# Harmadik 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).

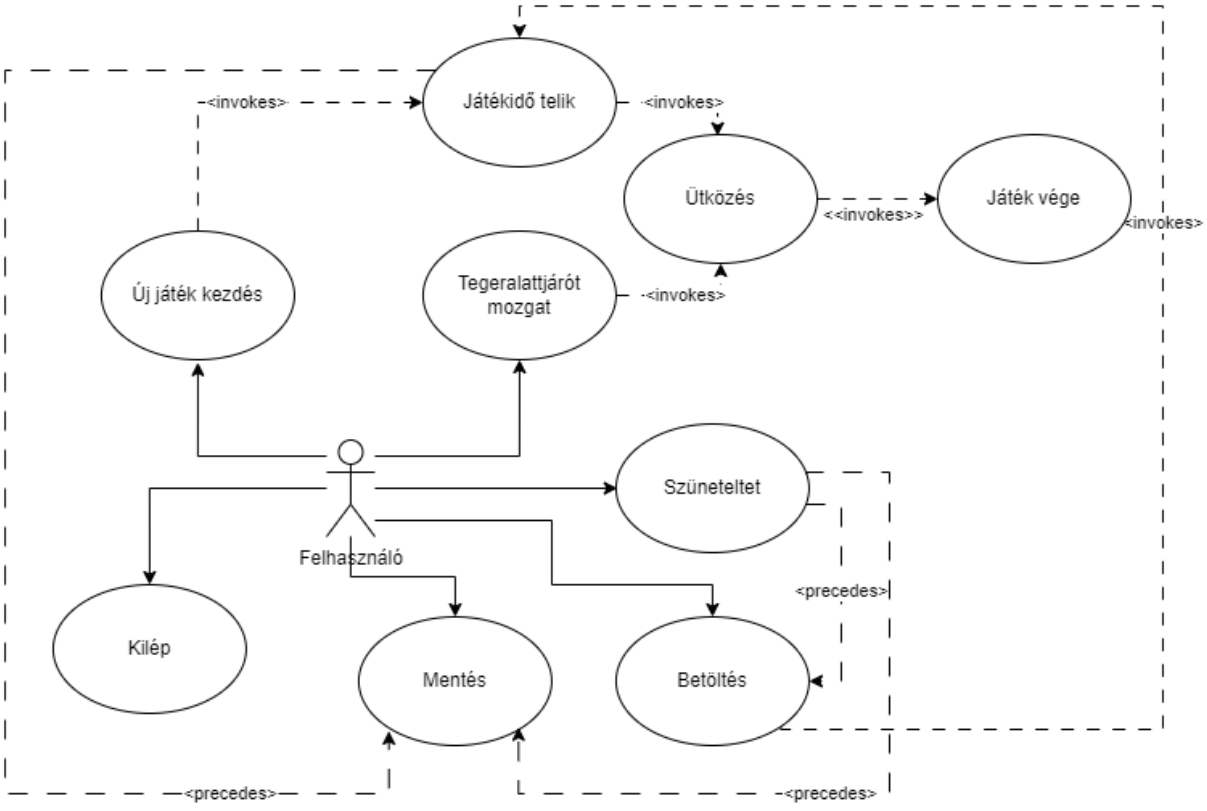
Az aknákat véletlenszerűen dobják a tengerbe, ám mivel a hajóskapitányok egyre türelmetlenebbek, egyre gyorsabban kerül egyre több akna a vízbe. A játékos célja az, hogy minél tovább elkerülje az aknákat. A játék addig tart, ameddig a tengeralattjárót el nem találta egy akna.

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 Avalonia applikációként, grafikus felülettel, kettő platformra: asztalra és Androidra. Utóbbi esetén korlátozzuk vízszintes tájolásra.
- A játék indításakor a játékos új játékot indíthat, de ha létezik, akkor folytathatja a legutóbbi bezáráskor automatikusan elmentett játékot.
- 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.

