1. Beadandó feladat dokumentáció

Készítette:

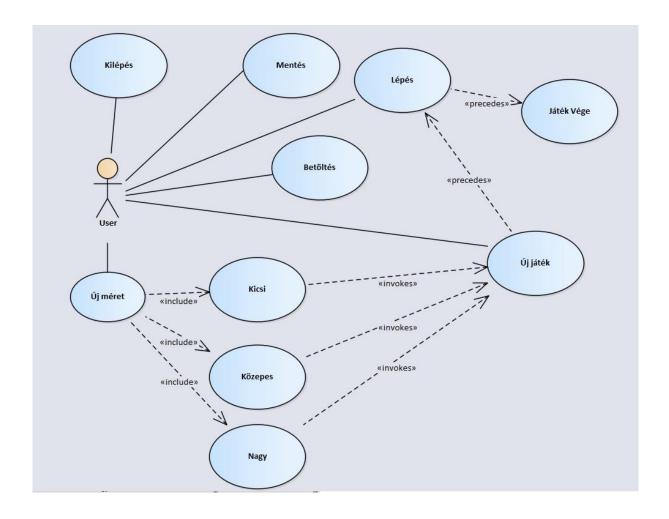
Hallgató Harald E-mail: haha@inf.elte.hu

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 játékot három méretben játszhatjuk: kicsi (6x6 mező, 5 akna), közepes (8x8 játékmező, 8 akna), nagy (16x16 játékmező, 12 akna). A program indításkor közepes nehézséget állít be, és automatikusan új játékot indít. Az aknák az adott számban véletlenszerűen lesznek elhelyezve a játékmezőn.
- A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (Új játék, Játék betöltése, Játék mentése), Új méret (Kicsi,közepes,nagy).
- A játéktáblát egy n×n nyomógombokból álló rács reprezentálja. A nyomógomb egérkattintás hatására felfedi az adott mező tartalmát. Ha 0 a mező értéke (azt üres mezővel fogja jelölni a játék), akkor a körülötte lévő mezőket is feloldja (ha azok között is van nulla, akkor a 0-s mező körüli mezők is feloldódnak és így tovább). A nem feloldott mezőket fekete színnel jelöljük. A 0-s mezőket kék mezőként jelenítjük meg.
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (minden mezőt felfedtünk ami nem akna, vagy az egyik játékos aknára lépett). 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:

<u>Programszerkezet:</u>

 A programot háromrétegű architektúrában valósíjuk meg. A megjelenítés a View, a modell a Model, míg a perzisztencia a Persistence névtérben helyezkedik el.

Perzisztencia:

- Az adatkezelés feladata a Mine_Detector táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
- A Mine_DetectorTable osztály egy érvényes Mine_Detector (aknakereső) táblát biztosít (azaz mindig ellenőrzi a beállított értékeket), ahol minden mezőre ismert az értéke (_fieldValues), illetve a zároltsága (_fieldLocks). A _fieldLocks Booleanokat tartalmazó mátrixban tároljuk azoknak a mezőknek a pozicióit, amiknek értéke még felfedetlen. A tábla alapértelmezés szerint 8x8-as 7 aknával, de ez a konstruktorban paraméterezhető. A tábla lehetőséget ad az állapotok lekérdezésére

(TableIsFull, IsLocked, IsEmpty, GetValue, Size), illetve direkt beállítás (SetValue, SetLock) elvégzésére.

- A hosszú távú adattárolás lehetőségeit az IMine_DetectorDataAccess 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 Mine_DetectorFileDataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a

Mine DetectorDataException 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.
- A fájl sorai tartalmazzák a táblaméretet, az aktuális játékost, a zárolt mezőket, illetve a tábla mezőinek értékeit.

<u>Modell:</u>

- A modell lényegi részét a Mine_DetectorGameModel osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, úgymint az aktuális játékost (_actplayer) és az aktuális győztest (_winner). A típus lehetőséget ad új játék kezdésére (NewGame), valamint lépésre (StepGame).
- A játékállapot változásáról a **GameOver** esemény tájékoztat. Az események argumentuma

(Mine_DetectorEventArgs) tárolja a győztest, valamint az utoljára kattintott mező paramétereit.

