

4. Beadandó feladat dokumentáció

Készítette:

Máté Zsolt Sándor

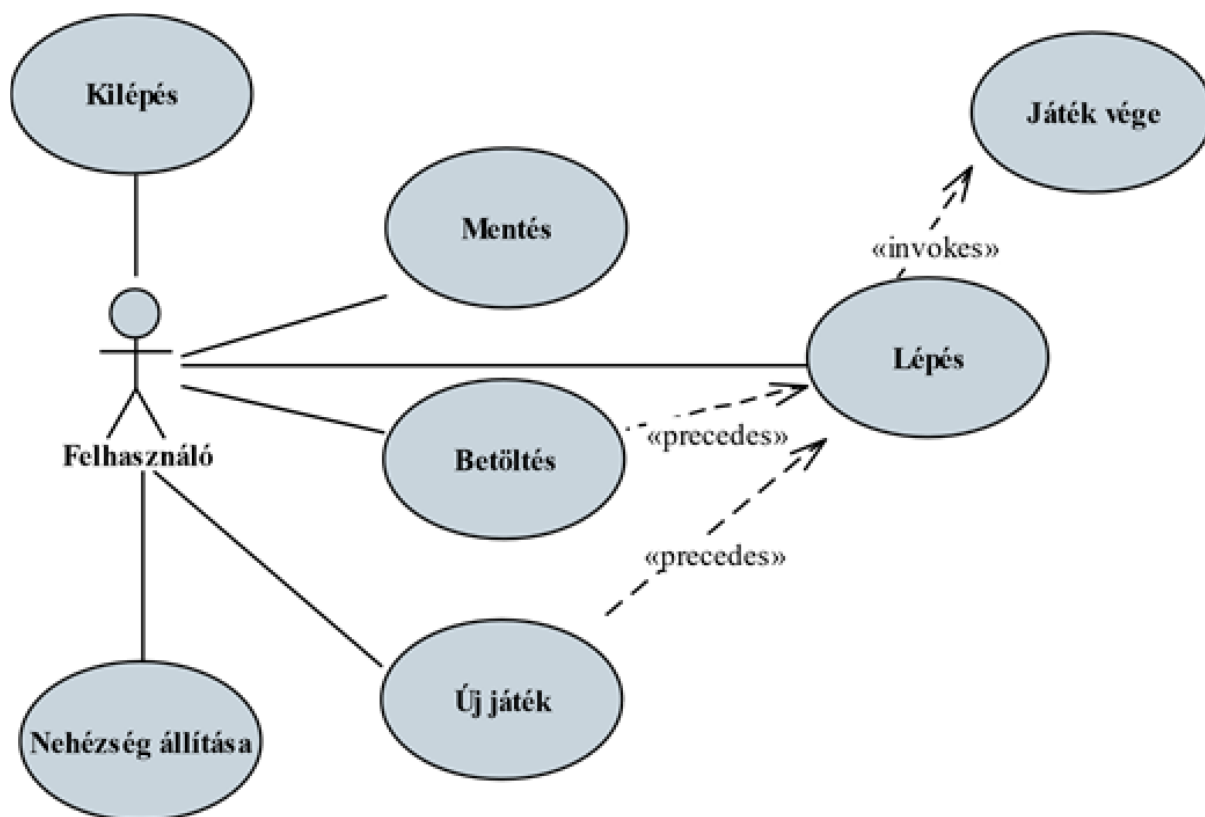
E-mail: s0y2on@inf.elte.hu

Feladat:

Készítsünk programot, amellyel a következő két személyes játékot játszhatjuk. Adott egy $n \times n$ mezőből álló tábla, amelyre a játékosok 2×1 -es méretű téglalapokat helyezhetnek el (vízszintesen, vagy függőlegesen). A játékosok felváltva léphetnek. A játék célja, hogy a téglalapokkal elhatároljuk a tábla egy részét (teljesen körbevéve téglalapokkal), amelyben így minden mező a játékosé lesz (beleértve az ellenfél által korábban elfoglalt mezőket is). A program külön jelölje meg a lehelyezett téglalapokat, illetve az elfoglalt területeket, és játék közben folyamatosan jelenítse meg az elfoglalt terület méretét játékosonként. Eseményvezérelt alkalmazások 2020/2021 őszi félév 8 A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (6×6 , 8×8 , 10×10), valamint játék mentésre és betöltésre. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

