Dokumentáció

Feladat:

15. Aknakereső

Készítsünk programot, amellyel az aknakereső játék két személyes változatát játszhatjuk.
Adott egy $n \times n$ mezőből álló tábla, amelyen rejtett aknákat helyezünk el. A többi mező szintén elrejtve tárolják, hogy a velük szomszédos 8 mezőn hány akna helyezkedik el.
A játékosok felváltva léphetnek. Egy mező felfedjük annak tartalmát. Ha az akna, a játékos veszített. Amennyiben a mező nullát rejt, akkor a vele szomszédos mezők is automatikusan felfedésre kerülnek (és ha a szomszédos is nulla, akkor annak a szomszédai is, és így tovább). A játék addig tart, amíg valamelyik játékos aknára nem lép, vagy fel nem fedték az összes nem akna mezőt (ekkor döntetlen lesz a játék).
A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (6×6, 10×10, 16×16), valamint játék mentésére és betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött (ha nem döntetlen).

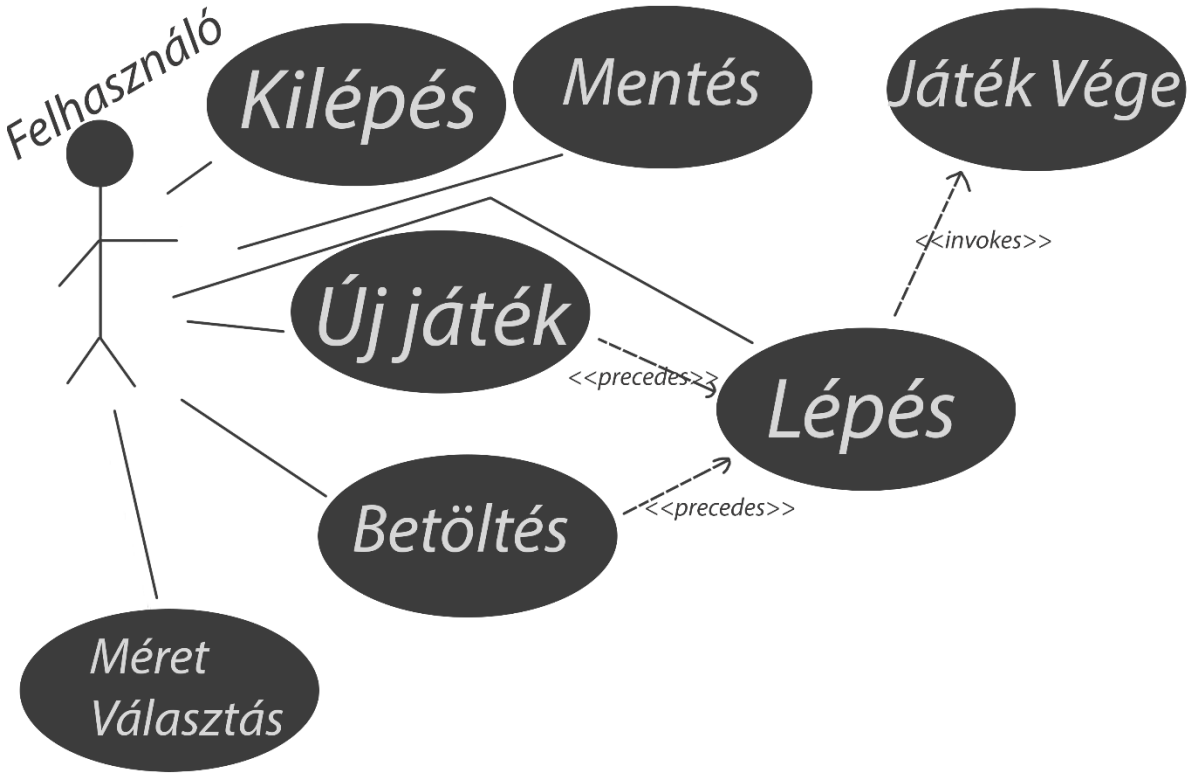
Elemzés:

