3. Beadandó feladat dokumentáció

Készítette: Gábris Attila

Email: ggloe7@inf.elte.hu

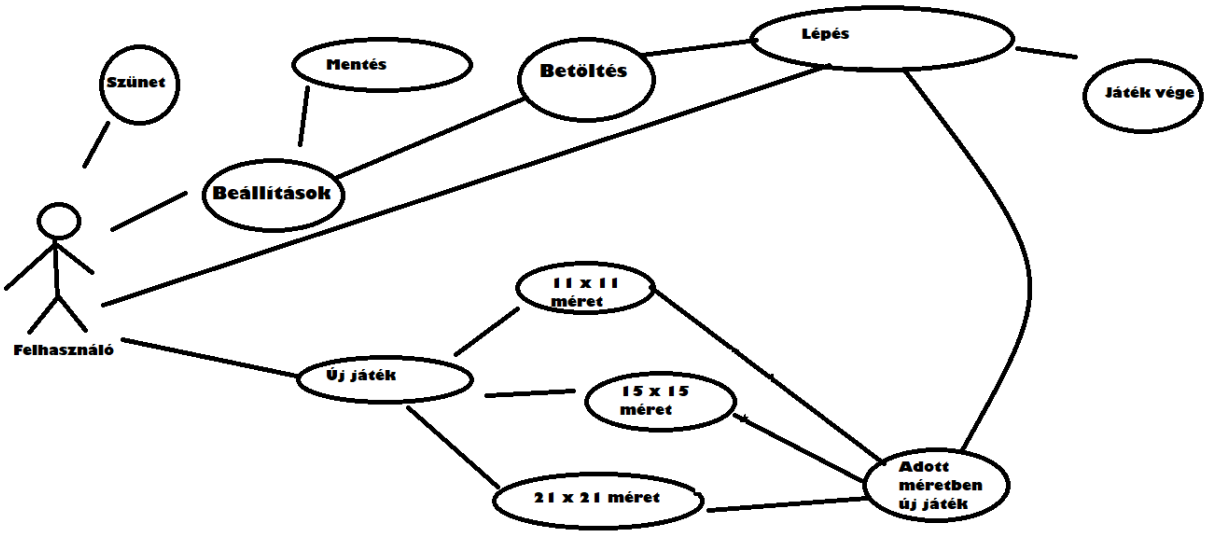
Feladat:

Menekülj:

Készítsünk programot, amellyel a következő játékot játszhatjuk. Adott egy $n \times n$ elemből álló játékpálya, ahol a játékos két üldöző elől próbál menekülni, illetve próbálja őket aknára csalni. Kezdetben a játékos játékpálya felső sorának közepén helyezkedik el, a két üldöző pedig az alsó két sarokban. Az ellenfelek adott időközönként lépnek egy mezőt a játékos felé haladva úgy, hogy ha a függőleges távolság a nagyobb, akkor függőlegesen, ellenkező esetben vízszintesen mozognak a játékos felé. A pályán véletlenszerű pozíciókban aknák is elhelyezkednek, amelyekbe az ellenfelek könnyen beleléphetnek, ekkor eltűnnek (az akna megmarad). A játékos vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán, és célja, hogy az ellenfeleket aknára csalja, miközben ő nem lép aknára. Ha sikerül minden üldözőt aknára csalnia, akkor győzött, ha valamely ellenfél elkapja (egy pozíciót foglal el vele), vagy aknára lép, akkor veszített. A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (11 × 11, 15 × 15, 21 × 21), valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet senki). Ismerje fel, ha vége a játéknak, és jelenítse meg, hogy győzött, vagy veszített-e a játékos. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére. A program játék közben folyamatosan jelezze ki a játékidőt.

