

# Android

## Az Android platform bemutatása

Dr. Ekler Péter

[peter.ekler@aut.bme.hu](mailto:peter.ekler@aut.bme.hu)



Department of  
Automation and  
Applied Informatics

# Oktatók

- Dr. Ekler Péter
  - > peter.ekler@aut.bme.hu
- Balogh Tamás
  - > balogh.tamas@autsoft.hu
- Gazdi László
  - > gazdi.laszlo@aut.bme.hu
- Kövesdán Gábor
  - > gabor.kovesdan@aut.bme.hu
- Braun Márton
  - > braun.marton@autsoft.hu
- Kapitány Erik
  - > kapitanyerik@gmail.com

# Tudnivalók

- Első héten nincs labor
- Tárgy honlap
  - > <https://www.aut.bme.hu/Course/android>
  - > <https://portal.vik.bme.hu/kepzes/targyak/VIAUAV21/>
- Neptun üzenet
- Levelező lista
  - > [bme-android@googlegroups.com](mailto:bme-android@googlegroups.com)
  - > <https://groups.google.com/group/bme-android>
- GitHub
  - > <https://github.com/peekler/Android-Kotlin-BME>
  - > <https://github.com/peekler/Android-BME> (Java)

# Tárgykövetelmények

- Heti 1 előadás és 1 labor
- *opcionális* Házi feladat, **de ajánlott**
- ZH: várhatóan **a félév közepén**, az előadás idejében
- Vizsgák: vizsgaidőszakban
- Elővizsga pótzH-val együtt pótlási héten
- Labor részvétel - 70%

# Előadás

- Az Android platform részletes bemutatása
- Példákkal fűszerezett tananyag
- Előadás és tankönyv
- Plusz pont szerzési lehetőségek!
- Mindenféle mobilos információk:
  - > Versenyek
  - > Mobil Klub
  - > „Expect the unexpected” 😊
- Neptun-ban szereplő e-mail cím aktualizálása!

# Könyv

- Android-alapú szoftverfejlesztés (2012)



# Kotlin

- Nyelv ismertető:
  - > <https://kotlinlang.org/>
- Kotlin Koan-ok gyakorolni:
  - > <https://kotlinlang.org/docs/tutorials/koans.html>
- KotifyDemo:
  - > <https://github.com/BaloghTamas/KotifyDemo>
- Java-ról Kotlinra tutorial:
  - > <https://github.com/Zhuinden/guide-to-kotlin>

# Labor

- 4 labor különböző időben
- A laborok legalább 70%-án részt kell venni
- Az előző előadáshoz kapcsolódó feladat megoldása a laborvezető támogatásával
- Labor végén érdemjegy (jóindulatú osztályzás 😊)
- Megajánlott jegybe beleszámít
- **Laborbeosztás:**
  - > Csere: e-mailben (peter.ekler@aut.bme.hu) az időpontok és a két fél nevének és Neptun kódjának megadásával, CC laborvezető



# Házi feladat

- A házi feladattal megajánlott jegyet lehet szerezni
- Nem kötelező, DE ajánlott!
- Jelentkezni a portálon:
  - > Rövid specifikáció
- Beadási határidő:
  - > Utolsó előtti hét vége
- Beadás:
  - > AUT portálra kell feltölteni
- Tesztelés:
  - > Emulátor
  - > Eszközöket is tudunk biztosítani
- Legalább 5 előadás téma használata
- Megfelelő komplexitás: 15-30 óra
- Ha nem sikerül megajánlott jegyet szerezni, akkor is szerezhető maximum 20 pont a vizsgára (100 pontos vizsgát feltételezve)

# Minta Házi Feladatok

- Játékok
- Szintax highlight-os Notepad két nyelv támogatásával
- Arcfelismerős contact szerkesztő/telefonhívás/stb.
- Sports tracker (idő/távolság/sebesség mérés, GPS, grafikon), háttérben futó szolgáltatás
- Szenzor alapú megoldás/játék
- Kiterjesztett valóság, kamera kezelés
- Adatkezelő, több lista, komplex adatbázis (pl.: túrák, fejlesztési feladatok nyomkövetése, kategorizálással, idő naplózással stb.)

