**2. Beadandó feladat dokumentáció**

**Készítette:** Csillag András Benedek (KGEB7Y)

E-mail: kgeb7y@inf.elte.hu (vagy cs.andras0323@gmail.com)

**Feladat:**

**20. Négyzetek**

Készítsünk programot, amellyel az alábbi két személyes játékot játszhatjuk.

Adott egy 𝑛×𝑛 pontból álló játéktábla, amelyen a játékosok két szomszédos pont között vonalakat húzhatnak (vízszintesen, vagy függőlegesen). A játék célja, hogy a játékosok a húzogatással négyzetet tudjanak rajzolni (azaz ők húzzák be a negyedik vonalat, független attól, hogy az eddigieket melyikük húzta). Ilyen módon egyszerre akár két négyzet is elkészülhet. A játék addig tart, amíg lehet húzni vonalat a táblán.

A játékosok felváltva húzhatnak egy-egy vonalat, de ha egy játékos berajzolt egy négyzetet, akkor ismét ő következik.

A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (3×3, 5×5, 9×9), játék mentésére és betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött (ha nem döntetlen). Játék közben a vonalakat, illetve a négyzeteket színezze a játékos színére.

**Elemzés:**

A játékot három különböző pályafelosztással játszhatjuk:

* 3x3 pontos (5x5 mezős tábla)
* 5x5 pontos (9x9 mezős tábla)
* 9x9 pontos (17x17 mezős tábla).

A program indításkor 3x3 pontos (5 x 5 mezős) táblát állít be, és automatikusan új játékot indít.

A feladatot egyablakos asztali alkalmazásként Windows Presentation Foundation grafikus felülettel valósítjuk meg.

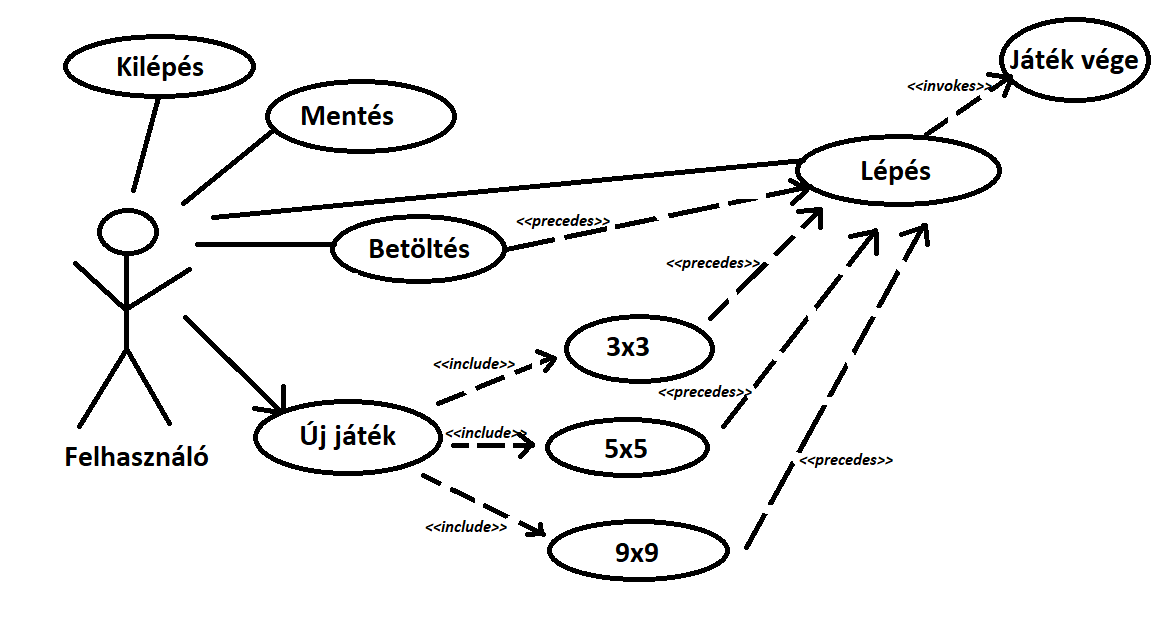
Az ablakban elhelyezünk egy menüt a következő menüpontokkal:

* New Game (3x3, 5x5, 9x9)
* File (Save, Load).

