BUTE Department of Automation and Applied Informatics
AMORG
APPLIED MOBILE RESEARCH GROUP

# Android alapú szoftverfejlesztés

## 1. Labor

## Az Android platform bemutatása

## Tartalom

# 1 Felkészülés a mérésre

A mérés célja, hogy bemutassa az Android fejlesztőkörnyezetet, az alkalmazáskészítés, illetve a tesztelés és fordítás folyamatát, az alkalmazás felügyeletét, valamint az emulátor és a fejlesztőkörnyezet funkcióit. Továbbá, hogy megismertesse egy több Activity-ből álló alkalmazás elkészítésének módját.
A mérés során a mérésvezető részletesen bemutatja az eszközöket, a feladat pedig bizonyos fejlesztőkörnyezettel és emulátorral kapcsolatos kisfeladatok megoldása, valamint egy több Activity-ból álló alkalmazás elkészítése lesz.
A mérési útmutató 2-5 fejezetei az előadás anyagát kiegészítve az Android platform jellemzőit ismerteti, míg a mérési feladatok a 6. fejezettől kezdődnek.
A méréshez az alábbi témákat érinti:

- Az Android platform alapfogalmainak ismerete
- Eclipse fejlesztőkörnyezet alapok
- Android Emulátor tulajdonságai
- DDMS nézet jellemzői
- Android projekt létrehozása és futtatása emulátoron
- Manifest állomány felépítése
- Activity fogalom, több Activity-t kezelő alkalmazás
- Activity életciklus modell

## *1.1 A mérés elvégzéséhez szükséges anyagok*

A mérés során egy jegyzőkönyvet kell készíteni (csak az első mérésen lesz! ☺), melyben a szöveges feladatokra 1-1 soros választ kell adni képernyőképekkel alátámasztva (ahol lehetséges). A mérés végén a jegyzőkönyvet és az alkalmazást kell egy betömörített ZIP állományban feltölteni.

# 2 Az Android platform

## *2.1 Android*

Az Android platform népszerűségének napjainkban egyre növekszik. A gyártók szempontjából a platform mellett a legkomolyabb érv az ingyenes elérhetőség és a nyíltság. Emellett a Google folyamatos innovatív megoldásainak köszönhetően a legújabb és legnépszerűbb fejlesztések is szinte azonnal elérhetővé válnak az Android alapú készülékeken. Az egyik ilyen tipikus funkció az arcfelismerés. Jelenleg az Android operációs rendszer mobilok esetén a 4.1-es verziónál tart. A platform melletti határozott lépésként a Google maga is kiadta saját márkaneve alatt készült telefonját először Nexus One azonosítóval, majd a 4.1-es Androidot futtató Nexus Prime azonosítóval.


**Nexus One készülék**

Az Android platform egyrészt a Java nyelvű fejlesztést támogatja, azonban elérhető egy úgynevezett natív SDK is (NDK), mellyel C++ nyelven készíthetünk alkalmazásokat. A natív támogatás különösen fontos, ha gyors, hatékony algoritmusokra van szükség.

**Android készülékek**

Napjainkban az úgynevezett tablet készülékek piaci részesedése egyre jelentősebb, sőt tulajdonképpen egy új piaci eszközcsoportként is gondolhatunk rájuk. Az Android platform első tablet zászlóshajója a Samsung Galaxy Tab, melyet több más gyártótól származó készülék is követett.


**Galaxy Tab**

## 2.2 A Platform néhány jellemzője

Az Android platformon alkalmazások fejlesztéséhez általánosan a Java nyelvet használhatjuk (SDK – Standard Development Kit), azonban alacsonyabb szintű funkciók eléréséhez lehetőségünk van natív kódot is készíteni (NDK – Native Development Kit). A natív kód hívásához használhatjuk a JNI-t (Java Native Interface), azonban a rendszer támogatja az osztott könyvtárak (shared libraries) használatát is. A fejlesztés megkönnyítésére lehetőségünk van arra, hogy egy projekten belül elhelyezzük a Java és C++ kódrészeket. A továbbiakban a natív irányt nem emeljük ki, hiszen érvényesek rá a Standard C++ szabályok.

Az Android alkalmazások az ún. „Dalvik” virtuális gépen futnak, mely tulajdonképpen egy átdolgozott, optimalizált Java virtuális gép.

Az Android követi a Java nyelv fejlődését, így használhatunk magasabb szintű nyelvi elemeket is, ellentétben például a Java ME-vel, ahol csak a 3-as nyelvi eszköztár támogatott.

