2. Beadandó feladat dokumentáció

Készítette:

Csányi Gábor

E-mail: a4shnl@inf.elte.hu

Feladat:

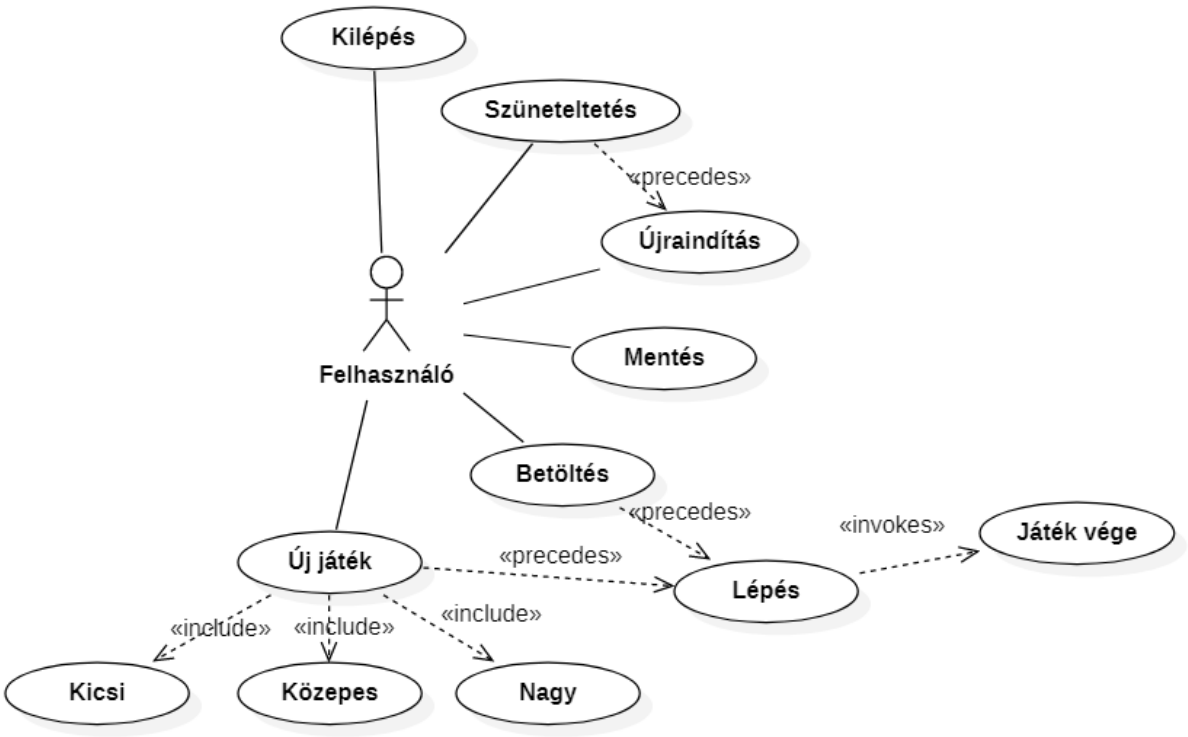
Készítsünk programot, amellyel az alábbi Reversi játékot játszhatjuk.
A játékot két játékos játssza $n \times n$-es négyzetrácsos táblán fekete és fehér korongokkal. Kezdéskor a tábla közepén X alakban két-két korong van elhelyezve mindkét színből. A játékosok felváltva tesznek le újabb korongokat. A játék lényege, hogy a lépés befejezéseként az ellenfél ollóba fogott, azaz két oldalról (vízszintesen, függőlegesen vagy átlósan) közrezárt bábuit (egy lépésben akár több irányban is) a saját színünkre cseréljük.
Mindkét játékosnak, minden lépésben ütnie kell. Ha egy állásban nincs olyan lépés, amivel a játékos ollóba tudna fogni legalább egy ellenséges korongot, passzolnia kell és újra ellenfele lép. A játékosok célja, hogy a játék végére minél több saját színű korongjuk legyen a táblán.
A játék akkor ér véget, ha a tábla megtelik, vagy ha mindkét játékos passzol. A játék győztese az a játékos, akinek a játék végén több korongja van a táblán. A játék döntetlen, ha mindkét játékosnak ugyanannyi korongja van a játék végén.
A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (10×10, 20×20, 30×30), játék szüneteltetésére, valamint játék mentésére és betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött. A program folyamatosan jelezze külön-külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépéséig tart, ezt is mentsük el és töltsük be).

