



Eötvös Loránd Tudományegyetem

Informatikai Kar

Komputeralgebra tanszék

Programmer Puzzle

Témavezető: Nagy Ádám
tanársegéd

Szerző: Kiss Dávid
Programtervező informatikus
BSc.

Budapest, 2018

Tartalomjegyzék

Bevezető.....	4
Felhasználói dokumentáció.....	5
Program rövid leírása	5
Telepítés	5
Rendszerkövetelmények	6
Alkalmazás funkciói	7
Kezdőképernyő	7
Új játék	8
Segítség gomb	9
Kihagyás gomb.....	11
How to play.....	12
Toplista	13
Beállítások	14
Fejlesztői dokumentáció	16
Követelményleírás	16
Specifikáció.....	16
Alapvető funkciók	17
Android ismertető	17
Forráskód	17
Erőforrások	18
Komponensek.....	18
Activity-k	19
Intent-ek.....	19
Szoftver architektúra	20
Adatbázis	20
ORM	20
GreenDAO	20
Entitások	21
Generált osztályok	22
Adatmenedzsment osztályok	23
Game Logic	26
Application	27
Nézet	28

Activity-k	29
Utility osztályok.....	40
XML fájlok.....	42
Mipmap.....	42
Raw.....	42
Values	42
Preference	43
Layout.....	43
Tesztelés	45
Unit tesztek	45
Instrumented tesztek.....	46
Manuális tesztek.....	47
Irodalomjegyzék.....	50

Bevezető

Szakedolgozatom 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 egerrel 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

Program rövid leírása

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.

A beállításokban 3 különböző programozási nyelvből lehet választani: C++, Java, Python, így a legtöbben megtalálhatják a kedvükre valót. Az alkalmazás célja az algoritmikus gondolkodás és a kreativitás fejlesztése, ebből kifolyólag nem csak egy helyes megoldás létezik a feladatokban. A játék teljesen angol, a feladatszövegek is így szerepelnek.

A játék működése:

Új játék elindításakor kap a játékos egy feladtleírást, alatta maximum 20 sornyi forráskódot sorokra felbontva random sorrendben. Minden sornyi kód egy mozgatható gombon szerepel, melyet megérintve lehet bemozgatni az alsó panelra, a következő üres sorba. Ugyanígy, ha egy gombot az alsó panelen megérint a játékos, az visszamegy az eredeti helyére felülre. Segítségképpen néhány feladatban előre be van mozgatva pár gomb. Ha a játékos elakad, tud segítséget kérni a megfelelő gombbal, illetve tovább lépni a következő feladatra. 5 feladatból áll egy játék, ha mindet befejezte vagy átlépte a játékos, a végén kiírja a program a szerzett pontszámot.

Telepítés

Az alkalmazás a letöltött apk-t futtatva telepíthető. A telepítéshez engedélyezni kell a telefon beállításában az ismeretlen forrásokból való telepítést. A szükséges engedélyek megadása után az apk fájl kiválasztásával a Telepítés lehetőségre kattintva automatikusan elindul a telepítés és az alkalmazás az eszköz kapacitásától függően néhány másodperc alatt kész a használatra.

Rendszerkövetelmények

Az alkalmazás a következő Android verziókon futtatható:

- Android 5.1 - Lollipop - API level 22
- Android 6.0 - Marshmallow - API level 23
- Android 7.0 - Nougat - API level 24
- Android 7.1 - Nougat - API level 25
- Android 8.0.0 - Oreo - API level 26
- Android 8.1.0 - Oreo - API level 27
- Android 9 - Pie - API level 28

És a későbbi verziók.

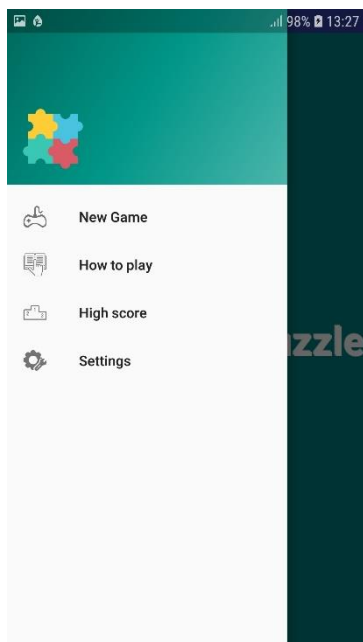
Alkalmazás funkciói

Kezdőképernyő

Az alkalmazás elindítása után a következő fogadja a felhasználót:



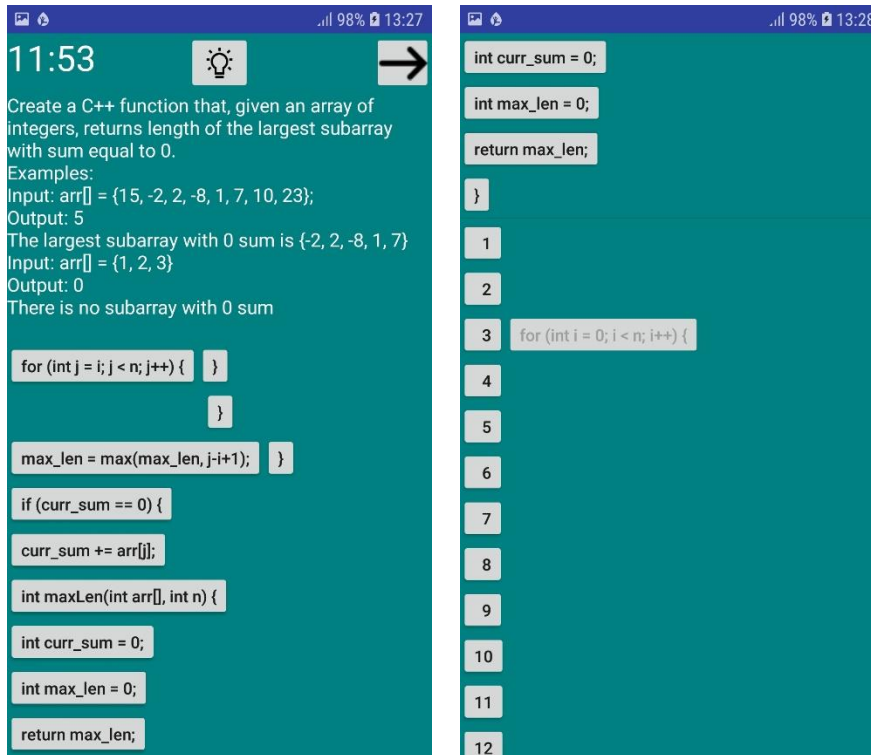
A bal felső sarokban levő gombra kattintva, vagy a képernyő bal oldaláról jobbra húzva előhozható a főmenü.



Új játék

A New Game menüpontra lépve elindul egy új játék a jelenlegi beállításokkal. Egy játék 5 puzzle-ből áll, mindegyikre 12 percnyi idő van.

Így néz ki egy C++ nyelvű puzzle:

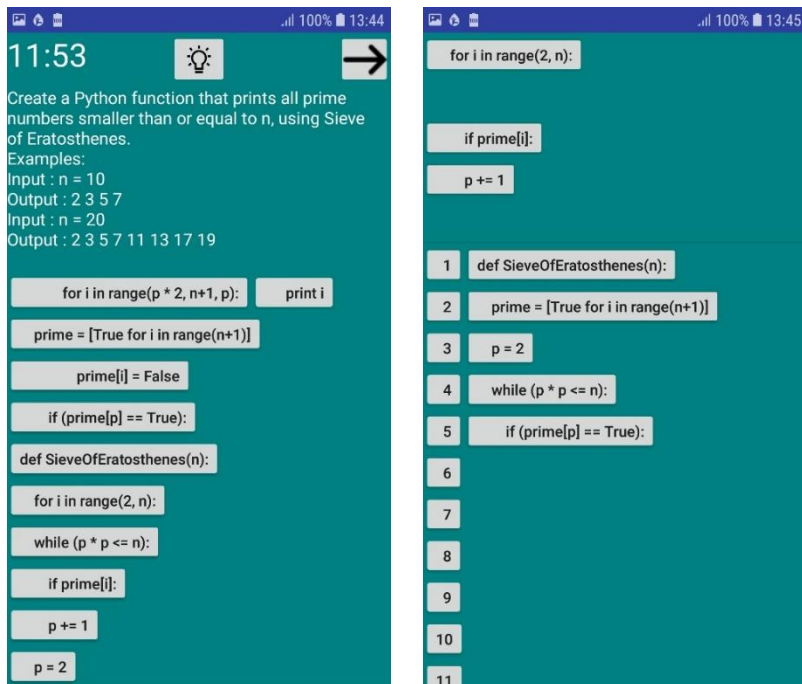


A bal oldali képen, bal felső sarokban látható az időzítő, felül középen segítség gomb, felül jobbra a kihagyás gomb. Ezek alatt szerepel az adott puzzle leírása. A feladatléírás alatt a mozgatható gombok vannak. A jobb oldali képen ugyanaz a puzzle szerepel, mindössze lejjebb lett csúsztatva a képernyő. A számozott gombok 1-től 20-ig reprezentálják a 20 sornyi területet, ahová lehet mozgatni a gombokat. A feladatok kiterjedése miatt egyik puzzle sem fér ki egy képernyőnyi területre, a legtöbb telefonon. Ennek a javításaképpen két oszlopban szerepelnek a mozgatható gombok a felső panelen. Ebben a példában 12 mozgatható gombunk van, és a csukó zárójelek külön gombokon szerepelnek, mivel az összerakott kódban is külön sorokban lesznek. Mint a jobb oldali képen látszik, egy gomb már be lett mozgatva a 3.sorba, és szürke a rajta levő szöveg. Ez azt jelenti, hogy az a sor segítségképpen be lett égetve a puzzle-ba, így azt mozgatni nem kell, és nem is lehet. Annak a gombnak a helye látszik is a bal oldali képen, így összesen 13 soros programkódot kell összeraknia a játékosnak.

