

# Beadandó feladat dokumentáció

Készítette: Nick Mónika (UHH5KS)

E-mail: nickmonkavera@gmail.com

## Feladat:

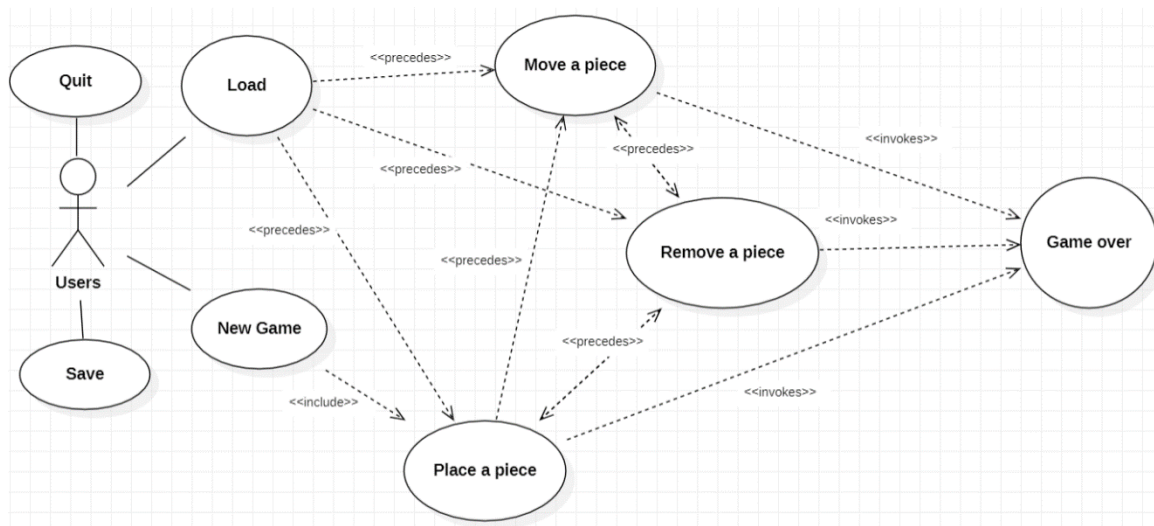
### 22. Malom

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, mentésre és betöltésre. 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 egérekattintás 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 státuszsortba, 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:



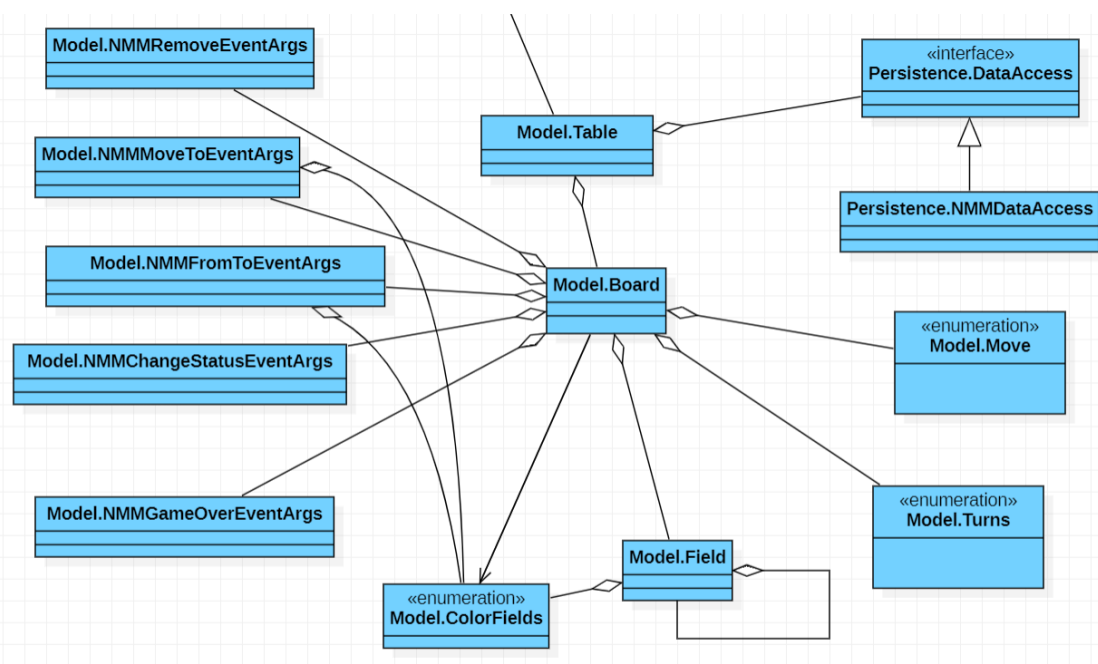
## Tervezés:

