 14, 2008
Aug
14
2008
Set
Cancel

API gazdagsága (globálisan igaz az Androidra)

- Hogy valósítanátok ezt meg? (nem sok *TextView* egymás után☺)

- Megoldás:
  - http://developer.android.com/reference/android/text/SpannableString.html
  - http://androidcocktail.blogspot.hu/2014/03/android-spannablestring-example.html

Egyedi nézetek – külső könyvtárak

- https://github.com/wasabeef/awesome-android-ui/

Dinamikus UI kezelés - LayoutInflater

- LayoutInflater feladata:
  - XML-ben összeállított felületi elemek példányosítása
- Használati mód:

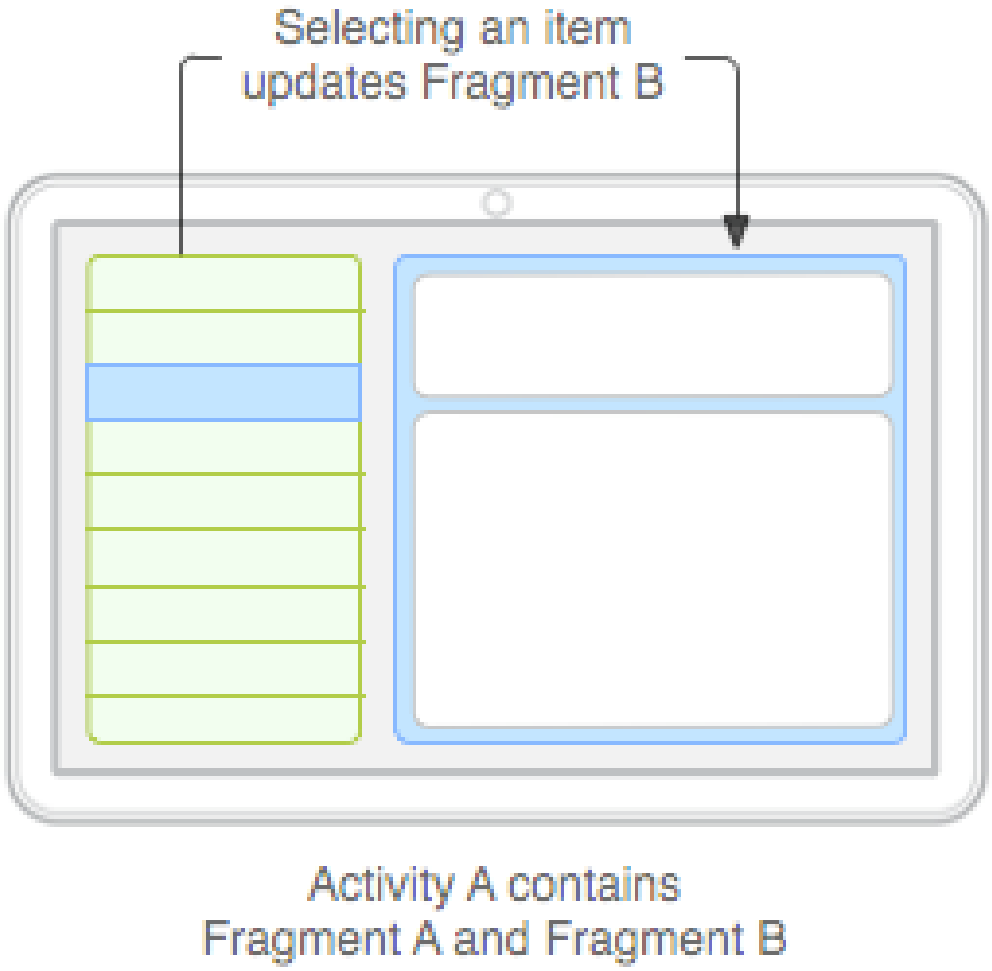
```
val myView = getLayoutInflater().inflate(
  R.layout.activity_main, null)
```

FRAGMENT-EK

Fragmentek

- Mik azok a Fragmentek?
  - Elsősorban: A képernyő egy nagyobb részéért felelős objektumok
  - Továbbá: A háttérben munkát végző objektumok is lehetnek
- Miért kellenek nekünk?
  - Nagy képernyőméret = több funkció egy képernyőn = bonyolultabb Activity-k
  - Fragment-ekkel modulárisabb, rugalmasabb architektúra építhető