A beállításokban választható C++, Java és Python nyelv, a feladatmegoldás szempontjából a C++ és Java hasonlítanak annyiban, hogy mindkettőben fel kell használni a csukó zárójeleket,

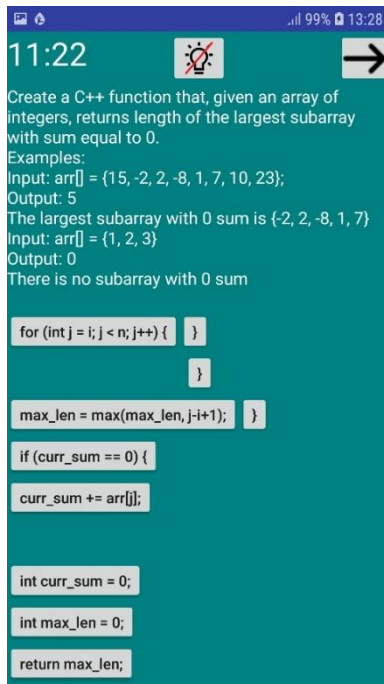
és a nyitó / csukó zárójelek függvényében változik a tabulálás a kirakott puzzle-ben. A Python nyelvű puzzle-k esetén mivel nincsenek blokk végét jelző karakterek, ezért a tabulálás megmarad a mozgatható gombokon, emiatt talán könnyebb kirakni őket, mert lehet a tabulálásból következtetni, hogy mely gombok lehetnek a következő sorban.

Példa:



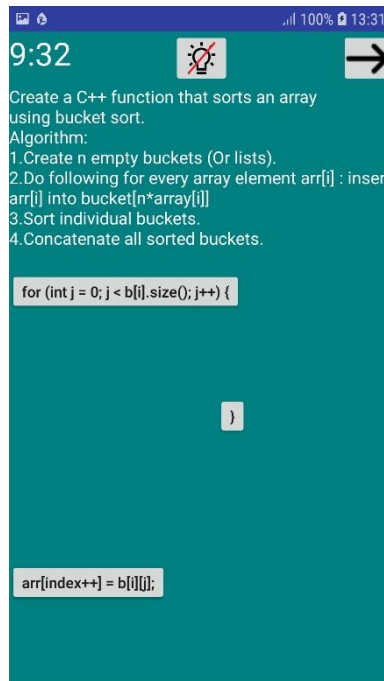
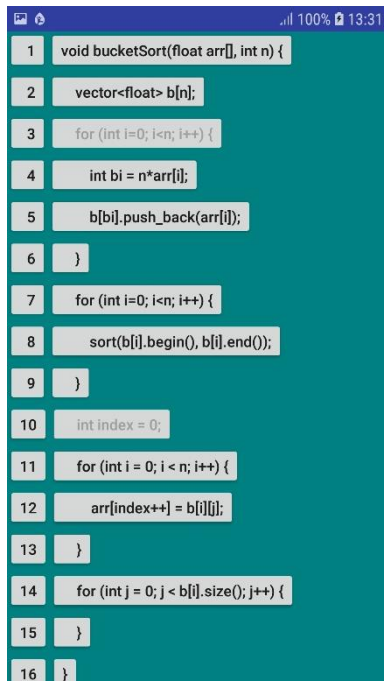
Segítség gomb

A fent középen levő gombot lehet egy puzzle-ban egyszer felhasználni segítségnek. Működése kétféleképpen mehet végbe, az egyik:



A program ellenőrzi, hogy az eddig felhasznált gombok helyes sorrendben vannak-e az alsó panelen, ha igen, akkor berakja a következő szabad helyre a helyes sort.

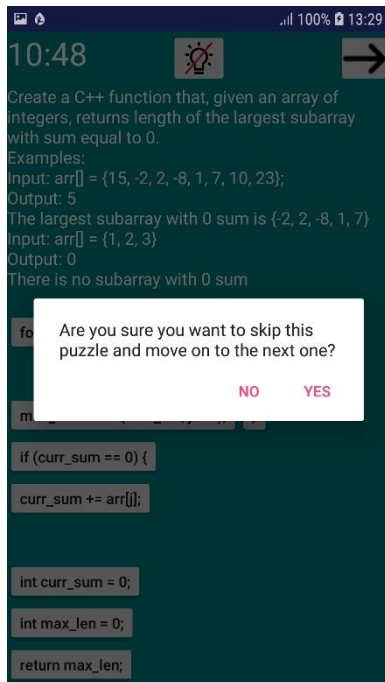
Ha viszont van hiba az eddig kirakott puzzle-ban, akkor az összes hibás sort visszateszi a felső panelre.



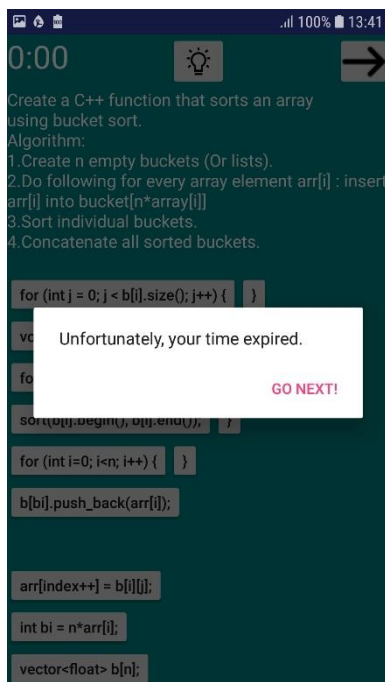
A bal oldali képen látható egy kirakott puzzle, amiben szerepelnek hibák. A jobb oldali képen a segítség gomb megnyomását követő állapot szerepel. A 3 rossz helyen szereplő sor visszakerült a felső panelre.

Kihagyás gomb

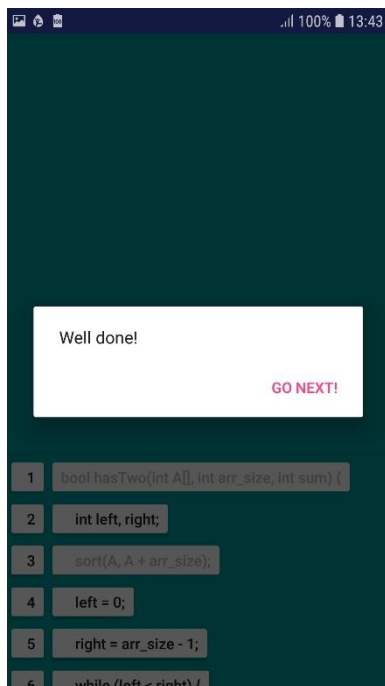
Ha egy puzzle-nál elakadt a játékos, lehetőség van ugrani a következőre a jobb felső sarokban levő gomb segítségével. Megnyomása után megkérdezi a játék, hogy biztos tovább akar-e lépni a játékos.



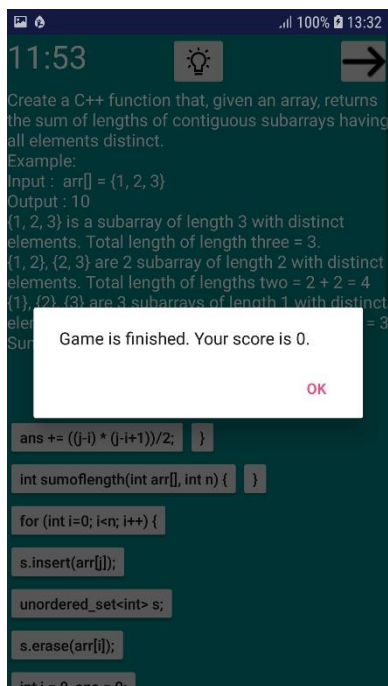
Ha lejár az idő, a játék jelzi ezt, és a következő puzzle-ra lép a Go next! gomb megnyomása után.



Amennyiben sikerült kirakni egy feladatot, a következő képernyő lesz látható:




Ha mind az 5 puzzle véget ért, a játék kiírja a játékos pontszámát, majd visszaléphet a főmenübe az OK gomb megnyomásával.




How to play

A második menüpontra lépve kap a játékos egy leírást a játék lényegéről, és magyarázatot a gombok működéséről.


The goal of every puzzle is to move every singular button provided at the top, to the correct lines provided at the bottom of the screen. You are provided with a short description of the puzzle, it is advised to read carefully. You can move the buttons back and forth by tapping on them. Tapping on a button will move it to the next available line. Tapping on a button if it has already been moved to a line will move it back to it's original position at the top. Some buttons have fixed positions when starting a puzzle, those have grey texts in them. Many of the puzzles are quite hard, but there is plenty of time to think, and if you need help, use this:




This button will do one of two things. If the buttons you already moved are correct, it will move a single button to the next line, which will be guaranteed to be at the correct position. If your work so far is incorrect however, it will move back every button that is in incorrect position. Be careful, this option is available once per puzzle. If the puzzle is too hard or uninteresting, you can move to the next one by using the following:



back and forth by tapping on them. Tapping on a button will move it to the next available line. Tapping on a button if it has already been moved to a line will move it back to it's original position at the top. Some buttons have fixed positions when starting a puzzle, those have grey texts in them. Many of the puzzles are quite hard, but there is plenty of time to think, and if you need help, use this:



This button will do one of two things. If the buttons you already moved are correct, it will move a single button to the next line, which will be guaranteed to be at the correct position. If your work so far is incorrect however, it will move back every button that is in incorrect position. Be careful, this option is available once per puzzle. If the puzzle is too hard or uninteresting, you can move to the next one by using the following:



This button will load the next puzzle for you. There are 5 puzzles in a single game, at the end you will receive a score based on how well you did. Only 100% completed puzzles give score. The programming language for the puzzles can be changed in the settings.

