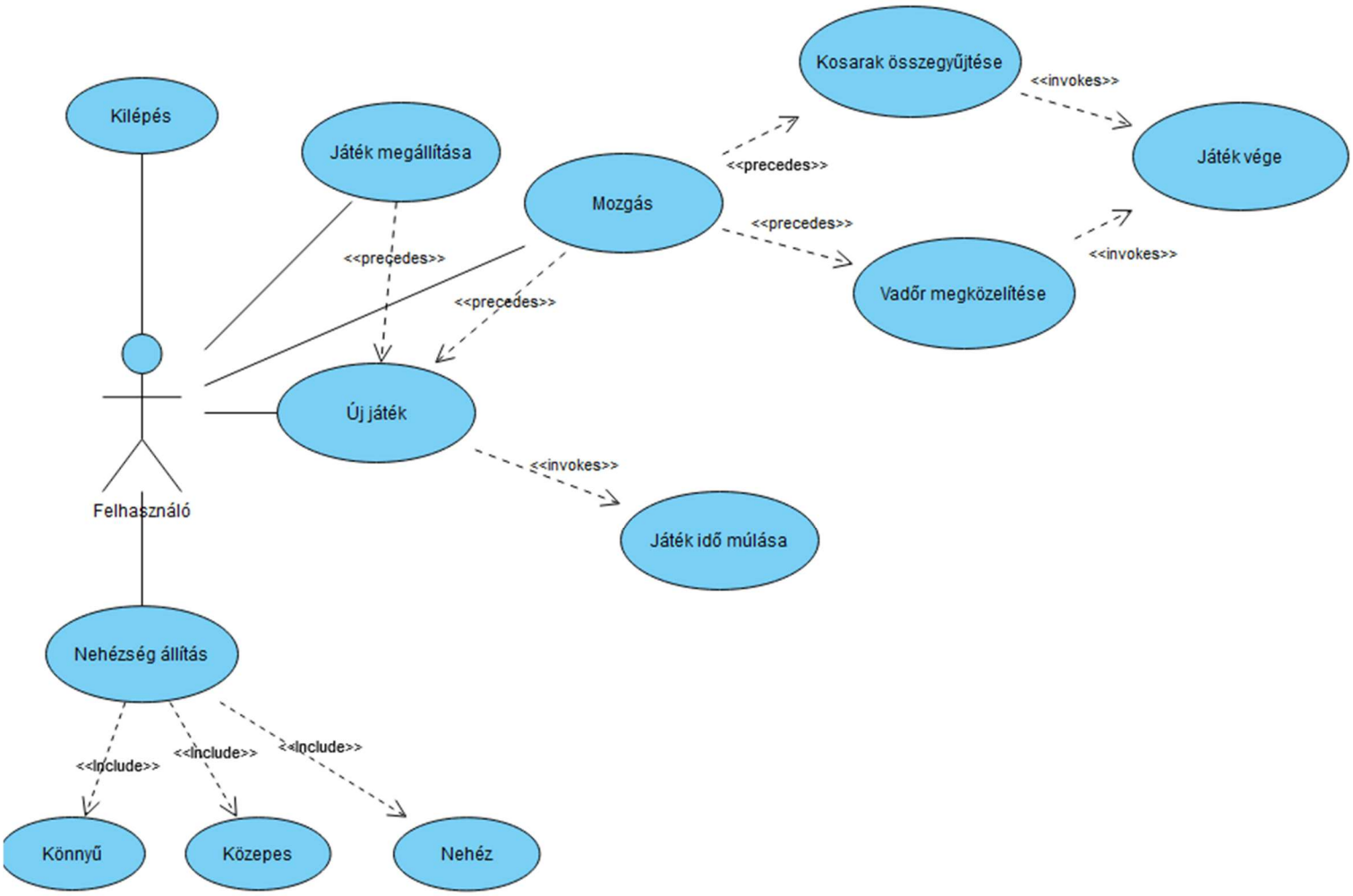
Feladat:

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 Windows Forms 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 a, w, s, d gombokkal irányítható. A játékot a szóköz billentyű lenyomásával állíthatjuk meg és szintén a szóközzel folytathatjuk.
    - A felhasználói esetek az 1. ábrán láthatóak.