- Adott pályaméretek vannak, ellenben nincs megadva aknaszám, szóval lesz 6-nál 6; 10-nél 12; 16-nál 24.
- A feladatot egy egy-ablakos Windows Form grafikus felülettel valósítjuk meg.
- Egy leugró menüvel választhatunk "*Új Játék / New Game*","*Játék Mentése / Save Game*","*Játék Betöltése / Load Game*" és "*Kilépés / Quit*" opciók között.
- A következő játékost felül egy szöveg valamint egy váltakozó színű panel mutatja.
- Az aknamező egy nxn-es "Button" objektumokból álló táblázattal reprezentáljuk.
- Ha vége a játéknak akkor egy eredményt mutató felugró ablak után megjelennek bombák és új játék választható. Véget ért játékot nem lehet menteni.
- A játék véget ér, ha az egyik játékos aknára lép, vagy ha minden nem akna mező fel lett fedve.

Felhasználió Esetek:

	Felhasználói Eset	Leírás	
1	Alkalmazás indítása	GIVEN:	Az alkalmazás telepítve van
		WHEN:	Alkalmazás indítása
		THEN:	Felugró ablak megjelenik

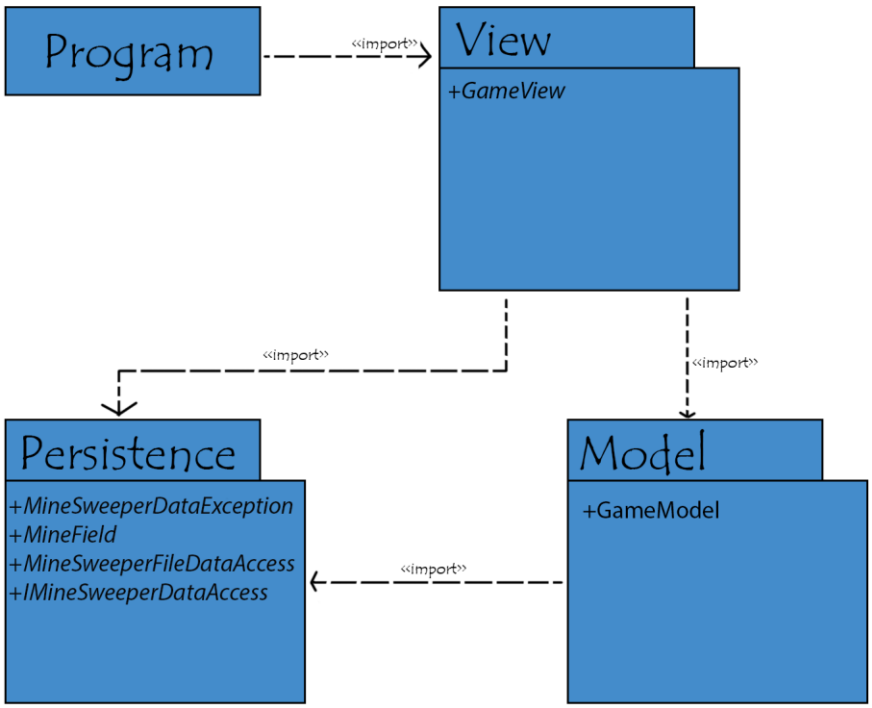
2	Pályaméret választás	GIVEN:	Az alkalmazás elindult
		WHEN:	Kiválasztunk egy opciót
		THEN:	Betöltődik a pálya
3	Kilépés	GIVEN:	Játék felülete
		WHEN:	Ablak záró ikonja vagy “Quit” menüpont
		THEN:	Az alkalmazás bezárul
4	Új játék	GIVEN:	A játék felülete
		WHEN:	“New Game” menüpont megnyomása
		THEN:	Új aknamező töltődik be alapállapotban
5	Mezőre kattintás	GIVEN:	Folyamatban levő játék
		WHEN:	Kattintás az egyik fel nem fedett cellára
		THEN:	A mező felfedi az értékét. Vagy véget ér a játék
6	Játék Vége	GIVEN:	Futó játékfelület
		WHEN:	Semlegesítve vannak az aknák, vagy valaki rálépett egyre.
		THEN:	A játék befejeződik. A menüponttal új játék kezdhető
7	Mentés	GIVEN:	Futó játékfelület
		WHEN:	“Save Game…” menüpont megnyomása
		THEN:	A játékállás elmentődik egy megadott nevű file-ba
8	Betöltés	GIVEN:	Futó játékfelület
		WHEN:	“Load Game…” menüpont megnyomásával egy file kiválasztása
		THEN:	Ha megfelelő a file akkor egy mentett játékállás betöltése