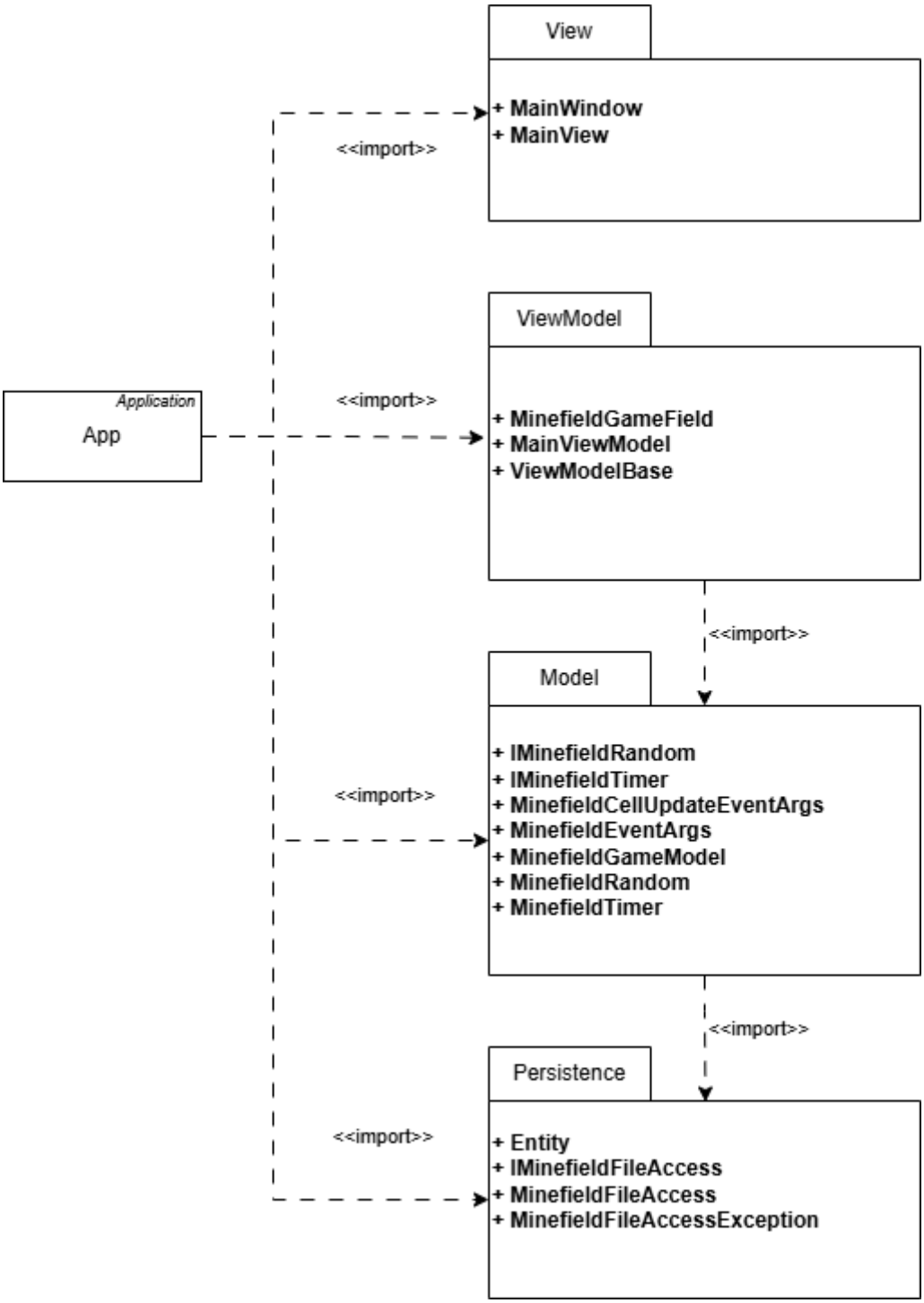
- A játék véget érhet a következő esetekben:
  - A tengeralattjáró nekimegy egy aknának.
  - Az idő teltével egy akna a tengeralattjáróra esik.
- A használati esetekért lásd az 1. ábrát.




*1. ábra: Használati eset diagram*

# Tervezés:

- Programszerkezet: MVVM architektúra:
  - Felületfüggetlen:
    - **modell**: Minefield.Model
    - **perzisztencia**: Minefield.Persistence
  - Avalonia:
    - **nézetmodell**: Minefield.Avalonia.ViewModels
    - **nézet:** A felületet megvalósító .axaml állományok mind a kettő platformra.
  - Rétegeket példányosító és összekötő környezet: **App** osztály
- A csomagdiarammért lásd a 2. ábrát.



