Beadandó feladat dokumentáció

Készítette: Nick Mónika (UHH5KS)

E-mail: nickmonkavera@gmail.com

Feladat:

22. Malom

Készítsünk programot, amellyel a következő két személyes játékot játszhatjuk. A malomjátékban két játékos egy 24 mezőből álló speciális játéktáblán játszhatja, ahol a mezők három egymásba helyezett négyzetben helyezkednek (mindegyikben 8, a sarkoknál és a felezőpontoknál), melyek a felezőpontok mentén össze vannak kötve. Kezdetben a tábla üres, és felváltva helyezhetik el rajta bábuikat az üres mezőkre. Az elhelyezés után a játékosok felváltva mozgathatják bábuikat a szomszédos (összekötött) mezőkre. Amennyiben egy játékos nem tud mozgatni, akkor passzolhat a másik játékosnak. Ha valakinek sikerül 3 egymás melletti mezőt elfoglalnia (azaz malmot alakít ki, rakodás, vagy mozgatás közben), akkor leveheti az ellenfél egy általa megjelölt bábuját (kivéve, ha az egy malom része). Az a játékos veszít, akinek először megy 3 alá a bábuk száma a mozgatási fázis alatt. A program biztosítson lehetőséget új játék kezdésére, mentésére és betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

Elemzés:

• A játékban három fázis van: elhelyezés, mozgatás, ugrálás. A program indításkor elhelyezéssel kezdünk, és automatikusan új játékot indul.

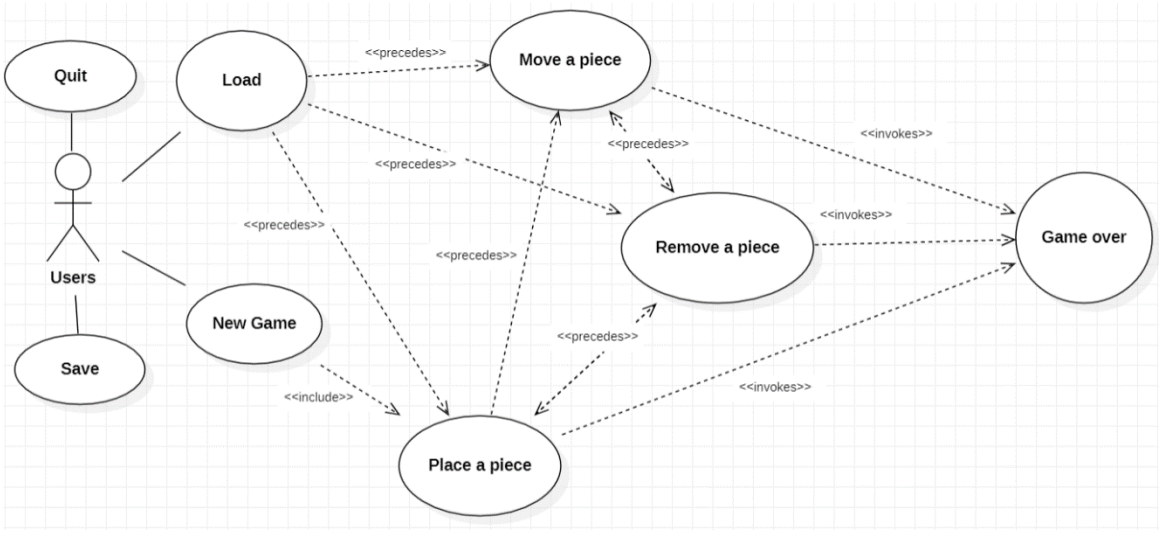
• A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.

• Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (Load Game, New Game, Save, Exit). Az ablak alján megjelenítünk egy státuszsort, amely azt jelzi, hogy ki jön, vagy azt, ha törlés következik, vagy, azt is, ha nem tudunk a kiválasztott mezőre lépni, vagy onnan máshova lépni, vagy törölni.

• A játéktáblát egy panel és azon 24 nyomógomb reprezentálja. A nyomógomb egérkattintás hatására megváltoztathatja a színét vagy nem. Attól függ, hogy az adott játékos helyezhet ide bábut, vagy elmozgathatja-e az adott mezőről.

• A játék kiírja a státuszsorba, amikor vége a játéknak és a bábukat már ilyenkor nem lehet mozgatni. Szintén dialógusablakokkal végezzük el a mentést, illetve betöltést, a fájlneveket a felhasználó adja meg.

