Android Studio

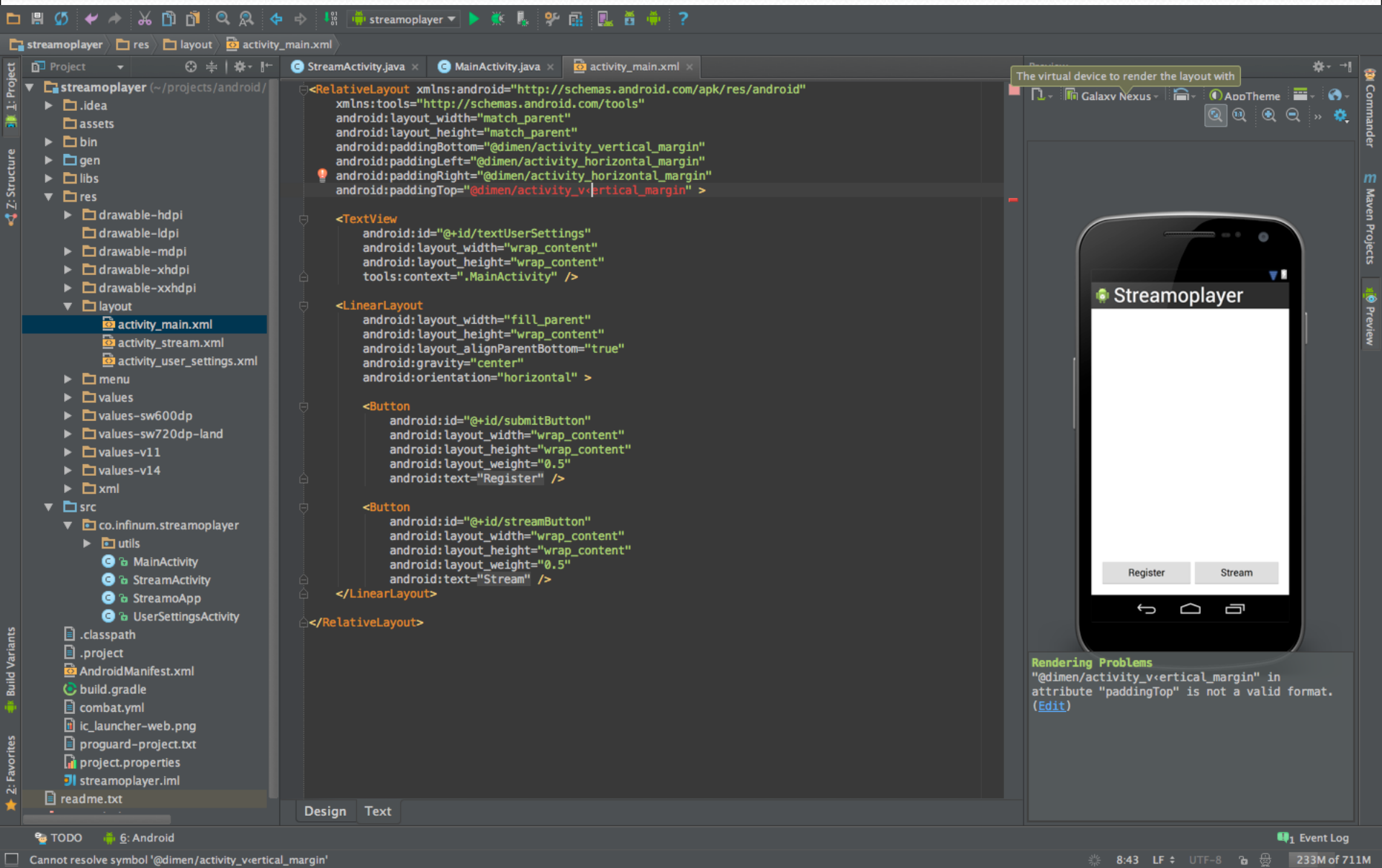
Integrált fejlesztői környezet (IDE)

