Mobil és webes rendszerek

Dr. Ekler Péter

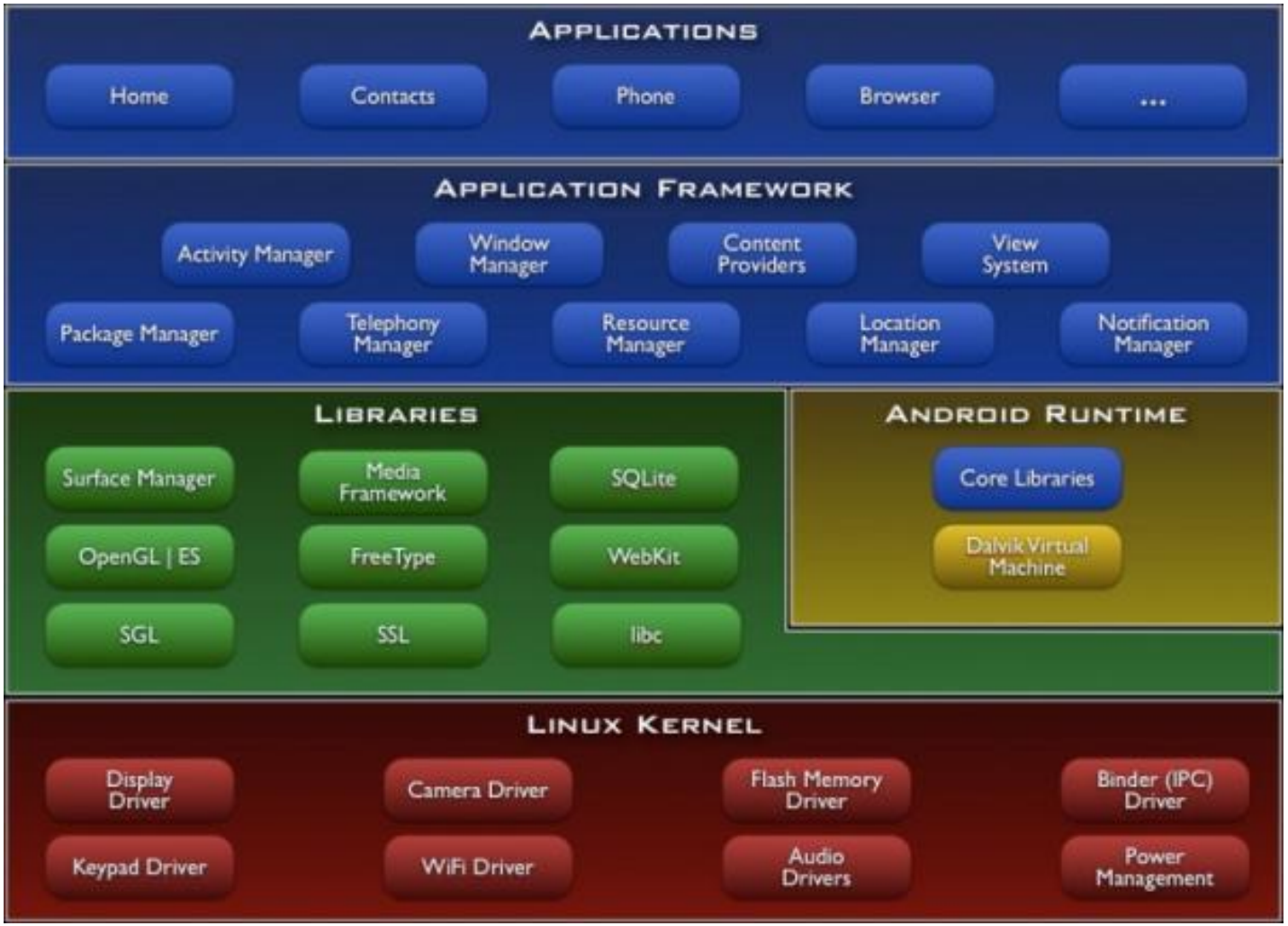
ekler.peter@aut.bme.hu

BME AUT
Department of Automation and Applied Informatics

Tartalom

- Android platform szerkezete
- Fordítás mechanizmusa
- Android alkalmazás komponensei
- Manifest állomány, jogosultságok
- Activity Back Stack
- Multitasking
- Android HelloWorld
- Egyszerű eseménykezelés
- Navigálás Activity-k között
- Felhasználói felülete alapok

Az Android platform szerkezete



A platform felépítése 1/3

- Távolról tekintve a platform felépítése egyszerű és világos
- A **vörös** színnel jelölt rész a Linux kernel, amely tartalmazza:
  - A hardver által kezelendő eszközök meghajtó programjait
  - Ezeket azon cégek készítik el, amelyek az Android platformot saját készülékükön használni kívánják, hiszen a gyártónál jobban más nem ismerheti a mobil eszközbe integrált perifériákat
  - Memória kezelés, a folyamatok ütemezése és az alacsony fogyasztást elősegítő teljesítmény-kezelés

A platform felépítése 2/3

- A Linux kernelen futnak a **zöld** részben található programkönyvtárak/szolgáltatások, mint például: libc, SSL, SQLite, stb. ezek C/C++ nyelven vannak megvalósítva
- Részben ezekre épül a sárga egységben található virtuális gép
  - Nem kompatibilis a Sun virtuális gépével
  - Teljesen más az utasítás készlete
  - Más bináris programot futtat
  - A Java programok nem egy-egy .class állományba kerülnek fordítás után, hanem egy nagyobb Dalvik Executable formátumba, amelynek kiterjesztése .dex, és általában kisebb, mint a forrásul szolgáló .class állományok mérete, mivel a több Java fájlban megtalálható konstansokat csak egyszer fordítja bele a fordító
  - A Java csak mint nyelv jelenik meg!

