# Android platform, fejlesztőkörnyezet és Kotlin nyelv bemutatása

Ekler Péter

BME VIK AUT, AutSoft

peter.ekler@aut.bme.hu


# Néhány szó a tanfolyamról

# Kik vagyunk?

- AutSoft Kft.
- 2011-ben alapítva a Budapest Műszaki Egyetem Automatizálási és Alkalmazott Informatika Tanszékén
- Szoros együttműködés az egyetemmel
- Fő tevékenységek
  - szoftver fejlesztés
    - egyedi, termék, UI/UX
  - oktatás
  - konzultáció, tervezés

# Oktatók

Braun Márton

braun.marton@autsoft.hu

Balogh Tamás

balogh.tamas@autsoft.hu

Ekler Péter

ekler.peter@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

# Jól választottam?

- Miért Android?
  - Legnépszerűbb mobil platform
  - Minden sikeres mobil megoldás elérhető Androidon
- Miért Kotlin?
  - Meglévő Java elvekre épít
  - Közös Java byte kód
  - Modern nyelv, több mint 6 éves múlttal
  - Hivatalos Google támogatás
  - Megtanult dolgok hosszú távon érvényesek **és nem Kotlin specifikusak**
- Igen

# Tanfolyam jellege

- Stabil elméleti alapok, de gyakorlat orientált
- Interaktív (chat/Slack)
- Gyakori demok
- Kód megosztás (live és GitHUB)
  - https://github.com/AutSoft/AndroidKotlin
- Élő szavazás
  - http://babcomaut.aut.bme.hu/votes/
  - https://kahoot.it/#/

# Mire utal a Kotlin név?

A. Semmire, csak egy kitalált szó

B. Egy lengyel falu nevére

C. Egy sziget nevére

D. Key Object Tool Language INsight rövidítése

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

# Android bevezetés, Kotlin alapok

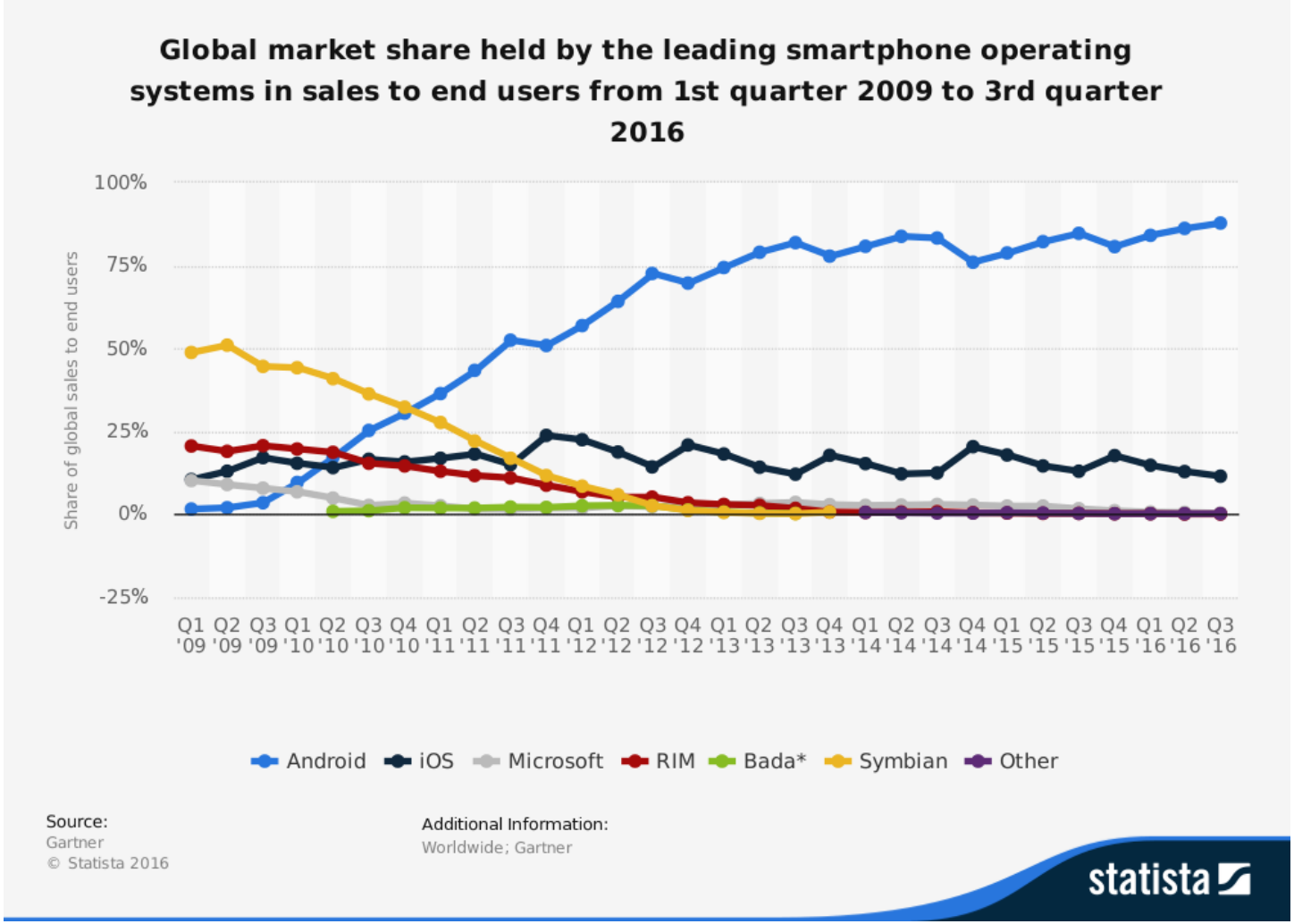
# Tartalom

- Android platform felépítése, Android verziók
- Android Studio bemutatása, Android SDK
- Fejlesztőkörnyezet fő és rejtett funkciói
- Projekt felépítése
- Hibakeresési eszközök és lehetőségek
- Emulátor és emulátor alternatívák
- Fejlesztés fizikai készüléken
- Kotlin nyelv alapozó
- Kotlin és Java kapcsolata
- Egyszerű játék alkalmazás fejlesztése Kotlin nyelven

# Bevezetés

- Okostelefonok térhódítása
- Táblagépek terjedése
- Háttértár növekedése
- Hálózat sebességének növekedése
- Adatforgalom árának csökkenése
- Alkalmazásboltok megjelenése