Elemzés:

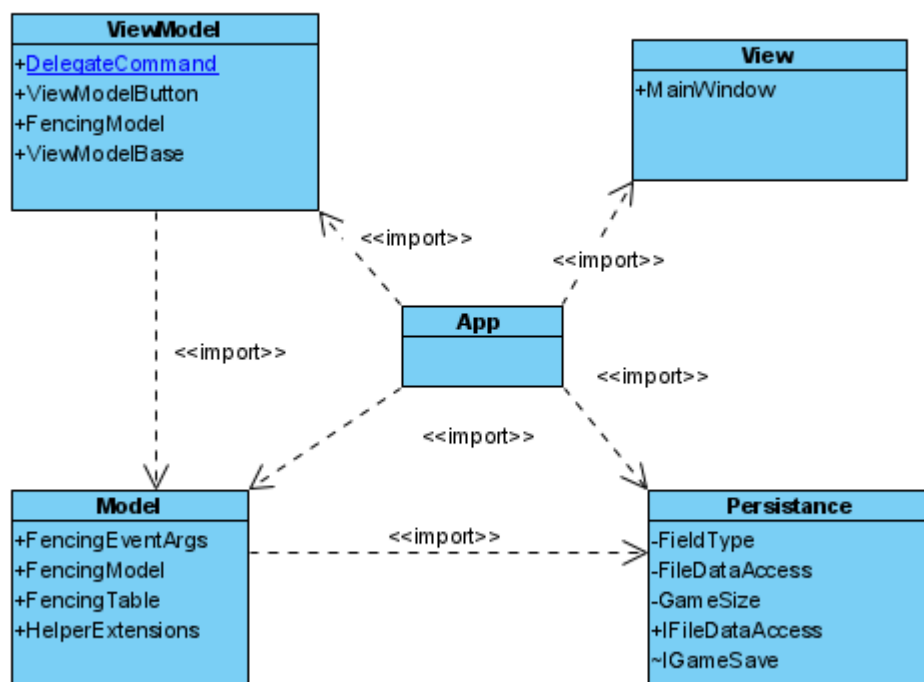
- A játékot három különböző méretű pályán átszhatjuk: 6×6 , 8×8 és 10×10 . A program indításkor a legkisebb pályamérete állít be, és automatikusan új játékot indít.
- A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: Új játék (6×6 , 8×8 , 10×10), Mentés, Betöltés. Az ablak alján megjelenítünk egy státuszsort, amely a pontok számát és aktuális játékost jelzni.
- A játéktáblát egy 6×6 (8×8 , 10×10) nyomógombokból álló rács reprezentálja. A nyomógomb egérekattintás hatására megváltoztatja zölddel jelzett gombokat. A már beszínezett, vagy szabálytalan elhelyezést nem engedjük.
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (jelzi hogy ki nyert és hány ponttal). Szintén dialógusablakokkal végezzük el a mentést, illetve betöltést, a fájln neveket 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** névttereket 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.
- A program csomagszerkezete a 2. ábrán látható.



2. ábra: Az alkalmazás csomagdiagramja

Perzisztencia:

- Az adatkezelés feladata a Fencing táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
- A **FencingTable** osztály egy érvényes Fencing táblát biztosít (azaz mindig ellenőrzi a beállított értékek), ahol minden mezőre ismert az értéke (`_field`) és az aktuális játékost.. A tábla alapértelmezés szerint 6×6-os, de ez a konstruktorban paraméterezhető. A tábla lehetőséget az állapotok lekérdezésére. (`IsFilled`, `GameSize`, `GetFiledType`), valamint szabályos léptetések ellenőrzésére és végrehajtására (`CanPlaceBlock`, `PlaceBlock`), illetve direkt beállítás (`SetFieldType`) elvégzésére.
- A hosszú távú adattárolás lehetőségeit az **IDataAccess** interfész adja meg, amely lehetőséget ad a tábla betöltésére (`LoadAsync`), valamint mentésére (`SaveAsync`). A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg.
- Az interfészt szöveges fájl alapú adatkezelésre a **FileDataAccess** osztály valósítja meg.

