Programmer Puzzle

|  |  |
| --- | --- |
| Témavezető: Nagy Ádám | Szerző: Kiss Dávid |
| tanársegéd | Programtervező informatikus BSc. |

# Szakdolgozat-téma bejelentő

Tartalomjegyzék

[Szakdolgozat-téma bejelentő 2](#_Toc530687488)

[Bevezető 4](#_Toc530687489)

[Felhasználói dokumentáció 5](#_Toc530687490)

[Fejlesztői dokumentáció 6](#_Toc530687491)

[Android ismertető 6](#_Toc530687492)

[Forráskód 6](#_Toc530687493)

[Erőforrások 6](#_Toc530687494)

[Komponensek 6](#_Toc530687495)

[Activity-k 7](#_Toc530687496)

[Szoftver architektúra 7](#_Toc530687497)

[Adatbázis 7](#_Toc530687498)

[ORM 7](#_Toc530687499)

[GreenDAO 7](#_Toc530687500)

# Bevezető

Szakdolgozatom témája egy, kifejezetten programozók számára készült játék Android operációs rendszeren. Az ötletet egy állásinterjút megelőző online tesztből merítettem, melyben különböző feladatokat kellett megoldani Java nyelven. A böngészőben felhasználóbarát módon megvalósított felületen kattintással vagy egérrel való húzással lehetett a sorokra tördelt forráskódot összerakni. Segítségképpen szerepelt feladatleírás, és több input-output pár. Ez a fajta szintfelmérés elnyerte a tetszésemet, és szerettem volna hasonlót megvalósítani egy más platformon, a saját ötleteimmel vegyítve. Így született meg a Programmer Puzzle telefonos alkalmazás, melyben 3-féle programozási nyelven lehet játszani (C++, Java, Python), összesen 41 különböző feladattal. A játék célja a sorokra felbontott forráskódot összerakni úgy, hogy a leírt feladatot megvalósítsa. Segítségképpen szerepelnek input-output párok, néhány algoritmus esetén pszeudokód is, valamint lehetőség van feladatonként egyszer segítséget kérni. Egységesen 12 perc időlimit van egy feladatra, és a játék végeztével pontszámot kap a játékos. Minél gyorsabban oldotta meg a feladatokat, annál több pont jár, amit az app elment a toplistapontok közé. Egy játék 5 feladatból áll, és ha valamelyik túl nehéz a játékos számára, akkor léphet a következőre. Csak 100%-ban megoldott feladatokra jár pont. Az alkalmazás teljesen angol, ennek több oka is van, egyrészt nagyobb célközönséget tudok vele elérni, másrészt segítheti a szakmai nyelvtudását a játékosnak, amelyre a legtöbb munkahelyen nagy szükség van.

# Felhasználói dokumentáció

# Fejlesztői dokumentáció

## Android ismertető

### Forráskód

Az Android alkalmazások esetében a felhasználói felület és a program forráskódja teljesen elkülönül. A felhasználói felület kialakítására XML állományokat használunk, és bár lehetséges a felhasználói felületek definiálása a forráskódon belül is, azonban célszerűbb ezt kerülni. A felhasználó felület elemei erőforrásként fognak megjelenni a programunkban. A forráskódok az src, míg az erőforrások a res alkönyvtárban elérhetőek. A kettő közötti kapcsolatot az R.java állomány fogja megteremteni, azaz ennek a segítségével tudjuk a forráskódunkból elérni a rendelkezésre álló erőforrásainkat. [Android könyv]

### Erőforrások

Az egyes alkalmazások által használt erőforrások jól különválnak az őket kezelő forráskódoktól, ily módon az üzleti logikát megvalósító kód és a felhasználói felület jól körülírt és egymástól elválasztott maradhat. [Android könyv]

Erőforrások típusai:

* Képek
* Hanganyagok
* Animációk
* Szöveges állományok
* Stílusok
* Színek
* Az Activity-khez tartozó layout XML fájlok

És még sok más. Ebben a projektben például az erőforrások között szerepel a beállítások menüpont komponenseinek XML fájlja is.