Fragmentek


Selecting an item
updates Fragment B
Activity A contains
Fragment A and Fragment B

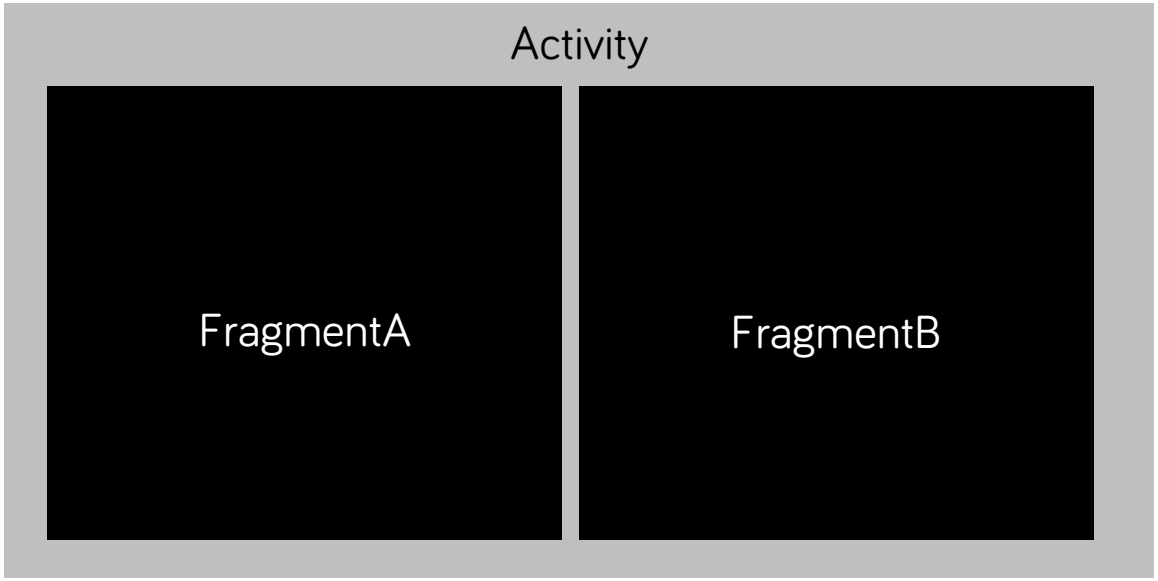
Fragmentek

- Miben másabb mint az Activity?
  - Kisebb granualitás, nem mindig teljes képernyő egy fragment
  - Az életciklusa nem mindig egyezik, pl. le lehet csatolni egy fragmentet úgy, hogy az activity előtérben marad.

