1. Beadandó feladat dokumentáció

Készítette:
Sényi Barnabás
E-mail: nw4l87@inf.elte.hu

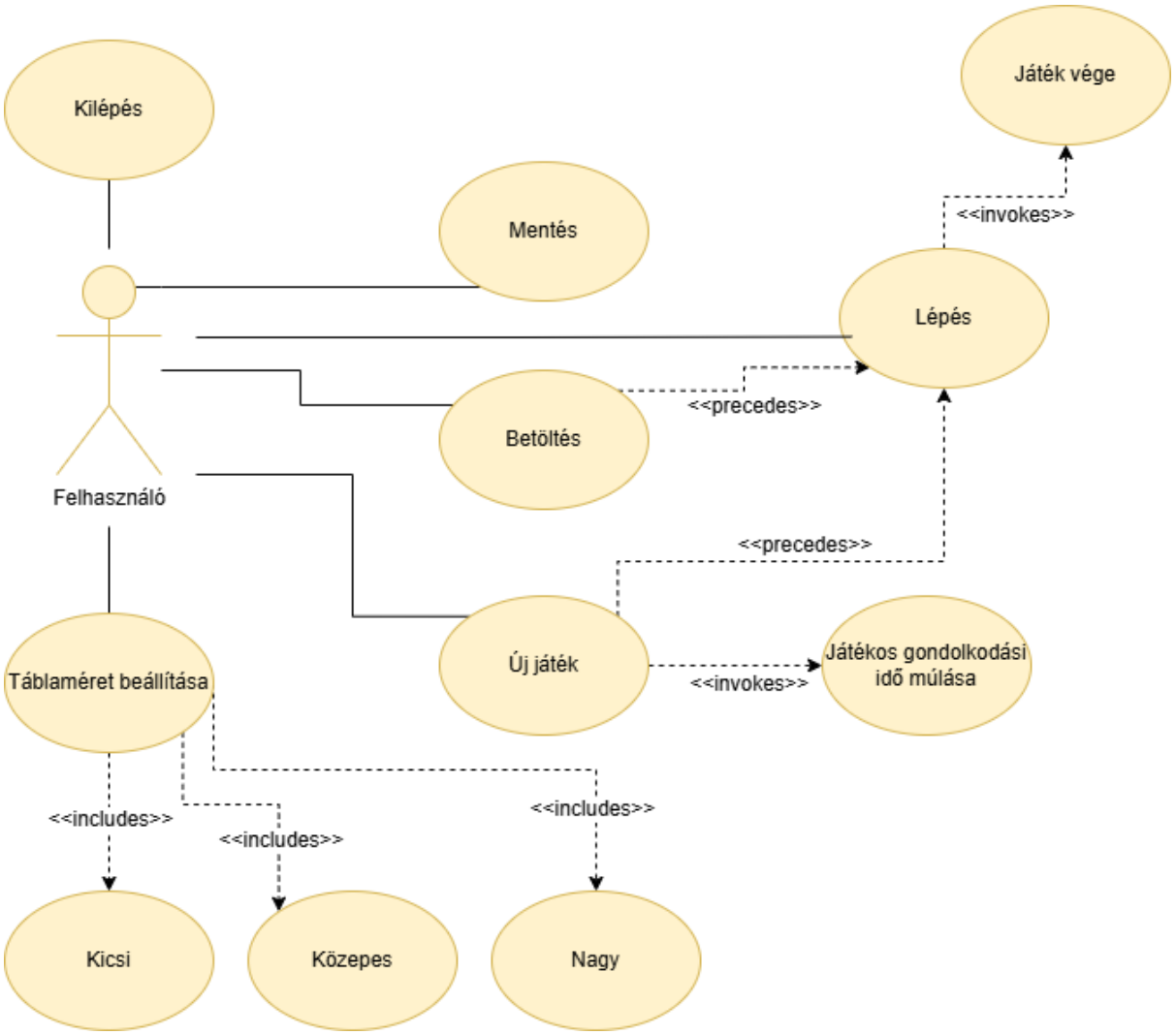
Feladat:

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 (10×10), közepes (20×20), nagy (30×30). A program indításkor a kicsi méretet állítja 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: File (New game, Load, Save, Exit), Settings (Small, Medium, Large). Az ablak alján megjelenítünk egy státuszsort, amely a két játékos idejét, valamint az aktuális játékost jelzi.
- A játéktáblát egy N × N nyomógombokból álló rács reprezentálja. A nyomógomb egérkattintás hatására “odarakja” az aktuális játékost reprezentáló fehér vagy fekete korongot. Csak olyan gombra lehet kattintani, ami érvényes lépést eredményez (ezeken a gombokon egy sárga csillag látható). Azokra a gombokra amin az előre elhelyezett korongok vannak, nem lehet kattintani.
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (betelt a tábla vagy már csak egyféle színű korong van a táblán. Szintén dialógusablakokkal végezzük el a mentést, betöltést, illetve a kilépést. Előbbi kettőnél a felhasználó adja meg a fájlnevet (.stl kiterjesztéssel).

- A felhasználói esetek az 1. ábrán láthatóak.



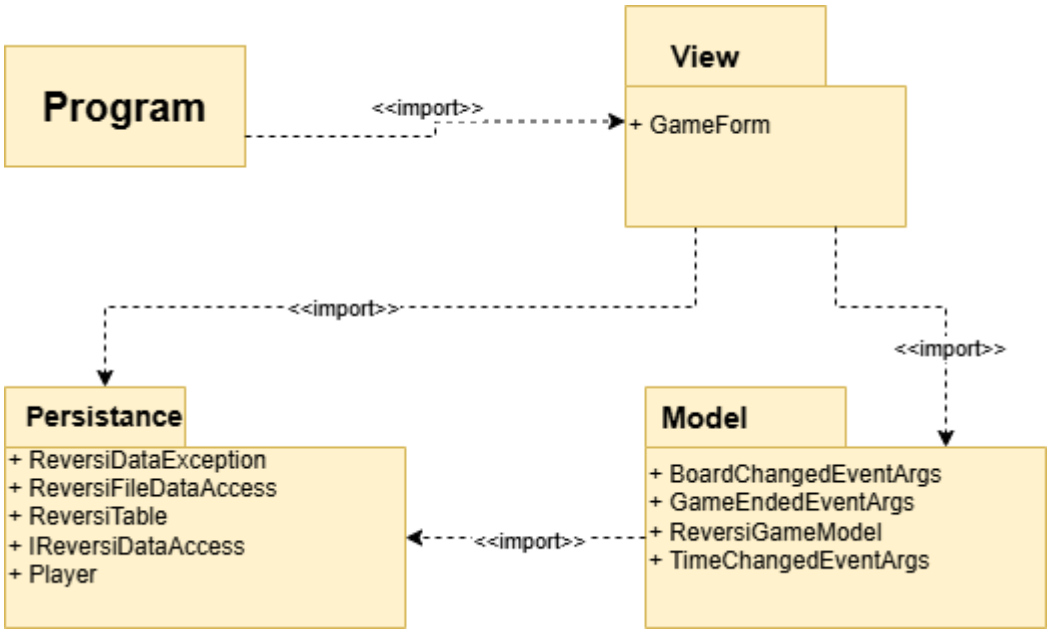
1. ábra: Felhasználói esetek diagramja

Tervezés:

- Programszerkezet:
  - A programot háromrétegű architektúrában valósíjuk meg. A megjelenítés a View, a modell a Model, míg a perzisztencia a Persistence névtérben helyezkedik el. A program csomagszerkezete a 2. ábrán látható.
  - A program szerkezetét két projektre osztjuk implementációs megfontolásból: a Persistence és Model csomagok a program felületfüggetlen projektjében, míg a View csomag a Windows Formstól függő projektjében kap helyet.
- Perzisztencia:
  - Az adatkezelés feladata a Reversi táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
  - A ReversiTable osztály egy érvényes Reversi táblát biztosít (azaz mindig ellenőrzi a beállított értékek), ahol minden mezőre ismert az értéke (_fieldValues), illetve a zároltsága (_fieldLocks), valamint a tábla mérete

(_tableSize). Utóbbit a játék kezdetekor generált, illetve értékekre alkalmazzuk. A tábla alapértelmezés szerint 10 × 10 -es, de ez a konstruktorban paraméterezhető. A tábla lehetőséget az állapotok lekérdezésére (IsFilled, Size, IsLocked, IsEmpty, GetValue), valamint direkt beállítás (SetValue, SetLock) elvégzésére.

