3. Beadandó feladat dokumentáció

Készítette:

Szucsányi István

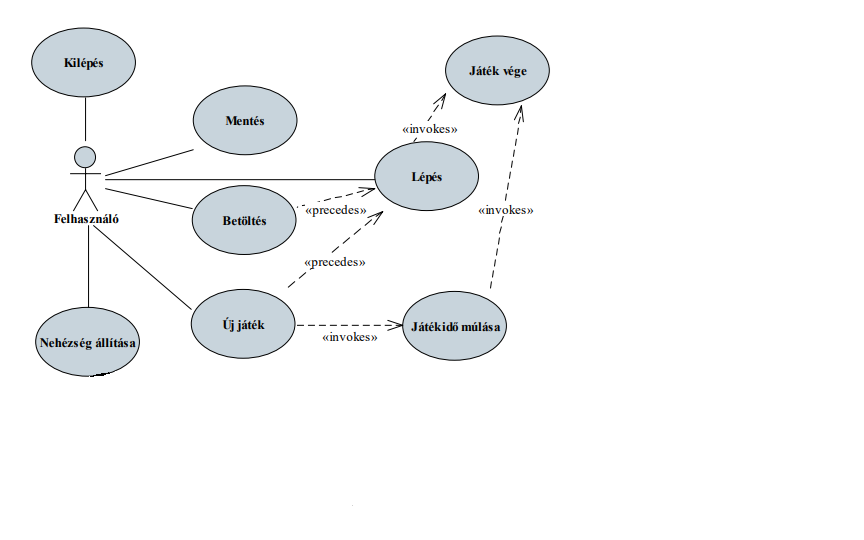
E-mail: fv9591@inf.elte.hu

**Feladat:**

Készítsünk programot, amellyel az alábbi Reversi játékot játszhatjuk. A játékot két játékos játssza 𝑛 × 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).

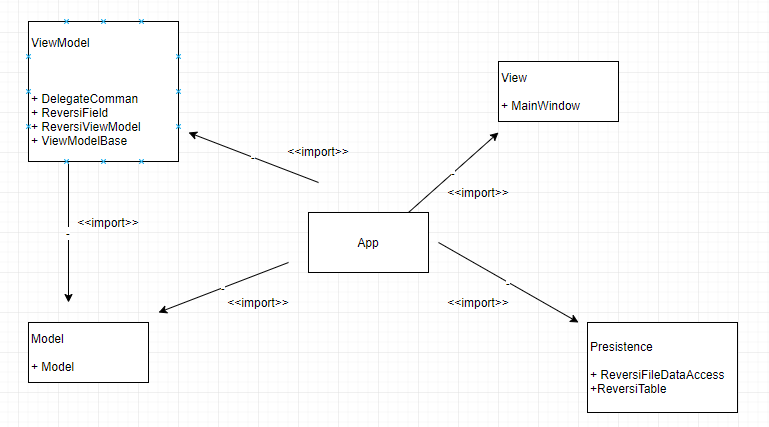
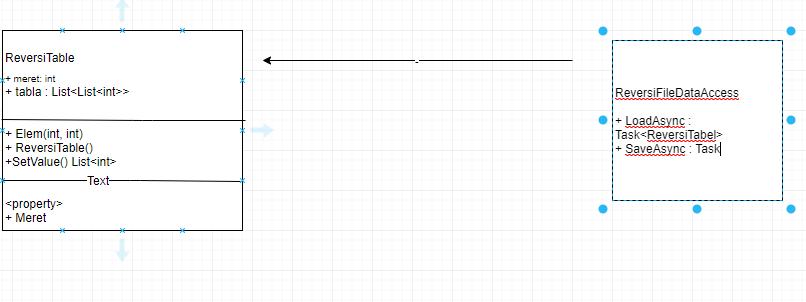
**Elemzés:**

* A játékot majd 3 fajta pályán lehet játszani, az egyik egy 10x10 egy 20x20 és egy 30x30 a program indításakor először a legkisebbet állítottam be
* A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
* Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (Új játék, Játék betöltése, Játék mentése, Kilépés), Beállítások (Könnyű játék, Közepes játék, Nehéz játék). Az ablak alján megjelenítünk egy státuszsort, amely az egyes játékosok gondolkozási idejét jelzi.
* A játéktáblát a fentebb említett méretű táblák reprezentálják amiken nyomógombok vannak elhelyezve és ezek egér kattintásra megváltoztatják a színüket. A táblán dinamikusan minden lépés után blokkolva vannak azok a mezők amikre nem lehet ütés útján beszínezni, és csak azokra a gombokra lehet kattintani amivel ütni tudunk
* A játék végénél egy felugró messageBox jelzi a játék végét, Dialógusablakokkal végezzük el a mentést, illetve betöltést, a fájlneveket a felhasználó adja meg.