Use Case Diagramm:



Tervezés:

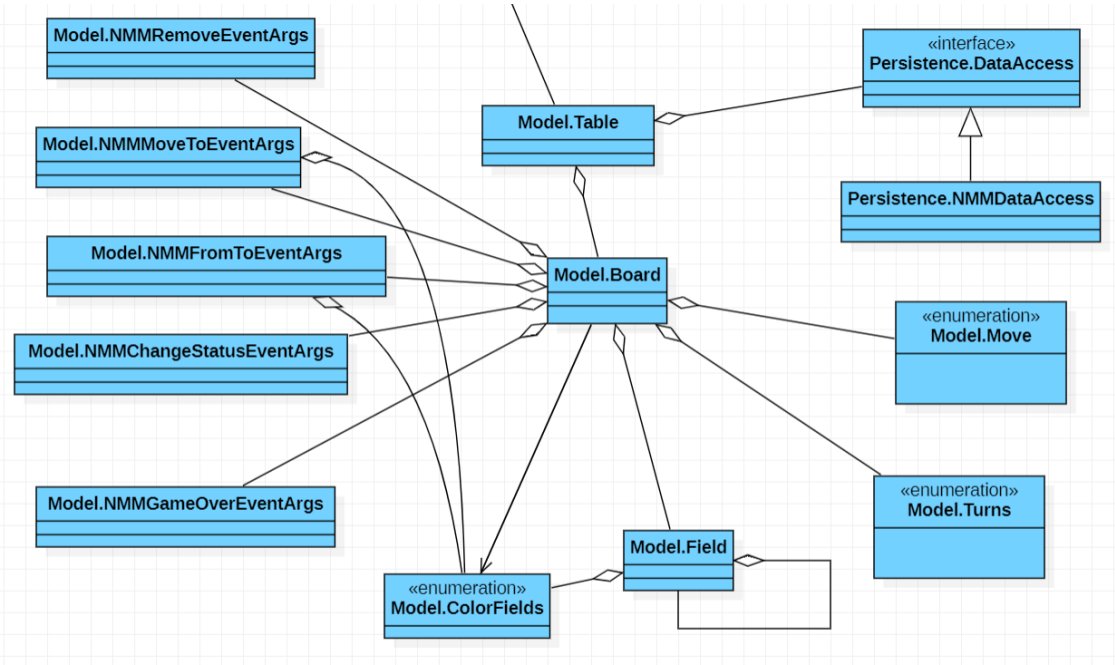
Programszerkezet:

• A programot MVVM architektúrában valósíjuk meg. A megjelenítés a NMMView, a megjelenítés modellje a NMMView.ViewModel, a modell a ModelNineMenMorris.Model, míg a perzisztencia a ModelNineMenMorris.Persistence névtérben helyezkedik el.

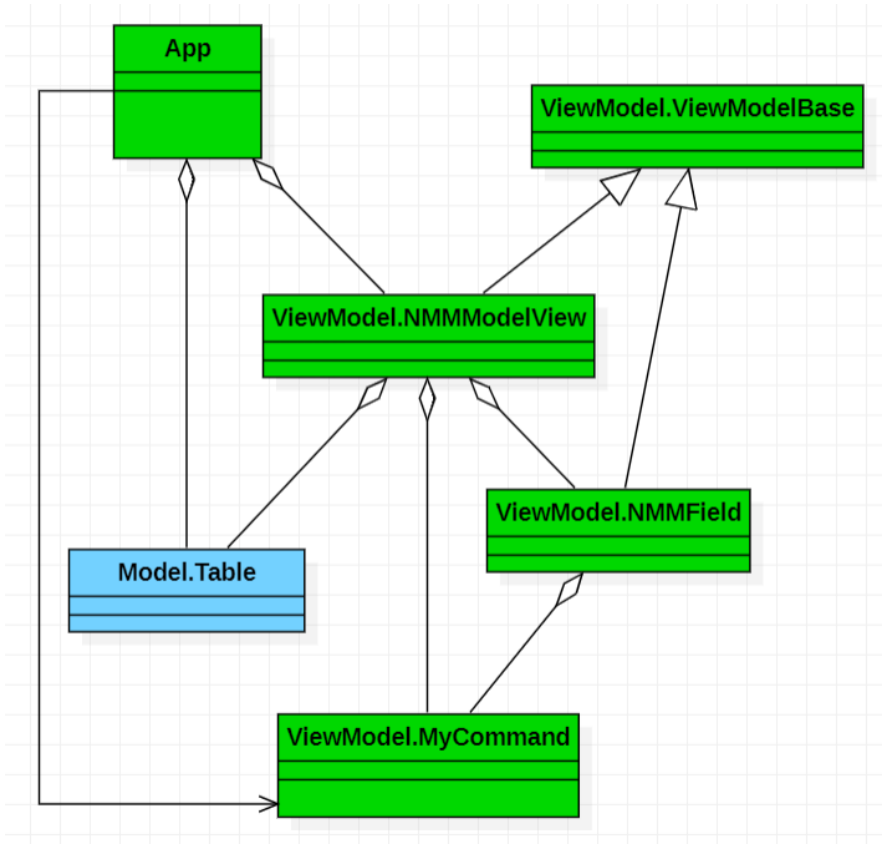
• A program szerkezetét két projektre osztjuk implementációs megfontolásból: a Persistence és Model csomagok a program felületfüggetlen projektjében, míg a NMMView csomag a WPF függő projektjében kap helyet.

