Felület felosztása

Fragmentek és tipikus használatuk

Balogh Tamás

balogh.tamas@autsoft.hu

Tartalom

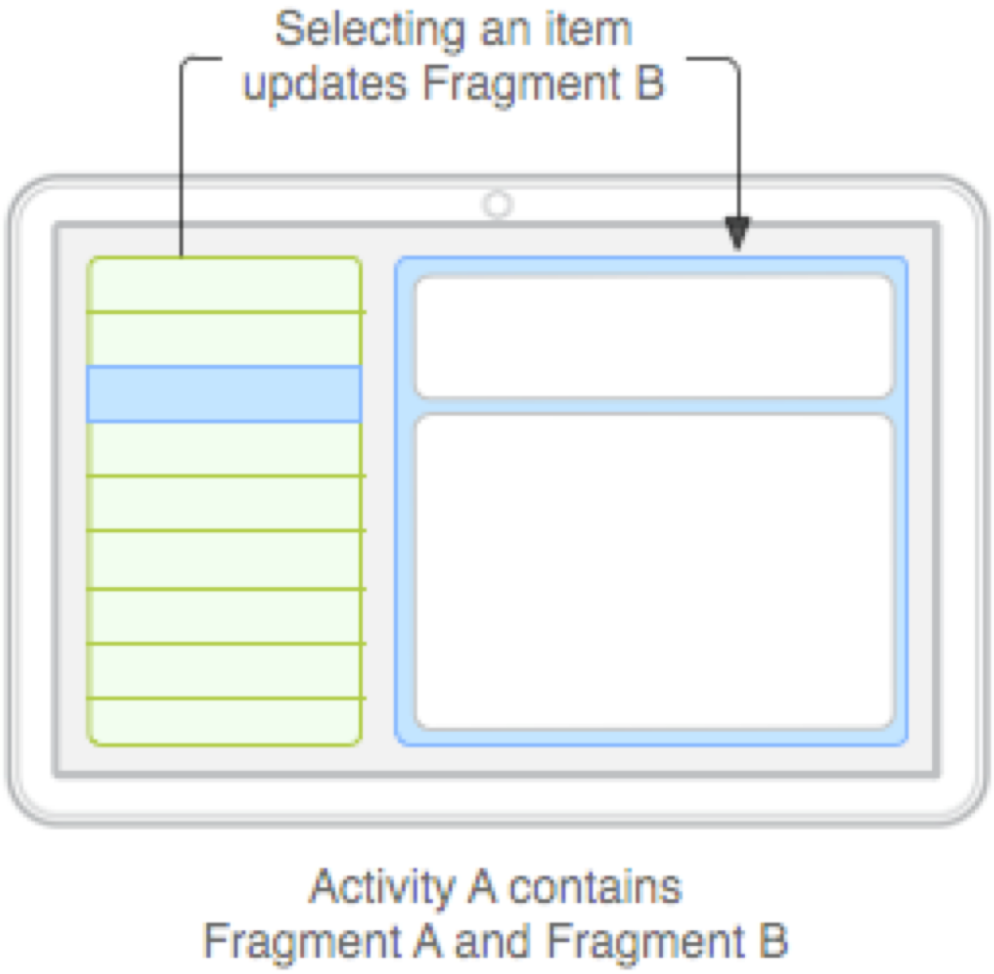
- Fragment fogalma, Fragment életciklus
- Statikus és dinamikus csatolás
- UI nélküli fragmentek
- Telefon és tablet egyidejű támogatása
- Lapozható felületek, tabok
- Komplex menü típusok

