3.Beadandó feladat dokumentáció

Készítette: Ferencz Tamás, G0820E

E-mail: ferencztamas@student.elte.hu

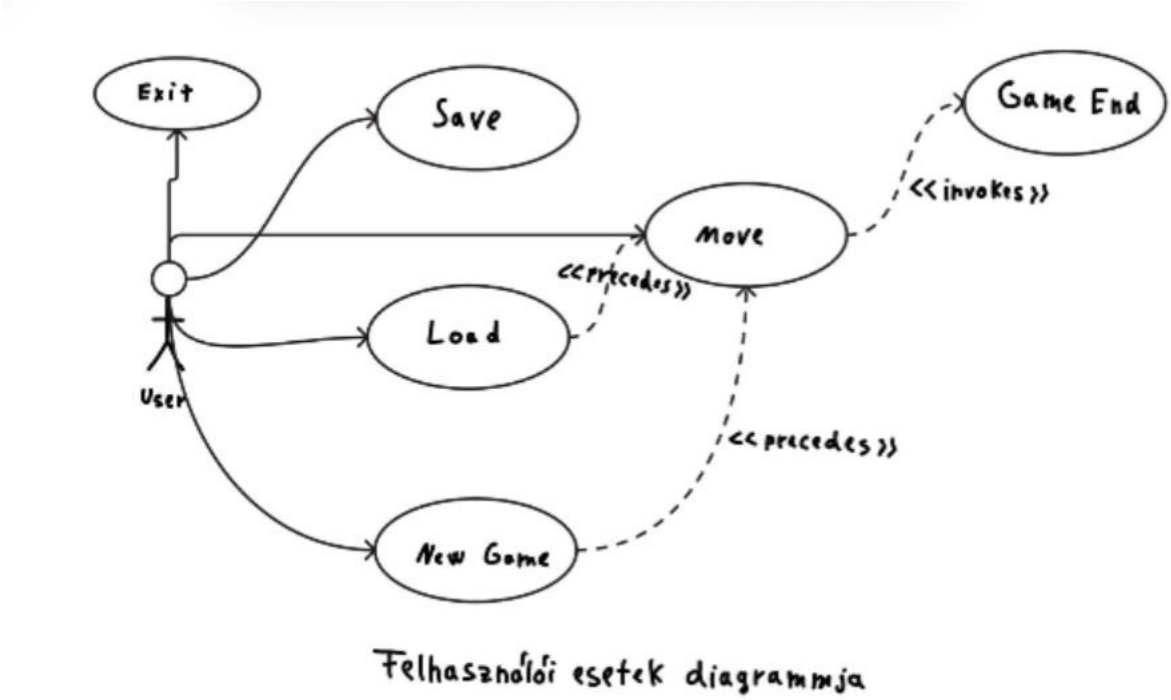
Feladat:

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.

Elemzés:

- A feladatot .NET Avalonia alkalmazásként, elsődlegesen asztali környezetre valósítjuk meg, amely MVVM (Model-View-ViewModel) architektúrára épül.
- A játék nézete (MainView) tartalmazza a játéktáblát, ahol a pozíciókat gombok reprezentálják. A tábla vonalait a Canvas elemen elhelyezett Line objektumok rajzolják ki. A jobb oldali panelen láthatóak a játék állapotinformációi (fázis, aktív játékos, bábuk száma), valamint a vezérlőgombok (Új játék, Mentés, Betöltés).
- Játék betöltésére és mentésére az operációs rendszer (Windows/Linux/macOS) natív fájlkezelő dialógusablakait használjuk (StorageProvider). A játék végeredményét felugró üzenetablakban (MessageBox) jelezzük.
-
- A játéktáblát egy 24 nyomógombokból álló rácsos szerkezet reprezentálja. A nyomógomb egérkattintás hatására a játékosnak zöld jelzésű, lehetséges szomszédos lépéseit mutatja.

- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (nyertes fél kiírtatása), amikor játék mentés, illetve betöltés történik.



