1. Beadandó feladat dokumentáció

Feladat:

Készítsünk programot, amellyel az alábbi két személyes játékot játszhatjuk.

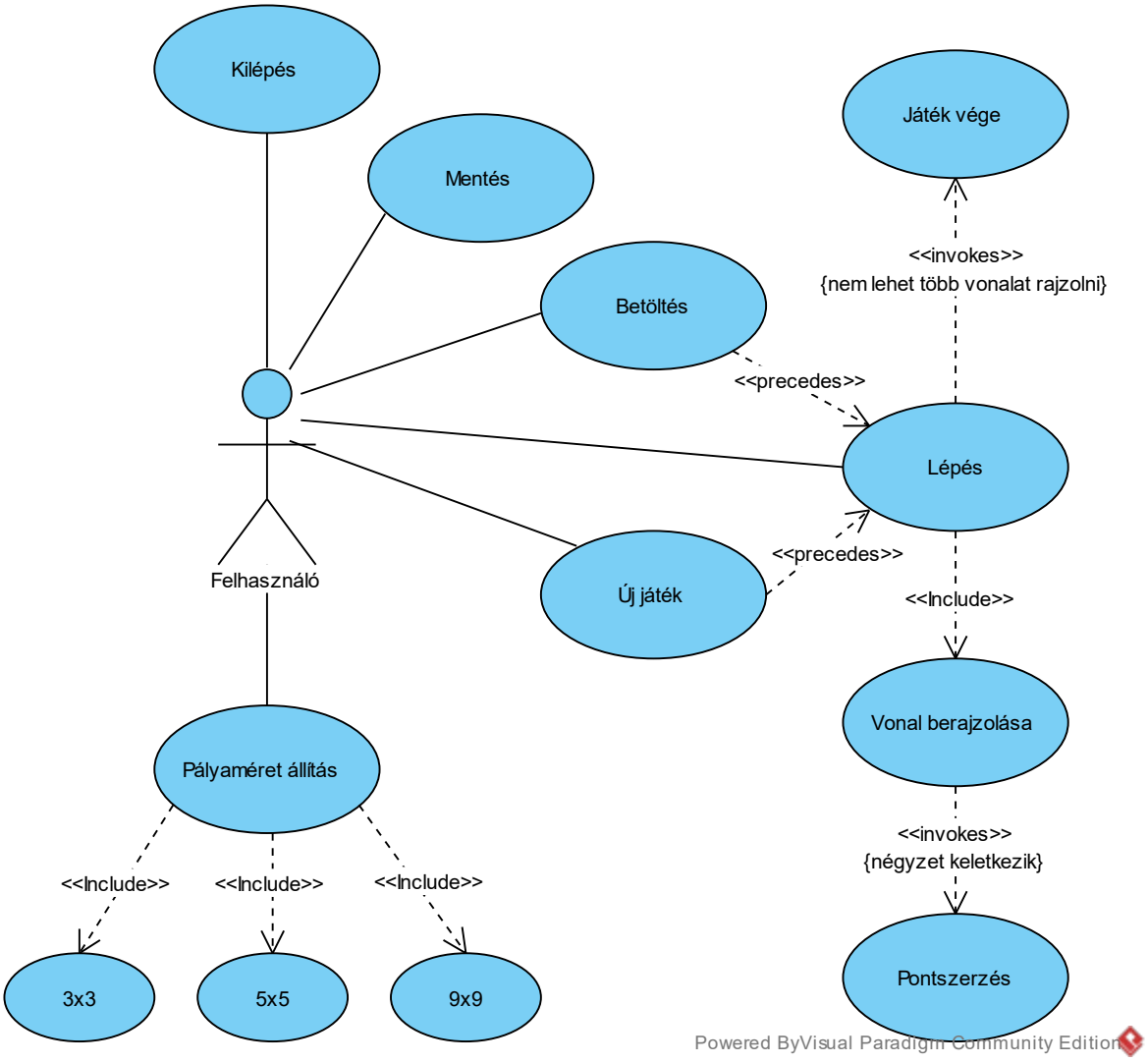
Adott egy $n \times n$ pontból álló játéktábla, amelyen a játékosok két szomszédos pont között vonalakat húzhatnak (vízszintesen, vagy függőlegesen). A játék célja, hogy a játékosok a húzogatással négyzetet tudjanak rajzolni (azaz ők húzzák be a negyedik vonalat, független attól, hogy az eddigieket melyikük 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 három pályamérettel játszhatjuk: 3x3-mas, 5x5-ös, 9x9-es. A program indításkor 5x5-ös pályamérettel töltődik be.
- A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (New Game, Load Game, Save Game, Exit), Settings (3x3, 5x5, 9x9). Az ablak alján megjelenítünk egy státuszsort, amely az aktuális pontszámokat és az aktuálisan lépő játékos nevét jelzi.
- A játéktáblát egy $n \times n$ db kör alakú gombból álló rács reprezentálja. A felület segíti a felhasználót, hogy az aktuális állapotban mely pontból (amelyiknek van olyan szomszédja, amely között még nincs vonal), s mely pontba húzható vonal (szomszédos és még nincs másik vonal). Egy vonal behúzásakor az aktuális játékos színére színeződik az, amennyiben keletkezik négyzet az szintén.
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (nem lehet több vonalat behúzni). Szintén dialógusablakokkal végezzük el a mentést, illetve betöltést, a fájlneveket a felhasználó adja meg.
- A felhasználói esetek az 1. ábrán láthatók.

