3. Beadandó feladat dokumentáció

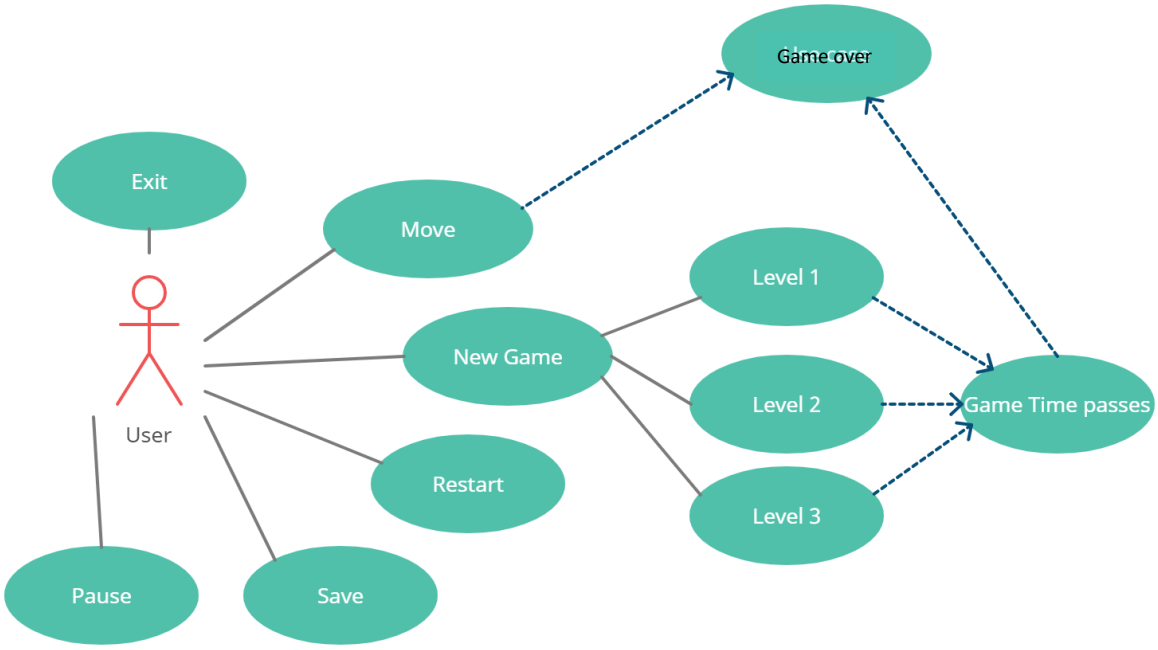
Készítette:
Melegh Dóra Marianna HCYGTM
e-mail: doramelegh@gmaill.com

Feladat:

Készítsünk programot, amellyel a következő játékot játszhatjuk. Adott egy $n \times n$ elemből álló játékpálya, amely falakból és padlóból áll, valamint őrök járőröznek rajta. A játékos feladata, hogy a kiindulási pontból eljusson a kijáratig úgy, hogy közben az őrök nem látják meg. Természetesen a játékos, illetve az őrök csak a padlón tudnak járni. Az őrök adott időközönként lépnek egy mezőt (vízszintesen, vagy függőlegesen) úgy, hogy folyamatosan előre haladnak egészen addig, amíg falba nem ütköznek. Ekkor véletlenszerűen választanak egy új irányt, és arra haladnak tovább. Az őr járőrözés közben egy 2 sugarú körben lát (azaz egy 5 × 5-ös négyzetet), ám a falon nem képes átlátni. A játékos a pálya előre megadott pontján kezd, és vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán. A pályák méretét, illetve felépítését (falak és kijárat helyzete, játékos és őrök kezdőpozíciója) tároljuk fájlban. A program legalább 3 különböző méretű pályát tartalmazzon. A program biztosítson lehetőséget új játék kezdésére a pálya kiválasztásával, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet a játékos). Továbbá ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hogy győzött, vagy veszített-e a játékos.