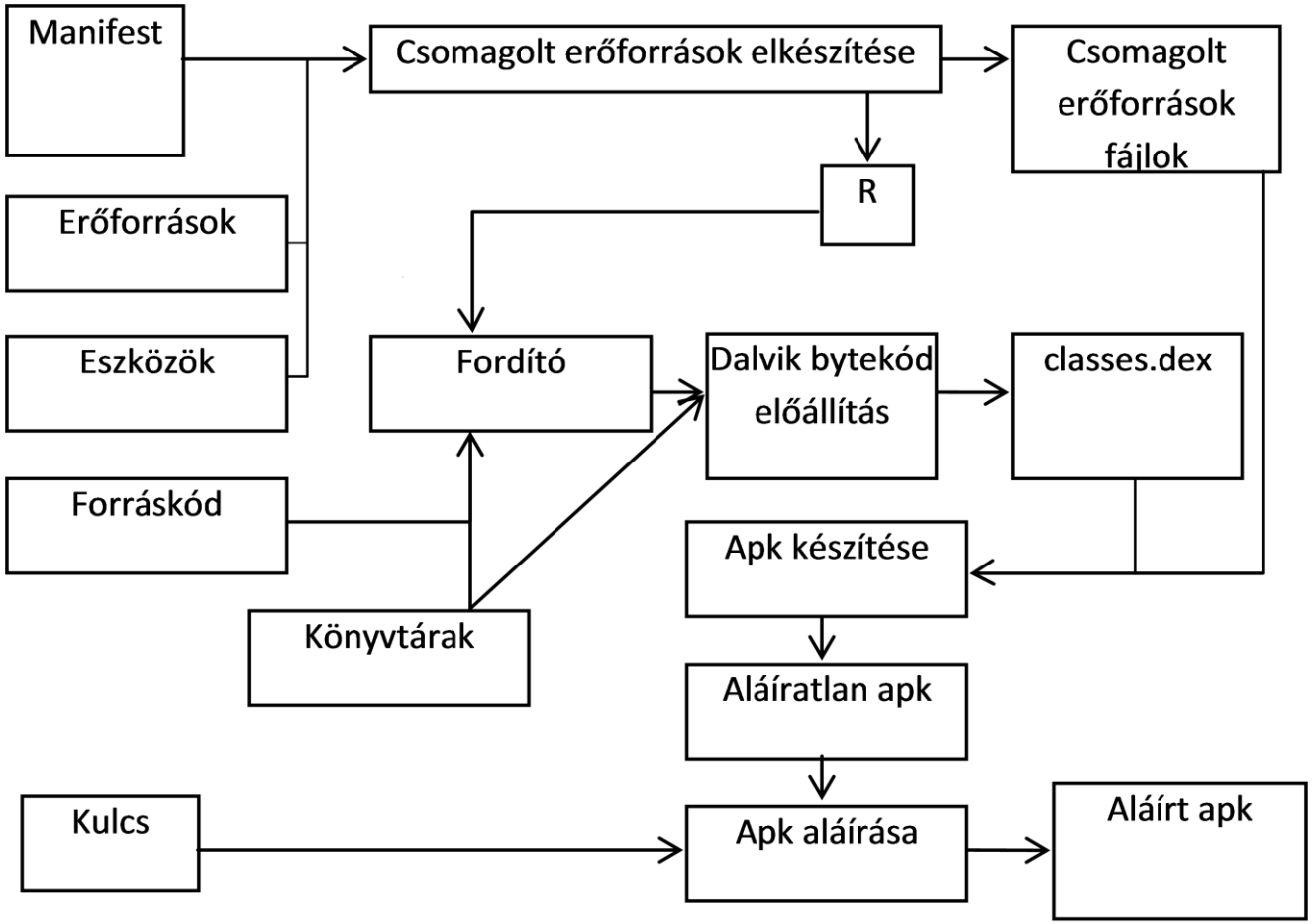
A platform felépítése 3/3

- A kék színnel jelölt részekben már csak Java forrást találunk
- Java forrást a virtuális gép futtatja, ez adja az Android lényegét:
  - Látható és tapintható operációs rendszert
  - Futó programokat
- A virtuális gép akár teljesen elrejti a Linux által használt fájlrendszert, és csak az Android Runtime által biztosított fájlrendszert láthatjuk

Szoftverfejlesztési eszközök Android platformra

- **Android SDK (Software Development Kit)**:
  - Fejlesztő eszközök
  - Emulátor kezelő (AVD Manager)
  - Frissítési lehetőség
  - Java
- **Android NDK (Native Development Kit)**:
  - Lehetővé teszi natív kód futtatását
  - C++
- **Android ADK (Accessory Development Kit)**:
  - Támogatás Android kiegészítő eszközök gyártásához (dokkoló, egészségügyi eszközök, időjárás kiegészítő eszközök stb.)
  - Android Open Accessory protocol (USB és Bluetooth)

A fordítás menete (forrás->.apk)



Az Android .apk állomány

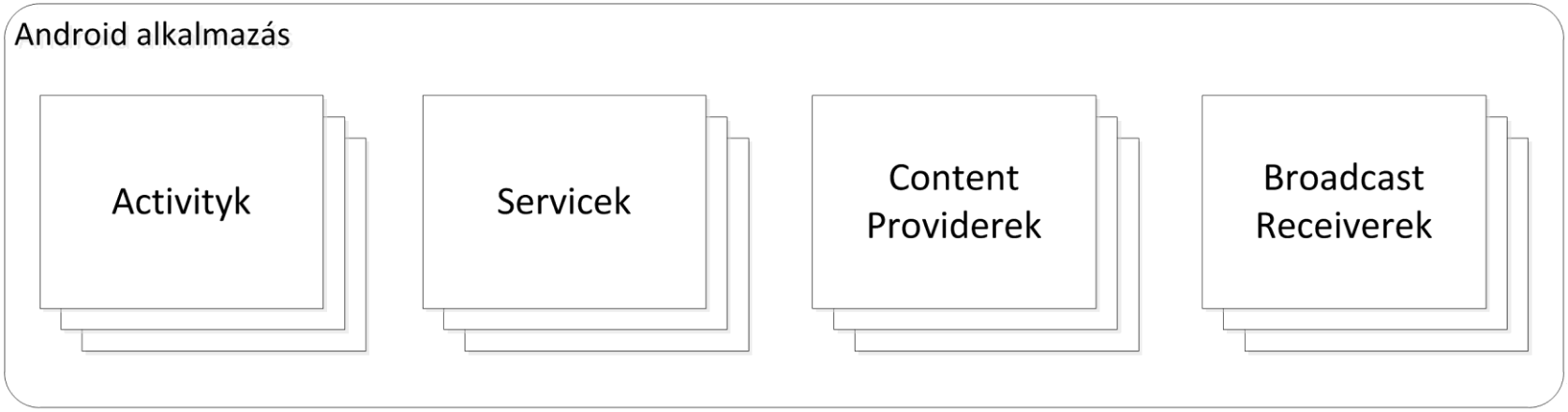
- Leginkább a Java világban megszokott .jar-hoz hasonlítható, de vannak jelentős eltérések
- Tömörített állomány, mely tipikusan a következő tartalommal rendelkezik:
  - META-INF könyvtár
    - CERT.RSA: alkalmazás tanúsítvány
    - MANIFEST.MF: meta információk kulcs érték párokban
    - CERT.SF: erőforrások listája és SHA-1 hash értékük, pl:

```
Signature-Version: 1.0
Created-By: 1.0 (Android)
SHA1-Digest-Manifest: wxqnEAI0UA5nO5QJ8CGMwjkGGWE=
...
Name: res/layout/exchange_component_back_bottom.xml
SHA1-Digest: eACjMjESj7Zkf0cBFTZ0nqWrt7w=
...
Name: res/drawable-hdpi/icon.png
SHA1-Digest: DGEqylP8W0n0iV/ZzBx3MW0WGCA=
```