Osztályok kapcsolatai:

Model:



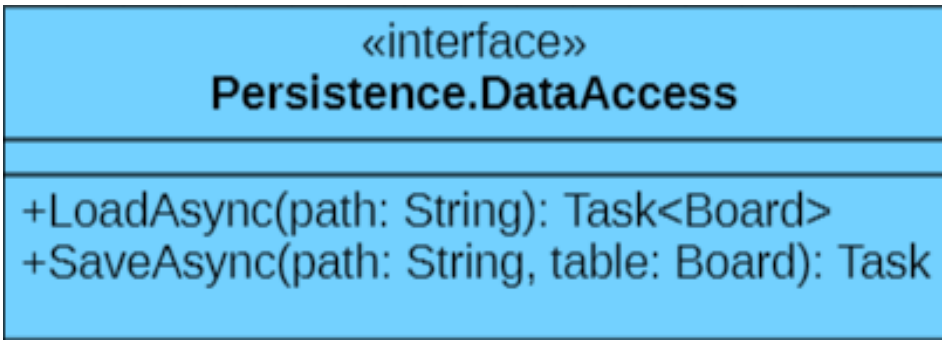
View:



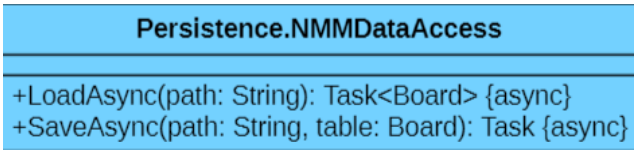
Perzisztencia:

• Az adatkezelés feladata a malom táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.

• A hosszú távú adattárolás lehetőségeit az DataAccess 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 NMMDataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a NMMDataException kivétel jelzi.