# Piaci rész - Android



# Android eszközök

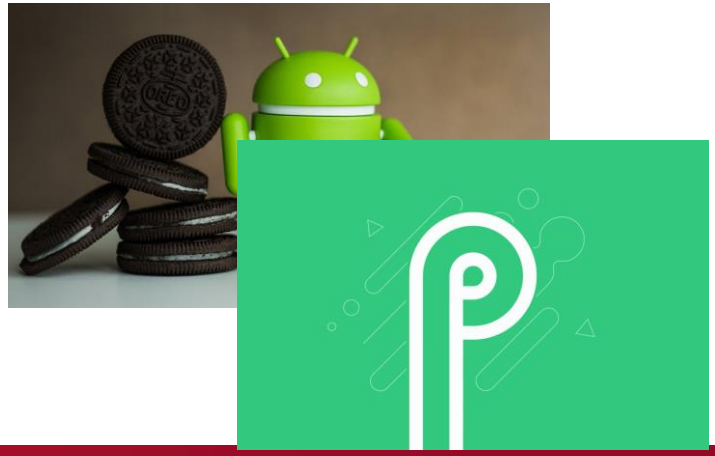
- Mobiltelefon és a Tablet gyártók
- Gépjárművek fedélzeti számítógépét és navigációját szállító cégek
- Android Wear
- Ipari automatizálás irányából is
- Minden olyan helyen kényelmes az Andro
  - Alapvetően kicsi a kijelző (Google TV megc
  - Más jellegű erőforrások
  - Az adatbevitel nem tipikusan egérrel és/vagy billentyűzettel történik
  - Android@Home


# Android verziók

- Fontos a verziók nyomon követése
- Egyes verziók között komoly API-beli különbségek lehetnek
- Törekednek a visszafele kompatibilitásra, de lehetnek éles szakadékok (pl. 3.0)
- Fejlesztés előtt alaposan gondoljuk át a támogatott minimum verziót
- Verzió kódnév: valamilyen édesség ☺

# Android verziószámok

- Android 1.0 – 2008. October
- Android 1.1 – 2009. February
- Android 1.5 (Cupcake) – 2009. April
- Android 1.6 (Donut) – 2009. September
- Android 2.0 and 2.1 (Eclair) – 2009. October
- Android 2.2 (Froyo) – 2010. May
- Android 2.3 (Gingerbread) – 2010. December
- Android 3.0-3.2 (Honeycomb) – 2011 January-July
- Android 4.0 (Ice Cream Sandwich) – 2011. October
- Android 4.1 (Jelly Bean) – 2012. July
- Android 4.2 (Jelly Bean) – 2012. November
- Android 4.3 (Jelly Bean)
- Android 4.4 (KitKat)
- Android 5.0, 5. (Lollipop)
- Android 6.0 (Marshmallow)
- Android 7.0, 7.1 (Nougat)
- Android 8.0, 8.1 (Oreo)
- Android 9.0 (P)



