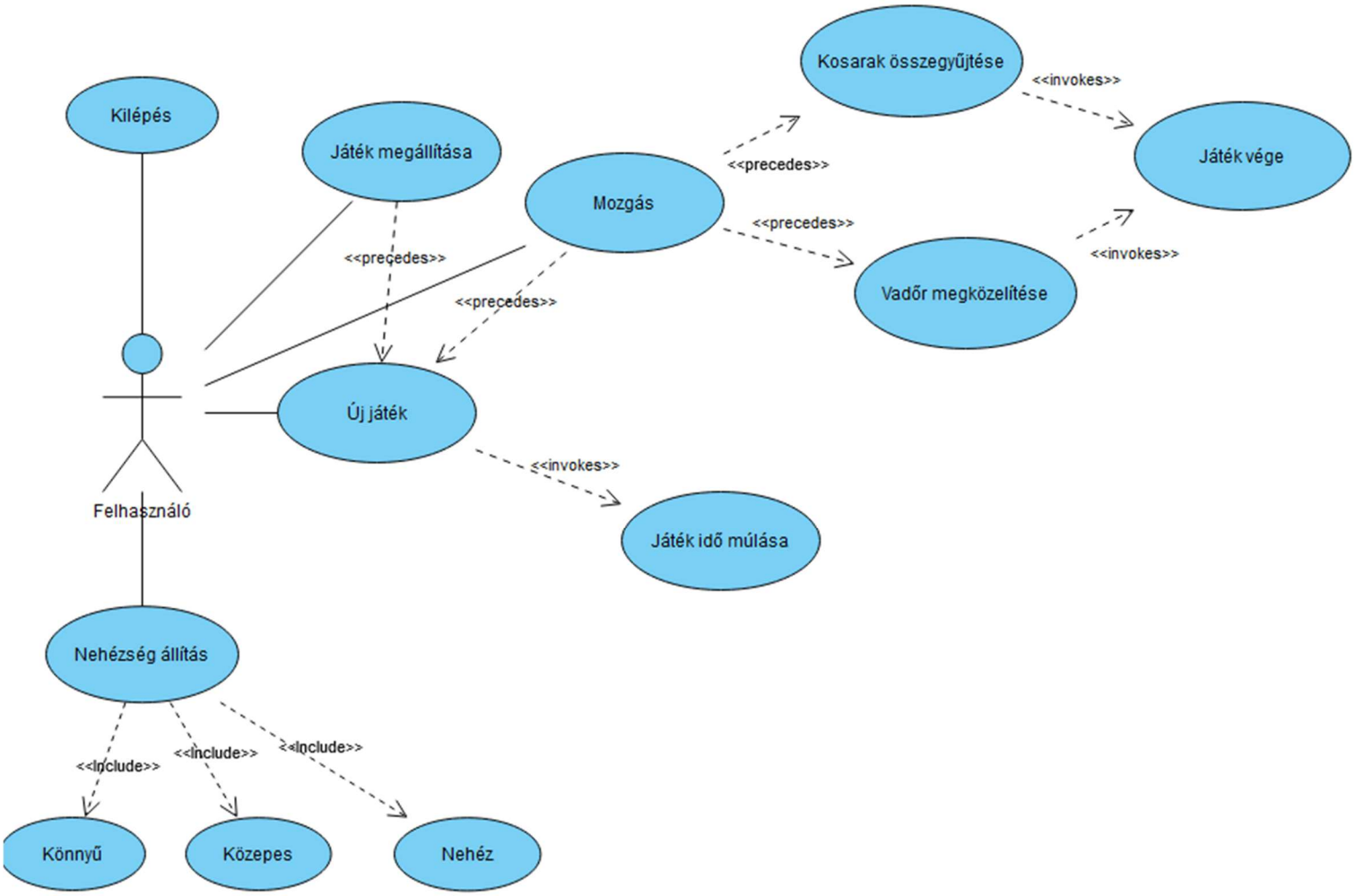
Feladat:

Készítsünk programot, amellyel a következő játékot játszhatjuk. Adott egy $n \times n$ elemből álló játékpálya, amelyben Maci Lacival kell piknikkosarakra vadásznunk. A játékpályán az egyszerű mezők mellett
elhelyezkednek akadályok (pl. fa), valamint piknikkosarak. A játék célja, hogy a piknikkosarakat minél gyorsabban begyűjtsük.
Az erdőben vadőrök is járőröznek, akik adott időközönként lépnek egy mezőt (vízszintesen, vagy függőlegesen). A járőrözés során egy megadott irányba haladnak egészen addig, amíg akadályba (vagy az erdő szélébe) nem ütköznek, ekkor megfordulnak, és visszafelé haladnak (tehát folyamatosan egy vonalban járőröznek). A vadőr járőrözés közben a vele szomszédos mezőket látja (átlósan is, azaz egy 3 × 3-as négyzetet).
A játékos kezdetben a bal felső sarokban helyezkedik el, és vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán, a piknikkosárra való rálépéssel pedig felveheti azt. Ha Maci Lacit meglátja valamelyik vadőr, akkor a játékos veszít.
A pályák méretét, illetve felépítését (piknikkosarak, akadályok, vadőrök
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). Ismerje fel, ha vége a játéknak, és jelezze, győzött, vagy veszített a játékos. A program játék közben folyamatosan jelezze ki a játékidőt, valamint a megszerzett piknikkosarak számát.

Elemzés:

- A feladatot egyablakos asztali alkalmazásként WPF grafikus felülettel valósítjuk meg

- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: Nehézség (Könnyű, Közepes, Nehéz), Játék. Az ablak alján megjelenítünk egy státuszsort, amely a összegyűjtött piknikk kosarak számát, illetve az eltelt időt számolja.
- A könnyű, nehéz, közepes három különböző pályát takar amelyek méreteikben is különböznek.
- A játéktábla panelokból áll, a füves panelek zöldek. Az akadályok szürkék, a kosarak sárgák, a vadőrök pirosak, Maci Laci pedig világos kék.
- A karakterünk gombokkal irányítható. A játékot a pause gombra való kattintással állíthatjuk meg és szintén a így folytathatjuk.
    - A felhasználói esetek az 1. ábrán láthatóak.