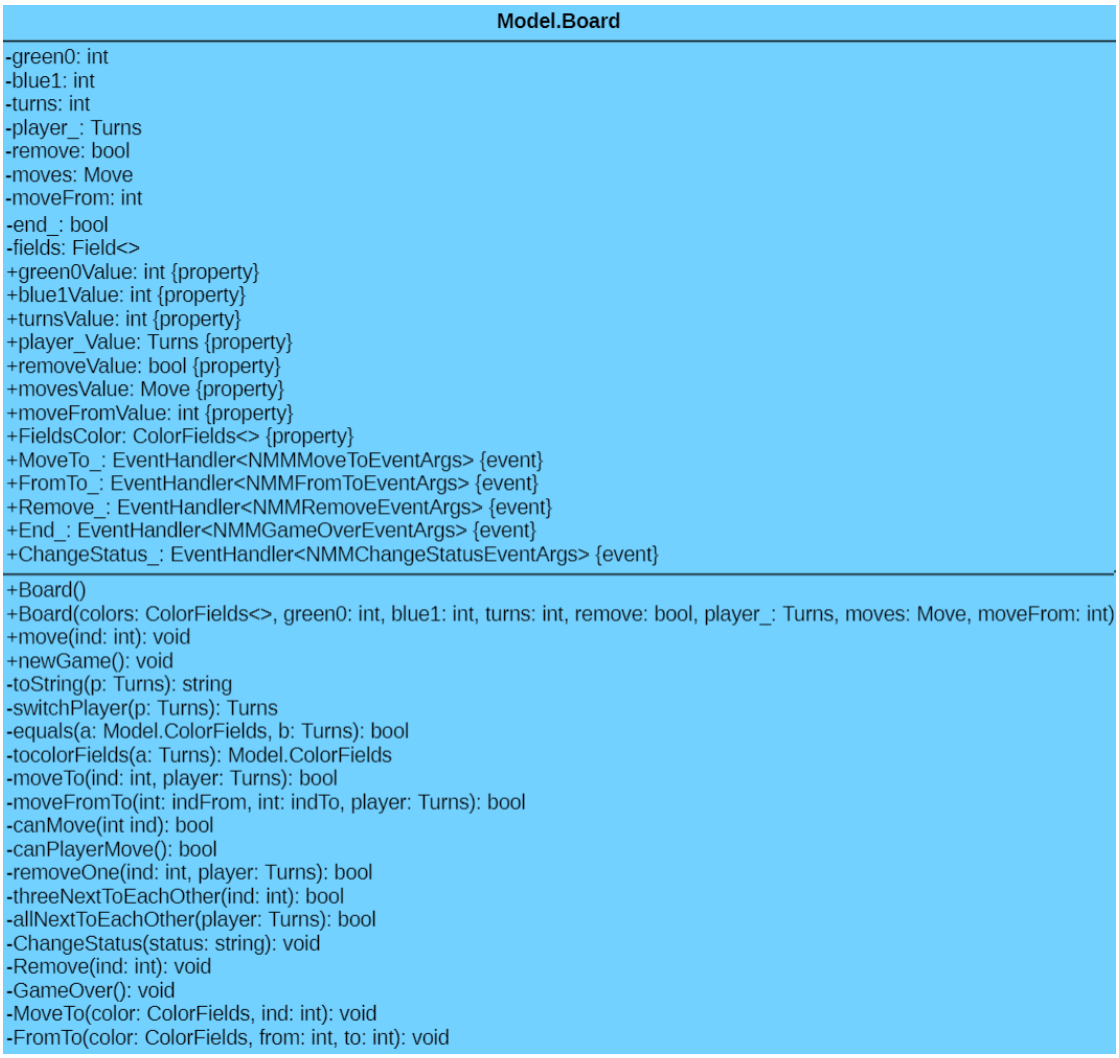


• A program az adatokat szöveges fájlként tudja eltárolni, melyek az txt kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.

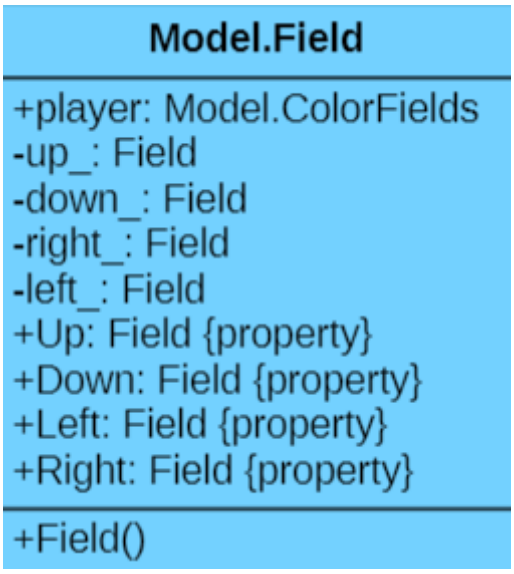
• A fájl első sora megadja a tábla méretét, hogy hol tartunk az elhelyezés fázisban, mennyi kék bábu van, mennyi zöld bábu van, ki következik, törlés következik-e, egy bábu kiválasztása vagy mozgatása következik- e, és melyik mezőről kell mozgatni. A második sor az összes mező színét jelzi (0-Green, 1-Blue, 2-Transparent).

