

Második beadandó feladat

Készítette

Hujber Ferenc Kristóf

BWSOU0@INF.ELTE.HU

Feladat:

Aknamező

Készítsünk programot a következő játékra.

A játékban egy tengeralattjárót kell irányítanunk a képernyőn (balra, jobbra, fel, illetve le), amely felett ellenséges hajók köröznek, és folyamatosan aknákat dobnak a tengerbe. Az aknáknak három típusa van (könnyű, közepes, nehéz), amely meghatározza, hogy milyen gyorsan süllyednek a vízben (minél nehezebb, annál gyorsabban).

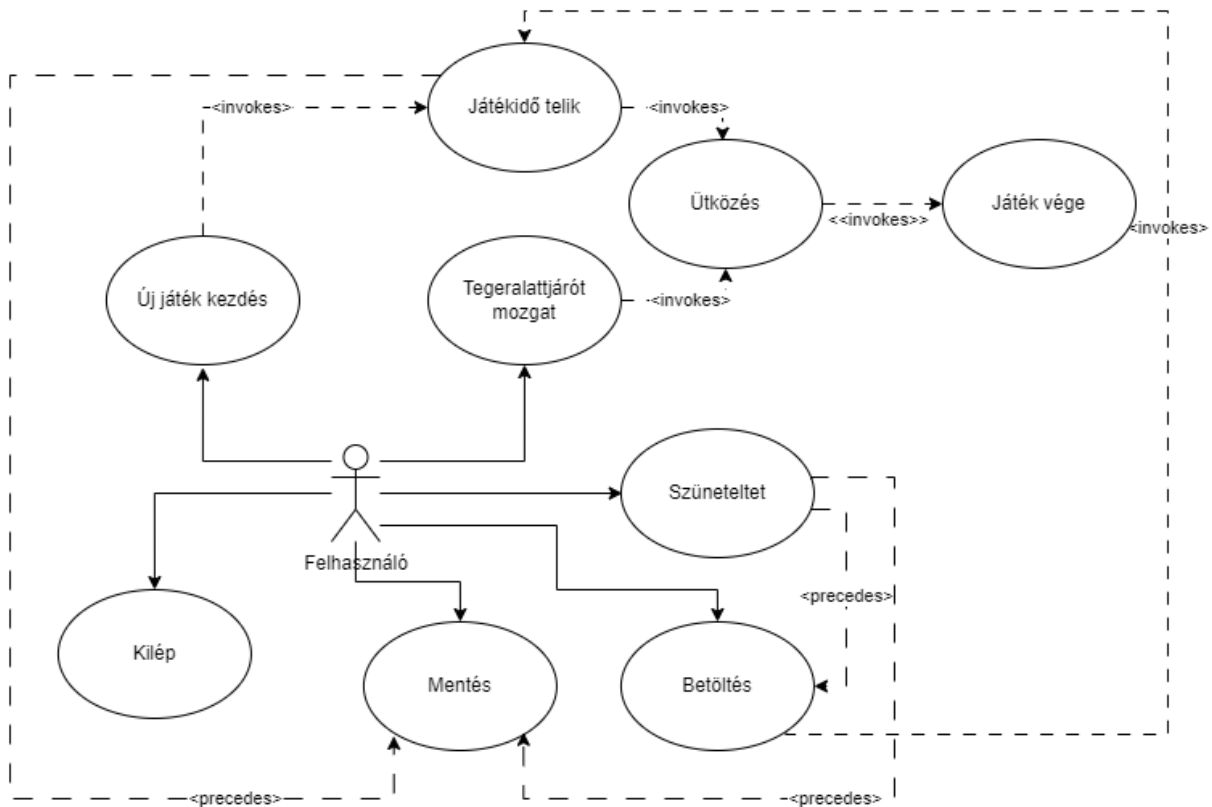
Az aknákat véletlenszerűen dobják a tengerbe, ám mivel a hajóskapitányok egyre türelmetlenebbek, egyre gyorsabban kerül egyre több akna a vízbe. A játékos célja az, hogy minél tovább elkerülje az aknákat. A játék addig tart, ameddig a tengeralattjárót el nem találta egy akna.

A program biztosítson lehetőséget új játék kezdésére, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem mozog semmi a játékban). Ismerje fel, ha vége a játéknak, és jelenítse meg, mennyi volt a játékidő. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére.