  - Res könyvtár: erőforrásokat tartalmazza
  - AndroidManifest.xml: név, verzió, jogosultság, könyvtárak
  - classes.dex: lefordított osztályok a VM számára érthető formátumban
  - resources.arsc

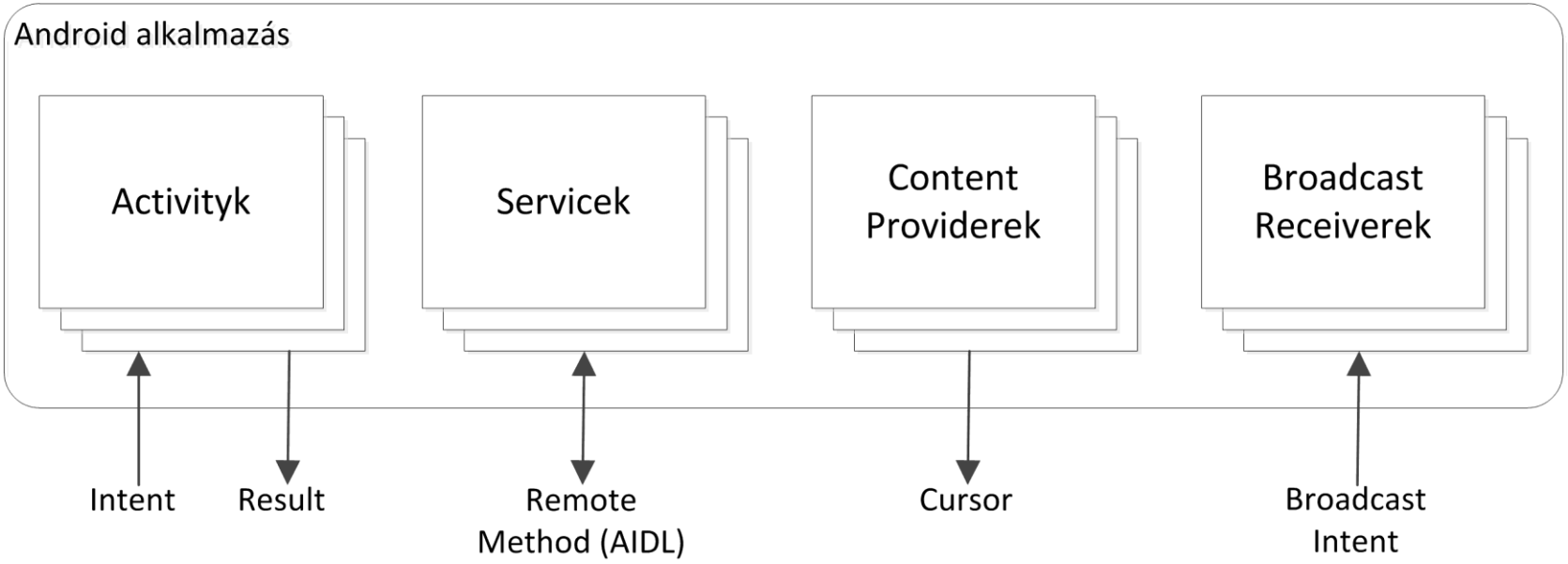
Android alkalmazás felépítése 1/3

- Egy Android alkalmazás egy vagy több alkalmazás komponensből épül fel:
  - Activity-k
  - Service-k
  - Content Provider-ek
  - Broadcast Receiver-ek



Android alkalmazás felépítése 2/3

- Minden komponensnek különböző szerepe van az alkalmazáson belül
- Bármelyik komponens önállóan aktiválódhat
- Akár egy másik alkalmazás is aktiválhatja az egyes komponenseket



Android alkalmazás felépítése 3/3

- Az alkalmazás leíró (*manifest*) állománynak deklarálnia kell a következőket:
  - Alkalmazás komponensek listája
  - Szükséges minimális Android verzió
  - Szükséges hardware konfiguráció
- A nem forráskód jellegű erőforrásoknak (képek, szövegek, nézetek, stb.) rendelkezésre kell állnia különböző nyelvű és képernyőméretű telefonokon

Activity-k

- Különálló nézet, saját UI-al
- Például:
  - Emlékeztető alkalmazás
  - 3 Activity: ToDo lista, új ToDo felvitele, ToDo részletek
- Független Activity-k, de együtt alkotják az alkalmazást
- Más alkalmazásból is indítható az Activity, például:
  - Kamera alkalmazás el tudja indítani az új ToDo felvitele Activity-t és a képet hozzá rendeli az emlékeztetőhöz
- Az **`android.app.Activity`** osztályból származik le

Service-k

- A Service komponens egy hosszabb ideig háttérben futó feladatot jelképez
- Nincs felhasználói felülete
- Például egy letöltő alkalmazás (torrent ☺) fut a háttérben, míg előtérben egy másik programmal játszunk
- Más komponens (pl. Activity) elindíthatja, vagy csatlakozhat (bind) hozzá vezérlés céljából
- Az `android.app.Service` osztályból kell öröklődnie

Content provider-ek

- A Content provider (tartalom szolgáltató) komponens feladata egy megosztott adatforrás kezelése
- Az adat tárolódhat fájlrendszerben, SQLite adatbázisban, web-en, vagy egyéb perzisztens adattárban, amihez az alkalmazás hozzáfér
- A Content provider-en keresztül más alkalmazások hozzáférhetnek az adatokhoz, vagy akár módosíthatják is azokat
- Például: CallLog alkalmazás, ami egy Content provider-t biztosít, és így elérhető a tartalom
- A `android.content.ContentProvider` osztályból származik le és kötelezően felül kell definiálni a szükséges API hívásokat

Broadcast receiver-ek

- A Broadcast receiver komponens a rendszer szintű eseményekre (broadcast) reagál
- Például: kikapcsolt a képernyő, alacsony az akkumulátor töltöttsége, elkészült egy fotó, bejövő hívás, stb.
- Alkalmazás is indíthat saját „broadcast”-ot, például ha jelezni akarja, hogy valamilyen művelettel végzett (letöltődött a torrent ☺)
- Nem rendelkeznek saját felülettel, inkább valamilyen figyelmeztetést írnak ki például a status bar-ra, vagy elindítanak egy másik komponenst (jeleznek például egy service-nek)
- A **`android.content.BroadcastReceiver`** osztályból származik le; az esemény egy Intent (lásd. Később) formájában érhető el

Mi nem igaz az alkalmazás komponensekkel kapcsolatban?