A Java nyelv és a Java alkalmazások felügyeltek, így a memóriakezelésért a

futtatókörnyezet és a virtuális gép felelős. A memória felszabadításért a GC (Garbage Collector) felelős, azonban ez nem jelenti azt, hogy felelőtlenül bánhatunk az objektumok létrehozásával. Törekedni kell azok folyamatos felszabadítására, hiszen a GC csak a már nem hivatkozott és nem használt objektumok memóriaterületét tudja felszabadítani.

Android fejlesztés esetén a forráskód és a felhasználói felület különválik, a felhasználói felület definiálására XML állományokat használhatunk, így lehetőségünk van a felület deklaratív módon való megadására. Ez azonban nem teszi kizárttá, hogy forráskód szintjén is létrehozzuk, elérjük és manipuláljuk a felhasználói felületet. Ökölszabályként elfogadott, hogy a felhasználói felület minél inkább különöljön el a forráskódtól.

A projektleíró állomány szintén XML formátumban érhető el (Android Manifest).

Az eseménykezelés a Java-ban megszokott módon történik, a megfelelő objektumokhoz ún. Listener-eket definiálhatunk és a megfelelő interface függvényeken keresztül kapunk értesítéseket az események bekövetkezésekor.

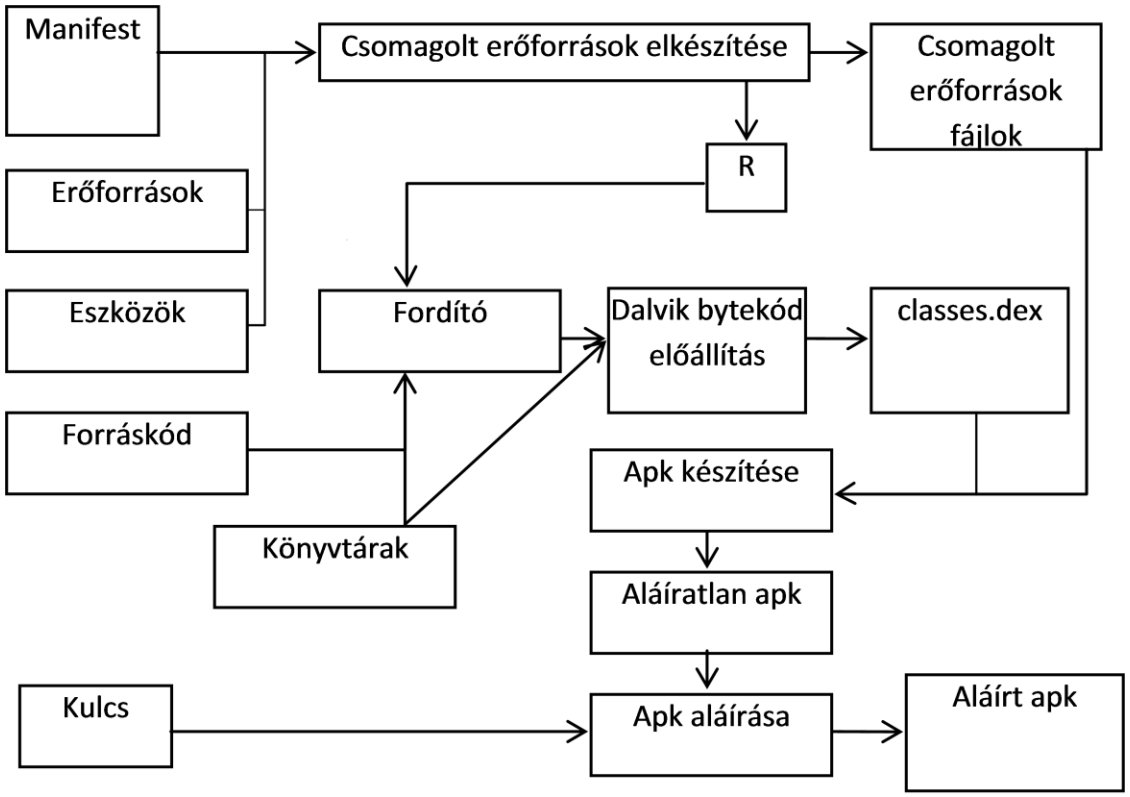
Kivételkezeléshez szintén a standard Java try-catch-finally kivételkezelő módszer használatos.

