**1. Beadandó feladat dokumentációja**

# **Készítette:**

Név: Peskó Márton

Neptun azonosító: YRQHGX

E-mail cím: [peskomarton@hotmail.com](mailto:peskomarton@hotmail.com)

## **Feladat:**

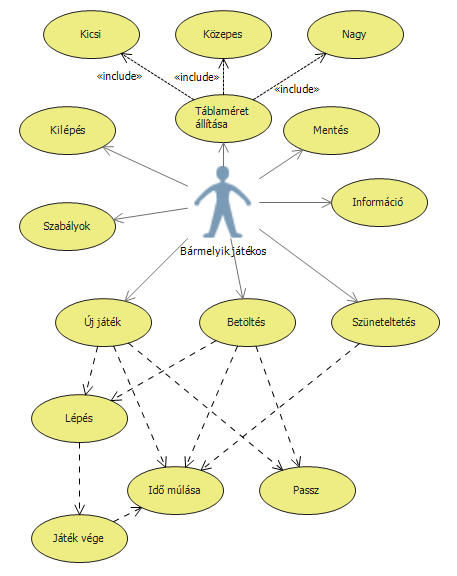
17. Reversi: Készítsünk programot, amellyel az alábbi Reversi játékot játszhatjuk. A játékot két játékos játssza n × n-es négyzetrácsos táblán fekete és fehér korongokkal. Kezdéskor a tábla közepén X alakban két-két korong van elhelyezve mindkét színből. A játékosok felváltva tesznek le újabb korongokat. 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).

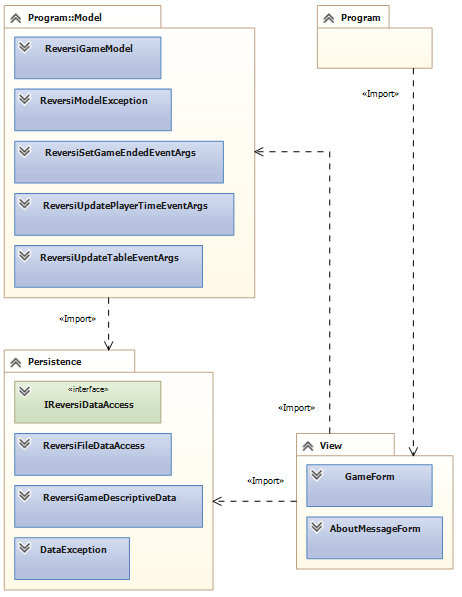
## **Elemzés:**

* A játékot három – konstansból dinamikusan – létrehozott mérettel játszhatjuk: kicsi (10 × 10-es tábla), közepes (20 × 20-as tábla) és nagy (30 × 30-as tábla). A program indításkor – konstansból dinamikusan – a kicsi méretet választja. Az új játék méreteket a *Game*/*Size* menüben lehet állítani.
* A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
* Az ablakban elhelyezünk egy menüt a következő menüpontokkal: *File* (*New*, *Load*, *Save*, *Exit*), Game (*Size*: (*Small*, *Medium*, *Large*)), *Help* (*Reles*, *About* *Reversi*). Az ablak alján megjelenítünk egy státuszsort, amely a játékosok aktuális pontjait mutatja.
* A játéktáblát egy n × n nyomógombokból álló rács reprezentálja. A nyomógomb egérkattintás hatására végrehajt egy lerakást. A táblán csak azon gombok aktívak, amelyekkel szabályos lerakás végezhető. A fekete játékos pontjai feketék, a fehéré fehérek. Az aktív gombok közepén egy kör van a soron következő játékos színével.
* A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak ez akkor történik, ha senki se tud már rakni. Kiírja a gyűjtött pontszámokat és hogy azok alapján ki nyert. A mentéshez és a betöltéshez is dialógus ablakokat használunk, ahogy a szabályok és a program adatok megjelenítéséhez is.
* A játékidő szüneteltetéséhez és a passzoláshoz egy-egy gombot használunk. A rács fölötti helyen. Itt jelenítjük meg még a játékosok eltelt idejét.
* A felhasználói esetek az 1. ábrán láthatóak.