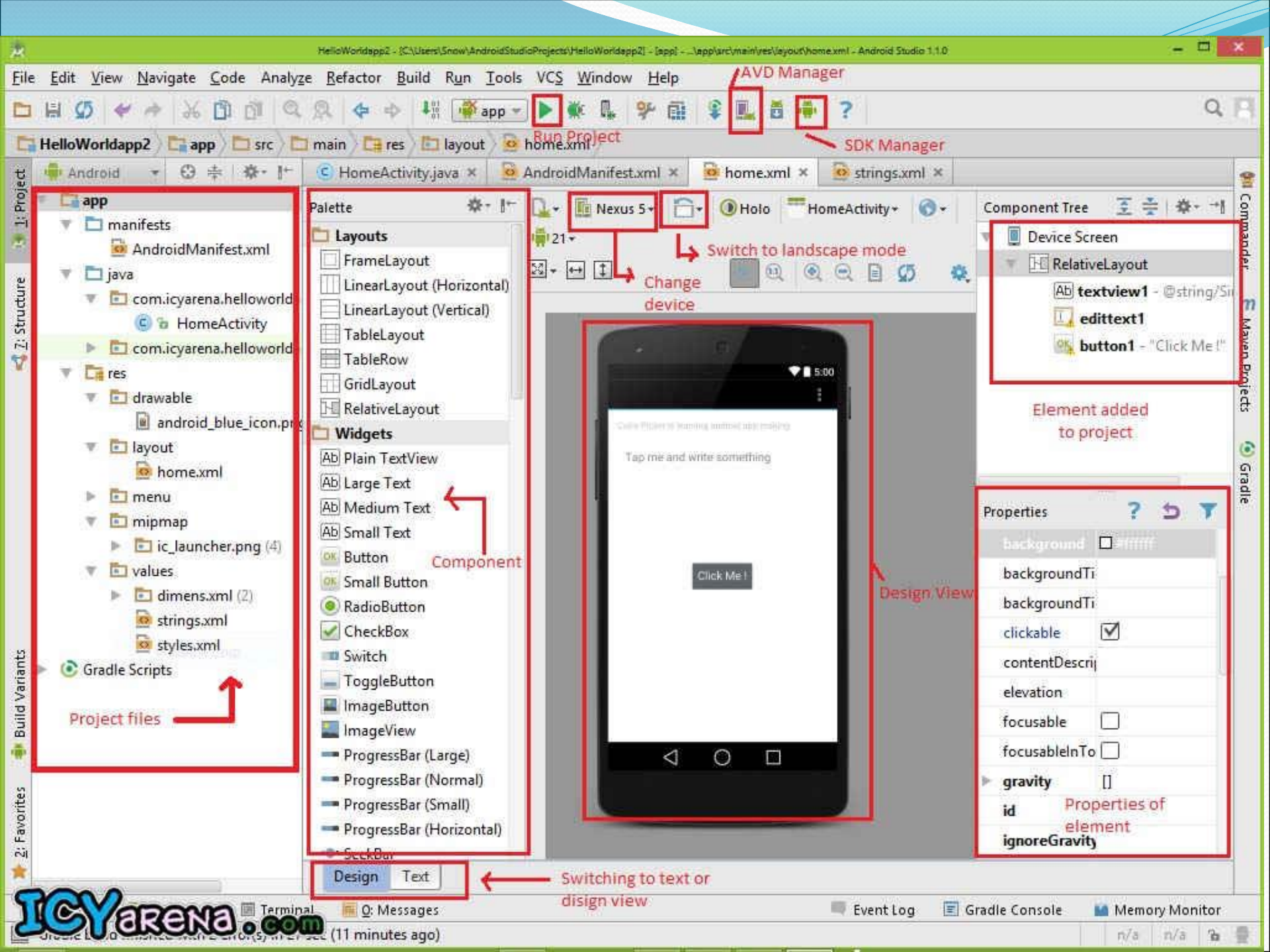
- IDE segítségével komplex projekteket kezelhetünk, amiben rengeteg segítséget kapunk
- Segítségével láthatunk akár egy virtuális telefonképernyőt is amibe behúzhatunk elemeket (UI), majd leprogramozhatjuk a viselkedésüket
- Az elemek tulajdonságainak a beállításaiban sok értéket egyszerűen átírhatunk
- A projektünkhöz tipikusan sok forrásfájl, multimédia (resource) tartozik, amiket könnyen kezelhetünk
- Kódolás közben javaslatokat tesz és hibákat is jelez