# 3 A fordítás menete Android platformon

A projekt létrehozása után a forráskód az src könyvtárban, míg a felhasználói felület leírására szolgáló XML állományok a res könyvtárban találhatók. Az erőforrás állományokat egy R állomány köti össze a forráskóddal, így könnyedén elérhetjük Java oldalról az XML-ben definiált felületi elemeket. Az Android projekt fordításának eredménye egy APK állomány, melyet közvetlenül telepíthetünk mobil eszközre.

## *3.1 A fordítás lépései*

- A fejlesztő elkészíti a Java forráskódot, valamint az XML alapú felhasználói felületleírást a szükséges erőforrás állományokkal.
- A fejlesztőkörnyezet az erőforrás állományokból folyamatosan naprakészen tartja az „R.java” erőforrás fájlt a fejlesztéshez és a fordításhoz.
- A fejlesztő a Manifest állományban beállítja az alkalmazás hozzáférési jogosultságait (pl. Internet elérés, szenzorok használata, stb.).
- A fordító a forráskódból, az erőforrásokból és a külső könyvtárakból előállítja a Dalvik virtuális gép gépi kódját.
- A gépi kódból és az erőforrásokból előáll a nem aláírt APK állomány.
- Végül a rendszer végrehajtja az aláírást és előáll a készülékekre telepíthető aláírt APK.



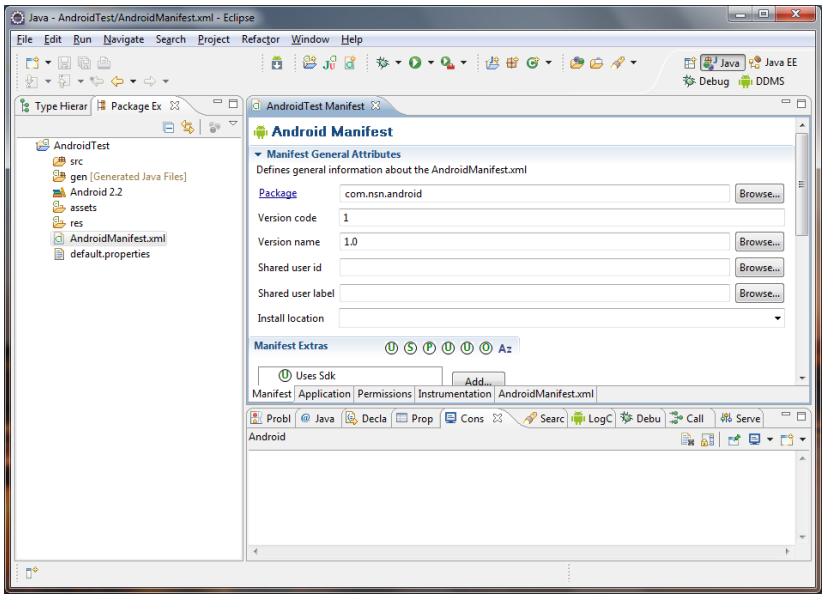

**A fordítás lépései Android platformon**

## 3.2 Megjegyzések

- A teljes folyamat a fejlesztői gépen megy végbe, a készülékekre már csak bináris állomány jut el.
- A külső könyvtárak általában JAR állományként, vagy egy másik projekt hozzáadásával illeszthetők az aktuális projekthez.
- Az APK állomány leginkább a Java világban ismert JAR állományokhoz hasonlítható.
- A Manifest állományban meg kell adni a támogatni kívánt Android verziót, mely felfele kompatibilis az újabb verziókkal, régebbi verzióra azonban már nem telepíthető.
- Az Android folyamatosan frissülő verziói nagy gondot jelentenek a fejlesztőknek.
- Az Android alkalmazásokat tipikusan az Android Marketben szokták publikálni, így az APK formátumban való terjesztés nem annyira elterjedt.
- A teljes folyamat a szoftverfejlesztők számítógépein megy végbe, az ügyfélhez futtatható gépi kód jut el.

# 4 Android fejlesztőkörnyezet

Android fejlesztésre a labor során a közismert Eclipse fejlesztőkörnyezetet fogjuk használni.




**Eclipse Android projekt - manifest állomány grafikus megjelenítésben**

### 4.1.1 Jellemzők

- Az Eclipse egy közkedvelt fejlesztő környezet Java, Java EE, C++ és egyéb projektek fejlesztéséhez, akár LaTeX projektek kezelésére is használható.
- Létezik Windows, Linux és Mac OS verzió egyaránt.
- Számos pluginnal kiegészíthető, így a fejlesztők rugalmasan testre szabhatják.
- Az Android SDK és Eclipse plugin megfelelő telepítése után a fejlesztőkörnyezet képes az elkészült projekteket egy emulátorban elindítani, mely nagymértékben megkönnyíti a fejlesztést.
- A fejlesztőkörnyezet támogatja az USB-vel csatlakoztatott készülékeken való fejlesztést is, továbbá az on-device debug is támogatott.