Elemzés:

- A játékot három táblamérettel játszhatjuk: kicsi (10x10), közepes (20x20), nagy (30x30). A program indításkor közepes táblaméretet állít be, és automatikusan új játékot indít.
- A feladatot egyablakos asztali alkalmazásként Windows Presentation Foundation grafikus felülettel valósítjuk meg.
- Az ablakon elhelyezünk egy menüt a következő menüpontokkal: File [New Game (10x10, 20x20, 30x30), Pause, Restart, Open, Save, Exit]. Az ablak alján megjelenítünk egy státuszsort, amely a játékosok gondolkodási idejét jelzi, és kiszínezi a soron következő játékost.
- A játéktáblát a táblamérettől függően 10x10, 20x20 vagy 30x30 nyomógomból álló rács reprezentálja. A nyomógomb egérkattintás hatására a játék szabályainak megfelelően megváltoztatja a mezők színeit. Csak az üres mezőkre lehet kattintani.
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (megtelt a tábla vagy, mindkét játékos passzolt) vagy ha nem tud lépni az egyik játékos. Szintén

dialógusablakkal végezzük el a mentést, illetve betöltést, a fájlnevet a felhasználó adja meg.

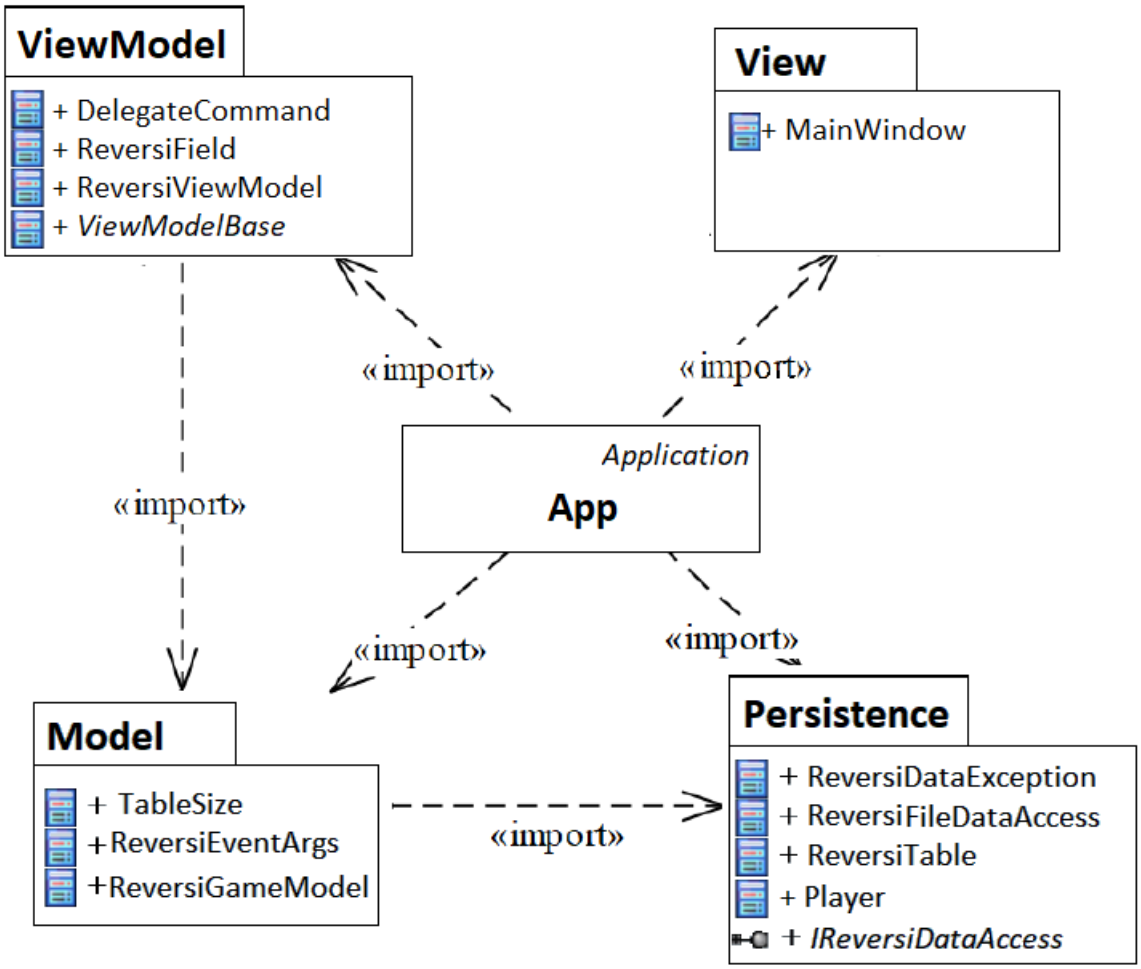
- A felhasználói esetek az 1. ábrán láthatóak.



