attrecto
NEXT TECH DIGITAL SOLUTIONS

Android szoftverfejlesztés
Alkalmazáskomponensek

ISO 9001 Certified

FT FINANCIAL TIMES 1000 Europe's Fastest Growing Companies

50™ Technology Fast 50 2016 CENTRAL EUROPE

500 Technology Fast 500 2016 EMEA

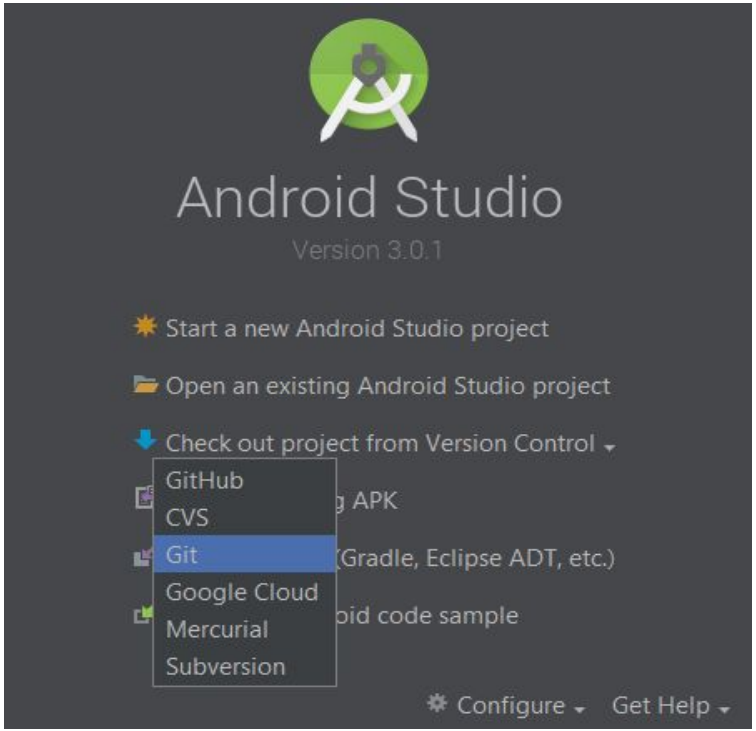
Környezet kialakítása

- Android SDK + Android Studio telepítése
  - Előny, ha van már fent Java SDK
- Git telepítése
  - A példa projektek a giten lesznek fent
- Fejlesztőkörnyezet kialakítása
  - Emulátor vs. fizikai eszköz
  - Emulátor: érdemes az órák előtt elindítani
  - Fizikai eszköz: USB debugging bekapcsolása

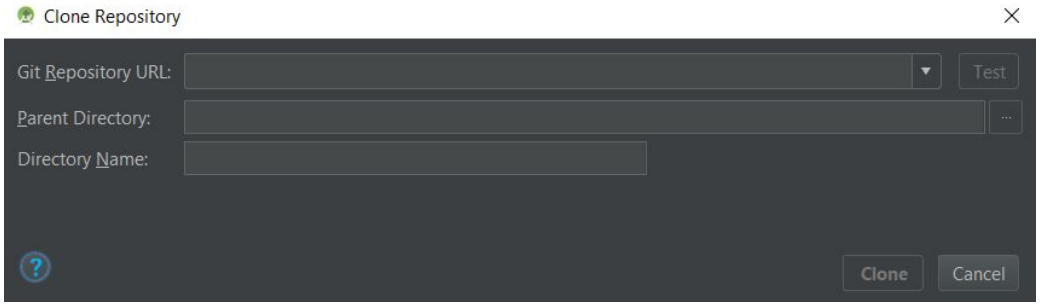
TARTALOM

- Hello World!
- LogCat
- Az Android alkalmazások felépítése
- Alkalmazáskomponensek
- A manifest állomány
- Erőforrások
- Egyszerű barkochba játék fejlesztése
- Activity életciklus és állapotmentés

HELLO WORLD


Android Studio
Version 3.0.1
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
GitHub
CVS
Git
Google Cloud
Mercurial
Subversion
APK
(Gradle, Eclipse ADT, etc.)
oid code sample
Configure
Get Help


Hello World!

