**Feladat**

Készítsünk programot, amellyel az aknakereső játék két személyes változatát játszhatjuk. Adott egy 𝑛 × 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 féle pályaméreten játszhatod. A 6x6-os a 10x10-es és a 16x16-os.
* A feladatot egyablakos asztali alkalmazásként Windows Presentation Foundation grafikus felülettel valósítjuk meg.
* Az ablak tetején egy lenyiló menü van elhelyezve amely a játék mentését betöltését és teljesen új játék indítását teszi lehetővé három féle méretben.
* A játéktáblát a már említett méretű rácsok alkothatják. Ezekre kattintva amennyiben a szereplő közvetlen környezetére kattintasz felnyil és elérhetővé válik az alatta lévő szám avagy aknára kattintva a játék a végééhez ér.
* A játék a játék végeztével feldob egy dialogus ablakot ami azt jelzi, hogy melyik fél nyert.

**Felhasználó esetek:**

A képen diagram, kör, sor látható

Automatikusan generált leírás

**Tervezés**

Programszerkezet:

* A programot MVVM architektúrával valósítjuk meg. A perzisztencia a persistence mappában a modell a model mappában és a view a view mappában. ennek megfelelően View, Model, ViewModel és Persistence névtereket valósítunk meg az alkalmazáson belül. A program környezetét az alkalmazás osztály (App) végzi, amely példányosítja a modellt, a nézetmodell és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést. A program csomagszerkezete a 2. ábrán látható.
* A program két projektre lett osztva implementációs megfontolásokból. A persistence és a model a proejkt felület független részében kapott helyett ezzel szemben View és a Viewmodel a WPF függő helyeken kapott helyet.

A képen szöveg, Betűtípus, képernyőkép, diagram látható

Automatikusan generált leírás

Persistencia

* Az adatkezelés a feladata, illetve a játékkal kapcsolatos információk tárolása és a mentés valamint a betöltés biztosítása.
* Az aknaTable egy érvényes játékteret biztosít amelyben nincsenek inkonzisztens adatok. A játék tér alapvetően 6x6-os vagy 10x10-es vagy éppen 16x16-os a kezdeti beállítás függvényében. Minden mezőről el van tárolva, hogy hány szomszédos akna van melette, valamint, hogy bomba-e vagy nem, illetve, hogy fel van-e fedve vagy nem. Az adott mező értékének a lekérdezését a GetValue függvénnyel tehetjük meg. Azt, hogy egy mező akna-e azt a Getbomb függvénnyel tehetjük meg. Azt, hogy egy mező már nyitott-e GetFelnyit függvénnyel kapjuk meg.
* A hosszú távú adattárolás lehetőségeit az IsaknaDataAccess interfész adja meg, amely lehetőséget ad a tábla betöltésére (LoadAsync), valamintmentésére (SaveAsync).
* Az interfészt szöveges fájl alapú adatkezelésre a aknaFileDataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat aknaDataException kivétel jelzi.
* A program lehetővé teszi az adatok szöveges fájlként történő mentését és betöltését, ahol az adatokat txt kiterjesztésű fájlokba mentheted. Ezen fájlok segítségével az aktuális adatállapotot bármikor visszaállíthatod, vagy éppen elmentheted az aktuális állapotot.
* A fájl első sora megadja a tábla méretét, majd az utána lévő négy szám a két játékos elhelyezkedését adja meg. Az első kordináta mindkét esetben az x kordináta. Majd az utána lévő 5. szám azt jelzi, hogy melyik játékos jön. Ezután a sorok száma által meghatározott méretű 3 db mátrix kerül kiiratásra. Az első mátrix a mező környezetében lévő aknák számát iratja ki mezőnként. A második mátrixon vannak 0-ával jelölve az aknák. A harmadikban 0 vagy 1 kiírásval van reprezentálva egy-egy mező felfedettsége.

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

Model

* A model lényegi részét az aknaGameModel végzi ami lehetsőéget ad új játék létrehozására, mezők felfedésére és a Gameover eseményre. Eltárolja, hogy hol tartózkodnak a játákosok és, hogy melyik játékos jön.
* A mezők felfedését a Mezofelfed esemény végzi. Az esemény argumentuma (FieldEventArgs) tárolja a megváltozott mező pozícióját és színét.
* A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (LoadAsync) és mentésre (SaveAsync).

A képen szöveg, képernyőkép, menü, sárga látható

Automatikusan generált leírás

NézetModel

* A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) osztályt.
* A nézetetmodel feladatait a VadászatViewModel osztály látja el. osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, mentéséhez. A parancsokhoz eseményeket kötünk, amelyek a parancs lefutását jelzik a vezérlőnek. A nézetmodell tárolja a modell egy hivatkozását (\_model), de csupán információkat kér le tőle, illetve a játékpálya méretét szabályozza. Direkt nem avatkozik a játék futtatásába.
* A játékmező számára egy külön mezőt biztosítunk (Field), amely eltárolja a pozíciót, szöveget, láthatóságát. A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (Fields).

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

Nézet:

* A nézet csak egy képernyőt tartalmaz, a MainWindow osztályt. A nézet egy rácsban tárolja a játékmezőt, a menüt és a státuszsort. A játékmező egy ItemsControl vezérlő, ahol dinamikusan felépítünk egy rácsot (UniformGrid), amely gombokból áll. Minden adatot adatkötéssel kapcsolunk a felülethez, továbbá azon keresztül szabályozzuk a gombok színét is.
* A fájlnév bekérését betöltéskor és mentéskor, valamint a figyelmeztető üzenetek megjelenését beépített dialógusablakok segítségével végezzük.

Környezet

* Az App osztály feladata az egyes rétegek példányosítása (App\_Startup), összekötése, a nézetmodell, valamint a modell eseményeinek lekezelése, és ezáltal a játék, az adatkezelés, valamint a nézetek szabályozása.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

Tesztelés

* A modell funkcionalitása egységtesztek segítségével lett ellenőrizve az AknaGameModelTest osztályban.
* Az alábbi tesztek kerültek megvalósításra:
  + Megfelelőhelyre lépés esetén a szereplő kordinátája változik és a felfedezett mezők száma is csökken.
  + Megnezzük, hogy nem törtenik semmi ha rossz helyre lep valaki.
  + Megnézzük, hogy a két szereplő cserélgeti egymásközött a lépések sorrendjét.
  + Megnézzük, hogy a bomba közelében 1 mező száma.
  + Megnézzük, hogy a bombára kattintva véget ér a játék.