1. ábra: Felhasználói esetek diagramja

Tervezés:

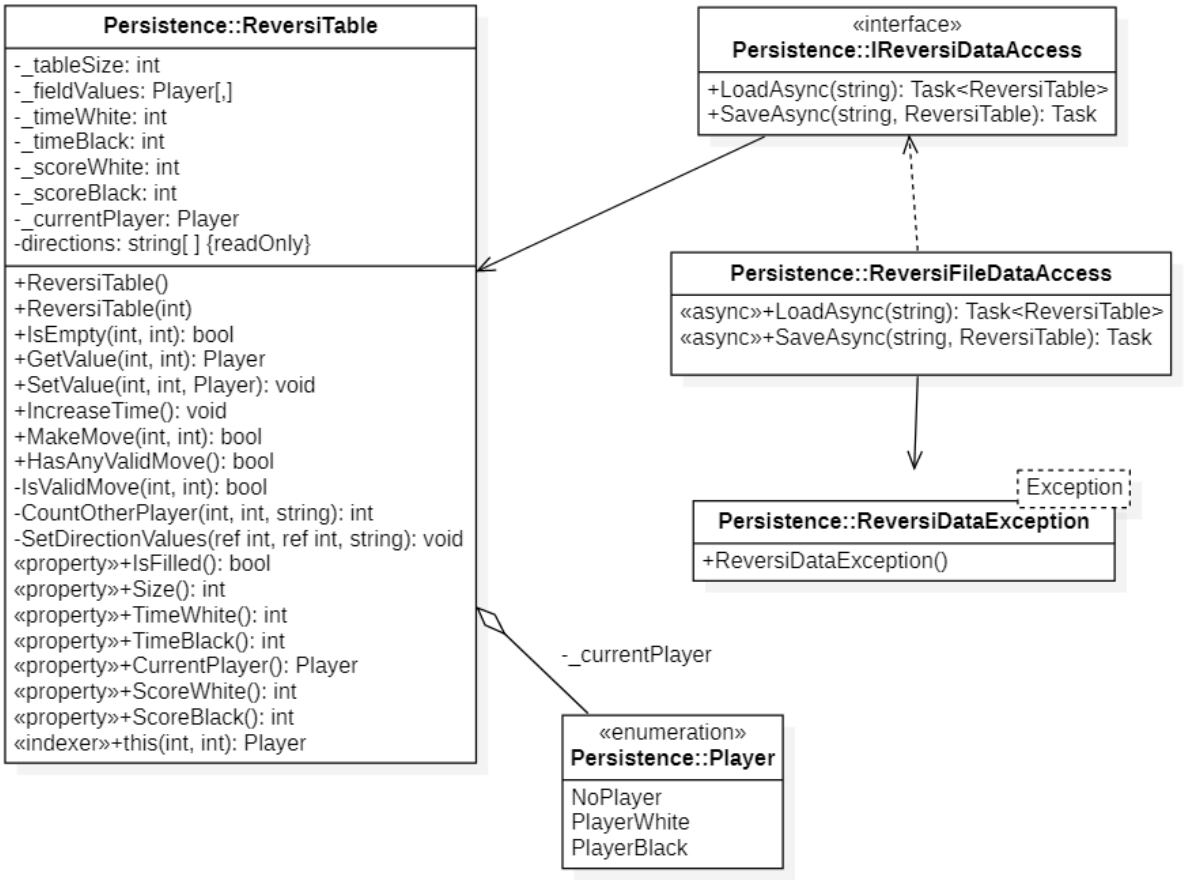
- Programszerkezet:
  - A programot MVVM architektúrában valósítjuk meg, ennek megfelelően `View`, `Model`, `ViewModel` és `Persistence` névtereket valósítunk meg az alkalmazáson belül. A program környezetét az alkalmazás osztály (`App`) végzi, amely példányosítja a modellt, a nézetmodell és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést.
  - A program csomagszerkezete a 2. ábrán látható.