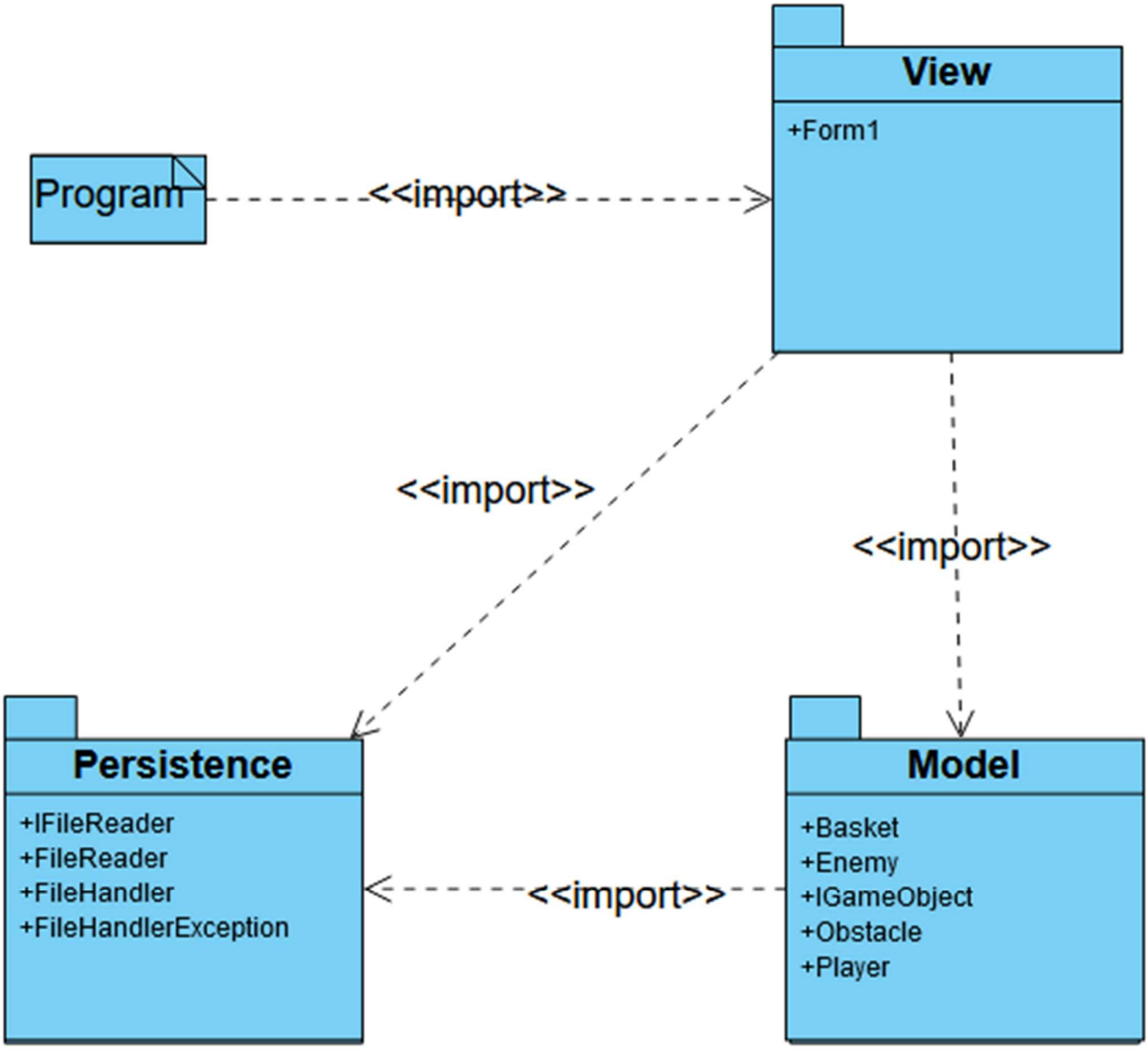


1. Ábra: Felhasználói esetek diagramja

Tervezés:

- Programszerkezet:
  - A programot háromrétegű architektúrában valósíjuk meg. A megjelenítés a `View`, a modell a `Model`, míg a perzisztencia a `Persistence` névtérben helyezkedik el. 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.
- 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