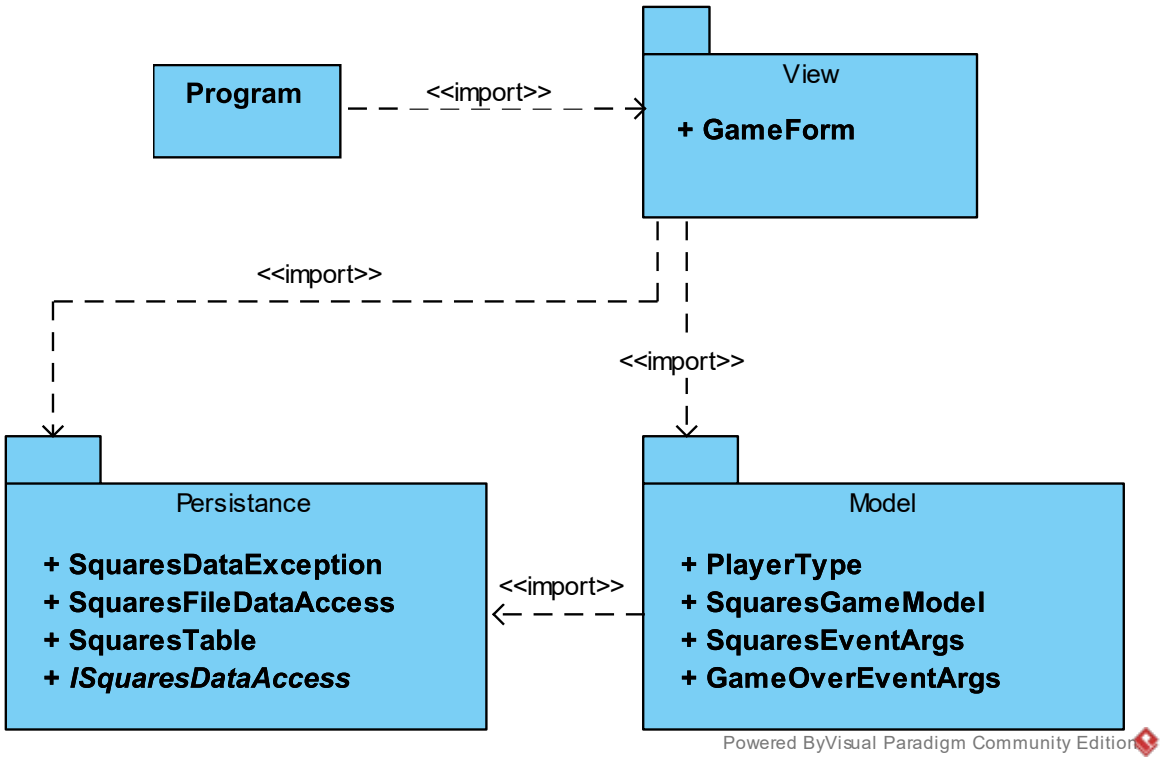


1. ábra: Felhasználói esetek diagramja

Tervezés:

❖ **Programszerkezet**

- A programot háromrétegű architektúrában valósítjuk meg. A megjelenítés a **`View`**, a modell a **`Model`**, míg a perzisztencia a **`Persistence`** névtérben helyezkedik el. A program csomagszerkezete a 2. ábrán látható.



2. ábra: Az alkalmazás csomagdiagramja

❖ Perzisztencia

- A perzisztencia feladata a játéktáblával (gráfszerű reprezentáció) kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
- A **`SquaresTable`** osztály egy érvényes játéktáblát biztosít (ellenőrzi a tárolt értékeket, kapott paramétereket). Tárolja a már berajzolt vonalakat (**`_edges`**), négyzeteket (**`_squares`**), és az aktuális játékost (**`_currentPlayer`**). A tábla alapértelmezés szerint 5×5-ös, de ez a konstruktorban paraméterezhető. Lehetőség van egy pontra illeszkedő vonalak lekérdezésére (**`GetIncidentEdges`**); egy pontból kiinduló lehetséges vonalak lekérdezésére (**`PossibleEdgesOf`**); két pont közötti vonal lekérdezésére (**`GetEdgeBetween`**); adott vonalat (két pont által meghatározott) tartalmazó négyzet(ek) lekérdezésére (**`FindSquaresWithVertices`**).
- A pontok (**`Vertex`**) és a vonalak (**`LabeledEdge`**) egy **`Utilities`** nevű osztályban vannak megvalósítva. A vonalak címkézve vannak (**`LabelType`** típusparaméterezhető címkével), és a tartalmazott pontok sorrendje nem számít egyenlőségvizsgálatkor, azaz irányítatlan.
- A tábla lehetőséget az állapotok lekérdezésére (**`IsFull`**, **`CurrentPlayer`**, **`Edges`**, **`Squares`**).
- A hosszú távú adattárolás lehetőségeit az **`ISquaresDataAccess`** 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 **`SquaresFileDataAccess`** osztály valósítja meg. A fájlkezelés során fellépő hibákat a **`SquaresDataException`** kivétel jelzi.
- A program az adatokat szöveges fájlként tudja eltárolni, melyek az **`sqt`** 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 szóközzel elválasztva. A második sora megadja az aktuális játékost. A harmadik a címkézett éleket (**`címke:x1,y1-x2,y2;`** formában). A nagyedik a címkézett négyzeteket (**`címke:x,y;`** formában, ahol x és y a bal felső sarok koordinátái).