Android Studio

- Ezzel a fejlesztői környezettel (IDE) programozhatunk Android okostelefonunkra
- Ajánlásokkal segíti a kódkiegészítést
- Van beépített kódanalízise
- Kódolás közben észreveszi a hibákat
- Appokat fejleszthetünk Android telefonra, tabletra, Android Wearre, Android TV-re, Android Autora és Google Glassra
- Virtuális Android kijelzőn akár a számítógépünkön is megnézhetjük az Appunkat (Android programunkat)
- http://developer.android.com/sdk/index.html

Android Studio (IDE)




AVD Manager
Run Project
SDK Manager
Switch to landscape mode
Change device
Component
Design View
Element added to project
Properties of element
Project files
Switching to text or disign view
Design
Text
Tap me and write something
Click Me!

Android Studio projekt

- File menü -> New Project
- Application Name legyen *My First App*
  - Alkalmazásnév amit a felhasználók látni fognak
- Company domain:
  - Fejlesztők itt adják meg a cégükhöz tartozó egyedi azonosítójukat (pl.: weboldal doménnevüket) Java konvenciók szerint az alkalmazásunkhoz tartozó Androidos mappánk (UNIX környezetben Data/Data/...) neve a doménnév fordítottja lesz azért, hogy az egy céghez tartozó játékok fájlsorrendben egymás alá kerüljenek, pl:
  - com.adobe.reader (Adobe Reader), com.adobe.photoshop (Adobe Photoshop)

Projekt kompatibilitás

- Kilkk a Nextre
- Itt a céleszközök kiválasztásánál első megközelítésben el kell döntenünk, hogy milyen eszközre szeretnénk fejleszteni és melyik oprendszertől legyen kompatibilis a programunk (kompatibilitási megfontolások!!), minél kisebbre rakjuk az API-t (fejlesztői függvénykönyvtár), annál több eszköz tudja majd futtatni a programunkat, de bizonyos újabb lehetőségektől/megoldásoktól elbúcsúzhatunk. (Írja a támogatott eszközök százalékát)
- http://developer.android.com/training/multiple-apks/api.html
- http://developer.android.com/google/play/publishing/multiple-apks.html