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

(SaveGameAsync)

• A játék méretét a FieldSize felsorolási típuson át kezeljük, és konstansok segítségével tároljuk az egyes méretek paramétereit.

<u>Nézet:</u>

- A nézetet a GameForm 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 (SetupTable) külön metódusok végzik.

A program teljes statikus szerkezete a 3. ábrán látható. Model::Mine_DetectorGameModel Program _buttonGrid: Button([,]) Main(): void dataAccess: IMine DetectorDataAccess _dataAccess: IMine_DetectorDataAccess _model: Mine_DetectorGameModel fieldsize: FieldSize ButtonGrid_MouseClick(Object, MouseEventArgs): void minecount: Int32 _table: Mine_DetectorTable _tableSize: Int32 GameForm_Load(Object, EventArgs): void winner: Players = Players. Nobody «enumeration» GenerateTable(): void MineCountLarge: const Int32 = 12 View:Players MenuFileLoadGame(Object, EventArgs): void MenuFileSaveGame_Click(Object, EventArgs): void MineCountMedium: const Int32 = 8 MineCountSmall: const Int32 = 5 MenuGameMediumClick(Object, EventArgs): void Player2 tableSizeLarge: const Int32 = 16 MenuGameSmall_Click(Object, EventArgs): void Nobody tableSizeMedium: const Int32 = 8 SetupTable(): void tableSizeSmall: const Int32 = 6 View_GameOver(Object, Mine_DetectorEventArgs): void Actplayer(): Players CheckDraw(int, int): void Model::Mine_DetectorEventArgs GameOver(): EventArgs < Mine_DetectorEventArgs > locationofmine: Tuple<Int32,Int32> GenerateTable(): voi winner: Players LoadGameAsync(String): async Task Locationofmine(): Tuple<int,int> Minecount(): int Mine_DetectorEventArgs(Int32, Int32, Players): void NewGame(): void Winner(): Players OnGameAdvanced(int, int, Players): void SaveGameAsync(String): async Task Step(int, int): void Persistence::Mine_DetectorFileDataAccess StepGame(int, int): void Table(): Mine DetectorTable LoadAsync (String): as ny c Task< Mine Detector Table TableSize(): int SaveAsync (String, Mine_DetectorTable); async Task Persistence:: «interface» Mine_DetectorDataException Persistence : Mine_DetectorTable Persistence:: IMine_DetectorDataAccess Mine_DetectorDataException():void _actplayer: Players LoadAsync(String):Task<Mine_DetectorTable fieldLocks: Boolean[,] SaveAsync(String, Mine_DetectorTable): Task fieldSize: Int32 fieldValues: Int32[.] Actplayer(): Players GetValue(int, int): Int32 IsEmpty(int, int): Boolean IsLocked(int, int): Boolean Mine_DetectorTable(int): void SetLocked(int, int): void SetValue(int, int, Boolean): void

3. ábra: Az alkalmazás osztálydiagramja

Size(): Int32 TableIsFull(): Boolean this[](int, int): Int32

Tesztelés:

• A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a

Mine_DetectrGameModelTest osztályban.

- Az alábbi tesztesetek kerültek megvalósításra:
- Mine_DetectorSmallNewGameTest,
 Mine DetectorMediumNewGameTest,

Mine_DetectorLargeNewGameTest: Új játék indítása, a mezők kitöltése, valamint az aknák számának, következő játékos helyességének ellenőrzése a pálya méretének függvényében.

- Mine_DetectorStepGameTestPlayer2Win: A győzelem vizsgálata, az eseménykezelő értékeivel, amikor a második játékos győz.
- Mine_DetectorStepGameTestPlayer1Win: A győzelem vizsgálata, abban az esetben, mikor az első játékos lép bombára hamarabb.
- Mine_DetectorStepGameTestDraw: A játékot illetve az eseménykezelők értékeit ellenőrízzü akkor, ha a játék kimenetele döntetlenre végződik.