Elemzés:

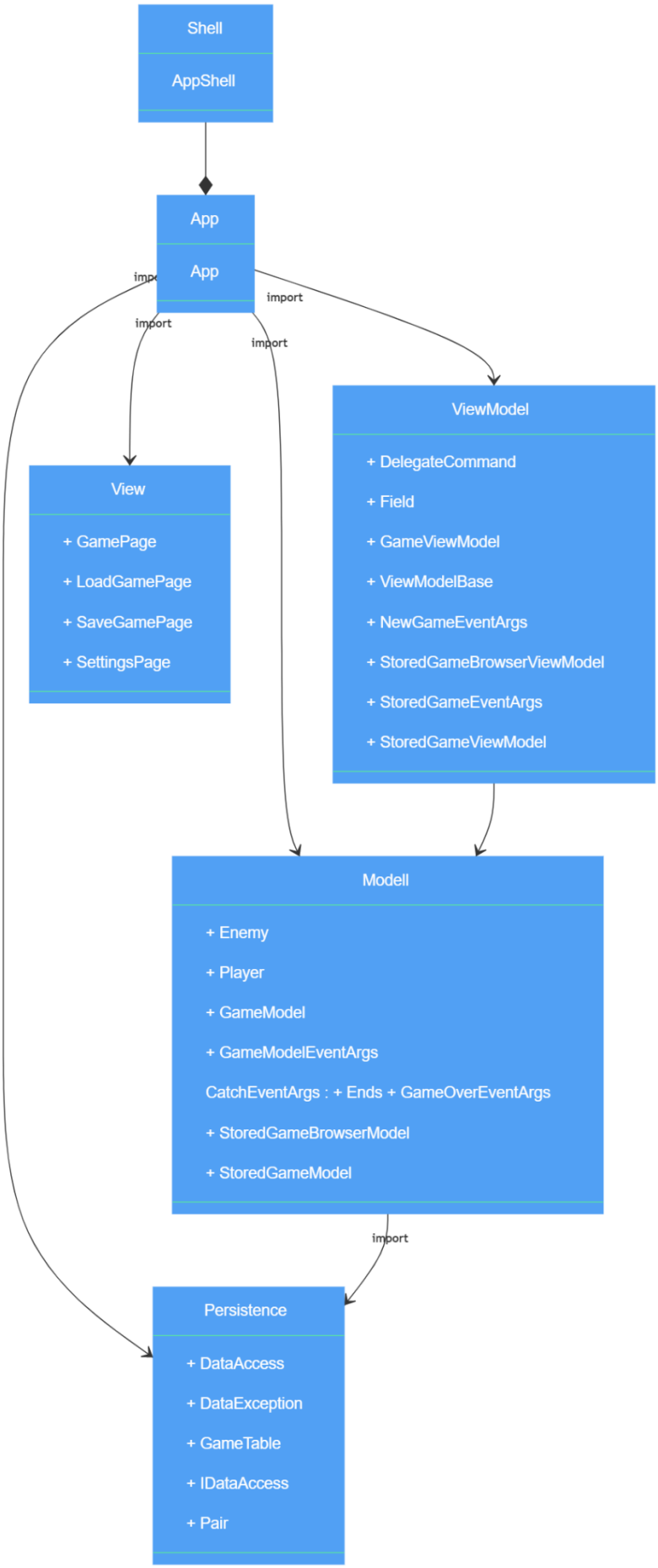
- A feladatot .NET MAUI alkalmazásként, elsődlegesen Windows és Android platformon valósítjuk meg. Az alkalmazás négy lapból fog állni. Az alkalmazás portré tájolást támogat.
- A játék négy képernyőn fog megjelenni.
    - Az első képernyő (Játék) tartalmazza a játéktáblát, a játék állását (lépések száma, fennmaradó idő) a lap alján, az új játék 3 méretben, a beállítások gombja és egy megállításra lehetőséget nyújtó gomb a lap tetején.
    - A második képernyőn van lehetőség betöltésre, illetve mentésre.
    - A további két képernyő a betöltésnél, illetve mentésnél megjelenő lista, ahol a játékok elnevezése mellett a mentés dátuma is látható. Mentés esetén ezen felül lehetőség van új név megadására is.
- A játékot három nehézségi szinttel játszhatjuk: könnyű (11 x 11 mező), közepes (15 x 15 mező), nehéz (21 x 21 mező). A program indításkor közepes nehézséget állít be, és automatikusan új játékot indít.
- A játékos, aki kezdetben a legfelső sor közepén helyezkedik el, menekül (lép) akkor amikor megérint egy számára tetszőleges irányba lévő mezőt. Az érintés hatására a játék lépteti az játékost a mező felé úgy, hogy amennyiben a függőleges távolság nagyobb, akkor az csökken, egyébként a vízszintes.
- A játék automatikusan jelzi előugró üzenettel, ha vége a játéknak. (Mindkét üldözőt aknára csaltuk – nyertünk, aknára léptünk - elkaptak - vesztettünk)
- A felhasználói esetek az 1. ábrán láthatóak:



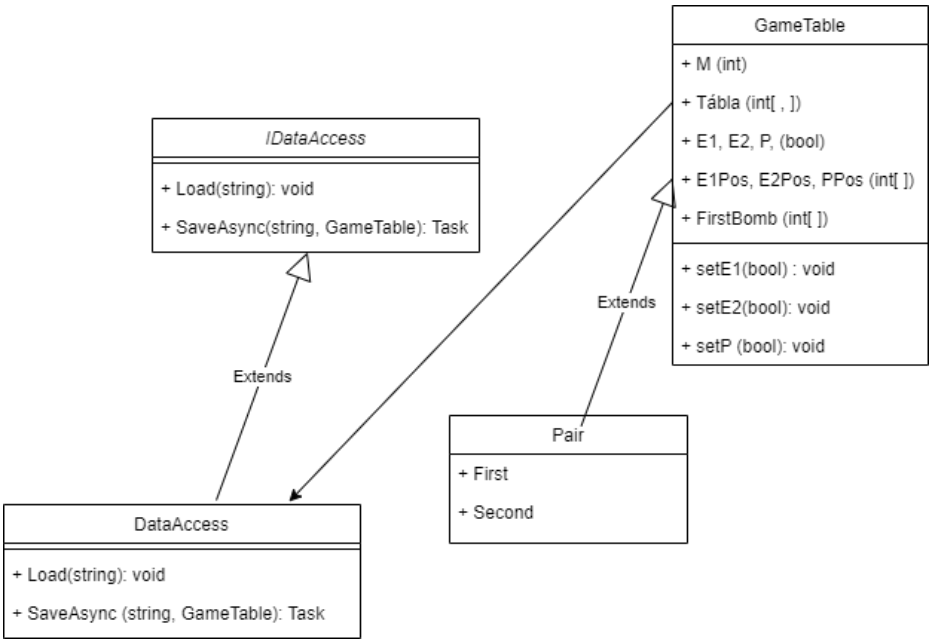
1. ábra

Tervezés:

- Programszerkezet:
  - A szoftvert két projektből építjük fel: a modellt és a perzisztenciát tartalmazó osztálykönyvtárból (.NET Maui Class Library), valamint a .NET MAUI többplatformos projektből, amelyet Windows és Android operációs rendszerre is le tudunk fordítani.
  - 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 csomagdiagramja a 2. ábrán látható.