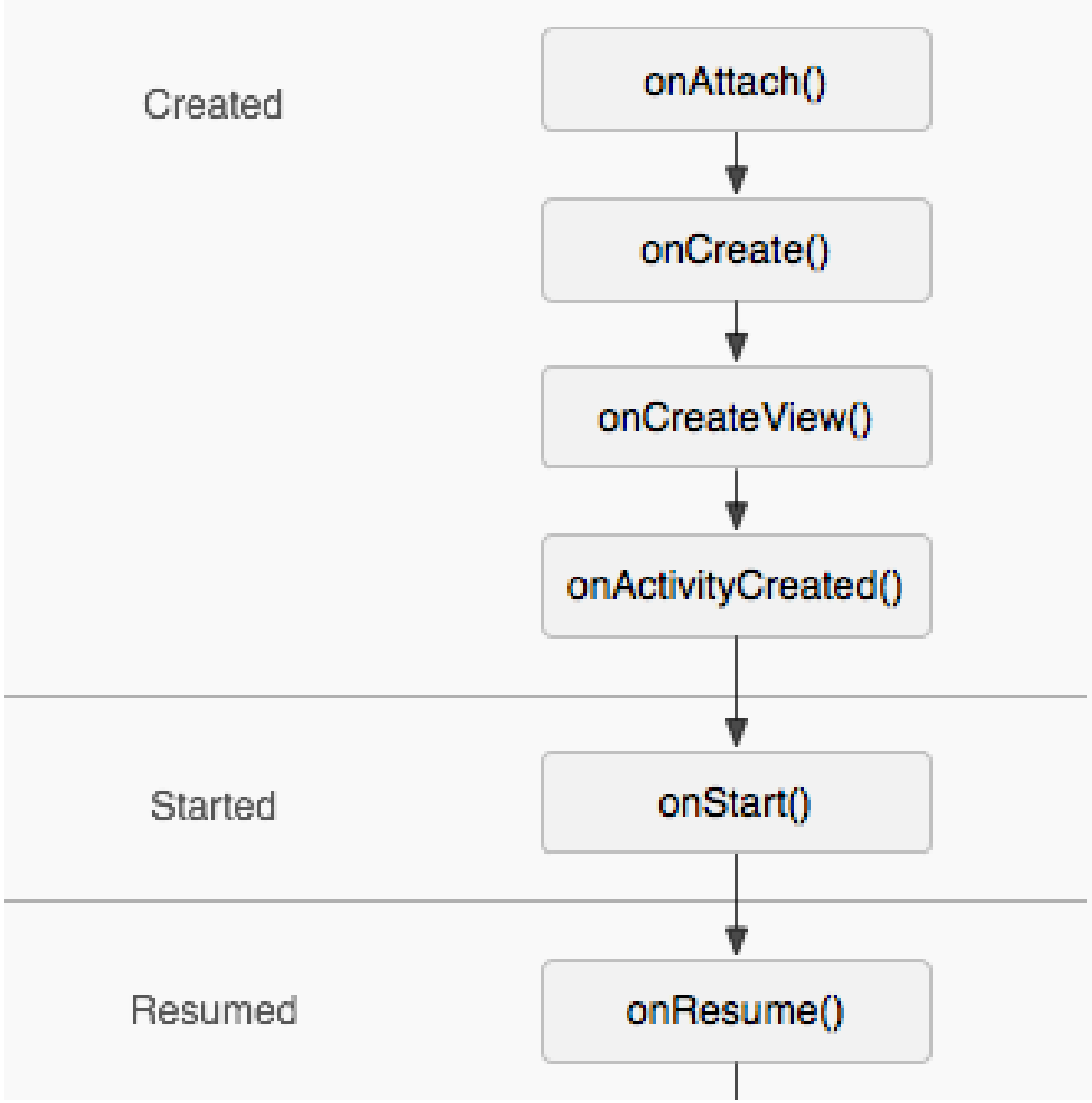
- Miben másabb mint egy Custom View
  - Összetett életciklus, mely az activity-t is figyelembe veszi
  - Előny, de hátrány is lehet!