# ZH

- Az előadás idejében
  - > November 26.
- Időpont és terembeosztás a tárgy honlapján
- Elméleti és gyakorlati feladatok
- Tipikusan az előadáson elhangzott kérdésekből és feladatokból válogatunk
- 40%-ba beleszámít a vizsgás jegybe
- 40%-ba beleszámít a megajánlott jegybe

# Vizsga

- Vizsgaidőszakban (elővizsga lehetőség szerint lesz)
- Elméleti és gyakorlati feladatok
- Tipikusan az előadáson elhangzott kérdésekből és feladatokból válogatunk
- Érdemjegy: 40% ZH + 60% Vizsga
- Inkább a megajánlott jegy legyen a cél 😊

# Megajánlott jegy

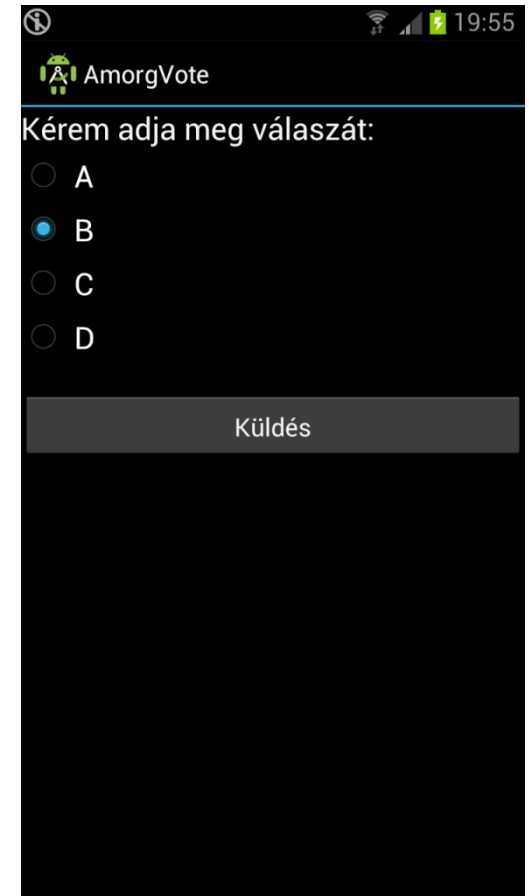
- Laborvezetővel egyeztetett specifikáció
- Legjobb 6 labor eredménye
- Megajánlott jegy: 40% ZH + 20% labor (legjobb 6) + 40% HF

# Labor - GIT

- Labor anyagok GIT-en
  - > <https://github.com/VIAUAV21/Labor>
- GitHub account létrehozása szükséges
- Regisztráció (csak BME hálózathoz):
  - > <https://android-labor-reg.herokuapp.com/>



# Gyakorlás előadás közben!



BME Vote

<https://play.google.com/store/apps/details?id=hu.bme.aut.amorg.amorgvote>

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

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

# Mi nem igaz az Android tárgyra?

- A. Megajánlott jegy szerezhető Házi Feladattal!
- B. A laborok során lehetőség van az előadáson elhangzottak gyakorlására!
- C. A házi feladatnak legalább 5 technológiát érintenie kell!
- D. Kötelező írásbeli vizsgát írni!