Fragmentek

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

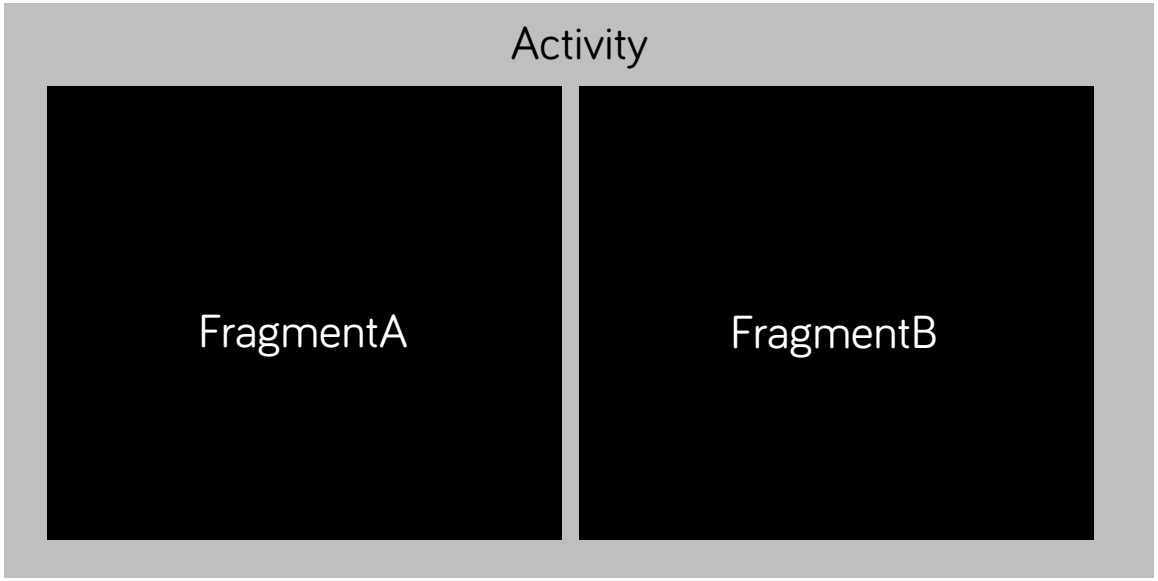
Fragmentek

- 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.