Modell:

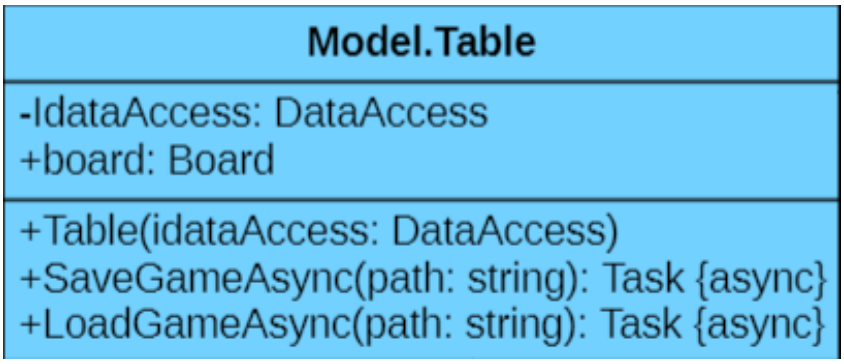
• A Board osztály egy malom táblát biztosít, ami 24 mezőből áll, és a mezők a szomszédjaikra mutatnak. Az osztályban a játékosok lépései és az adott játékfázis szerint változnak a mező színei. Az osztály számolja a bábuk számát, és azt is, hogy a játéknak mikor van vége. Egy mezőre kattintáskor mindig a move függvényt kell meghívni. Új játékkezdés esetén meg a newGame-t. moveTo és moveFromTo függvények mozgatják a bábukat. threeNextToEachOther ellenőrzi, ha malom van, allNextToEachOther pedig, ha egy játékos összes bábuja malomban van. canMove, hogy tud-e a bábu mozogni, canPlayerMove, hogy tud-e a játékos összes bábuja mozogni. removeOne törli a bábut, ha tudja. end véget vet a játéknak.



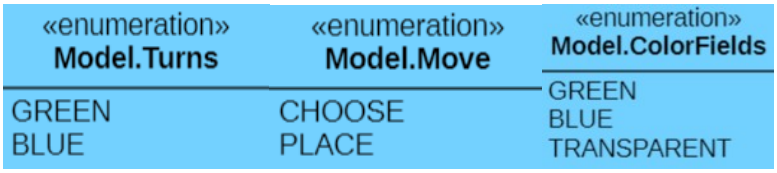
• Field osztály egy mezőt jelez, ami rámutat a szomszédjaira, és a saját színét tárolja.



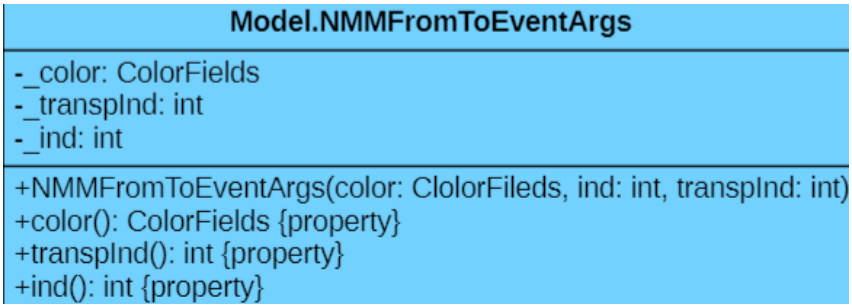
• A Table osztály tartalmaz egy Board osztályt, így betöltés (LoadGameAsync) és mentés (SaveGameAsync) esetén le tudja cserélni a malomtáblát.

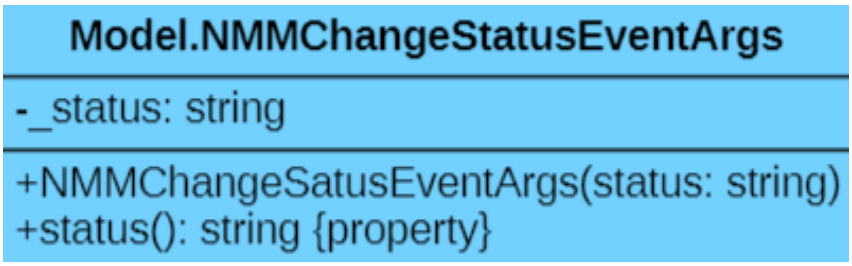


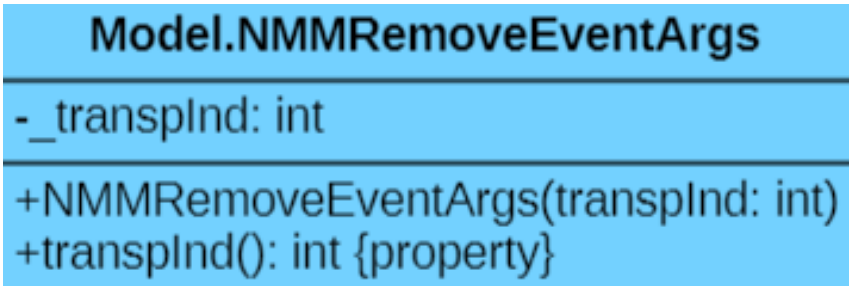
• A modellben vannak felsorolók is, a színek beállításához és a mozgatáshoz.