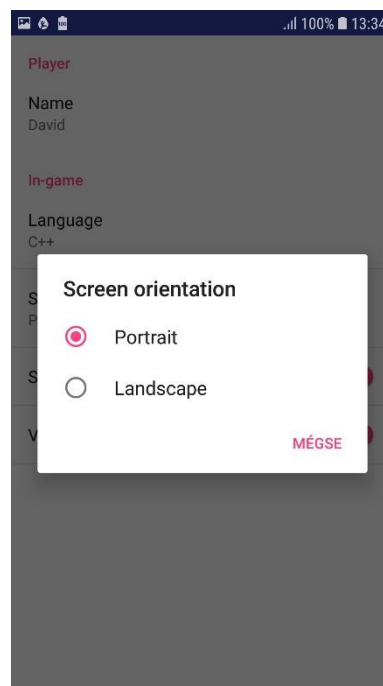
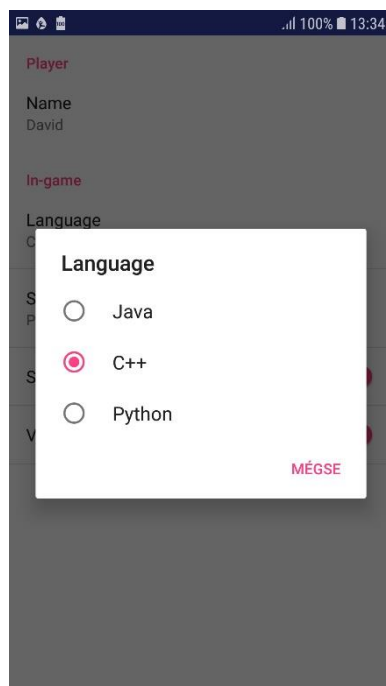
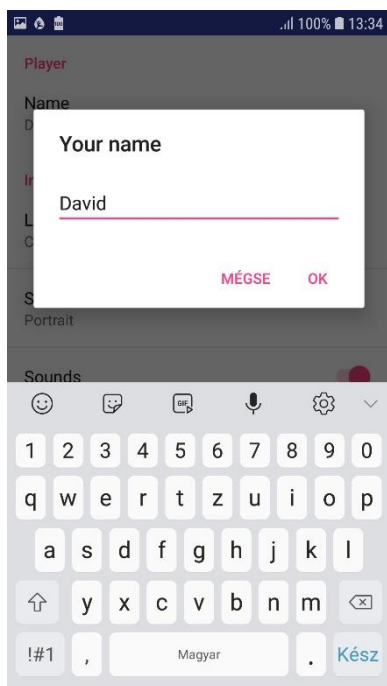
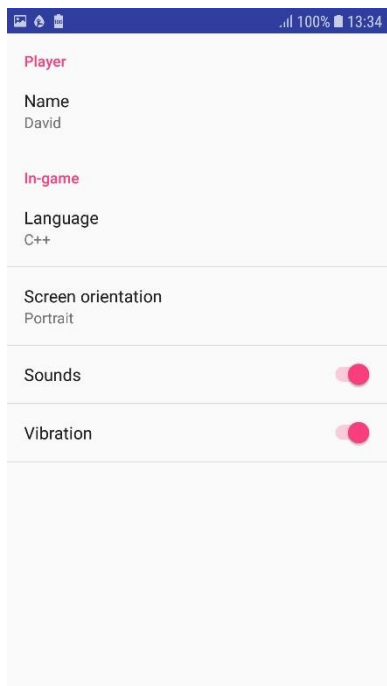
Toplista

A harmadik menüpontra lépve tekinthető meg a top 10 legtöbb pontot szerző játékos.

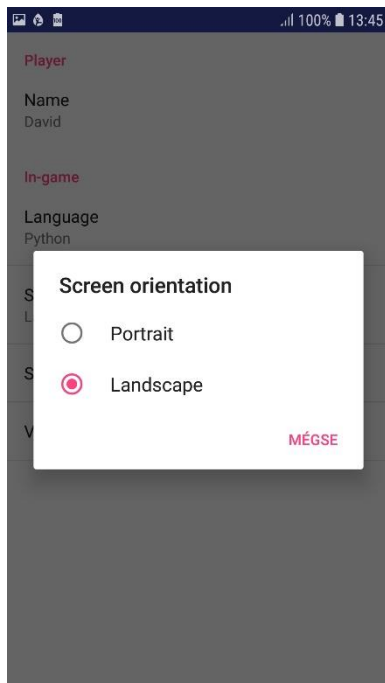
Player	Score
1. David	832
2. David	627
3. David	591
4. David	255
5. David	0
6. David	0
7. David	0
8. David	0
9. David	0
10. David	0

Beállítások

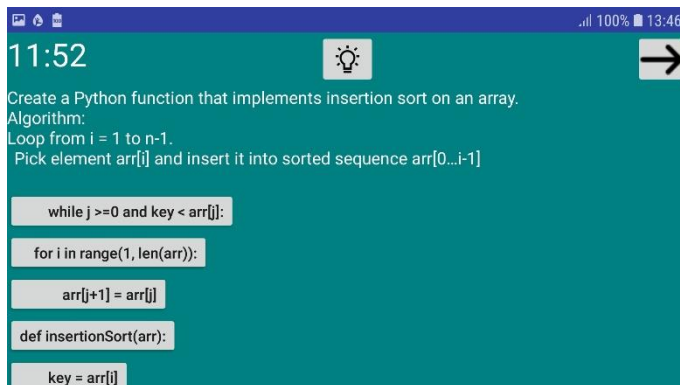
A negyedik menüpontra kattintva a beállításokon változtathat a felhasználó. Beállíthatja a nevét, a puzzle-k programozási nyelvét, képernyő-orientációt, hangok és rezgések engedélyezését.



Választható fekvő orientáció a játékban, ez akkor javasolt, ha nem férnek ki a képernyőre a mozgatható gombok egy játékban. Elsősorban álló képernyőre lett tervezve a játék, több készüléken is lett tesztelve, és nem volt egyikkel sem probléma, de előfordulhat, hogy a fekvő képernyő jobban megfelel.



Ekkor így néz ki a játék:



Fejlesztői dokumentáció

Követelményleírás

Specifikáció

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.

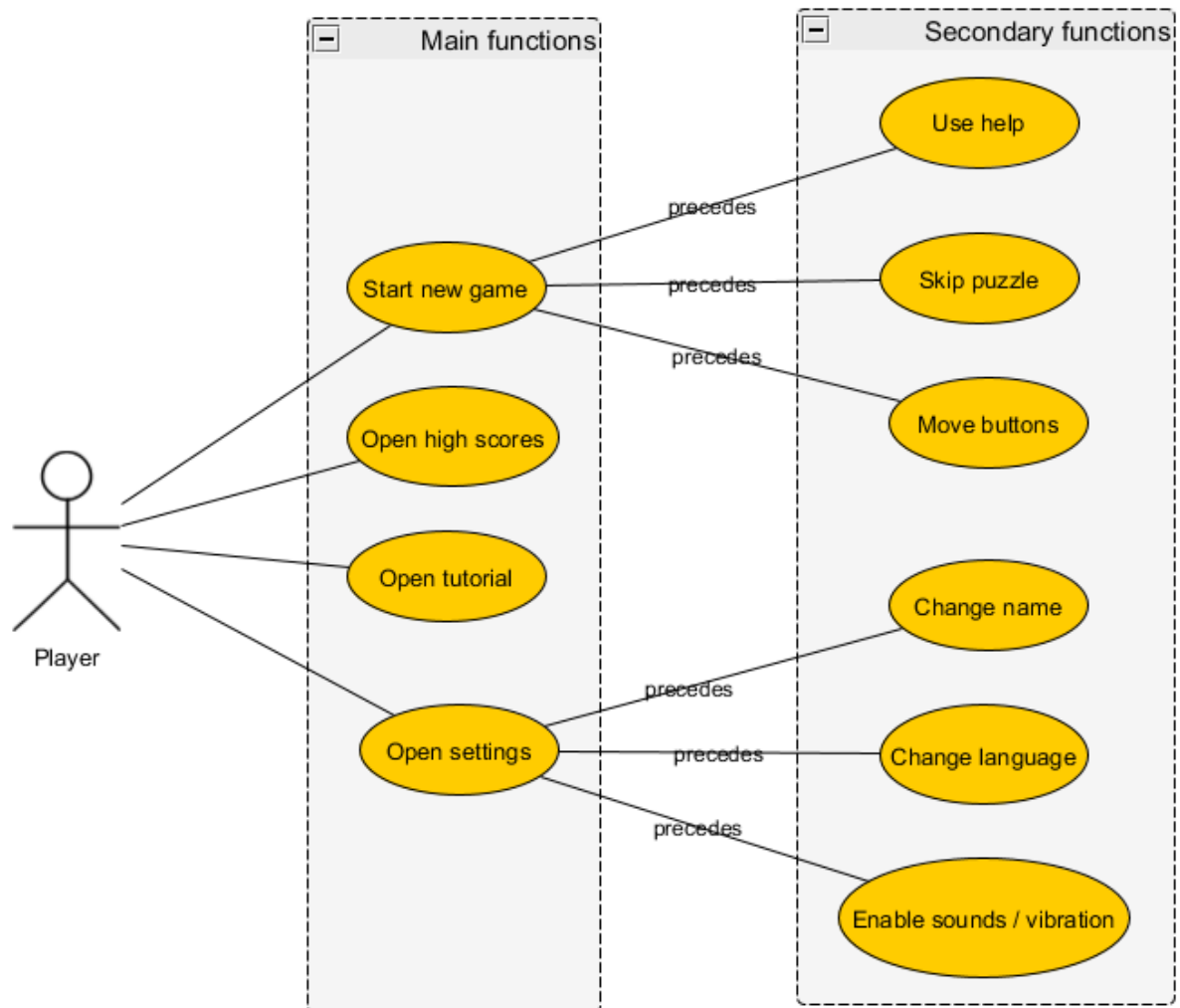
A beállításokban 3 különböző programozási nyelvből lehet választani: C++, Java, Python, így a legtöbben megtalálhatják a kedvükre valót. Az alkalmazás célja az algoritmikus gondolkodás és a kreativitás fejlesztése, ebből kifolyólag nem csak egy helyes megoldás létezik a feladatokban. A játék teljesen angol, a feladatszövegek is így szerepelnek.