# Az Android jövője

Pletykák: Fuchsia

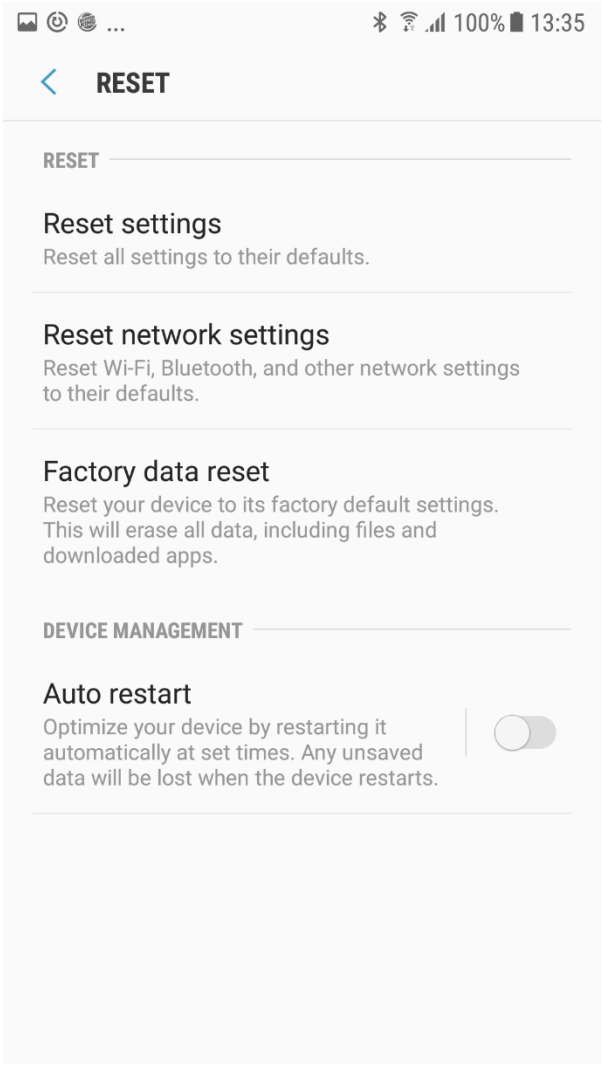
A B C D E F G H I J

K L M N O P Q R S

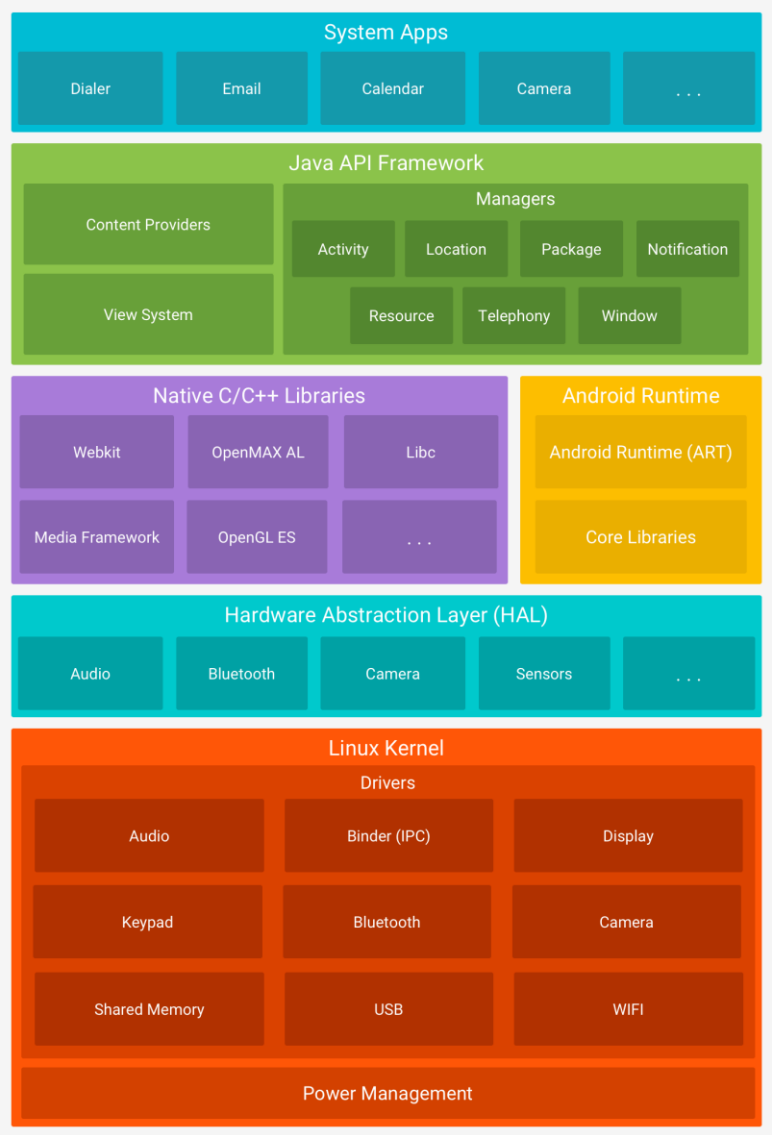
T U V W X Y Z

Még van ~9 évünk ☺

# Érdekesség ☺


100% 13:35
RESET
RESET
Reset settings
Reset all settings to their defaults.
Reset network settings
Reset Wi-Fi, Bluetooth, and other network settings to their defaults.
Factory data reset
Reset your device to its factory default settings. This will erase all data, including files and downloaded apps.
DEVICE MANAGEMENT
Auto restart
Optimize your device by restarting it automatically at set times. Any unsaved data will be lost when the device restarts.

# Az Android platform szerkezete



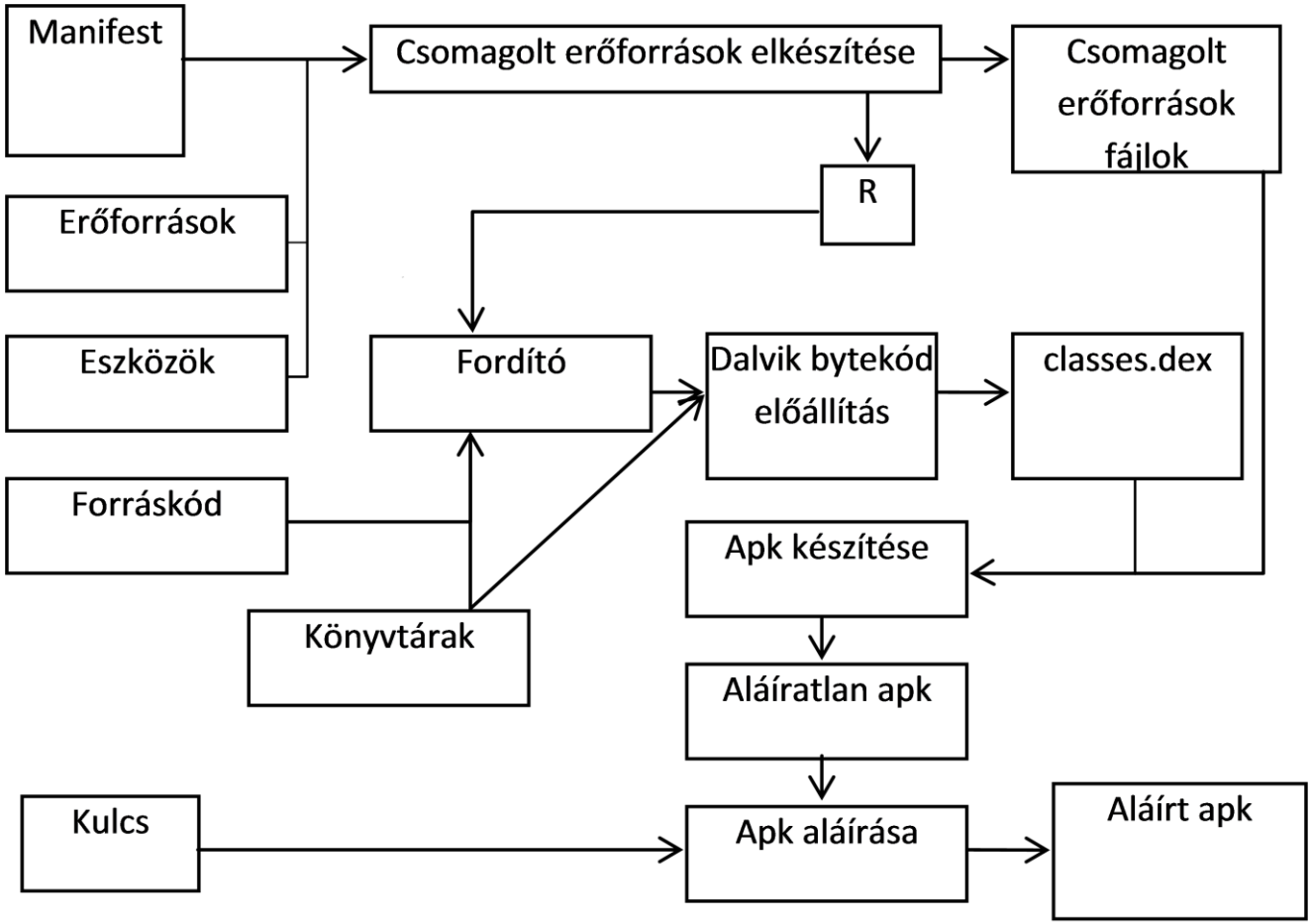
# Szoftverfejlesztési eszközök Android platformra

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

# SDK komponensek

- SDK minden Android verzióra
- Dokumentáció
- Példakódok
- USB Driverek (ADB)
- Third party kiegészítők
  - Google APIs (Térkép)
  - Galaxy Tab API
  - Stb.
- Konzolos felhasználás is támogatott, pl projekt létrehozás:
  - android create project --target android-16 --name MyFirstApp --path D:\tmp\MyFirstApp --activity MainActivity --package com.example.myfirstapp

# 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 Dalvik számára érthető formátumban
  - resources.arsc

# Emulátor

- Teljes operációs rendszer emulálása (lassú)
  - Beépített alkalmazások elérhetők
  - Ctrl+F11 (screen orientáció állítás)
- Alternatíva: Genymotion emulátor (https://www.genymotion.com/)