Shell
AppShell
App
App
import
import
import
import
View
+ GamePage
+ LoadGamePage
+ SaveGamePage
+ SettingsPage
ViewModel
+ DelegateCommand
+ Field
+ GameViewModel
+ ViewModelBase
+ NewGameEventArgs
+ StoredGameBrowserViewModel
+ StoredGameEventArgs
+ StoredGameViewModel
Modell
+ Enemy
+ Player
+ GameModel
+ GameModelEventArgs
CatchEventArgs : + Ends + GameOverEventArgs
+ StoredGameBrowserModel
+ StoredGameModel
import
Persistence
+ DataAccess
+ DataException
+ GameTable
+ IDataAccess
+ Pair

- Perzisztencia:
    - Az adatkezelés feladata a játéktáblával kapcsolatos információk tárolása, valamint a betöltés és a mentés megvalósítása.
    - A GameTable osztály egy érvényes játéktáblát tárol, melynek ismert a mérete (hányszor hányas) és a mező értékei. (m, tabla[i, j]). Ezen felül tárol még 3 bool változót (p, e1, e2) melyeknek értékei egy adott játék betöltésekor inicializálódnak attól függően, hogy van-e játékosnak / enemy1-nek / enemy2-nek megfelelő értékű mező a táblán. Ezek helyétől függően állítunk a karaktereknek megfelelő pozíciókat. (_ppos, _e1pos, _e2pos) Ezekhez készített segéd fv.-kel. (setP, setE1, setE2)