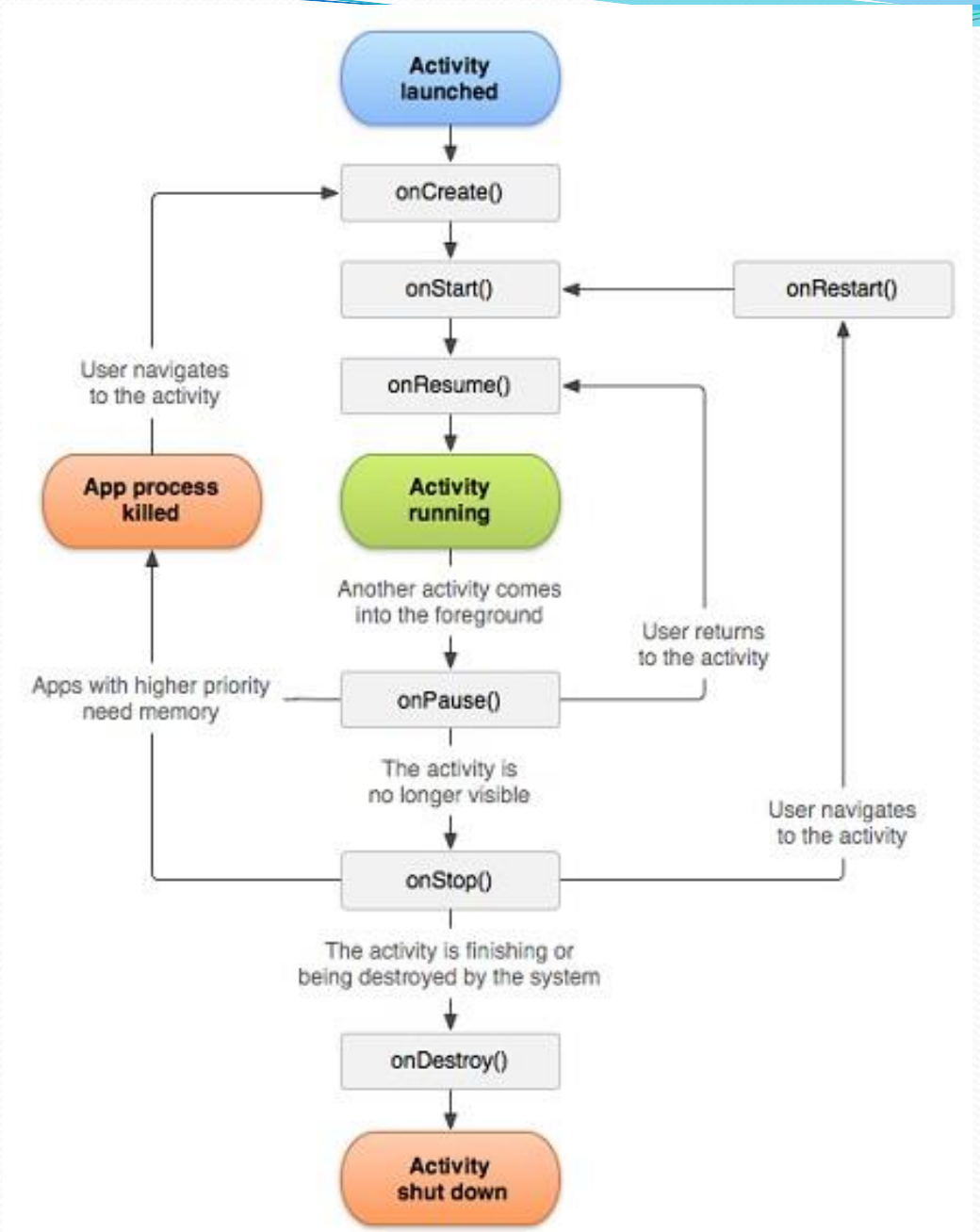
Android Studio sablonok

- Válasszuk most a *Phone and Tablet, API 8-at*
- Fejlett IDE programok felajánlanak többféle alapsablont, amiből alkalmazásunkhoz függően választhatunk, így a létrejött projektünk forráskódjaiban a legelső inicializálások automatikusan elkészülnek
- Válasszuk a *Blank Activityt* -> Next
- Customize the Activity résznél:
  - Activity Name: MyActivity
  - Többi automatikusan beállítódik
- Klikk a Finishre
- Így létrejön egy egyszerű „Hello World” Android Applikáció (App/program)(várjunk egy kicsit)

Android Studio Activities

- Az Activity az Android fejlesztőkörnyezetének a specialitása. Activity-k segítségével tudnak a felhasználók kapcsolatba lépni a programunkkal és ezekből több is lehet. Az applikációnk tipikusan egy fő Activity-vel indul, aztán más Activity-k jönnek be (esetleg kisebb ablak formában), ha másmilyen tartalmat (programrészt) szeretne a felhasználó elérni. (Gyakorlatilag többféle GUI összessége)
- Ezek átadják egymásnak a program futását, így az újabb Activity-k veszik át az irányítást, a régik meg pl. készenléti állapotba állnak (onPause())
- Ezeknek létrejönnek java főosztályok (classok)

Activites


Activity launched
onCreate()
onStart()
onRestart()
onResume()
Activity running
App process killed
User navigates to the activity
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Virtuális render error

- Virtuális render problémákat írhat ki az Android Studiónk azért, mert régi API szintet állítottunk be és az IDE alapértelmezetten csak a legújabb (Android 6) SDK toolokat szedte le, ezt megjavíthatjuk, ha Android SDK Managernél (Tools -> Android -> SDK Manager ): Android 4.2.2, API 17, revision 3-at leszedjük, majd ezt a megjelenítésben beállítjuk
- Az API verziót átállíthatjuk a vizuális nézetben az android ikonnal, rakjuk API 17-re, ez lesz a compiledSdkVersion és a targetSdkVersion (ezzel az SDK-val fordítjuk a programunkat és ez az SDK lesz a legújabb amivel teszteltük a programunkat), de az SDK-t érdemes minél magasabbra állítani a különböző optimalizálások miatt
- Tehát a render errorok nem jelentenek gondot, legközelebb magasabb API szintet nyugodtan beállíthatunk

