2.Beadandó feladat dokumentáció

# Készítette:

Komornik Márton

E-mail: [pr2nl9@inf.elte.hu](mailto:pr2nl9@inf.elte.hu)

# Feladat:

*Potyogós amőba*

Készítsünk programot, amellyel a potyogós amőba játékot lehet játszani, vagyis az amőba azon változatát, ahol a jeleket felülről lefelé lehet beejteni a játékmezőre. A játékmező itt is 𝑛×𝑛 -es tábla, és ugyanúgy X, illetve O jeleket potyogtathatunk a mezőre. A játék akkor ér véget, ha betelik a tábla (döntetlen), vagy valamelyik játékos kirak 4 egymás melletti jelet (vízszintesen, vagy átlósan).

A program minden lépésnél jelezze, hogy melyik játékos következik, és a tábla egy üres mezőjére kattintva helyezhessük el a megfelelő jelet. Természetesen csak a szabályos lépéseket engedje meg a program.

A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (10×10, 20×20, 30×30), játék szüneteltetésére, valamint játék mentésére és betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött (a táblán jelölje meg a győztes 4 karaktert). A program folyamatosan jelezze külön-külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépéséig tart, ezt is mentsük el és töltsük be).

# Elemzés:

* A játékot három táblával játszhatjuk: 1.tábla mérete: 10x10, 2.tábla: 20x20, 3.tábla: 30x30
* A feladatot egyablakos asztali alkalmazásként Windows Presentation Foundation grafikus felülettel valósítjuk meg.
* Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (Új játék (kis tábla:10x10, közepes tábla:20x20, nagy tábla:30x30), Pause, Játék mentése, Játék betöltése). Az ablak tetején, a menüsor alatt megjelenítünk egy státuszsort, amely a két játékos játékidejét jelzi.
* A játéktáblát egy 10x10 vagy egy 20x20, vagy egy 30x30 nyomógombokból álló rács reprezentálja.
* A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak
* (valamelyik játékosnak összejött a 4 szimbólum vagy betelt a tábla). Szintén dialógusablakokkal végezzük el a mentést, illetve betöltést, a fájlneveket a felhasználó adja meg.

A felhasználói esetek az 1. ábrán láthatóak.

A képen szöveg, diagram, Betűtípus, képernyőkép látható

Automatikusan generált leírás  
**1. ábra: A felhasználói esetek diagramja**

# Tervezés:

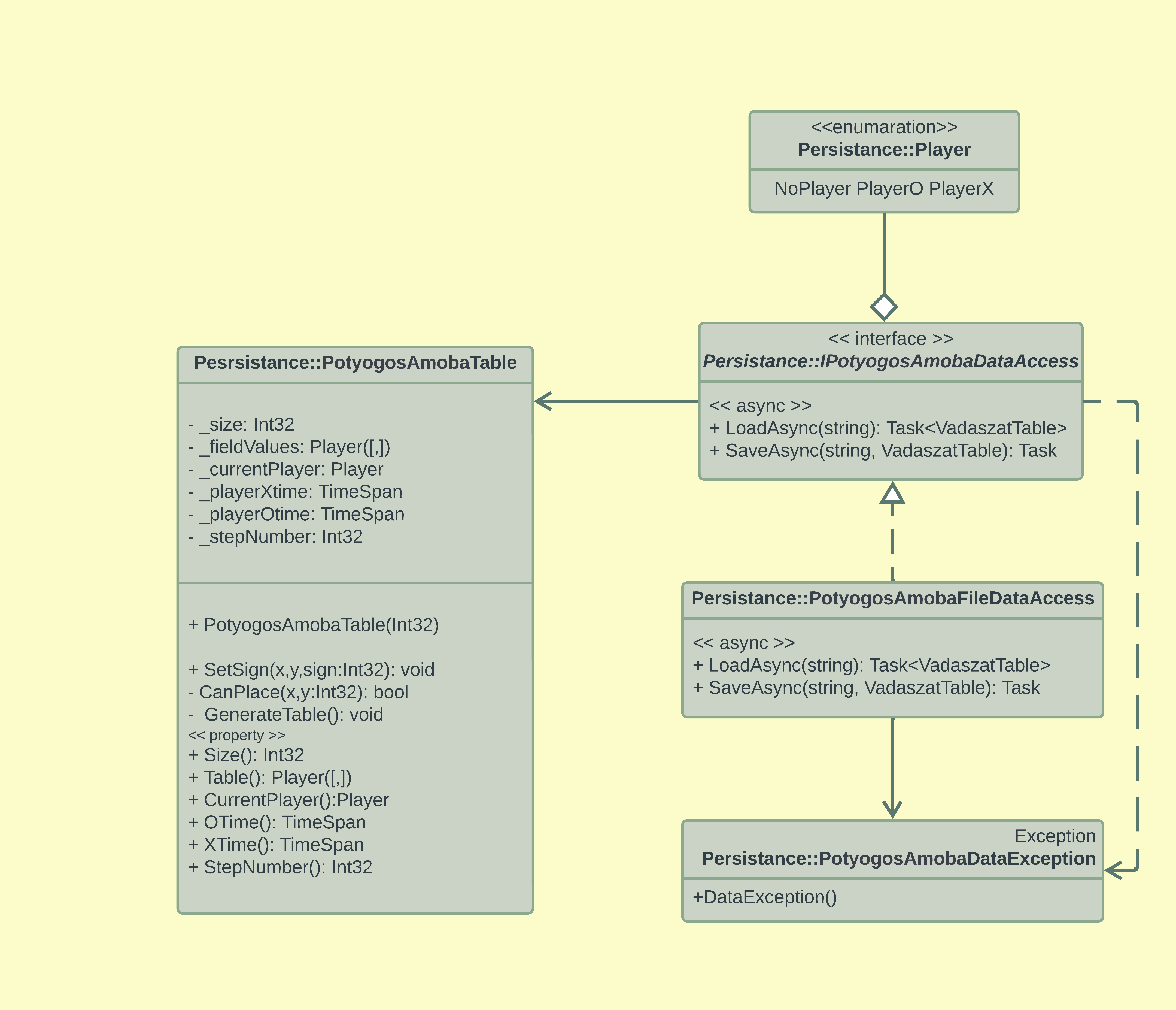
* Programszerkezet:
  + A programot MVVM architektúrában valósítjuk meg, ennek megfelelően View, Model, ViewModel és Persistence névtereket valósítunk meg az alkalmazáson belül. A program környezetét az alkalmazás osztály (App) végzi, amely példányosítja a modellt, a nézetmodell és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést. A program csomagszerkezete a 2. ábrán látható.
  + A program szerkezetét két projektre osztjuk implementációs megfontolásból: a Persistence és Model csomagok a program felületfüggetlen projektjében, míg a ViewModel és View csomagok a WPF függő projektjében kapnak helyet.

