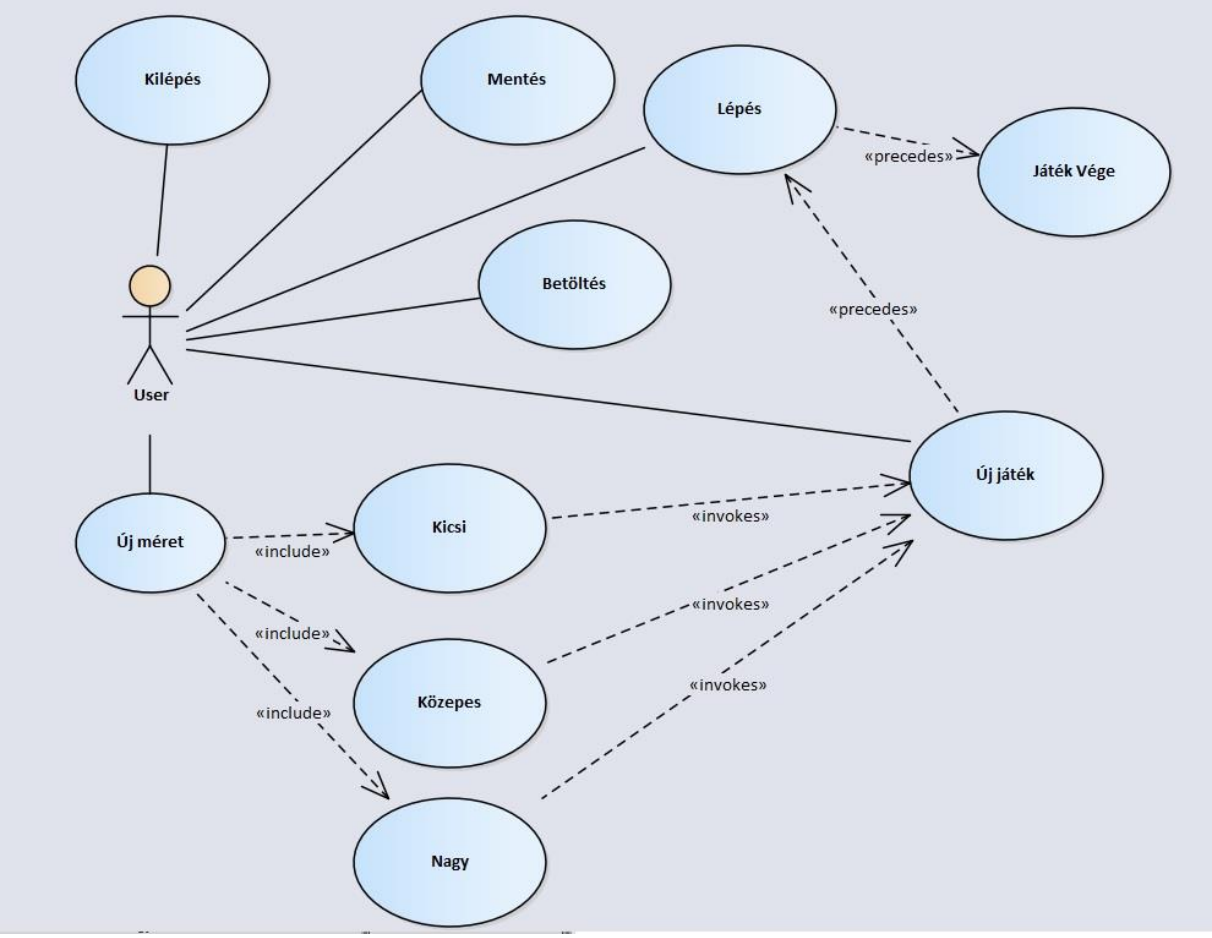
2. Beadandó feladat dokumentáció

Készítette:
Kovács Levente

Feladat:
Készítsünk programot, amellyel az aknakereső játék két személyes változatát játszhatjuk. Adott egy $n \times n$ mezőből álló tábla, amelyen rejtett aknákat helyezünk el. A többi mező szintén elrejtve tárolják, hogy a velük szomszédos 8 mezőn hány akna helyezkedik el. A játékosok felváltva léphetnek. Egy mező felfedjük annak tartalmát. Ha az akna, a játékos veszített. Amennyiben a mező nullát rejt, akkor a vele szomszédos mezők is automatikusan felfedésre kerülnek (és ha a szomszédos is nulla, akkor annak a szomszédai is, és így tovább). A játék addig tart, amíg valamelyik játékos aknára nem lép, vagy fel nem fedték az összes nem akna mezőt (ekkor döntetlen lesz a játék). A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (6 × 6, 10 × 10, 16 × 16), valamint játék mentésére és betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött (ha nem döntetlen).

