1. Beadandó feladat dokumentáció

Készítette:
Czigány Kristóf Péter
LGLFAT

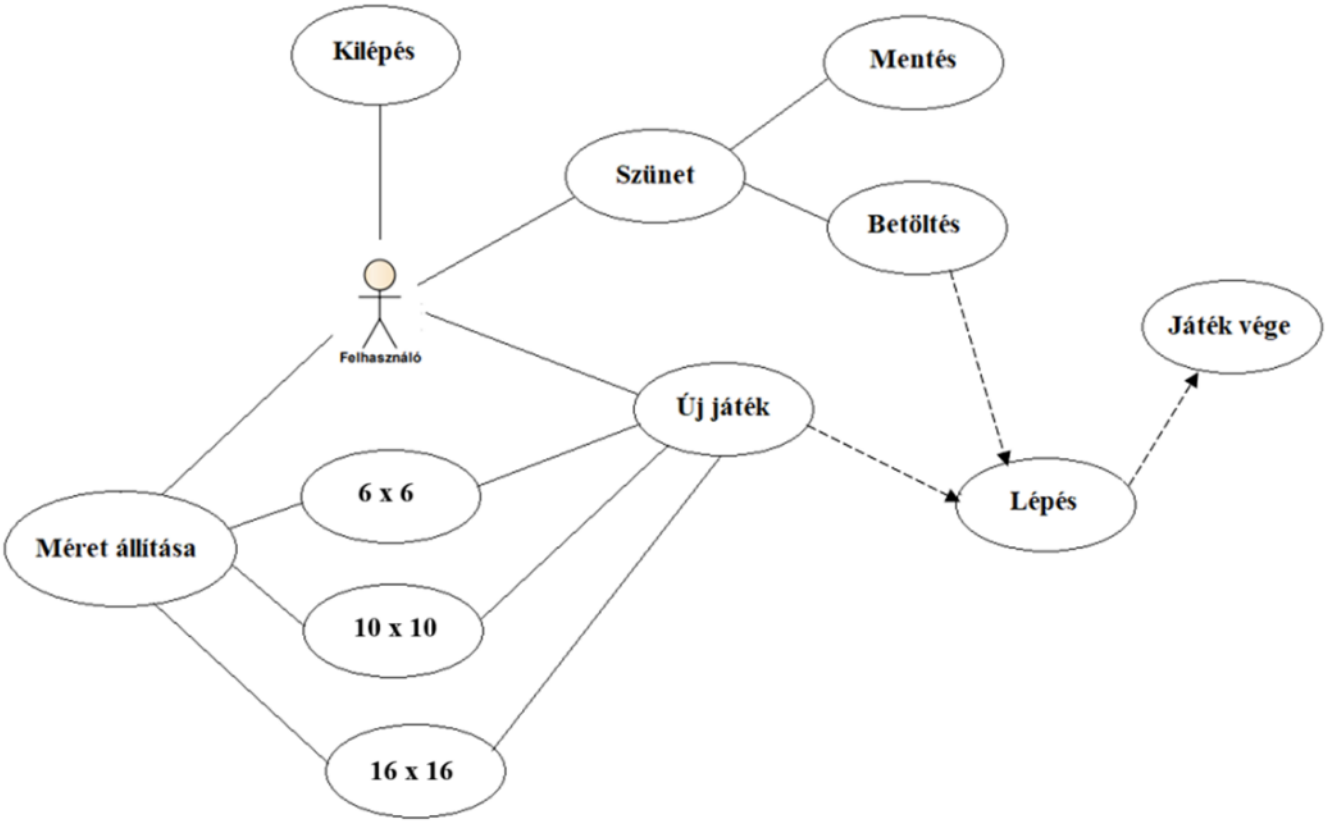
Feladat (15):

Aknakereső

Készítsünk programot, amellyel a következő játékot játszhatjuk.
Adott egy $n \times n$ elemből álló játékpálya, ahol a játékos két üldöző elől próbál menekülni, illetve próbálja őket aknára csalni.
Kezdetben a játékos 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 aknamezőt (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).