Fragment és Activity

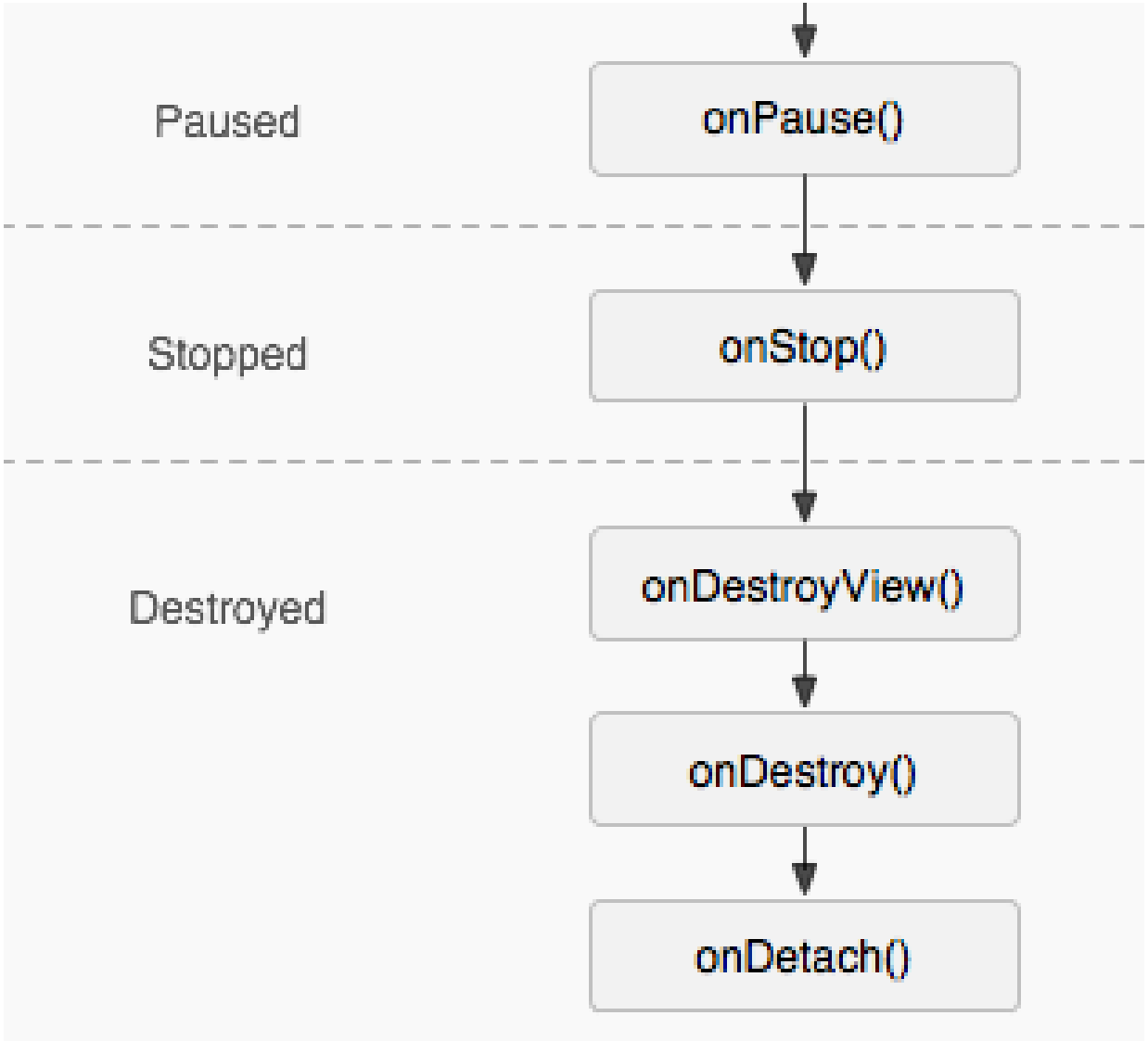
- Egy Fragment mindig egy Activity-hez csatoltan jelenik meg
- Az Activity életciklusa ráhatással van a Fragmentére

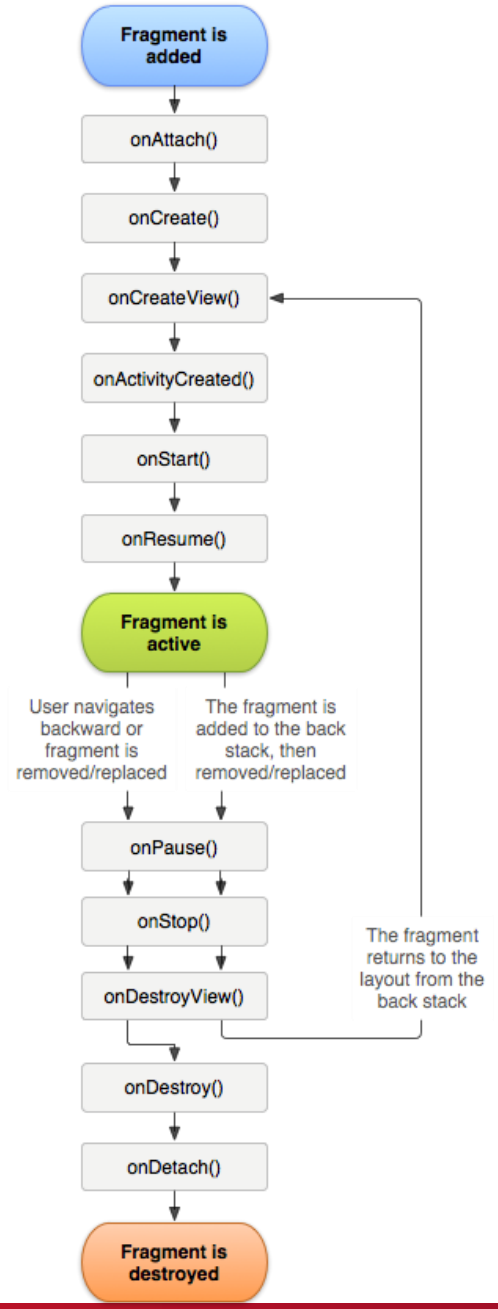


Fragment életciklus I.