Elemzés:

- A játékot három méretben játszhatjuk: kicsi (6x6 mező, 5 akna), közepes (8x8 játékmező, 8 akna), nagy (16x16 játékmező, 12 akna) . A program indításkor közepes nehézséget állít be, és automatikusan új játékot indít. Az aknák az adott számban véletlenszerűen lesznek elhelyezve a játékmezőn.
- A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (Új játék, Játék betöltése, Játék mentése), Új méret (Kicsi,közepes,nagy).
- A játéktáblát egy n×n nyomógombokból álló rács reprezentálja. A nyomógomb egérkattintás hatására felfedi az adott mező tartalmát. Ha 0 a mező értéke (azt üres mezővel fogja jelölni a játék) , akkor a körülötte lévő mezőket is feloldja (ha azok között is van nulla, akkor a 0-s mező körüli mezők is feloldódnak és így tovább). A nem feloldott mezőket fekete színnel jelöljük. A 0-s mezőket kék mezőként jelenítjük meg.
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (minden mezőt felfedtünk ami nem akna, vagy az egyik játékos aknára lépett). Szintén dialógusablakokkal végezzük el a mentést, illetve betöltést, a fájlneveket a felhasználó adja meg.
- A felhasználói esetek az 1. ábrán láthatóak.



1. ábra: Felhasználói esetek diagramja

Tervezés:

Programszerkezet:

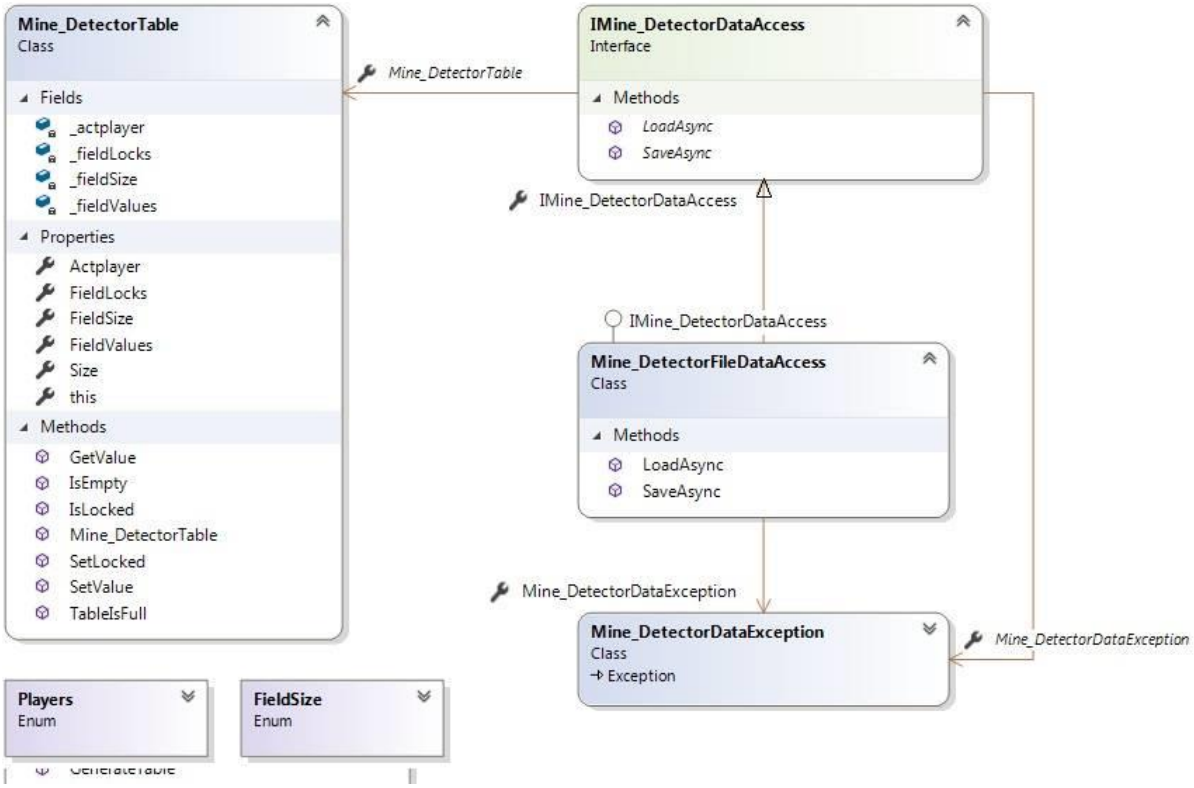
- A programot MVVM architektúrában valósítjuk meg, ennek megfelelően **`View`**, **`Model`**, **`ViewModel`** és **`Persistence`** névtereket valósítunk meg az alkalmazáson belül. A program környezetét az alkalmazás osztály (**`App`**) végzi, amely példányosítja a modellt, a nézetmodell és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést.

