## Készítette:

Név: **Kis Gergely Domonkos**

E-mail: vmt982@inf.elte.hu

## Feladat:

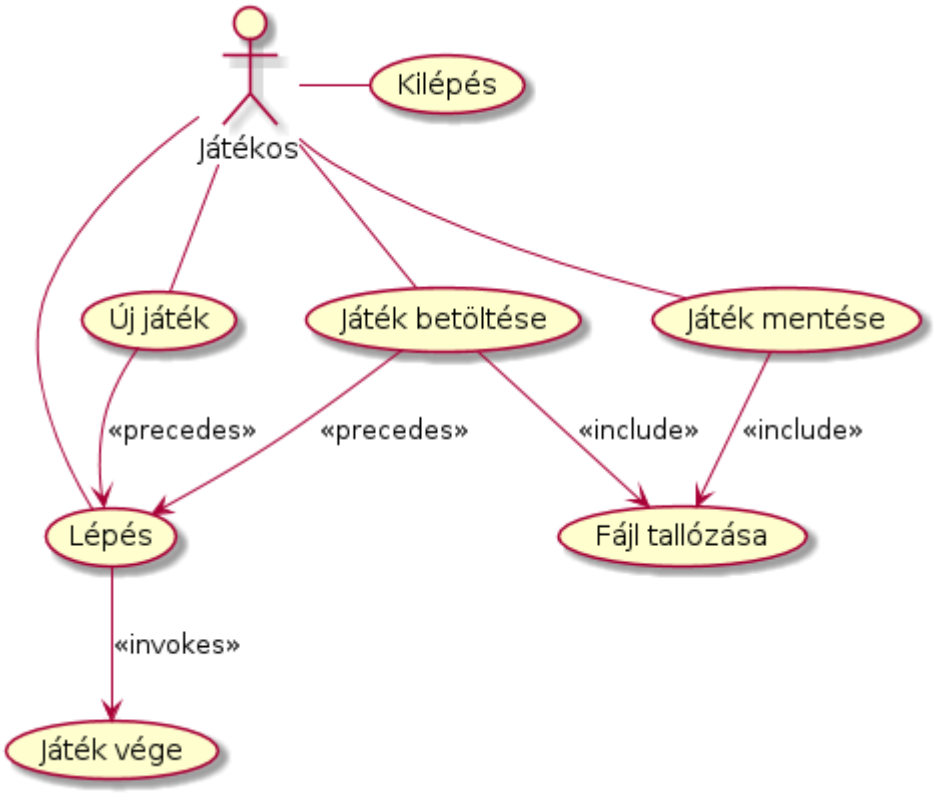
Készítsünk programot, amellyel az alábbi két személyesjátékot játszhatjuk. Adott egy $n \times n$ pontból álló játéktábla, amelyen a játékosok két szomszédos pont között vonalakat húzhatnak (vízszintesen, vagy függőlegesen). A játék célja, hogy a játékosok a húzogatással négyzetet tudjanak rajzolni (azaz ők húzzák be a negyedik vonalat, független attól, hogy az eddigieket melyikük húzta).Ilyen módon egyszerre akár két négyzet is elkészülhet. A játék addig tart, amíg lehet húzni vonalat a táblán. A játékosok felváltva húzhatnak egy-egy vonalat, de ha egy játékos berajzolt egy négyzetet, akkor ismét ő következik. A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (3×3, 5×5, 9×9), játék mentésére és betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött (ha nem döntetlen). Játék közben a vonalakat, illetve a négyzeteket színezze a játékos színére.

## Elemzés

- A játékot 3 táblamérettel lehet játszani (3x3, 5x5, 7x7), melyet a „New Game” gomb megnyomása után feljövő ablakban lehet beállítani. A program indulása után nem indul játék egészen addig amíg a felhasználó nem adja meg az új játék indításához szükséges adatokat (játékosok adatai és táblaméret)
- A feladatot ablakos alkalmazással valósítjuk meg Windows Presentation Foundation grafikus felülettel. Külön ablakot kap a játék és az új játék indítása.
- Az menüsorban elhelyezünk 3 menüpontot („New Game”, „Save Game” és „Load Game”) feliratokkal, melyek az új játék indításáért, játék mentésért, és játék betöltésért felelősek.
- A játék indulását követően megjelenik 2-2 címke, amely a játékosok neveit és pontszámait jeleníti meg
- A játéktábla egy canvas, amelyre Shape elemek (Ellipse,Line,Rectangle) vannak felhelyezve.