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



# Tartalom

- Bevezetés
- Mi is az Android?
- Milyen készülékeket támogat az Android?
- Az Android SDK fejlett képességei
- Miért fejlesszünk mobil alkalmazásokat és miért Androidra?
- Az Android fejlesztőkörnyezet és SDK bemutatása
  - > Telepítés
  - > Használat
  - > Eszközök
- Összefoglalás

# Bevezetés

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

# Piaci rész - Android



# Android eszközök 1/2

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



# Android eszközök 2/2



# 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 kódnev: 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, 5.1 (Lollipop)
- Android 6.0 (Marshmallow)
- Android 7.0, 7.1 (Nougat)
- Android 8.0, 8.1 (Oreo)
- Android 9.0 (Pie)
- Android 10 (Q)



# Az Android jövője

## Pletykák: Fuchsia

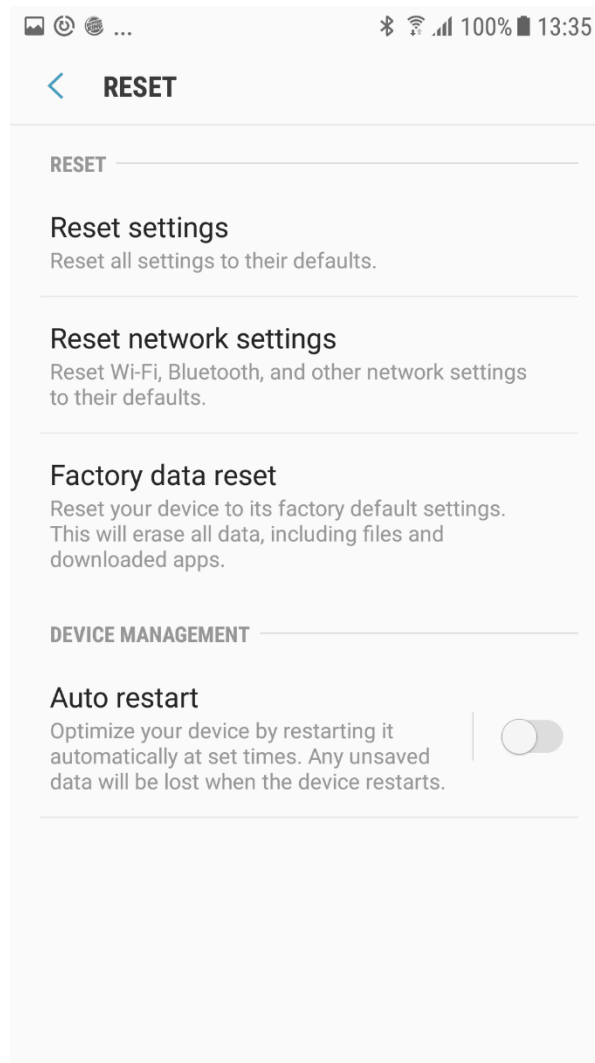
A B C D E F G H I J  
K L M N O P Q R S  
T U V W X Y Z