A. 4 Android alkalmazás komponens van.

B. Kötelező legalább egy Activity egy alkalmazáshoz.

C. Készíthetünk UI nélküli alkalmazásokat is.

D. A ContentProvider WebServeren tárolt adatokat is elérhetővé tud tenni.

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

Play Store: BME VOTE

Manifest állomány

- Alkalmazás leíró, definiálja az alkalmazás kompone
- XML állomány
- Komponens indítás előtt a rendszer a manifest állományt ellenőrzi, hogy definiálva van-e benne a kért komponens
- További feladatokat is ellát (pl. mik az alkalmazás futtatásának minimális követelményei)
- Alkalmazás telepítésekor ellenőrzi a rendszer

Manifest állomány tartalma

- Alkalmazást tartalmazó java package – **egyedi azonosítóként szolgál**
- Engedélyek, amelyekre az alkalmazásnak szüksége van (pl. internet elérés, névjegyzék elérés, stb.)
- Futtatáshoz szükséges minimum API szint
- Hardware és software funkciók, amit az alkalmazás használ (pl. kamera, bluetooth, stb.)
- Külső API könyvtárak (pl. Google Maps API)

Manifest példa 1/2

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android=
  "http://schemas.android.com/apk/res/android"
    package="hu.bute.daai.amorg.examples"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="7" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity …>…</activity>
    </application>
</manifest>
```

Egyedi package név (azonosító)

Legkisebb támogatott verzió

Alkalmazás ikon és cimke

Manifest példa 2/2

```
<?xml version="1.0" encoding="utf-8"?>
<manifest …/>
    …
    <application …>
        <activity
            android:name=".AndHelloWorldActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name=
                  "android.intent.action.MAIN"/>
                <category android:name=
                  "android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Activity osztály és cím

Alkalmazás belépési pont jelölő

Megjelenik a futtatható alkalmazások listájában (Launcher)

Mi igaz a Manifest állományra?

A. Csak az Activity komponenseket kell felsorolni benne.

B. Csak egy Service komponenst tartalmazhat.

C. Az összes alkalmazás komponenst fel kell sorolni benne kivéve a dinamikusan regisztrálható BR komponenseket.

D. XML és Java kód keveredhet benne.

Alkalmazás erőforrások

- Egy Android alkalmazás nem csak forráskódból áll, hanem erőforrásokból is, úgy mint: képek, hanganyagok, stb.
- Emellett erőforrások az XML-ben definiált felületek is: elrendezés, animáció, menü, stílus, szín.
- Erőforrások használatával sokkal rugalmasabban változtatható az alkalmazás
- Minden erőforráshoz a rendszer automatikusan egy egyedi azonosítót generál, amin keresztül elérhető a forráskódból

Erőforrás hivatkozás példa

- Tegyük fel, hogy készítettünk egy *logo.png*-t és elmentettük a *res/drawable/* könyvtárba
- Az SDK eszköz előállít egy egyedi erőforrást hozzá mentés után automatikusan
- Az azonosító: *R.drawable.logo*
- Ezzel az azonosítóval lehet hivatkozni bárhol az erőforrásra
- Az azonosítók az *R.java* állományban tárolódnak (soha ne módosítsuk ezt az állományt!)

Erőforrás használat előnyei

- Az egyik legnagyobb előny, hogy a készülék képességeihez lehet igazítani az erőforrásokat
- A könyvtárak után „minősítő”-ket írhatunk, amellyel megadjuk hogy mely tulajdonságok teljesülése esetén vegye a rendszer ebből a könyvtárból az erőforrásokat
- Többnyelvűség támogatása:
  - *Strings.xml*
  - *res/values/*
  - *res/values-fr/*
  - *res/values-hu/*

Activity bevezetés

- Egy Activity tehát tipikusan egy képernyő, amin a felhasználó valamilyen műveletet végezhet (login, beállítások, térkép nézet, stb.)
- Az Activity leginkább egy ablakként képzelhető el
- Az ablak vagy teljes képernyős, vagy pop-up jelleggel egy másik ablak fölött jelenik meg
- Egy alkalmazás tipikusan több Activity-ből áll, amik lazán csatoltak
- Legtöbb esetben létezik egy „fő” Activity, ahonnét a többi elérhető
- Bármelyik Activity indíthat újabbakat
- Tipikusan a „fő” Activity jelenik meg az alkalmazás indulása után elsőként