A játéktáblát egy 5x5 /9x9/17x17 nyomógombokból álló rács reprezentálja megfelelő helyen nyomógombokkal ellátva (ahova lehet vonalat húzni). A nyomógomb egérkattintás hatására megváltoztatja a megjelenített mezőt, figyelembe véve, hogy melyik játékos lépett, valamint ha ezzel teljessé tett egy (vagy két) négyzetet, akkor a 4 csúcs közötti területet ábrázoló mező a játékos színére vált. A már kiválasztott mezők elvesztik nyomógomb funkcionalitásukat. A táblán a pontokat ábrázoló mezőket nem engedjük megváltoztatni, ezeket változatlanok a játék során. Az ún. belső mezőket sem lehet direkt mód változtatni.

A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (minden négyzet teljes lett). Szintén dialógusablakokkal végezzük el a mentést, illetve betöltést, a fájlneveket a felhasználó adja meg.

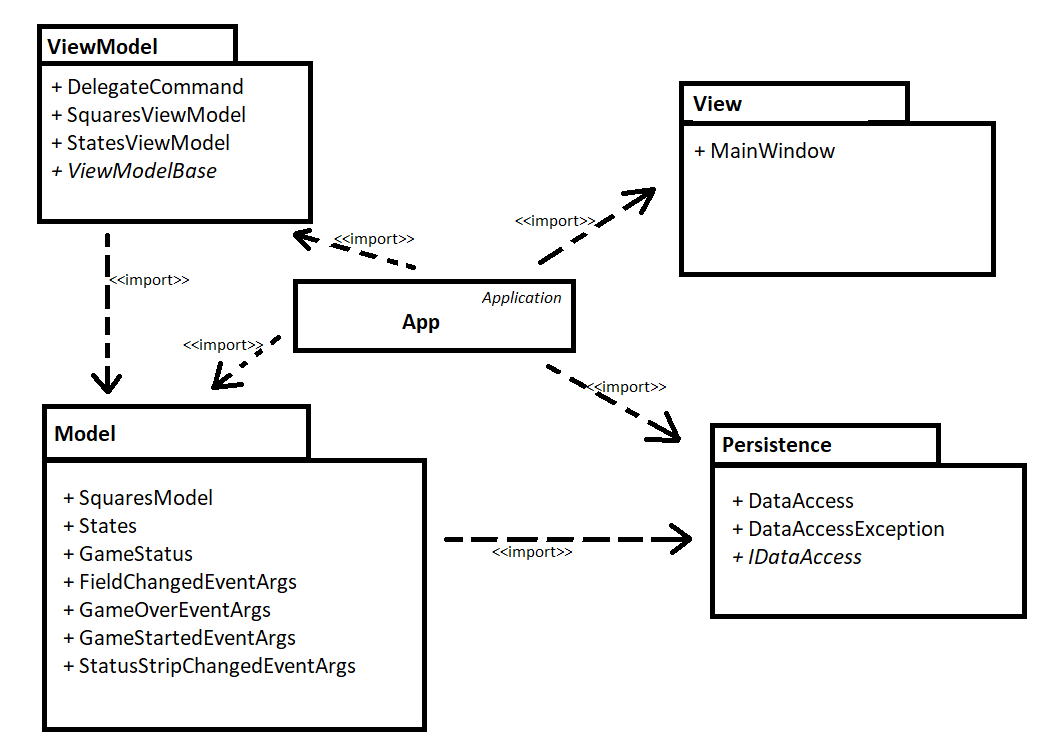
A felhasználói esetek az 1. ábrán láthatóak.