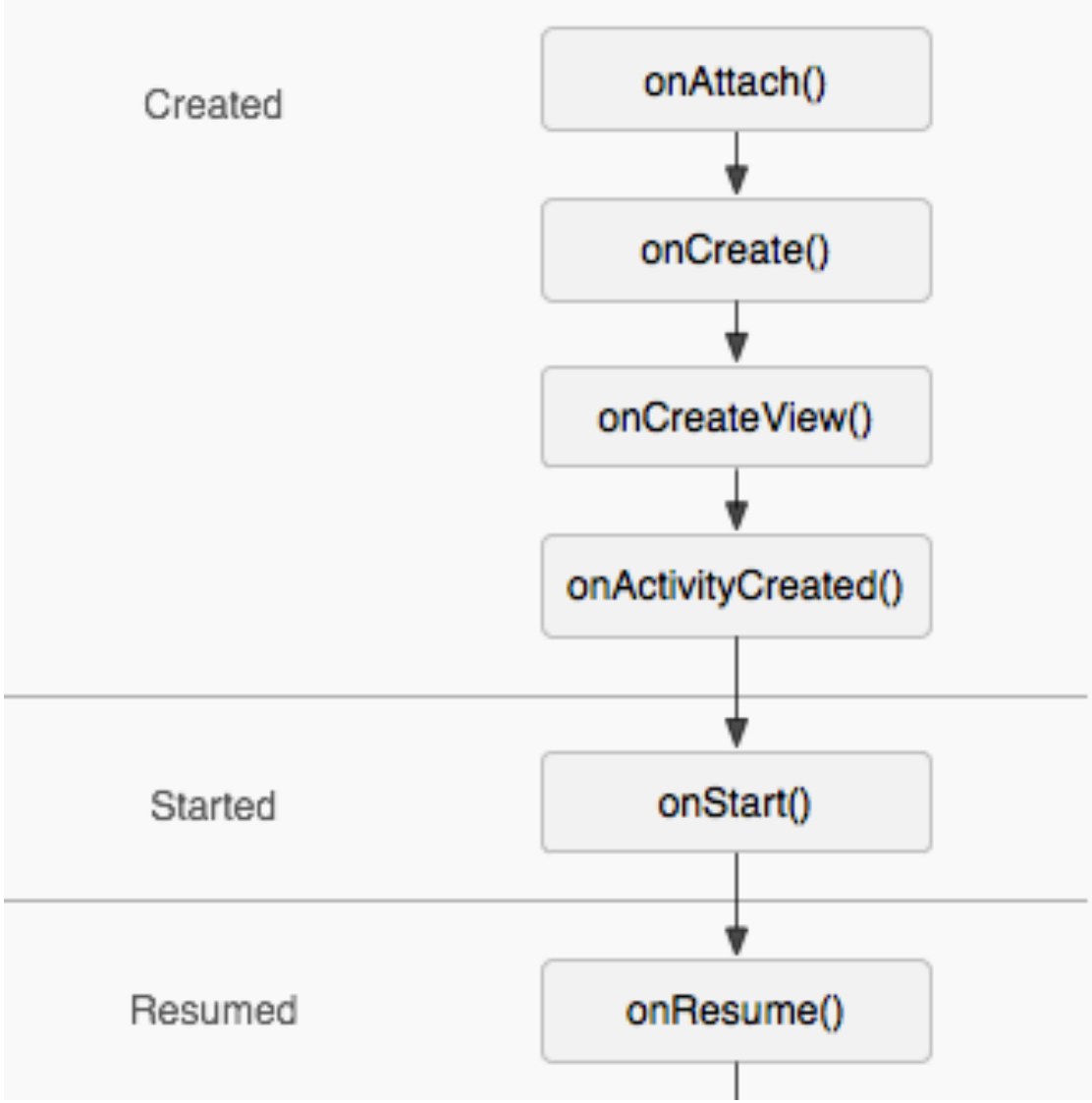
- Miben másabb mint egy Custom View
  - Összetett életciklus, mely az activity-t is figyelembe veszi
  - Előny, de hátrány is lehet!