Létrejött projektfájlok

- app/src/main/res/layout/activity_my.xml
  - Adott Activity-n levő elemek/tartalmak általános leírója
- app/src/main/res/layout/content_my.xml
  - Egy konkrét tartalom leírója (itt: Hello World doboz)
- app/src/main/java/com.mycompany.myfirstapp/MyActivity.java
  - Java forrásfájl, tartalmazza az első létrehozott főosztályunkat, aminek a neve
- app/src/main/AndroidManifest.xml
  - Az Appunk karakterisztikáit tartalmazza
- app/build.gradle
  - A Gradle (Android Java fordító) beállításai
- drawable-<density>/ - Képernyőhöz méretezett képek
- mipmap/ - Indító ikonok
- Az .xml fájlok vizuálisan szerkeszthetőek

Appunk futtatása

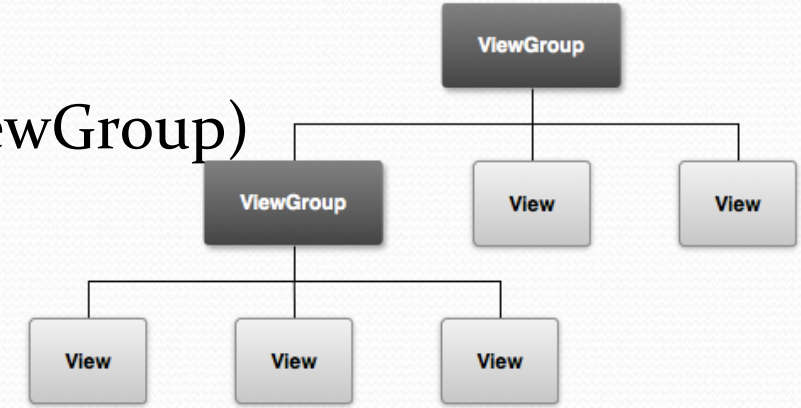
- Androidos telefonunkon:
  - *Settings* > *Developer options* résznél be kell állítani az *USB debuggingot* (Ha nem látható, akkor: Settings > About phone and tap Build number seven times)
  - Telefonunkat USB kábel segítségével csatlakoztassuk a gépünkre, Android Studioban menjünk a Run (zöld háromszög) ikonra, válasszuk ki az eszközünket és ha minden jól ment a telefonunkon le fog futni az első Hello World Android App programunk!
  - Alsó eszköztárból bezárhatjuk (Terminate) az Appunkat és a programfutást (Run) is megállíthatjuk
- Android Studio AVD (Android Virtual Devices segítségével)
  - Run segítségével válasszunk ki egy virtuális Android eszközt

Tag szabályok

- A HTML és a leíró nyelvekhez hasonlóan a különböző szövegrészeinket, elemeinket tagekkel kell körbefogni (körbezárni) pl: <tag>szöveg</tag>
- A nyitó tageken belül írhatunk attribútum tulajdonságokat
- Viszont az önbezáró tagekre itt most oda kell figyelni és be kell őket zárni (XHTML!)
  <tag attribútum1=„érték1”
  attribútum2=„érték2” />
- HTML-hez hasonlóan értelmezhetjük az elemek/tartalmak hierarchiáját (leszármazásait)
- Nagybetű érzékeny! (case-sensitive)

Android Layout

- Nyissuk meg a létrehozott content_my.xml fájlt (WYSIWYG vizuális szerkesztőben nyílik meg)
- Állítsuk át lent a Design fület Textre, itt attribútum tulajdonság érték párosok találhatóak, hasonlóan mint a HTML-nél
- XML szerkesztőben írjuk át:
  RelativeLayout legyen LinearLayout
  android:orientation=„horizontal”
  Töröljük ki a Paddingokat és a tools:contextet
- android:layout_width=„match_parent” és android:layout_height=„match_parent” mindig szükséges
- Adott Contentünk match_parent tulajdonsága miatt a szülőelem (ViewGroup) méreteit veszi fel teljesen



LinearLayout vs RelativeLayout

- LinearLayout orientáció alapján állítja be a leszármazott elemeit, mindegyiket egy irányba
- RelativeLayout segítségével az egyes leszármazott elemeket máshogy is feloszthatjuk
- A Layoutok (View) összességei, csoportosításai a ViewGroupok, amik a szülőelemek
- http://developer.android.com/guide/topics/ui/declaring-layout.html
- http://developer.android.com/guide/topics/ui/layout/linear.html
- http://developer.android.com/guide/topics/ui/layout/relative.html