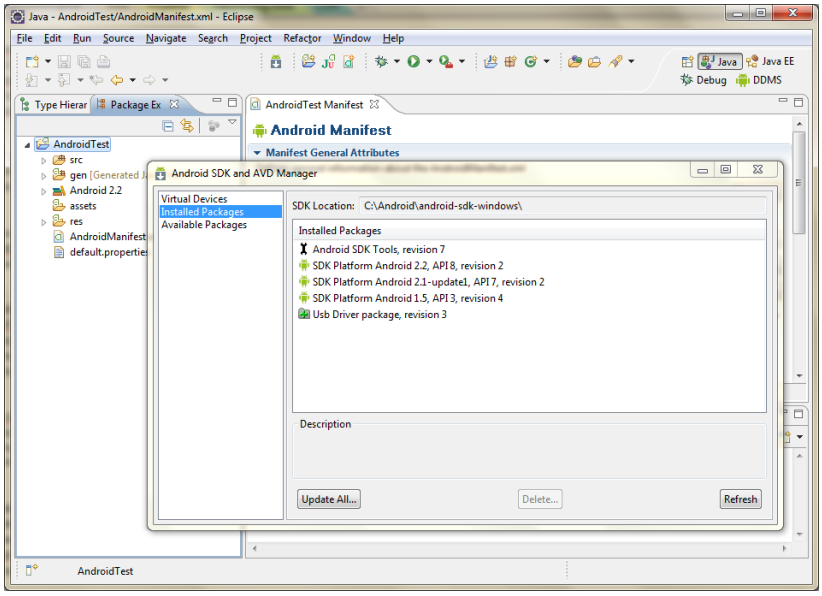
### 4.1.2 Használat

Az Eclipse fejlesztői környezet felhasználói felületére nagy mértékben jellemző a rugalmasság, hiszen minden segédablak áthelyezhető és átméretezhető. Ez a szabadság azonban néha problémákkal, illetve nem várt működéssel is jár. A következőkben az alapértelmezett elrendezés szerint mutatjuk be a környezetet.

- Tipikusan bal oldalt található a Project tree, ahol a Workspace-ben megnyitott projektek találhatók.
- A fenti menüsorban az Eclipse-re vonatkozó részletes beállítások érhetők el.
- Középen a szerkesztő nézet található, ahol az éppen megnyitott állományt jelenik meg, vagy éppen egy megfelelő designer.
- Az eclipse különféle perspektívákat támogat, melyek a teljes elrendezést és a megnyitott ablakokat fogják egybe. Tipikus példa a Debug perspektíva, mely a debugoláshoz szükséges ablakokat jeleníti csak meg, hogy a fejlesztő egyből láthassa a szükséges információkat.
- Android esetében a DDMS perspektívát szokás még használni, ahol az emulátor, vagy a mobil készülék beállításai, jellemzői érhetők el. Pl.: fájlrendszer, aktuális pozíció, stb.
- Az alkalmazások futtatása és a kód kiegészítés a megszokott módon működik, az emulátor első indításkor eléggé lassú.

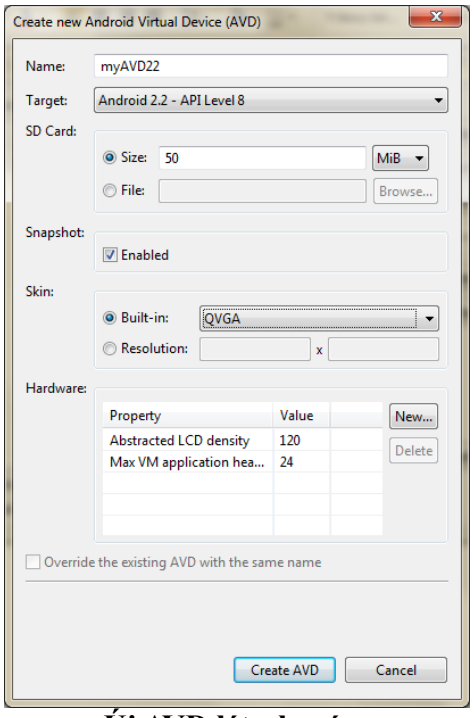
### 4.1.3 Mobil fejlesztésre

- A projekt létrehozásához válasszuk a „New->Android project” funkciót, mely alapértelmezetten egy HelloWorld alkalmazást generál a megfelelő állományokkal és könyvtárstruktúrával.