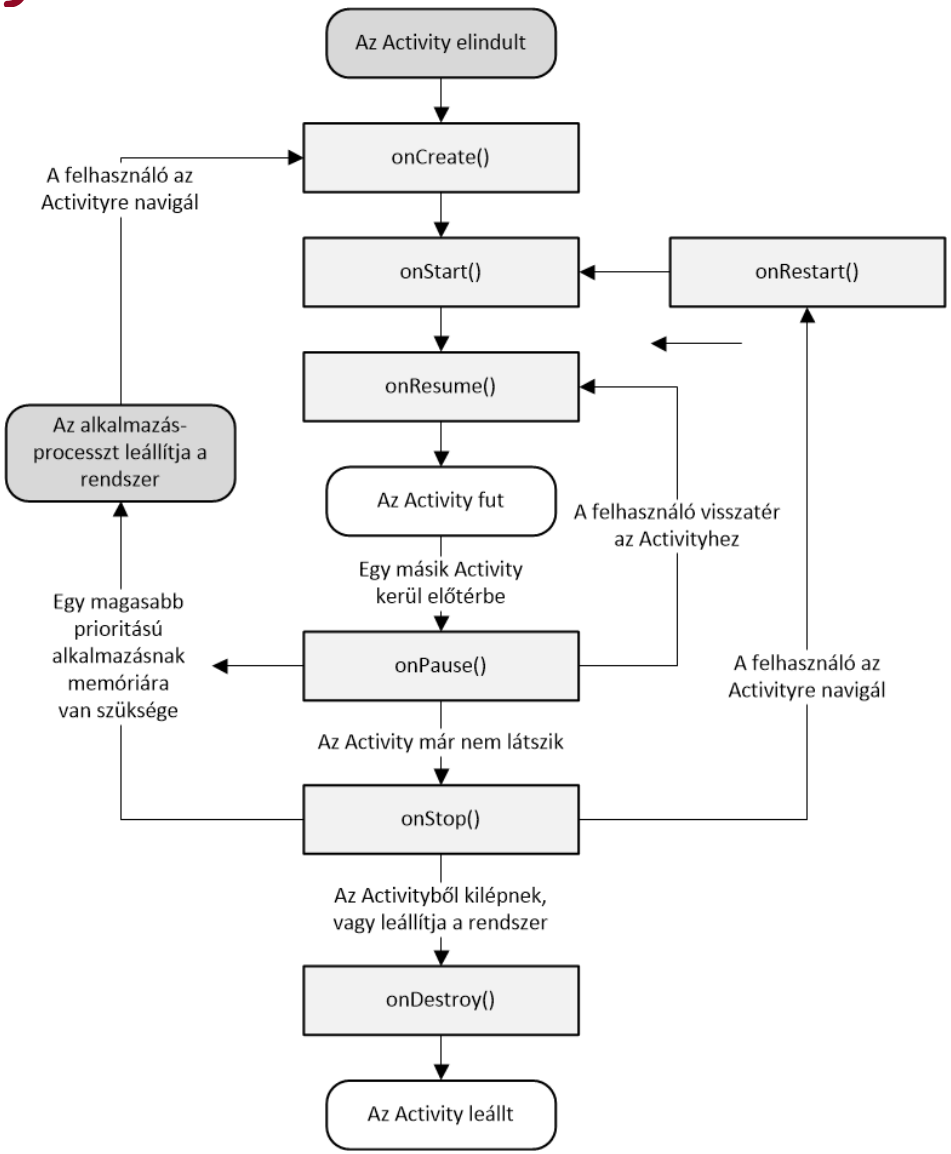
Activity életciklus-callback

- Amikor egy Activity leáll egy másik indulása miatt, az Activity az eseményről értesítést kap az úgynevezett életciklus-callback metódusokon keresztül
- Számos callback metódus támogatott (create, stop, resume, destroy, stb.), amikre megfelelően reagálhat az Activity
- Például stop esemény hatására tipikusan a nagyobb objektumokat érdemes elengedni (DB/hálózati kapcsolat)
- Amikor az Activity visszatér (resume), újra kell kérni az erőforrásokat
- Ezek az átmenetek tipikus részei az Activity életciklusának

Activity életciklus

- Egy megbízható és flexibilis alkalmazás esetén kritikus fontosságú az Activity életciklus-callback függvények megfelelő felüldefiniálása
- Az Activity életciklusát a vele együttműködő többi Activity határozza meg
- Elengedhetetlen az Activity működésének tesztelése a különböző életciklus állapotokban

Activity életciklus modell



Activity bezárása a rendszer által

- Paused, vagy Stopped állapotban a rendszer bármikor leállíthatja memória-felszabadítás céljából
- A leállítás történhet a *finish()* hívással, vagy kritikusabb esetben a Process leállításával
- Ha az Activity-t újra megnyitják (miután be lett zárva), a rendszer újra létrehozza

Életciklus callback függvények

- Amikor az Activity állapotot vált, megfelelő callback függvények hívódnak meg
- Ezek a callback függvények „hook” jellegű függvények, melyeket a rendszer hív
- Fontos a metódusok felül definiálása és a megfelelő részek implementálása
  - Mindig meg kell hívni az ős osztály implementációját is (pl. `super.onCreate();`)!
- A rendszer felelőssége meghívni ezeket a függvényeket, de a fejlesztő felelőssége a helyes implementáció

Activity skeleton 1/2

```
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Most jön létre az Activity
    }
    @Override
    protected void onStart() {
        super.onStart();
        // Most válik láthatóvá az Activity
    }
    @Override
    protected void onResume() {
        super.onResume();
        // Láthatóvá vált az Activity
    }
```

Activity skeleton 2/2

```
    @Override
    protected void onPause() {
        super.onPause();
        // Másik Activity veszi át a focus-t
        // (ez az Activity most kerül „Paused” állapotba)
    }
    @Override
    protected void onStop() {
        super.onStop();
        // Az Activity már nem látható
        // (most már „Stopped” állapotban van)
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // Az Activity meg fog semmisülni
    }
}
```

Activity életciklus callback függvények 1/2

- *onCreate()*: Activity létrejön és beállítja a megfelelő állapotokat (layout, munka szálak létrehozása, stb.)
- *onDestroy()*: Minden még lefoglalt erőforrás felszabadítása
- *onStart()*: Az Activity látható, a vezérlők is. Például BroadcastReceiverek-re feliratkozás, amik módosítják a UI-t
- *onStop()*: Az Activity nem látható. Például BroadcastReceiverek-ről leiratkozás
  - Az Activity élete során többször válthat látható és nem látható állapotok között.

Activity életciklus callback függvények 2/2

- *onRestart()*: Az Activity leállítása (onStop()) majd újraindítása után hívódik meg, még az indítás (onStart()) előtt
- *onResume()*: Az Activity láthatóvá válik és előtérben van, a felhasználó eléri a vezérlőket és tudja kezelni azokat
- *onPause()*: Az Activity háttérbe kerül, de valamennyire látszik a háttérben, például egy másik Activity pop-up jelleggel előjön, vagy sleep állapotba kerül a készülék

Mi igaz az Activity életciklus függvényekre?