A játék működése:

Új játék elindításakor kap a játékos egy feladtleírást, alatta maximum 20 sornyi forráskódot sorokra felbontva random sorrendben. Minden sornyi kód egy mozgatható gombon szerepel, melyet megérintve lehet bemozgatni az alsó panelra, a következő üres sorba. Ugyanígy, ha egy gombot az alsó panelen megérint a játékos, az visszamegy az eredeti helyére felülre. Segítségképpen néhány feladatban előre be van mozgatva pár gomb. Ha a játékos elakad, tud segítséget kérni a megfelelő gombbal, illetve tovább lépni a következő feladatra. 5 feladatból áll egy játék, ha mindet befejezte vagy átlépte a játékos, a végén kiírja a program a szerzett pontszámot.

Alapvető funkciók

A program alapvető funkcióit az alábbi Use Case diagram mutatja.



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.” [1]

Erőforrások

„Az egyes alkalmazások által használt erőforrások jól különválnak az őket kezelő forráskódotól, így módon az üzleti logikát megvalósító kód és a felhasználói felület jól körülírt és egymástól elválasztott maradhat.” [2]

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:

```
Button okButton = (Button) findViewById(R.id.okButton);
```

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:

```
@BindView(R.id.newGameActivity_Timer) TextView timer;  
@BindView(R.id.newGameActivity_Hint) ImageButton hintButton;  
@BindView(R.id.newGameActivity_Skip) ImageButton skipButton;
```

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,

alap esetben 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.” [3]

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.” [4] Intent-ek segítségével lehet adott Activity-ből másik Activity-t elindítani, illetve ha szükséges, extra információt küldeni az új Activity-nek.

Példa:

```
Intent newGameIntent = new Intent( packageContext MainActivity.this, NewGameActivity.class);
newGameIntent.putExtra( name: "language", language);
startActivity(newGameIntent);
```

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 leképezés (**Object-Relational Mapping**) fogalmára. „Az ORM egy programozási technika adatok konvertálására nem kompatibilis típusos rendszerek és objektumorientált programozási nyelvek között.” [5] 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

A greenDAO egy nyílt forráskódú Android ORM, SQLite beépített relációs adatbázis alappal. „Néhány funkciója:

- Könnyen használható
- Minimális memóriefelhasználás
- Kis könyvtárméret (<100 kB)
- Adatbázis titkosítás
- Egyik leggyorsabb Android ORM”

[6]

A projektben ezt a könyvtárat használtam fel segítségképpen az adatok könnyebb modellezésére. Bár számomra teljesen új dolog volt, viszonylag könnyedén megismerkedtem vele, és egyszerű volt a használata, már csak azért is, mert az entitásaim nem igényeltek bonyolult műveleteket. A továbbiakban részletezem a perzisztenciához tartozó osztályokat.

Entitások

Puzzle.java

Az osztály előtt `@Entity` annotáció jelzi a greenDAO-nak, hogy ez egy entitás, és generáljon hozzá táblát és DAO osztályt. A játékban szereplő puzzle-ök osztálya.

Adattagok:

private Long id	automatikusan generált ID a puzzle-öknek
private String description	szöveges feladatleírás
private String code	a puzzle teljes programkódja
private String language	a puzzle programozási nyelve

Metódusok:

public Long getId()	visszatér a puzzle ID-jével
public void setId(Long id)	beállítja a puzzle ID-jét a paraméterben kapott értékre
public String getDescription()	visszatér a puzzle feladatleírásával
public void setDescription(String description)	beállítja a puzzle feladatleírását a paraméterben kapott értékre
public String getCode()	visszatér a puzzle programkódjával
public void setCode(String code)	beállítja a puzzle programkódját a paraméterben kapott értékre
public String getLanguage()	visszatér a puzzle programozási nyelvével
public void setLanguage(String language)	beállítja a puzzle programozási nyelvét a paraméterben kapott értékre

HighScore.java

A toplistán szereplő pontok osztálya. Megvalósítja a `Comparable` interface-t, hogy sorrendet lehessen felállítani a toplistapontok között.

Adattagok:

private Long id	automatikusan generált ID a toplistapontoknak
private String player	a pontot szerző játékos neve
private long points	játékos pontszáma

Metódusok:

public Long getId()	visszatér a toplistapont ID-jével
public void setId(Long id)	beállítja a toplistapont ID-jét a paraméterben kapott értékre
public String getPlayer()	visszatér a toplistaponthoz tartozó játékos nevével
public void setPlayer(String player)	beállítja a toplistapont játékosának nevét a paraméterben kapott értékre
public long getPoints()	visszatér a szerzett pontszámmal
public void setPoints(long points)	beállítja a szerzett pontszámot a paraméterben kapott értékre
public int compareTo(HighScore other)	rendezést határoz meg a toplistapontok között, csökkenő sorrendben

Generált osztályok

DaoMaster.java, DaoSession.java

Segítségükkel létrejönnek az entitásokhoz a megfelelő DAO osztályok, és az adatbázis minden fontos konfigurációval együtt. Mint nevük sugallja, a DaoMaster ismeri az összes DAO-t, a DaoSession pedig létrehozza a database session-t az applikáció elindításakor. Részlet a MainApplication.java kódjából:

```
daoSession = new DaoMaster(new DaoMaster
    .DevOpenHelper( context: this, name: "ProgrammerPuzzle.db")
    .getWritableDatabase())
    .newSession();
```

PuzzleDao.java, HighScoreDao.java

Ezek a generált DAO osztályok a megfelelő Puzzle és HighScore entitásoknak.

Adatmenedzsment osztályok

DaoManager.java

Egy réteget biztosít a PuzzleDao és HighScoreDao osztályok egyszerű kezelésére.

Adattagok:

private PuzzleDao puzzleDao	a konstruktorban megkapott PuzzleDao
private HighScoreDao highScoreDao	a konstruktorban megkapott HighScoreDao

Metódusok:

public List<Puzzle> getAllPuzzles()	meghívja a puzzleDao loadAll() metódusát, mely visszatér az összes puzzle-vel
public Long insertPuzzle(Puzzle puzzle)	meghívja a puzzleDao insert() metódusát, mely a paraméterben kapott puzzle-t elmenti az adatbázisba
public void updatePuzzle(Puzzle puzzle)	meghívja a puzzleDao update() metódusát, mely a paraméterben kapott puzzle-t frissíti az adatbázisban
public void deletePuzzle(Puzzle puzzle)	meghívja a puzzleDao delete() metódusát, mely a paraméterben kapott puzzle-t törli az adatbázisból
public void deleteAllPuzzles()	meghívja a puzzleDao deleteAll() metódusát, mely kitörli az összes puzzle-t az adatbázisból
public List<HighScore> getAllHighScores()	meghívja a highScoreDao loadAll() metódusát, mely visszatér az összes toplistaponttal
public Long insertHighScore(HighScore highScore)	meghívja a highScoreDao insert() metódusát, mely a paraméterben kapott toplistapontot elmenti az adatbázisba
public void deleteAllHighScores()	meghívja a highScoreDao deleteAll() metódusát, mely kitörli az összes toplistapontot az adatbázisból

DataCache.java

Az adatfeldolgozásban ez a legfelső réteg, mellyel az alkalmazás többi komponense is dolgozik. Listákban eltárolja a szükséges adatokat, melyeket az adatbázisból nyerünk, és interfészt biztosít az adatok manipulálására.

Adattagok:

private DaoManager daoManager	a konstruktorban kapott DaoManager
private List<Puzzle> puzzleList	Lista a puzzle-ök tárolására
private List<HighScore> highScoreList	Lista a toplistapontok tárolására

Metódusok:

public List<Puzzle> getPuzzleList()	visszatér a puzzle listával
public List<Puzzle> getPuzzleList(String language)	visszatér egy olyan puzzle listával, melyben csak a konstruktorban kapott programozási nyelvű puzzle-ök szerepelnek
public void createPuzzle(Puzzle puzzle)	hozzáadja a puzzle listához a paraméterben kapott puzzle-öt, és meghívja a daoManager insertPuzzle() metódusát, mely elmenti ezt a puzzle-t az adatbázisba
public void deletePuzzle(Puzzle puzzle)	kitörli a puzzle listából a paraméterben kapott puzzle-öt, és meghívja a daoManager deletePuzzle() metódusát, mely kitörli ezt a puzzle-t az adatbázisból
public void updatePuzzle(Puzzle puzzle)	frissíti a paraméterben kapott puzzle-t a listában, és meghívja a daoManager updatePuzzle() metódusát, mely frissíti ezt a puzzle-t az adatbázisban
public void deleteAllPuzzles()	üres listát csinál a puzzle listából, és meghívja a daoManager deleteAllPuzzles()

	metódusát, mely kitörli az összes puzzle-t az adatbázisból
<code>public List<HighScore> getHighScoreList()</code>	visszatér a toplistapont listával
<code>public void createHighScore(HighScore highScore)</code>	hozzáadja a toplistapont listához a paraméterben kapott toplistapontot, és meghívja a <code>daoManager insertHighScore()</code> metódusát, mely elmenti ezt a toplistapontot az adatbázisba
<code>public void deleteAllHighScores()</code>	üres listát csinál a toplistapont listából, és meghívja a <code>daoManager deleteAllHighScores()</code> metódusát, mely kitörli az összes toplistapontot az adatbázisból

PuzzleInitializer.java

Az `src/main/assets` mappában szereplő fájlokat soronként feldolgozza és elmenti a `DataCache` osztályba őket. A puzzle feladatokat tartalmazzák ezek a fájlok.