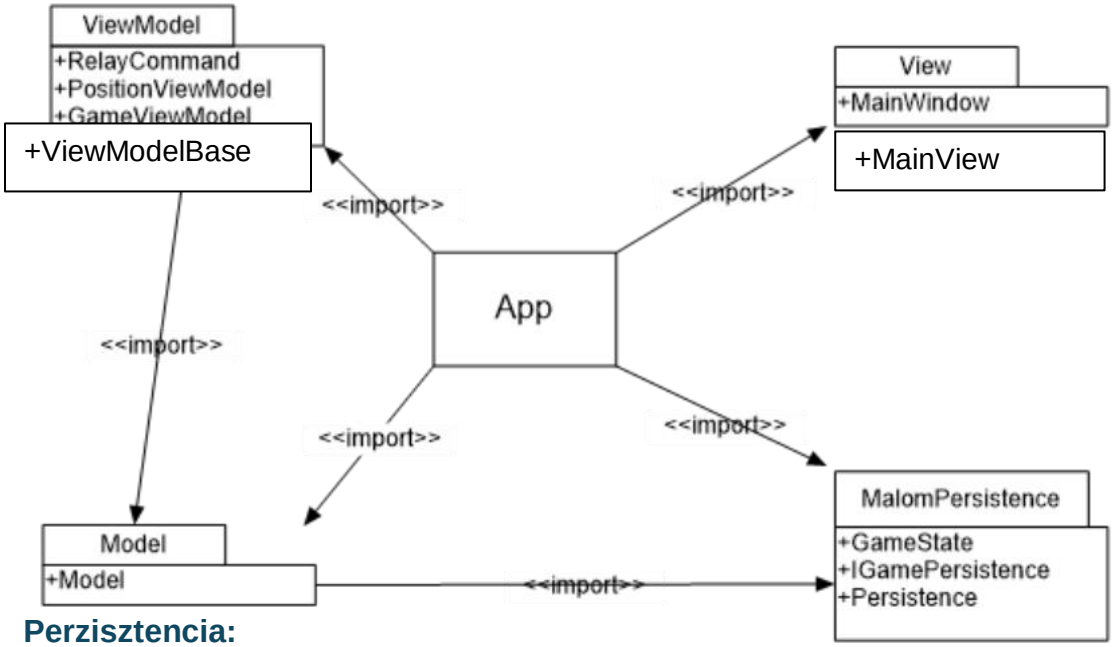
Felhasználói esetek diagrammja

Tervezés:

A szoftvert öt projektből építjük fel:

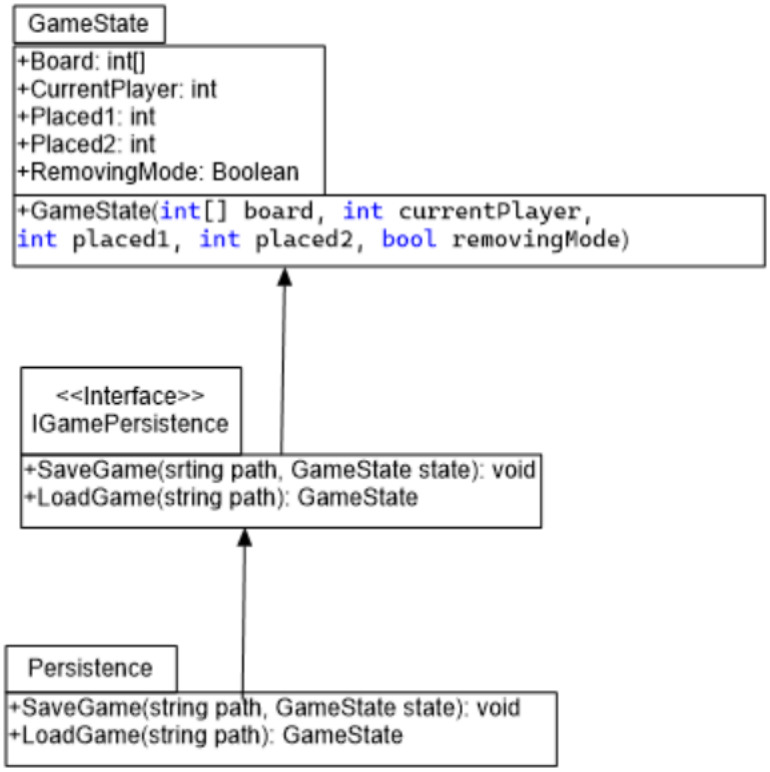
1. MalomModel: A tiszta játéklogikát és szabályrendszert tartalmazó osztálykönyvtár.
2. MalomPersistence: Az adatok tartós tárolásáért felelős réteg.
3. MalomAvalonia: A platformfüggetlen UI logika (View és ViewModel).
4. MalomAvalonia.Desktop: A futtatható asztali alkalmazás keretrendszere.
5. MalomTests: Az egységteszteket tartalmazó projekt.

A programot szigorú MVVM architektúrában valósítjuk meg, elválasztva a megjelenítést a logikától.



Perzisztencia:

- A Persistence réteg feladata a játékállás hosszú távú tárolása és visszatöltése. Az adatkezelés a Persistence osztályban történik, amely a játékhoz kapcsolódó adatokat szöveges fájlban tárolja.
  Ehhez a réteghez tartozik az IGameModelPersistence interfész is, amely az adatok mentéséhez és betöltéséhez szükséges műveleteket definiálja. Az interfészt a Persistence osztály valósítja meg.