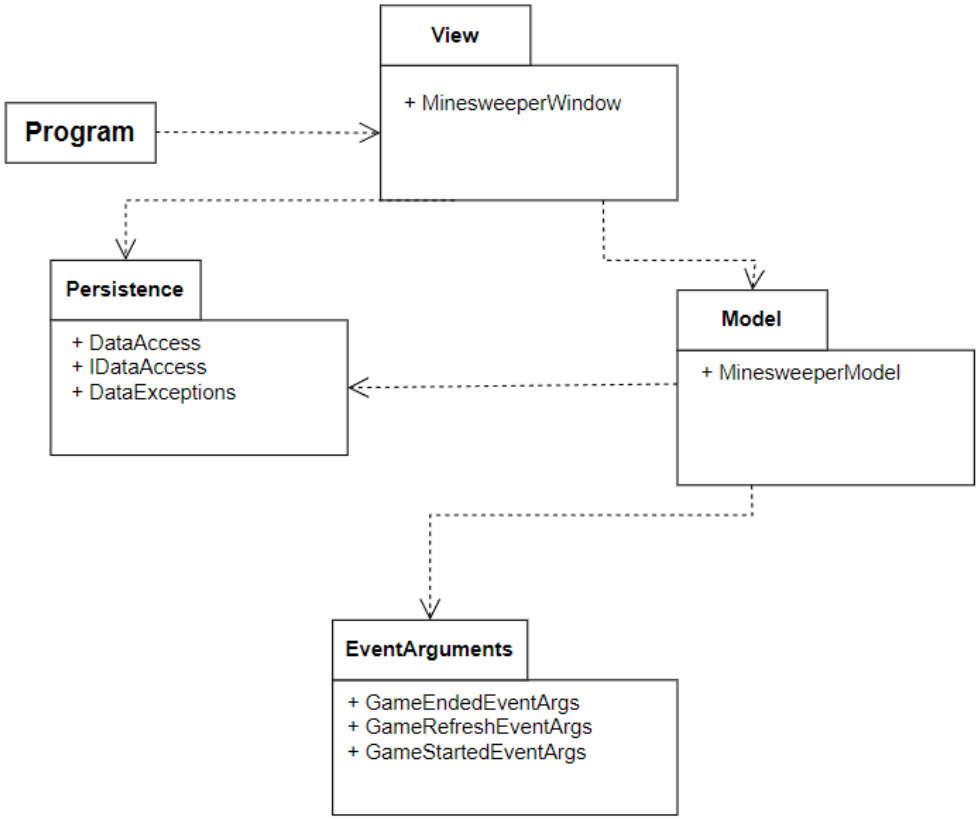
Elemzés:

- A játékot 3 különböző méretű pályán lehet játszani, ami alapértelmezetten 10 x 10 méretű.
- A feladatot egyablakos asztali Windows Forms grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (Save Game, Load Game), New Game (6 x 6, 10 x 10, 16 x 16), Start Game, Continue, Pause. Az ablak alján megjelenítünk egy státuszsort, amely az eltelt időt mutatja, valamint a soron következő játékost.
- A játékosok egymás ellen játszanak, a cél, hogy ne kattintson aknára.
- A játéktéren véletlenszerűen elhelyeztünk aknákat, a nem akna mezők pedig számmal jelzik, hogy hány akna van körülötte. Kezdetben egyik mező sincs felfedve.
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak és tájékoztatja a játékosokat, hogy melyik játékos nyert, esetleg döntetlen lett, valamint a játékidőről is.
- Felhasználói esetek:


Kilépés
Mentés
Szünet
Betöltés
Játék vége
Felhasználó
Új játék
6 x 6
Méret állítása
Lépés
10 x 10
16 x 16

Tervezés:

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

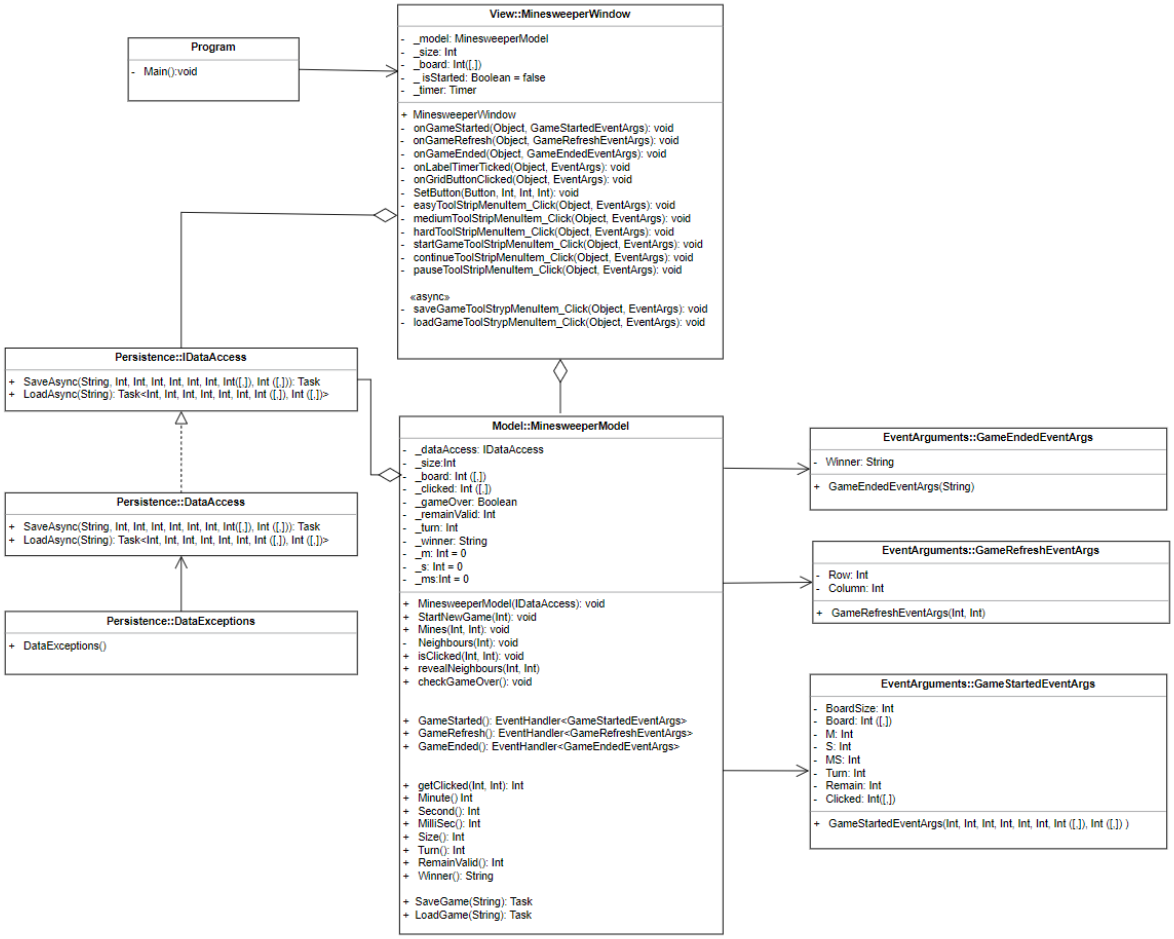


  - A program szerkezetét két projektre osztjuk implementációs megfontolásból: a Persistence és Model csomagok a program felületfüggetlen projektjében, míg a View csomag a Windows Formstól függő projektjében kap helyet.

- Perzisztencia:
  - Az adatkezelés feladata az Minesweeper táblával kapcsolatos információk tárolása, valamint a betöltés és a mentés biztosítása.
  - A hosszú távú adattárolás érdekében létrehoztuk az IDataAccess interfészt amely lehetőséget ad az adott játék elmentésére(SaveGame) és előzetes mentések betöltésére(LoadGame).
  - Az interfész szöveges alapú adatkezelésére a DataAccess osztály szolgál.
  - A fájlkezelés során fellépő hibákat a DataExceptions kivétel jelzi.

    - A program az adatokat szöveges fájlként tudja eltárolni, melyek az txt 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, a második sor a játékban eltelt percet, a harmadik sor a játékban eltelt másodpercet, a negyedik sor a játékban eltöltött századmásodpercet, az ötödik sor az éppen soron lévő játékost, a hatodik sor a még nem felfedett mezők számát, a fájl többi része pedig a pálya tartalmát, mely 0 és 9 közötti számok lehetnek, ahol 0-8 a szomszédos aknák számát, 9 az aknát reprezentálja, valamint -1 a még nem felfedett mezőt, 0 pedig a felfedett mezőt reprezentálja.
- Modell:
    - A modell lényegi részét az *MinesweeperModel* osztály valósítja meg, amely szabályozza a játéktábla tevékenységeit, valamint a játék egyéb paramétereit. Ez az osztály ad lehetőséget új játék kezdésére (*StartNewGame*). A játék végét a *checkGameOver* ellenőrzi. A mezők felfedéséért az *isClicked* valamint a *revealNeighbours* felel. Az aknák elhelyezéséért a *Mines* felel.
    - A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (*LoadGameAsync*) és mentésre (*SaveGameAsync*).
    - A játék állapotát a *ViewRefresh* frissíti.
- Nézet:
    - A néztet az MinesweeperWindow osztály biztosítja, amely tárolja a modell egy példányát (*_model*), valamint az adatelérés konkrét példányát (*_dataAccess*).
    - A játéktáblát egy dinamikusan létrehozott gombmező reprezentálja. A felületen létrehozzuk a megfelelő menüpontokat, illetve státuszsort, valamint dialógusablakokat, és a hozzájuk tartozó eseménykezelőket.
    - A játék időbeli kezelését egy időzítő végzi (*_timer*), amelyet mindig aktiválunk játék során, illetve inaktiválunk, amennyiben bizonyos menüfunkciók futnak.

A program teljes szerkezete:



Tesztelés:

A modell funkcionalitása egységtesztek segítségével lett elllenőrizve az EscapeTest osztályban.

Az alábbi tesztesetek kerültek megvalósításra:

- StartNewGameBoardSize
- StartNewGameBoardFields
- CheckGameOverIfFirstPlayerWin
- CheckGameOverIfSecondPlayerWin
- CheckGameOverIfTie