Még van ~8 évünk 😊

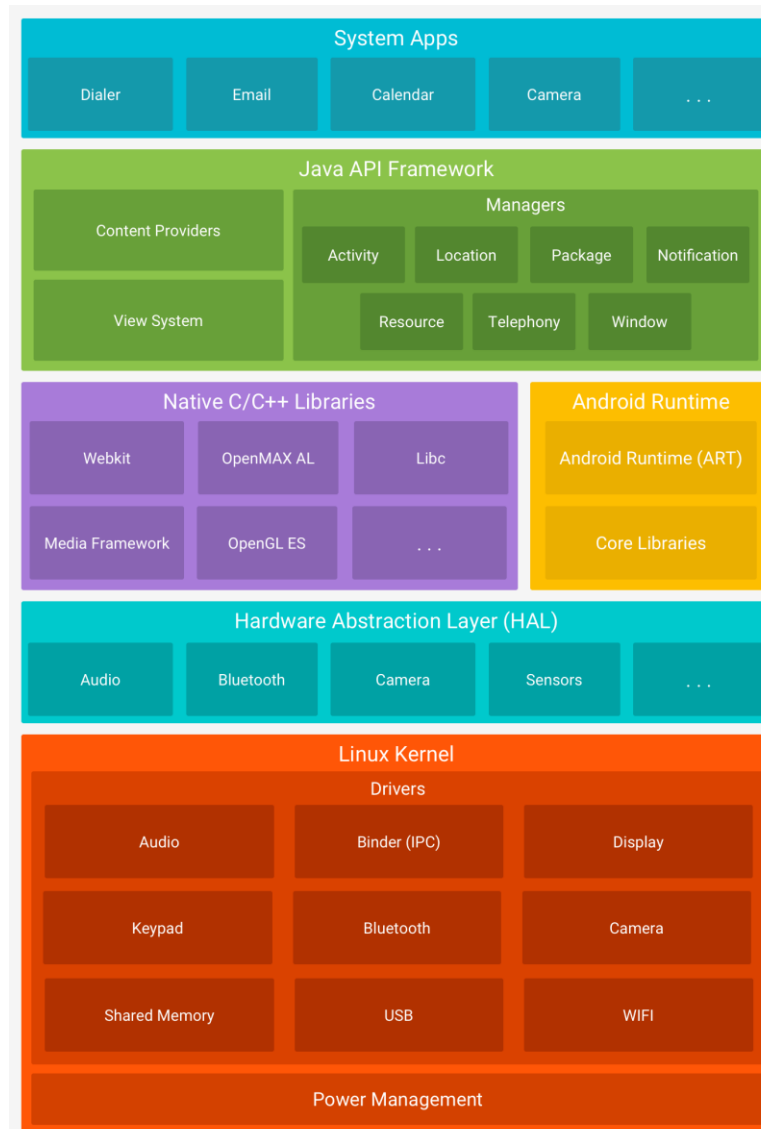




# Érdekesség ☺



# 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)



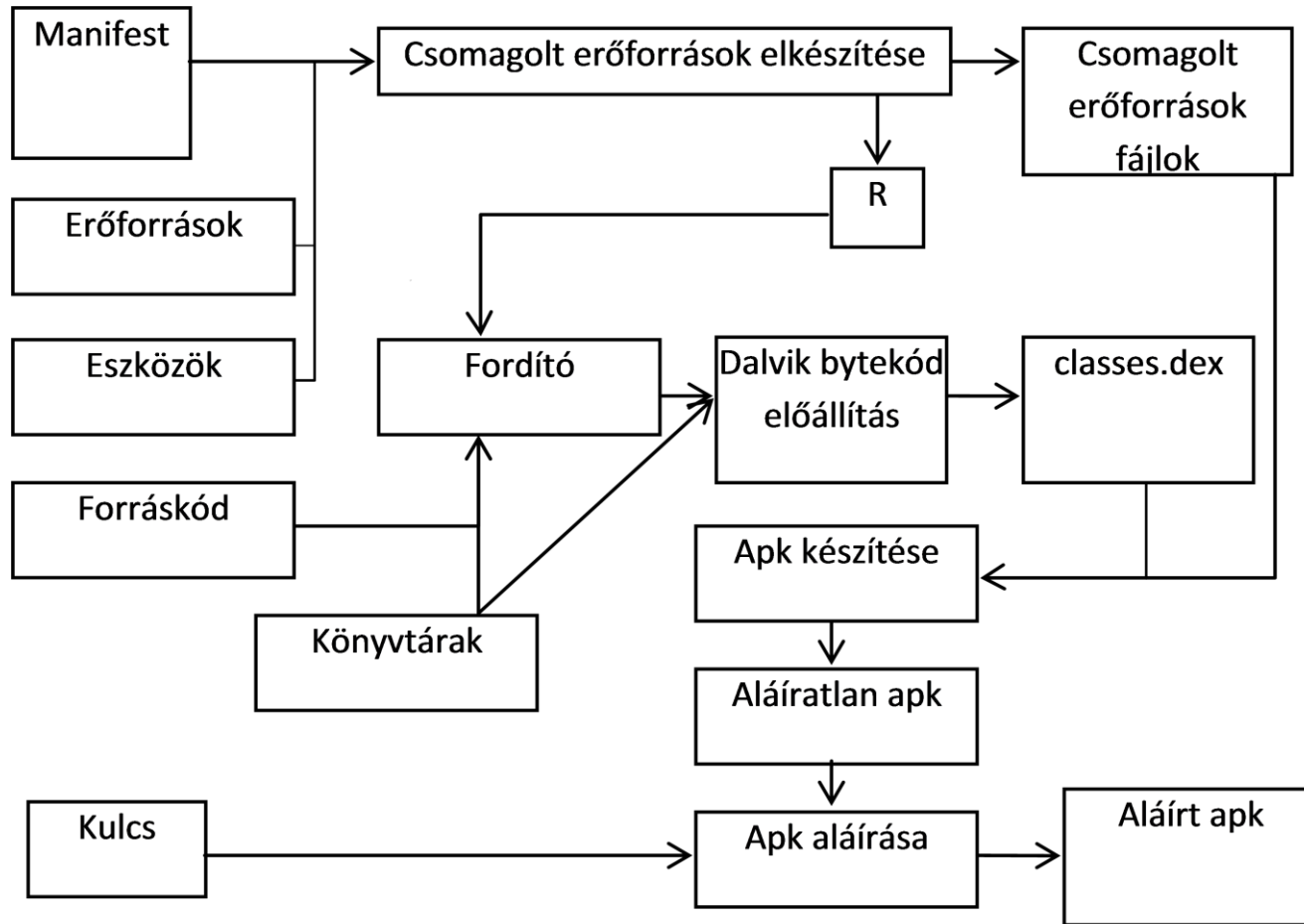
**Holnaptól Kotlin  
kell tanítanunk**

**Bazzeg**

# SDK komponensek

- SDK minden Android verzióra
- Dokumentáció