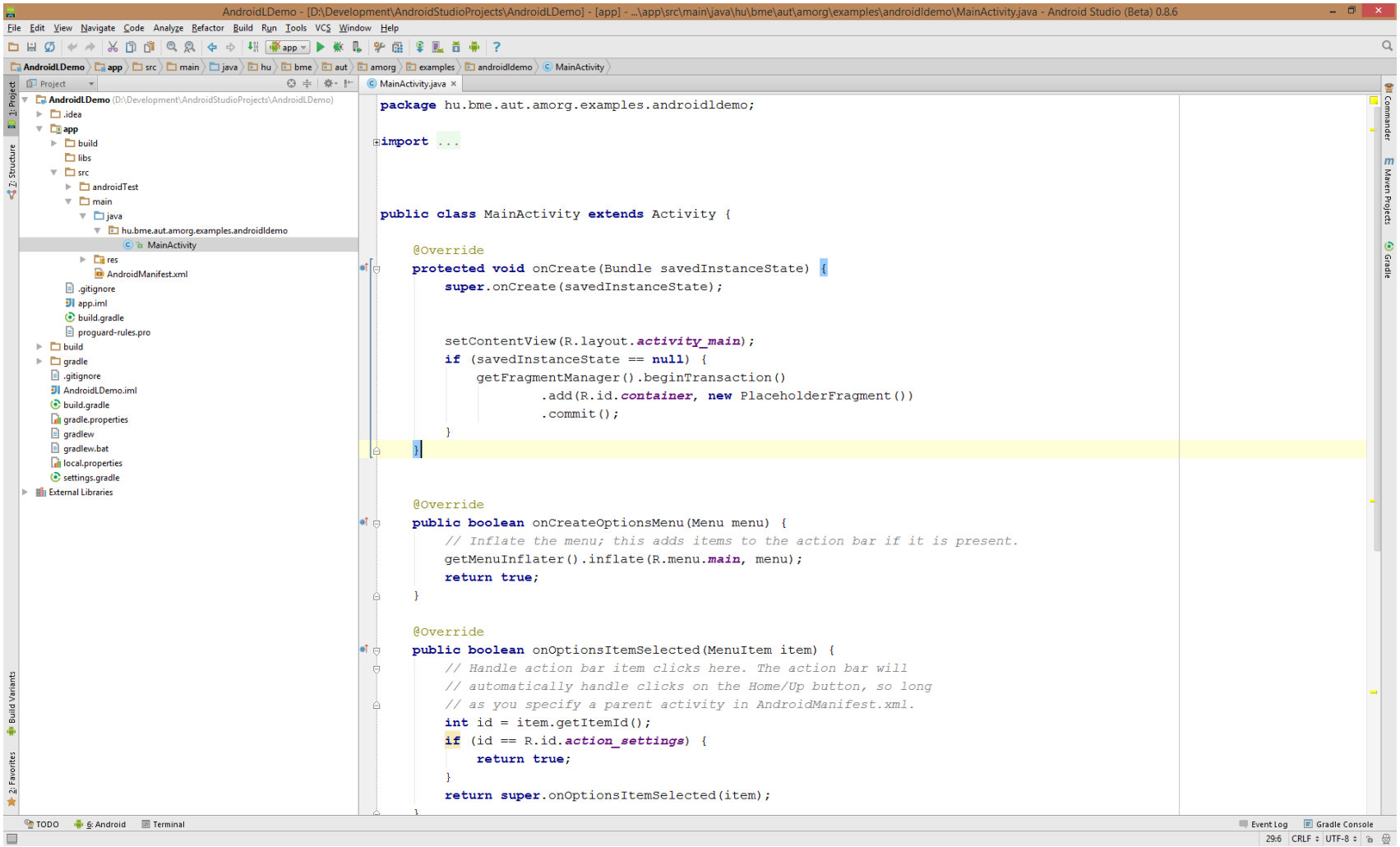

# Emulátor elérése konzolról

- Csatlakoztatott emulátorok/eszközök listázása:
  - adb devices
- Shell elérése
  - adb shell
- Csatlakozás telneten keresztül:
  - Indítsunk telnet klienst
  - o localhost 5554
- SMS küldése:
  - sms send <küldő száma> <üzenet>
- Hanghívás
  - gsm call <hívó száma>

# Debugolás folyamata

- On-device debug teljes mértékben támogatott
  - Megfelelő USB driver szükséges!
  - Készüléken engedélyezni kell az USB debugolást
- Minden alkalmazás önálló process-ként fut
- Minden ilyen process saját virtuális gépet (VM) futtat
- Minden VM egy egyedi portot nyit meg, melyre a debugger rácsatlakozhat (8600, 8601, stb.)
- Létezik egy úgynevezett „base port” is (8700), mely minden VM portot figyel és erre csatlakozva az összes VM-et debugolhatjuk