Elemzés:

- Megvalósítás egyablakos WPF applikációként, grafikus felülettel.
- A játék indításakor a játékos új játékot indíthat.
- A felhasználó tudja a játékot szüneteltetni. Ekkor a következő menüpontokat veheti igénybe: mentés, betöltés, új játék indítása. A játék futása közben nincs szükség a menü elérhetőségére.
- A felhasználó tudja mozgatni a tengeralattjárót (azaz a játékos karaktert) a játéktéren.

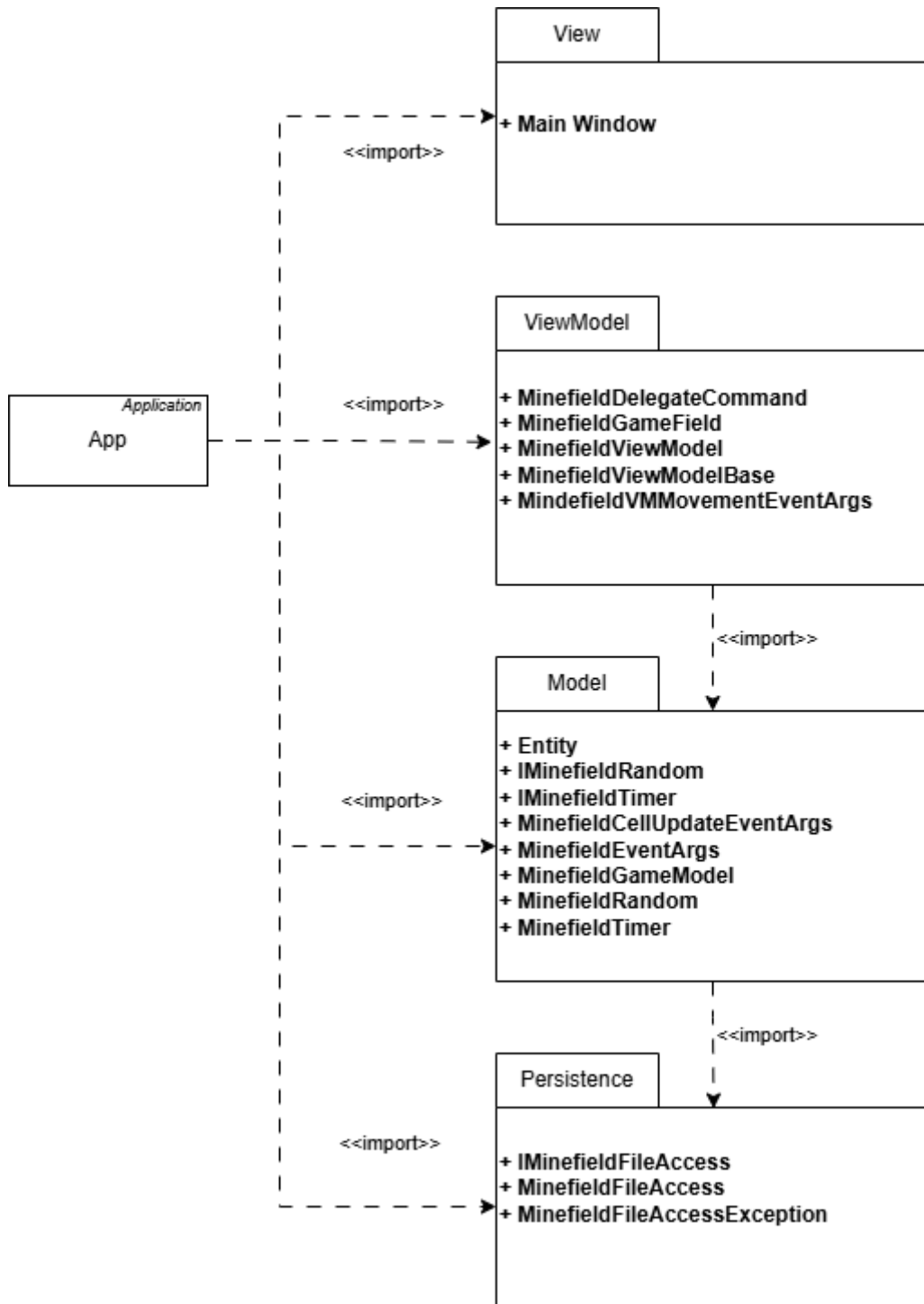
- A játék véget érhet a következő esetekben:
 - o A tengeralattjáró nekimegy egy aknának.
 - o Az idő teltével egy akna a tengeralattjáróra esik.
- A használati esetekért lásd az 1. ábrát.



