

# 1. Beadandó feladat dokumentáció

## Készítette:

Restye János Barnabás

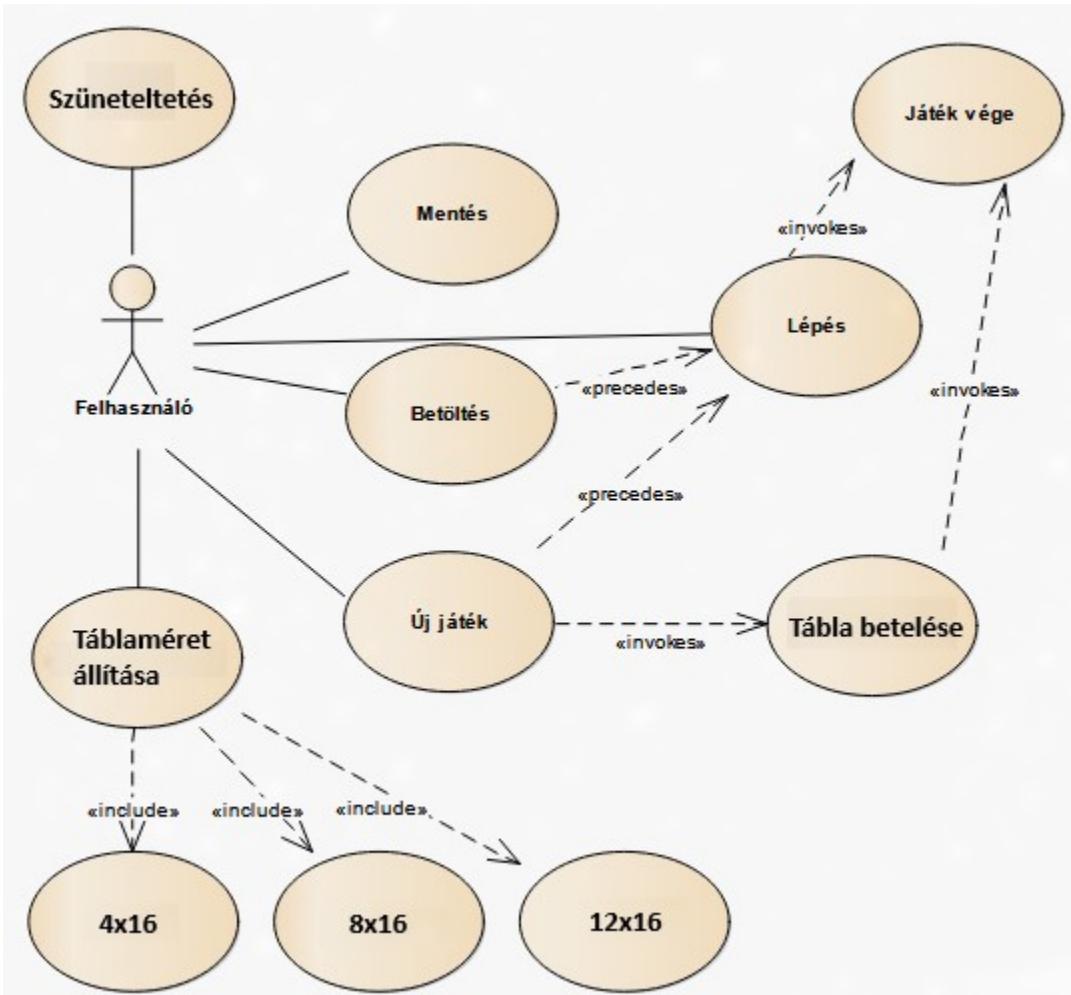
[f8u9i2@inf.elte.hu](mailto:f8u9i2@inf.elte.hu)