- Példakódok
- USB Driverrek (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)



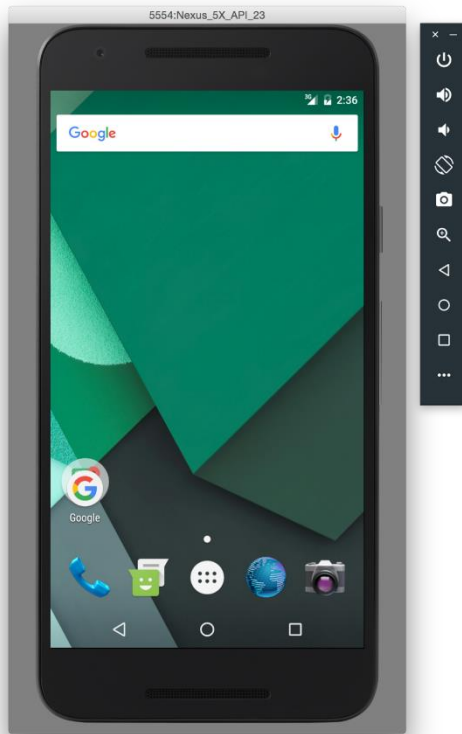
# Az Android .apk állomány

- 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: wxqnEAI0UA5nO5QJ8CGMwj kGGWE=
...
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/>)





