1. Beadandó feladat (WinForms) dokumentáció

Készítette:

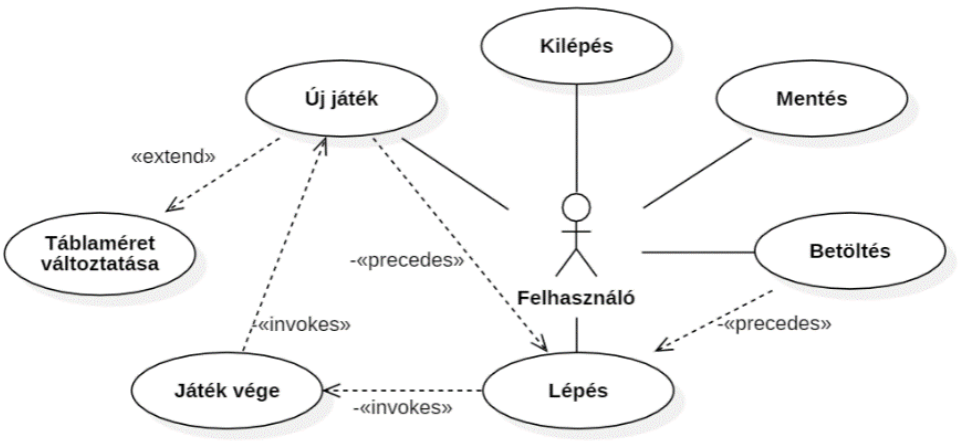
Szabó Péter Bence
E-mail: QDMPVQ@inf.elte.hu

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. 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:

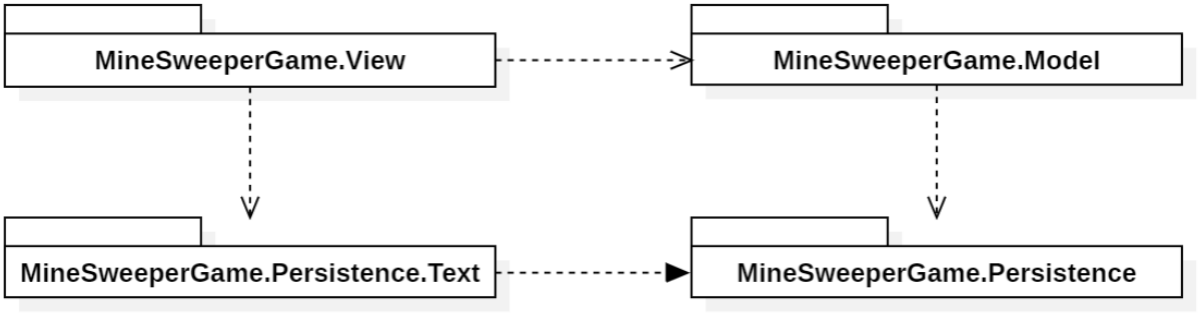
- A játék három nehézségi szinttel rendelkezik: könnyű (6x6-os méretű tábla), közepes (10x10-es méretű tábla), nehéz (16x16-os méretű tábla). A program indításkor automatikusan 6x6-os (könnyű) táblát állít be, és elindul egy új játék.
- A feladatot kétablakos (fő ablak, méretválasztó ablak) asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- A fő ablakban elhelyezünk három gombot: Új játék (méretválasztás), Játék mentése, Játék betöltése. A jelenleg soron következő játékost a bal felső sarokban jelenítjük meg.
- A játéktáblát egy nyomógombokból álló rács reprezentálja (amely a korábban említett nehézségi szintek közül választottnak megfelelő méretű). A nyomógomb egérkattintás hatására felfedi az adott mezőt. Ha a mezőn akna van, a játék véget ér, minden akna helyzete láthatóvá válik. Ha nincs akna, akkor a kattintott mező tartalma láthatóvá válik, illetve ha az 0, akkor minden körülötte lévő mezőé is (amíg nem ütközik nem 0 értékbe).
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (valamelyik játékos nyert, vagy minden nem akna mező felfedésre került, azaz döntetlen a játék). 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.