**Tervezés:**

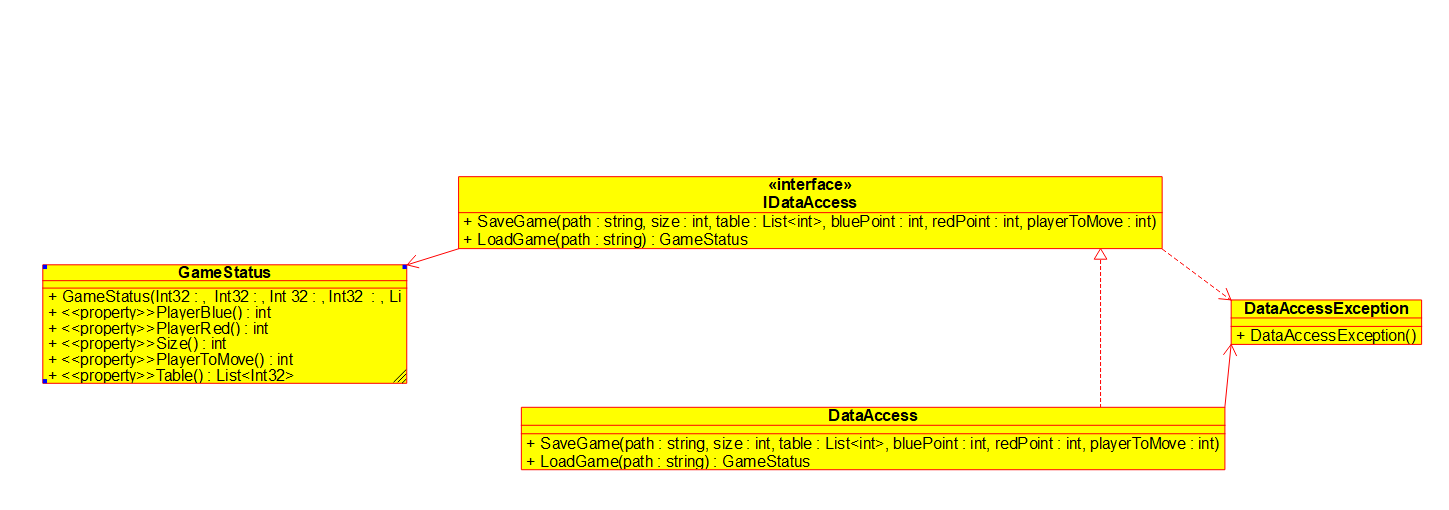
Programszerkezet:

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. A program csomagszerkezete a 2. ábrán látható.



Perzisztencia: (3. ábra):

* Az adatkezelés feladata a betöltés/mentés biztosítása.
* Az adattárolás lehetőségeit az **IDataAccess** interfész adja meg, amely lehetőséget ad a játék állapotának betöltésére (LoadGame), valamint mentésére (SaveGame).
* Az interfészt szöveges fájl alapú adatkezelésre a **DataAccess** osztály valósítja meg. A fájlkezelés során fellépő hibákat a **DataAccessException** kivétel jelzi.
* A program az adatokat szöveges fájlba tudja kiírni, a felhasználó által választott fájlba. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
* A fájl első sora megadja a tábla méretét, valamint a kék játékos pontszámát, piros játékos pontszámát, soron következő játékos számát (kék -> 0/ piros -> 1). A fájl többi része izomorf leképezése a játéktáblának, azaz összesen x sor következik, és minden sor x számot tartalmaz szóközökkel választva. A számok 0-8 közöttiek lehetnek, ezek az értékek a model States nevű felsorolójának tagjainak megfeleltethetők. (x = tábla mérete)

****

Modellábra)

* A modell lényegi részét a **SquaresModel** osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, úgymint a játékosok pontszáma és a következő lépésre jogosult játékos. A típus lehetőséget ad új játék kezdésére (**StartNewGame**), valamint lépésre (**FieldClicked**). Új játéknál automatikusan generálódnak a kezdő mezők.
* A játékállapot változásáról a **FieldChanged, StatusStripChanged** események, míg a játék végéről a **GameOver** esemény tájékoztat.
* Az események argumentumai:

**FieldChangedEventArgs {sor, oszlop, új állapot}**

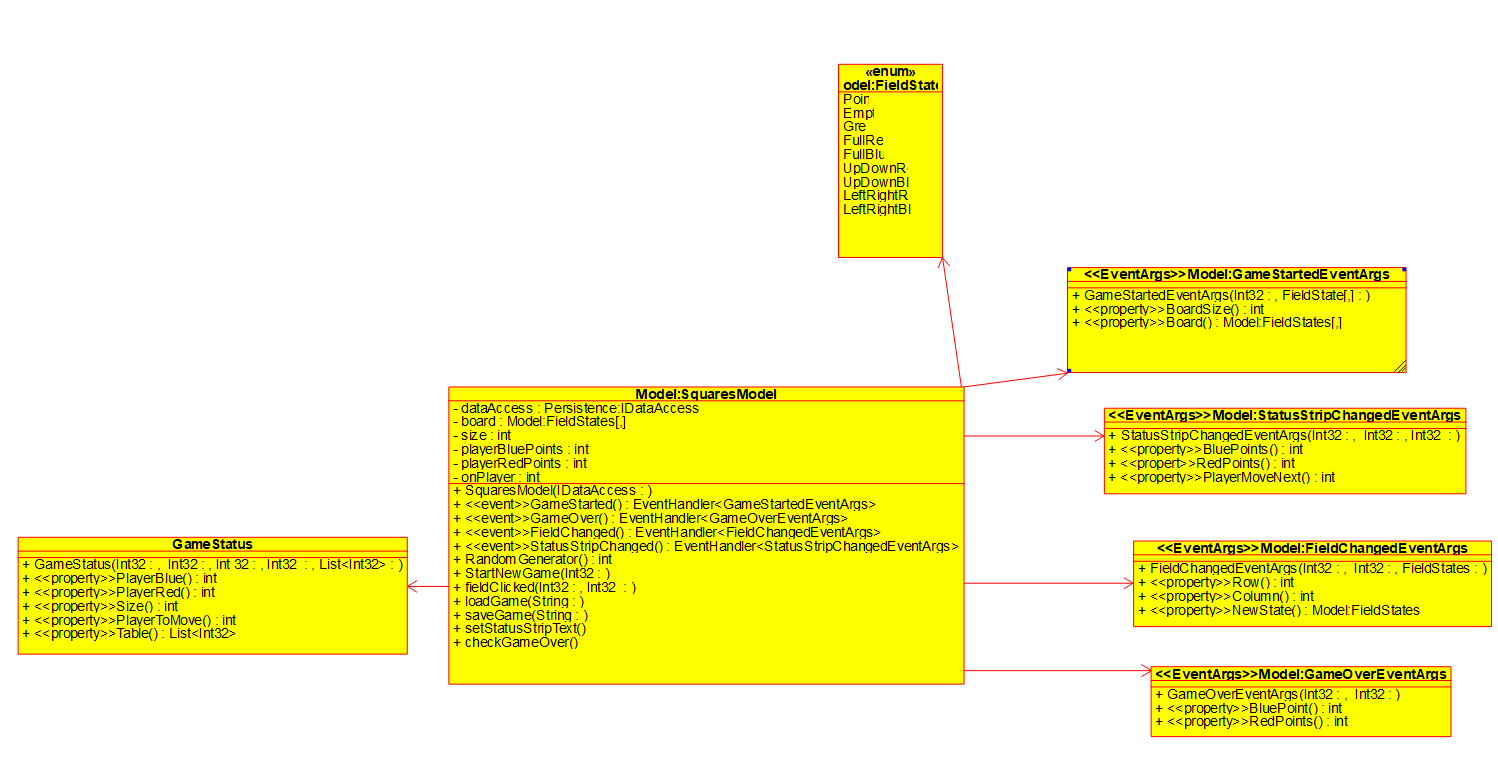
**GameOverEventArgs, {kék pontszám, piros pontszám}**