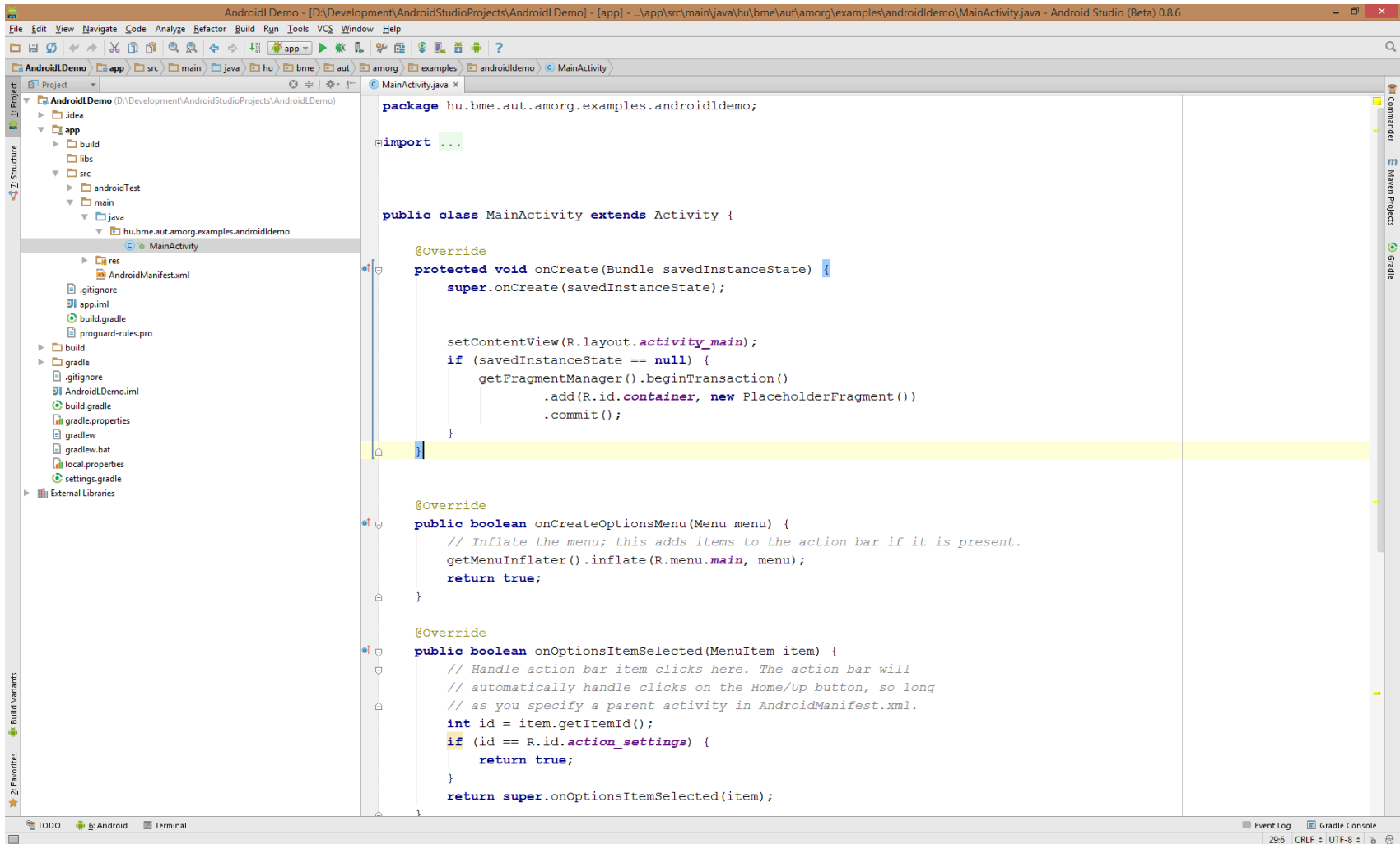
# Emulátor elérése konzolról

- 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 ☺

# Az első Android alkalmazás

# Az első Android alkalmazás

Ősosztály

```
public class HelloAndroid extends Activity {
```

Ősosztály  
implementáció  
meghívása

```
    Called 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);
```

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>
```

Egyedi ID



# 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" />
```