Perzisztencia:

- Az adatkezelés feladata a Mine_Detector táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
- A **`Mine_DetectorTable`** osztály egy érvényes Mine_Detector (aknakereső) táblát biztosít (azaz mindig ellenőrzi a beállított értékeket), ahol minden mezőre ismert az értéke (**`_fieldValues`**), illetve a zároltsága (**`_fieldLocks`**). A _fieldLocks Booleanokat tartalmazó mátrixban tároljuk azoknak a mezőknek a pozicióit, amiknek értéke még felfedetlen. A tábla

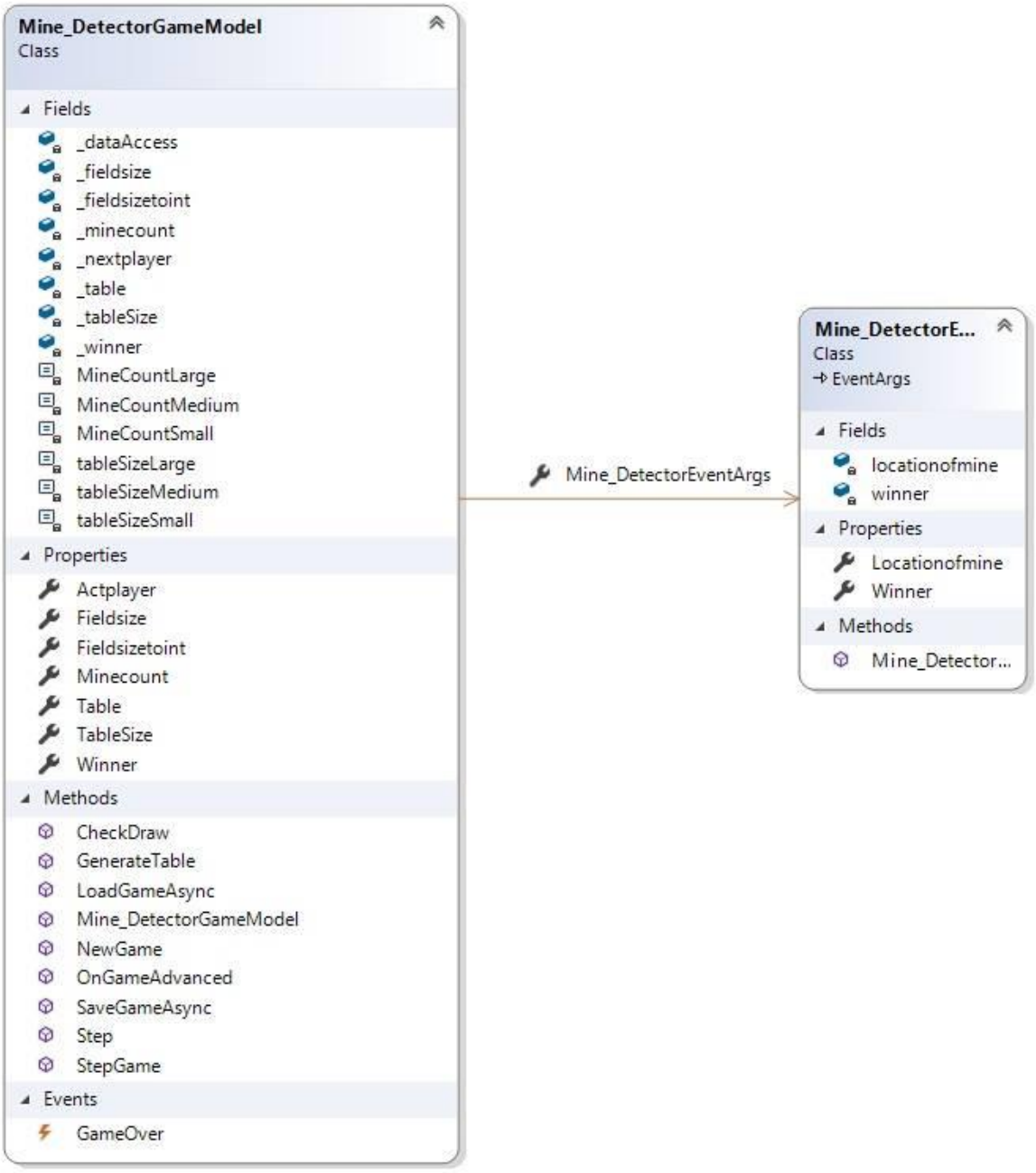
alapértelmezés szerint *8x8-as 7 aknával*, de ez a konstruktorban paraméterezhető. A tábla lehetőséget ad az állapotok lekérdezésére (**`TableIsFull`**, **`IsLocked`**, **`IsEmpty`**, **`GetValue`**,**`Size`**), illetve direkt beállítás (**`SetValue`**, **`SetLock`**) elvégzésére.