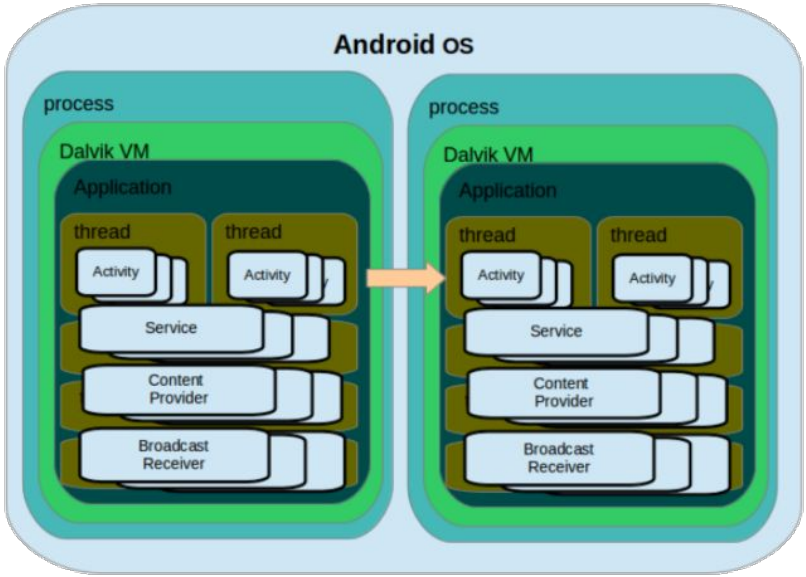

Clone Repository
Git Repository URL:
Test
Parent Directory:
Directory Name:
Clone
Cancel

DEBUG

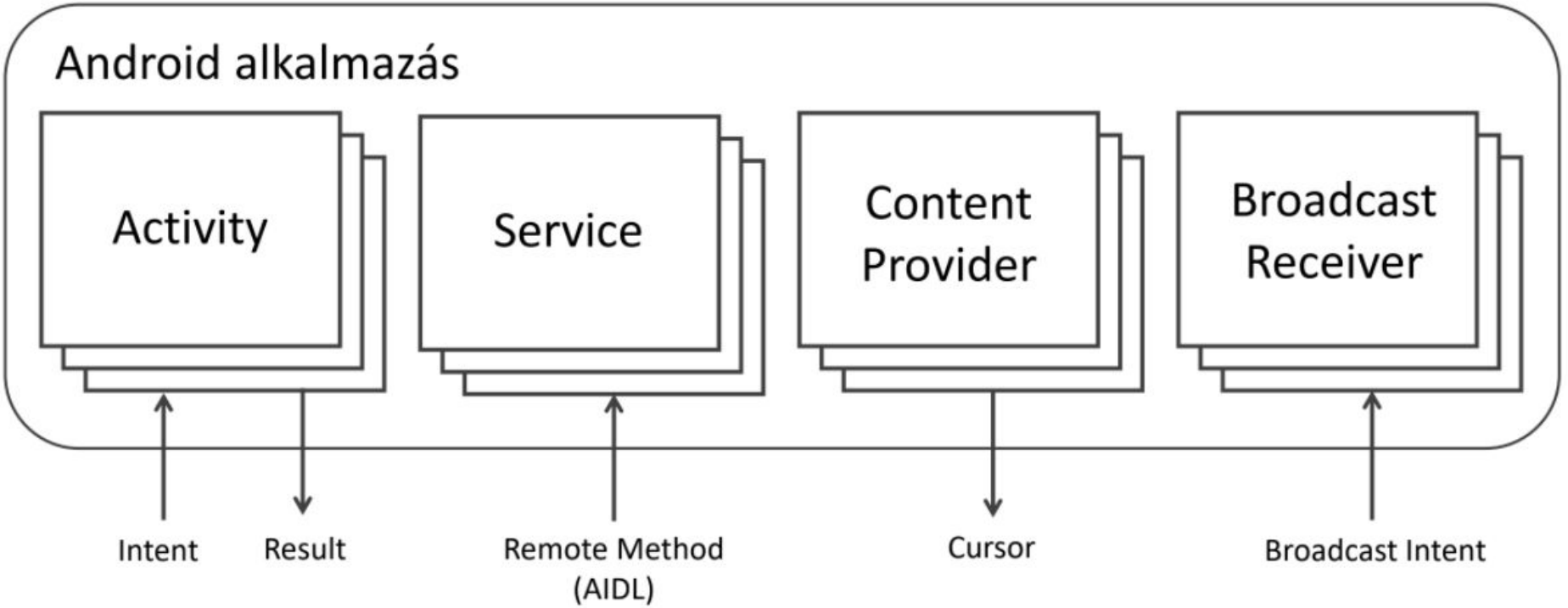
- Android LogCat
  - a Javával ellentétben itt nincs klasszikus karakteres „standard output”
  - `System.out.println()` helyett
    - `Log.v(String tag, String msg)`- verbose level
    - `Log.d(String tag, String msg)`- debug level
    - `Log.i(String tag, String msg)`- information level
    - `Log.w(String tag, String msg)`- warning level
    - `Log.e(String tag, String msg)`- error level
    - `Log.wtf(String tag, String msg)`- **what a terrible failure :)**
- On device debugging