### Programszerkezet:

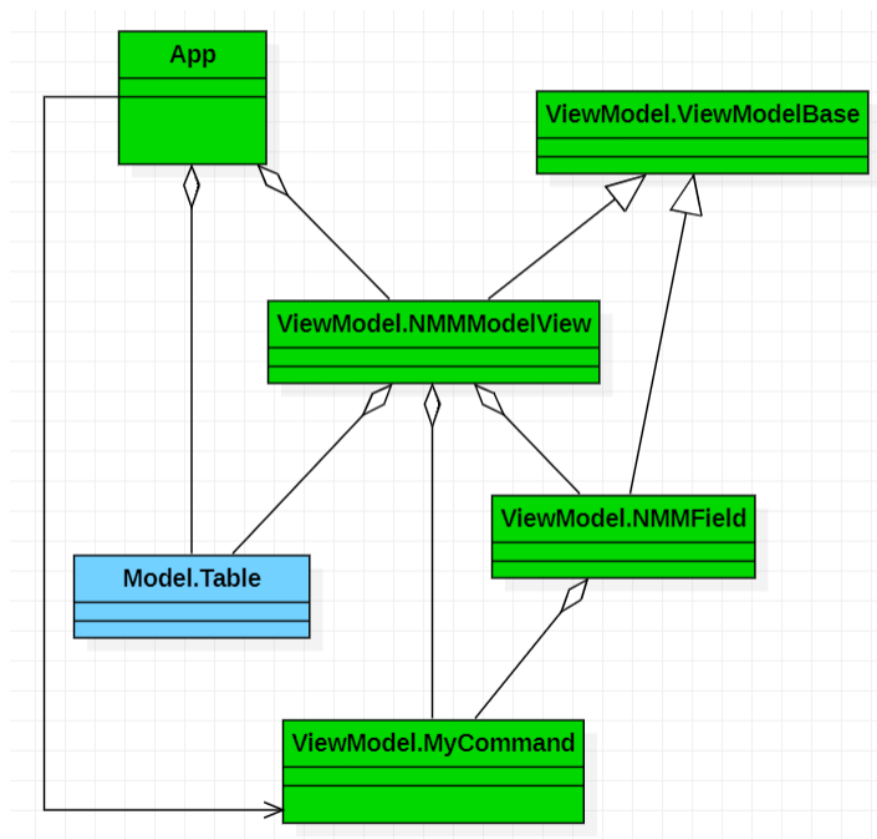
- A programot MVVM architektúrában valósítjuk meg. A megjelenítés a NMMView, a megjelenítés modellje a NMMView.ViewModel, a modell a ModelNineMenMorris.Model, míg a perzisztencia a ModelNineMenMorris.Persistence névtérben helyezkedik el.
- A program szerkezetét két projektre osztjuk implementációs megfontolásból: a Persistence és Model csomagok a program felületfüggetlen projektjében, míg a NMMView csomag a WPF függő projektjében kap helyet.