1. ábra: Használati eset diagram

Tervezés:

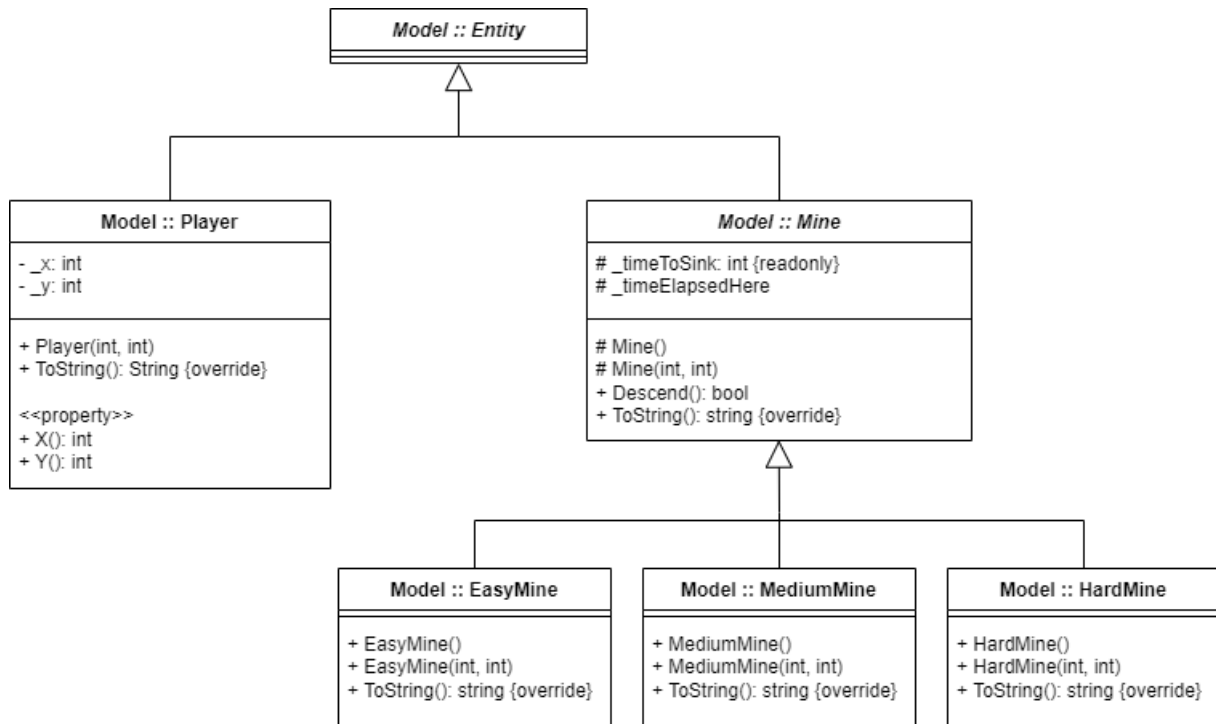
- Programszerkezet: MVVM architektúra:
 - o Felületfüggetlen:
 - **modell:** Minefield.Model
 - **perzisztencia:** Minefield.Persistence
 - o WPF:
 - **nézet:** Minefield.WPF.View
 - **nézetmodell:** Minefield.WPF.ViewModel
 - o Rétegeket példányosító és összekötő környezet: **App** osztály
- A csomagdiagrammért lásd a 2. ábrát.