2. ábra: Az alkalmazás csomagdiagramja

- Perzisztencia (3. ábra):
  - Az adatelérés feladata a Reversi táblával kapcsolatos információk tárolása, valamint betöltés/mentés biztosítása.
  - A `ReversiTable` egy Reversi táblát biztosít, ahol, minden mezőre ismert az értéke (`_fieldValues`), ami lehet `NoPlayer`, `PlayerWhite` vagy `PlayerBlack`, tároljuk még az aktuális játékost, azok pontjait és gondolkodási időit. A tábla alapértelmezés szerint 20x20-as de ez a konstruktorban paraméterezhető. A tábla lehetőséget ad az állapotok lekérdezésére (`isFilled`, `isEmpty`, `GetValue`, `HasAnyVAlidMove`) valamint lépésre (`MakeMove`), az idő léptetésére (`IncreaseTime`) illetve direkt beállítás (`SetValue`) elvégzésére.
  - Hosszú távú adattárolás lehetőségeit az `IReversiDataAccess` interfész adja meg, amely lehetőséget ad a tábla betöltésére (`LoadAsync`), valamint mentésére (`SaveAsync`). A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg.

- Az interfészt szöveges alapú adatkezelésre a `ReversiFileDataAccess` osztály valósítja meg. A fájlkezelés során fellépő hibákat a `ReversiDataException` kivétel jelzi.
- A program szöveges fájlként tudja eltárolni, melyek `rtl` kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
- A fájl első sora megadja a tábla méretét, második sora a játékosok gondolkodási idejét, harmadik sora a játékosok pontszámait, negyedik sora a soron következő játékost. Ezután a játéktábla sorai következnek, a 0 reprezentálja az üres mezőket, a fehér játékost az 1-es, a feketét pedig a 2-es szám.



