Eseményvezérelt Alkalmazások Fejlesztése

4. beadandó/15. feladat

Gonda Dávid
BIXU0S
gonda.david18@gmail.com

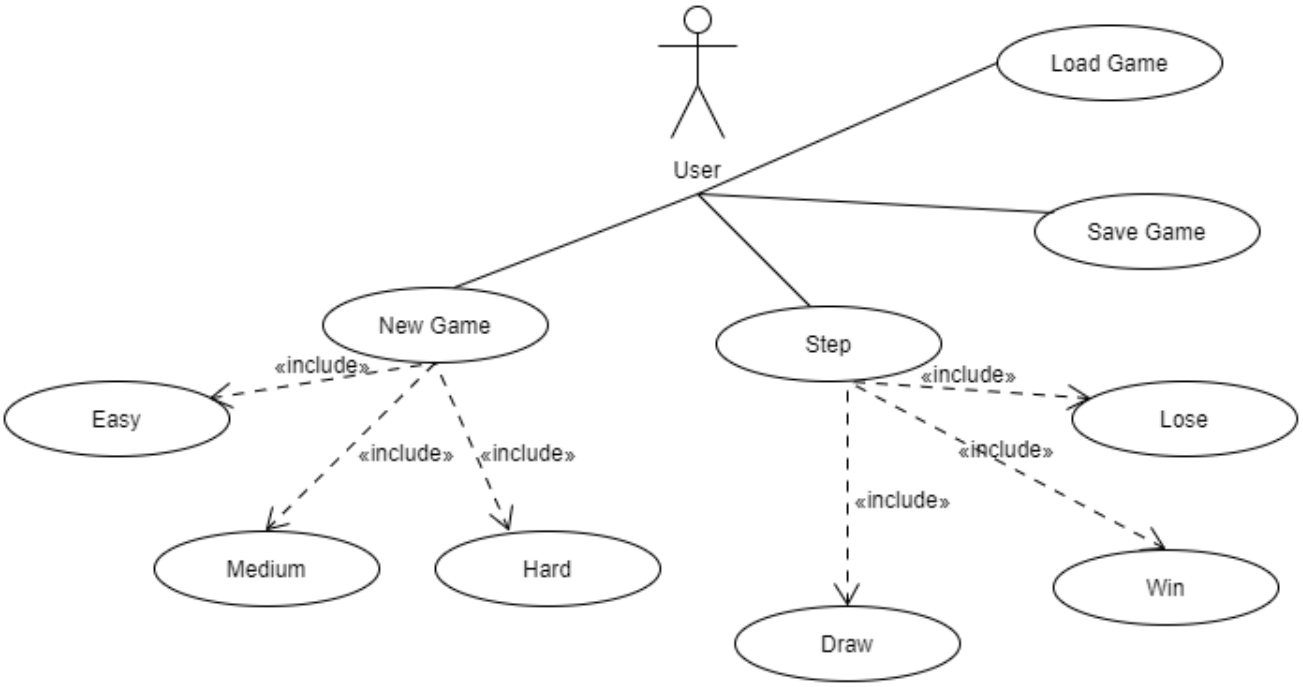
Feladat

Aknakereső

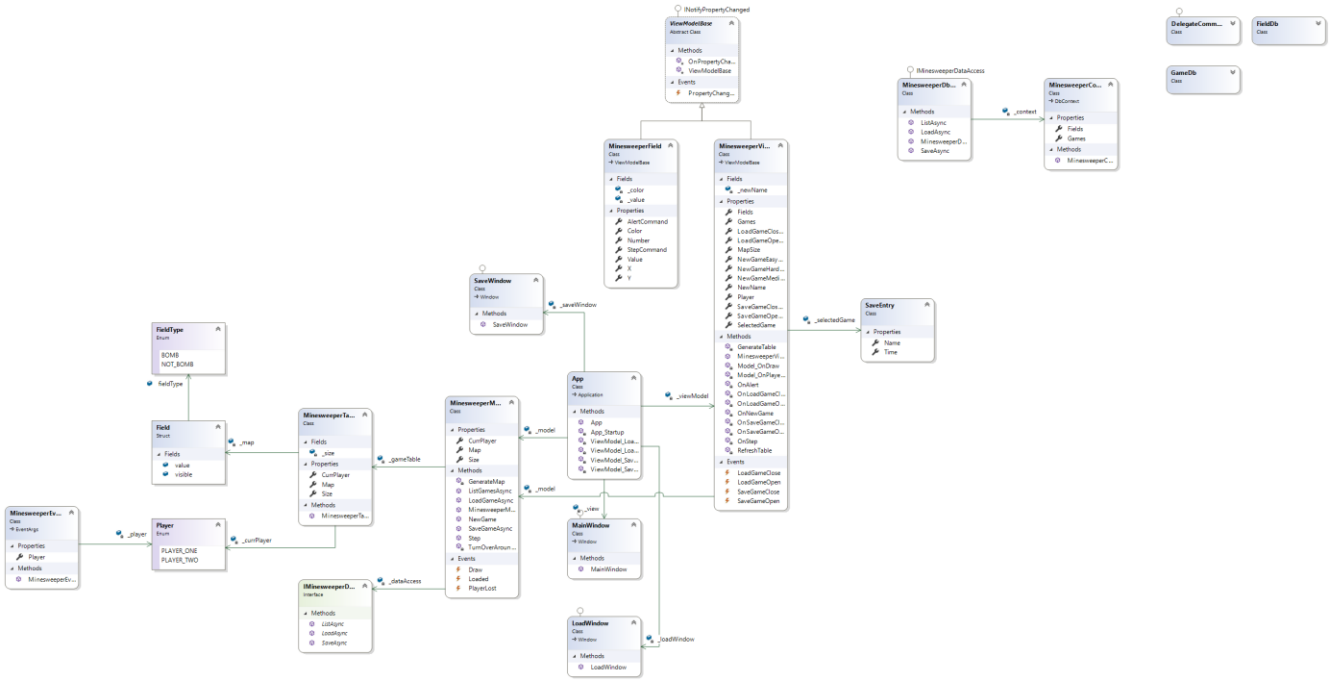
Készítsünk programot, amellyel az aknakereső játék két személyes változatát játszhatjuk.
Adott egy $n \times n$ mezőből álló tábla, amelyen rejtett aknákat helyezünk el. A többi mező szintén elrejtve tárolják, hogy a velük szomszédos 8 mezőn hány akna helyezkedik el.
A játékosok felváltva léphetnek. 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

- A játékot 3 nehézségi szinten lehet játszani, könnyűn (6 x 6-os pálya), közepesen (10 x 10-es pálya), és nehezen (16 x 16-os pálya).
- A feladatot egyablakos asztali alkalmazásként WPF grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: New Game (Easy, Medium, Hard), Save Game, Load Game. Az ablak alján egy státuszsorba kiírjuk az aktuális játékost.
- A játéktáblát dinamikusan létrehozott nyomógombokkal fogjuk megvalósítani. Ezek alapból fehér gombok, kattintás után felfedjük mi van alatta.
- Bomba esetén az aktuális játékos veszít, ez után új játék kezdődik az aktuális nehézséggel.
- Nulla esetén a környező mezőket is felfedjük (ha ott is van nulla, ugyanígy járunk el).
- Abban az esetben, ha már csak bomba van lefedve a táblán, a játék döntetlennel végződik, ez után új játék kezdődik az aktuális nehézséggel.
- Jobb egérgombbal egy mezőt megjelölhetünk, ezzel jelezve a bomba veszélyét.



Osztálydiagram



A program 4 nagyobb egységből áll, egy *model*ből, egy *viewmodel*ből, egy *view*ból, és egy *persistence* rétegből.