## Osztályok kapcsolatai:

### Model:

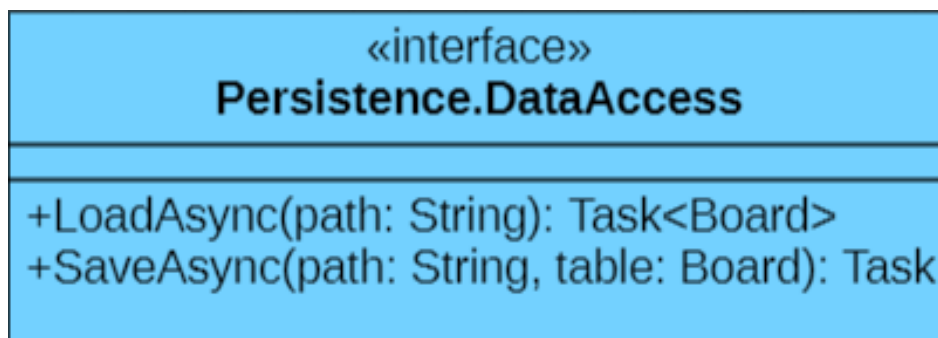


## View:

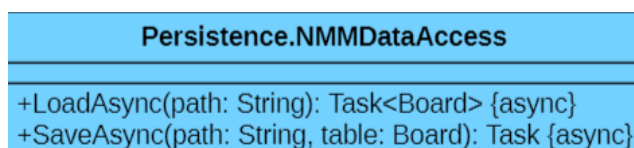


### Perzisztencia:

- 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 NMMDDataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a NMMDDataException 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 bábú van, mennyi zöld bábú van, ki következik, törlés következik-e, egy bábú 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 moveToFrom függvények mozgatják a bábukat.

threeNextToEachOther ellenőrzi, ha malom van, allNextToEachOther pedig, ha egy játékos összes bábujja malomban van. canMove, hogy tud-e a bábú mozogni, canPlayerMove, hogy tud-e a játékos összes bábujja mozogni. removeOne törli a bábút, ha tudja. end véget vet a játéknak.

Model.Board
-green0: int -blue1: int -turns: int -player_: Turns -remove: bool -moves: Move -moveFrom: int -end_: bool -fields: Field<> +green0Value: int {property} +blue1Value: int {property} +turnsValue: int {property} +player_ Value: Turns {property} +removeValue: bool {property} +movesValue: Move {property} +moveFromValue: int {property} +FieldsColor: ColorFields<> {property} +MoveTo_: EventHandler<NMMMMoveToEventArgs> {event} +FromTo_: EventHandler<NMMFromToEventArgs> {event} +Remove_: EventHandler<NMMRemoveEventArgs> {event} +End_: EventHandler<NMMGameOverEventArgs> {event} +ChangeStatus_: EventHandler<NMMChangeStatusEventArgs> {event}
+Board() +Board(colors: ColorFields<>, green0: int, blue1: int, turns: int, remove: bool, player_: Turns, moves: Move, moveFrom: int) +move(ind: int): void +newGame(): void -toString(p: Turns): string -switchPlayer(p: Turns): Turns -equals(a: Model.ColorFields, b: Turns): bool -tocolorFields(a: Turns): Model.ColorFields -moveTo(ind: int, player: Turns): bool -moveFromTo(int: indFrom, int: indTo, player: Turns): bool -canMove(ind: int): bool -canPlayerMove(): bool -removeOne(ind: int, player: Turns): bool -threeNextToEachOther(ind: int): bool -allNextToEachOther(player: Turns): bool -ChangeStatus(status: string): void -Remove(ind: int): void -GameOver(): void -MoveTo(color: ColorFields, ind: int): void -FromTo(color: ColorFields, from: int, to: int): void

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

Model.Field
+player: Model.ColorFields
-up_: Field
-down_: Field
-right_: Field
-left_: Field
+Up: Field {property}
+Down: Field {property}
+Left: Field {property}
+Right: Field {property}
+Field()

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

Model.Table
-IdataAccess: DataAccess
+board: Board
+Table(idataAccess: DataAccess)
+SaveGameAsync(path: string): Task {async}
+LoadGameAsync(path: string): Task {async}

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

«enumeration» Model.Turns	«enumeration» Model.Move	«enumeration» Model.ColorFields
GREEN BLUE	CHOOSE PLACE	GREEN BLUE TRANSPARENT

- A Board()-nak 5 eseménye van: MoveTo\_, FromTo\_, Remove\_, ChangeStatus\_ és End\_. Ezekre tud a nézet feliratkozni.