Elemzés:

- A játékot három szinttel játszhatjuk, amik három különböző pályával vannak megvalósítva.
- A feladatok Xamarin Forms alkalmazásként, Android platformon valósítjuk meg. Az alkalmazás portré tájolást támogat.
- Az képernyő (Game) tartalmazza a játéktáblát, a játék állását (lépések száma, idő) a lap alján az irányító gombok, valamint a pályák gombjai a lap tetején. Restart, Pause gombok is a felső részen találhatók.
- A játéktáblát egy 10 × 10 mezőből áll. Az irány gombok lenyomásának hatására a játékos megváltoztatja a helyzetet. A játékos fel, le, jobbra, balra mozoghat, ha nincs az útjában fal. A táblán falak és üres mezők, valamint egy Exit szerepel.
- A játék automatikusan jelzi előugró üzenettel, ha vége a játéknak (eljutottunk az Exithez, vagy megláttak az őrök).
- A felhasználói esetek az 1. ábrán láthatóak.

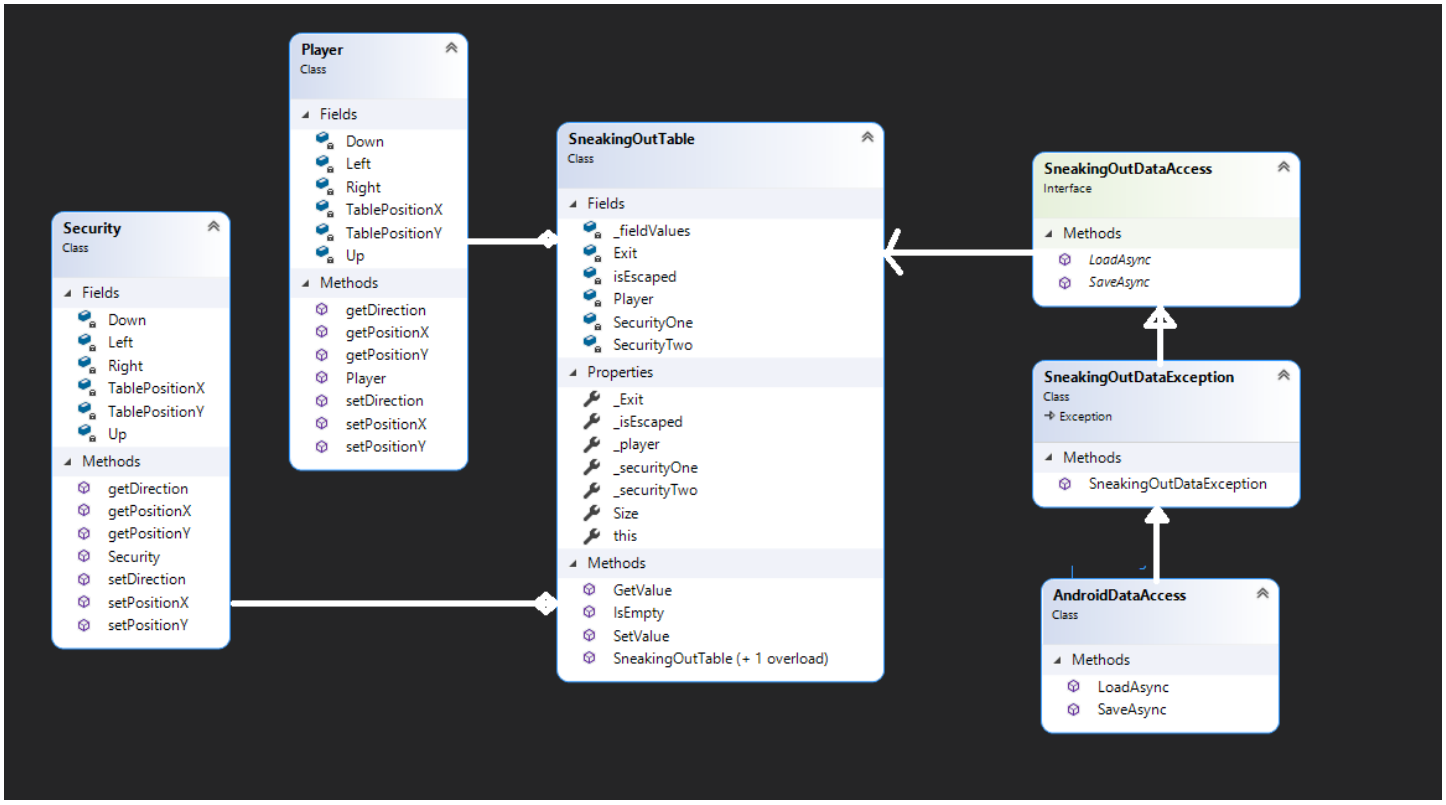


1. ábra Felnasználói esetek ábrája

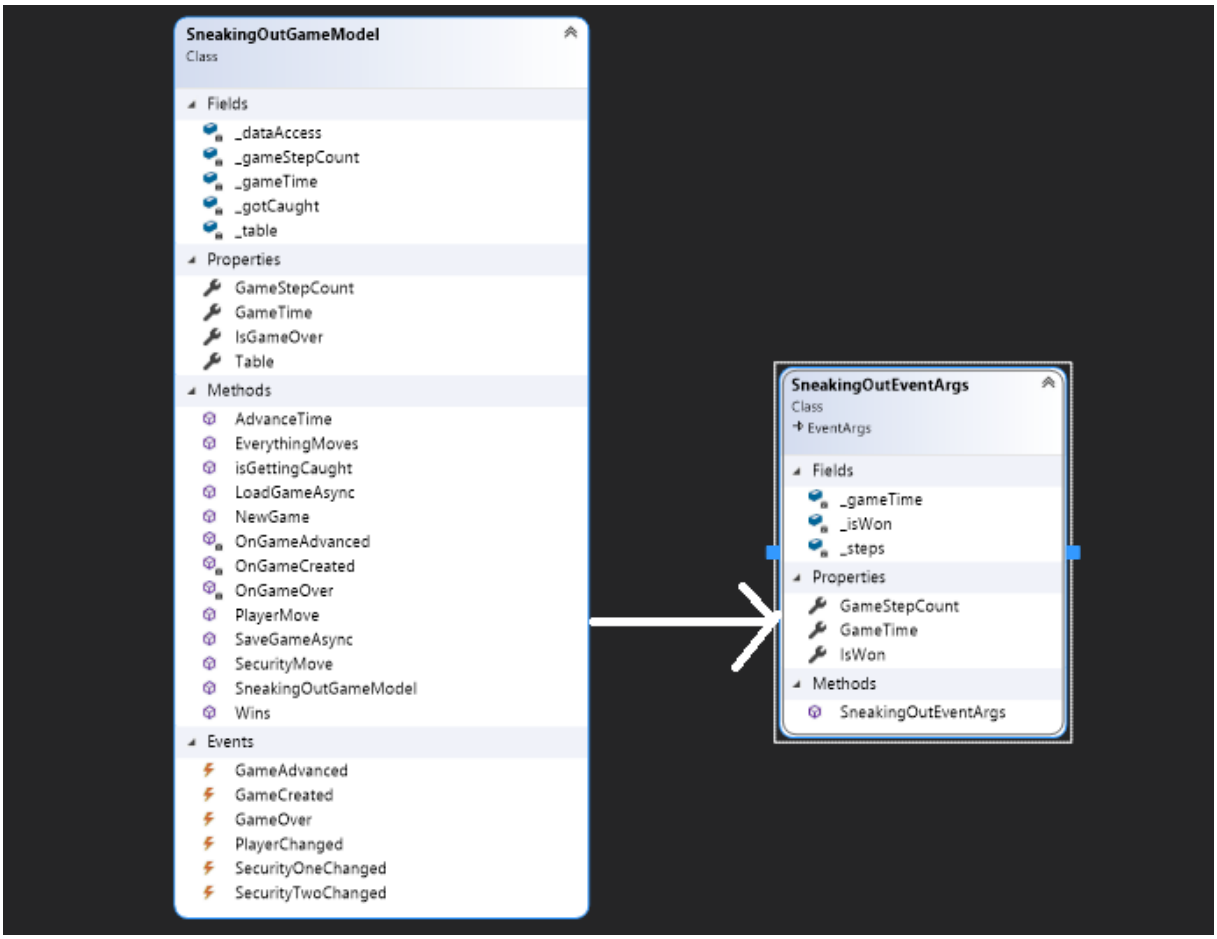
Tervezés

- Programszerkezet
  - A szoftvert két projektből építjük fel, a Xamarin Forms megvalósítást tartalmazó osztálykönyvtárból (.NET Standard Class Library), valamint az Android platform projektből. Utóbbi csupán a perzisztencia Android specifikus megvalósítását tartalmazza, minden további programegységet az osztálykönyvtárban helyezünk el.
  - A programot MVVM architektúrában valósítjuk meg, ennek megfelelően View, Model, ViewModel és Persistence névtereket valósítunk meg az alkalmazáson belül.
  - A megvalósításból külön építjük fel a játék, illetve a betöltés és mentés funkciót, valamennyi rétegben. Utóbbi funkcionalitást újrahasznosítjuk egy korábbi projektből, így nem igényel újabb megvalósítást.
  - A program vezérlését az alkalmazás osztály (App) végzi, amely példányosítja a modellt, a nézetmodell és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést.
  - A program csomagszerkezete a 2. ábrán látható.