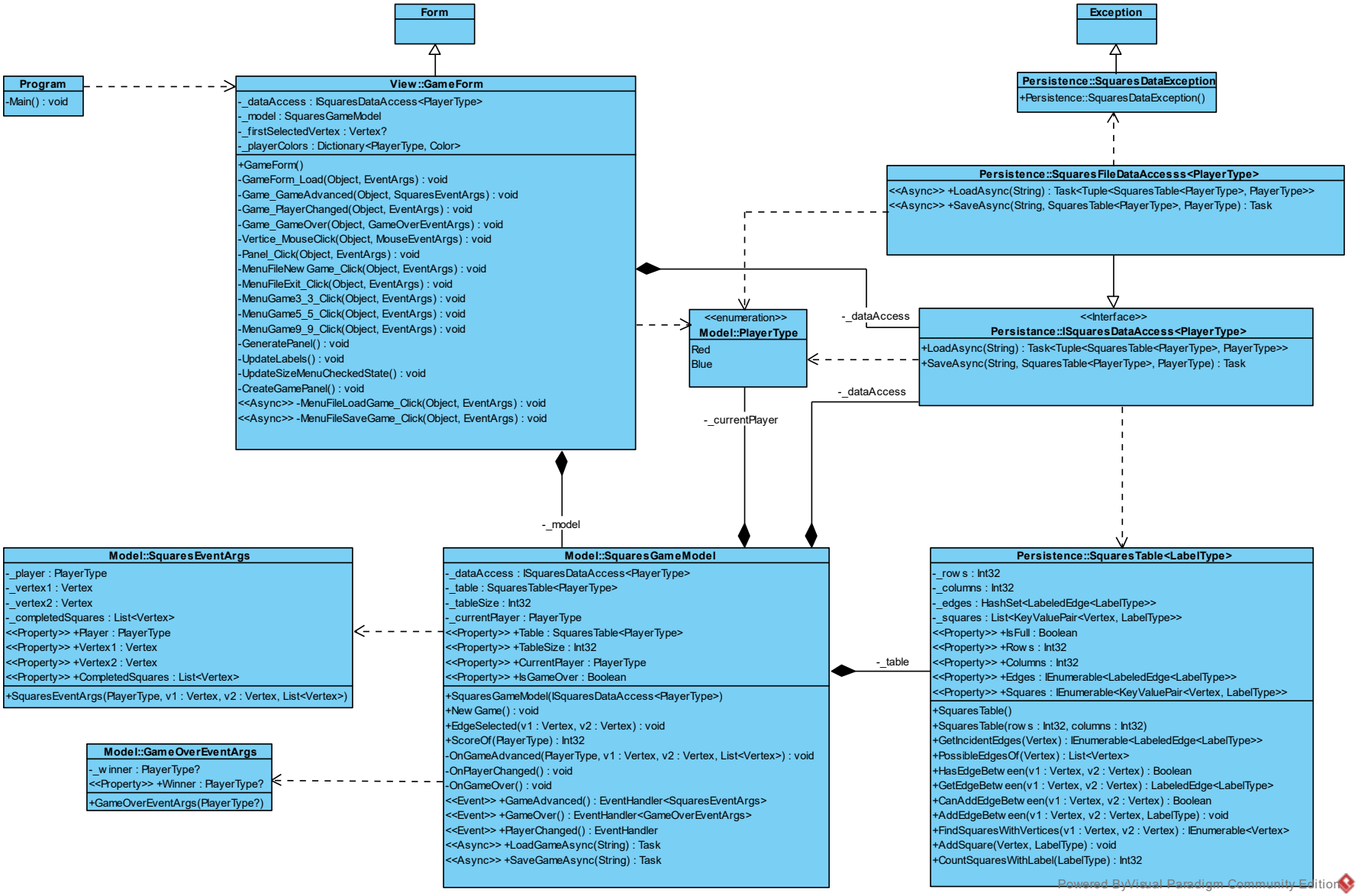
❖ Modell

- A modell lényegi részét a **`SquaresGameModel`** osztály valósítja meg, amely szabályozza a tábla tevékenységeit. A típus lehetőséget ad új játék kezdésére (**`NewGame`**), lépésre (**`EdgeSelected`**), valamint egy játékos pontszámának lekérdezésére (**`ScoreOf`**).
- A játékállapot változásáról a **`GameAdvanced`** esemény, a játék végéről a **`GameOver`** esemény, míg a játékos változásáról a **`PlayerChanged`** esemény tájékoztat. A GameAdvanced esemény argumentuma (**`SquaresEventArgs`**) tárolja az újonnan behúzott vonal végpontjait, valamint a vonal behúzásával keletkező négyzet(ek)et. A **`GameOver`** esemény argumentuma (**`GameOverEventArgs`**) tárolja a nyertest (ha van egyértelmű).
- A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (**`LoadGameAsync`**) és mentésre (**`SaveGameAsync`**).
- A játékosok típusát a **`PlayerType`** felsorolási típus reprezentálja. A modell fel van készítve ennek bővítésére.

