Nádasi Tamás INXGWX nadasi.tamas22@gmail.com

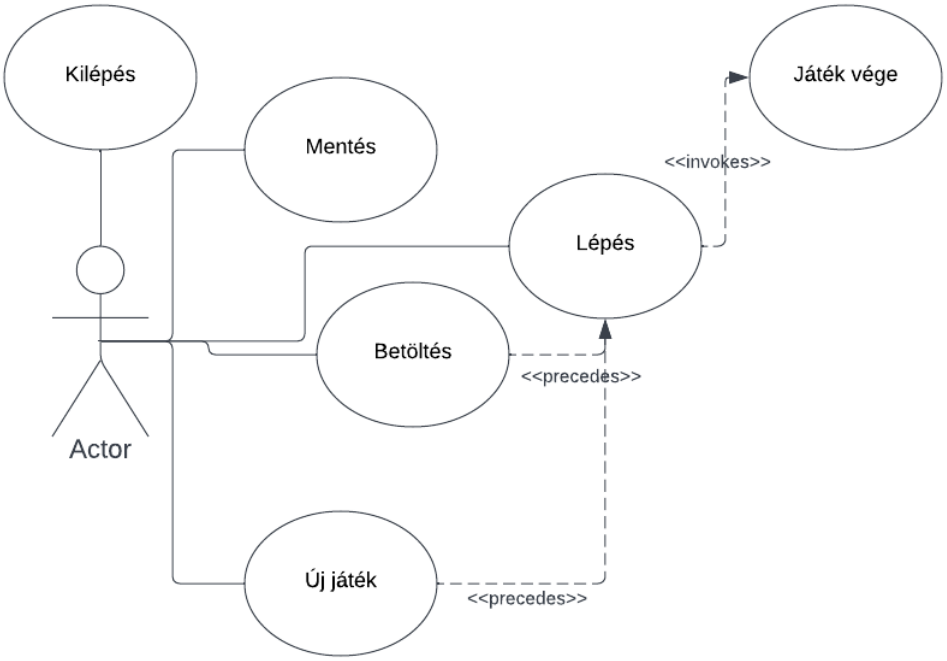
Aknakereső dokumentáció

Feladat:

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. Amikor egy mezőre kattintunk, 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:

- A program indításkor eldönthetjük, hogy hány mezőn akarjuk játszani a játékot (6x6, 10x10, 16x16).
- A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: Menu (New Game, Save Game, Load Game), a New Game-en belül, pedig a mezők száma választható: New Game (6x6, 10x10, 16x16). Az ablak tetején megjelenítjük, hogy melyik játékos van soron.
- A játéktábla mérete a felhasználó által választott (6x6, 10x10, 16x16). Amikor egy mezőre kattintunk, 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 játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (egyik vagy másik játékos nyert, vagy döntetlen az eredmény). Szintén dialógusablakokkal végezzük el a mentést, illetve betöltést, a fájlneveket a felhasználó adja meg.



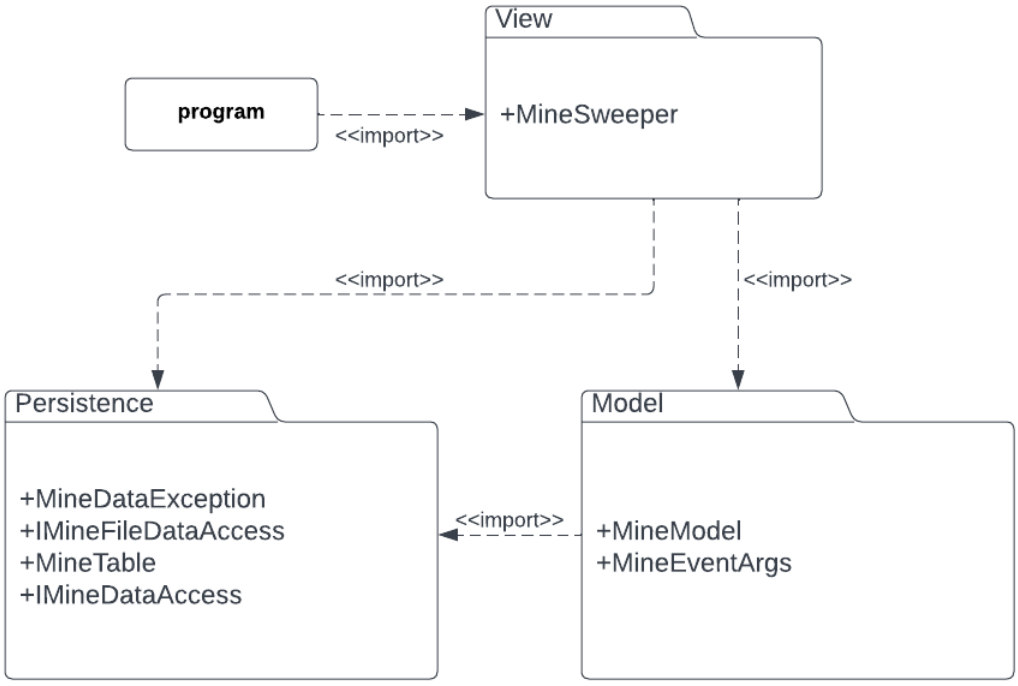
Tervezés

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 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 Minesweeper táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
- A hosszú távú adattárolás lehetőségeit az IMineDataAccess 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 MineFileDataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a MineDataException 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.