Model.NMMFromToEventArgs
- _color: ColorFields - _transplnd: int - _ind: int
+NMMFromToEventArgs(color: ClolorFiledS, ind: int, transplnd: int) +color(): ColorFields {property} +transplnd(): int {property} +ind(): int {property}

Model.NMMGameOverEventArgs
- _isGameOver: bool
+NMMGameOverEventArgs(isGameOver: bool) +isGameOver(): bool {property}

Model.NMMChangeStatusEventArgs
- _status: string
+NMMChangeSatusEventArgs(status: string) +status(): string {property}

Model.NMMRemoveEventArgs
- _transplnd: int
+NMMRemoveEventArgs(transplnd: int) +transplnd(): int {property}

Model.NMMMMoveToEventArgs
- _color: ColorFields - _ind: int
+NMMMMoveToEventArgs(color: ColorFields, ind: int) +color(): ColorFields {property} +ind(): int {property}

Nézetmodell:

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

ViewModel.MyCommand
-_execute: Action<Object> {readOnly} -_canExecute: Func<Object; Boolean> {readOnly}
+MyCommand(execute: Action<Object>) +MyCommand(canExecute: Func<Object; Boolean>, execute: Action<Object>) +CanExecute(Object): Boolean +Execute(Object): void +RaiseCanExecuteChanged(): void +CanExecuteChanged(): EventHandler

ViewModel.ViewModelBase
#ViewModelBase() #OnPropertyChanged(String) +PropertyChanged(): PropertyChangedEventHandler

- 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.

ViewModel.NMMModelView
+Fields: ObservableCollection<NMMField> +table_: ModelNineMenMorris.Model.Table -status: string
+NewGameCommand(): MyCommand {property} +LoadGameCommand(): MyCommand {property} +SaveGameCommand(): MyCommand {property} +ExitGameCommand(): MyCommand {property} +LoadGame(): EventHandler {property} +SaveGame(): EventHandler {property} +ExitGame(): EventHandler {property} +ShowMessage(): EventHandler<MessageEventArgs> {property} +Fields(): ObservableCollection<NMMField> {property} +Status(): string {property} -OnLoadGame(): void -OnSaveGame(): void -OnExitGame(): void -OnPushButton(object): void -Model_FromTo(object, NMMFromToEventArgs): void -Model_MoveTo(object, NMMMoveToEventArgs): void -Model_Remove(object, NMMRemoveEventArgs): void -Model_ChangeStatus(object, NMMChangeStatusEventArgs): void -addFields(): void +OnNewGame(): void +Refresh(): void +NMMModelView(ModelNineMenMorris.Model.Table)

- A játéklemező 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).

ViewModel.NMMField
-color: ColorFields -ind: int -top: double -left: double -size: double
+Color(): ColorFields {property} +Ind(): int {property} +Top(): double {property} +Left(): double {property} +Size(): double {property} +PushCircleButton(): MyCommand {property}

- Üzenetek küldésére használhatja a nézetmodell a MessegeEventArgs-ot.

ViewModel.MessageEventArgs
-str: string
+MessageEventArgs(string) + _str(): string {property}

### Nézet:

- A nézet csak egy képernyőt tartalmaz, a MainWindow osztályt. A nézet egy rácsban tárolja a játéklemezőt, a menüt és a státuszsort. A játéklemező 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 fájlnevé 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.

App
-table_: ModelNineMenMorris.Model.Table -viewModel_: NMMModelView -view_: MainWindow
+App() -App_Startup(object, StartupEventArgs): void -Model_GameOver(object, NMMGameOverEventArgs): void -VewModel_ShowMessage(object, MessageEventArgs): void -ViewModel_ExitGame(object, System.EventArgs): void -ViewModel_LoadGame(object, EventArgs): void {async} -ViewModel_SaveGame(object, EventArgs): void {async} -View_Closing(object, CancelEventArgs): void



# 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 csökkennek-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 báruk 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?