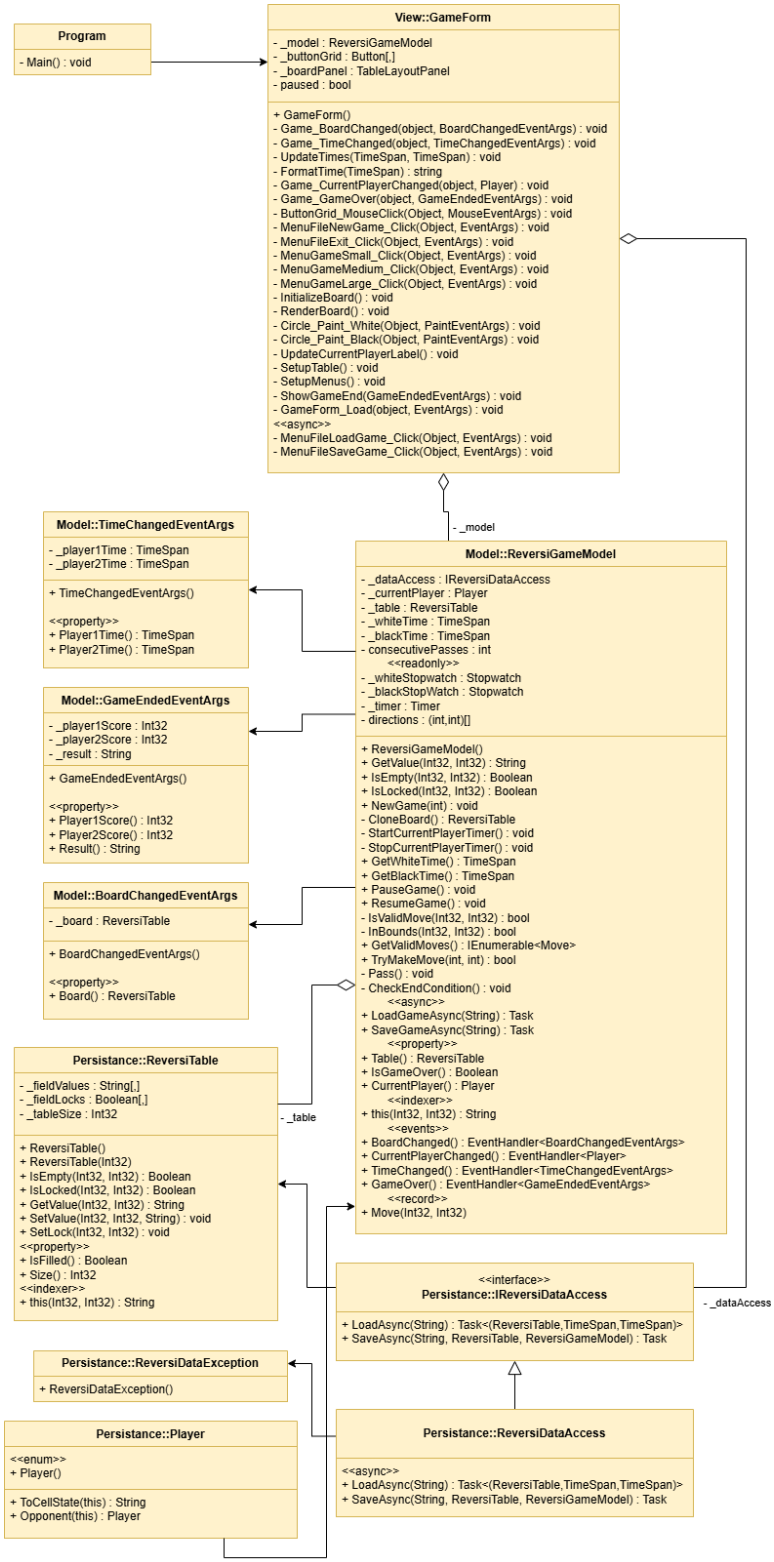
- A hosszú távú adattárolás lehetőségeit az IReversiDataAccess interfész adja meg, amely lehetőséget ad a tábla betöltésére (LoadAsync), valamint mentésére (SaveAsync). A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg.
- Az interfészt szöveges fájl alapú adatkezelésre a ReversiFileDataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a ReversiDataException kivétel jelzi.
- 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, valamint a két játékos gondolkozási idejét. A fájl többi része izomorf leképezése a játéktáblának, azaz összesen 10 sor következik, és minden sor 10 betűt tartalmaz szóközökkel választva. A betűk a következők lehetnek: “w”, “b”, “e”. Ezek jelzik, hogy az adott mezőn melyik játékosnak van korongja (“w”-fehér, “b”-fekete, “e”-üres). Ezután hasonlóan 10 sor következik, melyekben 10 szám áll: 0 vagy 1. Ezek jelzik a mezők zároltságát.