A. Kötelező minden életciklus függvényt felüldefiniálni, különben nem fordul az alkalmazás kódja.

B. Kötelező az ősosztály implementációjának meghívása.

C. Az Activity élete során minden függvény csak egyszer hívódhat meg.

D. Szükség esetén manuálisan is meg kell hívni.

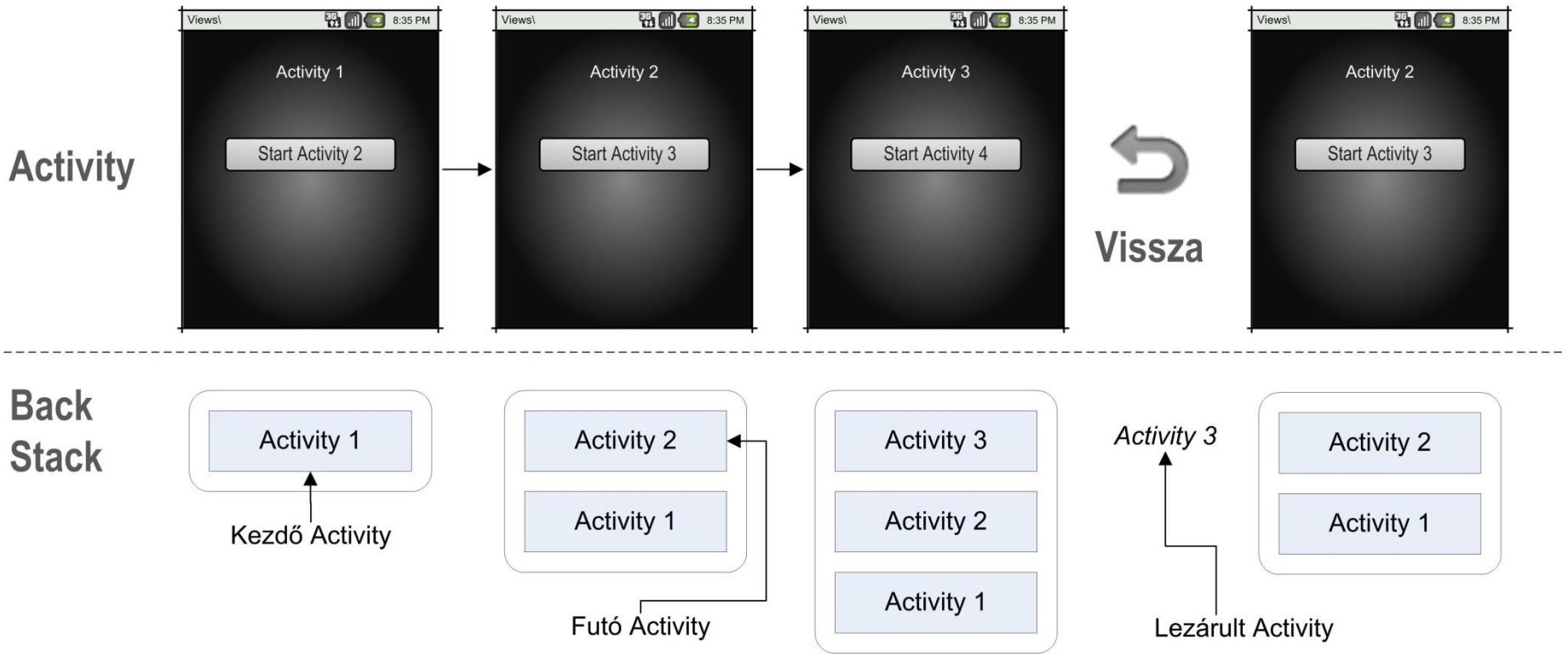
Activity váltás

- Életciklus callback függvények meghívási sorrendje:
  - A Activity *onPause()* függvénye
  - B Activity *onCreate()*, *onStart()* és *onResume()* függvénye (B Activity-n van már a focus)
  - A Activity *onStop()* függvénye, mivel már nem látható
- Ha a B Activity valamit adatbázisból olvas ki, amit az A ment el, akkor ez a mentés A-nak az *onPause()* függvényében kell megtörténjen, hogy a B aktuális legyen, mire a felhasználó előtt megjelenik

Activity Back Stack 1/2

- Egy feladat végrehajtásához a felhasználó tipikusan több Activity-t használ
- A rendszer az Activity-ket egy ún. Back Stack-en tárolja
- Az előtérben levő Activity van a Back Stack tetején
- Ha a felhasználó átvált egy másik Activity-re, akkor eggyel lejjebb kerül a Stack-ben és a következő lesz legfelül
- Vissza gomb esetén legfelülről veszi ki a rendszer az megjelenítendő Activity-t
- Last in, first out

Activity Back Stack 2/2



Activity vezérlés 1/2

- Legtöbb esetben az alapértelmezett Back Stack viselkedés kielégíti az igényeket
- Néha azonban szükség lehet ezen alapértelmezett viselkedés felül definiálására
- Back Stack törlése, ha a Vissza hatására mindig egy kezdő Activity-re kell visszalépni
- Az alapértelmezett viselkedés felülírása:
  - Manifest állományban az <activity>-be
  - *startActivity(...)* fv. Paramétereként
- Amennyiben az alapértelmezett viselkedést módosítjuk, mindenképp teszteljük az alkalmazást navigálás és felhasználói élmény szempontjából, mert sokszor a programozó szempontjából jó megoldás nem ideális felhasználói szempontból

Activity vezérlés 2/2

- Az <activity> tag attribútumai (új Activity hogyan viselkedjen a többihez képest):
  - taskAffinity: melyik taskhoz tartozik
  - launchMode: indítási mód (mindig új példány, stb.)
  - allowTaskReparenting: új taskhoz kerül át
  - clearTaskOnLaunch: minden Activity-t töröl a task-ból
  - alwaysRetainTaskState: a rendszer kezelje-e a task állapotát
  - finishOnTaskLaunch: le kell-e állítani az Activity-t ha a felhasználó kilép a task-ból (pl. HOME gomb)
- *startActivity(...)* függvény paraméter értékei (az új Activity hogyan viselkedjen a most futóhoz képest):
  - FLAG_ACTIVITY_NEW_TASK
  - FLAG_ACTIVITY_CLEAR_TOP
  - FLAG_ACTIVITY_SINGLE_TOP
- További részletek az Intent előadásban