Tervezés:

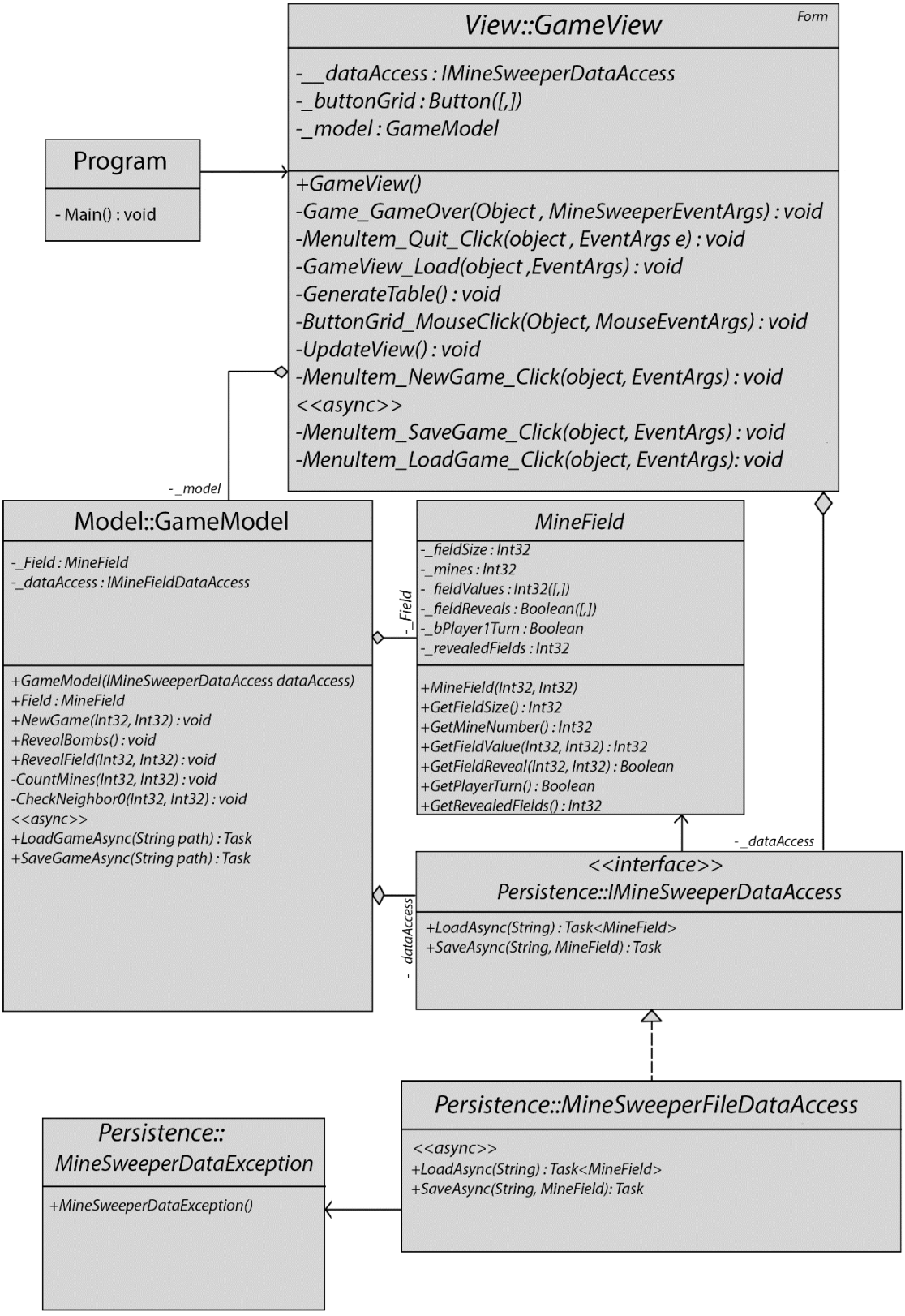
- **Programszerkezet:**
  A programot háromrétegű architektúrában valósítjuk meg. A megjelenítés a View, a modell a Model, míg a perzisztencia a *Persistence* névtérben helyezkedik el. A program csomagszerkezete a 2. ábrán látható.
- **Perzisztencia:**
  - Feladata az adatkezelés valamint a *mentés/betöltés* biztosítása.
  - Az MineField egy érvényes mezőt és a játékhoz fontos információkat tartalmaz, a mezők értéke (**_fieldValues**) ez lehet 6x6, 10x10, 16x16, mezőn lévő aknák összege(**_mines**), Cellák felfedettsége (**_fieldReveals**) a következő játékos(**_bPlayer1Turn**) egy számláló a már felfedett mezőknek (semleges végkimenetelhez) és definiálva vannak ezeknek az értékeknek a Get/Set-erei.
  - Hosszútávú adattárolásához lehetőséget ad az IMineSweeperDataAccess interface.
    Így elérhető egy Betöltés(**LoadAsync**) valamint egy Mentés(**SaveSync**) funkció.
    A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg.
  - Az interface szöveges file alapú kezelését az **MineSweeperFileDataAccess** osztály teszi lehetővé. A felmerülő hibákat az **MineSweeperDataException** kezeli.
  - A program az adatokat szöveges fileként tárolja ezeket az adatokat a programba bel lehet tölteni amikor nincs játék vége állapot.