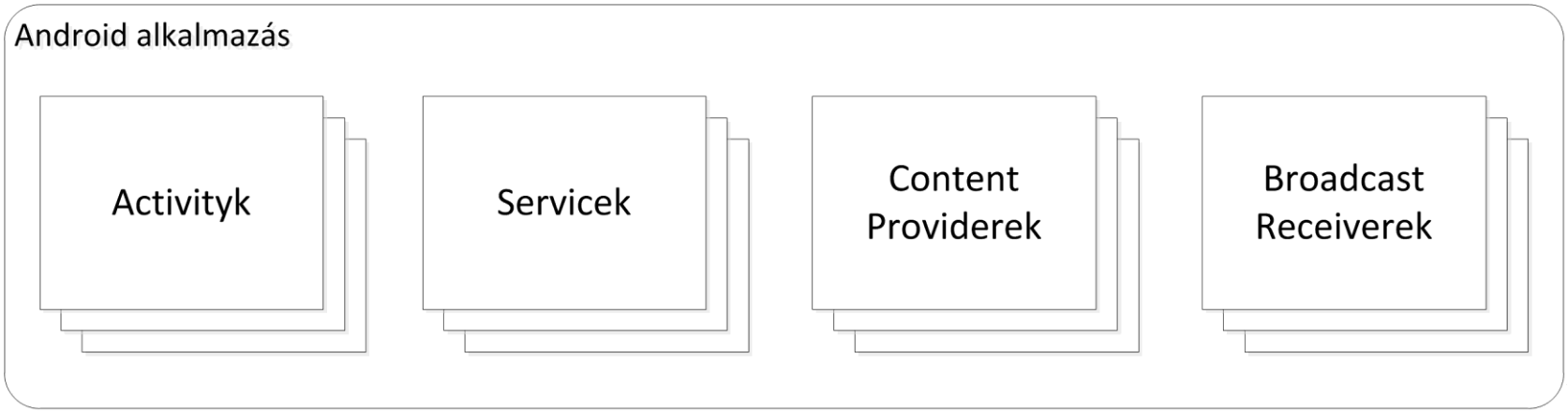
# Hello Android Studio




További részletek a laborokon☺

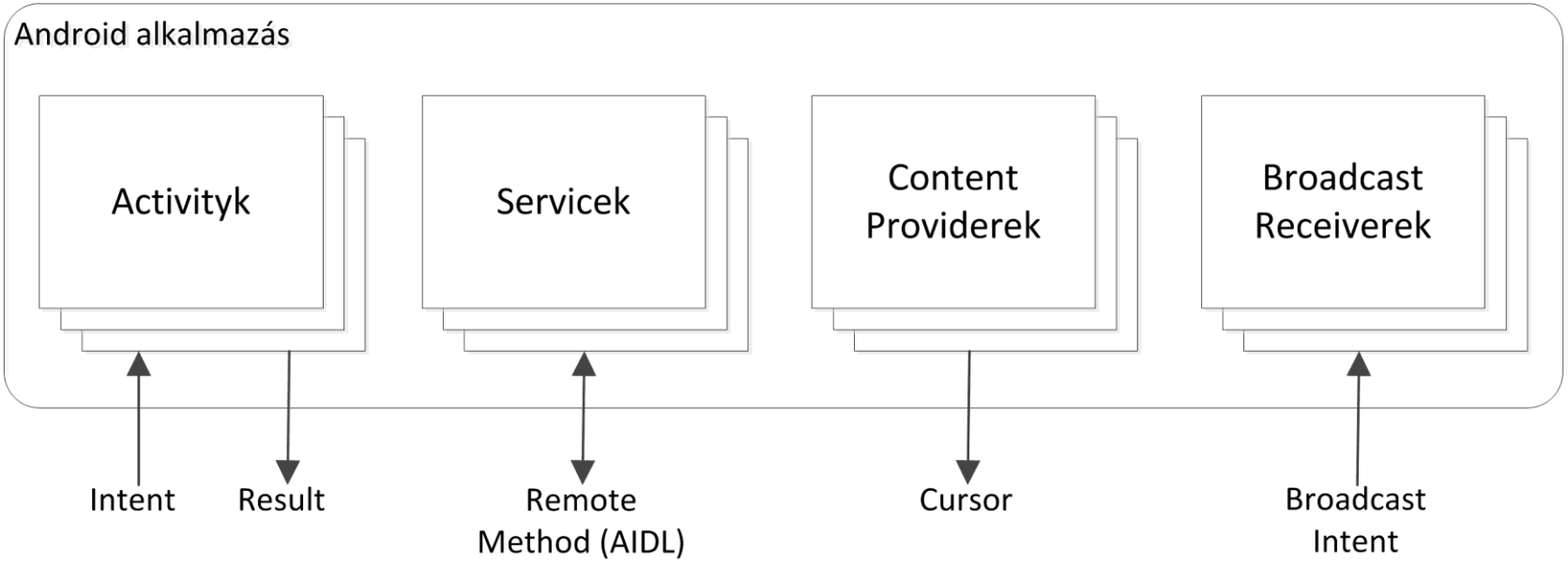
# Android alkalmazás felépítése 1/2

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



# Android alkalmazás felépítése 2/2

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



# Activity-k

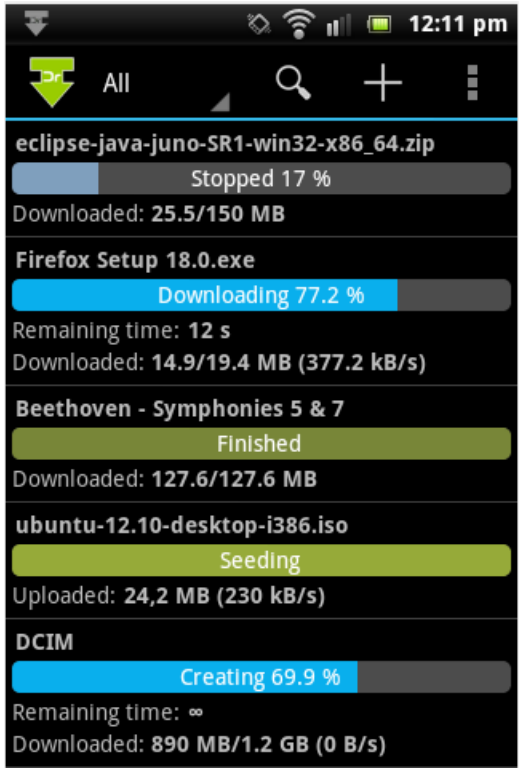
- Különálló nézet, saját UI-al
- Például:
  - Emlékeztető alkalmazás
  - 3 Activity: ToDo lista, új ToDo felvitele, ToDo részletek
- Független Activity-k, de együtt alkotják az alkalmazást
- Más alkalmazásból is indítható az Activity, például:
  - Kamera alkalmazás el tudja indítani az új ToDo felvitele 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

# DrTorrent

- BitTorrent kliens Android platformra
- Megszokott funkciók és háttérben működés

# 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

# Android alkalmazás struktúrája

- 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

## 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/

# 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.aut.android.kotlindemo">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Package

Alkalmazás név és ikon

# Manifest példa 2/2

```
<?xml version="1.0" encoding="utf-8"?>
<manifest …/>
    …
    <application …>
        <activity …
            android:name=".MainActivity"
            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)

# Manifest attribútumok és tag-ek

- `android:icon`: alkalmazás ikonja
- `android:name`: Activity teljes neve package-el együtt
- `android:label`: A készülék felületén, a felhasználók által látható név
- Komponensek:
  - `<activity>`: Activity
  - `<service>`: Service
  - `<provider>`: Content provider
  - `<receiver>`: Broadcast receiver
- A manifest-ben nem szereplő Activity-k, Service-k és Content provider-ek nem láthatók a rendszer számára
- Broadcast receiver-ek viszont dinamikusan is ki/be-regisztrálhatnak (kódból – `registerReceiver()`)

# Application beállítások

```
<application android:allowTaskReparenting=["true" | "false"]
        android:allowBackup=["true" | "false"]
        android:backupAgent="string"
        android:debuggable=["true" | "false"]
        android:description="string resource"
        android:enabled=["true" | "false"]
        android:hasCode=["true" | "false"]
        android:hardwareAccelerated=["true" | "false"]
        android:icon="drawable resource"
        android:killAfterRestore=["true" | "false"]
        android:largeHeap=["true" | "false"]
        android:label="string resource"
        android:logo="drawable resource"
        android:manageSpaceActivity="string"
        android:name="string"
        android:permission="string"
        android:persistent=["true" | "false"]
        android:process="string"
        android:restoreAnyVersion=["true" | "false"]
        android:requiredAccountType="string"
        android:restrictedAccountType="string"
        android:supportsRtl=["true" | "false"]
        android:taskAffinity="string"
        android:testOnly=["true" | "false"]
        android:theme="resource or theme"
        android:uiOptions=["none" | "splitActionBarWhenNarrow"]
        android:vmSafeMode=["true" | "false"] >
    . . .
</application>
```

# Activity futtatása másik process-ben