FUTTATÁSI KÖRNYEZET

- Minden alkalmazás egy külön Linux felhasználó
  - Egyedi Linux felhasználói azonosítóval
- Minden processznek saját virtuális gép példánya van
  - Minden alkalmazás egy Linux processz
  - Az alkalmazások egymástól elkülönülten futnak
- A processz életciklusa
  - Indítás: amikor az alkalmazás valamelyik komponense elindul
  - Bezárás: amikor már nincs rá szükség, vagy amikor memóriát kell felszabadítani



AZ ALKALMAZÁS FELÉPÍTÉSE



ANDROID ALKALMAZÁSKOMPONENSEK

- Mindegyik komponensnek különböző szerepe van
- Bármelyik komponens önállóan is aktiválódhat
    - Ekkor elindul az alkalmazásunk
- Akár egy másik alkalmazás is aktiválhatja az egyes komponenseket
    - A komponenseket deklarálni kell az alkalmazás leíró (manifest) állományában
- Az erőforrásoknak (képek, szövegek, nézetek stb.) rendelkezésre kell állniuk akár különböző nyelveken és kijelzőméreteken

AZ ACTIVITYRŐL RÖVIDEN

- Általában teljes képernyőn futó felhasználói felület
- A megjelenítésért és a felhasználóval való interakcióért felel
- Egy alkalmazás tipikusan több, lazán csatolt Activity-ből áll
    - Létezik egy „fő” Activity, ahonnan a többi elérhető
- Az egyes Activity-k közötti kommunikáció Intent objektumokkal
- Más alkalmazásból is indítható
    - pl. fénykép csatolásához a beépített kamera alkalmazás
- `android.app.Activity` osztályból származik

A SERVICE-RŐL RÖVIDEN

- Egy hosszabb ideig a háttérben futó feladat
- Nincs felhasználói felülete
- Példa: torrent kliens, amely a háttérben fut
- Más komponens (pl. Activity) elindíthatja, vagy csatlakozhat hozzá vezérlés céljából (bind)
- `android.app.Service` osztályból származik

A CONTENT PROVIDER-RŐL RÖVIDEN

- Tartalomszolgáltató, feladata egy megosztott adatforrás kezelése
- Az adatot tárolhatjuk fájlrendszerben, SQLite adatbázisban, weben stb., amelyhez az alkalmazás hozzáfér
    - a Content Provider-en keresztül
- Példa: CallLog alkalmazás
    - Egy Content Provider-t biztosít, így elérhető a tartalom
- `android.content.ContentProvider` osztályból származik
- Kötelező felülírni a szükséges API hívásokat

A BROADCAST RECEIVER-RŐL RÖVIDEN

- Rendszer szintű eseményekre reagál
  - Pl. kikapcsolt a képernyő, alacsony az akkumulátor töltöttsége, bejövő hívás stb.
- Az alkalmazás indíthat saját „broadcast”-ot, pl. ha jelezni akarja, hogy végzett egy művelettel
  - Pl. letöltődött a torrent
- Nincs saját felülete
  - Inkább figyelmeztetés a status bar-ra, vagy másik komponens elindítása
- `android.content.BroadcastReceiver` osztályból származik
- Az esemény egy Intent formájában érhető el

