**1.Beadandó feladat dokumentáció**

**Készitette:**

**Bence Sooki-Toth**

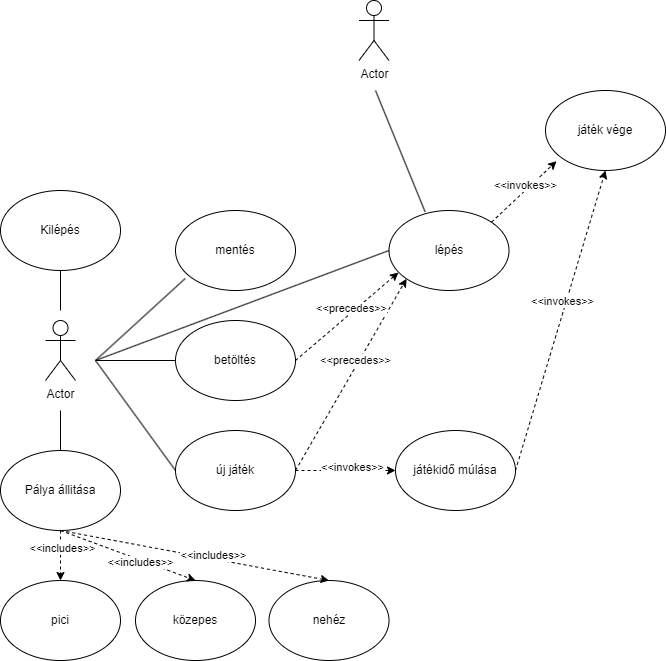
**Email:** [dwgg3k@inf.elte.hu](mailto:dwgg3k@inf.elte.hu)

**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:**

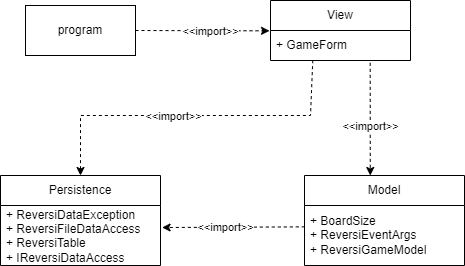
* A játékot három tábla mérettel s játszhatjuk: pici (10x10), közepes(20x20), nagy (30x30). A program indításkor közepes pályát állít be, és automatikusan új játékot indít.
* A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
* Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (Új játék, Játék betöltése, Játék mentése, Kilépés), Beállítások (Pici játék, Közepes játék, Nagy játék). Az ablak alján megjelenítünk egy státuszsort, amely a jelenlegi játékost számát, illetve az eltelt időt jelzi.
* A játéktáblát egy nyomógombokból álló rács reprezentálja. A nyomógomb egérkattintás hatására megváltoztatja a megjelenített mezőt a megfelelő szinre és a közrefogott mezőket a reversi szabályai szerint.
* A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (kiraktuk a táblát, vagy letelt az idő). 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 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 Reversi osztály egy legenerált Reversi táblát biztosít (azaz mindig középre helyezi a 4 kezdő bábut. A tábla alapértelmezés szerint 20x20 -es, de ez a konstruktorban paraméterezhető. A tábla lehetőséget az állapotok lekérdezésére (IsFilled, IsEmpty, GetValue), valamint szabályos léptetésre (Step), illetve direkt beállítás (SetValue) 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, egyik játékos idejét, a teljes eltelt időt és a jelenlegi játékost. A fájl többi része izomorf leképezése a játéktáblának, azaz összesen 10/20/30 sor következik, és minden sor 10/20/30 számot tartalmaz szóközökkel választva. A számok 0, 1 vagy 2 lehetnek, ahol 0 reprezentálja a még üres mezőt