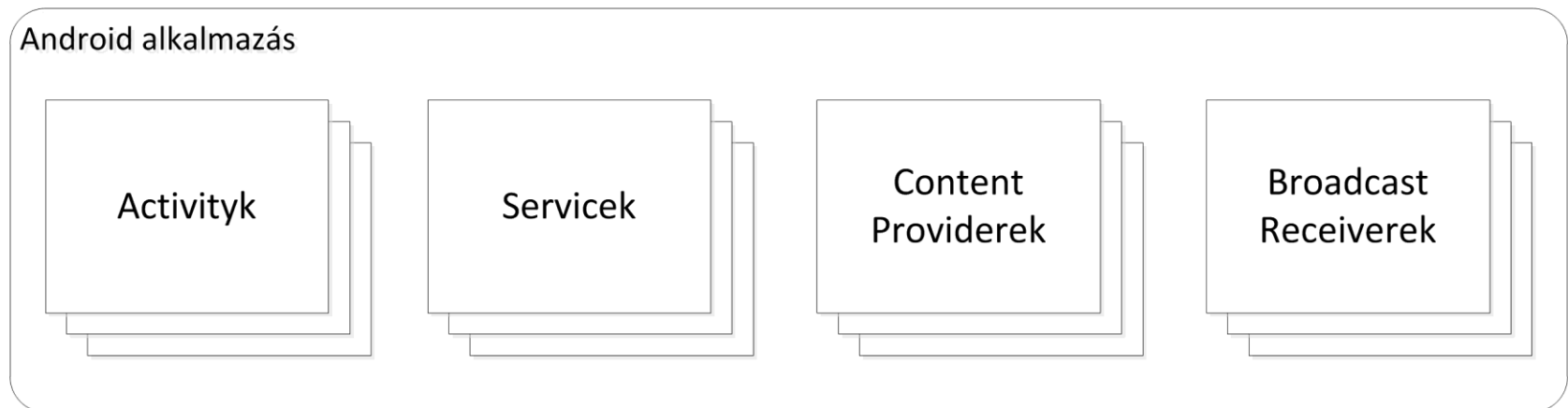
```
class MainActivity : AppCompatActivity() {

    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()
    }
}
```

# Android alkalmazás felépítése

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



# 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.daa1.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  
címke



# 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)

# 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>
```

# 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.

# Kotlin alapok

Forrás: <https://kotlinlang.org/docs/reference/>



# 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>

# 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 kifejezések 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
```

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

- > Lista, melyben lehetnek `null` elemek
- > Lista, mely lehet `null`
- > Lista, mely lehet `null` és az elemei is lehetnek `null`-ok

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

```
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}")  
    }  
}
```

constructor elhagyható

primary constructor  
paraméterekkel

primary constructor  
tagváltozóira lehet  
hivatkozni

primary constructor  
inicializáló blokk

secondary constructor

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

# Leszármaztatás

alap esetben minden final

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

öröklés

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

Később már nem  
lehet felülírni

# Láthatóság – top level

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



# Láthatóság – osztályon/interface-n belül

- ***private***: Osztályon 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

opcionális hozzáférés  
módosító

konstruktor  
opcionális  
hozzáférés  
módosítója

kulcsszó (kötelező, ha van  
hozzáférés módosítója a  
konstruktornak)

fejléc

konstruktor paraméterek

```
public class Car internal constructor(aPlateNumber: String) {
```

```
    val plateNumber: String
```

```
    var motorNumber: String? = null
```

```
    init {
```

```
        plateNumber = aPlateNumber.toUpperCase();
```

```
    constructor(aPlateNumber: String, aMotorNumber: String):
```

```
        this(aPlateNumber) {
```

```
            motorNumber = aMotorNumber.toUpperCase()
```

```
        }
```

```
    fun start(targetVelocity: Int) {
```

```
        // some code
```

```
    }
```

```
}
```

az elsődleges  
konstruktornak  
nincs body-ja

inicializáló blokk

másodlagos  
konstruktor

függvény

# Java field vs. Kotlin property

## Java

```
public class Car {  
    private String type;  
  
    public String getType() {  
        return type;  
    }  
  
    public void setType(String type) {  
        Log.d("TAG_CAR", "type SET");  
        this.type = type;  
    }  
}
```

## Kotlin

```
class Car {  
    var type: String? = null  
    set(type) {  
        Log.d("TAG_CAR", "type SET")  
        field = type  
    }  
}
```

# Összefoglalás

- Tárgykövetelmények
- Android verziók
- A platform felépítése
- Fejlesztőkörnyezet beállítása
- Android alkalmazás komponensei
- Manifest állomány, jogosultságok
- Kotlin alapok

# Kérdések