**Tervezés:**

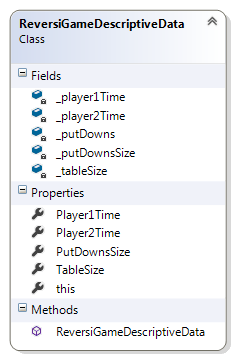
1. ábra: Felhasználói esetek diagramja

* Programszerkezet:
  + A programot háromrétegű architektúrában valósítjuk meg. A *megjelenítés* a **View**, a *modell* a **Model**, míg a *perzisztencia* a **Perzisztancia** névtérben helyezkedik el. A program csomagszerkezete a 2. ábrán látható.



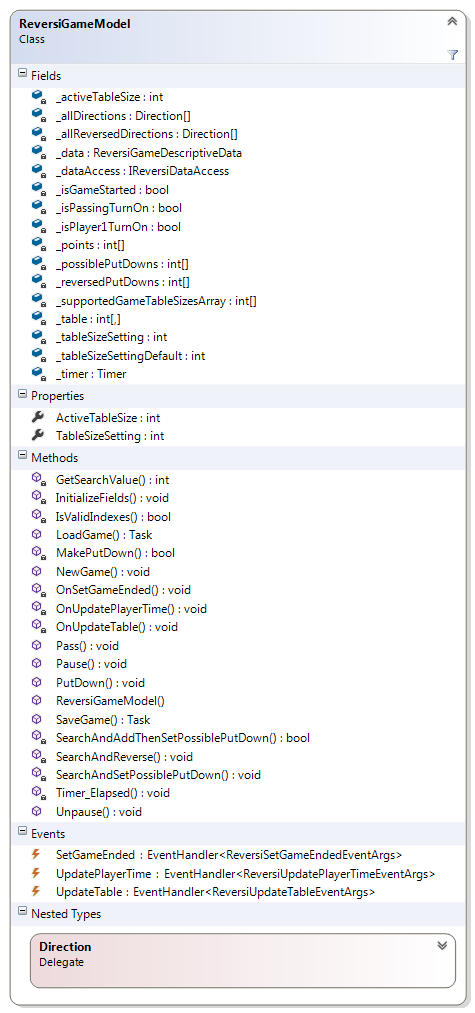
2. ábra: Az alkalmazás csomagdiagramja

* Perzisztencia:
  + A játékot leíró lépések, játékosok időinek és a táblaméret adatok kezelése, mentése és betöltése a feladata.
  + A **ReversiGameDescriptiveData** osztály mindig egy érvényes játék állapotot ír le, azaz beolvasásnál mindig ellenőrzi az adatokat. A lépéseket a \_**putDowns** tömbben tároljuk, mint *x* és *y* koordináták sorozatát (a passzt *-1*, *-1*-el jelöljük). Míg a tömbben lévő lényeges adatok számát a \_**putDownsSize** változóban. A játék elején, ha nem töltöttük be azt, akkor ennek értéke *0*. Ebbe a tömbbe játék közben a *modell* rak be ellenőrzötten elemeket és frissíti az többi változót (\_**tableSize**, \_**player1Time**, \_**player2Time**). Az állapotok lekérdezését és frissítést a *modell* szabvány *Property*-k segítségével végezi.
  + A hosszú távú adattárolás lehetőségeit az **IReversiDataAccess** interfész adja meg, amely lehetőséget ad a tábla betöltésére (**Load**), valamint mentésére (**Save**). 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 a *reversi* kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális – nem győztes – állást.
  + A fájl első sora sorban egy helyközzel elválasztva tartalmazza a tábla méretet, az első játékos idejét, a második játékos idejét, majd végül az elmentett lépések koordinátáinak a számát. A második sorban pedig annyi koordináta, amennyit megadtunk az első sorban.
  + Őt a nézet hozza létre és küldi el a modellnek. Paraméterben kaphat támogatott táblaméret tömböt. Ha nem kap, akkor a 10 × 10-es mérettel fog csak dolgozni.
  + A beolvasott adatok helyességéről a *modell* fog meggyőződni, úgy, hogy végigjátssza a játékot a lerakások szerint. Ha hibát talál, akkor **ReversiDataException**-t dob a *modell***.**
  + A **ReversiGameDescriptiveData** osztály diagram a 3. ábrán látható.