Fragment életciklus II.




Fragment is added
onAttach()
onCreate()
onCreateView()
onActivityCreated()
onStart()
onResume()
Fragment is active
User navigates backward or fragment is removed/replaced
The fragment is added to the back stack, then removed/replaced
onPause()
onStop()
onDestroyView()
The fragment returns to the layout from the back stack
onDestroy()
onDetach()
Fragment is destroyed

Support Library

- Fragment API az Android API része Android 3.0 óta
- Support Library – Fragmentek Android 3.0 előtt
- Ma már 3.0 alá nem is targetáluk, miért használjuk?
  - Android verzionként eltér a beépített Fragment api (~~android.app.Fragment~~)
  - A support környtárral minden Android verzión ugyan azt kapjuk (android.support.v4.app.Fragment)

Nem alkalmazás komponens

- Mi hozzuk létre, nem a rendszer
- Nem kell feltüntetni a manifestben
- Nem intentekkel kommunikálunk
  - Mezei függvényhívás az objektumon
  - pl. Activity hívja a Fragment objetumon
  - getActivity()/activity property – szülő activity

UI Fragment készítése...

- A megjelenítendő View-hierarchiát az .onCreateView() metódusban kell visszaadni

```
class FragmentProfile : Fragment() {

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View? {
        val rootView = inflater.inflate(R.layout.fragment_profile, container, false)
        return rootView
    }

}
```

… és csatolása

- Dinamikusan
  - Az Activity futás közben tölti be a megfelelő Fragment-eket, adott ViewGroup-okba
  - Fragment-Tranzakciókkal módosítható
- Statikusan
  - Az Activity-hez tartozó layout-ban beégetjük a Fragment-et, nem módosítható később
  - <fragment …/> tag

Statikus csatolás példa

```
<fragment class="hu.bme.aut.fragment.MenuListFragment"
    android:tag="MenuListFragment"
    android:layout_width="0dip"
    android:layout_height="fill_parent"
    android:layout_weight="1"/>
```

A FragmentManager

- A FragmentManager-el menedzselhetők a Fragment-ek
  - Activity: supportFragmentManager property
  - FragmentTransaction indítása
  - Aktív Fragment-ek közt keres
    - Tag alapján
    - ID alapján
  - **Fragment-stack-et menedzseli**

A FragmentManager

- Az activitynek a FragmentActivityből kell származnia – Neki van FragmentManagerje
- Activity.supportFragmentManager== fragment.fragmentManager
- Kezeli a fragmenteket, backstacek, állapotmentést … stb.
- Fragmenten belül fragmentek, lehet: Fragment.childFragmentManager

FragmentTransaction osztály I.

- Ezen keresztül módosíthatók az aktív Fragment-ek
- A FragmentManager .beginTransaction() metódusával indítható
- Fontosabb műveletek:
  - .add(...) / .remove(...) / .replace(...)
    - Fragment példányok le- és felcsatolása az adott Activity-re
  - .commit()
    - Tranzakció végrehajtása

FragmentTransaction osztály II.

- .show(...) / .hide(...)
  - Fragment példány elrejtése / újra megjelenítése
- .setTransition(...) / .setCustomAnimations(...)
  - A tranzakció végrehajtásakor lejátszandó animáció beállítása
- .addToBackStack(...)
  - Rákerüljön-e a FragmentTransaction backstack-re a tranzakció?
- .commit()
  - Tranzakció végrehajtása

FragmentTransaction példa I.

