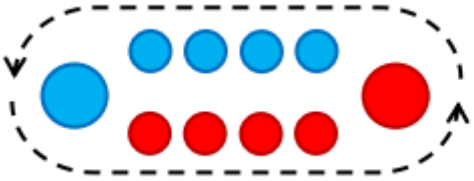
Készítette:

Bodó József
E-mail: i7p4uq@inf.elte.hu
Csoportszám: 7

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 háromféle tábla mérettel játszhatjuk: 4 kis tálkával (2-2 az egyes játékosoknak), 8 kis tálkával (4 - 4), illetve 12 kis tálkával. A program indításkor 8 kis tálkát állít be, és automatikusan új játékot indít.

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

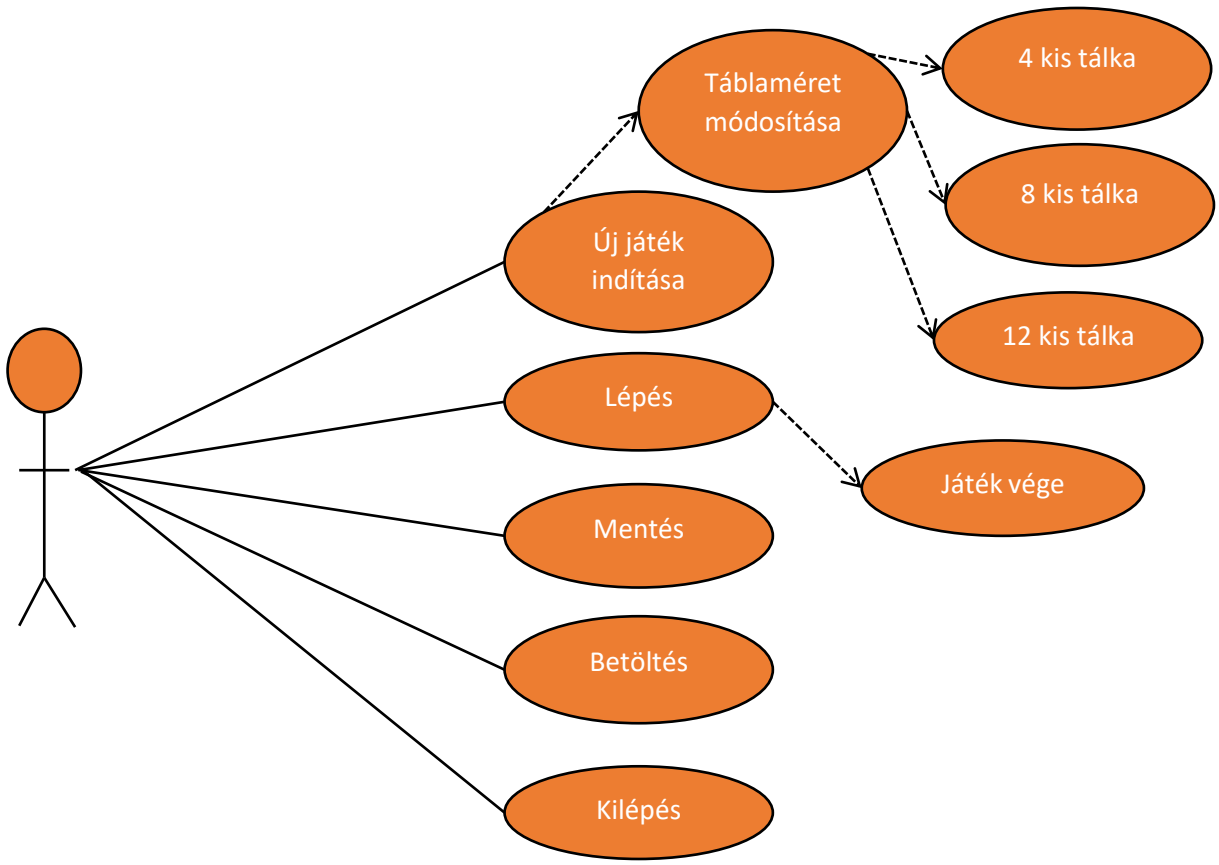
Az ablakban elhelyezünk egy menüt a következő menüpontokkal: **Új játék** (*New Small Game, New Medium Game, New Large Game*), **Játék betöltése** (*Load Game*), **Játék mentése** (*Save Game*).