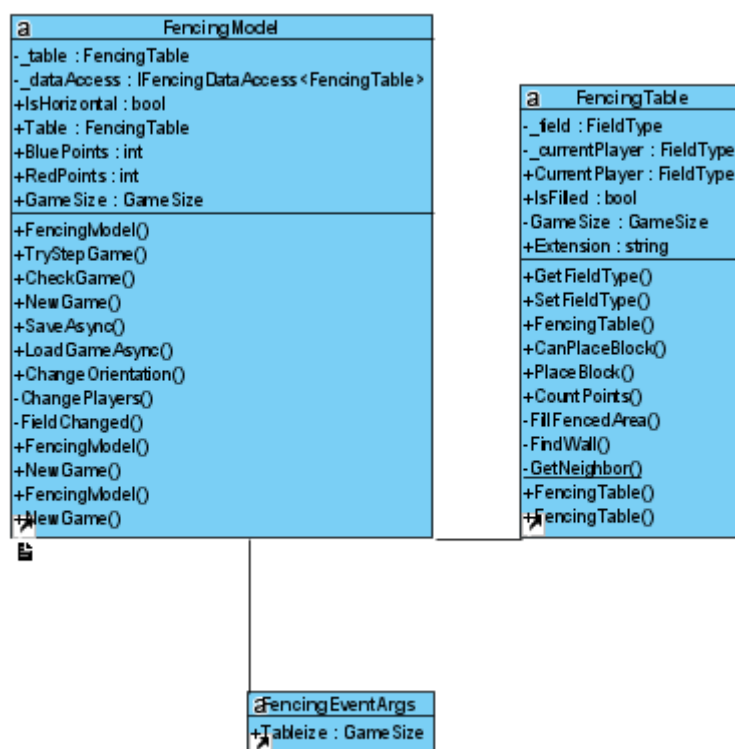
- A program az adatokat szöveges fájlként tudja eltárolni, melyek az **sav** kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
- A fájl egy FencingTable class ír le json állományként.



3. ábra: A Persistence csomag osztálydiagramja

Modell:

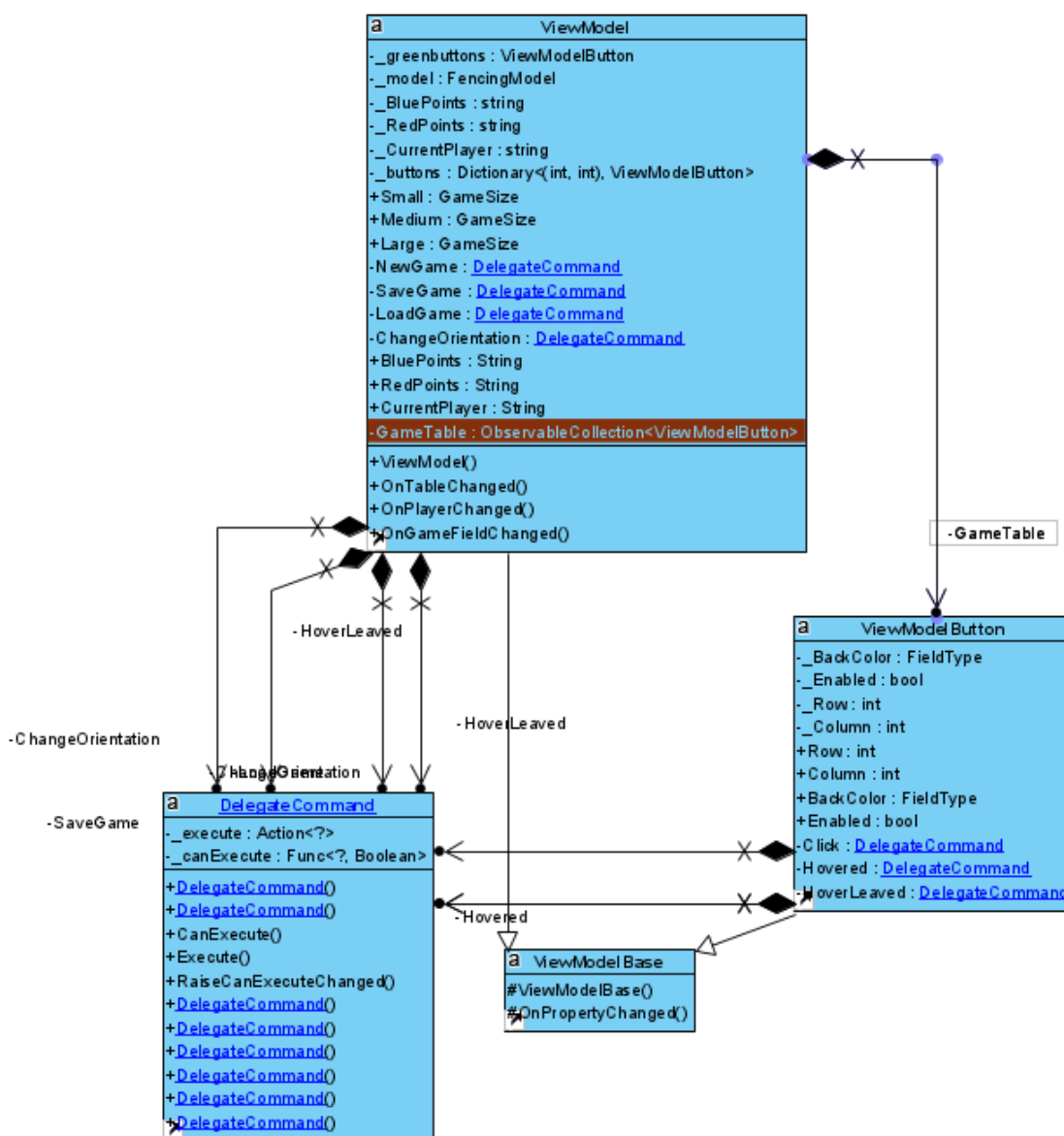
- A modell lényegi részét a **FencingModel** osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, úgymint a lépések (**_BluePoints**, **RedPoints**). A típus lehetőséget ad új játék kezdésére (**NewGame**), valamint lépésre (**TryStepGame**). Új játéknál megadható a kiinduló játéktábla is, különben automatikusan generálódnak kezdő mezők.
- A játékállapot változásáról a **FieldChanged** esemény, míg a játék végéről a **GameOver** esemény tájékoztat. Az események argumentuma objectumokat adunk át.
- A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (**LoadGameAsync**) és mentésre (**SaveGameAsync**)
- A játék mértetét a **Size** felsorolási típuson át kezeljük, és külön osztályban konstansok segítségével tároljuk az egyes méretek paramétereit.
- Hasonlóan **FieldType** a pályán elhelyezhető blokkot típusait tároljuk külön osztályban.



