2. Beadandó feladat dokumentáció

Készítette:

Tóth Bence

E-mail: m2s8c6@inf.elte.hu

Feladat:

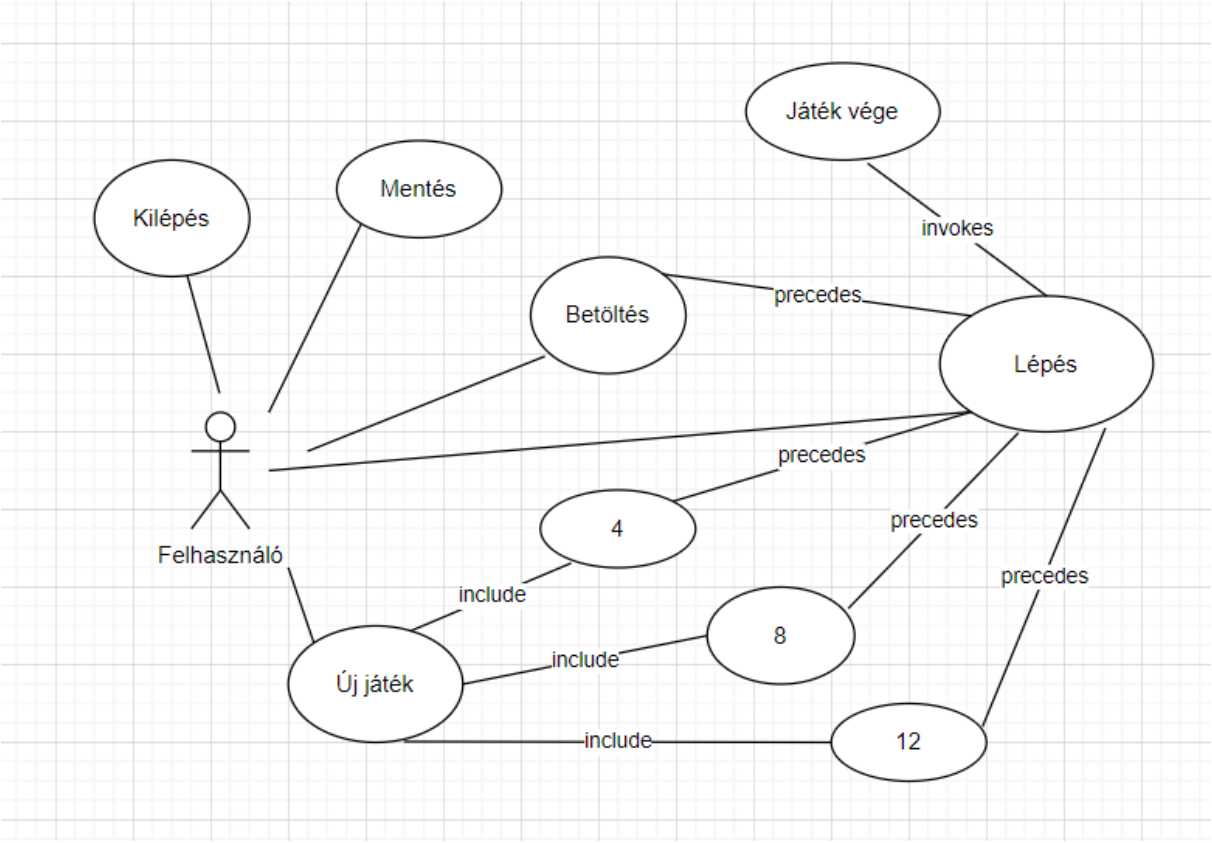
Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Két játékos egymással szemben helyezkedik el, közöttük pedig (paraméterként megadható) páros számú tálka és két gyűjtőtál az alábbi lerendezésben.