- A hosszú távú adattárolás lehetőségeit az **`IMine_DetectorDataAccess`** interfész adja meg, amely lehetőséget ad a tábla betöltésére (LoadAsync), valamint mentésére (SaveAsync). A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg.
- Az interfészt szöveges fájl alapú adatkezelésre a **`Mine_DetectorFileDataAccess`** osztály valósítja meg. A fájlkezelés során fellépő hibákat a **`Mine_DetectorDataException`** kivétel jelzi.
- A program az adatokat szöveges fájlként tudja eltárolni, melyek az **`stl`** kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
- A fájl sorai tartalmazzák a táblaméretet, az aktuális játékost, a zárolt mezőket, illetve a tábla mezőinek értékeit.



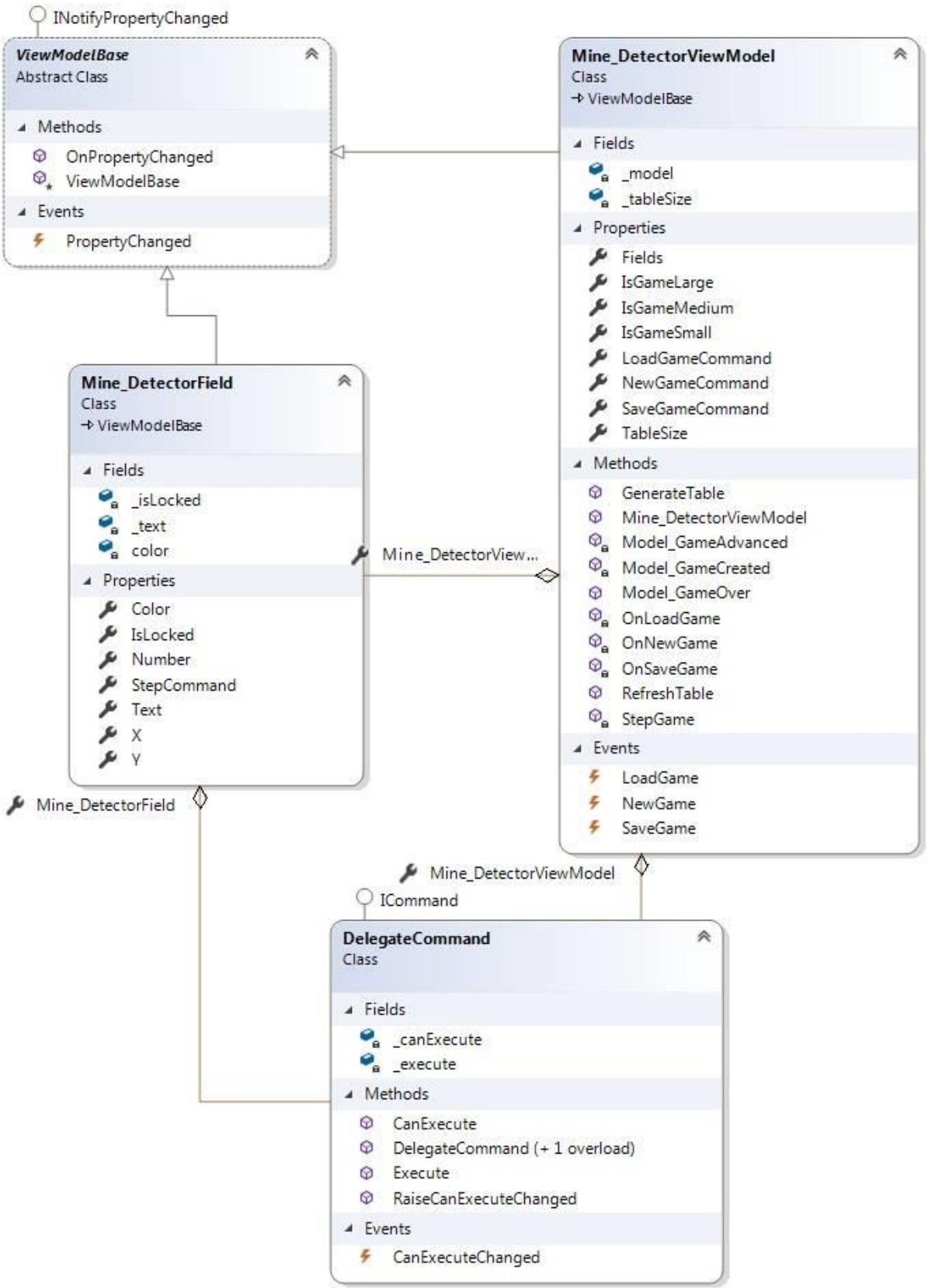
Modell:

- A modell lényegi részét a **`Mine_DetectorGameModel`** osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, úgymint az aktuális játékost (_actplayer) és az aktuális győztest (_winner). A típus lehetőséget ad új játék kezdésére (NewGame), valamint lépésre (**`StepGame`**).
- A játékállapot változásáról a **`GameOver`** esemény tájékoztat. Az események argumentuma (**`Mine_DetectorEventArgs`**) tárolja a győztest, valamint az utoljára kattintott mező paramétereit.
- A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (**`LoadGameAsync`**) és mentésre (**`SaveGameAsync`**)
