3. Beadandó feladat dokumentáció

Készítette:  
Kujbus Róbert Ádám  
E-mail: [a7hlth@inf.elte.hu](mailto:a7hlth@inf.elte.hu)  
  
Feladat

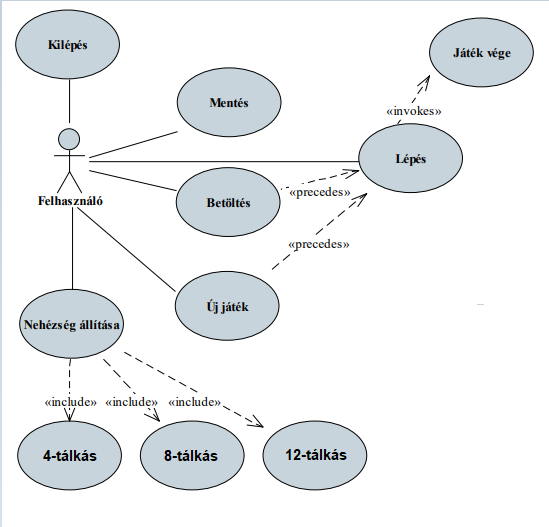
Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Két játékos egymással szemben helyezkedik el, közöttük pedig (paraméterként megadható) páros számú tálka és két gyűjtőtál az alábbi lerendezésben.   
Mindkét játékos a hozzá közelebbi tálkákat és a tőle jobb kézre eső gyűjtőtálat mondhatja sajátjának. (Így az ellenfél gyűjtőtálja baloldalra esik.)   
Kezdetben mindegyik tálkában 6-6 kavics van, a gyűjtőtálak pedig üresek. A játékban a soron következő játékos kiválasztja egyik saját tálkáját (ez nem lehet a gyűjtőtál), hogy azt kiürítse úgy, hogy tartalmát az óramutató járásával ellentétes irányban egyesével beledobálja, majd ismét a saját tálkáiba, de azt a tálkát kihagyva, amelyiknek a kiürítését végezzük, amíg el nem fogynak a kavicsok. Ha az egyik játékos rákattint valamelyik tálkájára, akkor a tálkában lévő kavicsok áthelyezése automatikusan történjen meg. Ha az utolsó kavics a játékos saját üres tálkáinak egyikébe kerül, akkor ezt a kavicsot, valamint a szemközti tálka tartalmát a saját gyűjtőládába teszi. Viszont, ha az utolsó kavics a játékos saját gyűjtőtálkájába esik, akkor újra ő következik, de ezt csak egyszer teheti meg, hogy ellenfele is szóhoz juthasson. A játéknak akkor van vége, ha az egyik térfél kiürült, azaz az egyik játékos tálkái mind kiürülnek. Ekkor az a játékos nyeri a játékot, akinek a gyűjtőtáljában több kavics van. A program biztosítson lehetőséget új játék kezdésére a tálkák számának megadásával (4, 8, 12), játék mentésére és betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

## Elemzés

* A játékot három nehézségi szinttel játszhatjuk: Négy-tálkás (4 tálka + 2 gyűjtőtálka),