A MANIFEST ÁLLOMÁNY

- XML állomány, definiálja az alkalmazás komponenseit
- A komponens indítása előtt ellenőrzi, hogy definiáltuk-e a kért komponenst
- Tartalmazhatja továbbá a következőket:
  - Az alkalmazást tartalmazó Java package (egyedi azonosító)
  - Szükséges minimális Android verzió
  - Szükséges hardware konfiguráció
  - Engedélyek (pl. internet-elérés, névjegyzék elérése)
  - Külső API könyvtárak (pl. Google Maps API)
- Alkalmazás telepítésekor ellenőrzi a rendszer

MANIFEST ATTRIBÚTUMOK ÉS TAGEK

- `android:icon`: az alkalmazás ikonja
- `android:name`: az Activity teljes neve package-dzsel együtt
- `android:label`: a felhasználói felületen megjelenő név
- komponensek:
  - `<activity>`: Activity
  - `<service>`: Service
  - `<provider>`: Content Provider
  - `<receiver>`: Broadcast Receiver
- A manifestben nem szereplő Activity-k, Service-ek és Content Provider-ek nem láthatók a rendszer számára
  - A Broadcast Receiver-ek viszont dinamikusan ki- és beregisztrálhatnak kódból

ERŐFORRÁSOK

- A forráskód mellett szükség van erőforrásokra is
  - Képek, hangok stb.
- Az XML-ben definiált felületek is erőforrások
  - Elrendezés, animáció, menü, stílus, szín stb.
- Az erőforrások rugalmassá teszik az alkalmazást
- A rendszer minden erőforráshoz egy egyedi azonosítót rendel
- Példa: logo.png kép
  - Másoljuk a képet a res/drawable mappába
  - Mentés után automatikusan azonosítót kap: R.drawable.logo, ezzel hivatkozhatunk rá
  - Az azonosítók az R.java fájlban vannak (soha ne módosítsuk!)

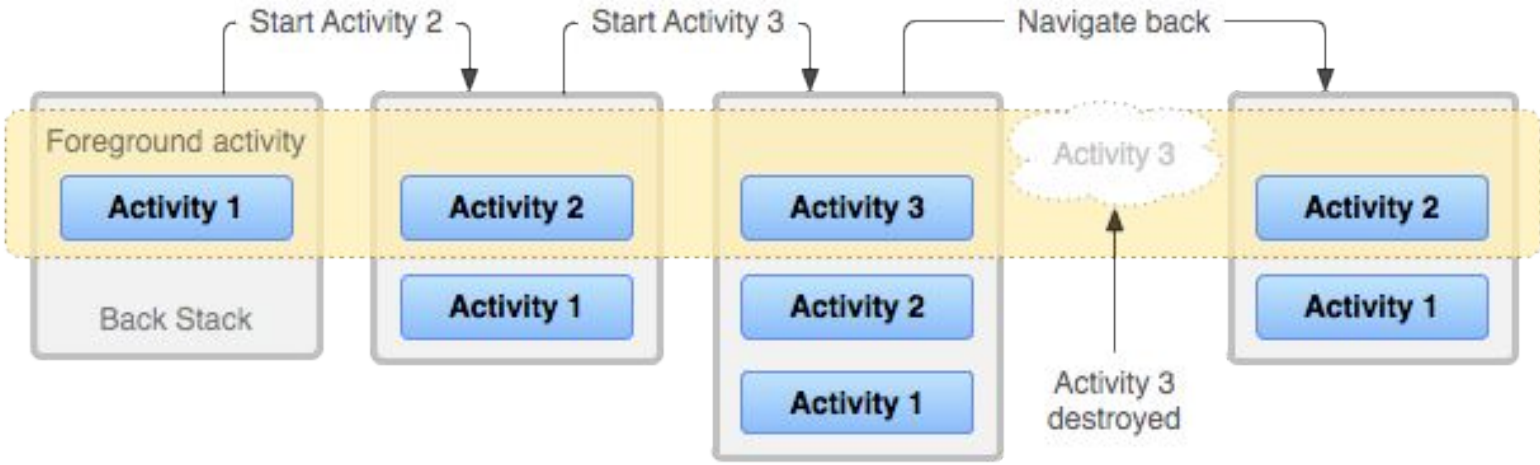
ERŐFORRÁSOK HASZNÁLATÁNAK ELŐNYEI

- Rugalmasság
  - A könyvtárak nevei után minősítőket írhatunk
- Példa: többnyelvűség támogatása
  - Strings.xml
    - res/values/
    - res/values-hu/
    - res/values-en/
    - res/values-de/