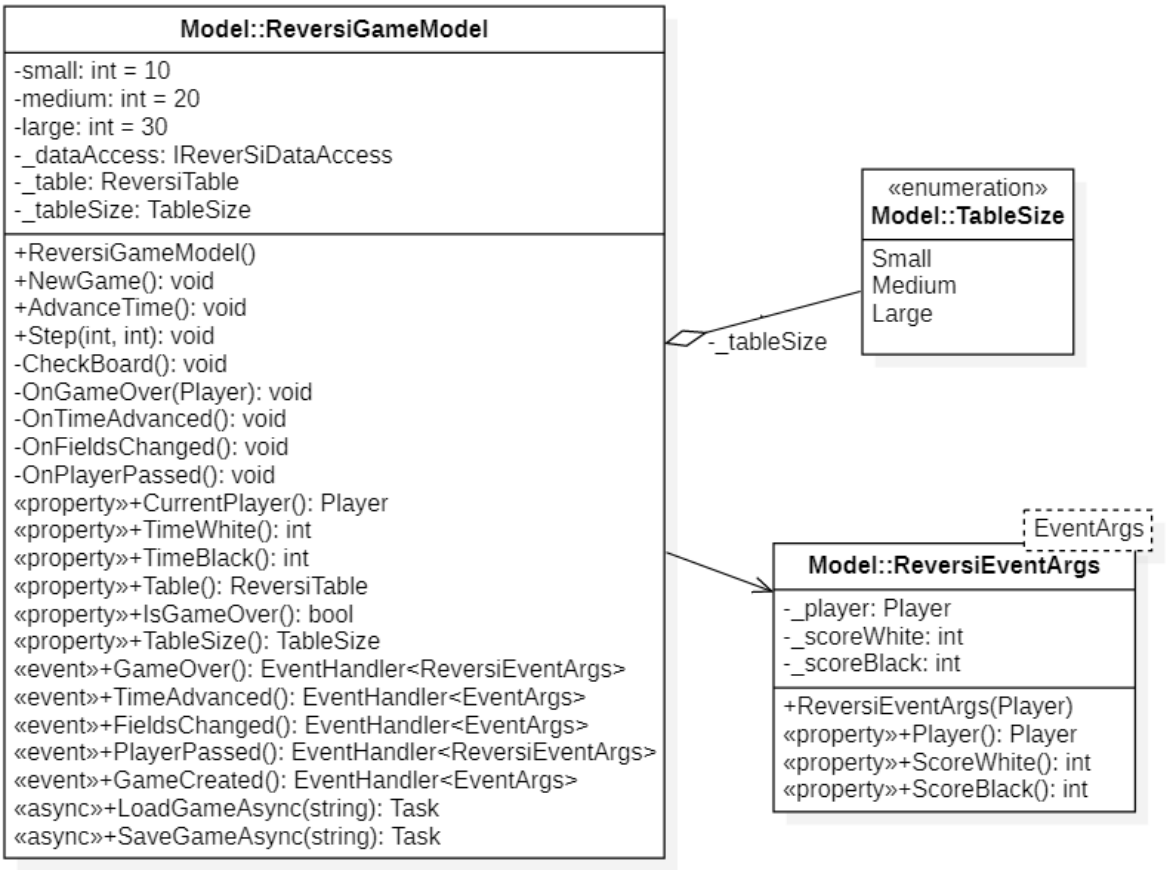
3. ábra: A Persistence csomag osztálydiagramja

- Modell (4. ábra):
  - A modell lényegi részét a `ReversiGameModel` osztály valósítja meg, amely szabályozza a tábla tevékenységét. A típus lehetőséget ad új játék kezdésére (`NewGame`), valamint lépésre (`StepGame`). Az idő előreléptetését időbeli lépések végzésével (`AdvanceTime`) tehetjük meg.

- A játék létrejöttéről a `GameCreated`, az idő változásáról a `TimeAdvanced`, a mezők állapotának változásáról a `FieldsChanged`. Passzolásról a `PlayerPassed` esemény tájékoztat a játék végéről a `GameOver` esemény ad információt ezeknek az argumentuma (`ReversiEventArgs`) tárolja a nyertes játékost és a pontszámokat, döntetlen esetén `NoPlayer` az értéke.