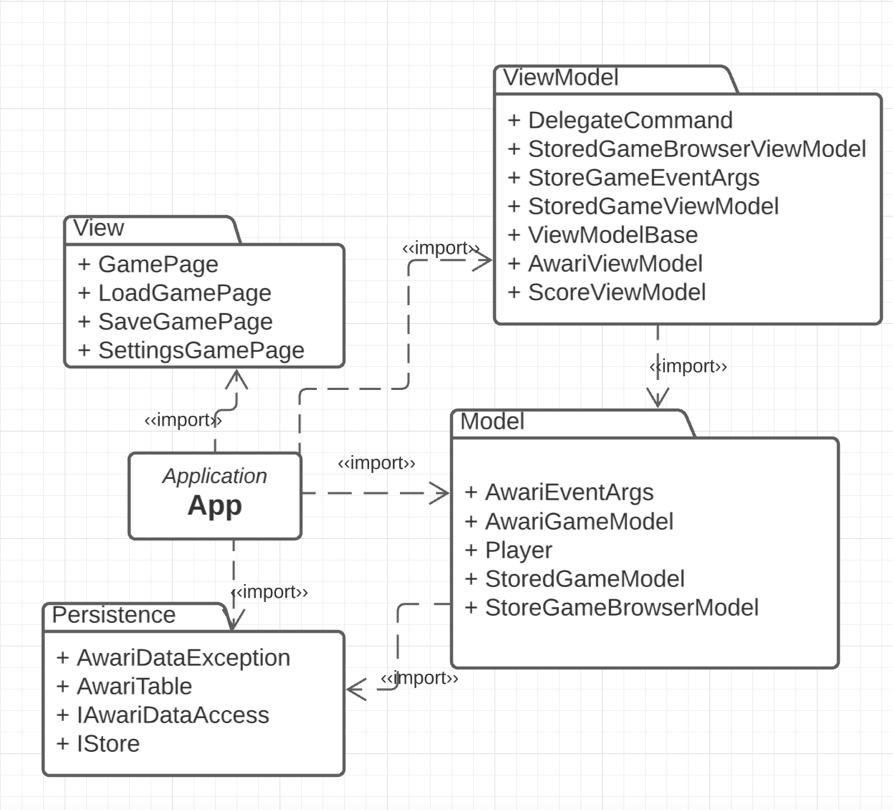
Nyolc-tálkás (8 tálka + 2 gyűjtőtálka) , Tizenkét-tálkás (12 tálka + 2 gyűjtőtálka) .   
A program indításkor nyolctálkás nehézséget állít be, és automatikusan új játékot indít.

* A feladatot egyablakos telefonos alkalmazásként Xamarin Forms grafikus felülettel valósítjuk meg.
* Az ablakban elhelyezünk egy menüt a következő menüpontokkal: Új játék, Beállítások (Játék mentése, játék betöltése, 4 tálkás játék, 8 tálkás játék, 12 tálkás játék).
* A játéktáblát N+2 ( N : sima tálkák száma ) nyomógomb reprezentálja. A nyomógomb egérkattintás hatására, az aktív játékos adott tálkájában lévő kavicsokat átpakolja egyesével, jobbra, míg el nem fogynak a kavicsok. Fontos, hogy ugyanabba a tálkába már vissza kavicsot nem rakunk. A két gyűjtőtálka sosem aktív (az értékük csak pakolással változik).
* A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (Valamelyik játékos táblája üres). Szintén dialógusablakokkal végezzük el a mentést, illetve betöltést, a fájlneveket a felhasználó adja meg.
* A felhasználói esetek az 1. ábrán láthatóak.