- Perzisztencia (3.ábra)
  - Az adatkezelés feladata a SneakingOut táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
  - A SneakingOutTable osztály egy érvényes táblát biztosít (azaz mindig ellenőrzi a beállított értékek), ahol minden mezőre ismert az értéke (_fieldValues), valamint tartmalaz egy játékost (Player), 2 őrt (Security1, Security2) és egy kijáratot (Exit). A tábla alapértelmezés szerint 10 x 10, de ez a konstruktorban paraméterezhető. A tábla lehetőséget az állapotok lekérdezésére (_player, securityOne,_securityTwo, _isEscaped, _Exit, IsEmpty, GetValue), illetve direkt beállítás (SetValue) elvégzésére.
  - A hosszú távú adattárolás lehetőségeit az SneakingOutDataAccess interfész adja meg, amely lehetőséget ad a tábla betöltésére (LoadAsync), valamint mentésére (SaveAsync). A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg.
  - Az interfészt szöveges fájl alapú adatkezelésre a SneaingOutFileDataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a SneakingOutDataException kivétel jelzi.
  - A program az adatokat szöveges fájlként tudja eltárolni, melyek az txt kiterjesztést kapják.
  - A fájl első sora megadja a tábla méretét, (ami alapértelmezés szerint 10). A fájl többi része izomorf leképezése a játéktáblának, azaz összesen 10 sor következik, és minden sor 10 számot tartalmaz szóközökkel választva. A számok 0-4 közöttiek lehetnek, ahol 0 reprezentálja az üres mezőt, az 1 a Security1 a 2 a Security2 a 3 a Játékos helyzetét, a 4 a fal és 5 az Exit helyét.