    - A hosszútávú adattárolás lehetőségeit az IDataAcces interfész adja meg, amely lehetőséget ad a tábla betöltésére, (Load) valamint mentésére (SaveAsync) ez utóbbi hatékonyság miatt asszinkron módon van megolva.
    - Az interfészt szöveges fájl alapú adatkezelést az DataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a DataException kivétel jelzi.
    - A program az adatokat szöveges fájlként tudja eltárolni, melyek a .txt kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
    - A fájl első sora megadja a tábla méretét (ami alapértelmezés szerint 15). A fájl többi része izomorf leképezése a játéktáblának, azaz összesen 15 sor következik, és minden sor 15 számot tartalmaz szóközökkel választva. A számok 0 és 1 lehetnek, ahol 0 reprezentálja a szabad mezőt, 1 pedig az akna mezőt.



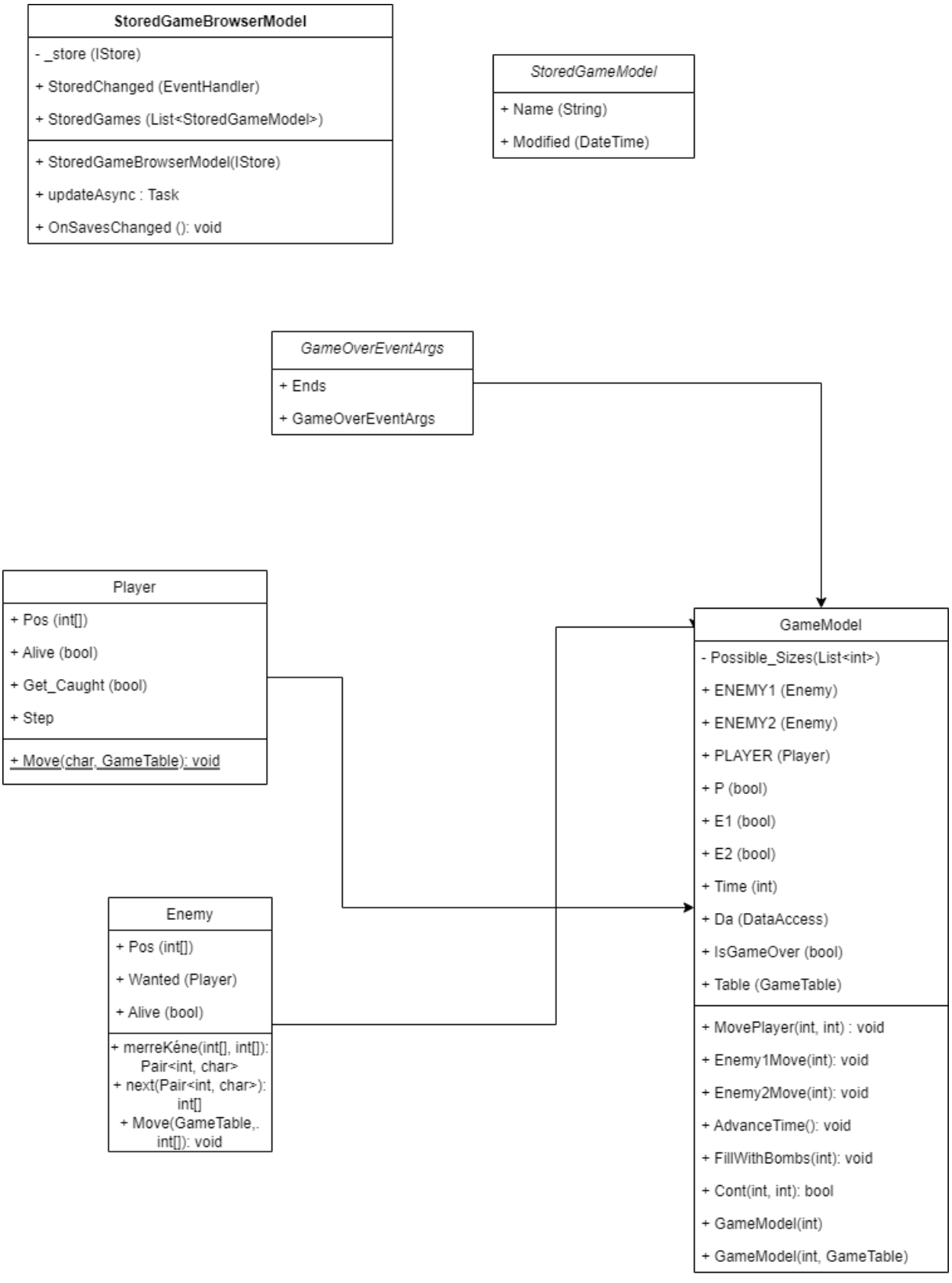
- 2.ábra: Az alkalmazás csomag diagrammja