A képen szöveg, Post-it cetli, kézírás, diagram látható

Automatikusan generált leírás

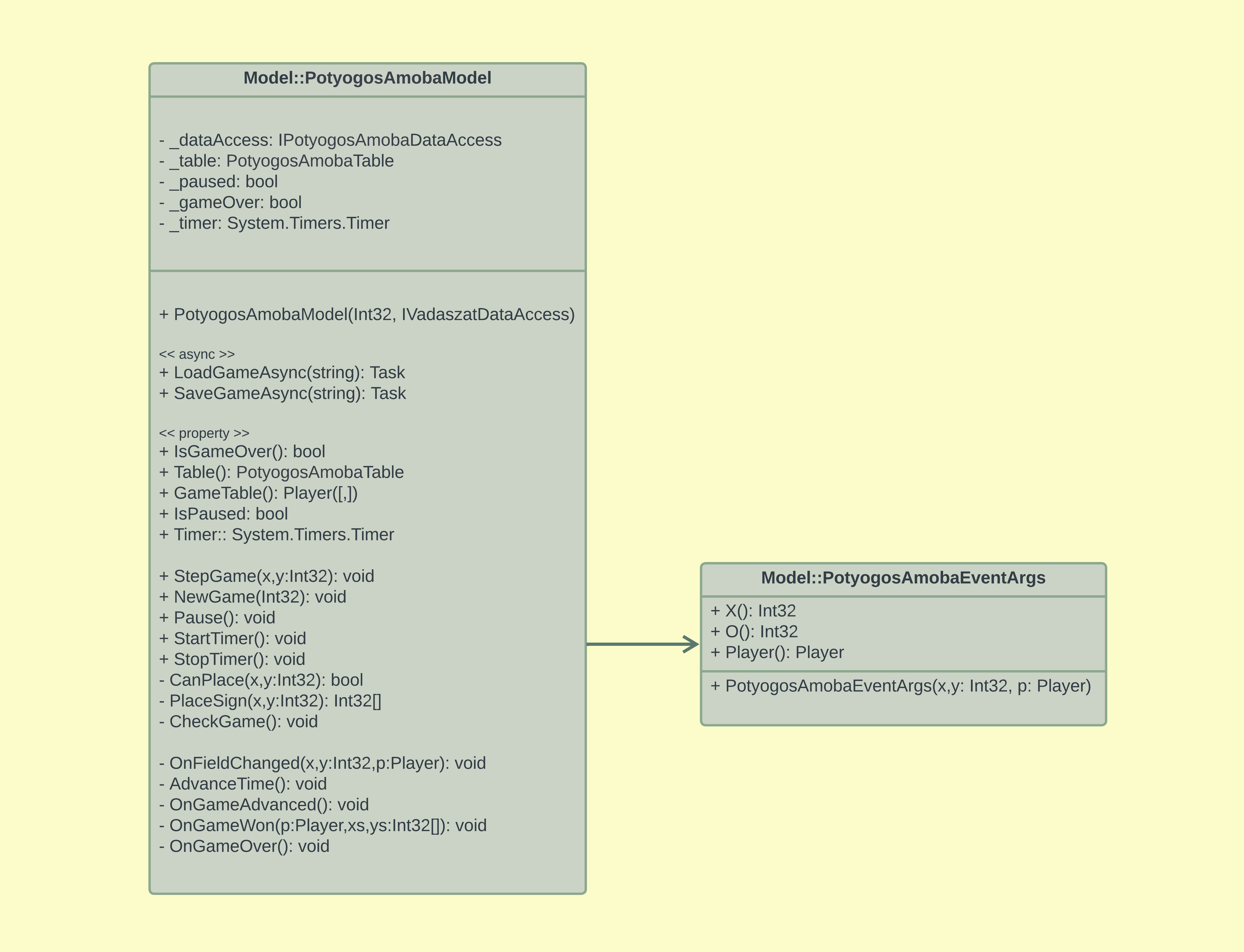
**2. ábra: Az alkalmazás csomagdiagramja**