Adattagok:

<code>private DataCache dataCache</code>	a konstruktorban kapott <code>DataCache</code>
--	--

Metódusok:

<code>public void initPuzzles()</code>	ellenőrzi, hogy üres-e a puzzle lista a <code>DataCache</code> -ben, ha nem, meghívja a <code>createPuzzles()</code> metódust
<code>private void createPuzzles()</code>	<code>AssetManager</code> segítségével megnyitja a fájlokat beolvasásra, és egyenként feldolgozza őket, létrehozza a puzzle-öket ezen fájlok alapján a <code>DataCache</code> -ben

Game Logic

GameLogic.java

Ez az osztály felel az adatbázis és a nézet közötti interakciók egy részéért, a toplistapont számontartásáért, időzítő beállításáért és eltárolja a megkapott puzzle listát amíg a játék megy. Megvalósítja a GameTimerInterface-t.

Adattagok:

private NewGameInterface newGameInterface	ezen az interfacen keresztül hívódnak meg azok a metódusok, melyek a NewGameActivity-n módosítanak
private List<Puzzle> puzzleList	a konstruktorban megkapott puzzle lista
private Puzzle currentPuzzle	a puzzle amit a NewGameActivity meg fog jeleníteni
private long score	játékos pontszáma
private GameTimer gameTimer	időzítő
private static final int SECONDS_FOR_PUZZLE	ebben tároljuk, hány másodperc idő van egy puzzle-ra

Metódusok:

public void setNewGameInterface(NewGameInterface newGameInterface)	beállítja a newGameInterface adattagot a paraméterben kapottra
private void startTimer(long time)	elindítja az időzítőt a paraméterként kapott időre milliszekundumban
public void newPuzzle()	kivesz a puzzle listából egy puzzle-t véletlenszerűen, és meghívja a newGameInterface showPuzzle() metódusát, mely megjeleníti ezt a puzzle-t a képernyőn, és elindítja az időzítőt SECONDS_FOR_PUZZLE ideig
public void addScore(long score)	hozzáadja a pontszámhoz a paraméterben kapott értéket

public long getScore()	visszatér a pontszámmal
public void saveHighScore()	elkéri a SharedPreferences segítségével a beállításokból a játékos nevét, és elmenti a név-pontszám párost az adatbázisba a toplistapontok közé
public void tick(long timeLeft)	mutatja az időt a newGameInterface-n keresztül, és ellenőrzi, ha lejárt
public void end()	leállítja az időzítőt

NewGameInterface.java

Ez az interface azokat a metódusokat tartalmazza, melyeket a GameLogic osztályon belül hívunk meg, és a NewGameActivity osztály valósít meg.

Application

MainApplication.java

Ez az osztály felel a globális application state-ért. A MainApplication leszármazik az android.app.Application osztályból, vagyis az összes többi osztály előtt lesz példányosítva az alkalmazás elindításakor. Itt hozzuk létre az adatbázishoz szükséges rétegeket.

Adattagok:

private static MainApplication mainApplication	ebben tároljuk a MainApplication példányát
private DaoSession daoSession	az adatbázishoz szükséges DaoSession objektum
private DaoManager daoManager	a perzisztencia középső rétegének objektuma
private DataCache dataCache	a perzisztencia legfelső rétegének objektuma

Metódusok:

public static MainApplication getInstance()	visszatér a MainApplication példányával
public void onCreate()	meghívja a super.onCreate metódust, illetve létrehozza az adatbázishoz szükséges

	objektumokat, és a PuzzleInitializer segítségével beolvassuk az összes puzzle-t
public DaoSession getDaoSession()	visszatér a DaoSession objektummal
public DaoManager getDaoManager()	visszatér a DaoManager objektummal
public DataCache getDataCache()	visszatér a DataCache objektummal

Nézet

PuzzleButton.java

Ez egy speciális Button osztály, minden olyan tulajdonsággal rendelkezik, amivel egy átlagos Button is, de mellé eltárol egyéb fontos adatokat. Ezeket a gombokat mozgatjuk a játék során le-fel. Megvalósítja a Comparable interface-t.

Adattagok:

private ArrayList<Integer> correctLines	ez a lista tárolja a lehetséges helyes sorpozíciókat (egy vagy több)
private int actualLine	eltárolja a jelenlegi sorpozíciót
private int originalX	eltárolja a gomb X koordinátáját a képernyőn, mielőtt el lett volna mozdítva
private int originalY	eltárolja a gomb Y koordinátáját a képernyőn, mielőtt el lett volna mozdítva
private boolean moved	eltárolja, hogy el lett-e már mozdítva a gomb

Metódusok:

public ArrayList<Integer> getCorrectLines()	visszatér a helyes sorpozíciók listájával
public void setCorrectLines(ArrayList<Integer> correctLines)	beállítja a helyes sorpozíciók listáját a paraméterben kapott listára
public int getActualLine()	visszatér az aktuális sorpozícióval
public void setActualLine(int actualLine)	beállítja az aktuális sorpozíciót a paraméterben kapott értékre

<code>public int getOriginalX()</code>	visszatér a gomb eredeti X koordinátájával
<code>public void setOriginalX(int originalX)</code>	beállítja a gomb eredeti X koordinátáját a paraméterben kapott értékre
<code>public int getOriginalY()</code>	visszatér a gomb eredeti Y koordinátájával
<code>public void setOriginalY(int originalY)</code>	beállítja a gomb eredeti Y koordinátáját a paraméterben kapott értékre
<code>public int compareTo(PuzzleButton other)</code>	rendezést állít fel a PuzzleButton-ök között, az aktuális sorpozíciójuk alapján

Activity-k

HighScoreActivity.java

Ez az Activity megjeleníti a legjobb 10 toplistapontot csökkenő sorrendben.

Adattagok:

TextView line_1_Player TextView line_2_Player ... TextView line_10_Player	10 db TextView a játékosnevek megjelenítéséhez
TextView line_1_Score TextView line_2_Score ... TextView line_10_Score	10 db TextView a pontszámok megjelenítéséhez
<code>private List<HighScore> highScores</code>	lista a toplistapontok tárolására
<code>private ArrayList<TextView> players</code>	lista a játékosneveket tartalmazó TextView objektumok tárolására
<code>private ArrayList<TextView> scores</code>	lista a pontszámokat tartalmazó TextView objektumok tárolására

Metódusok:

protected void onCreate(Bundle savedInstanceState)	az AppCompatActivity osztályból származó metódus, az Activity létrejöttkor fut le meghívja a super.onCreate() és az activityDesign() metódusokat
private void activityDesign()	meghívja az összes többi metódust az osztályban, melyek beállítják az Activity végső kinézetét
private void fillPlayersList()	a játékosneveket tartalmazó TextView objektumokat hozzáadja az ehhez tartozó listához
private void fillScoresList()	a pontszámokat tartalmazó TextView objektumokat hozzáadja az ehhez tartozó listához
private void setTextStyles()	beállítja az összes TextView betűméretét és betűszínét
private void getHighScores()	feltölti a toplistapontokat tartalmazó (eddig üres) listát az adatbázisból lekért adatokkal, és sorba rendezi őket
private void setHighScoreTexts()	beállítja a TextView objektumok szövegét a highScores lista felső 10 elemével

HowToPlayActivity.java

Ez az Activity segít eligazodni a játékban, ad egy leírást a játék lényegéről, és hogy hogyan kell végigjátszani. Magyarázatot ad a NewGameActivity tetején levő gombokról is.

Adattagok:

TextView text1	megjeleníti a szöveg első részét
TextView text2	megjeleníti a szöveg második részét
TextView text3	megjeleníti a szöveg harmadik részét

Metódusok:

protected void onCreate(Bundle savedInstanceState)	az AppCompatActivity osztályból származó metódus, az Activity létrejöttkor fut le meghívja a super.onCreate() és az activityDesign() metódusokat
private void activityDesign()	meghívja a szükséges metódusokat az Activity kinézetének beállításához
private void setTexts()	beállítja a TextView objektumokhoz a megfelelő szövegeket a string állományokból

MainActivity.java

A kiinduló Activity, az alkalmazás betöltése után ez jelenik meg legelőször. Alaphelyzetben csak az app ikon és név látható, de a bal felső sarokban látható gombbal, vagy a képernyő bal széléről jobbra húzva előhozható a menü, mely egy Navigation Drawer segítségével lett megvalósítva. Itt lesz példányosítva a GameLogic, ha elindítunk egy új játékot. Megvalósítja a NavigationView.OnNavigationItemSelectedListener interface-t.

Adattagok:

NavigationView navigationView	kihúzható menü objektuma
TextView applicationTitle	az alkalmazás nevét jeleníti meg a képernyőn
Toolbar toolbar	toolbar a NavigationView-hez
DrawerLayout drawer	a NavigationView „fiók”-ja, ezzel tudjuk kinyitni
public static int PUZZLES_IN_ONE_GAME	eltárolja, hány puzzle van egy játék során
public static int PUZZLE_COUNT	számontartja, hányadik puzzle-t töltöttük be
private String language	a puzzle-ök programozási nyelve
private static GameLogic gameLogic	a GameLogic osztály példánya, melyet új játék indításakor inicializálunk

Metódusok:

protected void onCreate(Bundle savedInstanceState)	az AppCompatActivity osztályból származó metódus megvalósítása, meghívja a super.onCreate() és az activityDesign() metódusokat
private void activityDesign()	megjeleníti az app nevét, és a NavigationDrawer-t konfigurálja
public void onBackPressed()	származtatott függvény, akkor hívódik meg, amikor a „vissza” gombra nyomunk a DrawerLayout-ot zárja be, ha meg van nyitva
public boolean onOptionsItemSelected(MenuItem item)	eseménykezelő metódus, a paraméterként kapott MenuItem alapján amelyik menüpontra nyomunk, az annak megfelelő Activity fog betöltődni
private void startNewGame()	példányosítja a GameLogic-ot, és elindít egy NewGameActivity-t
private void loadSettings()	betölti a beállításokból a kívánt programozási nyelvet
public static GameLogic getGameLogic()	visszatér a GameLogic példányával

NewGameActivity.java

A játék szempontjából a leglényegesebb és legterjengőbb Activity. Játék logikát is tartalmaz.