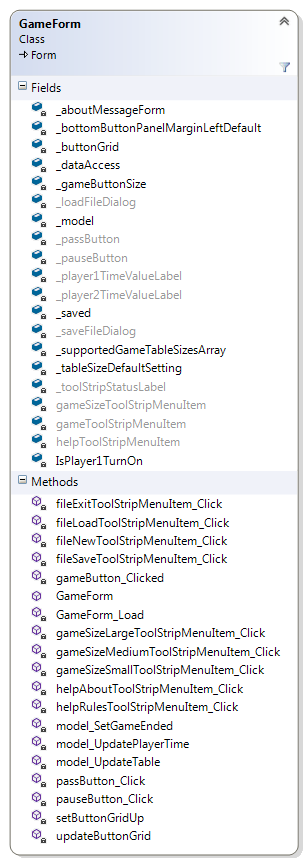


3. ábra: A **ReversiGameDescriptiveData** osztály diagramja

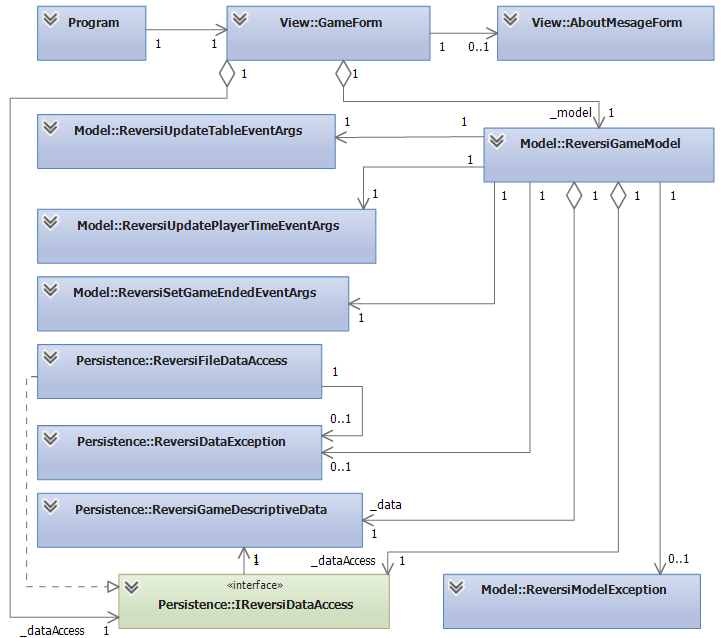
* Modell:
  + A *modell* lényegi részét a **ReversiGameModel** osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit. Az időt (\_**timer**), az aktív játékost (\_**isPlayer1TurnOn**), a passzolást (\_**isPassingTurnOn**), a játékosok pontjait (\_**points**), a megfordított lerakott pontokat (\_**reversedPutDowns**) és a lehetséges lerakatok helyeit (\_**possiblePutDowns**) és pár segéd változókat a helyes működéshez.
  + A tábla frissítéséről az **UpdateTable** esemény az idő múlásáról az **UpdatePlayerTime** esemény, míg a játék végéről a **SetGameEnded** esemény tájékoztatja a *nézetet*. Mindegyik eseménynek saját argumentuma van.
  + A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (**LoadGame**) és mentésre (**SaveGame**). Még kaphat alapértelmezett tábla méretet is.
  + A tábla méretét a \_**tableSizeSetting** és annak *property*-jén keresztül állíthatja a nézet. Az aktuális méretet pedig az \_**activeTableSize**-nak a *property*-jén keresztül kaphatja meg.
  + A nézettől kapott koordináták alapján elvégzi a lerakást (**PutDown**) és a vele járó minden műveletet. Először is megnézi, hogy rakhat-e oda, ha nem akkor csak figyelmen kívül hagyja. Ha rakhat, akkor elkezdi a műveletet (**MakePutDown**). Először elvégzi a megfordításokat (**SearchAndReverse**). Majd frissíti a lehetséges lerakatok tömbjét (**SearchAndSetPossiblePutDown**) és hozzáveszi a lehetséges frissített újakat (**SearchAndAddThenSetPossiblePutDown**). Ezekhez használ segédműveleteket.
  + Az idő múlásához a *.Net* által biztosított (T**imer\_Elapsed**) segédfüggvényt haszáljuk.
  + A **ReversiGameModel** osztály diagramja a 4. ábrán látható.
* Nézet:
  + A nézet a **GameForm** osztály biztosítja, amely tárolja a *modell* egy példányát (\_**model**), valamint az adatelérés konkrét példányát (\_**dataAccess**).
  + A játéktáblát egy dinamikusan létrehozott gomb két dimenziós tömb (\_**buttonGrid**) tárolja és reprezentálja. A felületen létrehozzuk a megfelelő menüpontokat, illetve státuszsort, valamint dialógusablakokat, és a hozzájuk tartozó eseménykezelőket. A játéktábla generálását (**setButtonGridUp**), illetve az értékek beállítását (**updateButtonGrid**) külön metódusok végzik.
  + A játékot bizonyos műveletek közben szüneteltetjük, majd újra elindítjuk.
  + A passzoláshoz (**passButton**), meg az idő szüneteltetéséhez (\_**pauseButton**) van két gombunk.
  + A **GameForm** osztály diagramja az 5. ábrán látható.
  + Az alkalmazás osztálydiagramja a 6. ábrán