## Feladat:

Készítsünk programot a közismert Tetris játékra. Adott egy  $n \times m$  pontból álló tábla, amely kezdetben üres. A tábla tetejéről egymás után új, 4 kockából álló építőelemek hullanak, amelyek különböző formájúak lehetnek (kocka, egyenes, L alak, tető, rombusz). Az elemek rögzített sebességgel esnek lefelé, és az első, nem telített helyen megállnak. Amennyiben egy sor teljesen megtelik, az eltűnik a játékmezőről, és minden felette lévő kocka eggyel lejjebb esik. A játékosnak lehetősége van az alakzatokat balra, jobbra mozgatni, valamint forgatni óramutató járásával megegyező irányba, így befolyásolhatja azok mozgását. A játék addig tart, amíg a kockák nem érik el a tábla tetejét. A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával ( $4 \times 16$ ,  $8 \times 16$ ,  $12 \times 16$ ), valamint játék szüneteltetésére (akkor nem telik az idő, és nem mozognak az elemek). 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:

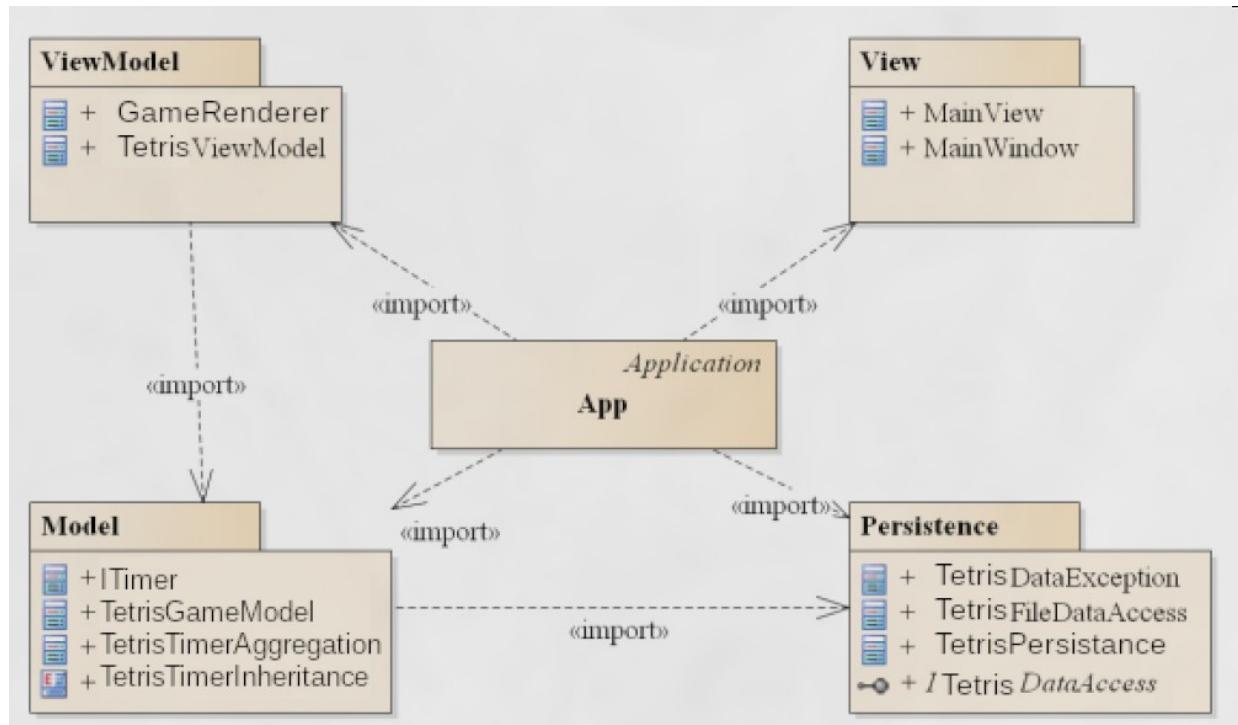
- A feladatot .NET Avalonia alkalmazásként, elsődlegesen asztali alkalmazásként és Android platformon valósítjuk meg, amely egy fő nézetből fog állni. Az alkalmazás portré tájolást támogat.
- A játékot három táblamérettel játszhatjuk:  $4 \times 16$ ,  $8 \times 16$  és  $12 \times 16$ . A program indításakor a játékos játék megkezdése előtt lehetőséget kap ezek közül választani.
- Az ablak bal oldalán lesz a tábla, a jobb oldalán a menü a következő menüpontokkal: Új játék, Szünet, Mentés, Betöltés. A menüpontok alatt lesz a időszámláló.
- A játéktábla a játékos által választott méretnek fog megfelelni. Új játék menüpont választása után a játék elindul egy tetrominó megjelenésével. A játékos a tetrominó mozgását a WASD gombokkal tudja irányítani.
- Ha a játékos veszít, egy dialógusablak értesíti a játék végéről. Dialógusablak értesíti még a játékosat a játék sikeres mentése és betöltése után.
- A felhasználói esetek az 1. ábrán láthatóak.