Fragment és Activity

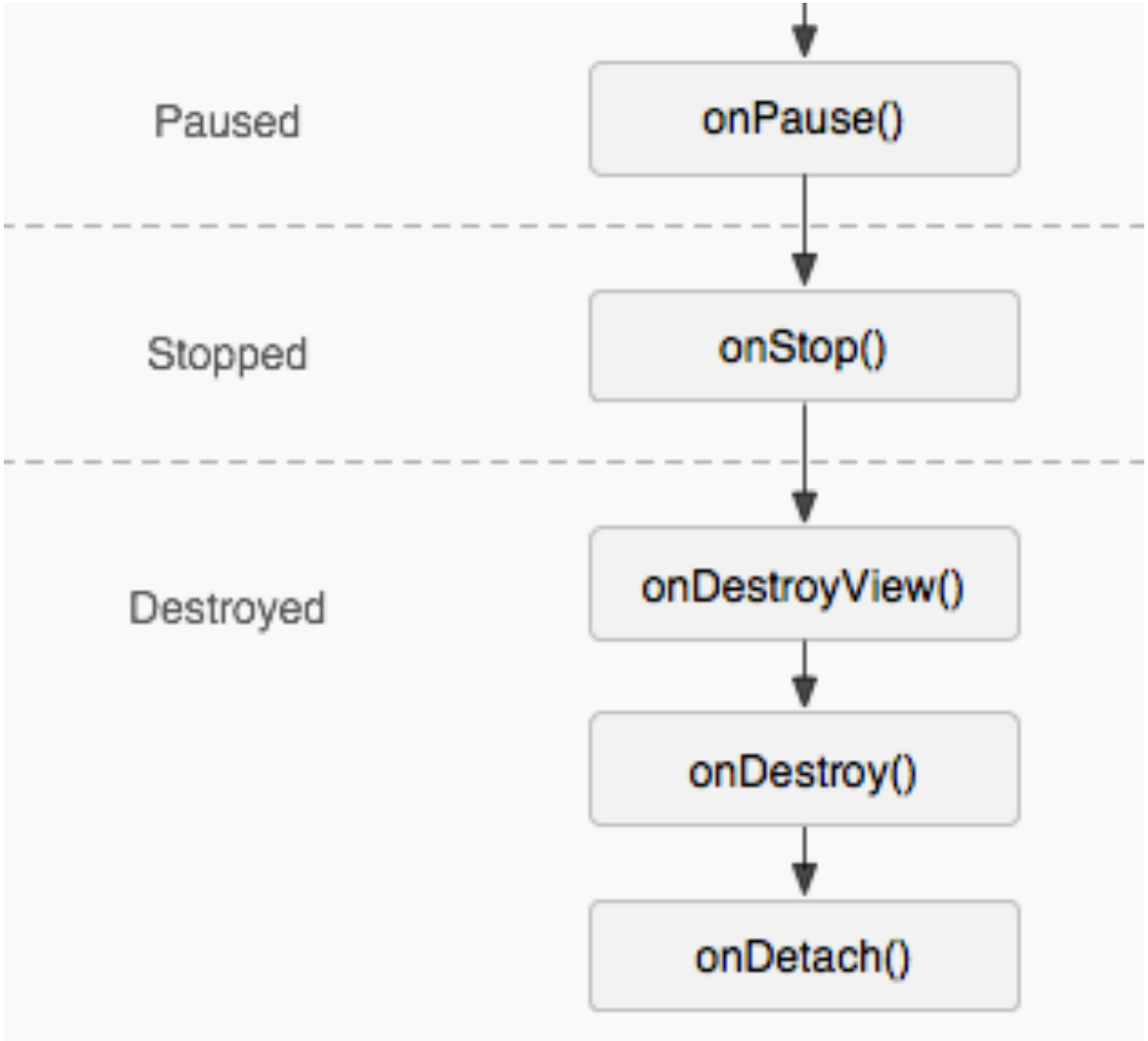
- Egy Fragment mindig egy Activity-hez csatoltan jelenik meg
- Az Activity életciklusa ráhatással van a Fragmentére

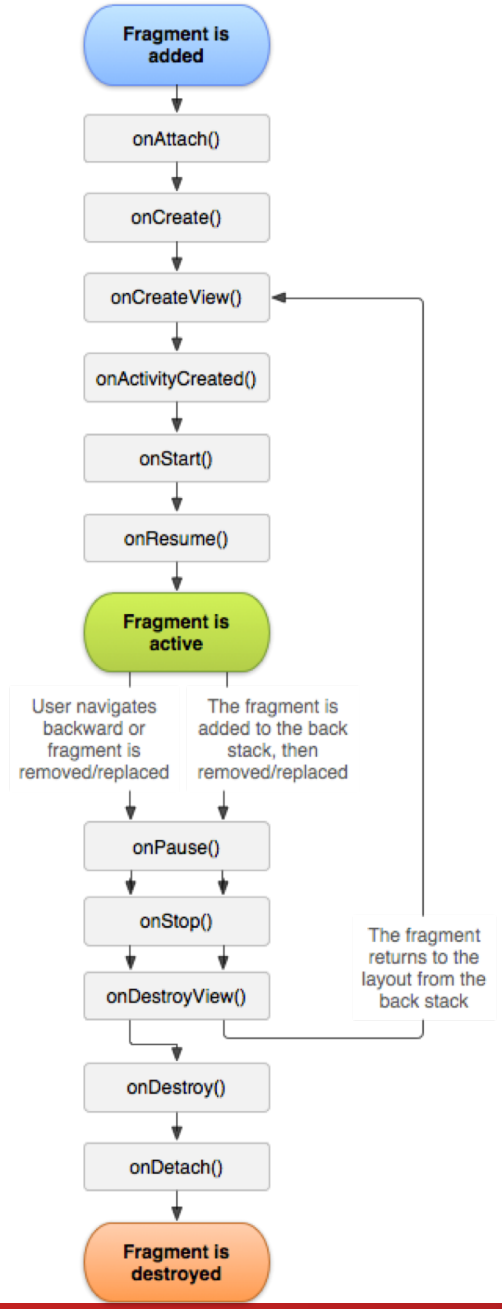


Fragment életciklus I.