```
<activity android:name=".Activity2"
android:process="hu.bme.aut.demo.p2">
```

- Külön *process*
- Külön memóriaterület
- *Shell*-ből *ps* paranccsal kipróbálható

# 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.

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

# Kotlin alapok

*Forrás: https://kotlinlang.org/docs/reference/*

# Történet és tulajdonságok

- 2011-ben jelent meg először
- JetBrains gondozásában
- Nyílt forráskódú nyelv
- 2017-es Google I/O: hivatalos támogatás Androidra
- Statikusan típusos
- Objektum orientáltság mellett a funkcionális programozást is támogatja

# A Kotlin főbb jellemzői

- JVM byte kódra (vagy akár JavaScriptre is) fordul
- Meglévő Java API-k, keretrendszerek és könyvtárak használhatók
- Automatikus konverzió Java-ról Kotlinra
- Null-safety
  - Vége a NullPointerException korszaknak
- Kód review továbbra is egyszerű
  - A nyelv alapos ismerete nélkül is olvasható a kód

# Konstansok, változók (val vs. var)

- Egyszeri értékadás – „val”

```
val score: Int = 1  // azonnali értékadás
val idx = 2   // típus elhagyható
val age: Int  // típus szükséges ha nincs azonnali értékadás
age = 3       // későbbi értékadás
```

- Változók (megváltoztatható) – „var”

```
var score = 0 // típus elhagyható
score += 1
```

- String sablonok

```
var score = 1
val scoreText = "$score pont"

score = 2
// egyszerű kifejezeések string-ek esetében:
val newScoreText = "${scoreText.replace("pont", "volt, most ")} $score"
```

# Változók `null` értéke

- Alapból a változók értéke nem lehet **`null`**

```
var a: Int = null
error: null can not be a value of a non-null type Int
```

- A ‘?’ operátorral engedélyezhetjük a null értéket

```
var a: Int? = null
```

  - Lista, melyben lehetnek **`null`** elemek

```
var x: List<String?> =
  listOf(null, null,
null)
```

  - Lista, mely lehet **`null`**

```
var x: List<String>? = null
```

  - Lista, mely lehet **`null`** és az elemei is lehetnek **`null`**-ok

```
var x: List<String?>?
= null
x = listOf(null,
null, null)
```

# Null tesztelés és az Elvis operátor

```
var nullTest : Int? = null
nullTest?.inc()
```

- `inc()` nem hívódik meg, ha `nullTest null`

```
var x: Int = 4
var y = x?.toString() ?: ""
```

- ha x null, akkor y “” értéket kap

# “Double bang” operator

- Kivételt dob, ha a változó értéke `null`

```
var x: Int? = null
x!!.toString()
kotlin.KotlinNullPointerException
```

# Függvények

- Függvény szintaxis

```
fun add(a: Int, b: Int): Int {
    return a + b
}
```

- Kifejezés törzs, visszatérési típus elhagyható

```
fun add(a: Int, b: Int) = a + b
```

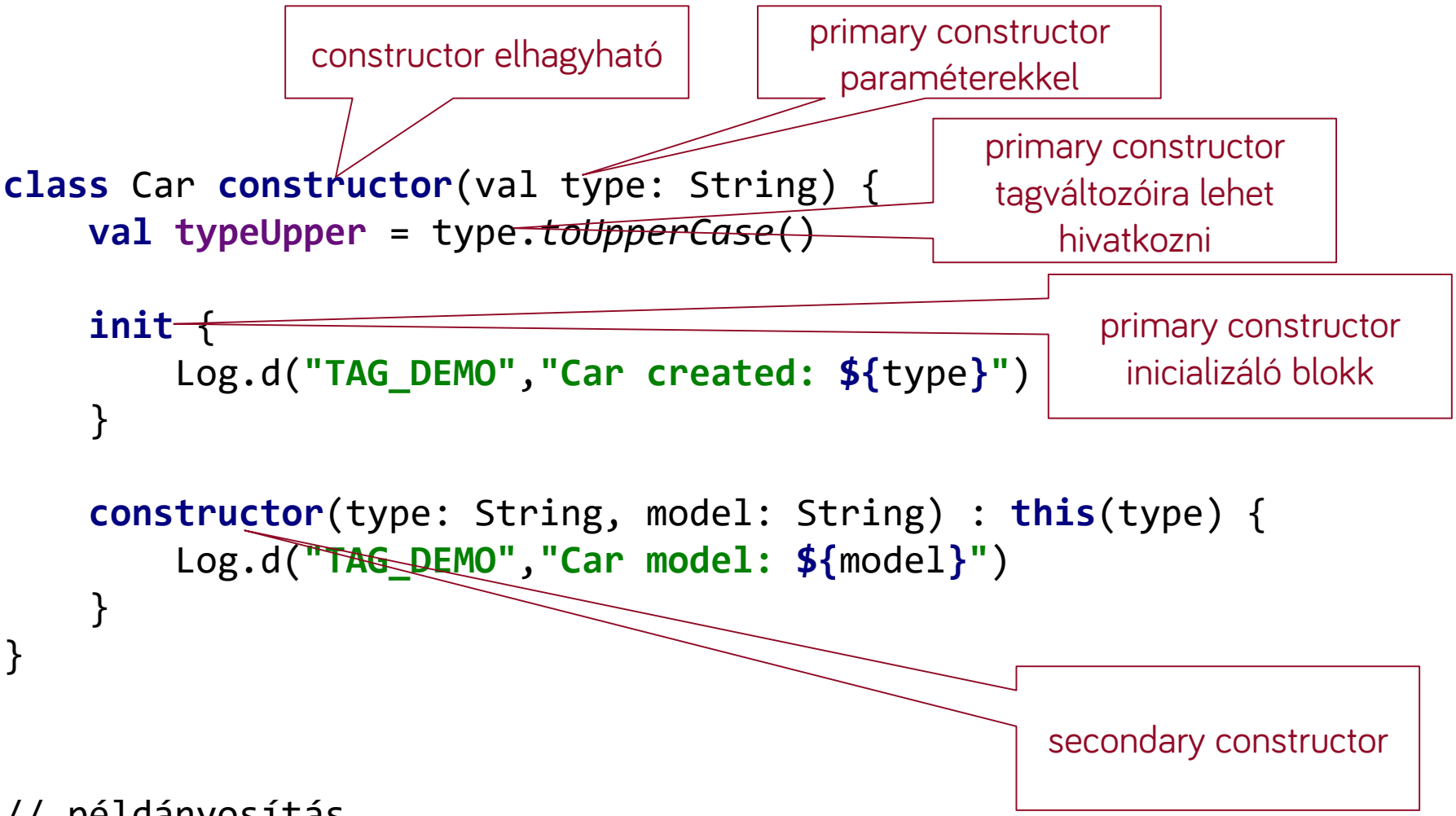
- Érték nélküli visszatérés – Unit

```
fun printAddResult(a: Int, b: Int): Unit {
    println("$a + $b értéke: ${a + b}")
}
```

- Unit elhagyható

```
fun printAddResult(a: Int, b: Int) {
    println("$a + $b értéke: ${a + b}")
}
```

# Osztályok