Mindkét játékos a hozzá közelebbi tálkákat és a tőle jobb kézre eső gyűjtőtálat mondhatja sajátjának. (Így az ellenfél gyűjtőtálja baloldalra esik.) Kezdetben mindegyik tálkában 6-6 kavics van, a gyűjtőtálak pedig üresek. A játékban a soron következő játékos kiválasztja egyik saját tálkáját (ez nem lehet a gyűjtőtál), hogy azt kiürítse úgy, hogy tartalmát az óramutató járásával ellentétes irányban egyesével beledobálja, majd ismét a saját tálkáiba, de azt a tálkát kihagyva, amelyiknek a kiürítését végezzük, amíg el nem fogynak a kavicsok. Ha az egyik játékos rákattint valamelyik tálkájára, akkor a tálkában lévő kavicsok áthelyezése automatikusan történjen meg. Ha az utolsó kavics a játékos saját üres tálkáinak egyikébe kerül, akkor ezt a kavicsot, valamint a szemközti tálka tartalmát a saját gyűjtőládába teszi. Viszont, ha az utolsó kavics a játékos saját gyűjtőtálkájába esik, akkor újra ő következik, de ezt csak egyszer teheti meg, hogy ellenfele is szóhoz juthasson. A játéknak akkor van vége, ha az egyik térfél kiürült, azaz az egyik játékos tálkái mind kiürülnek. Ekkor az a játékos nyeri a játékot, akinek a gyűjtőtáljában több kavics van. A program biztosítson lehetőséget új játék kezdésére a tálkák számának megadásával (4, 8, 12), játék mentésére és betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

Elemzés:

- A játékot három féle tál számmal játszhatjuk: 4 (6 előre generált tál), 8 (10 előre generált tál), 12 (14 előre generált tál). A program indításkor 8 db tálat állít be, és automatikusan új játékot indít.
- A feladatot egyablakos asztali alkalmazásként Windows Presentation Foundation grafikus felülettel valósítjuk meg.

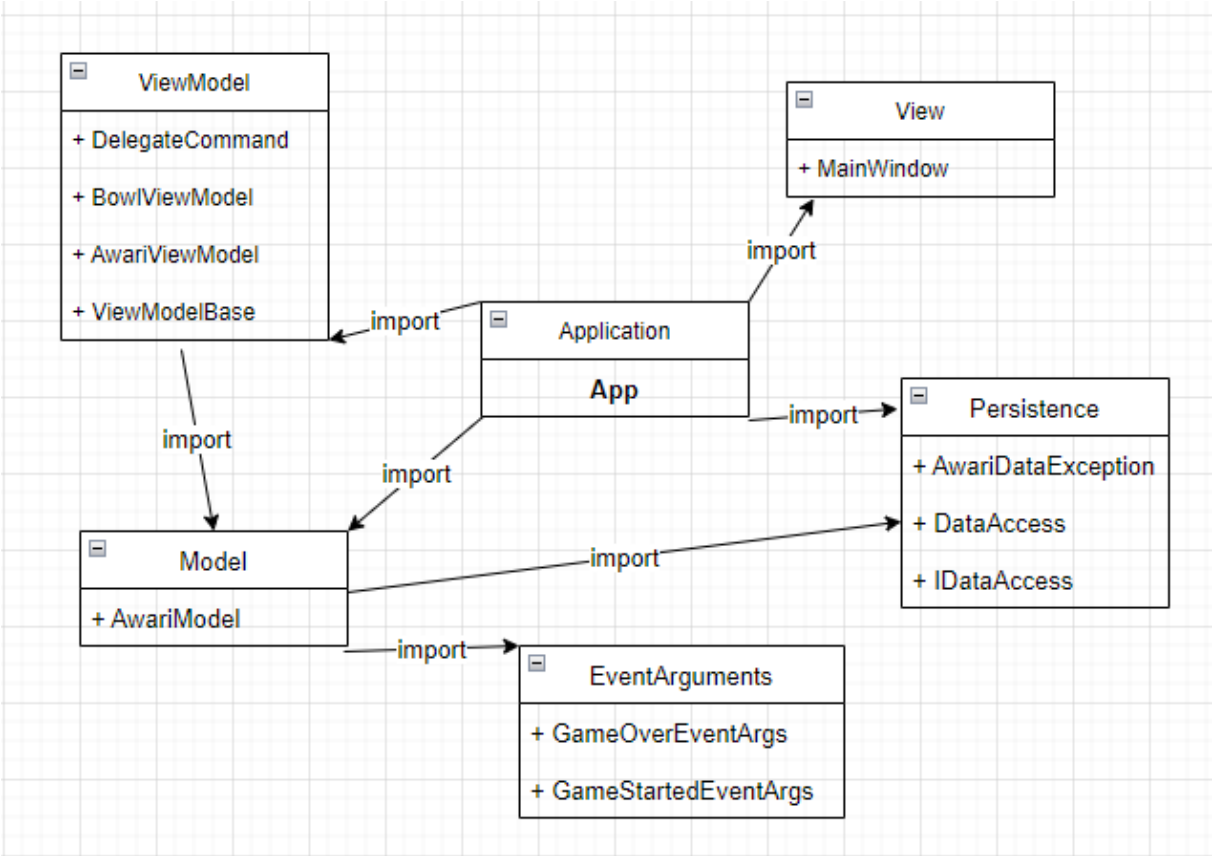
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (Save game, Load game), New Game (4 bowls, 8 bowls, 12 bowls).
- A játékteret a tálak számából adódóan létrehozott számú nyomógombok reprezentálják. A nyomógomb egérkattintás hatására kiosztja a benne lévő kavicsokat. A kavicsokat egyesével osztjuk ki a tálak között óramutatóval ellentétes irányban, abban az esetben, ha körbe ért újra kezdődik addig amíg el nem fogy, de azt a tálat kihagyva amelyikből osztjuk ki a kavicsokat. Abban az esetben, ha az utolsó kavics a játékos saját gyűjtőtálába érkezett akkor újra következik, de erre csak egyszer van lehetősége. Emellett, ha az utolsó kavics a játékos egyik saját üres táljába landolt, az utolsó kavicsot, és az azzal tállal szemben lévő ellenséges tál tartalmát a saját gyűjtőtálába teszi.
- A játék automatikusan feldob egy dialógusablakot, közölve a győztest, amikor vége a játéknak (Valamelyik játékos összes tálja üres). 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:

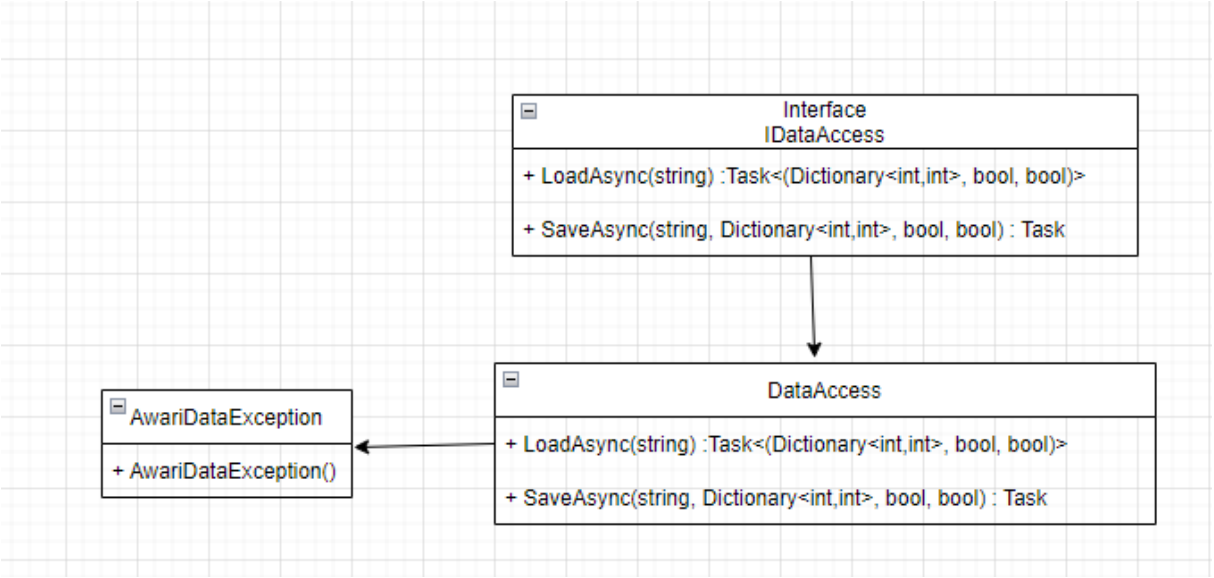
- Programszerkezet:
  - A programot MVVM architektúrában valósítjuk meg, ennek megfelelően View, Model, ViewModel és Persistence névtereket valósítunk meg az alkalmazáson belül. A program környezetét az alkalmazás osztály (App) végzi, amely példányosítja a modellt, a nézetmodell és a nézetet, biztosítja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