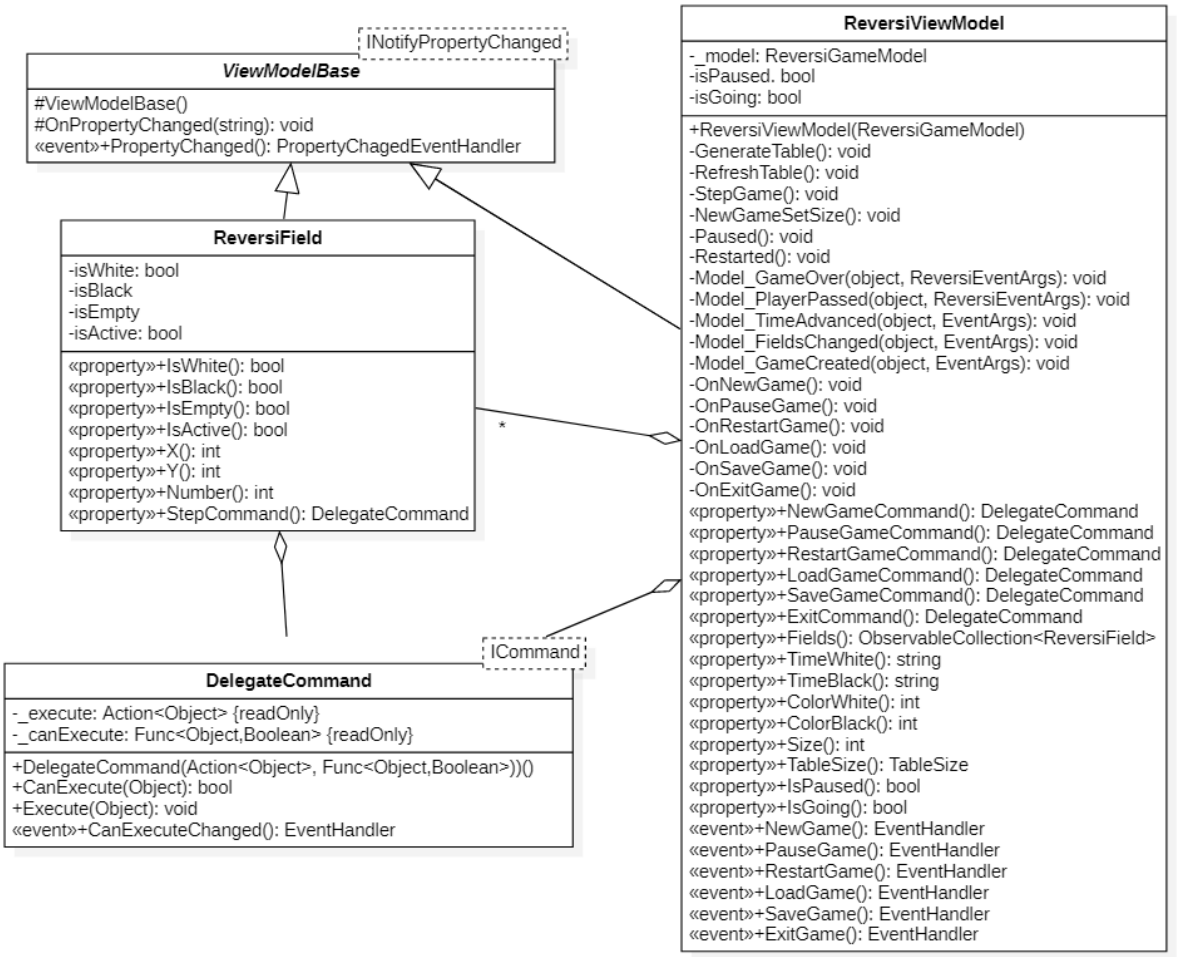
- A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (`LoadGameAsync`) és mentésre (`SaveGameAsync`)

- A tábla méretét a `TablaSize` felsorolási típuson át kezeljük, és a `ReversiGameModel` osztályban konstansok segítségével tároljuk az egyes táblaméreteket.