- **Modell:**
    - A modell lényegi részét a három karakter osztály (Player, Enemy1, Enemy2) valósítja meg, (azok lépéseit) melyeket a GameModel osztály hasznosít és foglal egységbe. A három felsorolt típus metódusai a következők:
        - Player:
            - Move: a játékos mozgatása
            - Valamint a játékosnak van egy jelenlegi pozíciója: pos

- Egy igaz / hamis értéke amely igaz, ha a játékos még él: _isAlive
- Még egy arra, hogy elkapták-e már: _caught
- És még egy arra, hogy éppen lép-e: _step

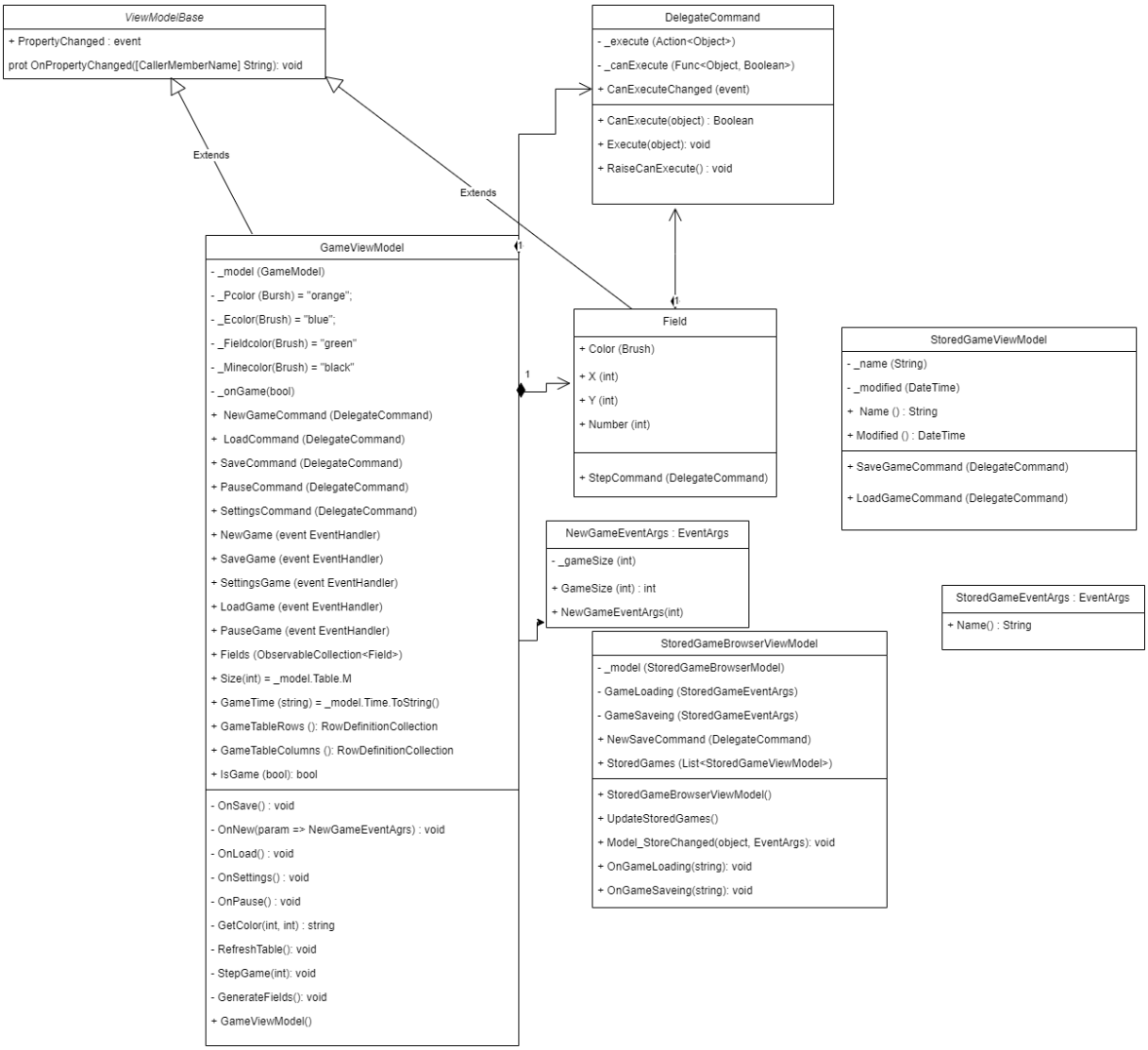
- Enemy1 és Enemy2:
  - merreKéne: egy sor/oszlop és abban pozitív/negatív irányt visszaadó metódus mely segéd fv.-e a következő metódusnak
  - next: megadja a következő mezőt amerre az üldöző mozog, felhasználva az előző metódust
  - Move: mozgatja az üldözőt
  - Az üldözőnek van egy pozíciója: pos
  - Egy játékost reprezentáló adattagja: player
  - És egy igaz/hamis értéke mely igaz, ha él még az üldöző: _isAlive

- Maga a GameModel pedig mint már írtam egybefoglalja ezeket a perzisztenciával.
  - Tárol egy perzisztencia típusú adattagot: da
  - Egy játéktáblát: _table
  - Egy játékost: _player
  - Két üldözőt: _enemy1, _enemy2
  - És 3 igaz/hamis értéket, melyek értéke igaz, ha a megfelelő karakter éppen létezik: van_p, van_e1, van_e2
  - A eseményei melyekkel jelzéseket ad a View-nak:
    - GameOver: értelemszerűen akkor váltódik ki, ha vége a játéknak
    - Step: Ha a játékos lép
  - Végül a metódusok melyeket használ:
    - OnGameOver, OnStep: a megfelelő események kiváltásait szolgálják
    - MovePlayer: A Player adattag Move fv.-ét használva kezeli a játékos mozgását.
    - Enemy1Move és Enemy2Move: A két üldözőt reprezentáló adattag Move fv.-eit meghívva mozgatja az üldözőket minden eltelt másodperc után.