- Új vonalat úgy húzhatunk, ha rákattintunk egy kirajzolt bal egérgombbal és jobb egérgombbal egy másik szabályos pontra

- A játék végét követően egy dialógusablak jelzi a győztes játékost, vagy a döntetlent, ez követően új játék indul ugyanazokkal a beállításokkal
- A felhasználói esetek az 1. ábrán láthatóak



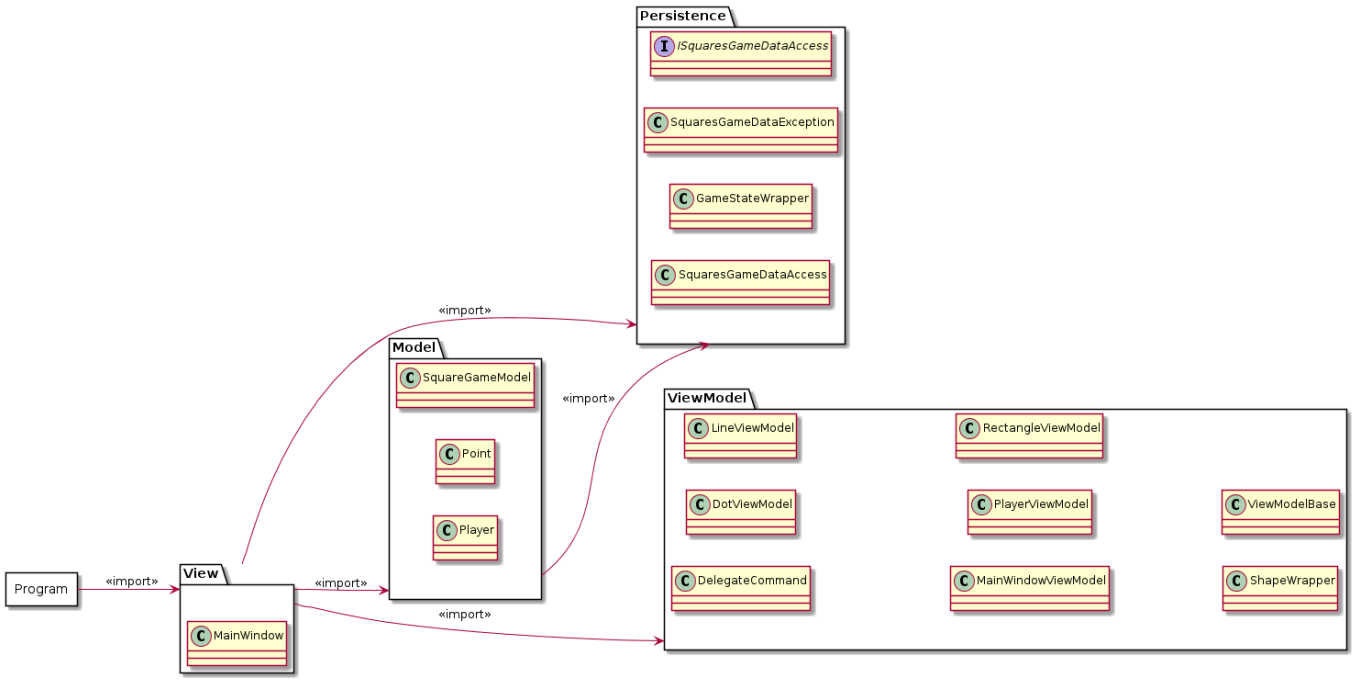

*1. ábra*

# Tervezés

## Programszerkezet

A programot MVVM architektúrában valósítjuk meg, 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 a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést.