2. ábra: Csomagdiagram

Modell

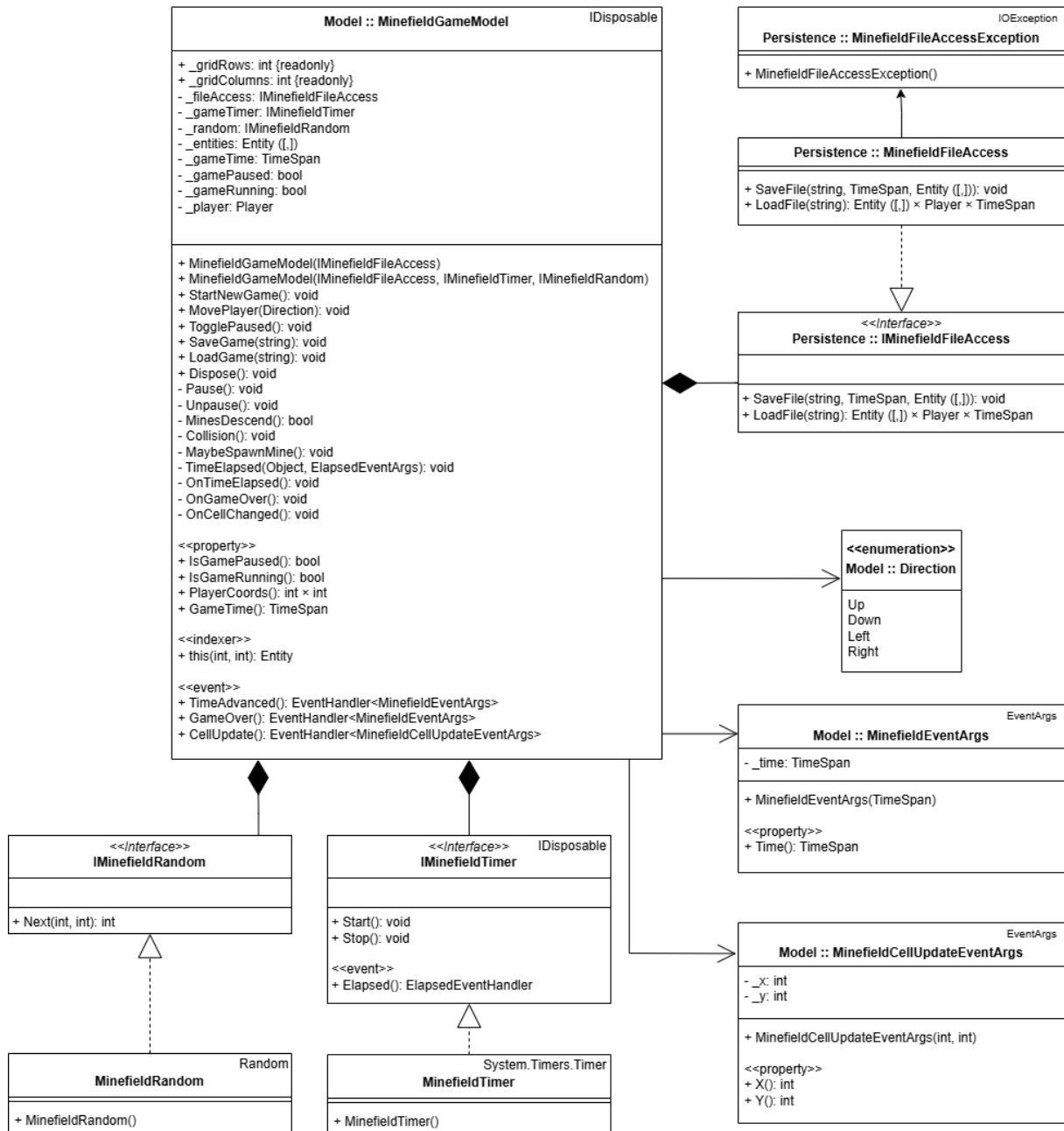
- Az üzleti logikát és a játéktér reprezentációját a MinefieldGameModel osztály tartalmazza.
- Az üzleti logika legfontosabb elemei:
 - Játék indítása, szüneteltetése, leállítása, mentés/betöltés kezelése
 - A játéktér változásainak kezelése, mint a játékos karakter mozgatása, akna létrehozása és mozgatása, ütközés ellenőrzése
 - Időzítő eseményének kezelése
- A játéktér egy Entitásmátrix (_entities) és a játékidő (_gameTime) reprezentálja. Ezen felül tárolunk a játékos entitásra egy referenciát (_player) annak gyors eléréséhez a mozgatás esetén.
- Az entitás típus és leszármazottai:
 - A játéktér résztvevőinek absztrakt őssztálya az Entity. Ennek Leszármazottja a Player, azaz a játékos karakter típusa és a Mine, azaz az akna típusa.
 - A Mine absztrakt szülőosztály. Ennek alosztályai a ténylegesen példányosított EasyMine, MediumMine, HardMine.
 - Az Entity típusok osztálydiagramjaiért lásd a 3. ábrát.
- A modell események által kommunikál a nézet felé.
 - TimeAdvanced: A futó játékban a játékidő haladásáról tájékoztat.
 - GameOver: A játék végéről tájékoztat, ami ütközéskor keletkezik be.
 - CellUpdate: A játéktér egy cellájának változásáról tájékoztat.
- Esemény-argumentum típusok:
 - MinefieldEventArgs: Tartalmazza a játékidőt.
 - MinefieldCellUpdateEventArgs: Tartalmazza az adott mező koordinátáit.
- MinefieldGameModel implementálja az IDisposable interfészt, mert a _gameTimer field-je Disposable.
- A modell 3-paraméteres konstruktora lehetőséget ad arra, hogy függőségi befecskendezéssel adjuk meg az időzítőt és a véletlenszám-generátort. Ezt mock teszteléskor használjuk ki.