**GameStartedEventArgs, {táblaméret, tábla}**

**StatusStripChangedEventArgs {kék pontszám, piros pontszám, játékos következik}**

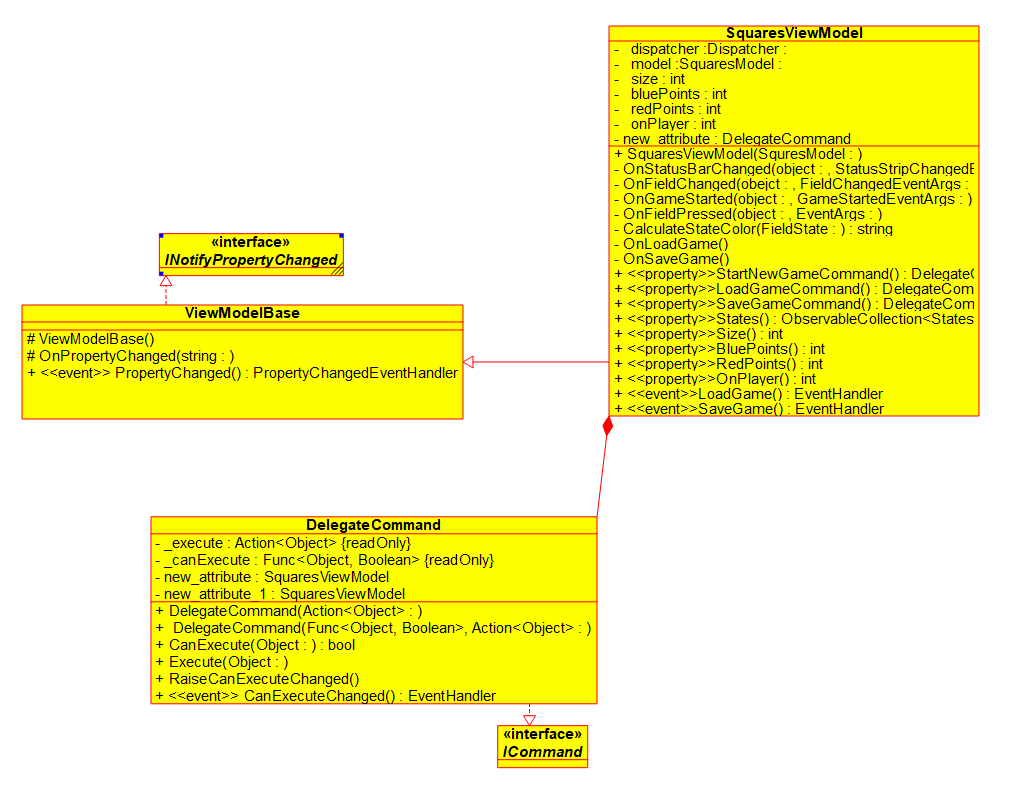
Különböző állapotokat tárolnak: {…}.

* A modell példányosításkor megkapja az adatkezelés felületét (dataAccess), amelynek segítségével lehetőséget ad betöltésre (**LoadGame**) és mentésre (**SaveGame**)

****

Nézetmodell: (5. ábra)

* A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) osztályt.
* A nézetmodell feladatait a SquaresViewModel osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez valamint mentéséhez. 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. Direkt nem avatkozik a játék futtatásába.
* A tábla méretét, a játékosok pontszámát és a soron következő játékos kilétét propertykben eltároljuk, (Size, BluePoints, RedPoints, OnPlayer) valamint az új játék, betöltés és mentés parancsát (StartNewGameCommand, LoadGameCommand, SaveGameCommand).
* A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellben (States).

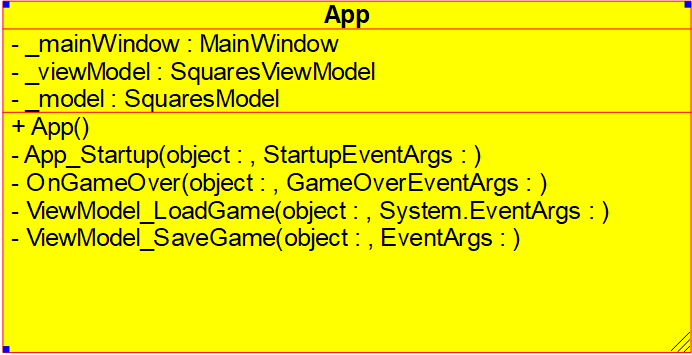
****

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: (6. ábra):

* 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 **SquaresTest** osztályban.

Az alábbi tesztesetek kerültek megvalósításra:

* **StartNewGameThreeByThree**, **StartNewGameFiveByFive**, **StartNewGameNineByNine**: Új játék indítása, a mezők feltöltése, valamint a játékosok pontszámának ellenőrzése a pályafelosztás függvényében.
* **FieldClickedNoSquareMade**, **FieldClickedOneSquareMade, FieldClickedTwoSquaresMade**: Játékbeli lépés hatásainak ellenőrzése, amennyiben azzal a játékos 0/1/2 négyzetet tesz teljessé.
* **LoadGameMockedVersion**: A játék modell betöltésének tesztelése mockolt perzisztencia réteggel.