- A file első sora megadja, a pálya méretét, benne lévő aknák számát, a következő játékos körét, és hogy eddig mennyi felfedett cella van.
  Alatta egy reprezentációa van a mező értékeinek.
  Az alatt pedig a felfedettséget reprezentációja.



- Modell:
  - A modell-t a ***GameModel*** osztály valósítja meg, ami a tábla logikai lépéseit kezeli le.
    Ebben az osztályban példányosítjuk az **MineField** osztályt.
  - Itt hívódik a **CheckNeighboro()** ami rekurzvívan nézi a 0-kat
  - A **RevealField()** Ami lényegében lépteti a játékot, és felfedi a mezőket
  - A **SetupValues()** ami kezdőértékeket ad egy fress játéknak.
  - A CountMines ami a **SetupValues()**-ben használt és a szomszédos aknákat számlálja majd adja értékként a mezőnek.
  - A **NewGame()** amivel alaphelyzetbe állítjuk a játékot.
  - Végül a **RevealBombs()** ami csak látványelemként van jelen
- Nézet:
  - A nézetet a **GameView** osztály valósítja meg ami tárloja a **GameModel** és a **DataAccess** egy példányát.
  - A játéktábla egy **button**-ökből álló 2 dimenziós tömbben jelenik meg.

- A pályaméret választásához meg van hívva egy InputForm ami egy átmeneti ablakként üzemel.



Tesztelés:

- A modell funcionalitását egységtesztekkel ellenőrizzük az ~~MineSweeperTest~~ **UnitTest1** osztályban
- Az álábbi tesztek kerültek megvalósításra
    - ***TestNewGame()*** →
      Új játék indítása és kezdőértékek ellenőrzése
    - ***TestReveal()*** →
      Játékosok lépésének ellenőrzése
    - ***TestPlayerTurn()***→
      Játékosok váltakozását teszteli

Készítette: Márton Zoltán (B44T65)