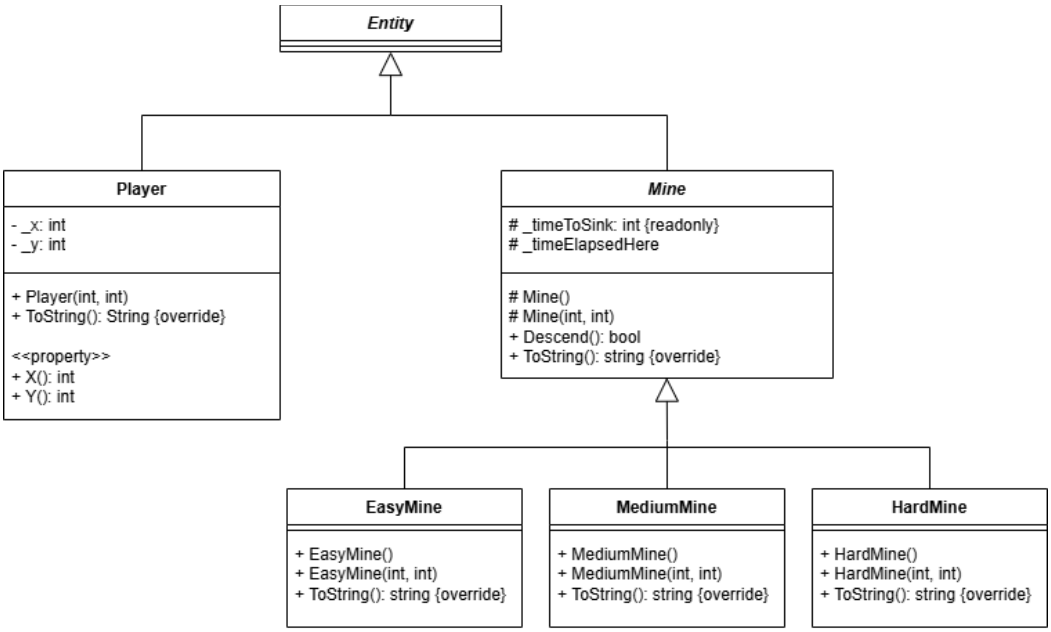

*2. ábra: Csomagdiagram*

## Modell

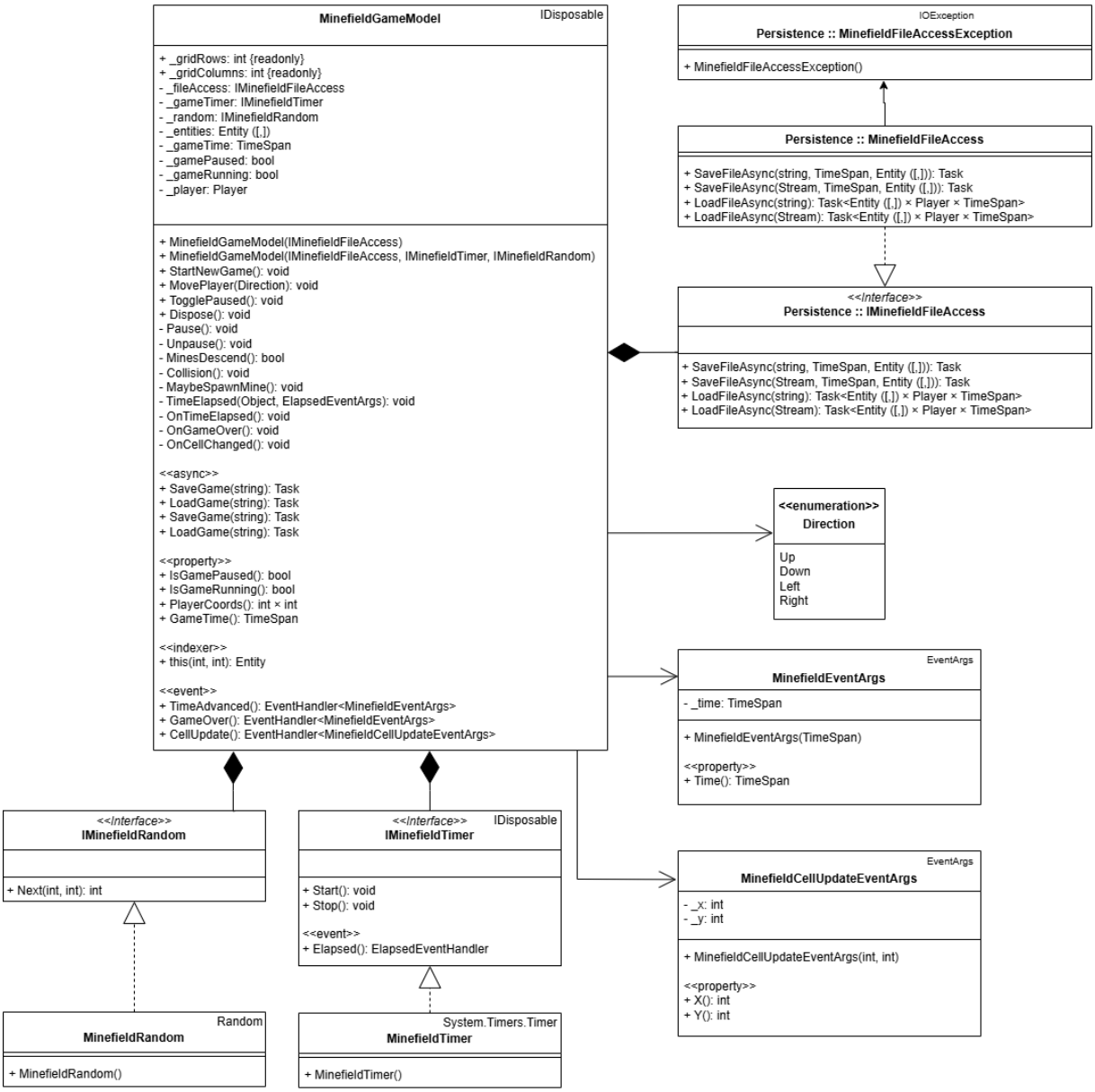
- Az üzleti logikát és a játéktér reprezentációját a MinefieldGameModel osztály tartalmazza.
- Az üzleti logika legfontosabb elemei:
  - Játék indítása, szüneteltetése, leállítása, mentés/betöltés kezelése
  - A játéktér változásainak kezelése, mint a játékos karakter mozgatása, aknák létrehozása és mozgatása, ütközés ellenőrzése
  - Időzítő eseményének kezelése
- A játékteret egy Entitásmátrix (_entities) és a játékidő (_gameTime) reprezentálja. Ezen felül tárolunk a játékos entitásra egy referenciát (_player) annak gyors eléréshez a mozgatás esetén.
- 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 iterfészt, 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.