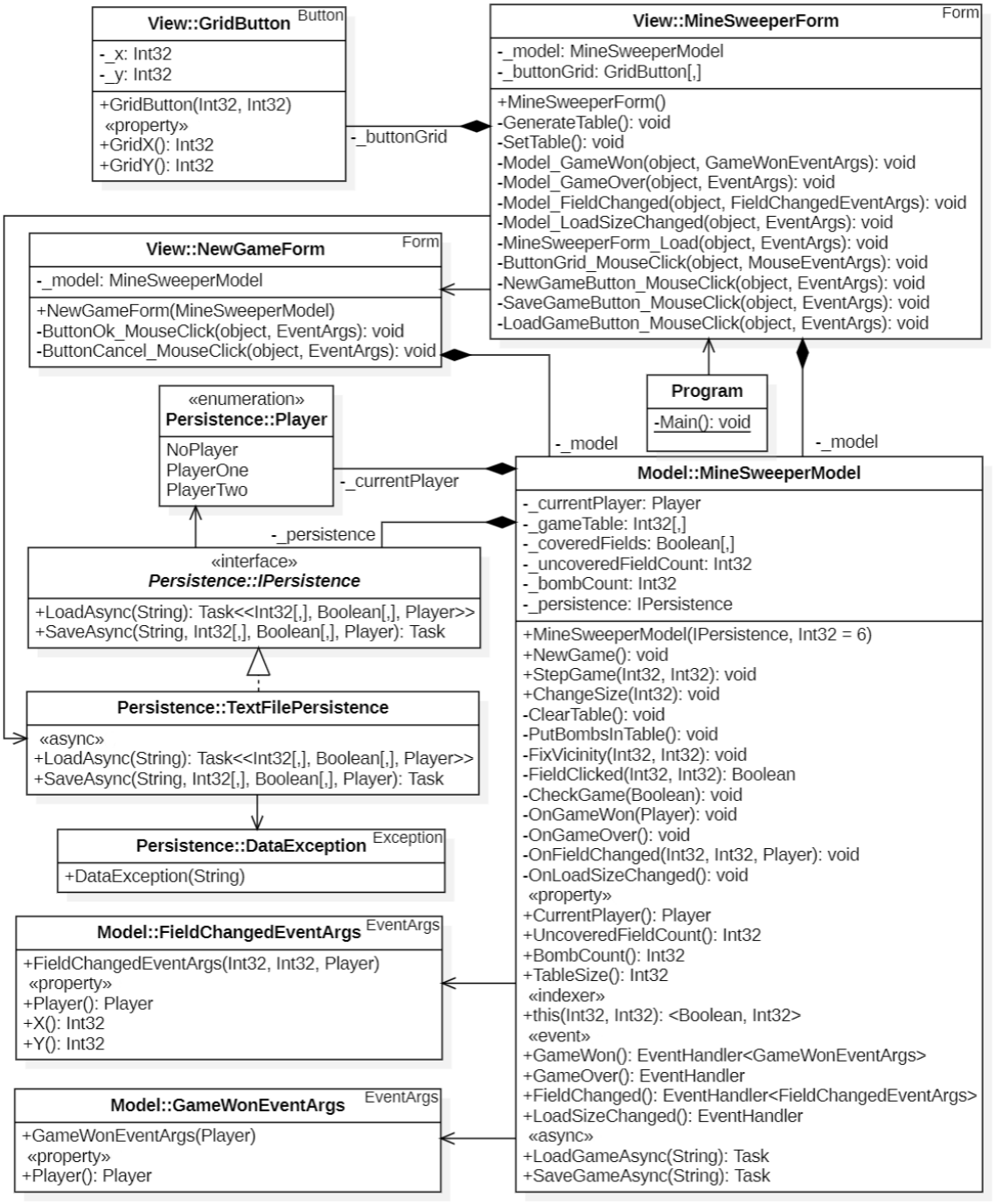


2. ábra: Az alkalmazás csomagdiagramja

- **Perzisztencia:**
  - Az adatkezelés feladata a betöltés/mentés biztosítása.
  - A hosszú távú adattárolás lehetőségeit az `IPersistence` 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 `Persistence.Text` projektben található `TextFilePersistence` osztály valósítja meg. A fájlkezelés során fellépő hibákat a `DataException` kivétel jelzi.
  - A program az adatokat szöveges fájlként tudja eltárolni, melyek a `.sav` kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
  - A fájl első sora megadja a játéktábla méretét (a tábla NxN-es, ahol N 6, 10 vagy 16 lehet) és a jelenleg soron következő játékos azonosítóját szóközzel elválasztva. A fájl további sorai a játéktábla értékeit tartalmazzák (a játéktábla méretének megfelelően 2*N adat található N sorban, és egy soron belül felváltva szerepelnek a mezők értékei, majd a felfedettségük, amely 1, ha még nem felfedett a mező, és 0, ha már igen).
- **Modell:**
  - A modell lényegi részét a `MineSweeperModel` osztály valósítja meg, amely szabályozza a tábla tevékenységeit, tartalmazza a játéktáblát (`_gameTable`) és a hozzátartozó felfedettségeket (`_coveredFields`), a játék egyéb paramétereit, úgymint az aknák száma (`_bombCount`) és a felfedetlen mezők száma (`_uncoveredFieldCount`), illetve az adatelérés konkrét példányát (`_persistence`). A típus lehetőséget ad új játék kezdésére (`NewGame`), lépésre (`StepGame`), valamint a játéktábla méretének megváltoztatására új játéknál (`ChangeSize`).
  - A játékállapot változásáról a `FieldChanged` esemény, míg a játék végéről a `GameOver` vagy a `GameWon` esemény tájékoztat (a játék kimenetelétől függően). Az események argumentumai, a `FieldChangedEventArgs` tárolja a megváltoztatott mező koordinátáit, illetve a rákattintó játékost, a `GameWonEventArgs` pedig a győztes játékost.