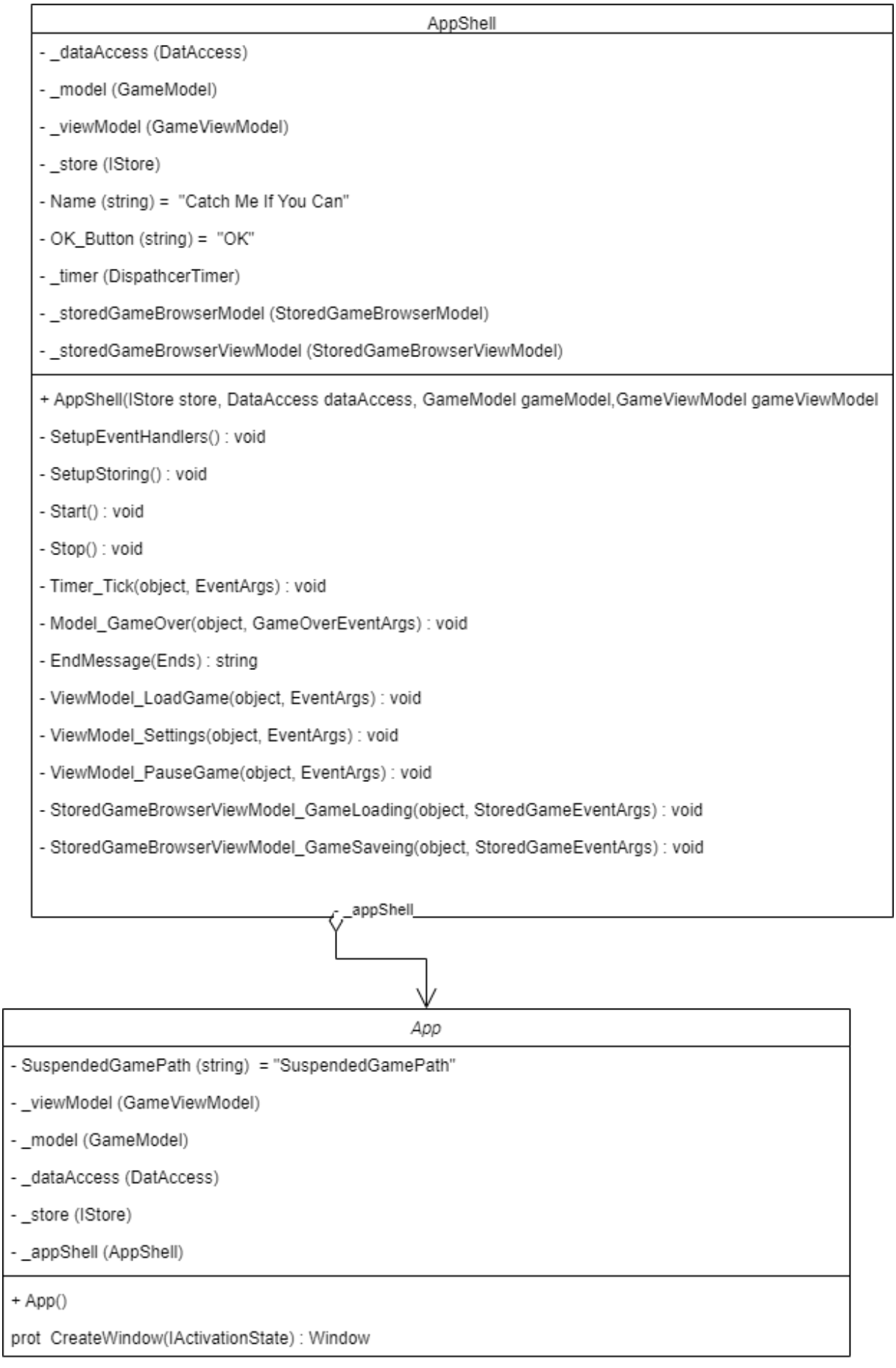
- Nézetmodell:
  - A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) osztályt.
  - A nézetmodell feladatait a GameViewModel osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, mentéséhez, 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 (Field), amely eltárolja a pozíciót, színt, engedélyezettséget, valamint a lépés parancsát (StepCommand). 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 Grid segítségével valósítunk meg, amelyben Button elemeket helyezünk el, valamint a különböző méretű új játékok indítására szolgáló gombokat.
  - A SettingsPage osztály tartalmazza a betöltés, mentés gombjait.
  - A LoadPage és a SavePage szolgál egy létező játékállapot betöltésére, illetve egy új mentésére.
- Vezérlés:
  - Az App osztály feladata az alkalmazás vezérlése, a rétegek példányosítása és az események feldolgozása.