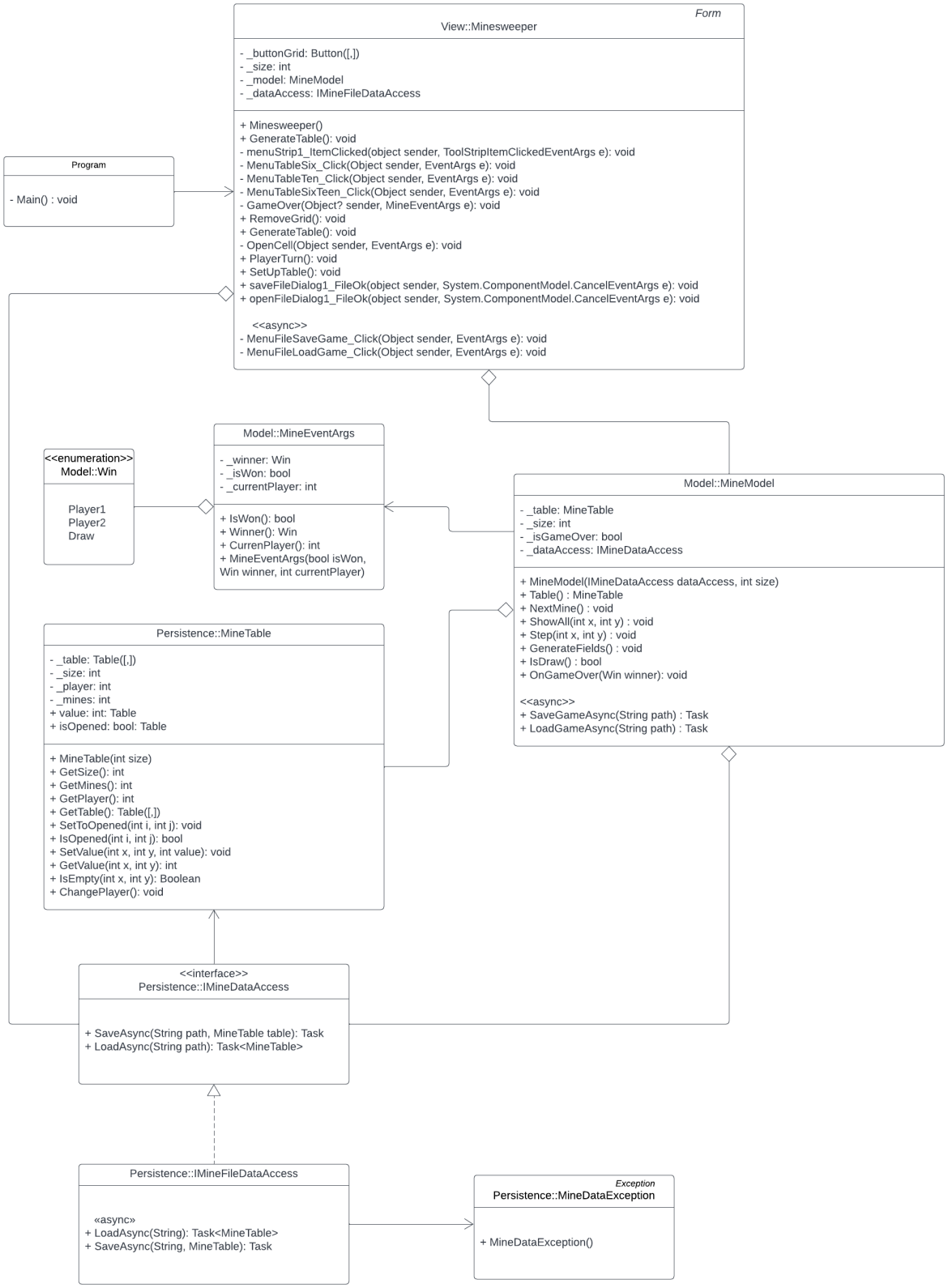


Modell:

- A modell lényegi részét a MineModel osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit. A típus lehetőséget ad új játék kezdésére (NewGame), valamint lépésre (Step). Új játéknál megadható a kiinduló játéktábla.

• A játékállapot változásáról a GameAdvanced esemény, míg a játék végéről a OnGameOver esemény tájékoztat. Az események argumentuma (MineEventArgs) tárolja a győzelem állapotát, a lépések számát.

• A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (LoadGameAsync) és mentésre (SaveGameAsync)



Nézet:

- A nézetet a MineSweeper 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 (OpenCell) külön metódusok végzik.

Tesztelés:

A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a UnotTest1 osztályban.

Az alábbi tesztesetek kerültek megvalósításra:

- Table6Test: Egy 6x6-os tábla legenerálása üres mezőkkel.
- Table10Test: Egy 10x10-es tábla legenerálása üres mezőkkel.
- StepTest: Egy mező nyitásának és a benne lévő értéknek és az aktuális játékosnak az ellenőrzése
- GameOverTest: Ellenőrzi a játék végén az aktuális értékeket, úgy mint egy mező értékét, azt, hogy valóban véget ért a játék, valamint a játék végén az aktuális játákost.
- Loadtest: Egy fájl betöltésének a tesztelése mockolt perzisztencia réteggel.