- A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (`LoadGameAsync`) és mentésre (`SaveGameAsync`).
- A játékosokat a `Player` felsorolási típuson át kezeljük.

- **Nézet:**
  - A nézetet a `MineSweeperForm` osztály biztosítja, amely tárolja a modell egy példányát (`_model`).
  - A játéktáblát egy dinamikusan létrehozott gombmező (`_buttonGrid`) reprezentálja. A felületen létrehozzuk a megfelelő gombokat, illetve címkét, valamint dialógusablakokat, és a hozzájuk tartozó eseménykezelőket. A játéktábla generálását (`GenerateTable`), illetve az értékek beállítását (`SetTable`) külön metódusok végzik.
  - A program teljes statikus szerkezete a 3. ábrán látható.



3. ábra: Az alkalmazás osztálydiagramja

Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a `MineSweeperModelTest` osztályban.
- Az alábbi tesztesetek kerültek megvalósításra:
  - `MineSweeperConstructorTest`: Új játék indítása, a tábla méretének és felfedettségének ellenőrzése, illetve az aknák számának, és körülöttük lévő mezők értékének ellenőrzése.
  - `MineSweeperStepGameTest`: Játékbeli lépés hatásainak ellenőrzése egy mockolt betöltésen. Lépés nem nulla számot tartalmazó mezőn, nullát tartalmazó mezőn, már felfedett mezőn.
  - `MineSweeperUncoveredFieldNumberTest`: Egy pár lépés után a felfedett mezők számának ellenőrzése.
  - `MineSweeperIndexerValidTest`: A tábla indexelése, visszatérési értékek ellenőrzése valós indexek esetén.
  - `MineSweeperIndexerInvalidTest`: A tábla indexelése, visszatérési értékek ellenőrzése helytelen indexek esetén. Ebben a metódusban kivételt várunk eredményként.
  - `MineSweeperGameWonTest`: A győzelem eseménye kiváltásának ellenőrzése. Azt is ellenőrizzük, hogy a helyes játékos-e a győztes.
  - `MineSweeperGameLoadAsyncTest`: A játék modell betöltésének tesztelése mockolt perzisztencia réteggel.
  - `MineSweeperGameSaveTest`: A játék modell mentésének tesztelése mockolt perzisztencia réteggel.