Adattagok:

PuzzleButton line_1 PuzzleButton line_2 ... PuzzleButton line_20	20 db PuzzleButton, melyeket megérintve lehet őket bemozgatni a következő üres helyre, vagy visszatenni a felső panelba
---	---

View separatorLine	a felső és alsó panelt vizuálisan elválasztó szeparáló vonal
Button placeholder_1 Button placeholder_2 ... Button placeholder_20	20 db placeholder gomb, melyeknek csak annyi a szerepe, hogy jelzik 1-től 20-ig a sorokat a programkódban, ezek mellé kerülnek be a PuzzleButton-ök
ViewGroup viewGroup	az activity minden nézeti objektuma egy Relative Layout-ban szerepel, a programkódban ezzel a ViewGroup objektummal hivatkozunk rá
TextView timer	az időzítőt megjelenítő TextView
ImageButton hintButton	a segítő gomb objektuma
ImageButton skipButton	az adott puzzle-ről a következőre lépés gombja
private static int CURRENT_LINE	eltárolja a következő „szabad” sor indexét
private static String CURRENT_STARTING_SPACE	eltárolja a nyitó és csukó kapcsos zárójelek száma alapján a tabuláláshoz szükséges space mennyiséget
private long timeLeftSec	hátralevő idő másodpercben
private String language	puzzle programozási nyelve
private boolean soundOn	flag változó annak jelzésére, hogy engedélyezve vannak-e a hangeffektek a beállításokban
private List<PuzzleButton> lines	lista a PuzzleButton-ök tárolására
private List<Button> placeholders	lista a placeholder Button-ök tárolására
private List<PuzzleButton> usedLines	lista a már felhasznált PuzzleButton-ök tárolására
private List<PuzzleButton> usedPlaceholders	lista, mely azokat a placeholder Button-okat tartalmazza, melyek mellett már szerepel egy sor kód
private MediaPlayer soundPlayer	MediaPlayer objektum a hangok lejátszására
private VibratorEngine vibratorEngine	a vibrálást létrehozó objektum

Metódusok:

protected void onCreate(Bundle savedInstanceState)	meghívja sorban a super.onCreate(), setLanguage(), loadSettings(), activityDesign(),fillLineList(), fillPlaceHoldersList(),setPlaceHoldersText(), removeButtonDependencies(),setOnTouchListeners(), loadPuzzle() metódusokat
private void setLanguage()	beállítja a programozási nyelvet, melyet Intent segítségével kapunk a MainActivity-tól
private void loadSettings()	a SharedPreferences-ből lekéri a beállításokat, és ezeknek megfelelően beállítja a képernyő-orientációt, vibrálás és hangeffektek létezését
private void activityDesign()	beállítja a felső statusbar színét
private void fillLineList()	feltölti a PuzzleButton listát a 20 PuzzleButton-nal
private void fillPlaceHoldersList()	feltölti a placeholder listát a 20 Button-nal
private void setPlaceHoldersText()	beállítja a placeholderek szövegét, 1-től 20-ig sorszámokkal, hasonlóan egy szövegszerkesztőhöz
private void removeButtonDependencies()	letörli a PuzzleButton-ök pozícióbeli függőségeit azzal, hogy lekéri a képernyőn az X / Y koordinátákat és ezeket állítja be abszolút pozíciónak erre azért van szükség, hogy elkerüljük a folyamatos átláncolgatást, és a körkörös függőségeket
private void setXY(View view)	segédmetódus a removeButtonDependencies()-hez, beállítja egy Button X és Y pozícióit a képernyőről lekérve
private void setOnTouchListeners()	meghívja a linesOnTouchListeners(), hintButtonOnTouchListener(), skipButtonOnTouchListener() metódusokat

private void linesOnTouchListeners()	beállítja, mi történjen, ha hozzáérnek egy PuzzleButton-hoz: meghívódik a moveButton() metódus
private void hintButtonOnTouchListener()	beállítja, mi történjen, ha hozzáérnek a segítő gombhoz: meghívódik a useHint() metódus
private void useHint()	ellenőrzi, hogy az eddig felhasznált PuzzleButton-ök jó helyen vannak-e, ha nem, akkor meghívja a moveIncorrectLinesBack() metódust, és letiltja a segítő gomb további használatát ha helyesek az eddig felhasznált PuzzleButton-ök, akkor beteszi a következő szabad sorba a helyes PuzzleButton-t, és letiltja a segítő gomb további használatát
private boolean areLinesCorrect()	segéd metódus a useHint()-hez, ellenőrzi, hogy az eddig felhasznált PuzzleButton-ök jó helyen vannak-e
private void moveIncorrectLinesBack()	segéd metódus a useHint()-hez, meghívja a moveButtonBack() metódust azokra a PuzzleButton-ökre, melyek rossz helyen vannak
private void skipButtonOnTouchListener()	beállítja, mi történjen, ha hozzáérnek a skip gombhoz: meghívja a showYesNoDialog() metódust
private void loadPuzzle()	beállítja a GameLogic NewGameInterface-jét erre az osztályra, és meghívja a GameLogic newPuzzle() metódusát
private void moveButton(View view, int j)	a megérintett PuzzleButton-t beteszi a usedLines listába, és TransitionManager segítségével átmozdul a gomb a következő szabad sorba beállítja a megérintett gomb jelenlegi sorpozícióját arra, ahová bekerült

		<p>a megfelelő placeholder bekerül a usedPlaceholders listába</p> <p>frissíti a tabulálást a setSpacing() metódussal, frissíti a listenereket a refreshListeners() metódussal, és ellenőrzi, hogy kész-e a puzzle a checkPuzzleDone() metódussal</p>
private	void	<p>azok a PuzzleButton-ök esetén hívódik ez meg, amelyek úgy lettek megírva, hogy be legyenek égetve fixen a megfelelő placeholder helyükre, segítségül szolgálva a játékosnak</p> <p>bemozgatja a PuzzleButton-t a megfelelő helyre, és letiltja használatát</p> <p>hozzáadja a PuzzleButton-t a felhasznált sorok listájához, és a megfelelő placeholder-t is hozzáadja a felhasznált placeholderek listájához</p>
private	void	<p>megvalósítja a már felhasznált PuzzleButton-ök esetén a felső panelra való visszamozgatást</p> <p>elkéri az eredeti X és Y koordinátáit a PuzzleButton-nak, és oda mozgatja vissza TransitionManager segítségével</p> <p>a tabulálást leszedi a szöveg elejéről a PuzzleButton-ön</p> <p>kiveszi a megfelelő placeholder-t a felhasznált placeholderek listájából</p> <p>beállítja a PuzzleButton aktuális sorpozícióját mínusz egyre</p> <p>kiveszi a PuzzleButton-t a felhasznált PuzzleButton-ök listájából</p>

	frissíti a tabulálást a <code>setSpacing()</code> metódussal, és frissíti a listenereket a <code>refreshListeners()</code> metódussal
<code>private void setSpacing()</code>	beállítja a megfelelő tabulálást minden már felhasznált <code>PuzzleButton</code> -ön a space mennyiség attól függ, hány nyitó és csukó kapcsos zárójel előzi meg az adott sort
<code>private void refreshListeners()</code>	frissíti a <code>PuzzleButton</code> -ök listenerjét úgy, hogy ha a <code>PuzzleButton</code> benne van a <code>usedLines</code> listában, akkor a <code>moveButtonBack()</code> fog meghívódni érintés esetén, különben a <code>moveButton()</code>
<code>public void showPuzzle(Puzzle puzzle)</code>	beállítja a <code>puzzleDescription</code> szövegét a puzzle feladatlírására, a puzzle szövegét felbontja sorokra a <code>tokenizeCode()</code> metódus segítségével meghívja a <code>disableEmptyButtons()</code> , <code>moveFixedLines()</code> metódusokat, és növeli a <code>PUZZLE_COUNT</code> -ot a <code>MainActivity</code> -ben 1-el
<code>private void disableEmptyButtons()</code>	láthatatlanná teszi azokat a <code>PuzzleButton</code> -öket, melyekbe nem került szöveg
<code>private void moveFixedLines()</code>	meghívja a megfelelő <code>PuzzleButton</code> -okra a <code>moveFixedButton()</code> metódust
<code>private void tokenizeCode(String[] puzzleLines, boolean languagePython)</code>	rendezi a forráskód sorait annak megfelelően, hogy az első oszlopban szereplő <code>PuzzleButton</code> -ök (<code>line_1 ... line_10</code>) szövegei hosszban csökkenő sorrendben szerepeljenek, a második oszlopban szereplő <code>Puzzlebutton</code> -ök (<code>line_11 ... line_20</code>) szövegei hosszban növekvő sorrendben, ennek az a célja, hogy minél jobban elférjenek a képernyőn meghívja a <code>setLineTexts()</code> metódust a sorokra