- **Model:**
  Ez a réteg felel a játék üzleti logikájáért. Főbb metódusai a *Step* (egy játéklépést hajt végre), a *NewGame* (új játékot generál térképnagyság szerint), és a *GenerateMap* (legenerálja a .pályát).

- **ViewModel:**
  A *view*ot a *model*el összekötő egység, leginkább paramétereket tartalmaz a *view* számára a *model*ből, de tartalmazza a pálya legenerálását is vizuálisan, az inputok hatását is kezeli, összeköti a *model* metódusaival, illetve a job egérgomb hatására jelzést rak le a *model*től függetlenül.

- **View:**
  A grafikust megjelenítést leíró egység. Áll egy mentő-/betöltő-/főablakból. Dolga a grafikus megjelenítésen túl a különböző **commandok, változók** nevének megadása.

- **Persistence:**
  A mentés/betöltést végző egység. Adatbázissal (Entity Framework) valósítjuk meg, aszinkron módon. Több mentett állást is lehet tárolni, amik közül aztán lehet választani betöltéskor.

Tesztelés

- A modell funkcionalitását egységtesztek segítségével ellenőriztük a **MinesweeperTest** osztályban. Ehhez létrehoztunk egy mock perzisztenciát (egy mock táblával együtt).

- Az alábbi tesztesetek kerültek megvalósításra:

  - **MinesweeperModelNewGameTest:**
    Új játék indításának az ellenőrzése.

  - **MinesweeperModelStepTest:**
    A lépésnek, annak hatásainak ellenőrzése.

  - **MinesweeperModelTurnOverAroundZeroTest:**
    Nullára való lépés után a mező körül lévő mezők felfedésének ellenőrzése, több egymás melletti nullával.

  - **MinesweeperModelGenerateMapTest:**
    Tábla generálásának az ellenőrzése.