**Android SDK és AVD Manager**

- Futtatás a szokásos módon a „Run” paranccsal történik, melynek hatására alapértelmezetten elindul az emulátor. Az emulátor használatához elsőként létre kell hoznunk egy Android Virtual Device-t (AVD) az Android SDK and AVD Manager alkalmazásban. Létrehozáskor legfontosabb, hogy megadjuk az virtuális eszköz nevét, emulált SD kártya méretét, valamint a cél felbontást.

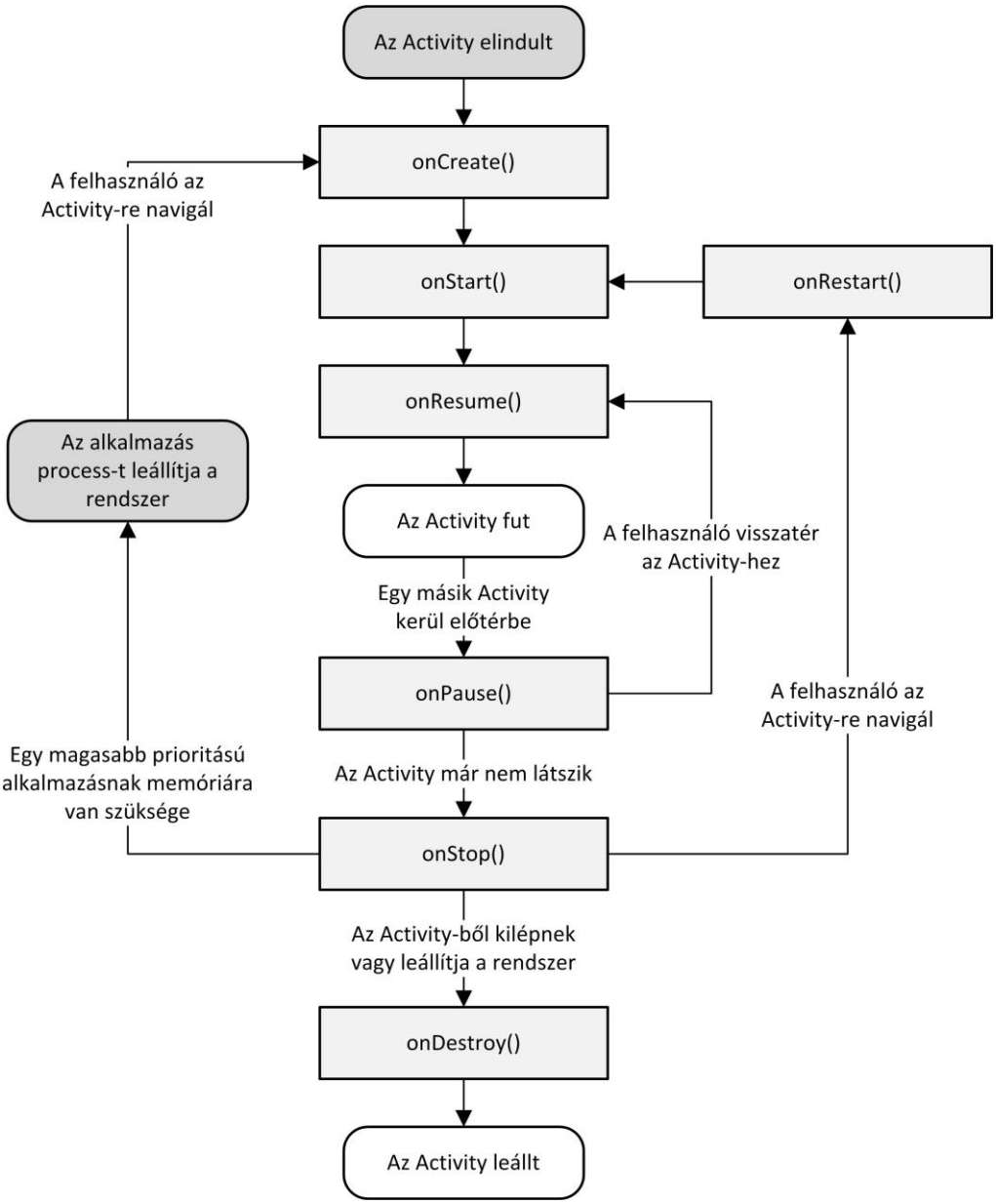



**Új AVD létrehozása**

a) Amennyiben USB kábellel egy androidos készüléket csatlakoztatunk a számítógéphez, lehetőség van az alkalmazás azonnali telefonon való tesztelésére is.