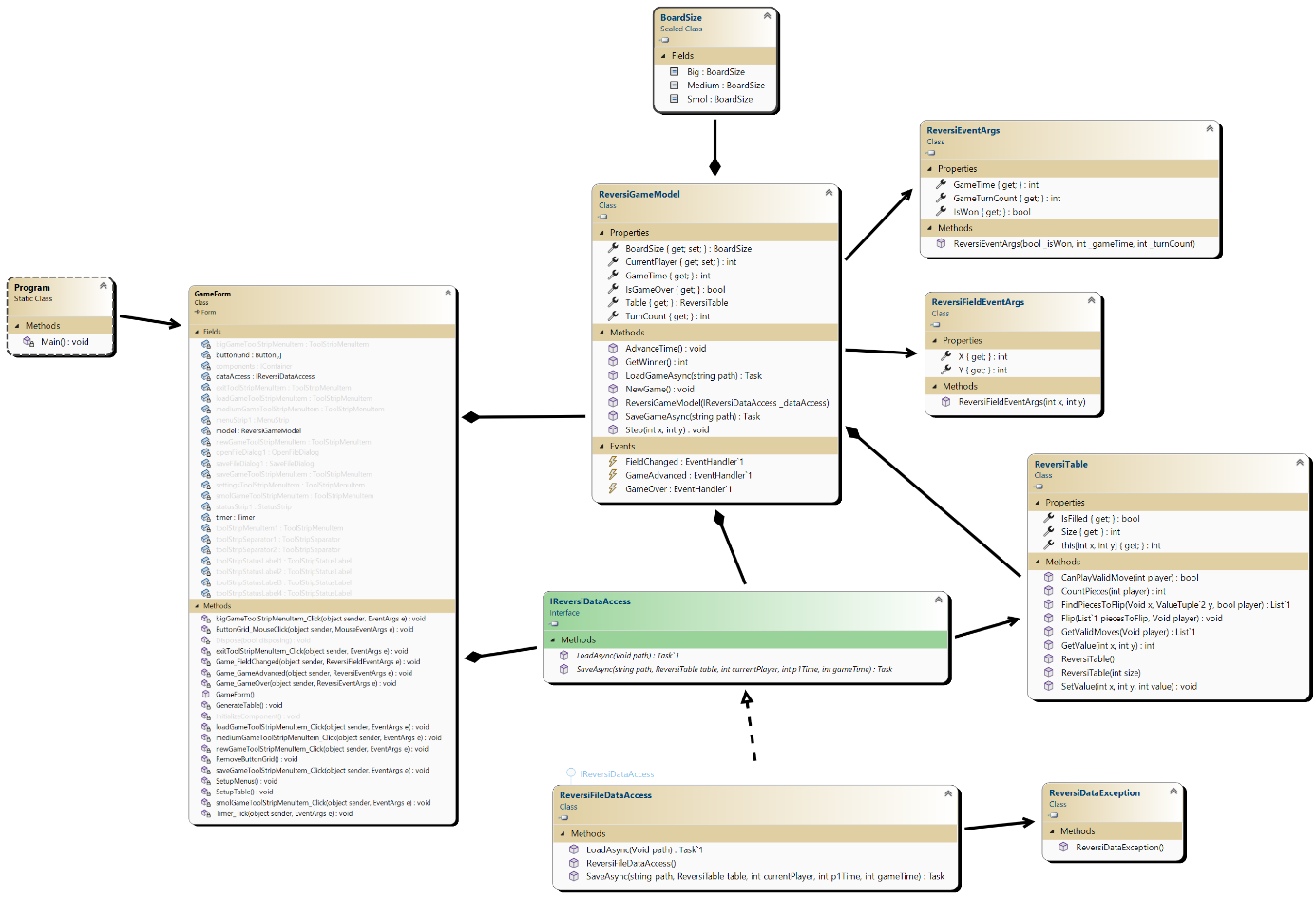
**2.ábra: Az alkalmazás csomagdiagramja**

Modell:

* A modell lényegi részét a ReversiGameModel osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, úgymint az idő (gameTime). A típus lehetőséget ad új játék kezdésére (NewGame), valamint lépésre (Step). Új játéknál megadható a kiinduló játéktábla is, különben automatikusan generálódnak kezdő mezők. Az idő előreléptetését időbeli lépések végzéséve (AdvanceTime) tehetjük meg.
* A mezők állapotváltozásáról a FieldChanged esemény tájékoztat. Az esemény argumentuma (ReversiFieldEventArgs) tárolja a megváltozott mező pozícióját.
* A játékállapot megváltozásáról (eltelt idő) a GameAdvanced esemény, míg a játék végéről a GameOver esemény tájékoztat. Az események argumentuma (ReversiEventArgs) tárolja a győzelem állapotát, eltelt idő, valamint az első játékos eltelt ideje.
* A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (LoadGameAsync) és mentésre (SaveGameAsync)
* A pálya méretét a BoardSize felsorolási típuson át kezeljük, és a ReversiTable osztályban konstansok segítségével tároljuk az egyes nehézségek paramétereit.

**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 (GenerateTable), illetve az értékek beállítását (SetupTable) külön metódusok végzik.
* A játék időbeli kezelését egy időzítő végzi (timer), amelyet mindig aktiválunk játék során, illetve inaktiválunk, amennyiben bizonyos menüfunkciók futnak.
* 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.
* Az alábbi tesztesetek kerültek megvalósításra:
  + Initialize() metódus: Ebben a metódusban felkészítjük a tesztkörnyezetet. Létrehozunk egy ReversiTable objektumot, egy hamisított adathozzáférés objektumot (mock), és inicializáljuk a ReversiGameModel objektumot. A tesztkörnyezetet a későbbi tesztekhez állítjuk be.
  + TestNewGame(): Ez a teszt ellenőrzi, hogy az új játék létrehozásakor a játékmodell helyesen inicializálódik. Ellenőrizzük a játékidőt, a fordulók számát, a táblaméretet, és azt is, hogy az első játékos legyen az aktív.
  + TestSmolGame(), TestMediumGame(), TestBigGame(): Ezek a tesztek az egyes táblaméretek beállítását ellenőrzik. Meggyőződünk róla, hogy a tábla mérete a várt értékkel egyezik meg.
  + TestAdvanceTime(): Ez a teszt ellenőrzi, hogy a játékmodell helyesen növeli a játékidőt.
  + TestGetWinner(): Ez a teszt ellenőrzi, hogy a játékmodell helyesen határozza meg a győztest a játék végén.
  + TestStep(): Ez a teszt ellenőrzi, hogy a játék lépése helyesen működik, és az aktív játékos helyesen váltódik.
  + ReversiGameModelAdvanceTimeTest(): Ez a teszt további játék előrehaladását teszteli.
  + TestPass(): Ebben a tesztben azt ellenőrizzük, hogy a játékos váltás a megfelelő módon működik egy adott játékállapotban.
  + TestLoadGameAsync(): Ez a teszt ellenőrzi, hogy a játék betöltése aszinkron módon helyesen működik. Ellenőrizzük a tábla állapotát, a fordulók számát, és megerősítjük, hogy a megfelelő adathozzáférési művelet történt.