private void setLineTexts(String[] puzzleLines, int j, boolean languagePython)	<p>annak megfelelően, hogy Python nyelvű-e a puzzle, felbontja a szöveg sorait a komment jelekkel</p> <p>beállítja a PuzzleButton-oknak a felbontott szövegeket</p> <p>beállítja minden PuzzleButton-nak a helyes sorokat</p>
private void checkPuzzleDone()	<p>ellenőrzi, hogy sikeresen összerakta-e a játékos a puzzle-t</p> <p>ha igen, hozzáad az eltelt idő és a puzzle hosszúsága alapján egy pontszámot a GameLogic-ban, és meghívja a showPuzzleDoneDialog() és playSound(R.raw.puzzledone) metódusokat</p>
public void timeExpired()	meghívja a showTimeExpiredDialog() metódust
public void gameEnd()	meghívja a GameLogic end() metódusát, és a showGameEndDialog() metódust
public void showTimer(long timeLeft)	a paraméterben kapott idő milliszekundumban van, ezt felosztja percekre és másodpercekre, és beállítja az időzítő szövegét ennek megfelelően
public void timeExpiring()	meghívja a playSound(R.raw.tick) metódust
public void setButtonsEnability(boolean enabled)	új puzzle betöltésénél ismét engedélyezetté teszi az összes PuzzleButton és a segítő gomb használatát
public void onBackPressed()	<p>ez a metódus mondja meg, mi történjen, ha a „vissza” gombot érintjük meg ebben az Activity-ben</p> <p>feljön egy dialog, mely megkérdezi a játékost, hogy vissza akar-e lépni a főmenübe, ha igenre nyom, ez meg is történik, ha nemre nyom, akkor visszatér a játékhoz</p>

private showGameEndDialog()	void	feljön egy dialog, mely kiírja, hogy vége a játéknak, és hogy mennyi a játékos pontszáma
private showTimeExpiredDialog()	void	meghívja a showOkDialog() metódust
private showPuzzleDoneDialog()	void	meghívja a showOkDialog() metódust
private showYesNoDialog(String message)	void	feljön egy dialog, melynek szövege a paraméterként kapott String ha igenre nyom a játékos, és még nem érte el a puzzle-ök száma a limitet, akkor újraindul az Activity ha elérte a limitet, akkor a gameEnd() metódust hívja meg ha nemre nyom a játékos, akkor visszatér a játékhoz
private void showOkDialog(String message)		feljön egy dialog, melynek szövege a paraméterként kapott String ha az OK-ra nyom a játékos, és még nem érte el a puzzle-ök száma a limitet, akkor újraindul az Activity ha elérte a limitet, akkor a gameEnd() metódust hívja meg
private void playSound(int resourceId)		a paraméterben kapott resourceId-hez tartozó hangeffektet játssza le MediaPlayer segítségével

SettingsActivity.java

A beállítások menüponthoz tartozó Activity, megvalósítása eltér a többi Activity-től, az osztálydeklarációban is látható, hogy a PreferenceActivity-ből származik le. Az ide tartozó nézeti objektumok az src/main/res/xml/preference.xml fájlban vannak definiálva.

Metódusok:

protected void onCreate(Bundle savedInstanceState)	meghívja a super.onCreate() metódust, beállítja a statusbar színét, és létrehoz egy SettingsFragment-et
--	---

Az Activity tartalmaz egy beágyazott statikus osztályt:

```
public static class SettingsFragment extends PreferenceFragment
```

Ez egy olyan Fragment osztály, mely SharedPreferences-el dolgozik. A SharedPreferences egyszerű adatok mentésére használható, melyeket az applikáció későbbi használata esetén is elő lehet venni. Egy applikáció beállításait ilyen módon egyszerűen el lehet menteni anélkül, hogy adatbázist kellene felépíteni hozzá. A preference.xml tartalmának részletezéséről az XML szekcióban lesz szó.

Metódusok:

public void onCreate(Bundle savedInstanceState)	meghívja a super.onCreate() metódust, beállítja a preference adatokat a preference.xml-ből, és meghívja a bindSummaryValue() metódust
private static void bindSummaryValue(Preference preference)	beállítja a preference onChangeListener-jét az osztályban létrehozott changeListener-re

Az osztályban létrehozunk egy statikus Preference.OnPreferenceChangeListener objektumot, melynek megvalósítjuk a következő metódusát:

public boolean onPreferenceChange(Preference preference, Object newValue)	az EditTextPreference-k esetén beállítja a preference leírását a user input-ra
---	--

Utility osztályok

A Tools package tartalmazza azon osztályokat, melyek segéd osztályként funkcionálnak a programban, és logikailag nem köthetők máshova.

GameTimer.java

Ez az osztály valósítja meg az időzítőt egy CountdownTimer objektum segítségével.

Adattagok:

private CountdownTimer countdownTimer	a CountdownTimer példánya az osztálynak
private GameTimerInterface gameTimerInterface	a konstruktorban megkapott GameTimerInterface

Metódusok:

private void initCountDownTimer(long time, long tickTime)	létrehozza a CountdownTimer objektumot, és megvalósítja az onTick() és onFinish() metódusait, melyek a gameTimerInterface tick() és end() metódusaival lesznek megegyezőek
public void start()	elindítja a CountdownTimer-t
public void stop()	megállítja a CountdownTimer-t

GameTimerInterFace.java

Ez az interface azokat a metódusokat tartalmazza, melyeket a GameLogic osztály valósít meg.

VibratorEngine.java

Ez az osztály valósítja meg a vibráláshoz szükséges motort.

Adattagok:

public static final int SHORT_VIBRATION_TIME	a vibrálás hossza milliszekundumban
private Vibrator vibratorEngine	az osztály Vibrator objektuma

Metódusok:

public void vibrate(int vibrationLength)	a paraméterben kapott milliszekundum ideig rezegteti a telefont
public void cancel()	leállítja a rezgést

XML fájlok

Mint ahogy az Android ismertető részben említésre került, az erőforrásokat .xml fájlokban tároljuk. Ezekből sokféle létezik, a projektben szereplők közül kiemelném a lényegeseket.

Mipmap

Az alkalmazásban fellelhető ikonok forrásfájljai szerepelnek a mipmap-hdpi, mipmap-mdpi, ... , mipmap-xxxhdpi mappákban. A különböző készülékek különböző képernyőfelbontással rendelkeznek, azért van 5 különböző méret minden fájlból, hogy az Android kiválaszthassa minden esetben a megfelelő felbontását.

Raw

Ez a mappa tartalmazza a hangfájlokat.

Values

arrays.xml

Ez a fájl tartalmazza azokat a string-tömböket, melyeket felhasználok a programkódban.

colors.xml

Ide kerülnek az alkalmazásban felhasznált színek kódjai, hogy könnyen lehessen rájuk hivatkozni.

dimens.xml

Ez a fájl tartalmaz bizonyos dp értékeket, hogy könnyebben lehessen rájuk hivatkozni.

strings.xml

Azok a szövegek, melyek ismétlődnek, vagy túl hosszúak ahhoz, hogy programkódban legyenek tárolva, ide kerülnek. Megkönnyítik az esetleges refaktorálást.

styles.xml

Különböző stílusokat lehet itt létrehozni, ha az alkalmazás megköveteli, hogy többször is fel lehessen használni ugyanolyan.

Preference

Az xml nevű mappában szerepel a preference.xml fájl, amelyről már volt szó a beállítások Activity-ben.

preference.xml

Ebben a fájlban definiáljuk a különböző beállítások menüpontok típusát és tartalmát. Egy PreferenceCategory-ban lehet csoportosítani a beállítások típusait. A programban „Player” és „In-game” kategóriák szerepelnek.

Háromféle preference-t használtam a programban:

- EditTextPreference
- ListPreference
- SwitchPreference

Az EditTextPreference egy szöveget tartalmazó mező, melyet szerkeszteni lehet a billentyűzetről megadott inputtal. A ListPreference nevéből adódóan egy listányi adatot tárol el, és elemeiből egyet lehet kiválasztani. A SwitchPreference a legegyszerűbb, be- vagy kikapcsolni lehet, boolean értéket tárol el.

Layout

Nézeti szempontból a leglényegesebb .xml fájlok itt szerepelnek. Ezekben definiáljuk az összes nézeti elemet, az egymáshoz való relációjukat, elhelyezésüket a képernyőn, méretüket, betűtípust, háttérszínt, és még sok más.

A programban RelativeLayout-ot használtam az elemek elrendezésére. Ez egy olyan view group, melyben a nézeti elemek pozícióit egymáshoz képest lehet definiálni, vagy a szülő konténerben definiált helyét is meg lehet adni. Példa mindkettőre:

android:layout_below="@id/name"

android:layout_alignParentLeft="true"

Néhány fontosabb attribútum magyarázata:

android:layout_width	a nézeti objektum szélességét adja meg
android:layout_height	a nézeti objektum magasságát adja meg

android:id	a nézeti objektum azonosítója
android:textSize	a nézeti objektumban szereplő szöveg mérete
android:layout_below	RelativeLayout-ban egy nézeti objektum helyének definiálása egy másik objektum alá
android:layout_toEndOf	RelativeLayout-ban egy nézeti objektum definiálása egy másik objektum jobb oldalára
android:src	ImageView-ek, ImageButton-ök esetén az ikon forrása
android:textColor	a nézeti objektumban szereplő szöveg színe

activity_high_score.xml

A toplistapontok megjelenítésére egy TableLayout-ot használunk, mely táblázatszerűen rendezi elemeit, és dinamikusan lehet hozzáadni / törölni elemeket. 2 TextView-et használunk a fejléchez, és 10-10 TextView-et a játékosnevek és pontszámok megjelenítésére.

activity_how_to_play.xml

A játékleírás szövegét egy RelativeLayout-ban tároljuk, mely egy ScrollView-be lett becsomagolva. Ennek oka az, hogy normál esetben nem fér ki a képernyőre a teljes szöveg, és a ScrollView segítségével lehet görgetni. A szöveget 3 TextView-ben tároljuk, a 2 megjelenítendő képet ImageView-ben, és sorban egymás alá vannak helyezve.

activity_main.xml

A kezdőképernyő kinézetét definiáljuk itt, mely a Navigation Drawer toolbar-jából, az app ikonjából és az app nevéből áll. Az ikonhoz ImageView-et használunk fel, az app nevéhez TextView-et.

activity_new_game.xml

A játékleírás Activity-hez hasonlóan itt is egy RelativeLayout tartalmazza a képernyő elemeit, mely egy ScrollView-be lett becsomagolva, így lehet görgetni a képernyőn fel-le. A képernyő tetején levő elemek az időzítő egy TextView-ben, a segítség és kihagyás gombok pedig 1-1 ImageButton. Ezek alatt szerepel a feladatléírás egy TextView-ben. A feladatléírás alatt szerepel a 20 PuzzleButton 2 oszlopba rendezve. A PuzzleButton-ök alatt van egy elválasztó vonal, ez alatt pedig a 20 placeholder gomb, 1-től 20-ig számozva egymás alatt.

activity_settings.xml

Mivel a preference.xml-ben definiáltuk a beállítások menüpontokat, itt nem szerepel semmi extra információ.

nav_header_main.xml

Ez a fájl a Navigation Drawer-hez tartozik, megadjuk benne a drawer ikonját.

activity_main_drawer.xml

Itt definiáljuk a Navigation Drawer választható menüpontjait az ikonjaikkal együtt.