A játéktábla mérete függ a felhasználó által kiválasztott **tálkák számától** (ez legyen most *n*), ekkor 3 x n-es nyomógombokból álló rács reprezentálja, amelyben nem mindegyik nyomógomb van megvalósítva. A táblán az alsó sorban megjelenő, illetve a fölötte lévő sorban jobbra a gombok piros-, a felső sorban megjelenő, illetve az alatta lévő sorban balra a gombok kék színűek.

A kis tálkákat reprezentáló gombok felirata alapból 6-ra van állítva, míg a nagy gyűjtőtálkákat reprezentáló gombok felirata alapból üres. A játék közben a gombok feliratát változtatjuk, ezzel megjelenítve az aktuális játékállapotot.

A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak, és jelzi az eredményt: ha valamelyik játékos nyert, akkor egy olyan üzenetet ír ki, illetve, ha döntetlen, akkor ezt jelzi. Dialógusablakokkal végezzük el a mentést, illetve a betöltést, a fájlneveket a felhasználó adja meg.

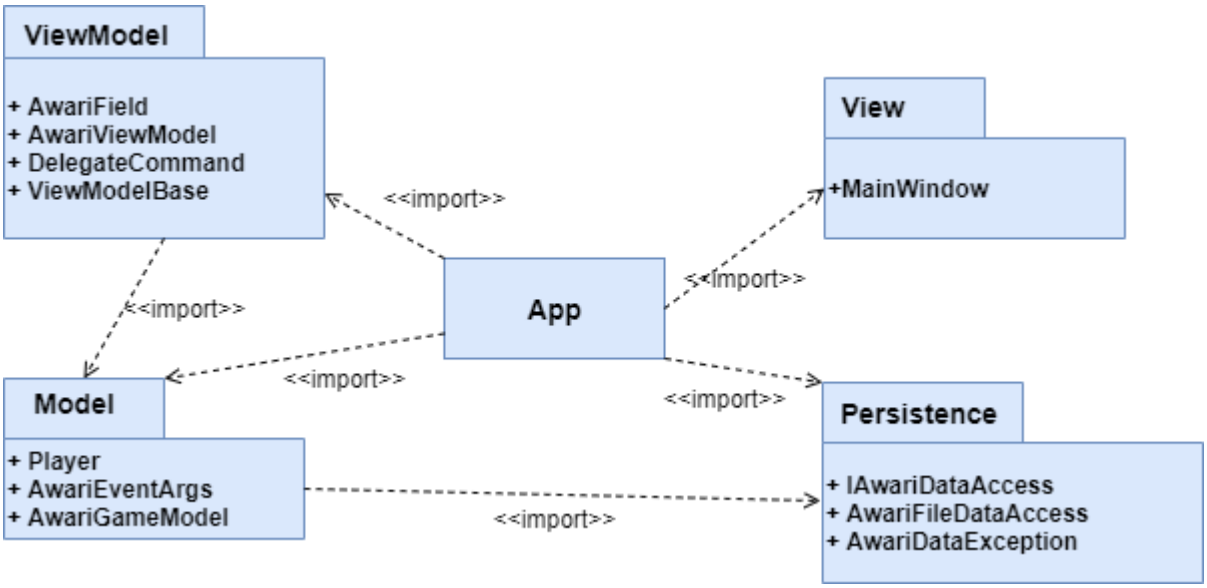
Felhasználói esetek:



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 nézetmodellt és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést.