# 5 Android Activity életciklus modell

Android platformon egy alkalmazás több Activity-ből állhat, melyek önálló életciklussal rendelkeznek.




**Activity életciklus modell**

Ha egy új Activity elindul, és a hozzá tartozó felhasználói felület a már futó Activity-k képernyőképe elé kerül, az előző Activity paused állapotba megy át, és bekerül az előzmények vermébe (history stack). A verem segítségével a felhasználó navigálhat vissza-irányba a korábban megnyitott ablakok között. Opcionálisan megadható, hogy egy Activity ne kerüljön be az előzmények közé. A következő ábra egy Activity életciklusát szemlélteti.

# 6 Laborfeladatok

## 6.1 Az Android forrás vizsgálata

a) Töltse le az Andriod forrást (git helyett, http://atleast.aut.bme.hu/EklerP/android/sources-2.1-eclair.zip).
b) Laborvezető segítségével fussák át a forrás szerkezetét.
c) Vizsgálja meg az AlarmClock alkalmazást (com.android.alarmclock), melyik osztály lehet felelős a riasztás beállításáért? Mit csinál a SetAlarm.java popAlarmSetToast(…) függvénye?

## 6.2 Feltelepített SDK áttekintése

a) Vizsgálja meg a laborvezető irányításával a feltelepített Android SDK-t és annak könyvtár szerkezetét.
b) Hol található az Android dokumentáció?
c) Hogyan lehet frissíteni az SDK-t? Hogyan lehet újabb platform verziókat letölteni?
d) Melyik nézeten lehet az emulátorok felbontását, tulajdonságait átállítani?
e) Indítson el Android 2.2-őt emulátoron!

## 6.3 Emulátor áttekintése

a) Vizsgálja meg a kamera működését emulátoron, mi a tapasztalat?
b) Próbáljon meg hívást indítani és SMS-t küldeni. Mik a tapasztalatok?
c) Mire szolgál a Dev Tools alkalmazás?

## 6.4 Ismerkedés az Eclipse-el

a) Indítsa el az Eclipse-t és a laborvezető segítségével tekintse át a fő funkciókat.
b) Tekintse át, hol lehet frissíteni az ADB plugint!
c) Indítsa el az SDK Managert Eclipse alól!
d) Hozzon létre egy HelloWorld Android projektet!
e) Próbáljon ki legalább 5 hotkey kombinációt, ami megkönnyíti Eclipse alatt a munkát!
    - Ctrl+click: navigálás a kódban
    - Ctrl+space: intelli sense és „kódgenerálás” ha egy felüldefiniálandó metódus első pár betűjét ütjük le, majd Enter-t nyomunk
    - **Alt+Shift+R**: változó átnevezése (mindenhol)
    - **Ctrl+Shift+T:** típus keresése
    - F3: deklaráció megnyitása függvényen, osztályon
    - Ctrl+Alt+H: hívás hierarchia megjelenítése
    - **Ctrl+Shift+O**: import fix

- **Ctrl+Shift+F**: forrás formázása
- Ctrl+F6: forrásfájlok közti váltás
- **Ctrl+Shift+G**: használat listázása
- Ctrl+Shift+L: hotkey lista
- Ctrl+PageUP/PageDown: Navigálás a megnyitott állományok között
- Alt+Bal/Jobb: Ugrás a kód korábban használt szakaszára
- Ctrl+1: Eclipse automatikus javaslat az adott helyen
- Ctrl+Shift+R: erőforrás keresés
- Alt+Shift+A, majd S: Szöveges konstans kihelyezése a szöveges értékeket tartalmazó Android strings.xml állományba
- Ctrl+Shit+[x/y]: kis-nagybetűvé alakítás

f) Indítsa el a HelloWorld Android projektet emulátoron!

## 6.5 DDMS nézet

a) Tekintse át a DDMS nézet funkcióit a laborvezető segítségével!
b) Hajtson végre egy telefonhívást a DDMS nézetről!
c) Küldjön egy SMS-t a DDMS nézetről!
d) Változtassa a készülék pozícióját a DDMS nézeten!
e) Vizsgálja meg az elindított HelloWorld projekt nyitott szálait, memóriafoglalását!
f) Vizsgálja meg a LogCat nézet tartalmát!