Erőforrásainkra a következő módon tudunk hivatkozni:



Mivel elég sok objektumra hivatkozok a programban, ezeket a hívásokat kikerültem a ButterKnife tool segítségével, így elkülönülve szerepelnek a view bind-olások a programkódban. Példa:



### Komponensek

Minden Android alkalmazás egy, vagy több komponensből épül fel. Az alkalmazás ugyanabból a komponensből is tartalmazhat többet. A komponensek különböző belépési pontokat biztosítanak a programunk felé a rendszer számára. Az Android programokban négy komponens létezik, amelyek mindegyike különböző célt szolgál. Életciklusuk is eltér egymástól, amelyek létrehozzák, vagy megsemmisítik őket. Egy új komponens indítása előtt a rendszer megnézi, fut-e már a komponenst tartalmazó alkalmazás egy processze. Ha igen, alapesetben ehhez a processzhez rendeli a létrehozandó komponenst. Ha nem, akkor elindítja az alkalmazást, majd példányosítja a komponenshez szükséges osztályokat. A négyféle komponensből egy lesz igazán érdekes, a többi jelen esetünkben nem kap hosszabb magyarázatot.

Ezek tehát:

* Activity
* Service
* Content provider
* Broadcast receiver

### Activity-k

Ez egy felhasználói felülettel rendelkező képe a programnak. Egy alkalmazás több Activityt is tartalmazhat, amelyek együtt biztosítják a program valamennyi szolgáltatását. Egy naptáralkalmazás például tartalmazhat olyan Activityt, amely havi bontásban mutatja a napokat, egy másikat, ami heti, vagy napi listát valósít meg. Ugyanakkor egy harmadik Activity szolgálhat új naptári események rögzítésére. Az Activity ablaka leggyakrabban teljes képernyő méretű, de más módokon is megjelenhet a felhasználó számára: van lehetőség úszó ablakban, vagy az ActivityGroup segítségével beágyazott ablakban megjeleníteni Activityt. [Android könyv]

Intent-ek

A platformon az alkalmazások komponensei közötti adatcsere eszköze, egy-egy esemény egzakt leírására szolgál. Különlegessége, hogy nemcsak egyazon alkalmazás komponensei, de különböző programok komponensei között is megvalósíthat kommunikációt. Sosem közvetlenül, hanem az operációs rendszeren keresztül történik a kézbesítése.[Android könyv] Intent-ek segítségével lehet adott Activity-ből másik Activity-t elindítani, illetve ha szükséges, extra információt küldeni az új Activity-nek.

Példa:



## Szoftver architektúra

Az alkalmazás felépítési szerkezete leginkább az MVP (model-view-presenter, magyarul modell-nézet-prezenter) architektúrának felel meg, de a játék jellege megkövetelte, hogy ettől helyenként eltérjek. Például a NewGameActivity tartalmaz logikát is, a nézeti szerepén kívül. Törekedtem arra, hogy a logikailag elkülöníthető részek más osztályokban szerepeljenek. Ennek alapján a nézetnek az Activity osztályok, a modellnek a Persistence package osztályai, a prezenternek a GameLogic package felel meg.

## Adatbázis

### ORM

Mielőtt belemennék a perzisztens adatok részletezésébe, kitérek az objektum-relációs leképzés (**O**bject-**R**elational **M**apping) fogalmára. Az ORM egy programozási technika adatok konvertálására nem kompatibilis típusos rendszerek és objektumorientált programozási nyelvek között.[Wikipédia ORM] Egy ORM könyvtár adott programozási nyelven enkapszulálja a kódot, mellyel manipuláljuk az adatokat, így nincs szükség tiszta SQL írására. A programkódban közvetlenül az entitás objektumainkkal tudunk dolgozni. Modernebb ORM-ek esetén ez annyit jelent, hogy egy egyszerű annotációval (pl. @Entity) legeneráltatjuk az osztályhoz tartozó táblákat, és azután tudjuk beszédes beépített metódusokkal lekérni, frissíteni, beszúrni, törölni az adatokat.

### GreenDAO