4. ábra: A Model csomag osztálydiagramja

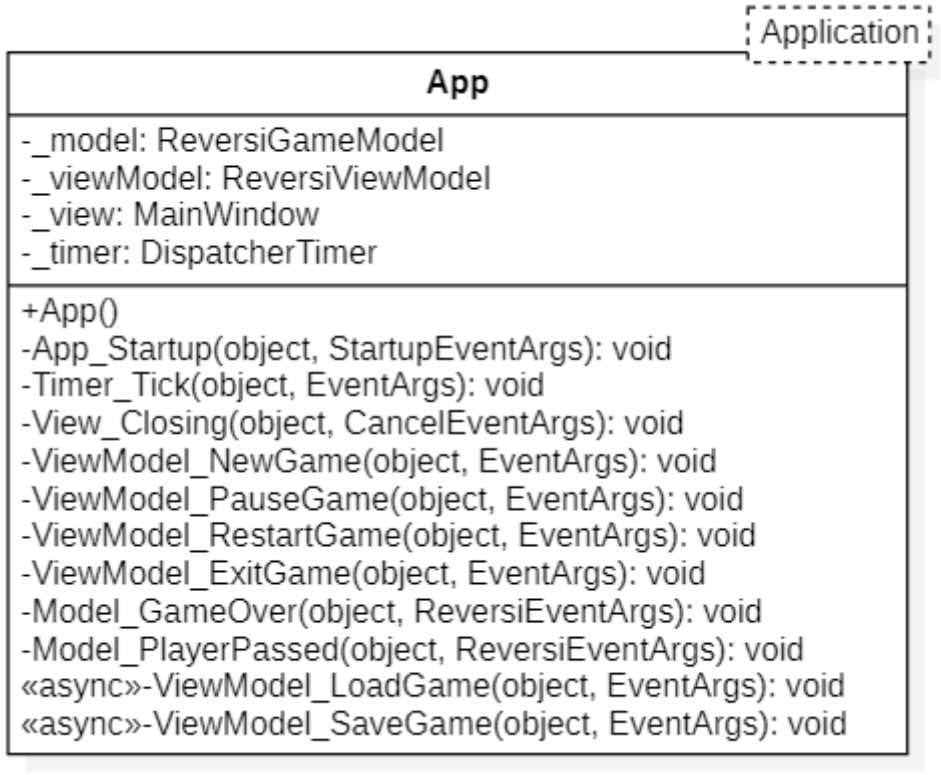
- Nézetmodell (5.ábra):
    - A nézetmodell megvalósításához felhasználunk egy általános utasítás (`DelegateCommand`), valamint egy ős változásjelző (`ViewModelBase`) osztályt.
    - A nézetmodell feladatait a `ReversiViewModel` osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék megállításához, újraindításához, 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 táblaméretet szabályozza. Direkt nem avatkozik a játék futtatásába.
    - A játékmező számára egy külön mezőt biztosítunk (`ReversiField`), amely eltárolja a pozíciót, szín, engedélyezettséget, valamint a lépés parancsát (`StepCommand`). A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (`Fields`).



5. ábra: A nézetmodell osztálydiagramja

- 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.
  - A játék léptetéséhez tárol egy időzítőt is (`_timer`), amelynek állítását is szabályozza az egyes funkciók hatására.



6. ábra: A vezérlés osztálydiagramja

Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a `ReversiGameModel` osztályban.

- Az alábbi tesztesetek kerültek megvalósításra:
  - `ReversiGameModelNewGameSmallTest`
  - `ReversiGameModelNewGameMediumTest`
  - `ReversiGameModelNewGameLargeTest`: Új játék indítása, a táblaméret, kezdőjátékos, játékosok pontjainak, időinek ellenőrzése.
  - `ReversiGameModelStepTest`: Játékbeli lépés hatásinak ellenőrzése.
  - `ReversiGameModelAdvanceTimeTest`: A játékosok gondolkodási időinek kezelésének ellenőrzése.
  - `ReversiGameModelLoadTest`: A játék modell betöltésének tesztelése mockolt perzisztencia réteggel.