*2. ábra*



## Perzisztencia

- Az adatkezelés feladata a játékállapot tárolása, valamint ennek a mentése, és az ebből való betöltés.
- A **GameStateWrapper** nevű osztály egy játékállapotot csomagoló osztály, amelyből a teljes játékállapot előállítható. Ez az alábbi adattagokat tartalmazza:

  - **PlayerOne** – Az első játékost leíró adattag, ez tárol további 3 mezőt (PlayerName, PlayerColor, Points) – (Játékosnév, Játékos szín, Pontok)
  - **PlayerTwo** – A második játékost leíró adattag hasonlóan az előzőhöz
  - **ActivePlayer** – Az aktív játékosra mutató referencia (egyike a PlayerOne-nak és a PlayerTwo-nak)
  - **Lines** – A behúzott vonalak végpontjait tároló lista
  - **Rectangles** – A kialakult négyzetek bal felső és jobb alsó végpontjait tároló lista
  - **RegisteredRectCount** – A kialakult négyzetek száma (összpontszám)
  - **FieldSize** – A tábla mérete

- A hosszútávú adattárolás lehetőségeit az **ISquaresGameDataAccess** interfész adja meg, amely lehetőséget ad a játékállapot betöltésére (*LoadAsync*), valamint mentésére (*SaveAsync*). A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg.
- Azinterfészt szöveges fájl alapú adatkezelésre a **SquaresGameDataAccess** osztály valósítja meg. A fájlkezelés során fellépő hibákat a **SquaresGameDataException** kivétel jelzi.
- A program az adatokat szöveges fájlként tudja eltárolni. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
- A fájl:

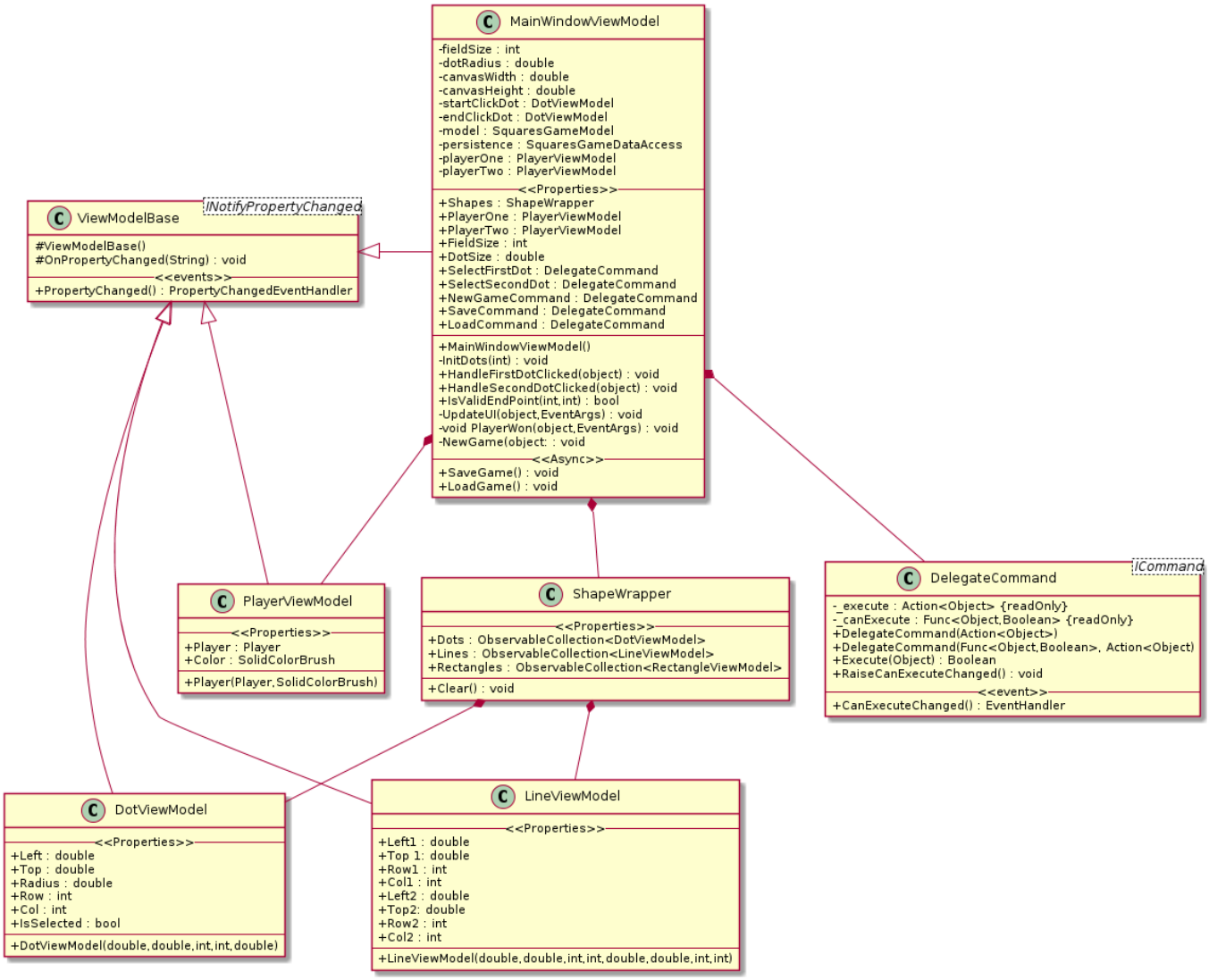
  - 1-2. sora megadja a két játékos adatait (**Játékosnév**, **Játékos szín** ARGB kódja, **pontszám**)
  - 3. sora megadja az **aktív játékos indexét**, és a **táblaméretet**
  - 4. sora megadja a **behúzott vonalak számát** (N)
  - A következő N sor egy-egy **behúzott vonal adatait tartalmazza** (2 pont sor és oszlop indexeit, valamint egy játékosindexet, ami jelzi melyik játékos húzta be)
  - Az ezt követő sor a **kialakult négyzetek számát** jelöli (M)
  - A következő M sor egy-egy négyzet bal felső és jobb alsó csúcspontjának sor és oszlop indexeit, valamint egy játékosindexet, ami jelzi melyik játékos húzta be az utolsó vonalat.