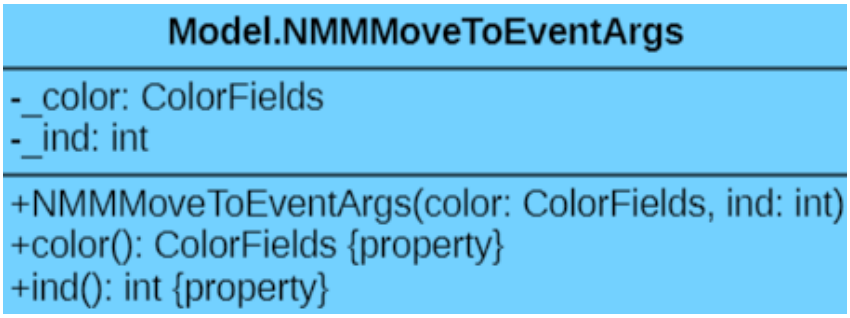


• A Board()-nak 5 eseménye van: MoveTo_, FromTo_, Remove_, ChangeStatus_ és End_. Ezekre tud a nézet feliratkozni.



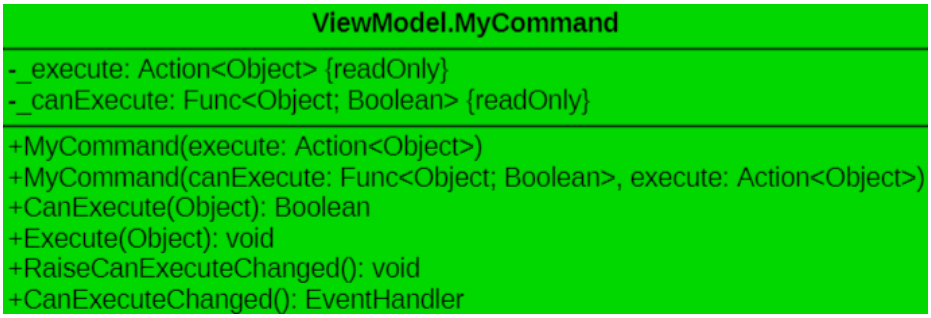