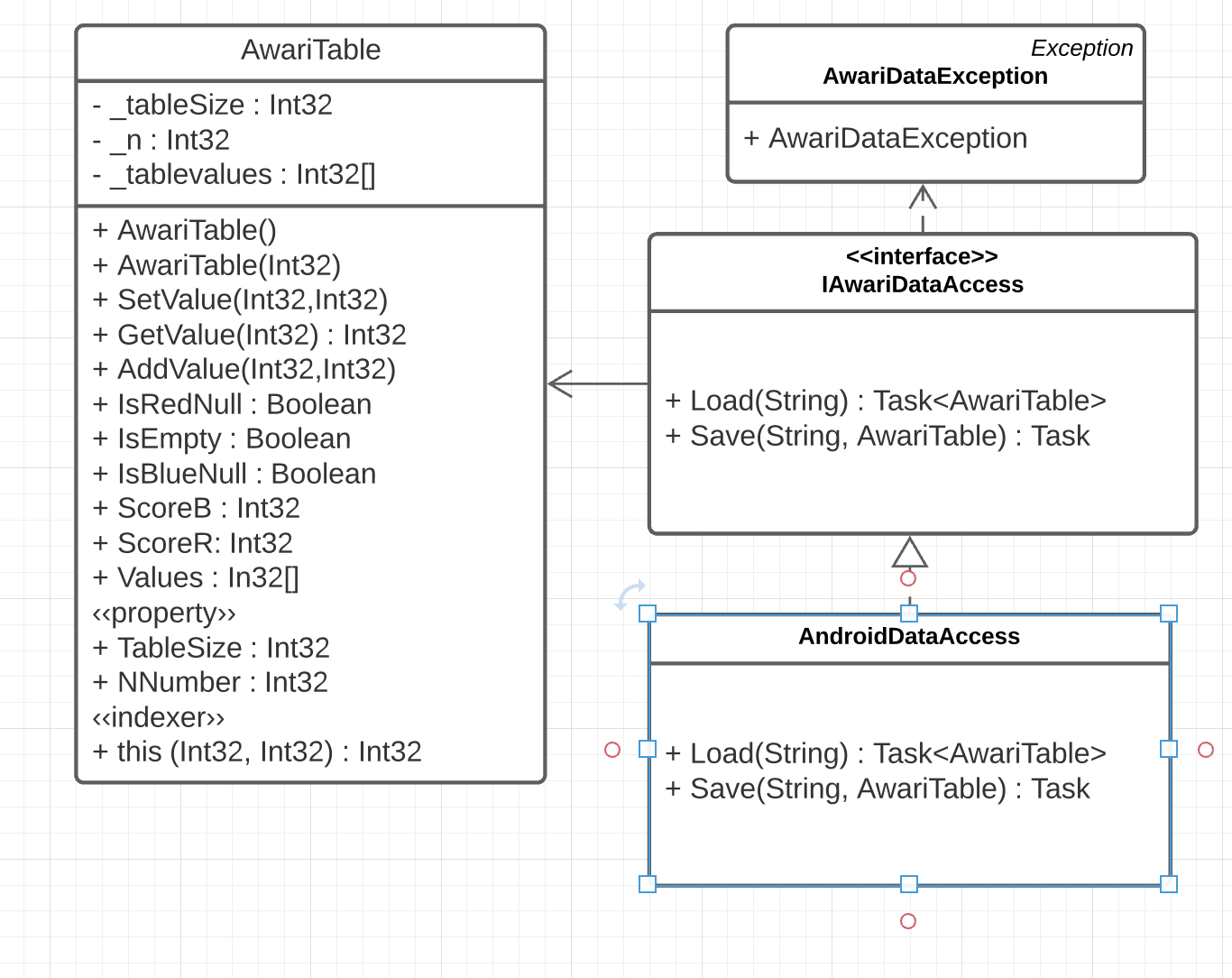
  
1. ábra: Felhasználói esetek diagramja

## Tervezés

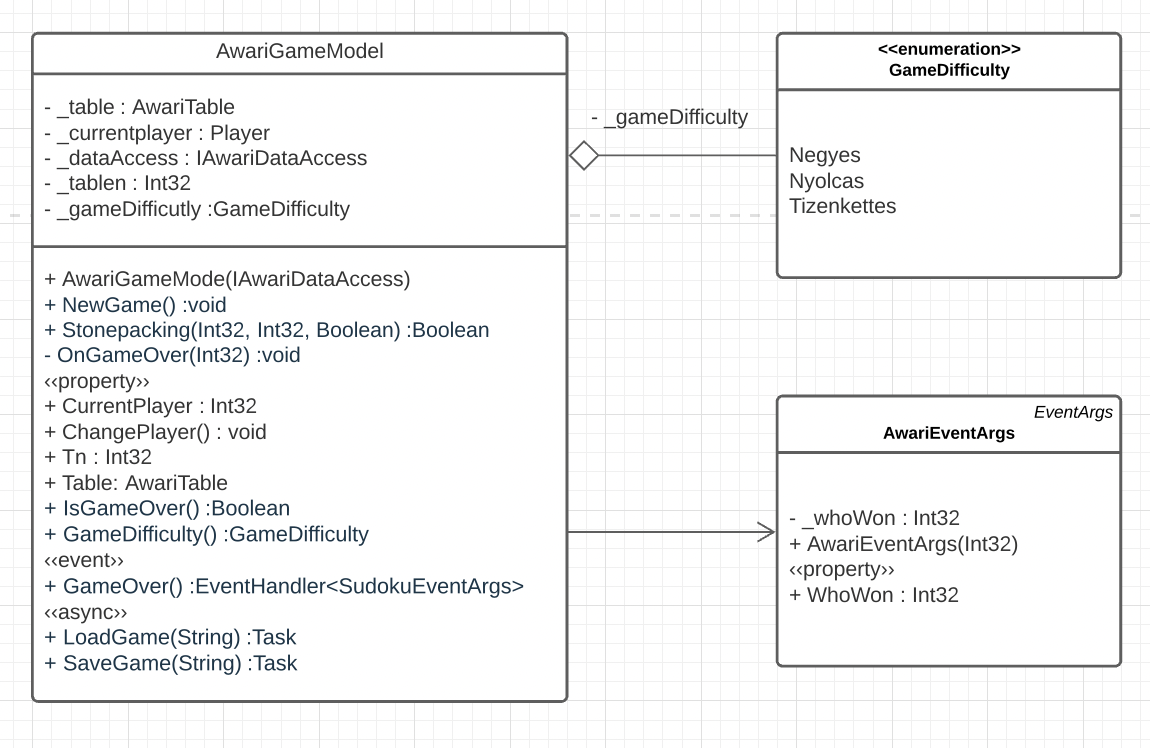
* Programszerkezet
  + A programot MVVM architektúrában valósítjuk meg, ennek megfelelően **View**, **Model**, **ViewModel** és **Persistence** névtereket valósítunk meg az alkalmazáson belül.
  + A program vezérlését az alkalmazás osztály (**App**) végzi, amely példányosítja a modellt, a nézetmodell és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést.
  + A program csomagdiagramja a 2. ábrán látható.