❖ **Nézet**

- A nézetet a **`GameForm`** osztály biztosítja, amely tárolja a modell egy példányát (**`_model`**), valamint az adatelérés konkrét példányát (**`_dataAccess`**).
- A játéktáblát egy dinamikusan létrehozott gombmező (**`_vertexViews`**) reprezentálja, melyek egy egyedi csoportosító nézetbe vannak beágyazva (**`DrawablePanel`**), mely lehetőséget ad a hozzáadott vezérlők közötti vonalak, valamint négyzetek rajzolására. A gombok is testreszabottak (**`VertexView`**), hogy lehetőséget adjanak index hozzárendelésére, egyedi megjelenésre. A felületen létrehozzuk a megfelelő menüpontokat, illetve státuszsort, valamint dialógusablakokat, és a hozzájuk tartozó eseménykezelőket. A játéktábla generálását (**`GenerateTable`**), illetve a státuszsor frissítését (**`UpdateLabels`**) külön metódusok végzik.
- Az felhasználót vizuálisan is támogatja a nézet, kijelölve számára a kiválasztható pontokat.

A program teljes statikus szerkezete a 3. ábrán látható


Form
Exception
Program
-Main() : void
View::GameForm
-_dataAccess : ISquaresDataAccess<PlayerType>
-_model : SquaresGameModel
-_firstSelectedVertex : Vertex?
-_playerColors : Dictionary<PlayerType, Color>
+GameForm()
-GameForm_Load(Object, EventArgs) : void
-Game_GameAdvanced(Object, SquaresEventArgs) : void
-Game_PlayerChanged(Object, EventArgs) : void
-Game_GameOver(Object, GameOverEventArgs) : void
-Vertice_MouseClick(Object, MouseEventArgs) : void
-Panel_Click(Object, EventArgs) : void
-MenuFileNew Game_Click(Object, EventArgs) : void
-MenuFileExit_Click(Object, EventArgs) : void
-MenuGame3_3_Click(Object, EventArgs) : void
-MenuGame5_5_Click(Object, EventArgs) : void
-MenuGame9_9_Click(Object, EventArgs) : void
-GeneratePanel() : void
-UpdateLabels() : void
-UpdateSizeMenuCheckedState() : void
-CreateGamePanel() : void
<<Async>> -MenuFileLoadGame_Click(Object, EventArgs) : void
<<Async>> -MenuFileSaveGame_Click(Object, EventArgs) : void
Persistence::SquaresDataException
+Persistence::SquaresDataException()
Persistence::SquaresFileDataAccesss<PlayerType>
<<Async>> +LoadAsync(String) : Task<Tuple<SquaresTable<PlayerType>, PlayerType>>
<<Async>> +SaveAsync(String, SquaresTable<PlayerType>, PlayerType) : Task
<<enumeration>>
Model::PlayerType
Red
Blue
-_dataAccess
<<Interface>>
Persistance::ISquaresDataAccess<PlayerType>
+LoadAsync(String) : Task<Tuple<SquaresTable<PlayerType>, PlayerType>>
+SaveAsync(String, SquaresTable<PlayerType>, PlayerType) : Task
-_dataAccess
-_currentPlayer
-_model
Model::SquaresEventArgs
-_player : PlayerType
-_vertex1 : Vertex
-_vertex2 : Vertex
-_completedSquares : List<Vertex>
<<Property>> +Player : PlayerType
<<Property>> +Vertex1 : Vertex
<<Property>> +Vertex2 : Vertex
<<Property>> +CompletedSquares : List<Vertex>
+SquaresEventArgs(PlayerType, v1 : Vertex, v2 : Vertex, List<Vertex>)
Model::SquaresGameModel