Text mező hozzáadása

- Layoutokon (dobozokon) belül kell a különböző belső elemeket (pl.: text mezőket) megadni XML segítségével
- Illesszük be:

```
<EditText android:id="@+id/edit_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
```

- wrap_content: akkora amekkora a szövegnek (hint) szükséges
  match_parent: szülő méreteit veszi fel
- EditText azonosítója id: edit_message, ezzel hivatkozhatunk majd erre a konkrét elemünkre (mint HTML-ben)
- Hint: ez az alapszöveg jelenik meg amíg a felhasználó nem írt be semmit a text mezőbe

Resource Objects

- Adatforrás elemeket hivatkozás (referencia, mutató) segítségével használhatunk, ezek karakterláncok (stringek), képek, azonosítók vagy akár fájlok is lehetnek
- A mappastruktúra egységesen meg van adva az Android Studioban, ezekben találhatóak az adatforrásaink
- Ezekre az adatokra a @ segítségével hivatkozhatunk
- A stringeket a res/values/strings.xml fájlban definiálhatjuk name attribútumok segítségével (HTML-hez hasonló)
- Egyéni azonosítóknál (id) egyszer a megfelelő XML definíció helyén a + segítségével definiálhatunk egy új id-t, majd a @ segítségével plusz nélkül hivatkozhatunk rá a későbbiekben
- http://developer.android.com/guide/topics/resources/providing-resources.html

strings.xml

- res/values/strings.xml fájlban hozzuk létre a hiányzó string adatforrásainkat
- A jelenlegi helyett ez álljon:

```
<resources>
    <string name="app_name">My First App</string>
    <string name="edit_message">Enter a message</string>
    <string name="button_send">Send</string>
    <string name="action_settings">Settings</string>
</resources>
```

- Hivatkozott stringek helyett a megfelelő helyre akár inline (hardcoded string) is beírhattuk volna simán a stringünket (android:hint=„szoveg”), ez nem jó szokás, ugyanis a meglévő szövegünket hivatkozás segítségével tudjuk újra felhasználni és így elég csak egy helyen átírni (fordító figyelmeztet az inline string definíciókért)

Gomb hozzáadása

- res/layout/content_my.xml fájlt nyissuk meg újra
- Adjunk hozzá gombot:

```
<Button
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/button_send" />
```

- Ennek most nem adunk egyéni azonosítót (id-t), mert később nem akarunk majd erre az elemre hivatkozni
- android:layout_weight attribútum segítségével egymás melletti elemeknek adhatunk súlyozott szélességet (arányos szélességelosztást, pl: 1:2) (ez megfeleltethető a CSS százalékos megadásával)

Méretezési megfontolások

- Nemcsak a különféle eszközök képernyőméretei különböznek egymástól, hanem a felbontásaik is (density) így egy pixel minden telefonon más méretű, tehát a px mértékegység helyett valami állandóbb egységet kéne használni
- Erre vannak a density-independent pixelek *dp* mértékegységgel
- És a scale-independent pixelek *sp* mértékegységgel
- A kettő mértékegység számítása ugyanúgy történik, de az *sp* figyelembe veszi a szövegméreti felhasználói beállításokat is (szövegeknél mindig ilyet használjunk)
- Tehát ezt a két mértékegységet mindenhol és akkor minden képernyőfelbontáson ugyanolyanok lesznek a távolságok/méretek
- De ez még nem oldja meg a képernyőméreti problémák, ugyanis egy telefoni dobozméret tableten pazarlóan kicsi lenne

Különböző képernyőméretek

- A dp, sp értékeket tároljuk adatforrásokban, amiknek az egyedi attribútum nevei alkalmazásfüggőek
  (pl: <dimen name=„button_width”>10dp</dimen>)
- Ezek a méretértékek legyenek a res/values/dimens.xml fájlban és itt a dimen taget használjuk, majd az alkalmazási helyeken hivatkozunk rájuk „@dimen/button_width” segítségével
- Ha nagyobb képernyőméretekre is akarunk tervezni, akkor a dimens.xml fájlra jobb klikk new Value resource fájl segítségével különböző pl. képernyőméreti feltételeket adhatunk a fájlhoz, így a különböző fájlokban levő dimen egyforma nevű értékek közül az lesz használva, amihez tartozó fájl feltételei teljesülnek
- http://android4beginners.com/2013/07/appendix-c-everything-about-sizes-and-dimensions-in-android/