2. ábra: Az alkalmazás csomagdiagramja

- Modell:
  - A modell lényegi részét a ReversiGameModel osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit,

úgymint az idő (_whiteTime, _blackTime), aktuális játékos (_currentPlayer) és időzítők (_whiteStopwatch, _blackStopwatch). A típus lehetőséget ad új játék kezdésére (NewGame), valamint lépésre (TryMakeMove). Új játéknál megadható a kiinduló játéktábla is, különben automatikusan generálódnak kezdő mezők.
    - A játék időbeli kezelését egy időzítő végzi (_timer), amelyet inaktiválunk majd (PauseGame), amennyiben bizonyos menüfunkciók futnak, majd újraindítjuk (ResumeGame).
    - A tábla állapotváltozásáról a BoardChanged esemény tájékoztat. Az esemény argumentuma (BoardChangedEventArgs) tárolja a megváltozott táblát.
    - Az idő változásáról (játékos időzítők cseréje, indítása, megállítása) a TimeChanged esemény tájékoztat. Az esemény argumentuma (TimeChangedEventArgs) tárolja a két játékos idejét.
    - A játék végéről a GameOver esemény tájékoztat. Az esemény argumentuma (GameEndedEventArgs) tárolja a két játékos pontját és az eredményt.
    - A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (LoadGameAsync) és mentésre (SaveGameAsync)
- Nézet:
    - A nézetet a GameForm osztály biztosítja, amely tárolja a modell egy példányát (_model), valamint az adatelérés konkrét példányát (_dataAccess).
    - A játéktáblát egy dinamikusan létrehozott gombmező (_buttonGrid) reprezentálja. A felületen létrehozzuk a megfelelő menüpontokat, illetve státuszsort, valamint dialógusablakokat, és a hozzájuk tartozó eseménykezelőket. A játéktábla generálását (InitializeTable, RenderTable), illetve az értékek beállítását (SetupTable) külön metódusok végzik.
- A program teljes statikus szerkezete a 3. ábrán látható.



3. ábra: Az alkalmazás osztálydiagramja

- Tesztelés:
  - A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a ReversiGameModelTest osztályban.
  - A modell adatelérését az IReversiDataAccess interfészből származtattuk le, így azt a teszt projektben egy MockDataAccess megvalósítással mockolhatjuk.

- Az alábbi tesztesetek kerültek megvalósításra:
  - NewGame_InitialBoard_CorrectSetup:
    - A játéktábla inicializálása helyesen történt-e meg, azaz ott szerepel “b”, ahol fekete korong van, és ott szerepel “w”, ahol fehér, valamint a jelenlegi játékos aktuális értéke helyes-e.
  - Step_PlayerFlipsCorrectly:
    - Létrehozunk egy új játékot és megnézzük, hogy ha végrehajtunk egy lépést, akkor az érvényes lépés-e, az érintett korongoknak megváltozott a színe és értéke, valamint megváltozott-e a jelenlegi játékos.
  - Timer_TracksTimeCorrectly:
    - Létrehozunk egy új játékot, várunk (Sleep()), végrehajtunk egy érvényes lépést, utána ismét várunk, végül megállítjuk a játékot és ellenőrizzük, hogy ténylegesen telt-e mindkét játékos ideje.
  - LoadAsync_ValidFile_LoadsCorrectly:
    - Írunk egy mintafájlt, amit ezután betöltünk, és ellenőrizzük, hogy minden adat helyesen olvasódott-e be.
  - Event_BoardChanged_FiresCorrectly:
    - Létrehozunk egy új játékot, elvégzünk egy lépést és ezután ellenőrizzük, hogy ténylegesen kiváltódott-e a BoardChanged esemény.