- Fragment kicserélése:

```
val fragment=DetailsFragment.newInstance()

val ft = supportFragmentManager.beginTransaction()
ft.replace(R.id.fragmentContainer, fragment, DetailsFragment.TAG)
ft.commit()
```

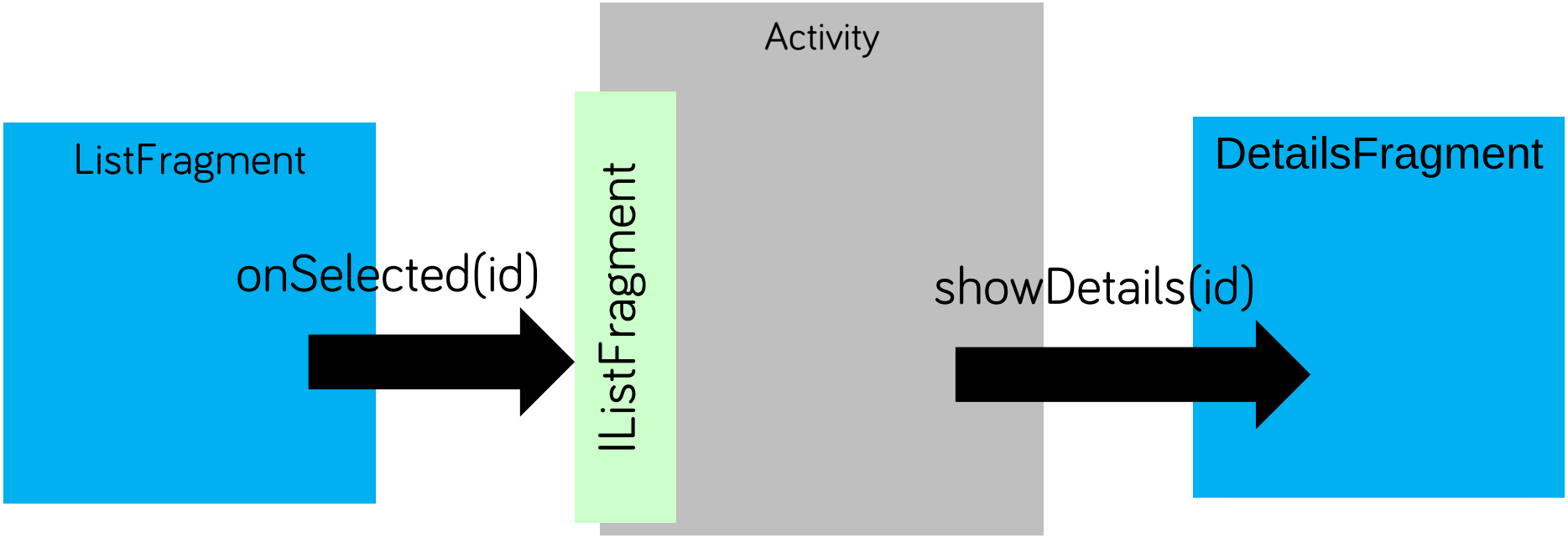
FragmentTransaction példa II.

- Fragment hozzáadása, a tranzakciót a backstack-re téve:

```
val fragment=DetailsFragment.newInstance()

val ft = supportFragmentManager.beginTransaction()
ft.add(R.id.fragmentContainer, fragment, TAG)
ft.setCustomAnimations(R.anim.slide_in_top,R.anim.slide_out_bottom)
ft.addToBackStack(null)
ft.commit()
```

Fragment kommunikáció I.

- Egy Fragment-nek egységbezártnak kell lennie
  => közvetett kommunikáció
  - > Az Activity közvetít



Fragment kommunikáció II.

- Ha mégis szükség van a közvetlen Fragment-kommunikációra:
  - > Fragment.targetFragment propertyvel állítható/elérhető
- Az így létrejövő kapcsolat túléli a forgatásokat is

Fragment paraméterek

- Szükség lehet paraméter átadás
  - pl. Melyik cikk részleteit jelenítse meg a hírolvasó, stb..
- Első ötlet
  - ~~Mi inicializáljuk – Konstruktor paraméter~~
  - Nem csak mi inicializálhatjuk! – pl. Elforgatás
- Használjuk a Factory pattern-t
  - Fragment.arguments:Bundle property
  - Mentésre kerül a Fragment példánnyal

Fragment paraméterek

- Egy Fragmentet a publikus, paraméter nélküli konstruktorával példányosítunk
- Ha paramétereket kell átadnunk:
  - Példányosításkor Bundle-ben adjuk át őket
    - .setArguments(...) (Kotlinban arguments property)
  - Inicializáláskor ezt a Bundle-t kérjük el
    - .getArguments()
  - A forgatást is túléli az Arguments Bundle

Próbáljuk ki!