  
 2.ábra: Az alkalmazás csomagdiagramja

* + Perzisztencia (3. ábra):
    - Az adatkezelés feladata az Awari táblával kapcsolatos információk tárolása, valamint betöltés/mentés biztosítása.
    - Az ***AwariTable*** osztály egy érvényes Awari táblát biztosít, ahol minden mezőre ismert az értéke \_tableValues. A tábla lehetőséget az állapotok lekérdezésére (IsRedNull, IsBlueNull,GetValue), illetve direkt beállítás (SetValue, AddValue) elvégzésére.
    - A hosszú távú adattárolás lehetőségeit az IAwariDataAccess 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 Android adatkezelésre a **AndroidDataAccess** osztály valósítja meg. A fájlkezelés során fellépő hibákat a **AwariDataException** kivétel jelzi.
    - A program az adatokat szöveges fájlként tudja eltárolni, melyek az awt kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
    - A fájl egy sor, mely megadja a tábla méretét, majd szóközökkel elválasztva a tábla mérete+2 adat, melyek a tábla értékeiül szolgálnak.

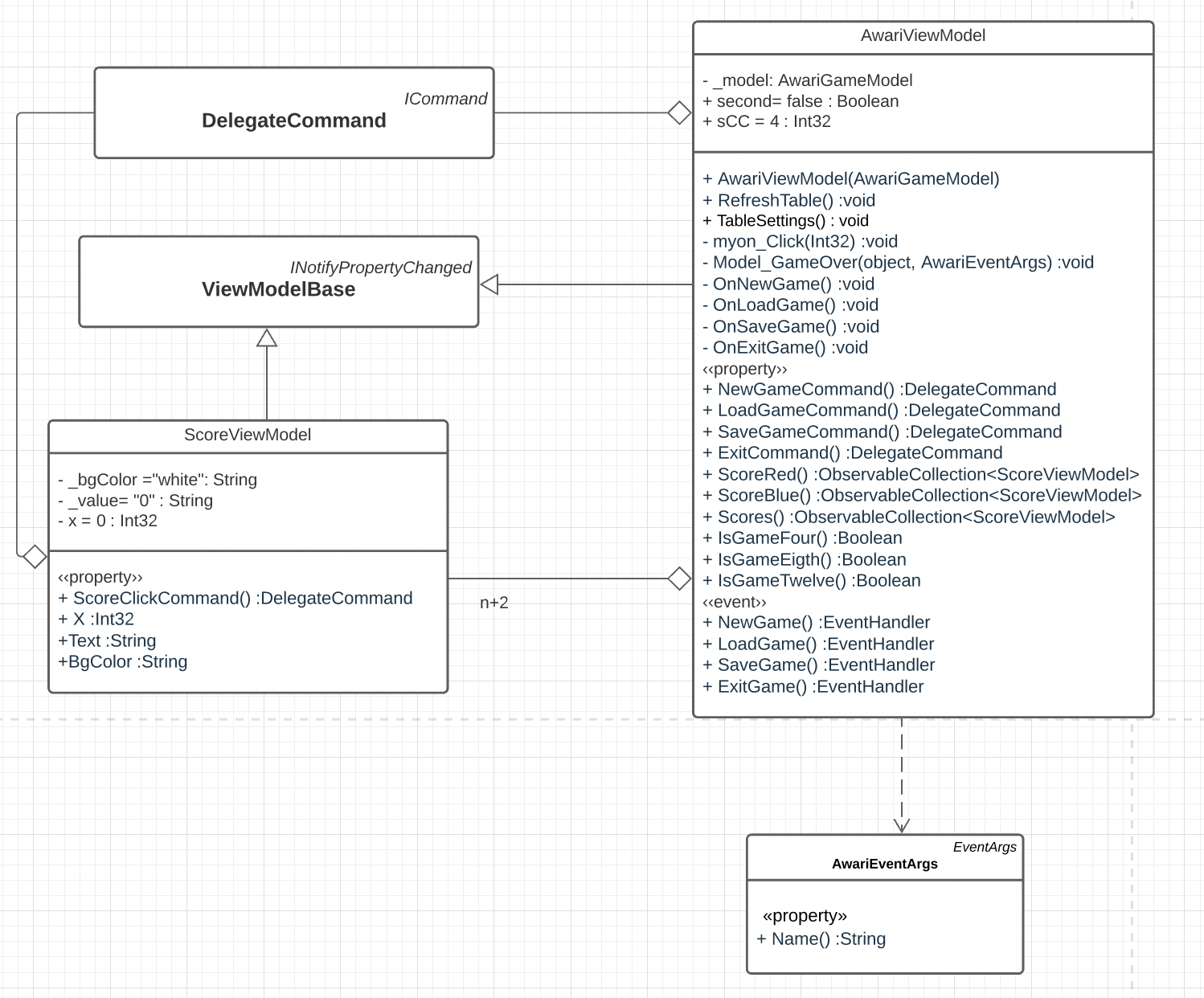
  
3. ábra: A Persistence csomag osztálydiagramja

* Modell (4. ábra):
  + - A modell lényegi részét az ***AwariGameModel*** osztály valósítja meg, amely szabályozza a tábla tevékenységeit. A típus lehetőséget ad új játék kezdésére (***NewGame***), valamint lépésre (***StonePacking***). Új játéknál megadható a kiinduló játéktábla is, különben automatikusan generálódnak a tábla értékei.
    - A játék végéről a ***GameOver*** esemény tájékoztat. Az események argumentuma (***AwariEventArgs***) tárolja a győzelem állapotát, és a győztes ki létét.
    - A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (***LoadGameAsync***) és mentésre (***SaveGameAsync***)