## Perzisztencia

- Feladata a játékállás mentésének betöltése, valamint egy létező mentés-fájlból a játékállás beolvasása. Ez mind aszinkron módon történik.
- A mentés és betöltés metódusait az IMinefieldFileAccess interfész adja meg.
- 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
- Az entitás típus és leszármazottai:
  - A játéktér résztvevőinek absztrakt ősosztálya 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.




*3. ábra: Entity osztály és leszármazottainak osztálydiagramja*



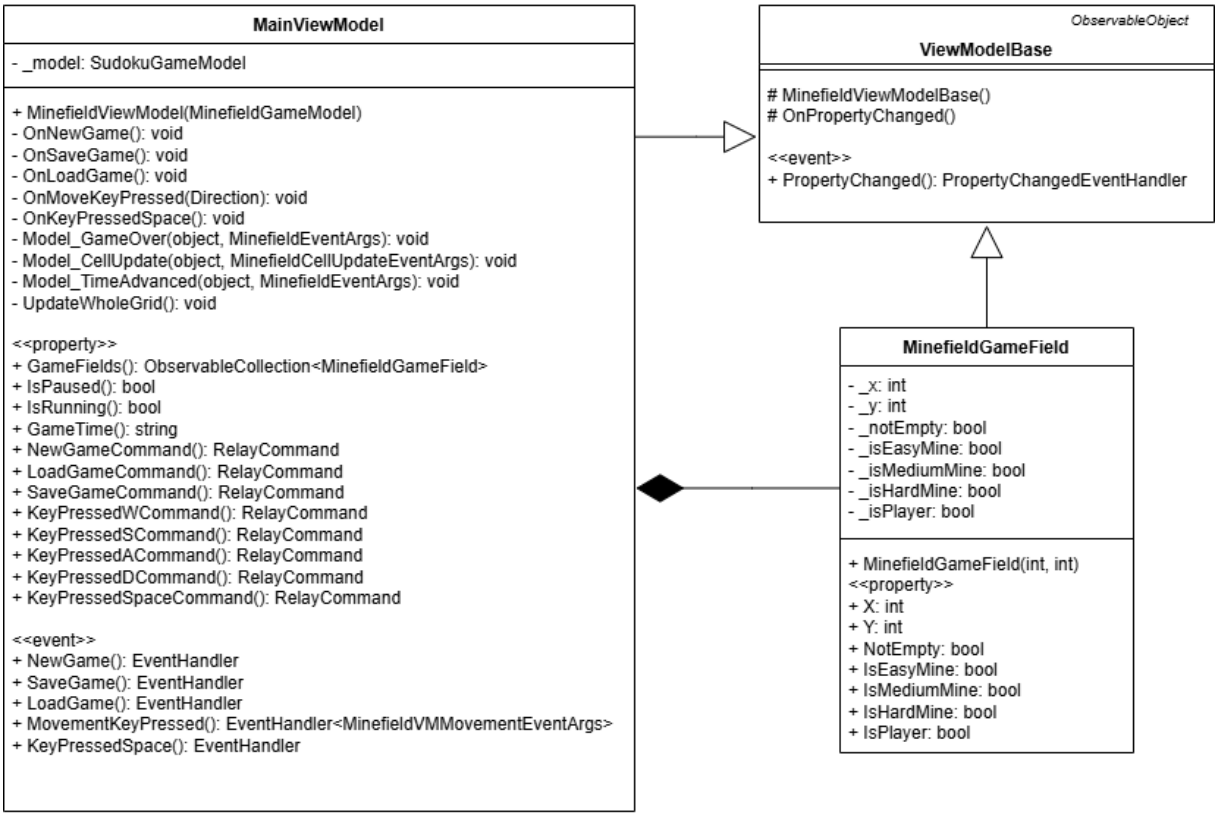

*4. ábra: Modell és perzisztencia osztálydiagramja*

## Nézetmodell

- A nézetmodell és a játékmező osztályát a **ViewModelBase** ősosztályból származtatjuk, ami implementálja az **ObservableObject**interfészt.
- Parancsok definiálásához a **RelayCommand** típust használjuk.
- A nézetmodell elemeit a **MainViewModel** 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 játéktér megjelenítéséhez szükséges adatokat egy ObservableCollection tartalmazza. A gyűjtemény elemeinek típusa **MinefieldGameField**, ami tartalmazza egy mező koordinátáit (X, Y) és tartalmát.