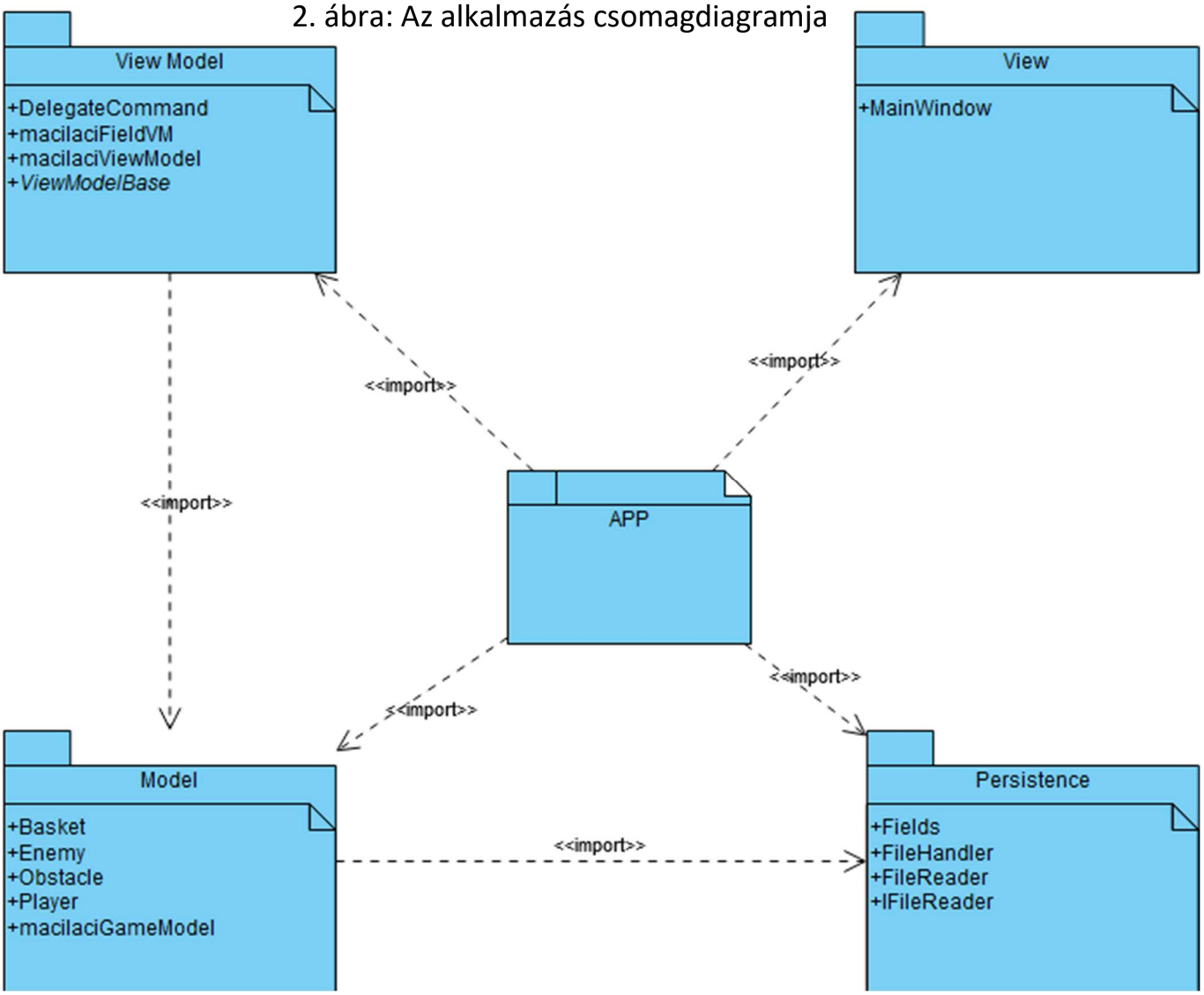


1. Ábra: Felhasználói