    - A játék nehézségét a ***GameDifficulty*** felsorolási típuson át kezeljük, és az ***AwariGameModel*** osztályban konstansok segítségével tároljuk az adott játékost.



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

* Nézetmodell (5.ábra) :
  + - A nézetmodell megvalósításához felhasználunk egy általános utasítás (***DelegateCommand***), valamint egy ős változásjelző (***ViewModelBase***) osztályt.
    - A nézetmodell feladatait a ***AwariViewModel*** 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 játéknehézséget szabályozza. Direkt nem avatkozik a játék futtatásába.

**5.ábra: A ViewModel csomag osztálydiagramja**

* Nézet:
  + - A nézetet navigációs lapok segítségével építjük fel.
    - A GamePage osztály tartalmazza a játéktáblát, amelyet egy FlowListView

segítségével valósítunk meg (ez egy külső komponens), amelyben Button

elemeket helyezünk el.

* + - A SettingsPage osztály tartalmazza a betöltés, mentés gombjait, illetve

Switch példányokat a nehézség állítására.

* Vezérlés(6.ábra):
  + Az **App** osztály feladata az alkalmazás vezérlése, a rétegek példányosítása

és az események feldolgozása.

* + Kezeljük az alkalmazás életciklust, így felfüggesztéskor (**OnSleep**)

elmentjük az aktuális játékállást (**SuspendedGame**), folytatáskor (**OnResume**) és újraindításkor (**OnStart**) pedig folytatjuk, amennyiben történt mentés.

**A képen szöveg látható

Automatikusan generált leírás6.ábra: A vezérlés osztálydiagramja**

## Tesztelés

* A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a AwariTest osztályban.
* • Az alábbi tesztesetek kerültek megvalósításra:
  + AwariGameModelOriginal
  + AwariGameModelNewGameFourTest
  + AwariGameModelNewGameTwelveTest
  + AwariGameModelLoadTest
  + AwariGameModelStonepacking
  + AwariGameModelStoneckingScoreCup