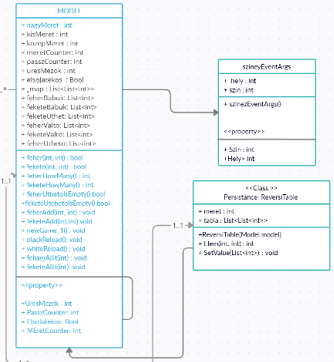
1. ábra: Felhasználói esetek diagramja

**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.
* Perzisztencia:
  + Ennek az osztálynak a feladata maga mentés és a betöltés biztosítása a felhasználó számára.
  + Maga a ReversiTable class tarolja el a pontos koordinátáit a játéktáblának és a tábla 2 tulajdonsággal rendelkezik egy table ami tarolja hogy a mezőkön most éppen melyik játékosnak van bábuja és egy meret int taggal ami letárolja annak a pályának az aktuális méretét ami a menteni vagy betülteni kívánt játékon folyt.
  + A ReversiFileDataAccess pedig azokat a metodusokat tarolja mivel a felhasználó tud majd menteni és betölteni egy bizonyos játékot, ezek a metódusok a program gyors futása érdekében aszinkron módon vannak megoldva.
  + Az interfészt szöveges fájl alapú adatkezelésre a ReversiFileDataAccess osztály valósítja meg.
  + ****A program az adatokat szöveges fájlként tudja eltárolni, melyek az stl 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. A fájl többi része nagyban függ attól, hogy pontosan mekkora tábla merete de általánosítva számokból áll amik reprezentálják annak a pontnak a színét amit éppen beolvasott a program.

**Model**

* A model lényegi részét a Model osztaly valositja meg es az osztalyon beluli alprogramok valositjak meg a fontosabb részek működését.
* A programon belűl a legfontosabb részeket szeretném kiemelni ami például a blackReload és a whiteReload ami dinamikusan egy tömbe helyezi azokat a cellákat amik a soron következő játékos tud lépni ezzel segíti a játékost hogy ne felugró ablakok formájában tájékozódjon arról hogy ő majd arra a pontra amit kijelőlt szeretne lépni hanem a BlockNonWhiteButtons és a BlockNonBlackButtons segítségével dinamikusan lépések után blokkolja azokat a mezőket ahova a soron következő játékos nem tud lépni.
* A játék nehézségét propertykét tároljuk ami lehet kics nagy vagy közepes

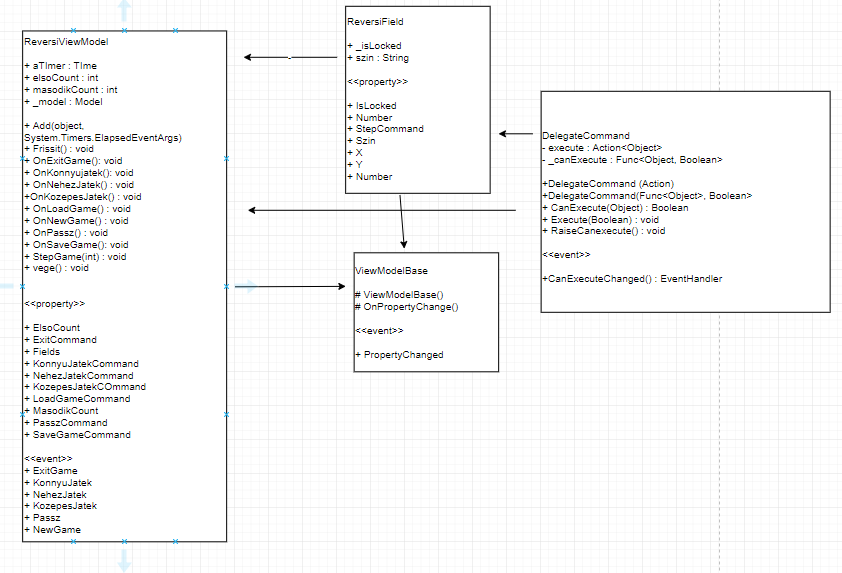
****

**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 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 játéknehézséget szabályozza. Direkt nem avatkozik a játék futtatásába.

• A játékmező számára egy külön mezőt biztosítunk (ReversieField), amely eltárolja a pozíciót, szöveget, engedélyezettséget, valamint a lépés parancsát (StepCommand). A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (Fields).

****

Tesztesetek:

A teszteléseket a UnitTest1 nevű tesztosztályban végeztem el:

* FekteFeherAlapElemek
* FeketeFeherNemAlapElemek
* FeketeUtehtoElemek

Ezekben teszteltem az álltalános dolgokat amikor létrehozom magát az model osztályt, hogy benne szerepelnek e a feher és fekete elemek és vagy nem szerepelnek benn azok az elemek

* AddMethod

Ebben a metodusban pedig szimuláltam egy felhasználó kattintását.