2. **Az alkalmazás csomagdiagramja**

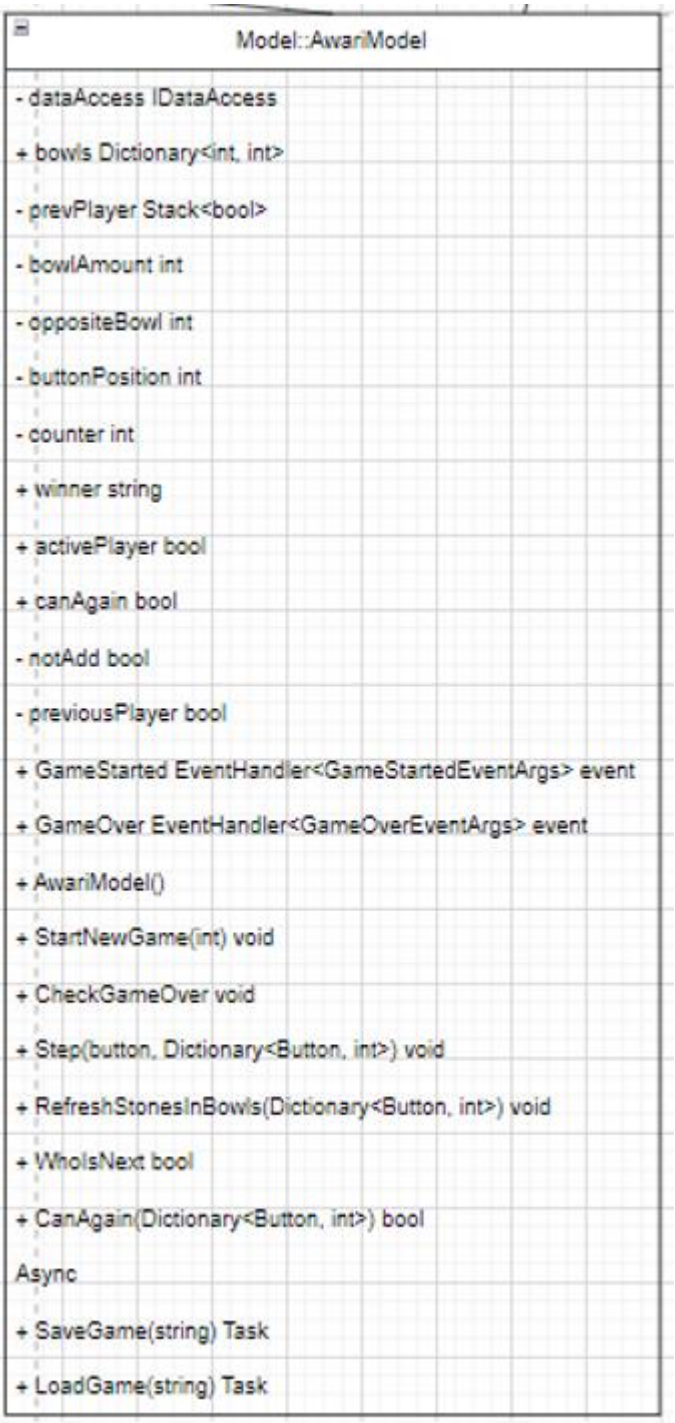
- Perzisztencia(3. ábra):
  - Az adatkezelés feladata a betöltés/mentés biztosítása.
  - A hosszú távú adattárolás lehetőségeit az IDataAccess interfész adja meg, amely lehetőséget ad a nyomógombok betöltésére (LoadAsync), valamint mentésére (SaveAsync). A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg.
  - Az interfészt szöveges fájl alapú adatkezelésre a DataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a AwariDataException kivétel jelzi.

- A program az adatokat szöveges fájlként tudja eltárolni, melyek az txt kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
- A fájl első sora megadja, hogy 4, 8 vagy 12 tállal játszott játék volt, valamint, hogy a piros vagy kék játékos következett és hogy jöhetett e újra (ami alapértelmezetten a piros és nem). A fájl többi része izomorf leképezése a tálaknak, azaz összesen a tálak számával megegyező szám következik, ami a tálakban lévő kavicsok értékét adja meg, ahol 0 reprezentálja az üres tálat.