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()
```

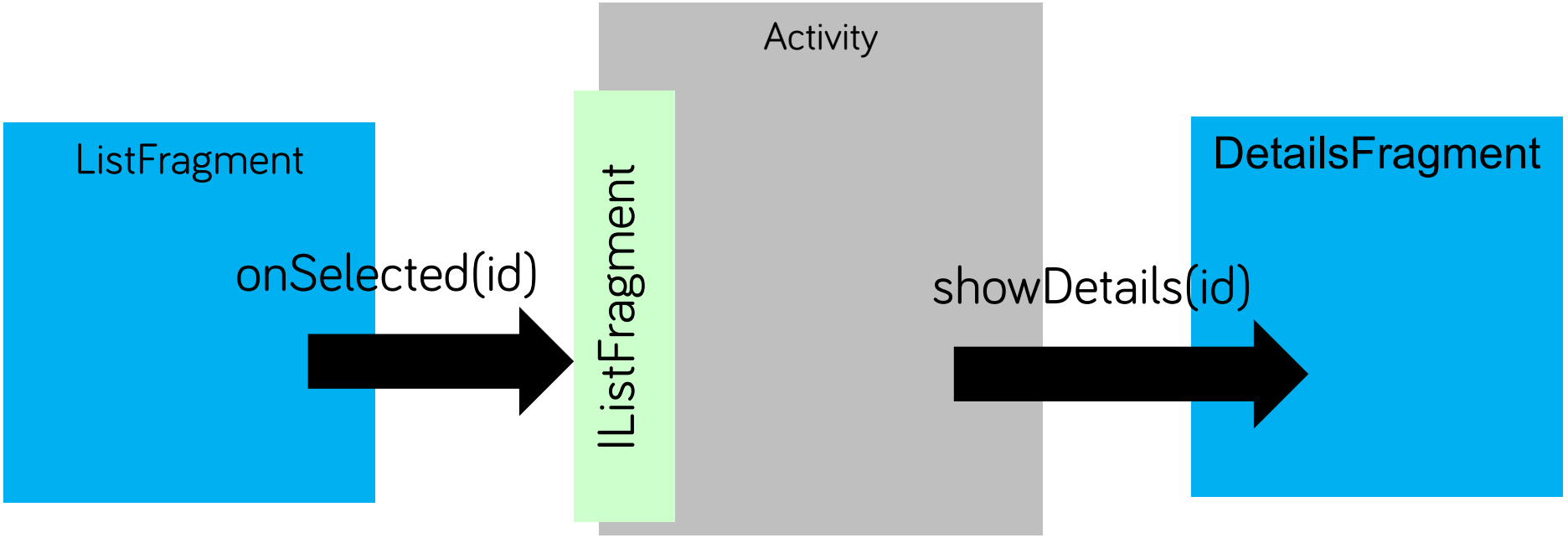
FragmentTransaction példa II.

- 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

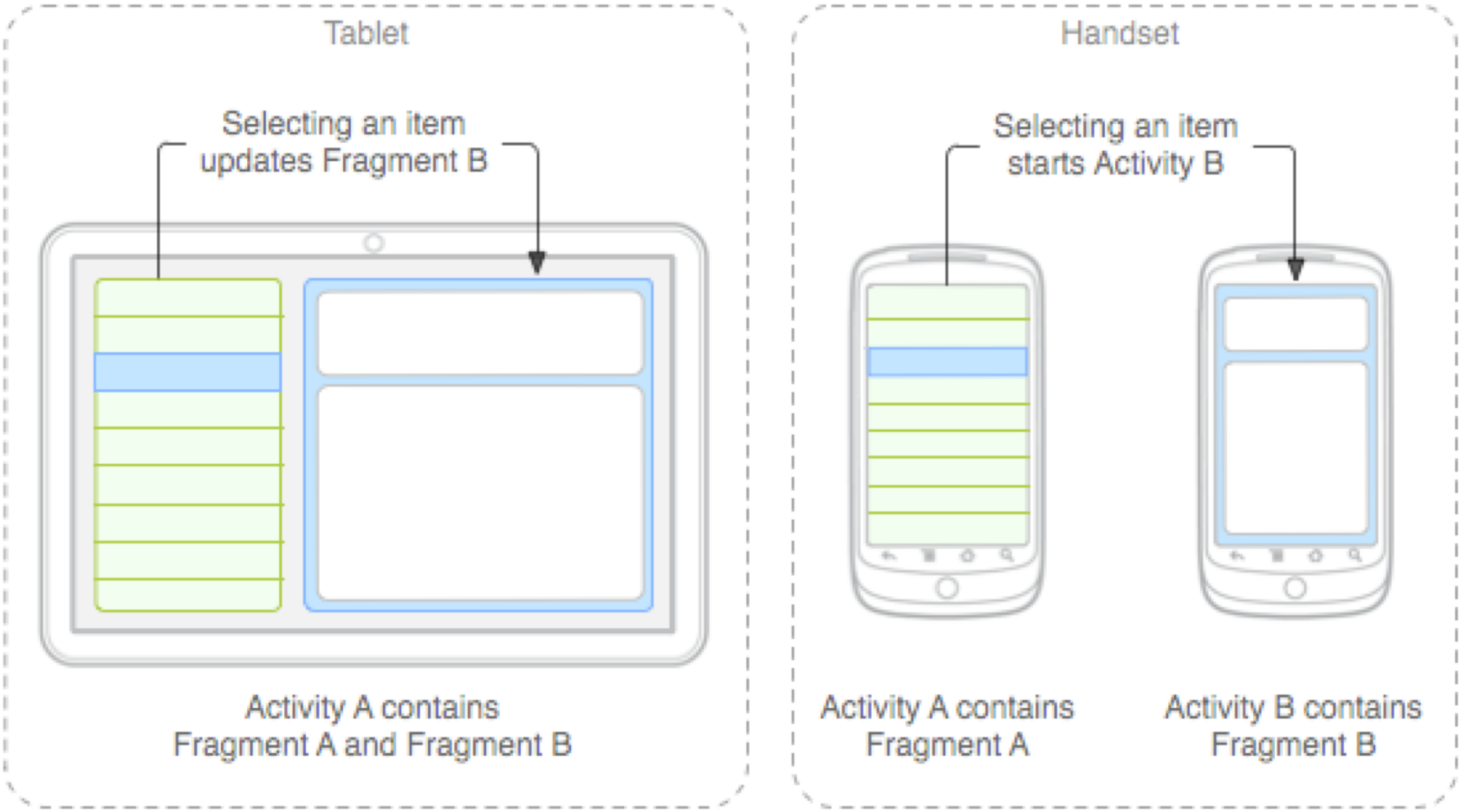
- Egy Fragmentet a publikus, paraméter nélküli konstruktorával példányosítunk
- Ha paramétereket kell átadnunk:
  - Példányosításkor Bundle-ben adjuk át őket