1. ábra felhasználói esetek diagramja

## Tervezés:

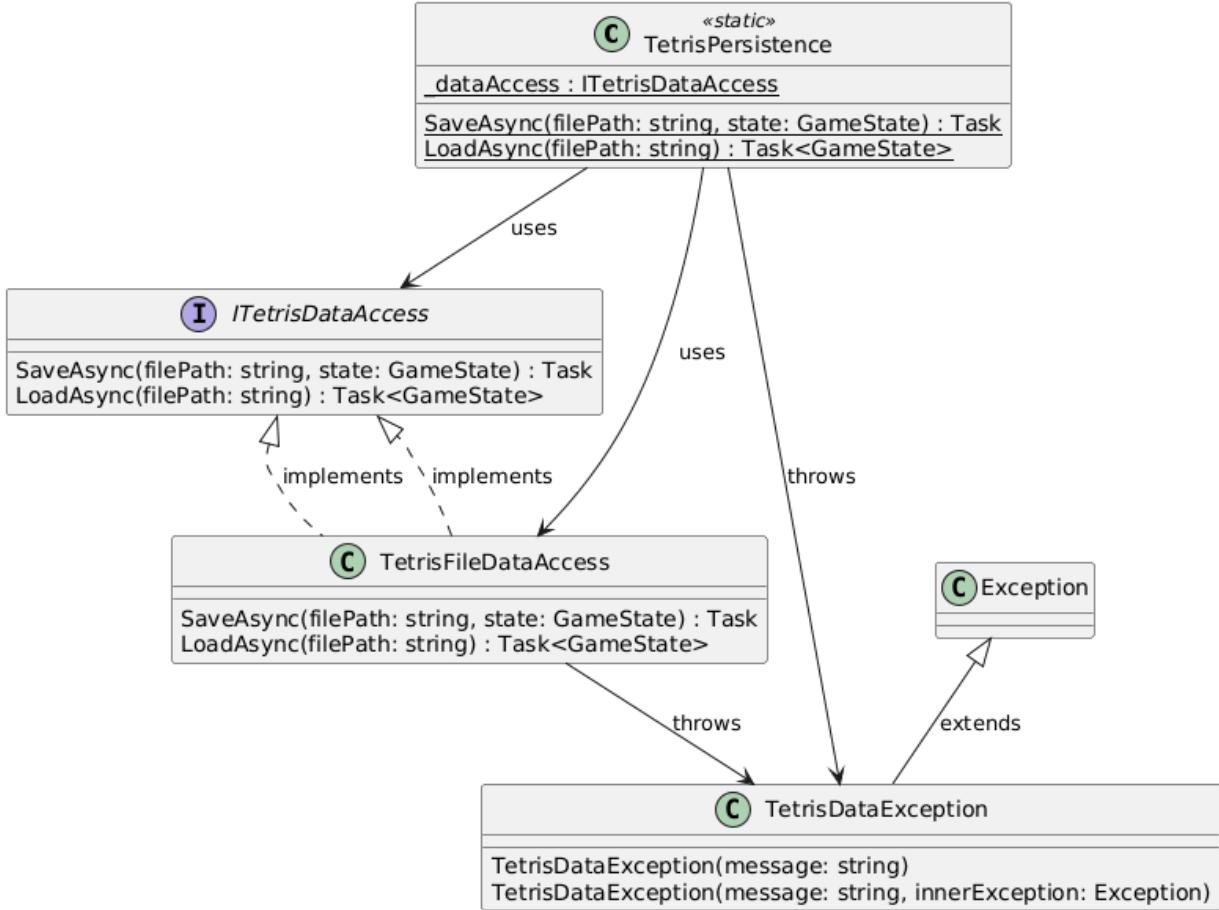
- A programszerkezet:
  - A szoftvert négy projektből építjük fel: a modellt és a perzisztenciát tartalmazó osztálykönyvtárból (.NET Standard Class Library), valamint a .NET Avalonia projektjeiből (platformfüggetlen osztálykönyvtár és platformfüggő végrehajtható projektek), amelyet így Windows és Android operációs rendszerekre is le tudunk fordítani.
  - A programot MVVM architektúrában valósítjuk meg, ennek megfelelően View, Model, ViewModel és Persistence névtereket valósítunk meg az alkalmazáson belül.
  - A program vezérlését az alkalmazás osztály (App) végzi, amely példányosítja a modellt, a nézetmodell és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést. Az alkalmazás környezeti réteg feladata a platformfüggő alkalmazás életciklus helyénvaló kezelése is.
  - A program csomagdiagramja a 2. ábrán látható.