ACTIVITY BACK STACK

- Új Activity indításakor az aktuálisan előtérben lévő leáll (stopped)
- A leállított Activity-t a rendszer megőrzi a back stack-en
    - Back stack: stack adatszerkezet (LIFO)
- Amikor az új Activity elindul, rákerül a back stack-re, és megkapja a vezérlést (focus)
- Vissza gomb megnyomásakor az aktuálisan futó Activity lekerül a back stack tetejéről, és az alatta lévő kapja meg a vezérlést

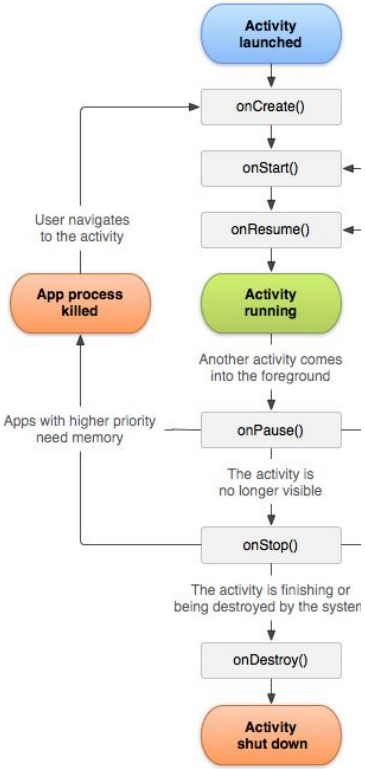
ACTIVITY BACK STACK

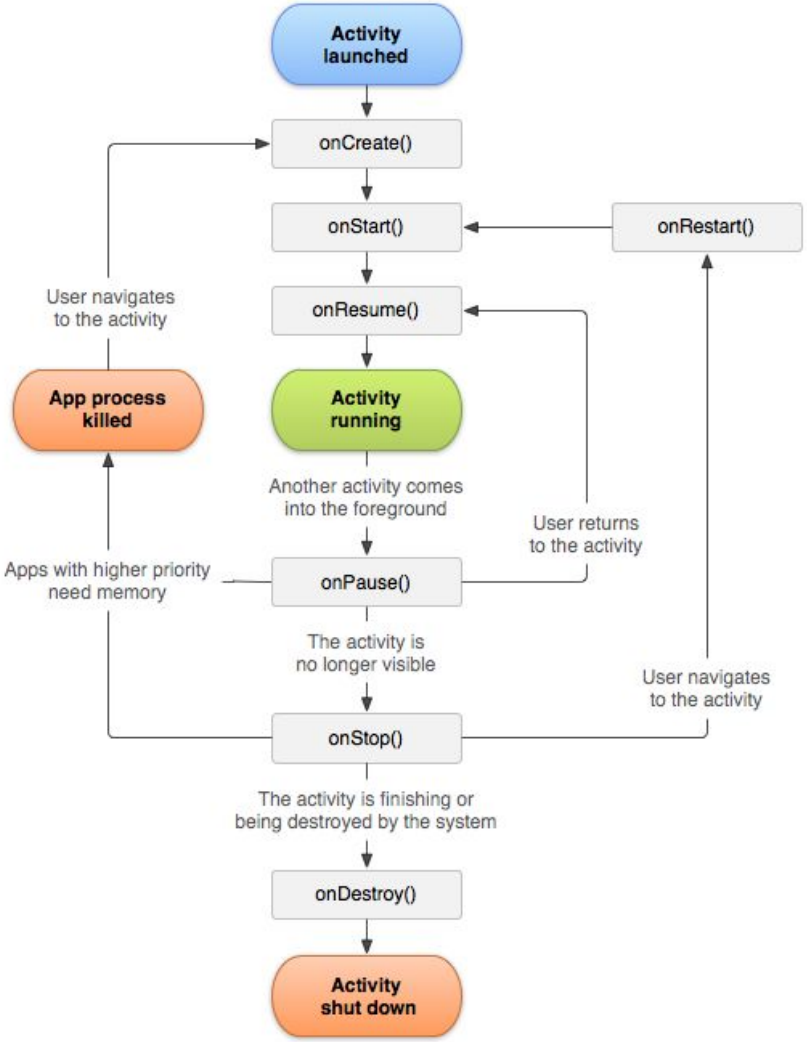


ACTIVITY ÉLETCIKLUS CALLBACK

- Amikor egy Activity leáll egy másik indulása miatt, akkor erről értesítést kap az életciklus callback metódusokon keresztül
  - onCreate, onStop, onResume, onDestroy stb.
- Az Activity ezekre az eseményekre reagálhat
  - Pl. stop esetén tipikusan érdemes a nagyobb objektumokat (DB/hálózati kapcsolat) elengedni
- Amikor az Activity visszatér (resume), újra kell kérni az erőforrásokat