Tesztelés

3-féle tesztelést vittem véghez az alkalmazáson, eredetileg 2-félét terveztem, de a játék nézetének bonyolultsága miatt néhány dolgot manuálisan teszteltem, mert automatizálni nehéz őket. Név szerint:

- Unit tesztek
- Instrumented tesztek
- Feketedoboz tesztelés

Unit tesztek

DataCacheTest.java

Teszteltem az adatbázis azon rétegét, mellyel a többi nem perzisztens komponens érintkezik. A külső függőségek stub objektumait Mockito segítségével hoztam létre, melyek csak azt a célt szolgálták, hogy lehessen tesztelni az osztály funkcionalitását ezen függőségek tényleges példányosítása nélkül.

A DataCache.java minden metódusának működése tesztelve lett, minden teszt sikeres volt.

Az adatbázis többi rétegét külön nem teszteltem, mert a greenDAO API-ra épülnek, amelyen rengeteg fejlesztő dolgozott sok ideig, így feltehető, hogy alaposabban tesztelték, mint én valaha tudnám egymagam.

Azokat az osztályokat, melyek gettereket és settereket tartalmaznak, szintén nem teszteltem, mert ezen metódusok működése triviális.

Instrumented tesztek

Instrumented teszteknek hívjuk azokat a teszteket, melyek tényleges eszközön vagy emulátoron futnak le, és az alkalmazás olyan elvárt viselkedéseit tudjuk automatizált módon ellenőrizni, melyeket nem lennének képesek egyszerű unit tesztekkel.

Espresso

A User Interface működésének egy részét Espresso API-val teszteltem. Egyszerű, könnyen tanulható interfészt nyújt automatizált tesztek írására. Az automatizált tesztek nagy előnye, hogy sok különböző eszközön lehet lefuttatni, és sokkal gyorsabb, mint az ember által végrehajtott manuális tesztek.

Az Espresso tesztek alanya Samsung Galaxy A5 volt.

HighScoreActivityTest.java

Teszteltem az Activity-ben szereplő nézeti objektumok láthatóságát. Részlet a kódból:

```
@Test
public void testVisibilities() {
    getActivity();

    onView(withId(R.id.line_1_Player))
        .check(matches(isDisplayed()));
}
```

A @Test annotáció jelzi, hogy ez egy teszt metódus. Az onView-vel kezdődő utasítás ellenőrzi, hogy a line_1_Player azonosítóval rendelkező nézeti objektum látható a képernyőn.

HowToPlayActivityTest.java

Teszteltem az Activity-ben szereplő szövegek láthatóságát, a bennük levő szöveg egyezését az elvárttal, és az ikonok láthatóságát. Részlet a kódból:

```
onView(withId(R.id.howToPlayActivity_Text1))
    .check(matches(withText(R.string.howToPlayActivity_Text1)));
```

Ellenőrzi, hogy a howToPlayActivity_Text1 azonosítóval rendelkező nézeti objektum szövege megegyezik a howToPlayActivity_Text1 azonosítóval rendelkező szöveggel az erőforrásokból.

MainActivityTest.java

Ebben az osztályban teszteltem a MainActivity nézeti elemeinek láthatóságát, a Navigation Drawer működését, és a NewGameActivity egyes részeinek funkcionalitását.

Ellenőriztem, hogy a Navigation Drawer kinyitható a drawer-re kattintva, és hogy meg lehet nyitni az egyes menüpontokat rájuk kattintva.

Ellenőriztem, hogy a NewGameActivity-t megnyitva látszanak a nézeti elemei, letekerve a képernyő aljára az utolsó elem is látszik.

Ellenőriztem, hogy a „vissza” gomb megnyomására feljön a dialog, mely megkérdezi, ki akar-e lépni a játékos a főmenübe, és az igen-re kattintva ez meg is történik.

Ellenőriztem, hogy a segítség gombra kattintás után az már nem lesz használható az adott puzzle-ban.

Ellenőriztem, hogy a kihagyás gomb-ra kattintva feljön egy dialog, mely megkérdezi, hogy tovább akar-e lépni a játékos a következő puzzle-ra.

Ellenőriztem, hogy 5 puzzle-t átlépve feljön a játék végét jelző dialog, mely kiírja a játékos pontszámát, majd az OK-ra kattintva visszalép a főmenübe.

Ellenőriztem, hogy a New Game-ra kattintva bejön a NewGameActivity, a How to Play-ra kattintva bejön a HowToPlayActivity, a High Score-ra kattintva bejön a HighScoreActivity.

Manuális tesztek

A NewGameActivity egyéb funkcionalitásait és a SettingActivity-t saját magam teszteltem 3 különböző eszközön:

- Samsung Galaxy A5
- Nexus 5X (emulátorral)
- Huawei P8 Lite

Esemény	Elvárt eredmény	Kapott eredmény
a NewGameActivity-ben bármelyik mozgatható gombot megérintjük a felső panelről	a megérintett gomb átmozog animálva a következő üres sorba	megegyezik az elvárttal

a NewGameActivity-ben bármelyik mozgatható gombot megérintjük az alsó panelről	a megérintett gomb átmozog animálva a felső panelra az eredeti helyére	megegyezik az elvárttal
a NewGameActivity-ben rosszul kirakunk egy puzzle-t, majd megérintjük a segítség gombot	az összes helytelen sorban levő mozgatható gomb átmozog animálva a felső panelra az eredeti helyükre	megegyezik az elvárttal
a NewGameActivity-ben mielőtt megérintenénk bármelyik gombot, megérintjük a segítség gombot	a következő üres sorba a helyes mozgatható gomb átmozog animálva	megegyezik az elvárttal
a NewGameActivity-ben kirakunk helyesen félig egy puzzle-t majd megérintjük a segítség gombot	a következő üres sorba a helyes mozgatható gomb átmozog animálva	megegyezik az elvárttal
a NewGameActivity-ben kirakunk helyesen egy puzzle-t	a játék gratulál a játékosnak szöveges dialog formában, melyet leokézva bejön a következő puzzle	megegyezik az elvárttal
a NewGameActivity-ben lejár az idő (00:00 lesz)	a játék jelzi, hogy lejárt az idő szöveges dialog formában, melyet leokézva bejön a következő puzzle	megegyezik az elvárttal
a NewGameActivity-ben a kihagyás gombra kattintunk, és a feljövő dialogon igen-re kattintunk	betöltődik a következő puzzle	megegyezik az elvárttal
a NewGameActivity-ben befejezünk 5 puzzle-t	a játék kiírja a játékos pontszámát szöveges dialog formában, melyet leokézva bejön a főmenü	megegyezik az elvárttal
a játék végén leokézzuk a dialog-ot, mely kiírja a játékos pontszámát	ha belefér a top10-be a pontszám, akkor bekerül a toplistába a név és pontszám	megegyezik az elvárttal
a NewGameActivity-ben megérintünk egy olyan mozgatható	az alsó panelen a már bemozgatott gombok tabulálása	megegyezik az elvárttal

gombot, mely nyitó vagy csukó zárójelt tartalmaz	megváltozik a zárójelnek megfelelően	
a SettingsActivity-ben bekapcsoljuk a hangokat	a megfelelő események bekövetkeztekor lejátszanak a hozzájuk tartozó hangeffektek	megegyezik az elvárttal
a SettingsActivity-ben bekapcsoljuk a rezgést	a mozgatható gombok megérintésekor röviden rezeg a készülék	megegyezik az elvárttal
a SettingsActivity-ben átírjuk a játékos nevét	játék végeztével, ha belefér a top10-be a játékos pontszáma, már az új név kerül be a pontszámmal együtt a toplistába	megegyezik az elvárttal
a SettingsActivity-ben beállítjuk a nyelvet Java-ra / Python-ra / C++-ra	a NewGameActivity-ben a beállításoknak megfelelő programozási nyelvű puzzle-k szerepelnek csak	megegyezik az elvárttal
a SettingsActivity-ben beállítjuk a képernyőorientációt landscape-ra	a NewGameActivity fekvő orientációban lesz	megegyezik az elvárttal

Irodalomjegyzék

- [1] *Android alapú szoftverfejlesztés kezdőknek*, 2013, [69], 7.o..
- [2] *Android alapú szoftverfejlesztés kezdőknek*, 2013, [69], 19.o..
- [3] *Android alapú szoftverfejlesztés kezdőknek*, 2013, [69], 13.o..
- [4] *Android alapú szoftverfejlesztés kezdőknek*, 2013, [69], 16.o..
- [5] Wikipédia. (2018). Forrás: https://hu.wikipedia.org/wiki/Objektum-re%C3%A1ci%C3%B3s_lek%C3%A9p%C3%A9s
- [6] *greenDAO*. (2018). Forrás: <http://greenrobot.org/greendao/>