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 **ViewModel** 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 vezérlónek. A nézetmodell tárolja a modell egy hivatkozását (**_model**), de csupán információkat kér le tőle, illetve a játékméretet szabályozza. Direkt nem avatkozik a játék futtatásába.
- A játékmező számára egy külön mezőkezt biztosítunk, amely eltárolja a pozíciót, szöveget, engedélyezettséget, aktuális játékost és a pontokat. A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (**Fields**).



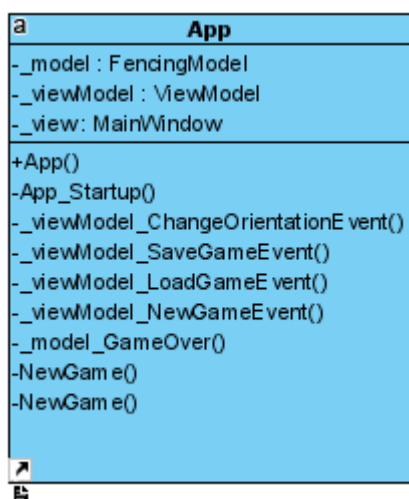
5. ábra: A nézetmodell osztálydiagramja

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 fájlnev bekérését betöltéskor és mentéskor, valamint a figyelmeztető üzenetek megjelenését beépített dialógusablakok segítségével végezzük.

Környezet (5. 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.



5. ábra: A vezérlés osztálydiagramja

Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **FencingModelTest** osztályban.
- Az alábbi tesztesetek kerültek megvalósításra:
- **FencingModelNewGame**: Új játék indítása, a mezők kitöltése
- **FencingPlacingCanPlaceBlock**: Játékbeli lépés ellenőrzése, játék. Több lépés végrehajtása azonos játékmezőn.
- **FencingChangePlayers**: A játékosok váltakozásának ellenőrzése.

- **FencingAreaFenced:** Annak ellenőrzése hogy az elkerített részek megfelelően átszineződnek.
- **FencingLoadTest:** Betöltés megfelelő működése, mezők ellenőrzés mérettel és aktuális játékkal.
- **FencingGameOver:** Játék vége eseménynek ellenőrzése.