esetek diagramja

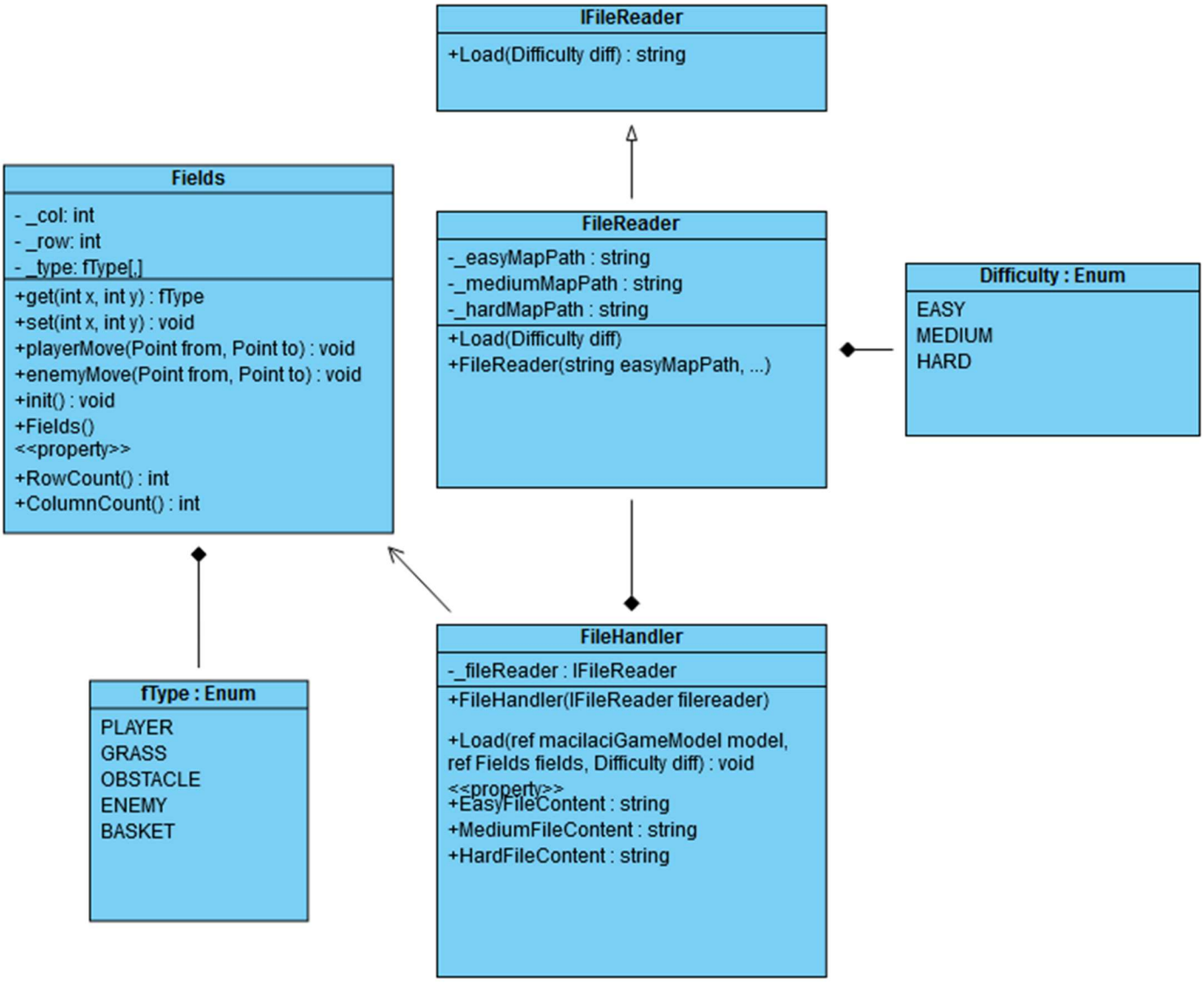
Tervezés:

- Programszerkezet:
  - A programot MVVM architektúrában valósítjuk meg. Ennek megfelelően a `View`, `ViewModel`, a `Model`, a perzisztencia a `Persistence` névtereket kell megvalósítanunk. A program csomagszerkezete a 2. ábrán látható.
  - A program szerkezetét két projektre osztjuk implementációs megfontolásból: a `Persistence` és `Model` csomagok a program felületfüggetlen projektjében, míg a `View` csomag a Windows Formstól függő projektjében kap helyet.
  - A program szerkezetét két projektre osztjuk implementációs megfontolásból: a `Persistence` és `Model` csomagok a program felületfüggetlen projektjében, míg a `ViewModel` és `View` csomagok a WPF függő projektjében kapnak helyet.

2. ábra: Az alkalmazás csomagdiagramja



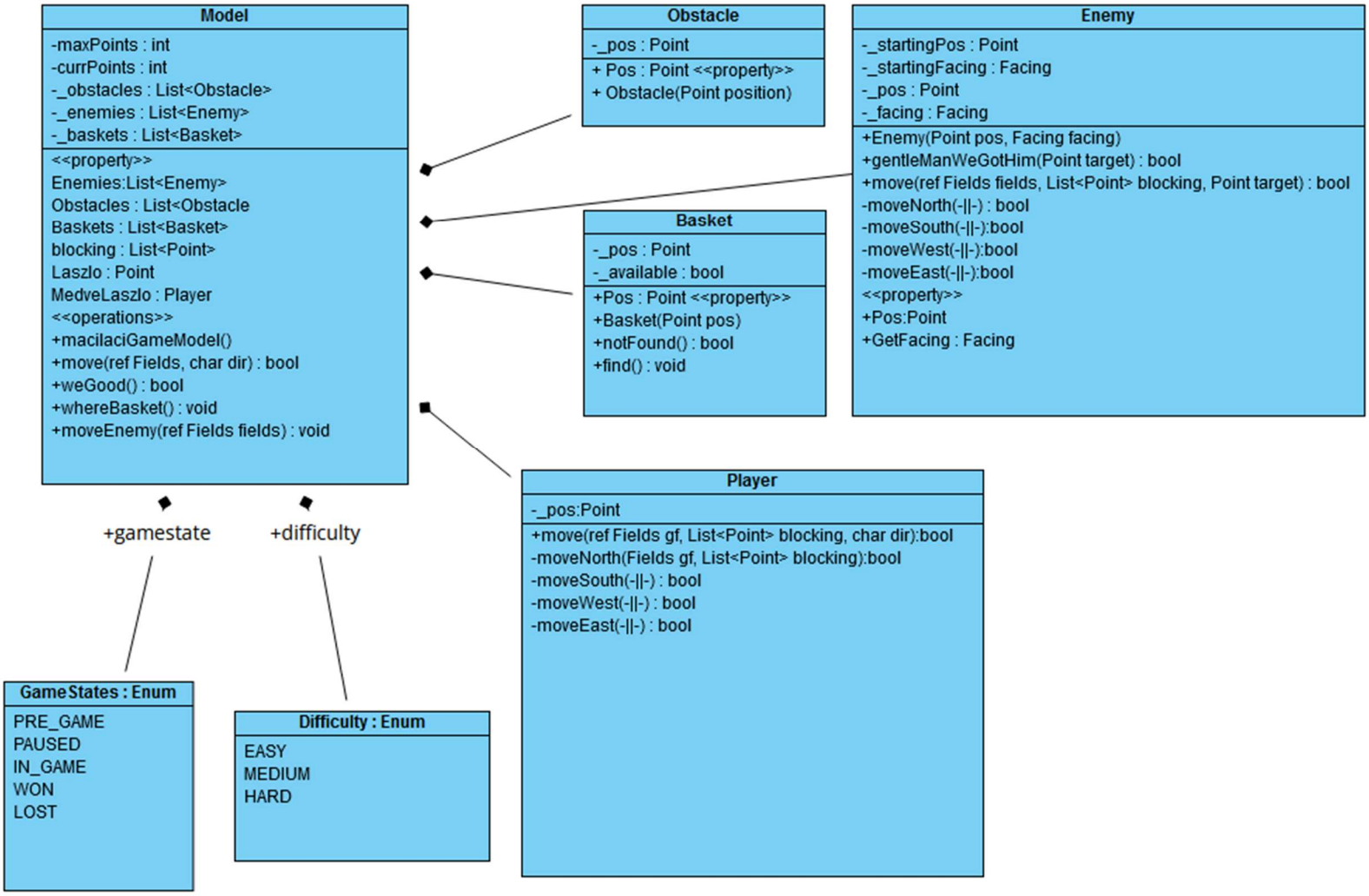
- Perszisztencia:
  - Az adatkezelés feladata a táblával kapcsolatos információk tárolása
  - A `Field` osztály egy két dimenziós tömbben tárolja a játék objektumokat. Amelyik mezőn nincsen objektum ott `null` értéket tárolunk.
  - Az interfészt szöveges fájl alapú fájl olvasást a `FileReader` osztály valósítja meg. A fájlokból olvasott információt pedig a `FileHandler` kezeli. A fájlkezelés során fellépő hibákat a `FileReaderException` kivétel jelzi.
  - A 3 különböző pályát 3 txt file tárolja. (Az exampleMap.txt pedig a fájlokban tárolt adatok eloszlását írja le.)
  - A fileHandler felel az osztályok inicializálásáért. Ami a `load(…)` metódusban történik
  - A fájlokban az első 2 szám a páya mérete (x, y). A második sor az akadályok számát. A következő sorok az akadályok koordinátáit. Ezekután megint egy egyedülálló szám következik amely a vadőrök számát tartalmazz majd ezeknek a koordinátái következnek illetve az irány amelybe néznek. Az utolsó részleg a kosarak számát majd az ezután következő sorok a koordinátákat tartalmazz.