FragmentDemo projekt

Mi nem igaz a Fragment-ekre?

A. Önálló életciklussal rendelkeznek.

B. Kötelező felhasználói felületet tartalmazniuk.

C. Dinamikusan és statikusan is csatolhatók.

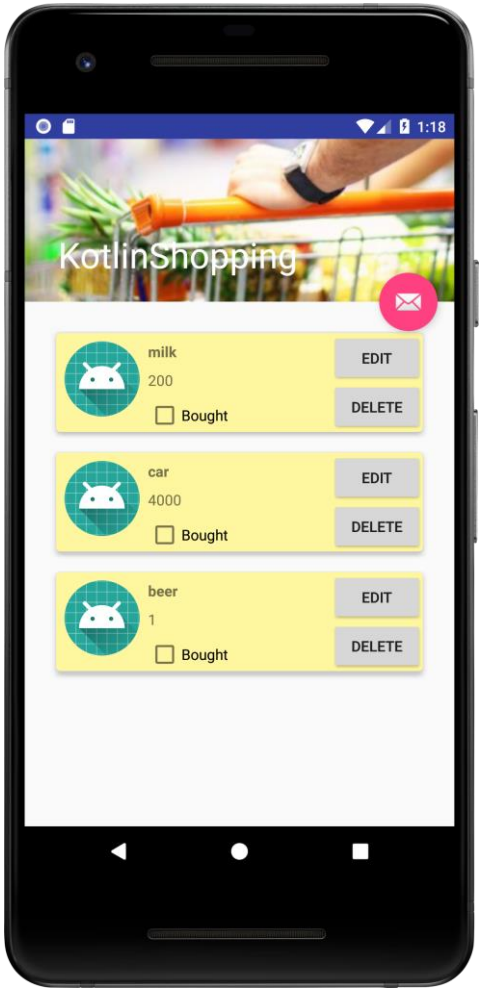
D. A tabletek felhasználói felületének kialakításakor különösen hasznosak.

http://babcomaut.aut.bme.hu/votes/

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

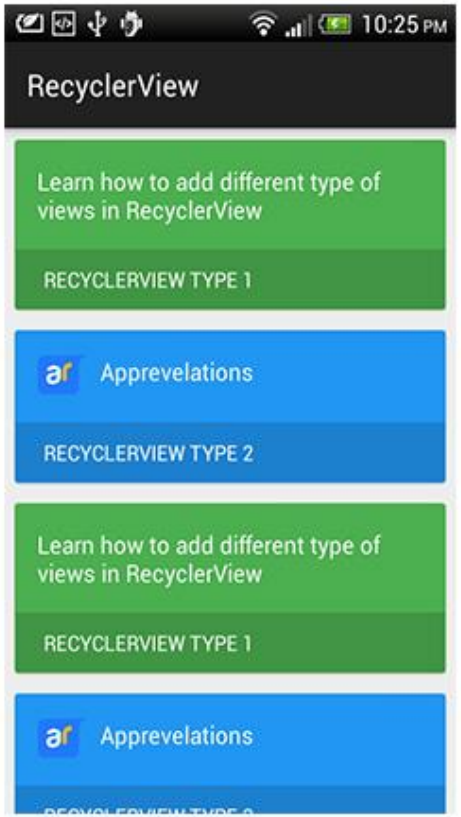
```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    val (name, price, bought) = items[holder.adapterPosition]
    holder.tvName.text = name
    holder.tvPrice.text = price.toString()
    holder.cbBought.isChecked = bought

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

RecyclerView.Adapter<ViewHolder> 3/3

- Elemek száma, hozzáadás, törlés

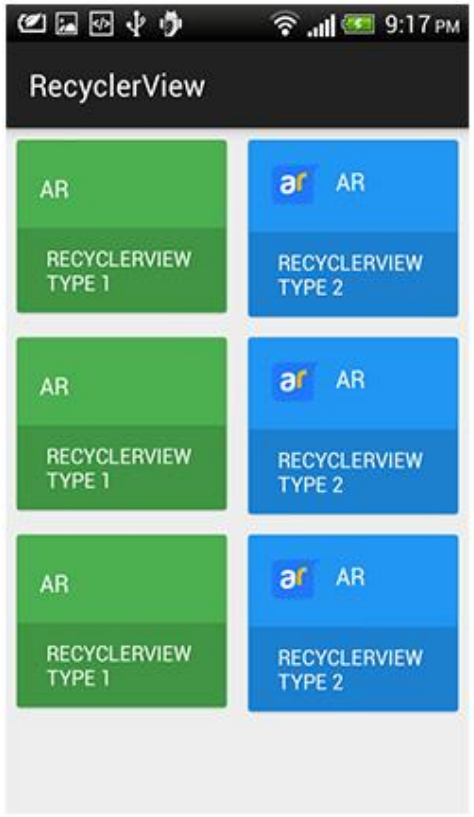
```
override fun getItemCount() = items.size