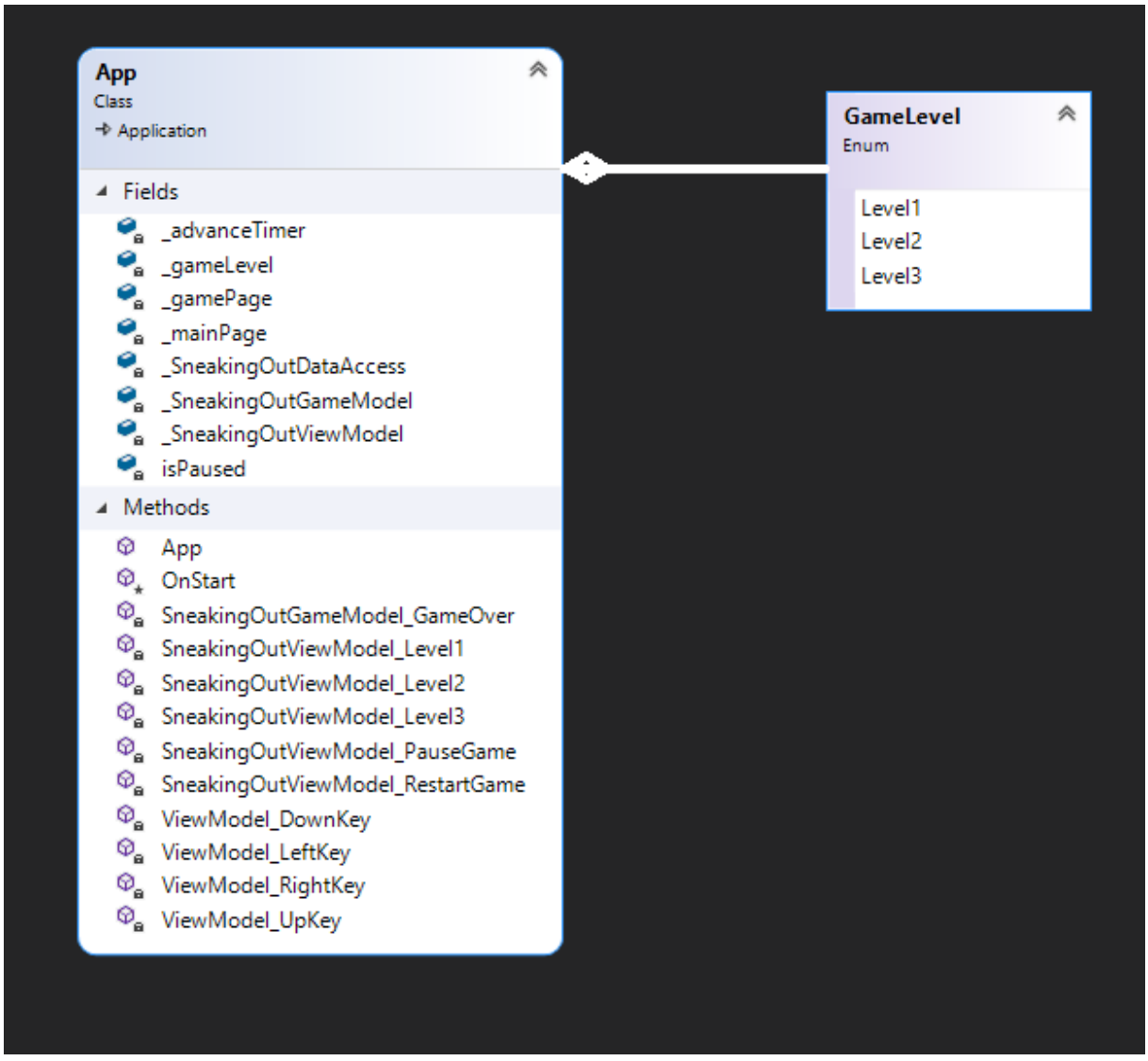


- Modell (4.ábra)
  - A modell lényegi részét a SneakingOutGameModel osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, úgymint az idő (_gameTime) és a lépések (_gameStepCount) . A típus lehetőséget ad új játék kezdésére (NewGame), valamint lépésre (PlayerMove). Új játéknál megadható a kiinduló játéktábla is. Az idő előreléptetését időbeli lépések végzéséve (AdvanceTime) tehetjük meg, ami az (EverythingMoves), a (SecurityMove) az (isGettingCaught),(Wins) segitségével frissíti a táblát és figyeli a játék állapotát.
  - A játékállapot változásáról a GameAdvanced esemény, míg a játék végéről a GameOver esemény tájékoztat. Az események argumentuma (SneakingOutEventArgs) tárolja a győzelem állapotát, a lépések számát, valamint a játékidőt.
  - A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (LoadGameAsync) és mentésre (SaveGameAsync)



- Nézetmodell (5.ábra)
    - A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) osztályt.
    - A nézetmodell feladatait a SneakingOutViewModel osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, mentéséhez, valamint a kilépéshez. A parancsokhoz eseményeket kötünk, amelyek a parancs lefutását jelzik a vezérlőnek. A nézetmodell tárolja a modell egy hivatkozását (_model), de csupán információkat kér le tőle.Direkt nem avatkozik a játék futtatásába.
    - A játékmező számára egy külön mezőt biztosítunk (SneakingOutField), amely eltárolja a pozíciót, az értéket, és azok tartozó kapcsolókat. A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (Fields).

- Nézet
  - A nézetet navigációs lapok segítségével építjük fel
  - A GamePage osztály tartalmazza a játéktáblát, amelyet egy FlowListView segítségével valósítunk meg (ez egy külső komponens), amelyben Button elemeket helyezünk el.
  - A fájlok betöltése a menüsor segítségével játék indításakor történik.
- Vezérlés
  - Az App osztály feladata az alkalmazás vezérlése, a rétegek példányosítása és az események feldolgozása.
    Kezeljük az alkalmazás életciklust, így felfüggesztéskor (OnSleep) elmentjük az aktuális játékállást folytatáskor és újraindításkor (OnStart) pedig folytatjuk, amennyiben történt mentés.



Tesztelés

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a SneakingOutModelTest osztályban.
- Az alábbi tesztesetek kerültek megvalósításra:
    - SneakingOutModelNewGameTest: Új játék indítása, valamint a lépésszám és az idő és játék végénel ellenőzése.
    - SneakingOutStepTest: Játékbeli lépés hatásainak ellenőrzése, a játékos helyzetének ellenőrzése.
    - SneakingOutEverythingMovesTest: A játékbeli őrök mozgatásának ellenőrzése, helyzetük meghatátozása.
    - SneakingOutGameModelLoadTest: A játék modell betöltésének tesztelése mockolt perzisztencia réteggel.