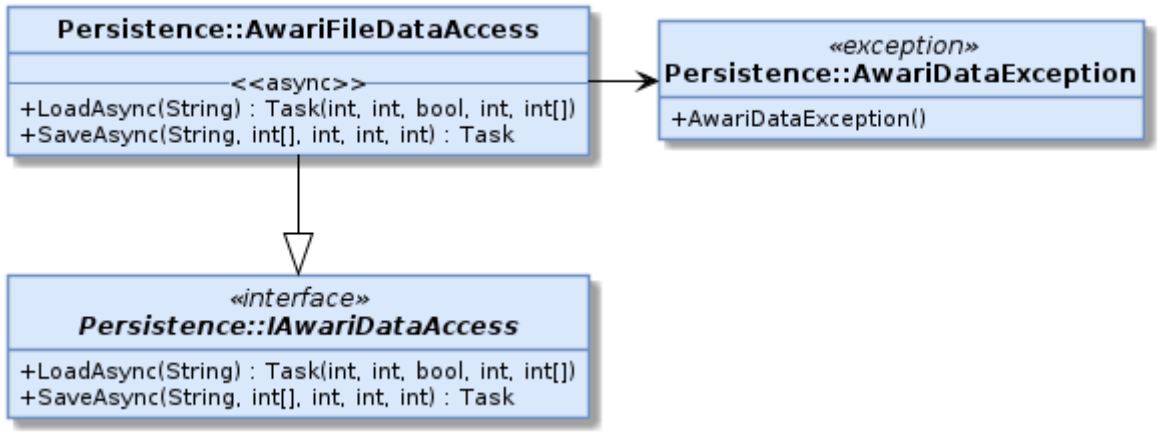
Perzisztencia:

Az adatkezelés feladata egy aktuális játékkal kapcsolatos információk tárolása, valamint a betöltés és mentés biztosítása.

A hosszú távú adattárolás lehetőségeit az **IAwariDataAccess** interfész adja meg, amely lehetőséget ad egy játékállás betöltésére (*LoadASync*), valamint mentésére (*SaveASync*). A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg.

Az interfészt szöveges fájl alapú adatkezelésre az **AwariFileDataAccess** osztály valósítja meg. A fájlkezelés során fellépő hibákat az **AwariDataException** kivétel jelzi.

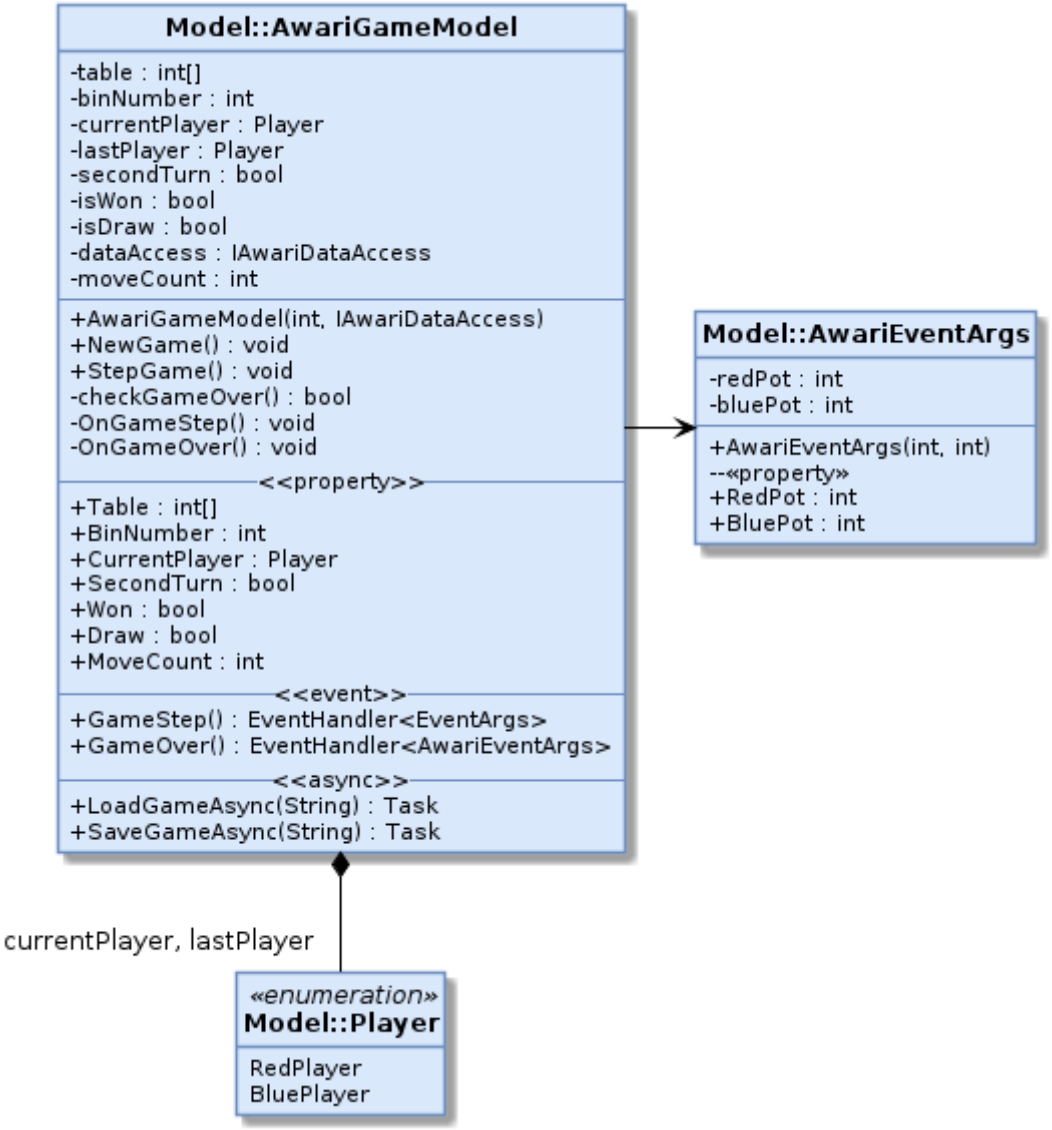
A program az adatokat szöveges fájlként tudja eltárolni, melyek az **awrg** kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.