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

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

RecyclerView LayoutManager-ek



Linear Layout View

Staggered Grid View



Grid View

Különböző elem megjelenítés a RecyclerView-ban

- Elemek/sorok típusa megadható pozíció alapján a *getItemViewType(...)* felüldefiniálásával
- *viewType* paraméter jelzi a megfelelő függvényekben a sor/elem típusát, amely alapján a megjelenítés szabályozható
- *ViewHolder* ismeri a *viewType*-ját

```
// determine which layout to use for the row
@Override
public int getItemViewType(int position) {
    Item item = itemList.get(position);
    if (item.getType() == Item.ItemType.ONE_ITEM) {
        return TYPE_ONE;
    } else if (item.getType() == Item.ItemType.TWO_ITEM) {
        return TYPE_TWO;
    } else {
        return -1;
    }
}
```

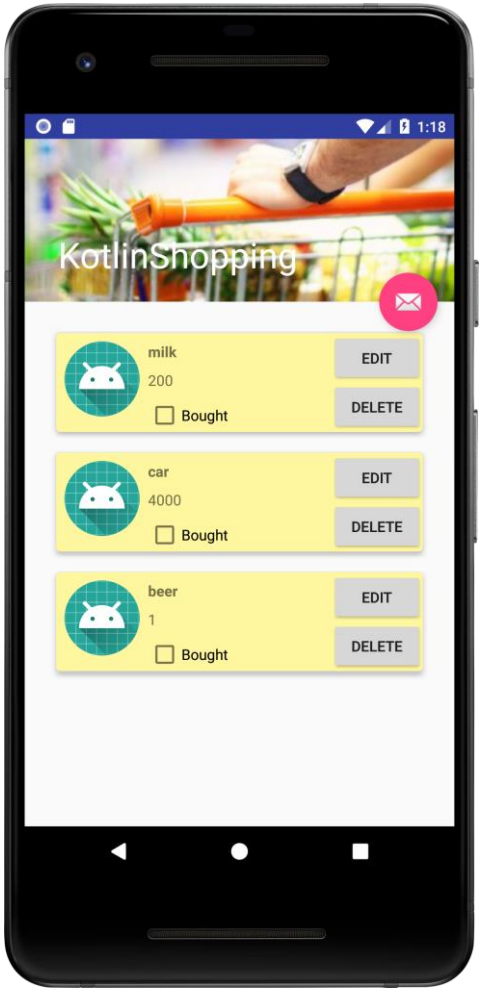
```
// specify the row layout file and click for each row
@Override
public RecyclerView.ViewHolder onCreateViewHolder(
  ViewGroup parent, int viewType) {
    if (viewType == TYPE_ONE) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(
  R.layout.list_item_type1, parent, false);
        return new ViewHolderOne(view);
    } else if (viewType == TYPE_TWO) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(
  R.layout.list_item_type2, parent, false);
        return new ViewHolderTwo(view);
    } else {
        throw new RuntimeException(
          "The type has to be ONE or TWO");
    }
}
```

Lista készítés – fő lépések

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

Bevásárló lista példa - RecyclerView

- Termékek listázása
  - *Név, ár, vásárlás állapota*
- Új termék felvitele
  - DialogFragment
- Törlés
- Szerkesztés
- Touch gesztusok



Összefoglalás

- UI építő elemek
  - Layout (ViewGroup)
  - View-k
- Android Fragment framework
  - Mik azok a Fragment-ek, hogyan használjuk?
- RecyclerView
- Összefoglalás

Kérdések