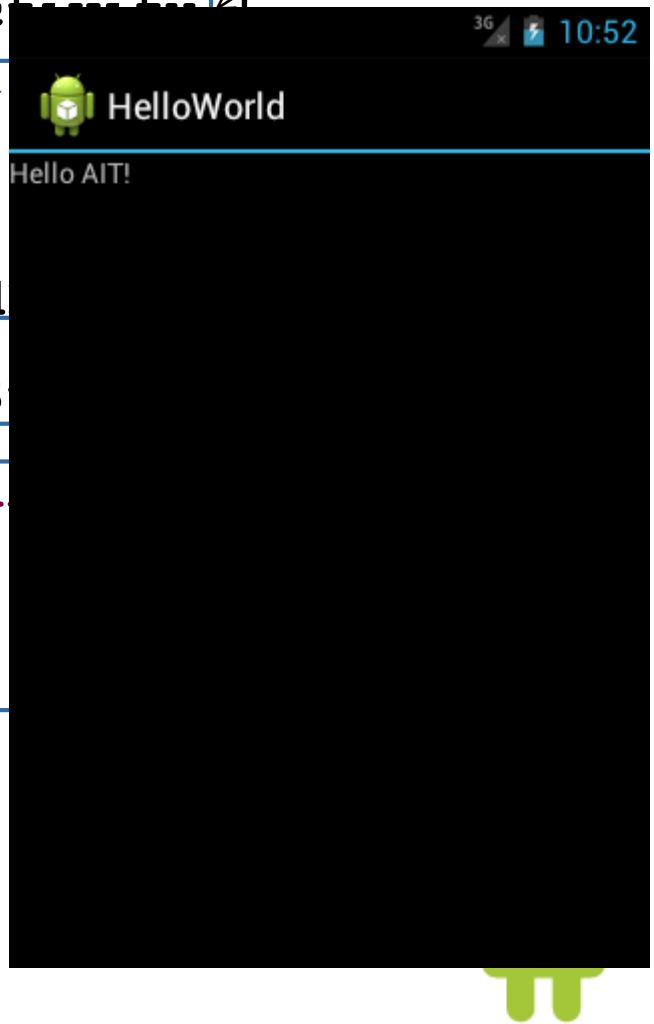
Az első Android alkalmazás

Ősosztály

Ősosztály implementáció meghívása

TextView megjelenítése

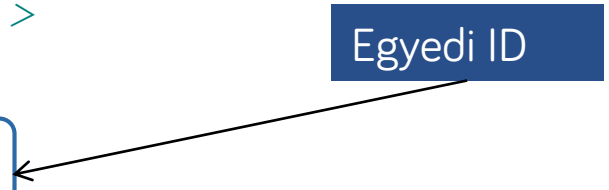
```
public class HelloAndroid extends Ac
    lled when the activity is f
    ide
    public void onCreate(Bundle saved
        super.onCreate(savedInstanceS
        TextView tv = new TextView(th
        tv.setText("Hello BME!");
        setContentView(tv);
    }
```



Android HelloWorld XML alapú UI-al 1/2

Hello Android XML (*layout/main.xml*):

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=
  "http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/tvHello"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
</LinearLayout>
```



Android HelloWorld XML alapú UI-al 2/2

```
package hu.bute.daai.amorg.examples;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloWorldActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView myTextView = (TextView)findViewById(R.id.tvHello);
        myTextView.append("\n--MODIFIED--");
    }
}
```

XML alapú layout

UI komponens kikeresése ID alapján

Egyszerű esemény kezelés

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    final TextView myTextView =
       (TextView)findViewById(R.id.tvHello);
    myTextView.append("\n--MODIFIED--");
    myTextView.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            myTextView.append("\n--CLICKED--");
        }
    });
}
```

Mivel anonim osztályból férünk hozzá

Egyszerű érintés esemény kezelés

Új Activity indítása

- SecondActivity indítása:

```
public void runSecondActivity() {
    Intent myIntent = new Intent();
    myIntent.setClass(MainActivity.this,
        SecondActivity.class);
    // Adat átadása
    myIntent.putExtra("MyValue", "Hi there!");
    startActivity(myIntent);
}
```

Felhasználói felület alapok

Android felhasználói felület felépítése

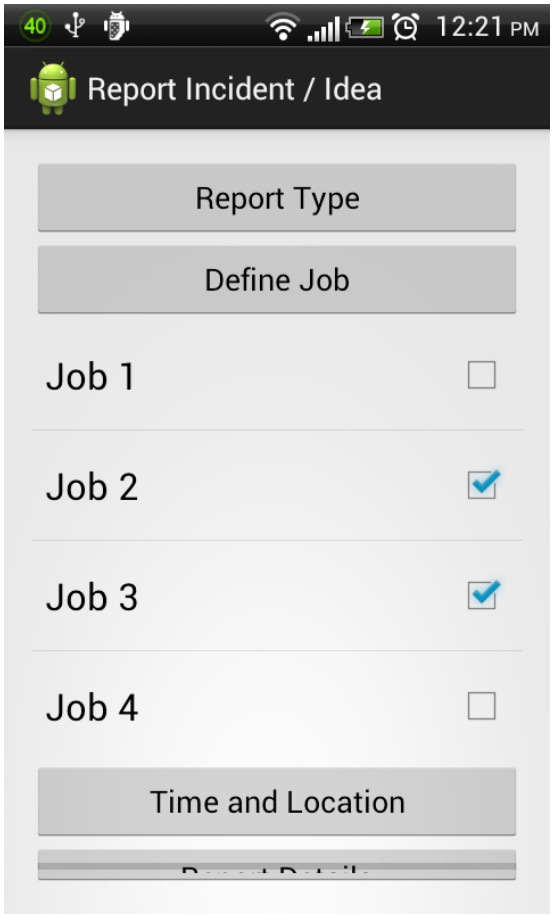
- Minden elem a View-ból származik le
- Layout-ok (elrendezések):
  - ViewGroup leszármazottak
  - ViewGroup is a View-ból származik le!
- ViewGroup-ok egymásba ágyazhatók
- Saját View és ViewGroup is készíthető, illetve a meglevők is kiterjeszthetők

Layout-ok (ViewGroup)

- LinearLayout
- RelativeLayout (~iOS AutoLayout, de magasabb szintű)
- AbsoluteLayout (NEM használjuk!)
- GridLayout
- RecyclerView
- …
- Teljes lista:
  - http://developer.android.com/reference/android/view/ViewGroup.html