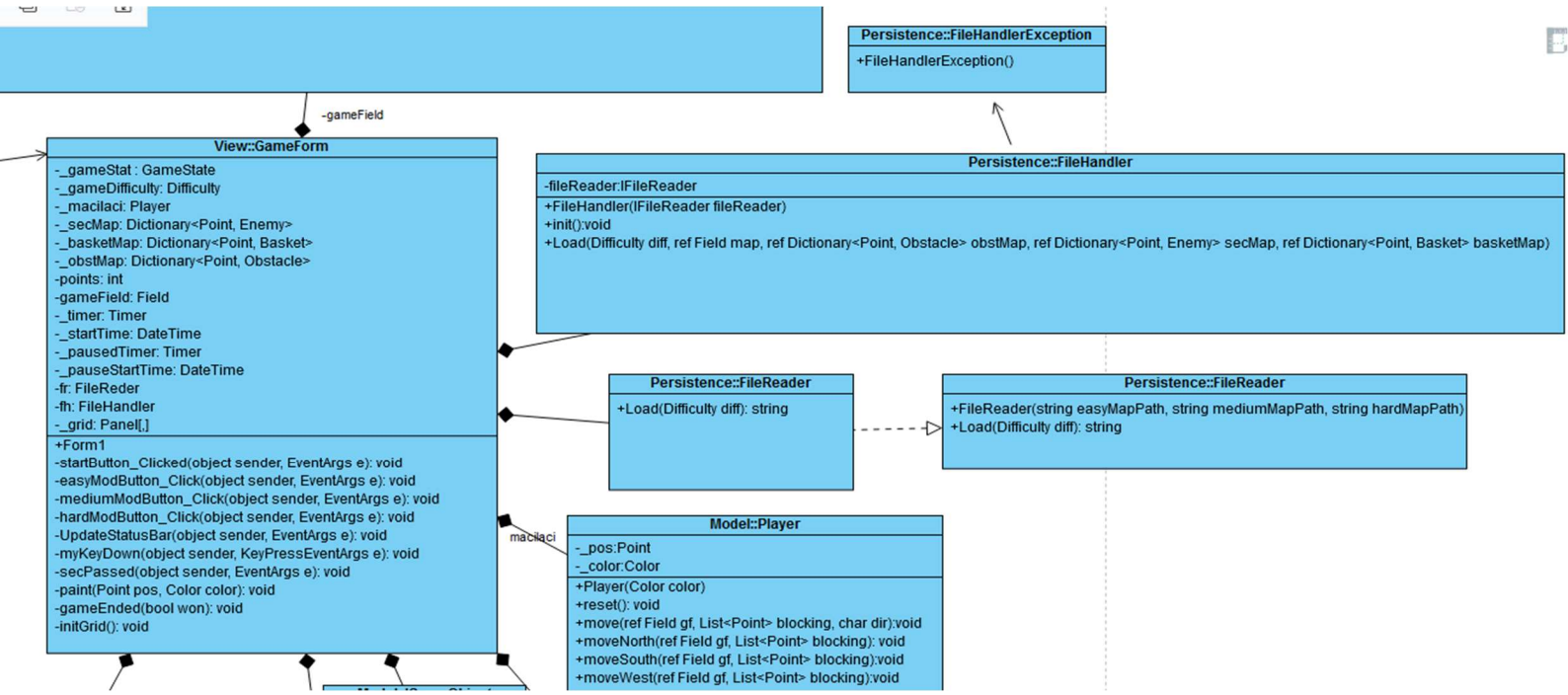
- A fájlokban az első 2 szám a páya mérete (x, y). A második sor az akadályok számát. A következő sorok az akadályok koordinátáit. Ezekután megint egy egyedülálló szám következik amely a vadőrök számát tartalmazz majd ezeknek a koordinátái következnek illetve az irány amelybe néznek. Az utolsó részleg a kosarak számát majd az ezután következő sorok a koordinátákat tartalmazz.