## Modell

- A modell lényegi részét a **SquaresGameModel** osztály valósítja meg, amely kezeli a játék belső állapotát (Játékosok pontszámai, játék vége, behúzott vonalak, behúzott négyzetek).
- Lehetőséget ad a játék újraindítására (**Restart**) az előző beállításokkal, játékállapot mentésére és betöltésére
- A fő játékmetódus a (**AddNewLine**) amely a játéktér 2 pontját regisztrálja behúzott vonalként, amennyiben ezek megfelelőek
- Az **UpdateUI** esemény kiváltódik amennyiben van újonnan regisztált vonal, ez jelzi a nézet felé, hogy a táblát újra kell rajzolni az új állapottal
- Az **EndGame** esemény kiváltódik amennyiben az összes lehetséges vonal be lett húzva, átadva a győztes játékosra mutató referenciát, vagy null-t, ha döntetlen.

## NézetModell

- A nézetmodell megvalósításához felhasználunk egy általános utasítás (**DelegateCommand**), valamint egy ős változásjelző (**ViewModelBase**) osztályt.
- A nézetmodell feladatait a **MainWindowViewModel** osztály látja el, amely parancsokat biztosít az új játék kezdéshez, játék betöltéshez, mentéshez. A nézetmodell tárolja a modell egy hivatkozását (**model**), de csupán információkat kér tőle, de nem avatkozik a játék futtatásába.
- A Játékalakzatok számára külön osztályokat (**ShapeWrapper, RectangleViewModel, LineViewModel, DotViewModel**) veszünk fel, amelyet az alakzatokhoz szükséges adatokat tárolják.




*3. ábra Az alkalmazás osztálydiagramja*

## Tesztelés

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **SquaresGameTest** osztályban

- Az alábbi tesztesetek kerültek megvalósításra
  - **Helyes** vonalak behúzása
    - Bal-jobb irány horizontális
    - Jobb-bal irány horizontális
    - Fel-le irány vertikális
    - Le-fel irány vertikális
  - **Helyes** vonalak **többszöri behúzása nem történik meg**
  - **Helytelen** vonalak behúzása nem történik meg
    - Átlós vonal
    - Horizontális – egy egységnél hosszabb
    - Vertikális – egy egységnél hosszabb
    - Nem vonal – Két azonos pont
    - Játéktéren kívül eső
  - **Helyes** egyszeres négyzet behúzása
  - **Helyes** kétszeres nézet behúzása
  - **Helyes** játékmenet szimuláció