4. ábra: A **ReversiGameModel** osztály diagramja

****

**5**. ábra: A **GameForm** osztály diagramja

**Tesztelés:**

6. ábra: A program osztálydiagramja

* A *modell* és *perzisztencia* funkcionalitását teszteljük a **ReversiTest** projektben unit tesztet használva.
* Az alábbi tesztesetek kerülnek megvalósításra:
  + **ReversiGameModelBeforeNewGameSaveTest, ReversiGameModelBeforeNewGameTest, ReversiGameModelNewGameInitializeOddTest, ReversiGameModelNewGameInitializeTooSmallTest**: Elronthatja-e a *nézet* a *modell* állapotát nem várt módon? Nem!
  + **ReversiGameModelNewGameLoadEmptyFileTest, ReversiGameModelNewGameLoadLessPutDownThenPutDownSizeTest, ReversiGameModelNewGameLoadNoPlayer2TimePutDownSizeTest, ReversiGameModelNewGameLoadNoPlayersTimePutDownSizeTest, ReversiGameModelNewGameLoadNoPutDownSizeTest, ReversiGameModelNewGameLoadTestOk0Step, ReversiGameModelNewGameLoadWrongePlayer1TimeTest, ReversiGameModelNewGameLoadWrongePlayer2TimeTest, ReversiGameModelNewGameLoadWrongePlayersTimeTest, ReversiGameModelNewGameLoadWrongeStep0Instead3Or4Test, ReversiGameModelNewGameLoadWrongeStep0Instead6Or4Test, ReversiGameModelNewGameLoadWrongeStep1Instead3Or4Test, ReversiGameModelNewGameLoadWrongeStep1Instead6Or4Test, ReversiGameModelNewGameLoadWrongeStep1InsteadPassTest, ReversiGameModelNewGameLoadWrongeStep3Instead6Or4Test, ReversiGameModelNewGameLoadWrongeStep3InsteadPassTest, ReversiGameModelNewGameLoadWrongeStep5Instead3Or4Test, ReversiGameModelNewGameLoadWrongeStep5Instead6Or4Test, ReversiGameModelNewGameLoadWrongeStep6Instead3Or4Test, ReversiGameModelNewGameLoadWrongeStep6InsteadPassTest, ReversiGameModelNewGameLoadWrongeStepMinus1Instead3Or4Test, ReversiGameModelNewGameLoadWrongeStepMinus1Instead6Or4Test, ReversiGameModelNewGameLoadWrongeStepMinus1InsteadPassTest, ReversiGameModelNewGameLoadWrongeStepPassInstead3Or4Test, ReversiGameModelNewGameLoadWrongeStepPassInstead6Or4Test, ReversiGameModelNewGameLoadWrongeTableSizeTest, ReversiGameModelNewGameLoadWrongeTooBigPutDownSizeTest, ReversiGameModelNewGameLoadWrongOddPutDownSizeTest, ReversiGameModelNewGameSizeTest**: Csak jó játékokat töltünk be? Igen! Ezzel a modell **MakePutDown** függvényét is teszteljük.
  + **ReversiGameModelAllPossibleSenario**: Csak kísérletezés, hogy minden lehetséges játék menetet megnézzünk. Már a legkisebb táblán (4 × 4) is 60060 menet van, persze ennél kevesebb játék állapot. Azt nem sikerült megállapítanom, hogy mennyi és a TODO-knél sem segített.
* TODO-k:
  + **ReversiGameModelAllPossibleSenario** tesztesetnél esetleg gráfos megoldásnál lehetne próbálkozni nagyobb mérettel, de az állapotok összehasonlítását le kell rövidíteni valahogy.
  + Tömbök méretének pontosabb behatárolása.
  + Gombok létrehozásának fokuszálása, eddigitől eltérő táblaméretnél.
  + A táblaméret megadásánál a menük fokuszáltabb frissítése.
  + Kör alakú gombok a mostaniak helyett.