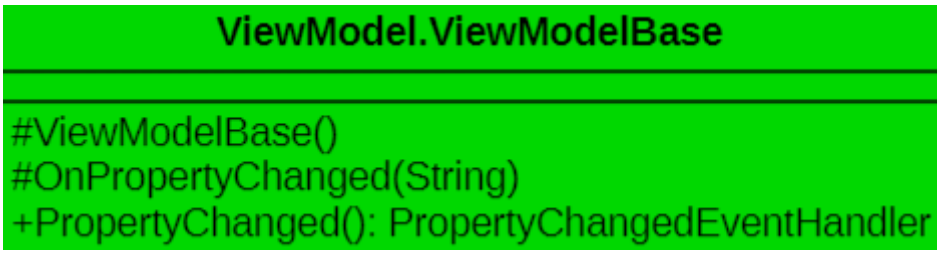




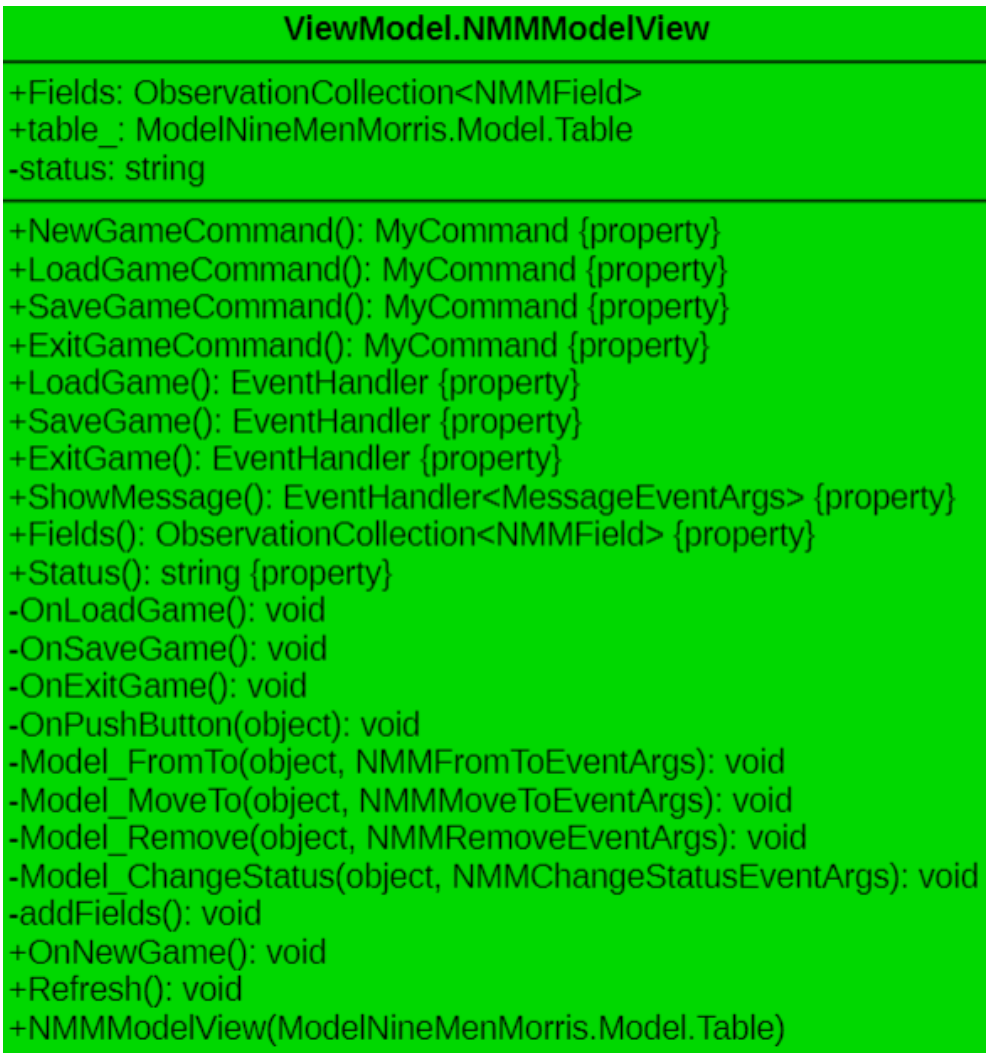
Nézetmodell:

• A nézetmodell megvalósításához felhasználunk egy általános utasítás (MyCommand), valamint egy ős változásjelző (ViewModelBase) osztályt.