- Az osztálydiagrammért lásd az 5. ábrát.

## Nézet

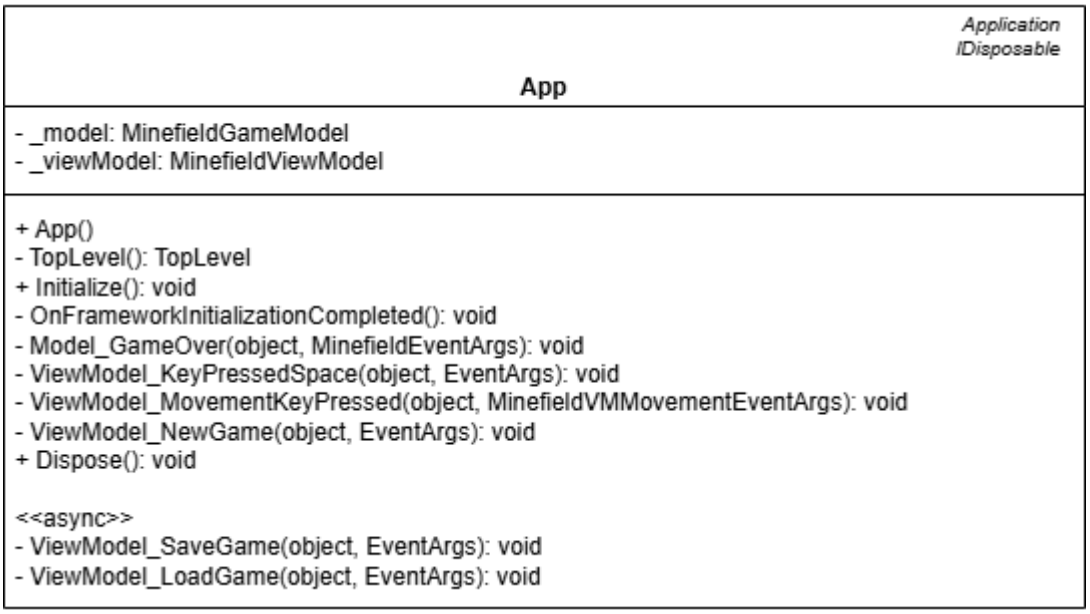
- A nézetet a **MainView** és az azt magába foglaló **MainWindow** definiálja
- Az felületelemek elhelyezését Grid és Viewbox elemekkel valósítjuk meg.
- Főbb elemek:
  - Menüsor
  - Játékmező: Egy ItemsControl vezérlő, ami egy rács szerkezetben (UniformGrid) jeleníti meg a játékelemeket képekkel (Image).
  - Sztalon státuszsornak megfelelő szöveges blokk: Az aktuális játékidőt jeleníti meg.
- A játék állapotát és a játékelemeket leíró adatokat adatkötéssel (Binding) kapcsoljuk a nézethez.
- A játékelemek megjelenéséhez kép erőforrásokat importálunk és style szelektorokkal szabályozzuk.




*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.
  - Itt definiáljuk a program felfüggesztésekor és betöltésekor történő automatikus mentéskészítést, és -betöltést.
  - 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 iterfészt, mert a MinefieldGameModel típusú _testModel field-je Disposable.