- A játék méretét a FieldSize felsorolási típuson át kezeljük, és konstansok segítségével tároljuk az egyes méretek paramétereit.


Mine_DetectorGameModel
Class
Fields
_dataAccess
_fieldsize
_fieldsizetoint
_minecount
_nextplayer
_table
_tableSize
_winner
MineCountLarge
MineCountMedium
MineCountSmall
tableSizeLarge
tableSizeMedium
tableSizeSmall
Properties
Actplayer
Fieldsize
Fieldsizetoint
Minecount
Table
TableSize
Winner
Methods
CheckDraw
GenerateTable
LoadGameAsync
Mine_DetectorGameModel
NewGame
OnGameAdvanced
SaveGameAsync
Step
StepGame
Events
GameOver
Mine_DetectorEventArgs
Mine_DetectorE...
Class
EventArgs
Fields
locationofmine
winner
Properties
Locationofmine
Winner
Methods
Mine_Detector...

Nézetmodell:

- A nézetmodell megvalósításához felhasználunk egy általános utasítás (**`DelegateCommand`**), valamint egy ős változásjelző (**`ViewModelBase`**) osztályt.
- A nézetmodell feladatait a **`Mine_DetectorViewModel`** osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, mentéséhe. 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 (**`Mine_DetectorField`**), amely eltárolja a pozíciót, szöveget, számelrejtést, valamint a lépés parancsát (**`StepCommand`**). A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (**`Fields`**).