3. ábra: A Persistence csomag osztálydiagramja

- Modell(4. ábra):
  - A modell lényegi részét az AwariModel osztály valósítja meg, amely szabályozza a tálak tevékenységeit, valamint a játék egyéb paramétereit, úgymint az aktív játékost (activePlayer) és hogy újra jöhet e az adott jtáékos(canAgain). Valamint az adatelérés konkrét példányát (dataAccess). A típus lehetőséget ad új játék kezdésére (StartNewGame), valamint lépésre (Step). Új játéknál automatikusan generálódnak kezdő tálak.
  - A játék kezdetéről a GameStarted esemény, míg a játék végéről a GameOver esemény tájékoztat. Az események argumentuma (GameStartedEventArgs és GameOverEventArgs) tárolja a tálakat, aktív játékost, jöhet e újra majd a győztes nevét és a tálak számát.
  - A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (LoadGame) és mentésre (SaveGame).

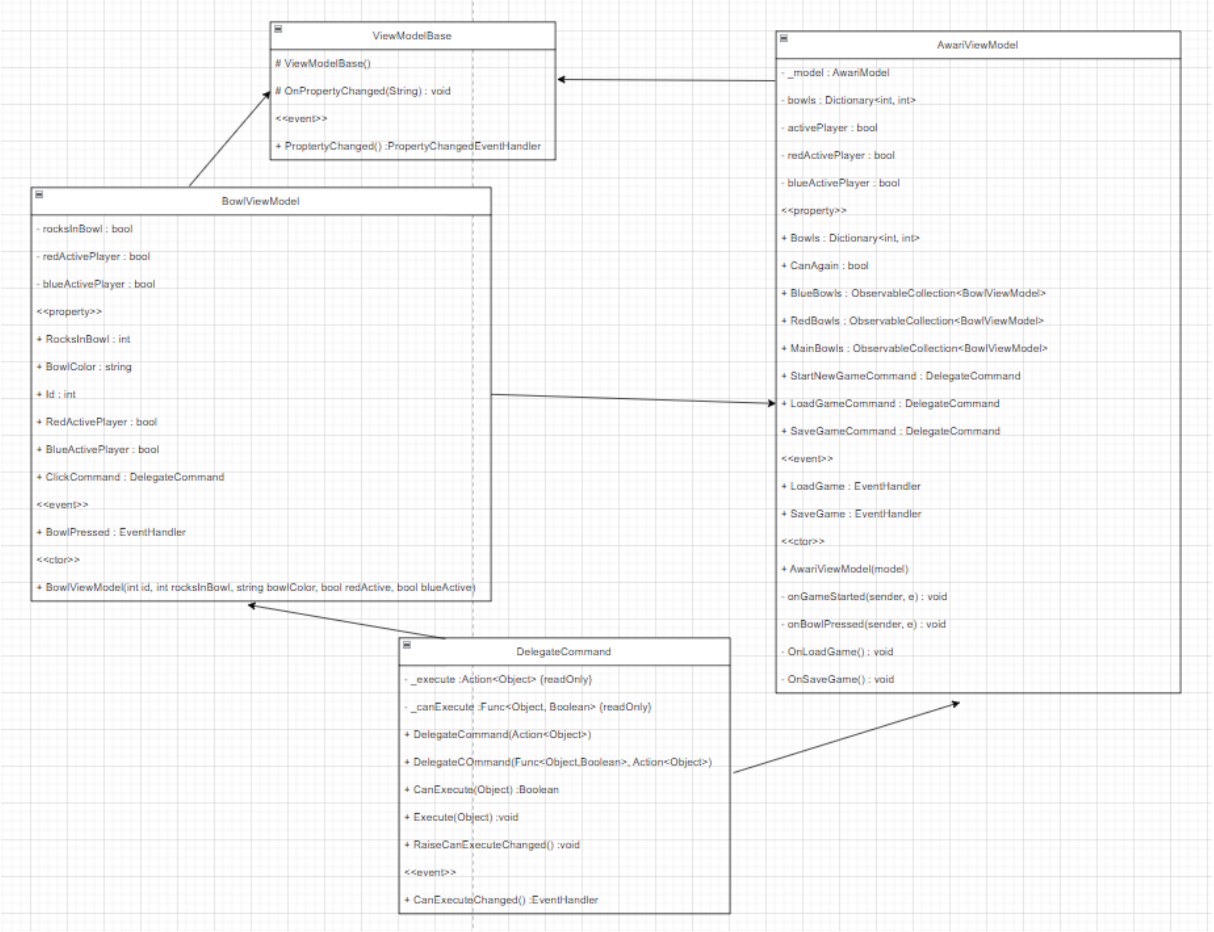


4. ábra: A Model csomag osztálydiagramja

- Nézetmodell(5. ábra):
  - A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) osztályt.
  - A nézetmodell feladatait az AwariViewModel osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, 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.