```
class Car constructor(val type: String) {
    val typeUpper = type.toUpperCase()

    init {
        Log.d("TAG_DEMO","Car created: ${type}")
    }

    constructor(type: String, model: String) : this(type) {
        Log.d("TAG_DEMO","Car model: ${model}")
    }
}


// példányosítás
val car = Car("Toyota")
```

# Leszármaztatás

alapesetben minden final

```
open class Item(price: Int) {
    open fun calculatePrice() {}
    fun load() {}
}


class SpecialItem(price : Int) : Item(price) {
    final override fun calculatePrice() {}
}
```

öröklés

Később már nem lehet felülírni

# Láthatóság – top level

- *„top level”: függvények, property-k, osztályok, objektumok és interfacek lehetnek egyből egy package-en belül*
- ***public***: mindenhonnan látható
- ***private***: Fileon belül látható
- ***internal***: modulon belül látható
- ***protected****: top level nem lehet*
- Default (**üres**) = *public*

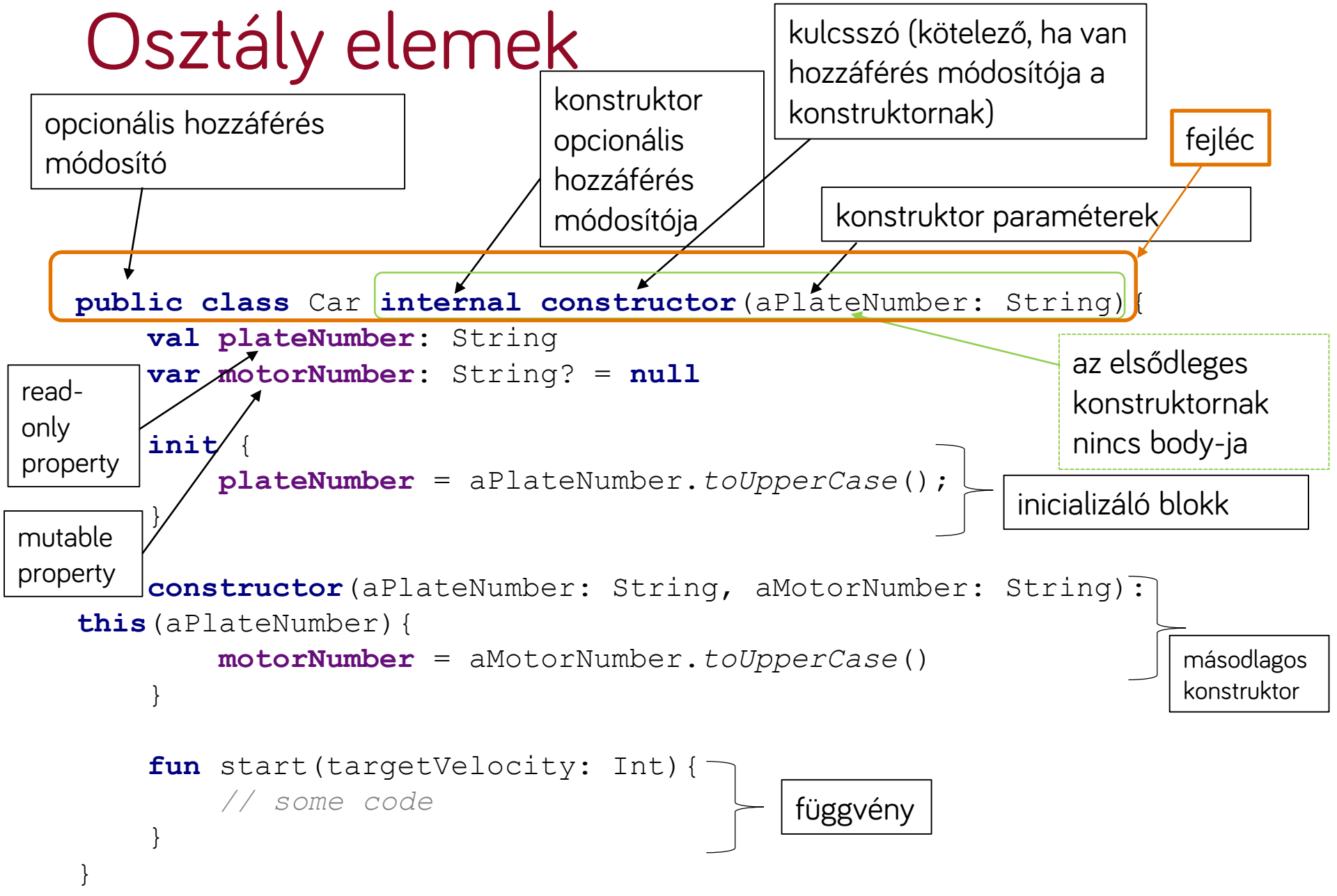
# Láthatóság – osztályon/interface-n belül

- ***private***: Fileon belül látható
  - Java-val ellentétben a külső osztályok nem látják az inner class-ok private tagjait
- ***protected****:* mint a private, de leszármazottban is látható
- ***internal***: modulon belül láthatja mindenki aki az osztályt látja
- ***public***: mindenki hozzáférhet aki az osztályt is látja

# Láthatóság példa

```
open class Outer {
    private val a = 1
    protected open val b = 2
    internal val c = 3
    val d = 4  // public alapértelmezetten

    protected class Nested {
        public val e: Int = 5
    }
}

class Subclass : Outer() {
    // a nem látható
    // b, c és d láthatók
    // Nested és e láthatók

    override val b = 5   // 'b' protected
}

class Unrelated(o: Outer) {
    // o.a, o.b nem láthatók
    // o.c és  o.d láthatók (ugyanaz a module)
    // Outer.Nested is nem látható és Nested::e is sem látható
}
```

# Osztály elemek



# Data class

```
data class Ship(val name: String, val age: Int)
```

- Automatikusan létrejön:
  - equals()/hashCode()
  - toString(): "Ship(name=Discovery, age=31)";
  - componentN() metódusok
  - copy() metódus

```
val discovery = Ship("Discovey", 31)
val (name, age) = discovery

//val name = discovery.component1()
//val age = discovery.component2()
```

- Követelmények Data osztályokkal szemben:
  - Primary constructor legalább 1 paraméterrel
  - Minden primary constructor paraméter *val* vagy *var*
  - *Data* classes nem lehet *abstract, open, sealed, vagy inner*

# Property-k

- **var** <propertyName>[: <PropertyType>] [= <property_initializer>] [<getter>] [<setter>]

```
class Ship(var name: String, var age: Int) {
    var detailedName: String
        get() = "$name $age"
        set(value) {
            name = value
        }
}
```

# Néhány érdekesség

- Range-k

```
val x = 4
val y = 3
if (x in 1..y+1) {
    Log.d("TAG_DEMO", "x benne van")
}
```

- Range iteráció