* Perzisztencia:
* Az adatkezelés feladata az Ambőba táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
* A PotyogosAmobaTable osztály egy érvényes Amőba táblát.
* A hosszú távú adattárolás lehetőségeit az IPotyogosAmobaDataAccess 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 PotyogosAmobaFileDataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a PotyogosAmobaDataException kivétel jelzi.
* A program az adatokat szöveges fájlként tudja eltárolni, melyek az stl 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, a játékosok gondolkodási idejét, valamint a soron következő játkost. A fájl többi része izomorf leképezése a játéktáblának, azaz összesen n sor következik, és minden sor n számot tartalmaz szóközökkel választva. A számok 0-2 közöttiek lehetnek, ahol 0 reprezentálja a még üres mezőt.

****

**3. ábra: A Persistance csomag osztálydiagramja**

* Modell
  + A modell lényegi részét a PotyogosAmobaModel osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, úgymint az idő (\_XTime) és a lépések (\_stepCount). A típus lehetőséget ad új játék kezdésére (NewGame), valamint lépésre (StepGame). Új játéknál megadható a kiinduló játéktábla mérete. Az idő előreléptetését időbeli lépések végzéséve (AdvanceTime) tehetjük meg.
  + A mezők állapotváltozásáról a FieldChanged esemény tájékoztat. Az esemény argumentuma (PotyogosAmobaFieldEventArgs) tárolja a megváltozott mező pozícióját.
  + A játékállapot megváltozásáról a GameAdvanced esemény, míg a játék végéről a GameOver esemény tájékoztat.
  + A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (LoadGameAsync) és mentésre (SaveGameAsync)



**4. ábra: A modell csomag osztálydiagramja**

* NézetModell:
  + A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) osztályt.
  + A nézetmodell feladatait a AmobaViewModel 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 (\_model), de csupán információkat kér le tőle, illetve a játék méretét szabályozza. Direkt nem avatkozik a játék futtatásába.
  + A játékmező számára egy külön mezőt biztosítunk (AmobaField), amely eltárolja a pozíciót, szöveget, engedélyezettséget, valamint a lépés parancsát (StepCommand). A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (Fields).

A képen szöveg, képernyőkép, Betűtípus, kézírás látható

Automatikusan generált leírás

**5. ábra: A nézetmodell osztálydiagramja**

* 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 rácsot (UniformGrid), amely gombokból áll. 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.

Környezet:

* 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.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

**6. ábra: A vezérlés osztálydiagrammja**

# Tesztelés:

* A modell funkcionalitása egységtesztek segítségével lett ellenőrizve az AmobalTest osztályban.
* Az alábbi tesztesetek kerültek megvalósításra:
  + New10TableTest
  + New20TableTest
  + New30TableTest: E három új játék inditását, a konstruktorok helyes működését ellenőrzi az adott mérettel
  + StepGameTest: Játékbeli lépés hatásainak ellenőrzése, játék megkezdése előtt, valamint után. Több lépés végrehajtása azonos játékmezőn, esemény kiváltásának ellenőrzése.
  + GameWonTest: A Játék megnyerésének felismerését teszteli
  + LoadTest: A mentett játék hibamentes betöltése kerül ellenőrzésre