-_dataAccess : ISquaresDataAccess<PlayerType>
-_table : SquaresTable<PlayerType>
-_tableSize : Int32
-_currentPlayer : PlayerType
<<Property>> +Table : SquaresTable<PlayerType>
<<Property>> +TableSize : Int32
<<Property>> +CurrentPlayer : PlayerType
<<Property>> +IsGameOver : Boolean
+SquaresGameModel(ISquaresDataAccess<PlayerType>)
+New Game() : void
+EdgeSelected(v1 : Vertex, v2 : Vertex) : void
+ScoreOf(PlayerType) : Int32
-OnGameAdvanced(PlayerType, v1 : Vertex, v2 : Vertex, List<Vertex>) : void
-OnPlayerChanged() : void
-OnGameOver() : void
<<Event>> +GameAdvanced() : EventHandler<SquaresEventArgs>
<<Event>> +GameOver() : EventHandler<GameOverEventArgs>
<<Event>> +PlayerChanged() : EventHandler
<<Async>> +LoadGameAsync(String) : Task
<<Async>> +SaveGameAsync(String) : Task
-_table
Persistence::SquaresTable<LabelType>
-_row s : Int32
-_columns : Int32
-_edges : HashSet<LabeledEdge<LabelType>>
-_squares : List<KeyValuePair<Vertex, LabelType>>
<<Property>> +IsFull : Boolean
<<Property>> +Row s : Int32
<<Property>> +Columns : Int32
<<Property>> +Edges : IEnumerable<LabeledEdge<LabelType>>
<<Property>> +Squares : IEnumerable<KeyValuePair<Vertex, LabelType>>
+SquaresTable()
+SquaresTable(row s : Int32, columns : Int32)
+GetIncidentEdges(Vertex) : IEnumerable<LabeledEdge<LabelType>>
+PossibleEdgesOf(Vertex) : List<Vertex>
+HasEdgeBetw een(v1 : Vertex, v2 : Vertex) : Boolean
+GetEdgeBetw een(v1 : Vertex, v2 : Vertex) : LabeledEdge<LabelType>
+CanAddEdgeBetw een(v1 : Vertex, v2 : Vertex) : Boolean
+AddEdgeBetw een(v1 : Vertex, v2 : Vertex, LabelType) : void
+FindSquaresWithVertices(v1 : Vertex, v2 : Vertex) : IEnumerable<Vertex>
+AddSquare(Vertex, LabelType) : void
+CountSquaresWithLabel(LabelType) : Int32
Model::GameOverEventArgs
-_w inner : PlayerType?
<<Property>> +Winner : PlayerType?
+GameOverEventArgs(PlayerType?)
Powered By Visual Paradigm Community Edition

Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **`SquaresTest`** osztályban.
- Az alábbi tesztesetek kerültek megvalósításra:
  - **`SquaresGameModelNewGameTest:`** új játék indításának tesztelése. Megfelelő a tábla mérete, behúzott vonalak, rajzolt négyzetek száma.
  - **`SquaresGameModelEdgeSelectedTest:`** él kiválasztásának tesztelése, hogy bekerül-e a táblába.
  - **`SquaresGameModelEdgeSelectedNewSquareTest:`** új négyzet keletkezésének tesztelése, megfelelőek a keletkező koordináták és megfelelő játékoshoz rendelődik-e. Eseménykezelő (**`GameAdvanced`**) tesztelése.
  - **`SquaresGameModelPlayerSteppingTest:`** következő játékosra lépés tesztelése és négyzet rajzolása esetén ugyanaz a játékos következik-e.
  - **`SquaresGameModelScoreOfTest:`** négyzet rajzolása esetén csak a megfelelő játékos kap pontot.
  - **`SudokuGameModelLoadTest:`** a játék modell betöltésének tesztelése mockolt perzisztencia réteggel.