A fájl egyetlen sort tartalmaz: az első négy szám, mindig ugyanazt az információt tárolja, először a tálkák száma, utána az aktuális játékos, ezután elmentjük, hogy esetleg ez egy második köre volt-e az aktuális játékosnak, és elmentjük az utolsó körben lévő játékost. A többi része a sornak, az a tábla aktuális állása kezdve a piros játékos tálkáival a legvégén a gyűjtőtálka, majd a kék játékos tálkái a legvégén a gyűjtőtálka.

Modell:

A modell lényegi részét a **AwariGameModel** osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, úgymint a *tálkák száma* (**binNumber**), az *aktuális játékos* (**currentPlayer**), a *második kör* (**secondTurn**), a *körök száma* (**moveCount**), az *előző kör játékosa* (**lastPlayer**). A típus lehetőséget ad új játék kezdésére (**NewGame**), valamint lépésre (**StepGame**).



A játékállapot változásáról a **GameStep**, míg a játék végéről a **GameOver** esemény tájékoztat. A **GameOver** esemény argumentuma (**AwariEventArgs**) tárolja a két játékos gyűjtőtálkájában lévő kavicsok számát.

A modell példányosításakor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (**LoadGameAsync**) és mentésre (**SaveGameAsync**).

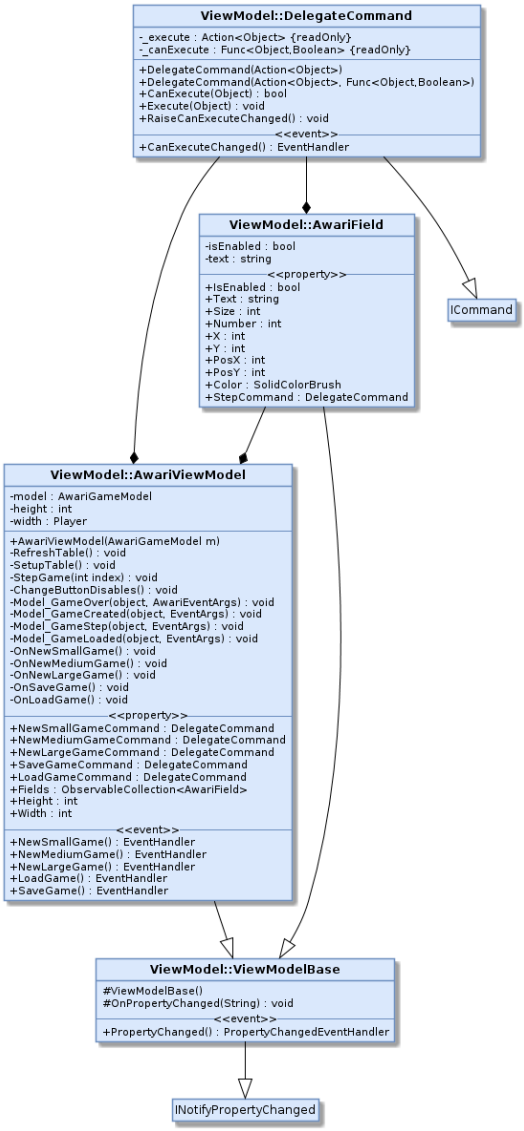
A játékosokat a **Player** felsorolási típus segítségével kezeljük, amely lehet **RedPlayer**, illetve **BluePlayer**.