3. ábra: Entity osztály és leszármazottainak osztálydiagramja

Perzisztencia

- Feladata a játéállás mentésének betöltése, valamint egy létező mentés-fájlból a játéállás beolvasása.
- A mentés és betöltés metódusait az IMinefieldFileAccess interfész adja meg:
 - SaveFile
 - LoadFile
- Az utóbbi fájlkezelő metódusok megvalósítását a MinefieldFileAccess tartalmazza.
- A fájlkezelés közben fellépő hiba esetén MinefieldFileAccessException kivétel váltódik ki, ami az IOException leszármazottja.
- Egy mentés-fájl (.MFS mint MineFieldSave) szerkezete a következő soronként:
 - játékidő milliszekundumban
 - játéktér sorszáma
 - játéktér oszlopszáma
 - a játéktér entitás-mátrixának elemei sorfolytonosan (1 entitás / 1 sor a fájlban) a következő szerint
 - 0 ha null
 - az entitás string-reprezentációja ha nem null
 - lásd: felüldefiniált ToString() metódusok



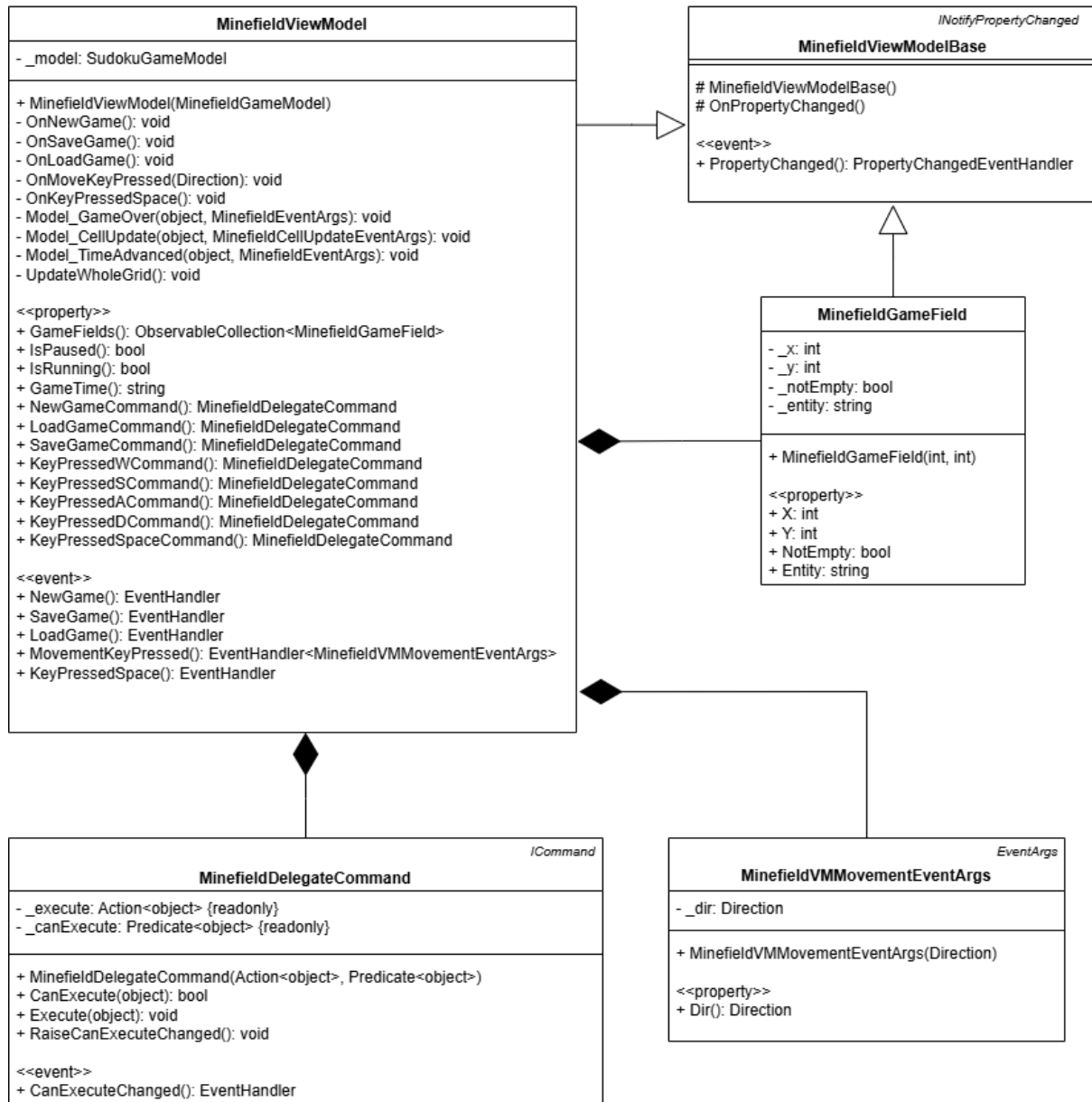
4. ábra: Modell és perzisztencia osztálydiagramja