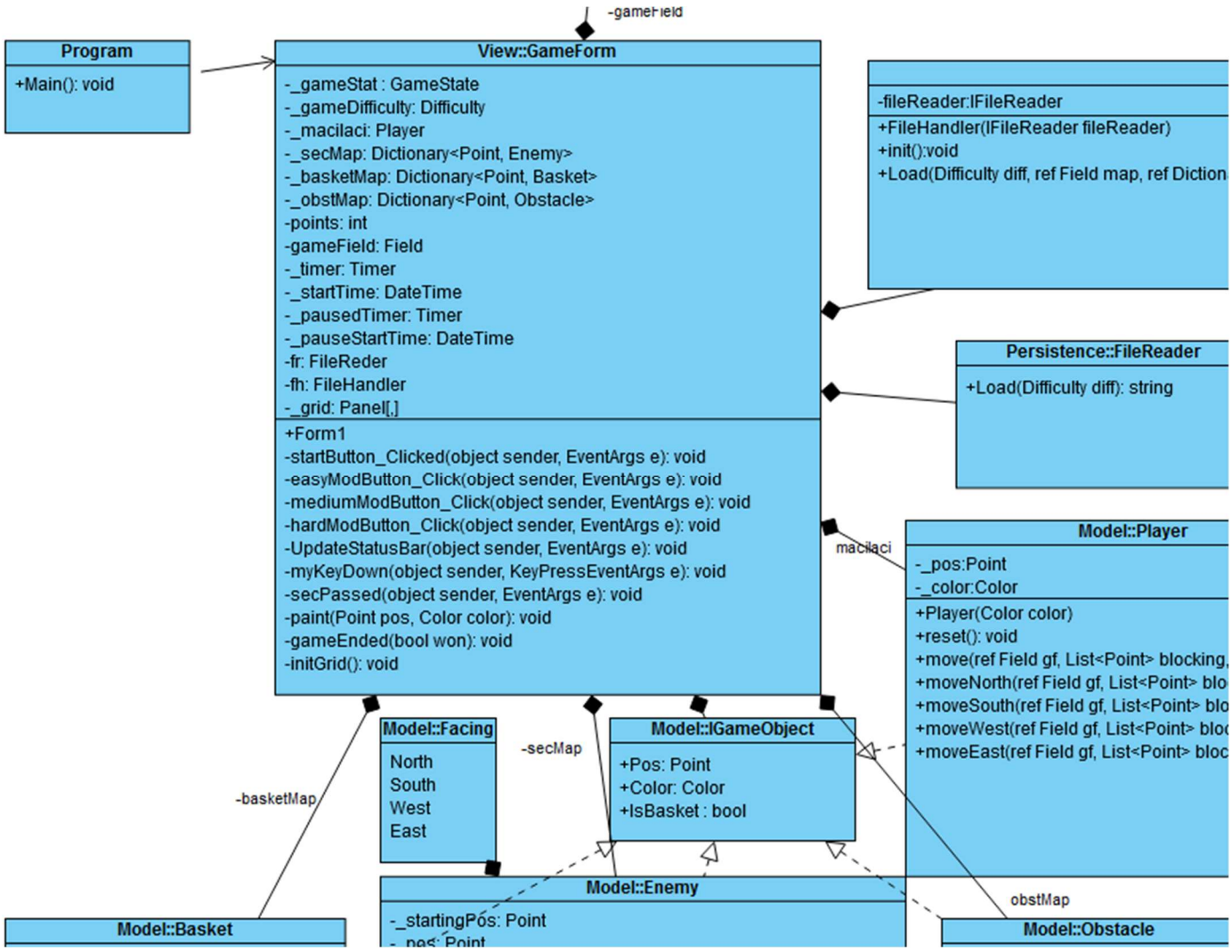
2. ábra: Az alkalmazás csomagdiagramja

- 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.
- Nézet:
    - A nézetet a `Form1` osztály biztosítja, amely tárolja az objektumokat Dicitonary adattagokban, illetve egy Field típusú pályát.
    - A Form1 felelős a játék státusz kezeléséért.
    - A játéktáblát egy dinamikusan létrehozott gombmező (`_grid`) reprezentálja. A felületen létrehozzuk a megfelelő menüpontokat, illetve státuszsort, valamint dialógusablakokat, és a hozzájuk tartozó eseménykezelőket. A játéktábla generálását, illetve az értékek beállítását (`initGrid`) végzi.
    - A játék időbeli kezelését egy időzítő végzi (`_timer`), amelyet mindig aktiválunk játék során, illetve inaktiválunk, amennyiben bizonyos menüfunkciók futnak.