```
for (nr in 1..10 step 2) {
    Log.d("TAG_DEMO", "szam $nr")
}
```

- Lambda műveletek kollekciókon

```
val fruits = listOf("alma", "mango", "mandarin", "narancs")
fruits
    .filter { it.startsWith("m") }
    .sortedBy { it }
    .map { it.toUpperCase() }
    .forEach { Log.d("TAG_DEMO", "$it") }
```

# Melyik nem helyes Kotlin osztály fejléc?

A. class Ship constructor(name: String)

B. Ship constructor(name: String)

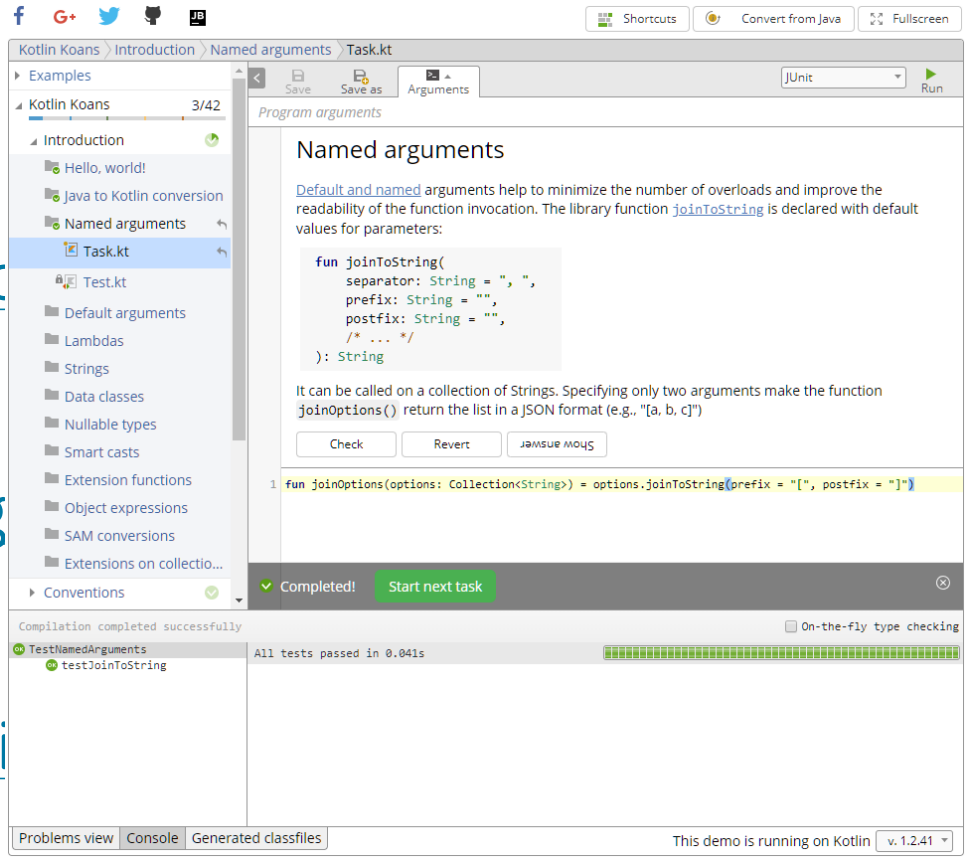
C. class Ship

D. class Ship internal constructor(name: String)

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

# Kotlin Koans

- Gyakorló kódrészek Uni
- IDEA és Android Studio
  - > https://kotlinlang.org/dc
- Online tutorial
  - > https://try.kotlinlang.org
- GitHub
  - > https://github.com/Kotli



# További hasznos helyek

- Kotlin in Action: By Dmitry Jemerov and Svetlana Isakova, Kotlin developers at JetBrains.
- Kotlin for Android Developers: By Antonio Leiva. One of the first books about Kotlin.
- Android Development with Kotlin: By Marcin Moskala and Igor Wojda.

# Az első Android alkalmazás

# Az első Android alkalmazás

Ősosztály

```
public class HelloAndroid extends Activity {
    lled when the activity is first created. */
    ide
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello Android!");
        setContentView(tv);
    }
```

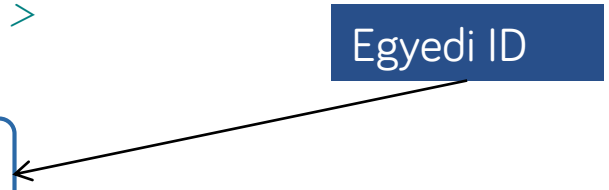
Ősosztály implementáció meghívása

TextView megjelenítése

# Android HelloWorld XML alapú UI-al 1/2

Hello Android XML (*res/layout/activity_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 {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_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.activity_main);
    final TextView myTextView =
       (TextView)findViewById(R.id.tvHello);
    myTextView.append("#");
    myTextView.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            myTextView.append("\n--CLICKED--");
        }
    });
}
```

Mivel anonim osztályból férünk hozzá

Egyszerű érintés esemény kezelés

# Az első Android alkalmazás Kotlin-ban ☺

# Egyszerű esemény kezelés

Kotlin extensions miatt használható

Lambda hívás

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        myTextView.append("#");

        myTextView.setOnClickListener{
            myTextView.append("\n--CLICKED--")
        }
    }
}
```

# Függvény mint paraméter

```
class MainActivity : AppCompatActivity() {

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

        btnTime.setOnClickListener(::click)
    }

    private fun click(view: View) {
        Toast.makeText(this,
                Date(System.currentTimeMillis()).toString(),
                Toast.LENGTH_LONG).show()
    }
}
```

# Eseménykezelő megadása layout-ban

```
<Button
    android:id="@+id/btnTime"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="click"
    android:text="Show" />
```

```
class MainActivity : AppCompatActivity() {

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

    fun click(view: View) {
        Toast.makeText(this,
                Date(System.currentTimeMillis()).toString(),
                Toast.LENGTH_LONG).show()
    }
}
```

# Gyakoroljunk!

- Készítsünk egy Pong alkalmazást!
- Érintett témák
  - Egyedi nézetek
  - Rajzolás
  - Szálkezelés
  - Képek kezelése

# Összefoglalás

- Android verziók
- A platform felépítése
- Fejlesztőkörnyezet beállítása
- Android projekt elemei
- Fordítás mechanizmusa
- Android Studio bevezetés
- Emulátor
- Első Android alkalmazás, eseménykezlés
- Pong alkalmazás

# A következő alkalommal...

- Alkalmazás komponensek
- Activity életciklus
- Haladó életciklus kezelés
- Állapot mentés
- Több képernyős alkalmazások
- Kotlin konvenciók
- Kotlin plugin képességei

# Köszönöm a figyelmet!

*peter.ekler@aut.bme.hu*

AutSoft