2. ábra: Az alkalmazás csomagdiagramja

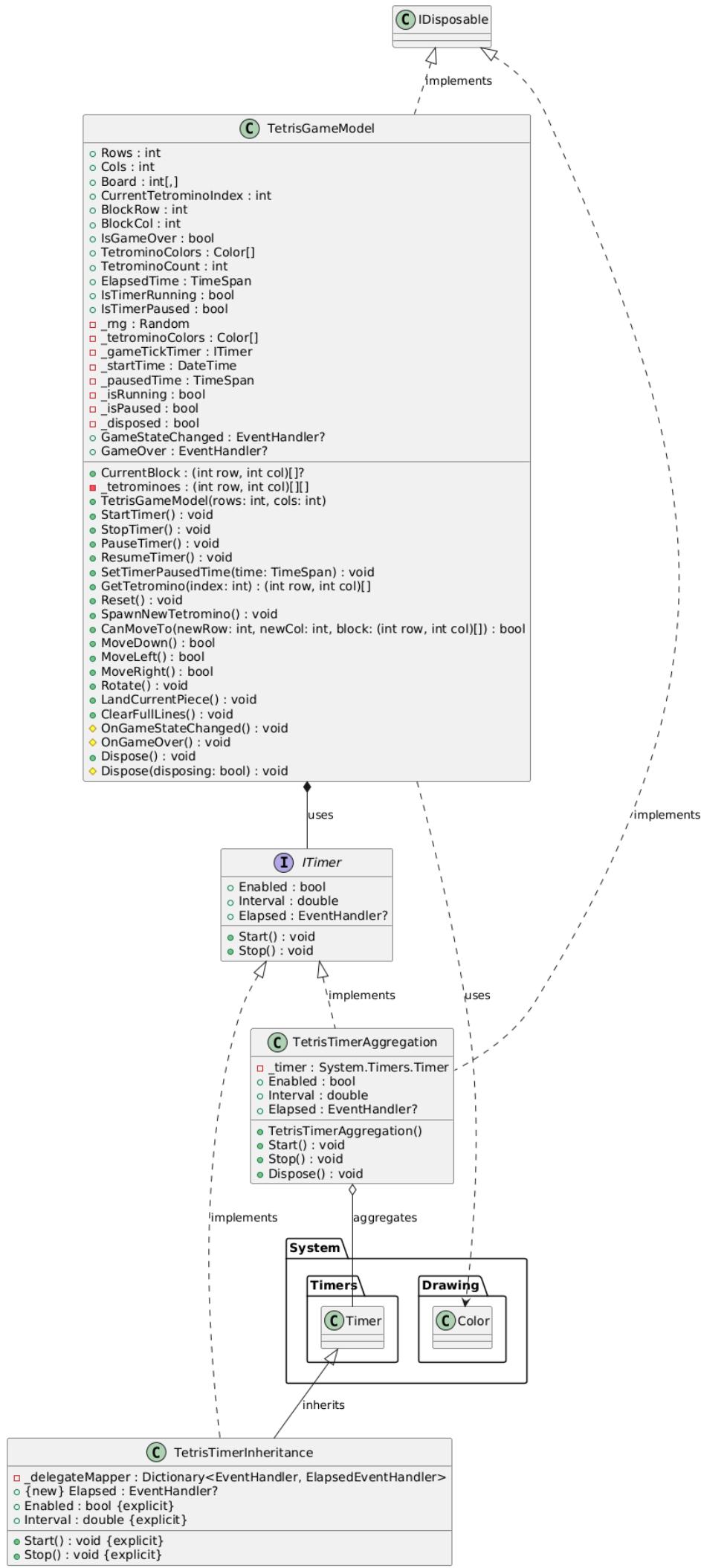
- Perzisztencia:

- Az adatkezelés feladata a Tetris táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása. A TetrisPersistence osztály egy érvényes Tetris táblát biztosít (azaz minden ellenőrzi a beállított értékeket). A tábla tartalmazza a méretet (Rows, Cols), a tábla tartalmát (Board), a jelenleg aktív tetrominó indexét és dimenzióit (CurrentTetrominoIndex, CurrentBlock), stb.
- A hosszú távú adattárolás lehetőségeit az ITetrisDataAccess interfész adja meg, amely lehetőséget ad a tábla betöltésére (LoadAsync), valamint mentésére (SaveAsync). A műveleteket hatékonyiségi okokból aszinkron módon valósítjuk meg.
- Az interfészt szöveges fájl alapú adatkezelésre a TetrisFileDataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a TetrisDataException kivétel jelzi.

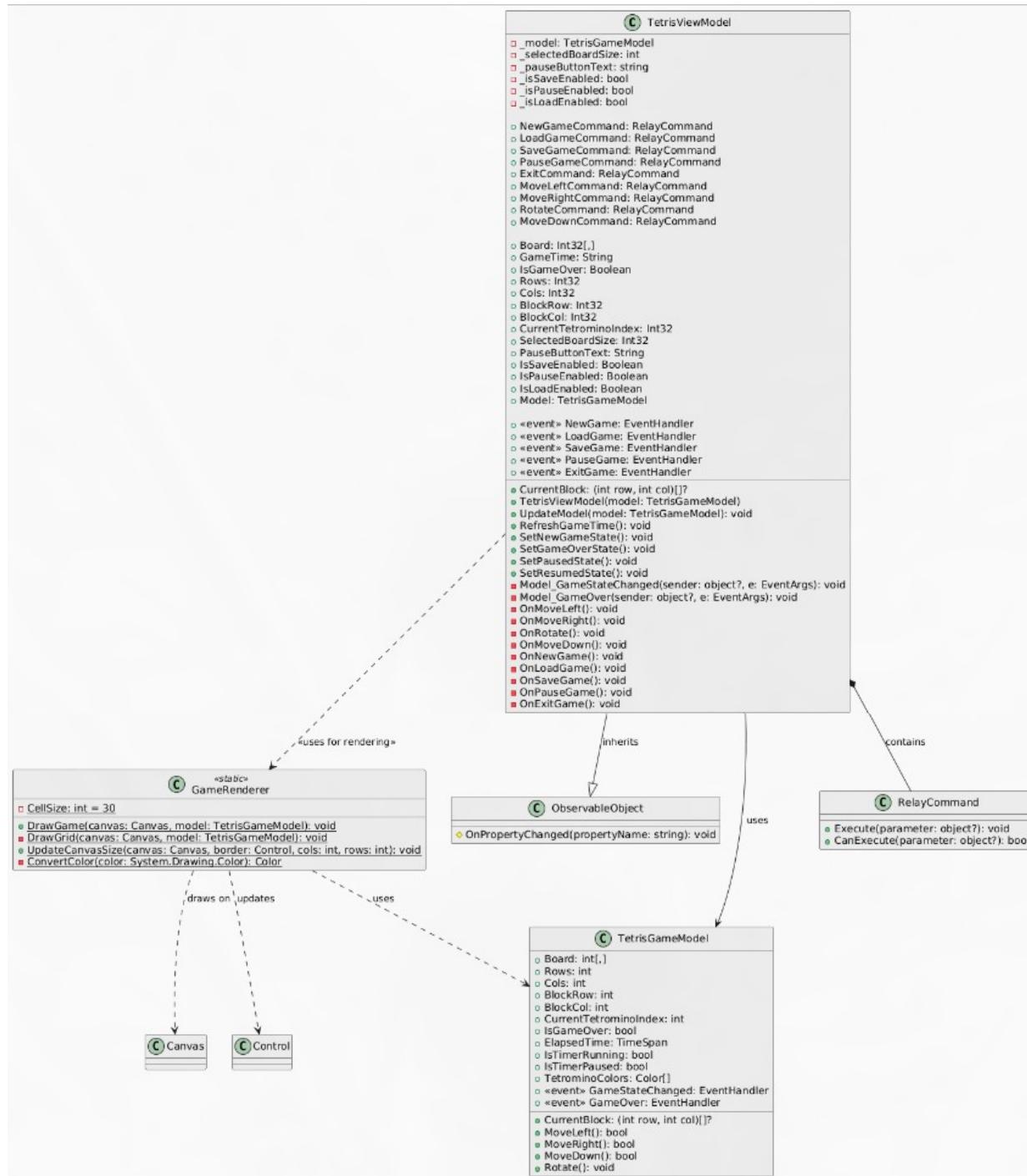


3. ábra: Az Persistance csomag osztálydiagramja

- Modell (4. ábra):
  - A modell lényegi részét a TetrisGameModel osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit. Új játéknál megadható a kiinduló játéktábla is.
  - A játék időbeli kezelését egy időzítő végzi (`_timer`), amelyet inaktiválunk majd (`PauseGame`), amennyiben bizonyos menüfunkciók futnak, majd újraindítjuk (`ResumeGame`).
  - A játékállapot megváltozásáról a `GameStateChange` esemény, míg a játék végéről a `GameOver` esemény tájékoztat. Az események argumentuma (`TetrisGameEventArgs`) tárolja a győzelem állapotát, a lépések számát, valamint a játékidőt.
  - A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (`LoadGameAsync`) és mentésre (`SaveGameAsync`)



- Nézetmodell (5. ábra):
  - A nézetmodell megvalósításához felhasználjuk az MVVM Toolkit csomagból elérhető általános utasítás (RelayCommand), valamint egy ős változásjelző (ObservableObject) osztályt.
  - A nézetmodell feladatait a SudokuViewModel osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, mentéséhez, valamint a kilépéshez. 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 játéknehezséget szabályozza. Direkt nem avatkozik a játék futtatásába.