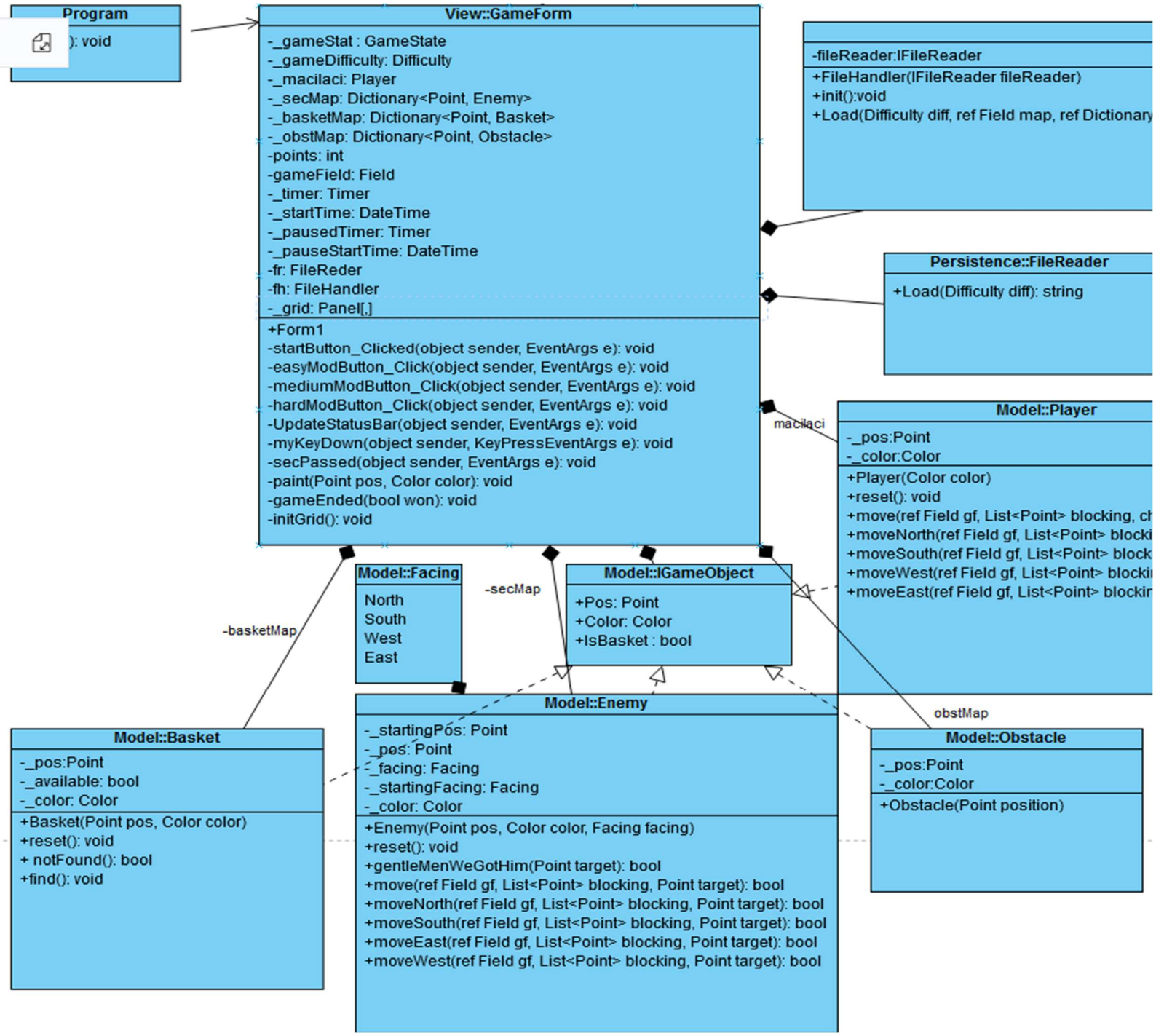


Perszisztencia réteg



Nézet réteg

Model réteg



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