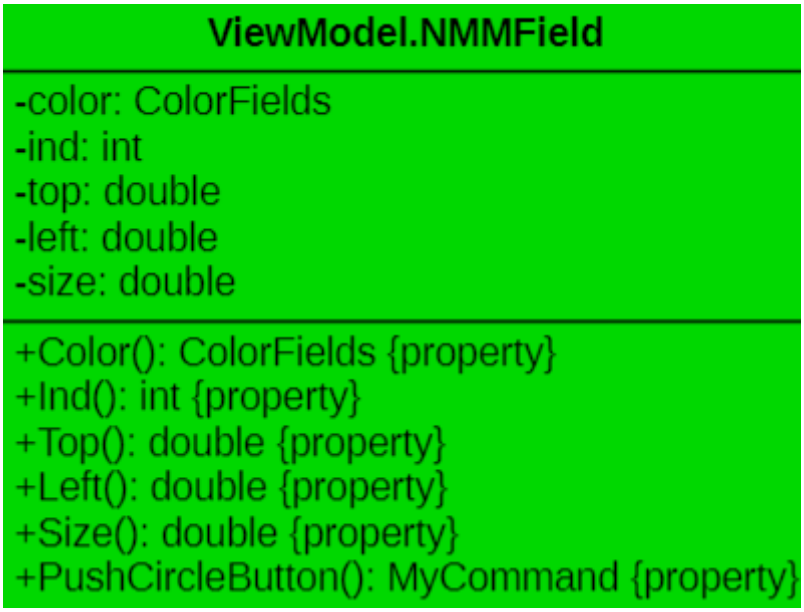




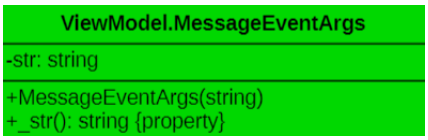
• A nézetmodell feladatait a NMMModelView osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, mentéséhez, valamint a kilépéshez. A parancsokhoz eseményeket kötünk, amelyek a parancs lefutását jelzik a vezérlőnek. A nézetmodell tárolja a modell egy hivatkozását (table_), de csupán információkat kér le tőle. Direkt nem avatkozik a játék futtatásába.



• A játékmező számára egy külön mezőt biztosítunk (NMMField), amely eltárolja a pozíciót, színét, méretét, indexét és parancsát (PushCircleButton). A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (Fields).



• Üzenetek küldésére használhatja a nézetmodell a MessegeEventArgs-ot.

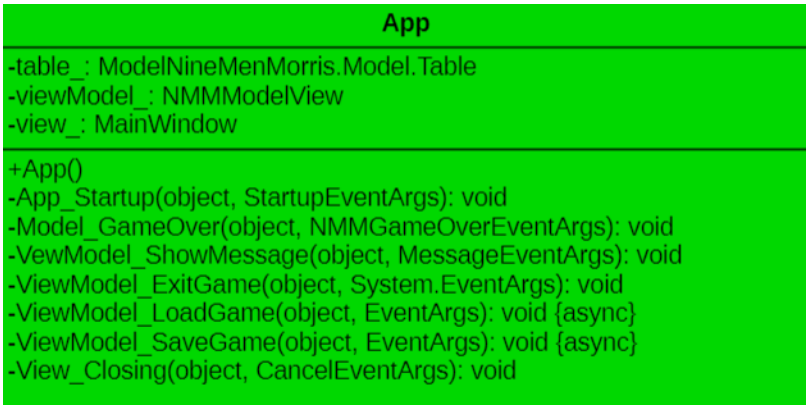


Nézet:

• 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 Canvas-t, amelyben gombok vannak. 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.

• 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.



Tesztelés:

A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a TestNMM osztályban.

• Az alábbi tesztesetek kerültek megvalósításra és következőket ellenőrzik le:

1. TestConstruct(): Létrejön-e a tábla?
2. TestPlaceOne(): Le tudunk-e helyezni egy bábut?
3. TestTurns(): Az első fázis lépései csökkenek-e?
4. TestSwitchPlayer(): Váltunk-e a két játékos között?
5. TestCannotPlace(): Nem tudunk lehelyezni bábut?
6. TestRemovePhase(): Törlés következik-e?
7. TestRemove(): Kitöröltük-e a bábut?
8. TestRemoveDecrease(): Csökkent-e a bábuk száma?
9. TestRemovingIfAllNextToEachOther(): Töröl-e, akkor is, ha minden bábu 3-ast alkot?
10. TestSecondPhaseChoosing(): A második fázisban választottunk-e bábut?
11. TestSecondPhaseMoving(): A második fázisban mozgattunk-e bábut?
12. TestSecondPhaseCannotMoveToTransparent(): Nem tudunk mozgatni, mert messze van.
13. TestSecondPhaseCannotMoveToOwnField(): Nem tudunk mozgatni saját mezőre.
14. TestSecondPhaseCannotMoveToOtherPlayersField(): Nem tudunk mozgatni másik játékos mezőjére.
15. TestHaveToPass(): Passzolnia kell-e a játékosnak?
16. TestThirdPhase(): Lehet-e ugrálni a 3. fázisban?
17. TestEndGame(): Vége van-e a játéknak?
18. TestNewGame(): Tudunk-e új játékot kezdeni?
19. TestLoad(): Meghívódik-e a Load?
20. TestLoadWrongFileFormat(): Rossz fájlformátum esetén kapunk-e exceptiont?
21. TestBoardIndex(): Fel tudunk-e helyezni a táblára korongokat?
22. TestBoardOutOfIndexing(): Ha nagyobb számot adunk meg, mint a tábla mezőinek száma, akkor kapunk-e exceptiont?
23. TestBoardOutOfIndexing2(): Ha kisebb számot adunk meg, mint a tábla mezőinek száma, akkor kapunk-e exceptiont?