- A CreateWindow metódus felüldefiniálásával kezeljük az alkalmazás életciklusát a megfelelő eseményekre történő feliratkozással. Így az alkalmazás felfüggesztéskor (Stopped) elmentjük az aktuális játékállást (SuspendedGame), míg folytatáskor vagy újraindításkor (Activated) pedig folytatjuk, amennyiben történt mentés.
- Az alkalmazás lapjait egy AppShell keretben helyezzük el. Ez az osztály felelős a lapok közötti navigációk megvalósításáért.


AppShell
- _dataAccess (DatAccess)
- _model (GameModel)
- _viewModel (GameViewModel)
- _store (IStore)
- Name (string) = "Catch Me If You Can"
- OK_Button (string) = "OK"
- _timer (DispathcerTimer)
- _storedGameBrowserModel (StoredGameBrowserModel)
- _storedGameBrowserViewModel (StoredGameBrowserViewModel)
+ AppShell(IStore store, DataAccess dataAccess, GameModel gameModel,GameViewModel gameViewModel
- SetupEventHandlers() : void
- SetupStoring() : void
- Start() : void
- Stop() : void
- Timer_Tick(object, EventArgs) : void
- Model_GameOver(object, GameOverEventArgs) : void
- EndMessage(Ends) : string
- ViewModel_LoadGame(object, EventArgs) : void
- ViewModel_Settings(object, EventArgs) : void
- ViewModel_PauseGame(object, EventArgs) : void
- StoredGameBrowserViewModel_GameLoading(object, StoredGameEventArgs) : void
- StoredGameBrowserViewModel_GameSaveing(object, StoredGameEventArgs) : void
_appShell
App
- SuspendedGamePath (string) = "SuspendedGamePath"
- _viewModel (GameViewModel)
- _model (GameModel)
- _dataAccess (DatAccess)
- _store (IStore)
- _appShell (AppShell)
+ App()
prot CreateWindow(IActivationState) : Window

Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a Test osztályban.
- Az alábbi tesztesetek kerültek megvalósításra:
    - Initalize11, Initalize15, Initalize21: Megfelelő méretű random játéktáblák inicializása
    - Load11, Load15, Load21: A megfelelő méretű betöltött játéktáblák Player.Pos, Enemy1.Pos, Enemy2.Pos pozíciók értékeinek összevetése a random generált táblák ugyanezen pozíciókon lévő értékeivel
    - EnemiesMove_And_Die_GameOver_Test_(11, 15, 21): Megfelelő méretű táblákon az üldözők mozgásának ellenőrzése pozíció váltás utáni érték ellenőrzésével. Majd aknára lépésük után isGameOver igaz/hamis érték igazra állítódásának ellenőrzése.
    - PlayerMove_And_PlayerDie_GameOver_Test_(11, 15, 21): Megfelelő méretű táblákon a játékos mozgásának ellenőrzése pozícióváltás utáni értékek ellenőrzésével. Majd aknára lépése után isGameOver igaz/hamis érték igazra állítódásának ellenőrzése.