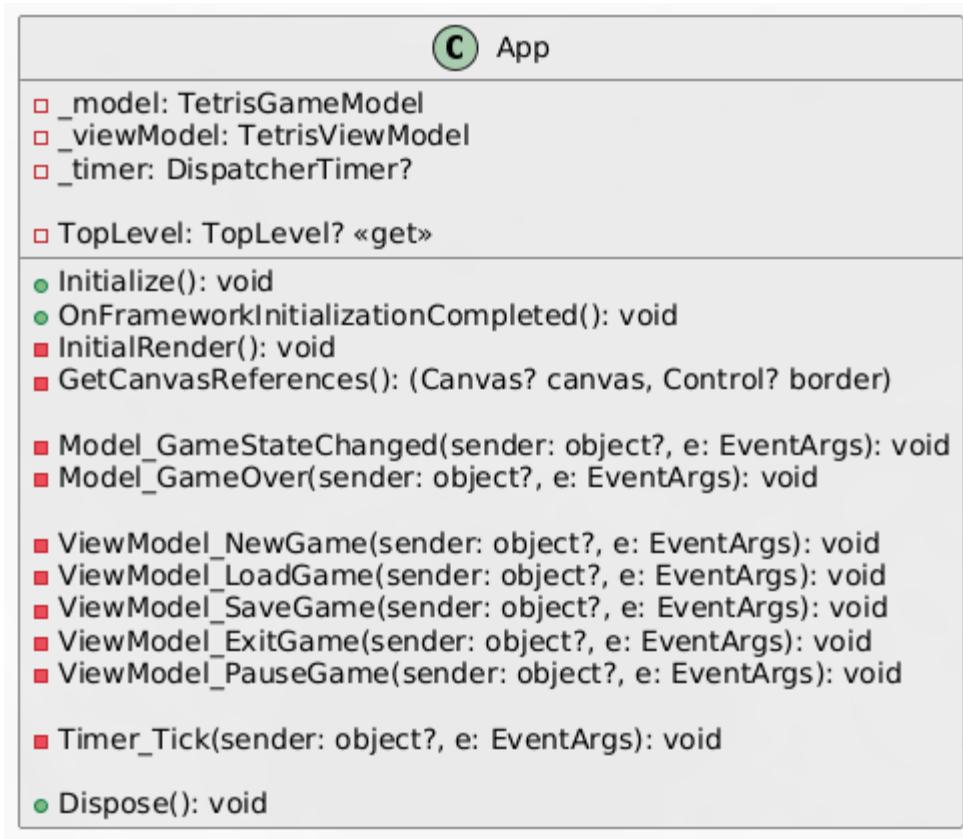


- Nézet:
  - A nézetet ablakok, egyedi vezérlők és dialógusablakok használatával valósítjuk meg.
  - A MainView osztály, mint UserControl leszármazott tartalmazza a játéktáblát, amelyet egy UniformGrid segítségével valósítunk meg, amelyben Button elemeket helyezünk el.
  - A MainWindow ablakra egyszerűen a MainView vezérlőt ágyazzuk be. Ilyen módon a felület asztali alkalmazásokban ablakos alkalmazásként, mobil platformon pedig lapként is megjeleníthető.
  - Betöltéshez és mentéshez a StorageProvider osztály által az OpenFilePickerAsync és SaveFilePickerAsync metódusokon keresztül biztosított, operációs rendszer specifikus dialógus ablakokat (lapokat) használjuk.
  - Felugró üzenetek megjelenítéséhez a MessageBox.Avalonia NuGet csomagot használjuk.

Vezérés (6. ábra):

- Az App osztály feladata az alkalmazás vezérlése, a rétegek példányosítása és az események feldolgozása.
- A OnFrameworkInitializationCompleted metódus felüldefiniálásával kezeljük a nézet platform specifikus megjelenítését, továbbá az alkalmazás életciklusát a megfelelő eseményekre történő feliratkozással. Így különösen mobil célplatform esetén az alkalmazás felfüggesztéskor (Deactivated) elmentjük az aktuális játékállást (SuspendedGame), míg folytatáskor vagy újraindításkor (Activated) pedig folytatjuk, amennyiben történt mentés.

6. ábra: A vezérlés osztálydiagramja



- Tesztelés:
  - A modell funkcionalitása egységesztek segítségével lett ellenőrizve a TetrisGameModelTest osztályban.
  - A modell időzítőjét egy ITimer interfészről származtattuk le, így azt a teszt projektben egy MockTimer megvalósítással mockolhatjuk.
  - Az alábbi tesztesetek kerültek megvalósításra:
    - InitializeTest,
    - ResetTest,
    - MoveDownTest,
    - MoveRightTest,
    - MoveDownBlockedTest,
    - LandPieceTest,
    - SpawnTetrominoTest,
    - ClearLineTest,
    - DetectWallsTest,
    - RotatesBlockCorrectly