3. ábra: Persistence csomag osztálydiagramja

- Modell:
  - A modell tartalmazza az összes játék objektum osztályát:
    - Játékos (`Player`)
    - Kosár (`Basket`)
    - Akadály (`Obstacle`)
    - Vadőr (`Enemy`)
  - Ezek az osztályok pedig a `IGameObject` osztályból származnak, közös tulajdonságuk a pozíció illetve a szín.

- A vadőr illetve a játékos osztályban a `move(…)` metódus felel a mozgásért amelyek több alosztállyal döntik el az irányt illetve, hogy lehetséges-e a mozgás.

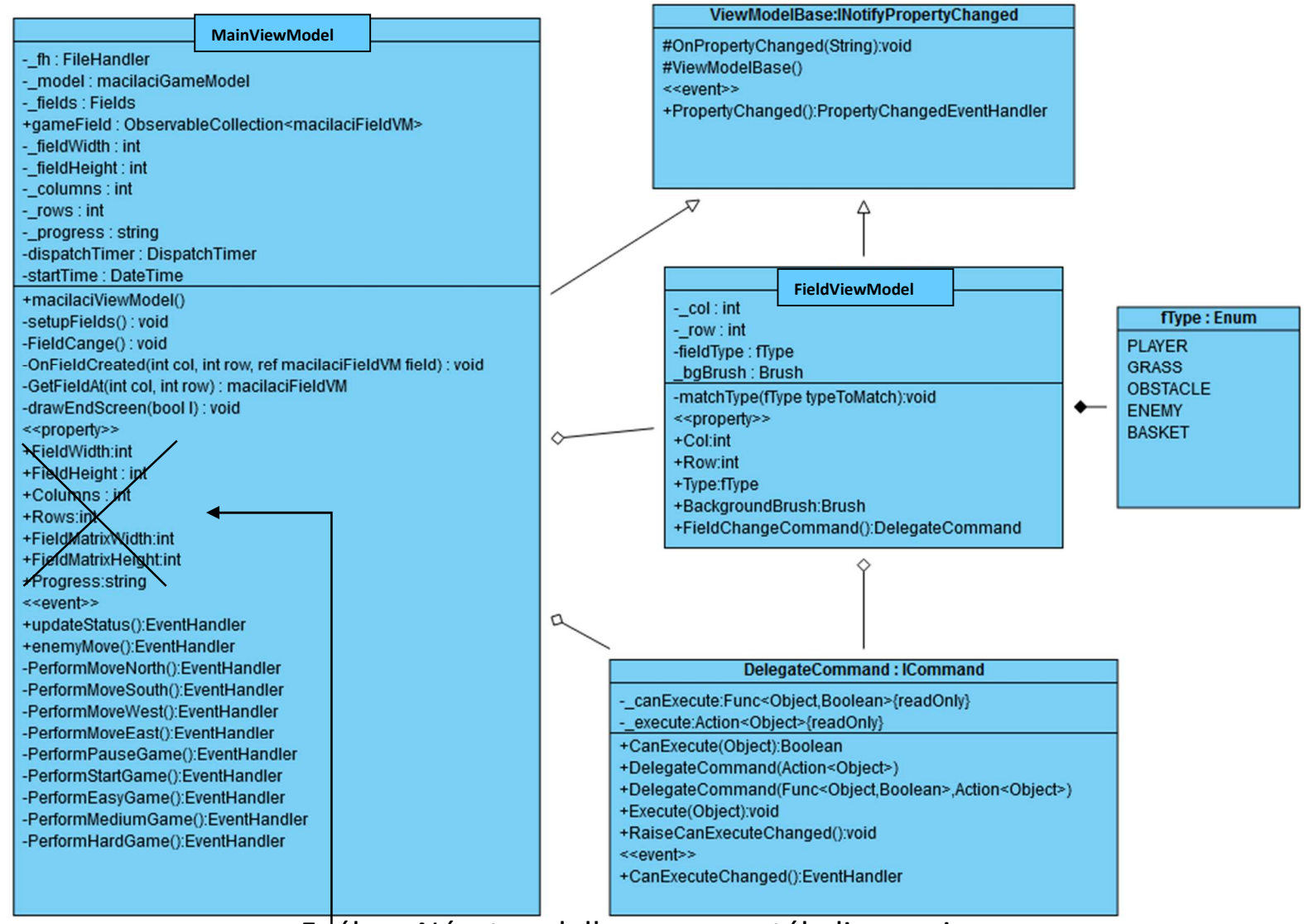


4. ábra: Modell csomag osztálydiagramja

- Nézetmodell:
  - A nézetmodell megvalósításához felhasználunk egy általános utasítás (`DelegateCommand`), valamint egy ős változásjelző (`ViewModelBase`) osztályt.
  - A nézetmodell feladatait a `macilaciViewModel` 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, illetve a játéknehézséget szabályozza. Direkt nem avatkozik a játék futtatásába.

- A játékmező számára egy külön mezőt biztosítunk (`macilaciFieldVM`), amely eltárolja a különböző mezők tulajdonságait. A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (`Fields`)



5. ábra: Nézetmodell csomag osztálydiagramja

- Nézet:
  - A nézet cask egy képernyőt tartalmaz, a `MainWindow` osztályt. A nézet egy rácsban tárolja a játékmezőt, a menüt és a státuszsort. A játékmező egy

`ItemsControl` vezérlő, ahol dinamikusan felépítünk egy rácsot (`UniformGrid`), amely négyszögekből áll. Minden adatot adatkötéssel
kapcsolunk a felülethez, továbbá azon keresztül szabályozzuk a négyszögek színét is.

- Környezet:
  - Az App osztály feladata az egyes rétegek példányosítása (`App_Startup`), összekötése, a nézetmodell, valamint a modell eseményeinek lekezelése, és ezáltal a játék, az adatkezelés, valamint a nézetek szabályozása.

6. ábra: A vezétlés osztálydiagramja

Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a SudokuGameModelTest osztályban.
- Az alábbi tesztesetek kerültek megvalósításra:
  - `TestLoadEasy`
    `TestLoadMedium`
    `TestLoadHard`: Új játék létrehozása objektumok elhelyezése.
  - `MaciLaciStartingPosition`: Játékos kezdő poziciójának ellenőrzése
  - `MaciLaciMovePosition`: Maci Laci mozgás utáni poziciójának ellenőrzése
  - `EnemyMove`: Vadőr mozgásának ellenőrzése
  - `BasketsUnFound`: Kosár kezdeti állapotának ellenőrzése

- `BasketWhenFound`: Ellenőrzi, hogy kosárra való lépés után állapotot vált-e a kosár