Nézetmodell:

A nézetmodell megvalósításához felhasználunk egy általános utasítás (**DelegateCommand**), valamint egy ős változásjelző (**ViewModelBase**) osztályt.

A nézetmodell feladait 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 bele a játék futtatásába.

A játékmező számára egy külön mezőt biztosítunk (**AwariField**), amely eltárolja a pozíciót, a feliratot, az engedélyezettséget, a lépés parancsát (**StepCommand**), valamint a gomb színét. A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe.



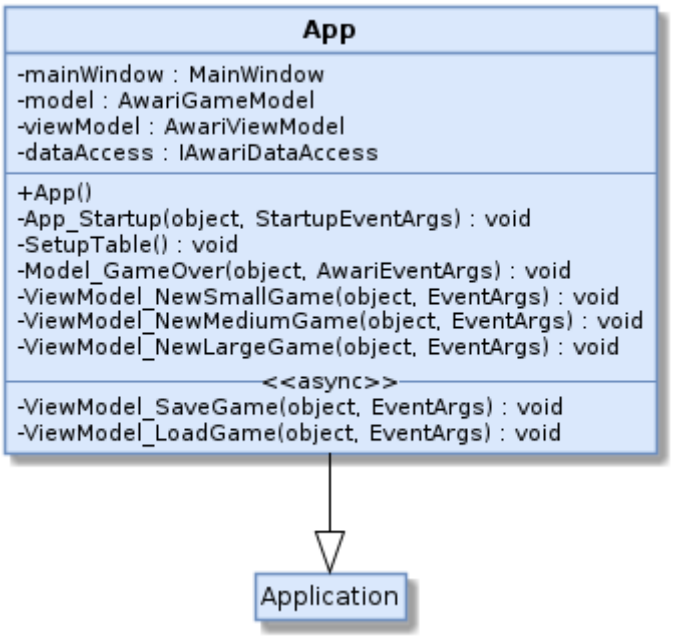
Nézet:

A nézet csak egy képernyőt tartalmaz, a **MainWindow** osztályt. A nézet egy rácsban tárolja a menüt és a játékmezőt. A játékmező egy **ItemsControl** vezérlő, ahol dinamikusan rajzolunk egy vászon (**Canvas**) felületre, amely gombokból áll. Minden adatot adatkötéssel kapcsolunk a felülethez, továbbá azon keresztül szabályozzuk a gombok színét is.

A fájlnév bekérést betöltéskor és mentéskor beépített dialógusablakok segítségével végezzük.

Környezet:

Az App osztály feladat az egyes rétegek példányosítása (App_Startup), összekötése, a nézetmodell és a modell eseményeinek lekezelése, és ezáltal a játék, az adatkezelés, valamint a nézetek szabályozása.



Tesztelés:

A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **AwariGameModelTest** osztályban.

A következő tesztesetek kerültek megvalósításra:

- **AwariNewEasyGameTest,**
- **AwariNewMediumGameTest,**
- **AwariNewHardGameTest**: Új játék indítása a mezők kitöltése, valamint a lépésszám és a tálkák számának ellenőrzése a nehézségi fokozat függvényében.
- **AwariGameStepTest**: Játékbeli lépés hatásainak ellenőrzése a játék megkezdése előtt, valamint után. Esemény kiváltásának ellenőrzés.
- **AwariGameStepAgain**: A játékbeli lépés hatásainak ellenőrzése a játék megkezdése előtt, valamint után. Azt a speciális esetet vizsgáljuk, amikor egy játékos egymás után kétszer jön.
- **AwariGameStepOpposite**: A játékbeli lépés hatásainak ellenőrzése a játék megkezdése előtt, valamint után. Azt a speciális esetet vizsgáljuk, amikor egy játékos utolsó kavicsa egy saját, már üres tálkába esik.
- **AwariGameModelLoadASyncTest**: A játék modell betöltésének tesztelése mockolt perzisztencia réteggel.