Model réteg

A **Model** osztály tartalmazza a teljes játéklogikát.
Feladata a játék szabályainak betartatása, a lépések kezelése, valamint az állapotok tárolása.

A modell a következő fő adattagokat tartalmazza:

- int[] Board – a tábla 24 mezőjének állapota (0 = üres, 1 = játékos 1, 2 = játékos 2),
- int CurrentPlayer – a soron következő játékos azonosítója,
- int Placed1, int Placed2 – az eddig lehelyezett bábuk száma játékosonként,
- bool RemovingMode – jelzi, hogy az aktuális játékos millt (sor) alkotott-e, és el kell távolítania az ellenfél egyik bábuját.

A modell biztosítja a következő főbb funkciókat:

- **PlacePiece(int idx)** – új bábu lehelyezése a megadott mezőre,
- **MovePiece(int from, int to)** – bábu mozgatása, ha a játékos már minden bábuját lehelyezte,
- **RemovePiece(int idx)** – az ellenfél bábu eltávolítása, ha mill alakult ki,
- **FormsMillAt(int idx, int player)** – ellenőrzi, hogy létrejött-e egy új sor (mill),
- **Reset()** – új játék indítása alapállapotból,
- **SetState(...)** – betöltött játékállapot beállítása a fájlból.
- **PlayerPieceCount(...)** – visszaadja a játékos bábuinak a számát
- **PlayerHasMove(...)** – a játékos maradék lépését adja vissza, ha van
- **SwitchPlayer(...)** – a játékosok soron következését módosítja

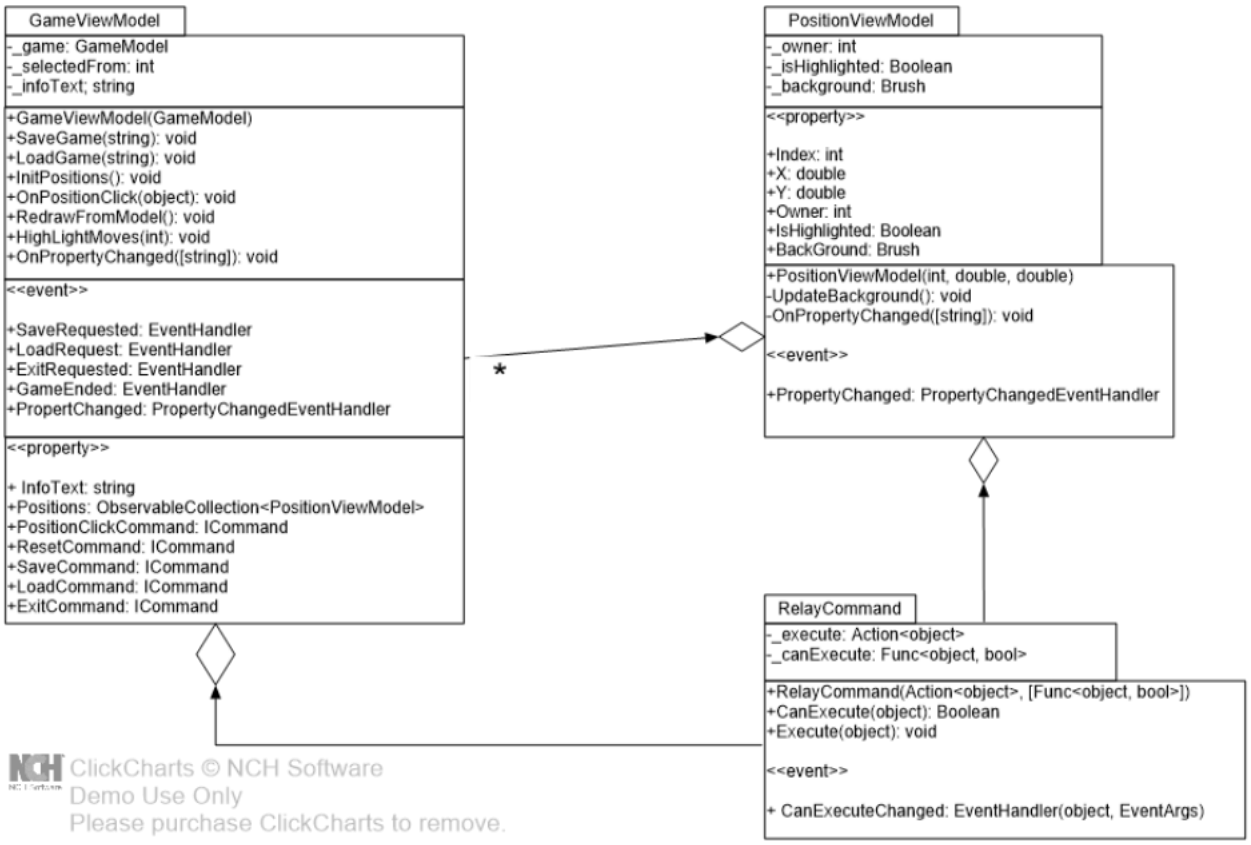
A logika betartja a malom szabályait:

- csak üres mezőre lehet bábut tenni,
- mozgatni csak szomszédos mezőre lehet (kivéve, ha a játékosnak 3 bábuja van – ekkor „repülhet”),
- ha mill alakul ki, az ellenfél egyik bábuja levehető, kivéve, ha minden bábuja millben van.
- Az alábbi diagramm a model csomag diagrammja.


Model
+Board: int[]
+CurrentPlayer: int
+Placed1: int
+Placed2: int
+RemovingMode: Boolean
- persistence: IGamePersistence
+MaxPieces: int
+MaxPieces: int
- mills: int[][]
-neighbors: int[][]
+SetState(GameState): void
+Reset(): void
+PlacePiece(int): Boolean
+RemovePiece(int): Boolean
+MovePiece(int,int): Boolean
+PlayerPieceCount(int): int
+PlayerHasMove(int): Boolean
+SwitchPlayer(): void
+OpponentHasNonMillPieces(): Boolean
+IsInMill(int): Boolean
+FormsMillAt(int, int): Boolean
+SaveGameModel(string): void
+LoadGameModel(string): GameState

ViewModel réteg

- A ViewModel kapcsolatot teremt a Model és a View között, parancsokkal és propertykkel adja át az adatokat a felületnek, miközben maga UI-független marad.
- Az alábbi ábra a nézetmodell osztálydiagrammja



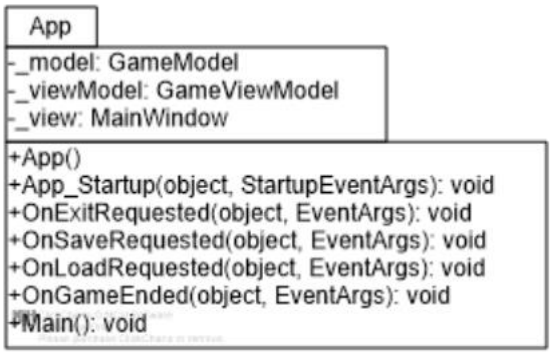
View réteg

- A View a felhasználóval való interakcióért felel, vizuálisan megjeleníti a ViewModelből érkező adatokat és továbbítja a parancsokat.

Vezérlés

- Az App osztály (App.axaml.cs) végzi a program inicializálását (OnFrameworkInitializationCompleted).
- Példányosítja a GameModel-t.
- Példányosítja a GameViewModel-t, átadva neki a modellt.

- Beállítja a MainWindow DataContext-ét.
- Feliratkozik a ViewModel eseményeire (pl. SaveRequested), hogy végrehajtsa az aszinkron fájlműveleteket a StorageProvider segítségével.



Tesztelés

- A **Malom játék modelljének** funkcionalitása egységtesztek segítségével került ellenőrzésre a MalomTests projektben.
- A tesztek célja a **játékmenet logikai helyességének** biztosítása: a bábu elhelyezés, mozgatás, levétel, malomképzés, valamint a játékosváltás és állapotkezelés helyes működésének ellenőrzése.

Az alábbi főbb tesztesetek kerültek megvalósításra:

- **Reset_ShouldInitializeGameState:** új játék indításakor az alapállapot ellenőrzése.
- **SetState_ShouldApplyCorrectValues:** a modell állapotának helyes visszaállítása betöltéskor.
- **PlacePiece_ShouldPlacePieceAndSwitchPlayer:** a bábu lerakás, játékosváltás és malomképzés ellenőrzése.
- **RemovePiece_ShouldRemoveOpponentPiece:** ellenfél bábujának levételének tesztelése különböző feltételek mellett.
- **ShouldAllowFlyingIfThreePiecesLeft:** bábumozgatás és „flying” szabály ellenőrzése.
- **PlayerHasMove és PlayerPieceCount tesztek:** a játékos mozgásképességének és bábu-számának helyes számítása.

- **FormsMillAt:** privát függvények tesztelése reflekszióval a malomképzés logikájának validálására.
- **OpponentHasNonMillPieces:** a játékosváltás és az ellenfél állapotának ellenőrzése.