Nézetmodell

- A nézetmodell és a játéklemező osztályát a **MinefieldViewModelBase** őssztályból származtatjuk, ami implementálja az **INotifyPropertyChanged** interfészt.
- A felhasználói vezérléséhez egy általános parancs osztályt használunk: **MinefieldDelegateCommand**, ami implementálja az **ICommand** interfészt. Így a nézetben egy megfelelő parancsot tudunk kötni például egy billentyű lenyomásához vagy egy menüpontra kattintáshoz.
- A nézetmodell elemeit a **MinefieldViewModel** osztály tartalmazza.
 - Parancsokat definiál.
 - A parancsokhoz eseményeket rendel.
 - A modell eseményeit lekezeli.
 - Tárolja a modell egy hivatkozását, de csak adatot olvas ki belőle, például futás állapota, indexer.
 - Eseményeket vált ki, amiket a környezet kezel le. A mozgás vezérlés esemény argumentumának típusa **MinefieldVMMovementEventArgs**, ami a mozgás irányát tartalmazza.
- A játéktér megjelenítéséhez szükséges adatokat egy ObservableCollection tartalmazza. A gyűjtemény elemeinek típusa **MinefieldGameField**, ami tartalmazza egy mező koordinátáit (X, Y) és tartalmát (NotEmpty, Entity).
- Az osztálydiagrammért lásd az 5. ábrát.