ACTIVITY I


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


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

ACTIVITY ÉLETCIKLUS CALLBACK METÓDUSOK

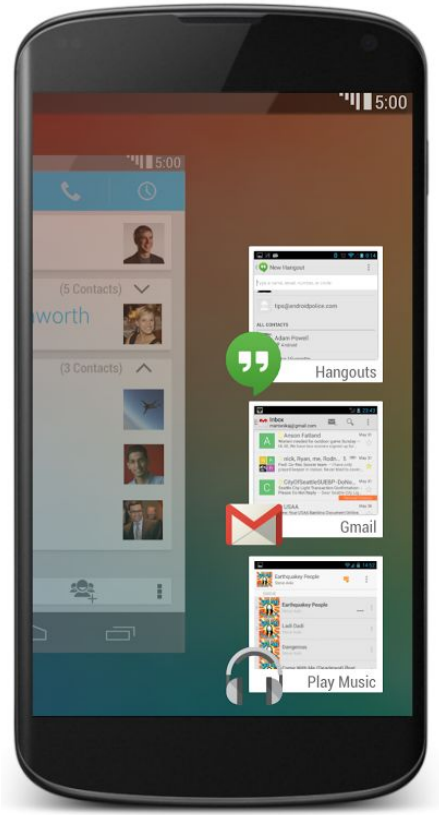
- onCreate() - az Activity létrejön és beállítja a megfelelő állapotokat (pl. a layoutot)
- onDestroy() - a még lefoglalt állapotban lévő összes erőforrás felszabadítása
- onStart() - az Activity már látható, feliratkozhatunk pl. Broadcast Receiver-ekre
- onStop() - az Activity nem látható, itt pl. leiratkozhatunk Broadcast Receiver-ekről
- onRestart() - az onStop() után hívódik meg, még az 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 pop-up jellegű Activity mögött
- **tapasztalat:** memóriahiány esetén nem az Activity-t, hanem a teljes processzt állítja le a rendszer

ACTIVITY VÁLTÁS

- Életciklus callback metódusok hívási sorrendje két Activity esetén:
  - **A** Activity onPause() metódusa
  - **B** Activity onCreate(), onStart() és onResume() metódusai (már **B** látható)
  - **A** Activity onStop() metódusa, mivel már nem látható
- Ha a **B** Activity adatbázisból olvas, melyet **A** ment el, akkor a mentés az onPause() metódusban kell, hogy megtörténjen
  - **B** csak így láthatja azt, amire szüksége van

MULTITASKING

- Task (elvégzendő feladat)
  - Több Activity-t használhat
  - Akár több alkalmazásból is
- A HOME gomb megnyomásával a rendszer a kezdőképernyőre lép
  - Új taskot indíthatunk
  - Ilyenkor a rendszer megőrzi az előző task back stack-jét
  - Memóriagondok esetén bezárhat Activity-ket
- Az új task új back stack-et kap



ACTIVITY ÁLLAPOTMENTÉS

- A felhasználó nem tudja, hogy amikor visszalép egy Activity-re, akkor a rendszer azt újra létrehozta, vagy csak megnyitotta a memóriából
- `onSaveInstanceState()`
  - A rendszer hívja meg, mielőtt az Activity-t bezárná
  - Pl. egy másik Activity kerül előtérbe
- Az értékeket egy `Bundle` objektumba lehet menteni
  - A `Bundle` objektumot paraméterként kapja az `onCreate()` metódus

DEMO

- Egyszerű barkochba játék

KÖSZÖNÖM A FIGYELMET!

Attrecto Zrt.
Attrecto Next Tech Digital Solutions

H-9024 Győr, Wesselényi str. 6.
info@attrecto.com