    - .setArguments(...) (Kotlinban arguments property)
  - Inicializáláskor ezt a Bundle-t kérjük el
    - .getArguments()
  - A forgatást is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

Próbáljuk ki!

FragmentDemo projekt

Eltérő képernyő méretek támogatása

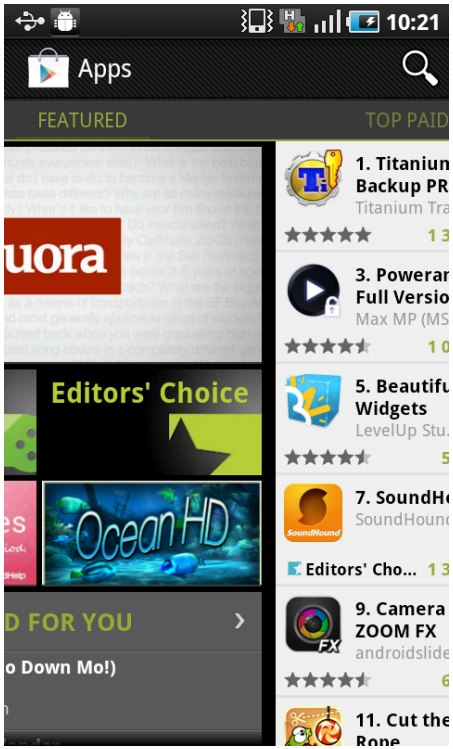
Eltérő képernyőméretek



Lapozható felületek

ViewPager