LinearLayout

- LinearLayout != Lista

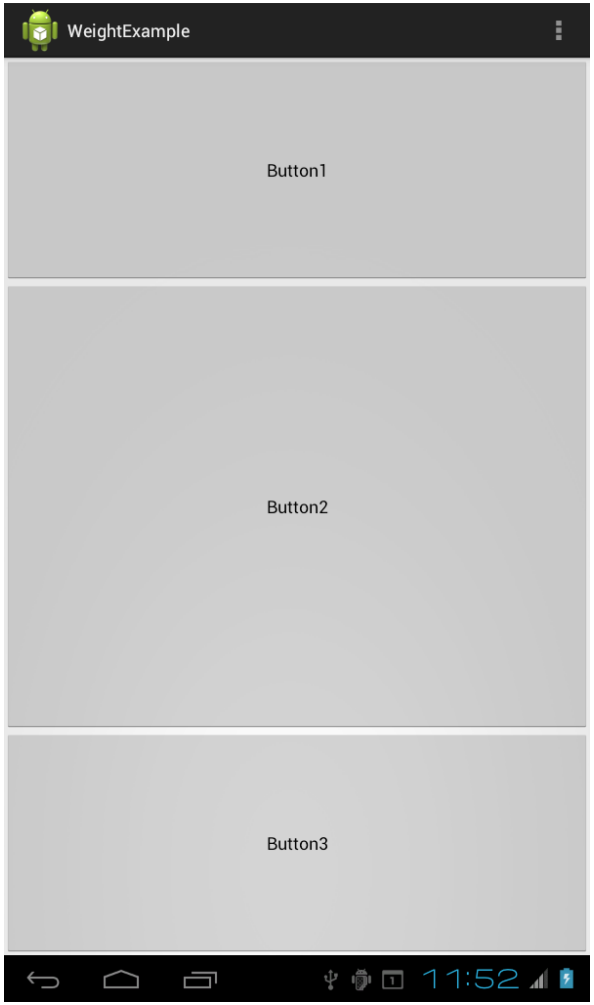


Súlyozás Layout tervezéskor

- 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

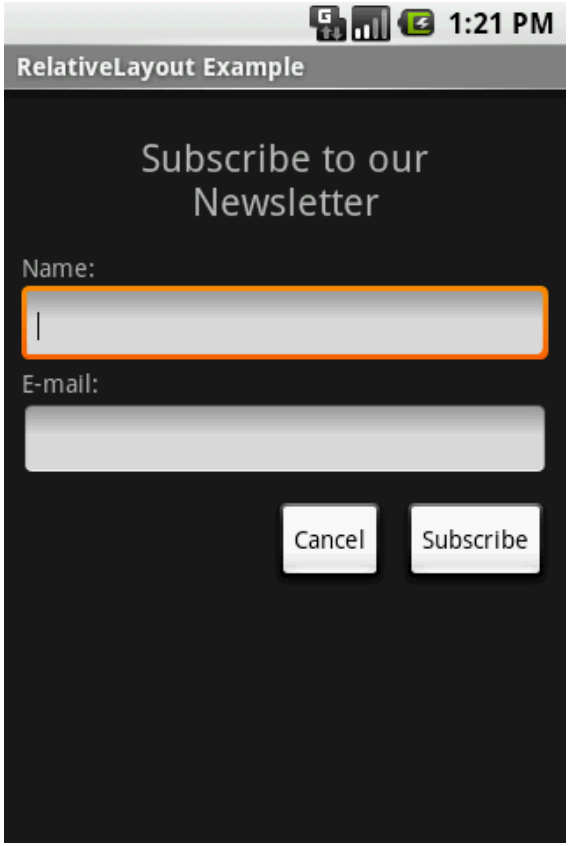
Layout súlyozás példa

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

</LinearLayout>
```



RelativeLayout

- Elemek egymáshoz való viszonya definiálható
- Demo

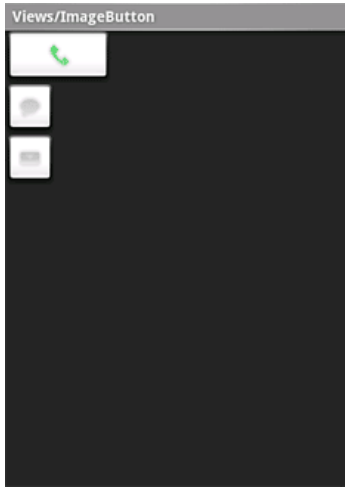


View-k 1/2

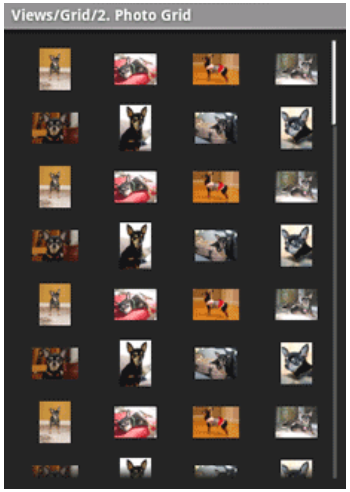
- *Button, EditText, CheckBox, RadioButton, ToggleButton*
- *ImageButton*
- *ListView*
- *GridView*
- *Spinner*
- *AutoCompleteTextView*
- *Gallery*
- *ImageSwitcher*
- *DatePicker, TimePicker*

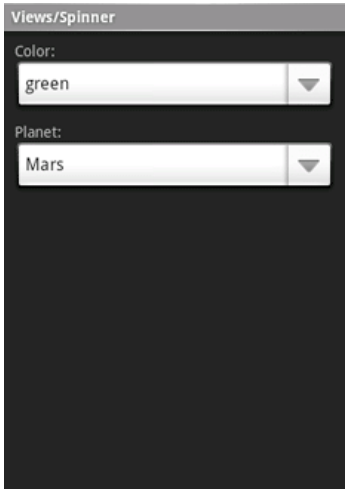
View-k 2/2

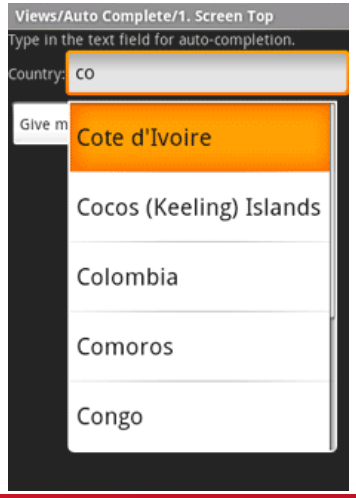

Views/Controls/1. Light Theme
Save
Checkbox 1
Checkbox 2
RadioButton 1
RadioButton 2
Star
OFF
OFF

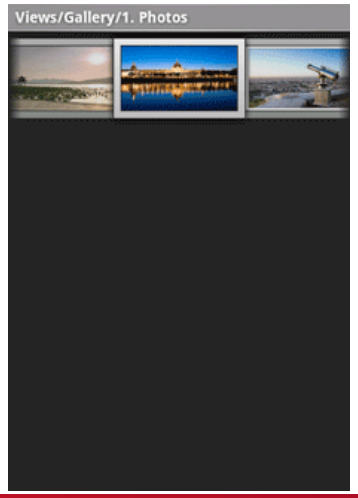

Views/ImageButton

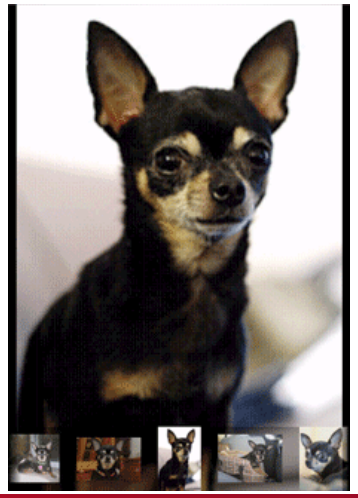

Views/Lists/1. Array
Abbaye de Belloc
Abbaye du Mont des Cats
Abertam
Abondance
Ackawi
Acorn
Adelost


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

Hogy is volt?

- Magyarázza el a fordítás mechanizmusát!
- 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!

Összefoglalás

- Android alkalmazás komponensei
- Manifest állomány, jogosultságok
- Activity Back Stack
- Multitasking
- Android HelloWorld
- Egyszerű eseménykezelés
- Navigálás Activity-k között
- Felhasználói felülete alapok