Nézet

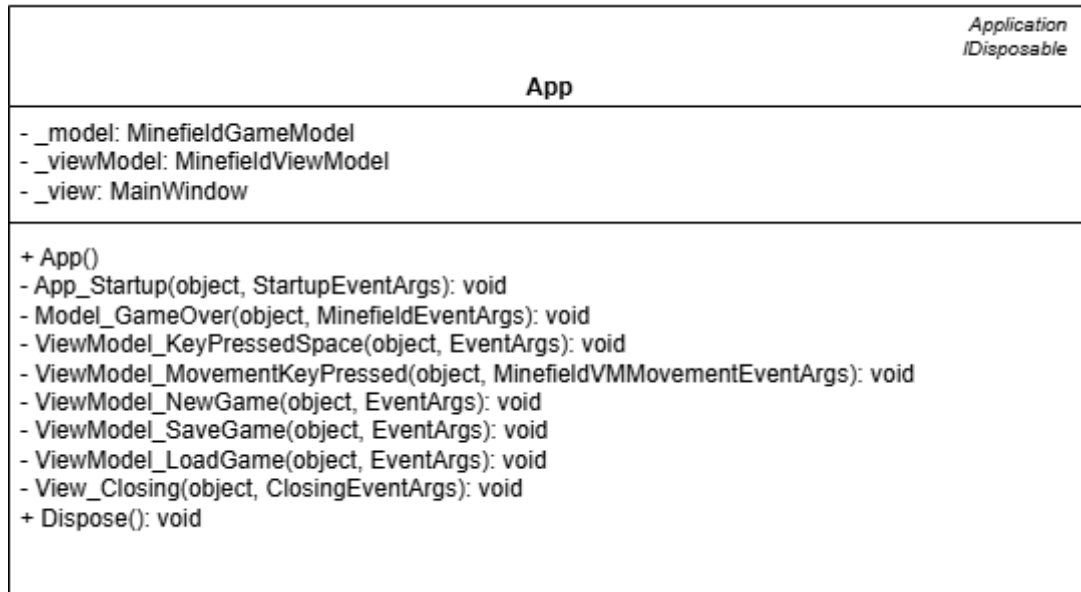
- A nézet egy ablakot tartalmaz, amit a **MainWindow** osztály definiál.
- Az ablak egy rácsban (Grid) tárolja a következőket:
 - Menüsor
 - Játéklemező: Egy ItemsControl vezérlő, ami egy rács szerkezetben (UniformGrid) jeleníti meg a játékelemeket képekkel (Image).
 - Státuszszor: Az aktuális játékidőt jeleníti meg.
- A játék állapotát és a játékelemeket leíró adatokat adatkötéssel (Binding) kapcsoljuk a nézethez.
- A használt képeket Bitmap erőforrásokként vesszük fel és Style Trigger használatával szabályozzuk a megjelenítésüket.
- A mentéshez és a betöltéshez fájlkiválasztó dialógusablakot használunk.



5. ábra: Nézetmodell osztálydiagramja

Környezet

- Az MVVM rétegek példányosítását, összekötését és szabályozását az App osztály végzi.
 - Az indításkor szükséges műveletek az App_Startup eseményvezérlő metódusban hajtódnak végre.
 - Mivel a modell disposable, ezért az App megvalósítja az IDisposable interfészt.



6. ábra: App osztálydiagramja

Tesztelés

A modell ellenőrzése egységtesztekkel valósult meg.

MinefieldUnitTests tesztosztály tesztmetódusai:

- *MinefieldModelStartNewGameTest*: Új játék beállításai.
- *MinefieldTestPausing*: Szüneteltetés és szüneteltetés feloldása.
- *MinefieldModelMovementTest*: Játékos karakter mozgatása.
- *MinefieldPausedMovement*: A mozgás fel van függesztve amíg a játék szünetel.
- *MinefieldMovementBoundsTest*: A mozgás a játéktérre van korlátozva.
- *MinefieldTimerTickTimeAdvanceTest*: Az időzítő tick eseményét kezeli a modell, telik a játékidő.
- *MinefieldMinesDescendingTest*: Az idő telésére helyesen esnek az aknák.
- *MinefieldGameOverMovementTest*: Játék vége bekövetkezik és a megfelelő esemény kiváltódik, ha a játékos „nekimegy” egy aknának.
- *MinefieldGameOverMineDescendingOnPlayerTest*: Játék vége bekövetkezik és a megfelelő esemény kiváltódik, ha a játékos karakterre „ráesik” egy akna.
- *MinefieldMineSpawningIncreasingTest*: Új akna megjelenésének az esélye nő ahogy telik a játékidő. A teszthez mock használatával szimuláljuk a véletlenszám-generátort, hogy mindig a felső és alsó korlát számtani közepét adja vissza.
- *MinefieldSaveGameCalledTest*: A perzisztenciában definiált mentés végrehajtódik.
- *MinefieldLoadGameCalledTest*: A perzisztenciában definiált betöltés végrehajtódik.
- *MinefieldSaveGameFailedTest*: A mentés közbeni hiba kiváltóik.
- *MinefieldLoadGameFailedTest*: A betöltés közbeni hiba kiváltóik.
- *MinefieldLoadGameTest*: Adott adatok helyesen betöltődnek a modellbe.

A mentést és betöltést mock használatával szimuláljuk.

A MinefieldUnitTests osztály implementálja az IDisposable interfészt, mert a MinefieldGameModel típusú _testModel field-je Disposable.