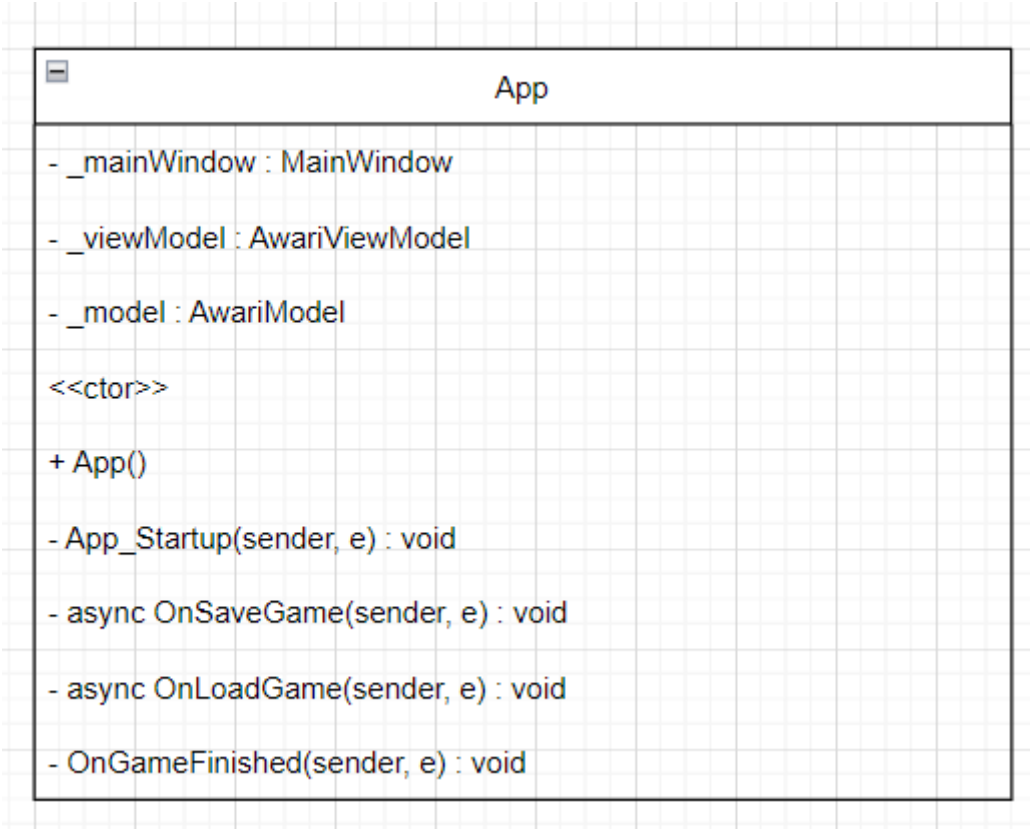
- A tálak számára egy külön ViewModelt biztosítunk (BowlViewModel), amely eltárolja az id-t, kavicsok számát, aktív játékost, piros e vagy kék e az aktív, valamint a lépés parancsát (ClickCommand). A mezőket három felügyelt gyűjteménybe helyezzük a nézetmodellbe (BlueBowls, RedBowls, MainBowls).



5. ábra: A nézetmodell osztálydiagramja

- Nézet:
  - A nézet csak egy képernyőt tartalmaz, a MainWindow osztályt. A játékmező három ItemsControl vezérlő, ahol dinamikusan felépítünk egy két rácsot (UniformGrid) és egy lineáris elrendezőt (StackPanel), amelyek gombokból áll. Minden adatot adatkötéssel kapcsolunk a felülethez, továbbá azon keresztül szabályozzuk a gombok színét is és a tálak értékét 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.



6. ábra: A vezérlés osztálydiagramja

Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a AwariTest osztályban.

- Az alábbi tesztesetek kerültek megvalósításra:

    - Tálak létrehozása esetében egyenlő méret, és ugyan azon adatok
    - CheckGameOver metódus helyes lefutás minden opcióban
    - WhoIsNext metódis helyes lefutás minden opcióban
    - LoadGame: A játék modell betöltésének tesztelése mockolt perzisztencia réteggel.