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

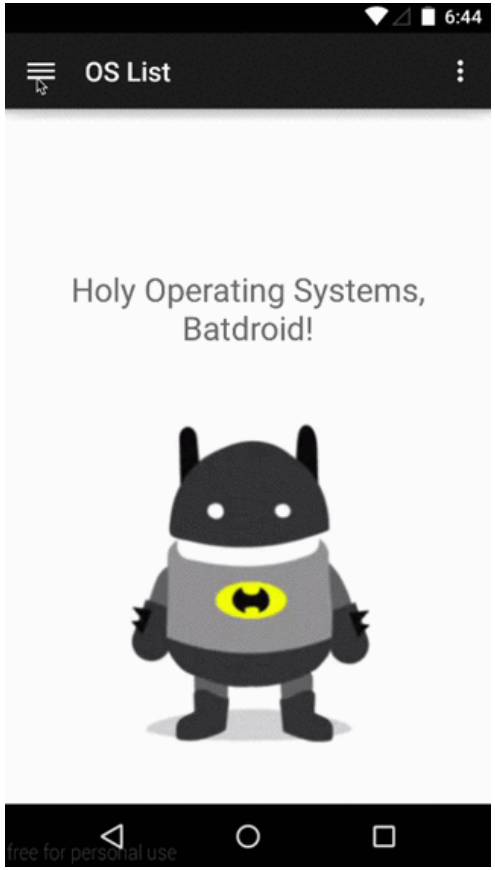


Komplex menü típusok

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

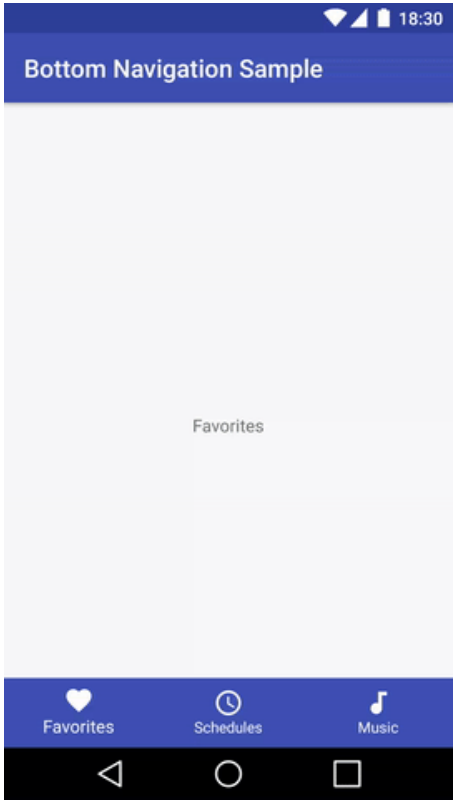
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"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



Köszönöm a figyelmet!

balogh.tamas@autsoft.hu

AutSoft