## 6.6 APK állomány tartalma

a) Keresse ki a létrehozott HelloWorld projekt mappáját és a *bin* könyvtáron belül vizsgálja meg az .apk állomány tartalmát.
b) Hol található a lefordított kód?

## 6.7 Több Activity-ből álló alkalmazás

A feladat egy több Activity-ből álló Android alkalmazás elkészítése. Az Activity-k között a navigációt menü segítségével és *TextView*-ra való kattintással kell megoldani.

### 6.7.1 Az első Android Activity

Készítsen egy Activity-ből álló alkalmazást, amely tartalmazzon egy „Show Activity2” feliratú *TextView*-t és egy „Exit” menüpontot. A „Show Activity 2” *TextView*-ra kattintva az alkalmazás jelenítsen meg egy tetszőleges szöveget egy felugró *Toast* (pop-up) ablakban! Az Exit menüpont hatására lépjen ki az alkalmazás. A kilépést a *finish()* függvényhívással oldja meg az Exit menüpont választásakor.

Az Activity XML-ben megadott felhasználói felülete *(/res/layout/activity_main.xml*):

```
<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" >
```

```
    <TextView
        android:id="@+id/tvShowSecond"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/showSecond"
        tools:context=".MainActivity" />

</LinearLayout>
```

Az Activity XML-ben megadott menüje *(/res/menu/activity_main.xml)*:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_exit" android:title="@string/menu_exit"/>
</menu>
```

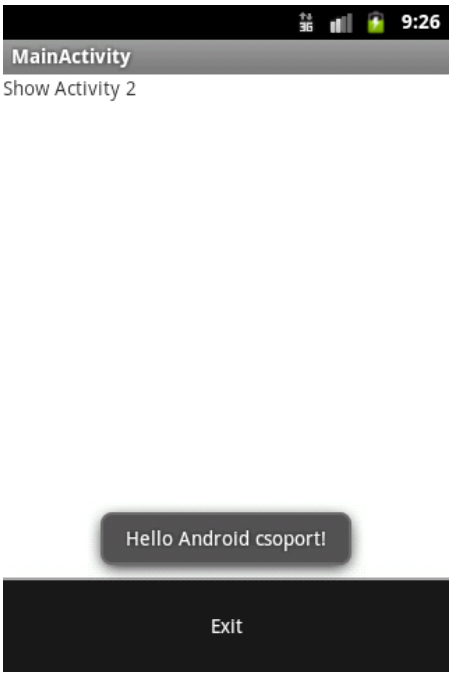
Figyeljük meg, hogy az XML állomány neve ugyanaz, mint a felület esetén, azonban mivel a *menu* könyvtár alatt található, nem fog gondot jelenteni az azonosítás.
Az Activity XML-ben megadott szöveges erőforrása *(/res/values/strings.xml)*:

```
<resources>
    <string name="app_name">Labor1</string>
    <string name="hello_world">Hello world!</string>
    <string name="title_activity_main">MainActivity</string>
    <string name="showSecond">Show Activity 2</string>
    <string name="menu_exit">Exit</string>
</resources>
```

Végül az Activity forráskódja:

```
public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TextView tvShow2 = (TextView)findViewById(R.id.tvShowSecond);
        tvShow2.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
              Toast.makeText(MainActivity.this,
              "Hello Android csoport!", Toast.LENGTH_LONG).show();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case R.id.menu_exit:
            finish();
            break;
        }
        return true;
    }
}
```

Az alkalmazás felhasználói felületét a következő ábra szemlélteti:



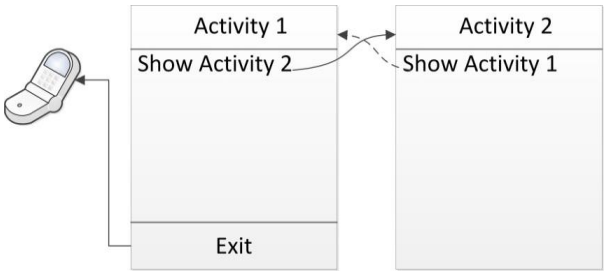

**Az első Android Activity**

*További feladatok:*

- Keresse ki és vizsgálja meg a *Manifest* állományt, hol van az Activity jelölve?
- Keresse ki és vizsgálja meg a */gen/[package]/R.java* állományt, mi célt szolgál?
- Készítsen paraméterezhető szöveges erőforrást és próbálja ki!
    - strings.xml-be:
        - <string name="txtPageViews">%1$d oldal %2$s szerző</string>
    - Java kódba:
        - int pvCount = 2;
        - String author = "John Doe";
        - String pv= getString(R.string.txtPageViews, pvCount, author);

### 6.7.2 Activity-k közti navigáció

Bővítse az alkalmazást, újabb Activity-vel. Az első Activity-n található „Show Activity 2" *TextView*-t módosítsa úgy, hogy rá kattintva jelenjen meg a második Activity. A második Activity-n egyetlen „Show Activity1" *TextView* szerepeljen, melyre kattintva az első Activity-re kerül a vezérlés, úgy hogy a második Activity befejeződik.




*Segítség:*

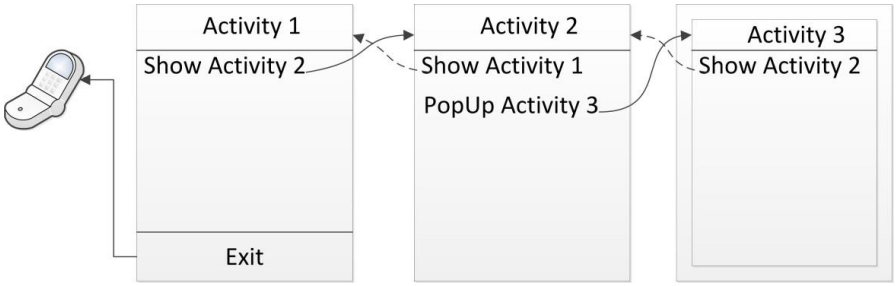
- Ne felejtse a Manifest állományba is definiálni az új Activity-t!
- Új Activity indítás:

```
Intent myIntent = new Intent();
myIntent.setClass(MainActivity.this, SecondActivity.class);
startActivity(myIntent);
```

- A második Activity-ből az elsőbe való visszalépéshez használja a *finish()* függvényt!

### 6.7.3 Activity megjelenítése felugró ablakban

Adjon hozzá az előző feladatban létrehozott alkalmazáshoz egy harmadik Activity-t, amely dialógus formában fog megjelenni. A második Activity felületére helyezzen el egy „Show Activity3” *TextView*-t, melyre kattintva jelenítse meg a harmadik Activity-t felugró ablakban. A harmadik Activity-n szerepeljen egy „Show Activity2” *TextView*, melyre kattintva a második Activity jelenik meg, úgy, hogy a harmadik Activity befejeződik.




Vegyük észre, hogy tulajdonképpen a vissza gomb eseményeit szimuláltuk!

*Segítség:*

- A második Activity Layout-jának XML kódja (figyeljük meg a *LinearLayout android:orientation="vertical"* tulajdonságát):

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical">
<TextView
     android:id="@+id/show1from2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:text="@string/show1"
    />
<TextView
     android:id="@+id/show3from2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/show3"
    />
</LinearLayout>
```

- A harmadik Activity dialógus formában történő megjelenítéséhez *Manifest*-ben történő beállítás szükséges:

```
<activity android:name=".ActivityThird"
              android:label="@string/app_name"
              android:theme="@android:style/Theme.Dialog">
</activity>
```

- A harmadik Activity-ben a másodikhoz való visszatéréshez használja a *finish()* függvényt!

### 6.7.4 Validáció alapok

Helyezzen el a második Activity-re egy *EditText*-et és valósítsa meg, hogy addig ne lehessen tovább menni a 3. Activity-re, amíg az *EditText* üres. Jelezze a felhasználónak, hogy az *EditText* tartalmát kötelező kitölteni az Android beépített hibajelző funkciójával (setError(…)):

```
EditText et = (EditTest)findViewById(R.id.edtPersonalID);
…
et.setError(R.string.err_personal_id_not_empty);
```

### 6.7.5 Életciklus függvények nyomon követése

Helyezzen az egyes Activity-k életciklus függvényeibe *LogCat* hívásokat egy saját Tag-el és elemezze az Activity-k életciklussait az Eclipse *LogCat* nézeten úgy, hogy szűrőt hoz létre a saját Tag-jére.

```
public static final String MYTAG = "DEMOTAG";
…
Log.d(MYTAG,"onCreate() meghívódott");
```

### 6.7.6 Bonus feladat: Alkalmazás ikon beállítása

Cserélje le az alkalmazás ikonját az alábbi oldalon található ikonra (**a későbbi laborokhoz ugyanitt megtalálja a megfelelő sorszámú ikonokat**):
http://avalon.aut.bme.hu/~tyrael/android/icons/app_icon2.png