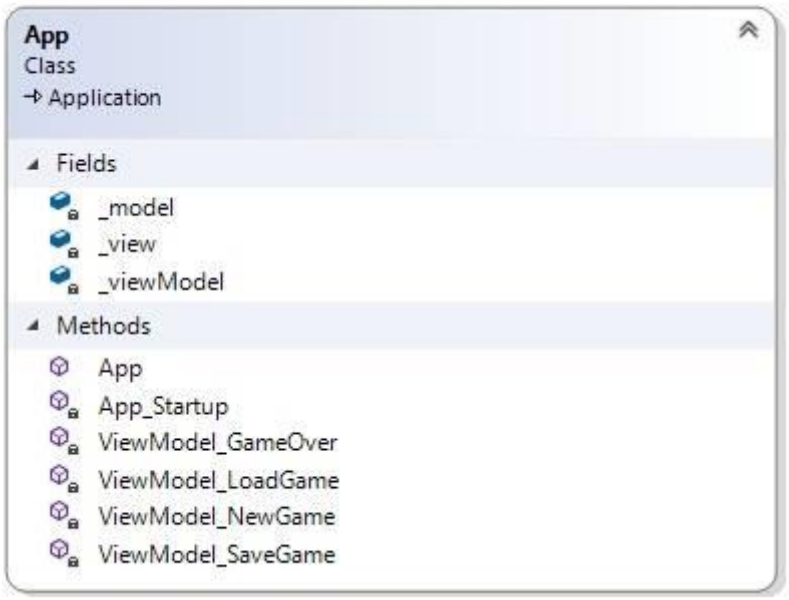

INotifyPropertyChanged
ViewModelBase
Abstract Class
Methods
OnPropertyChanged
ViewModelBase
Events
PropertyChanged
Mine_DetectorViewModel
Class
ViewModelBase
Fields
_model
_tableSize
Properties
Fields
IsGameLarge
IsGameMedium
IsGameSmall
LoadGameCommand
NewGameCommand
SaveGameCommand
TableSize
Methods
GenerateTable
Mine_DetectorViewModel
Model_GameAdvanced
Model_GameCreated
Model_GameOver
OnLoadGame
OnNewGame
OnSaveGame
RefreshTable
StepGame
Events
LoadGame
NewGame
SaveGame
Mine_DetectorField
Class
ViewModelBase
Fields
_isLocked
_text
color
Properties
Color
IsLocked
Number
StepCommand
Text
X
Y
Mine_DetectorView...
Mine_DetectorField
Mine_DetectorViewModel
ICommand
DelegateCommand
Class
Fields
_canExecute
_execute
Methods
CanExecute
DelegateCommand (+ 1 overload)
Execute
RaiseCanExecuteChanged
Events
CanExecuteChanged

Nézet:

- Nézet:
- A nézet csak 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 gombokból áll. Minden adatot adatkötéssel kapcsolunk a felülethez, továbbá azon keresztül szabályozzuk a gombok színét is.
- A fájlnév bekérését betöltéskor és mentéskor, valamint a figyelmeztető üzenetek megjelenését beépített dialógusablakok segítségével végezzük.

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.



Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **`Mine_DetectrGameModelTest`** osztályban.
- Az alábbi tesztesetek kerültek megvalósításra:
- **`Mine_DetectorSmallNewGameTest`**, **`Mine_DetectorMediumNewGameTest`**, **`Mine_DetectorLargeNewGameTest`**: Új játék indítása, a mezők kitöltése, valamint az aknák számának, következő játékos helyességének ellenőrzése a pálya méretének függvényében.
- **`Mine_DetectorStepGameTestPlayer2Win`**: A győzelem vizsgálata, az eseménykezelő értékeivel, amikor a második játékos győz.
- **`Mine_DetectorStepGameTestPlayer1Win`**: A győzelem vizsgálata, abban az esetben, mikor az első játékos lép bombára hamarabb.
- **`Mine_DetectorStepGameTestDraw:`** A játékot illetve az eseménykezelők értékeit ellenőrízzü akkor, ha a játék kimenetele döntetlenre végződik.
- **A mentést illetve a betöltést Mockolással végeztük el.**