**1. Beadandó feladat dokumentáció**

**Készítette:**

**Bárkányi Péter**

[h4u2hj@inf.elte.hu](mailto:h4u2hj@inf.elte.hu)

**Feladat:**

Készítsünk programot a közismert Tetris játékra.

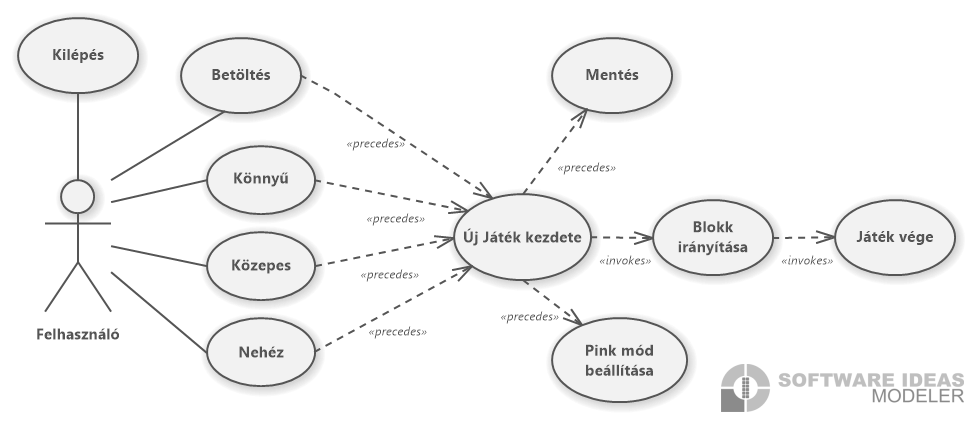
Adott egy 𝑛×𝑚 pontból álló tábla, amely kezdetben üres. A tábla tetejéről egymás után új, 4 kockából álló építőelemek hullanak, amelyek különböző formájúak lehetnek (kocka, egyenes, L alak, tető, rombusz). Az elemek rögzített sebességgel esnek lefelé, és az első, nem telített helyen megállnak. Amennyiben egy sor teljesen megtelik, az eltűnik a játékmezőről, és minden felette lévő kocka eggyel lejjebb esik.

A játékosnak lehetősége van az alakzatokat balra, jobbra mozgatni, valamint forgatni óramutató járásával megegyező irányba, így befolyásolhatja azok mozgását. A játék addig tart, amíg a kockák nem érik el a tábla tetejét.

A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (4×16, 8×16, 12×16), valamint játék szüneteltetésére (ekkor nem telik az idő, és nem mozognak az elemek). Ismerje fel, ha vége a játéknak, és jelenítse meg, mennyi volt a játékidő. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére.

**Elemzés:**

* A játékot három nehézségi szinttel játszhatjuk: könnyű (4 oszlop szélességgel), közepes (8 oszlop szélességgel), nehéz (12 oszlop szélességgel). A program indításakor ki kell választani a nehézséget, ekkor elindul a játék.
* A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
* Az ablak jobb oldalán elhelyezünk egy Load és egy Save gombot. Ezekkel új játékot tudunk betölteni, illetve elmenteni. A bal oldalon tudjuk gombokkal kiválasztani a nehézséget, majd megjelenik egy Pause és egy New Game gomb melyekkel szüneteltethetjük és új játékot kezdhetünk. Felettük megjelenik a pontszámunk, valamint a játék végén az eltelt idő. A jobb oldalon található még a Pink Mode gomb mellyel pink színűre állíthatjuk a játékot.
* A játéktáblát egy 4/8/12 × 16 képekből álló rács reprezentálja. A képek forrásának megváltoztatásával jelenik meg a pályán egy blokk vagy üres cella. A blokkokat lehet forgatni mindkét irányba és lehet őket leejteni alulra.
* A játék megjeleníti a végső pontszámunkat és eltelt időt amikor vége a játéknak (a rács tetejére értünk és nincs már hely blokkoknak). A játékban 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 háromrétegű architektúrában valósíjuk meg. A megjelenítés a **View**, a modell a **Model**, míg a perzisztencia a **Persistence** névtérben helyezkedik el.
* A program szerkezetét két projektre osztjuk implementációs megfontolásból: a **Persistence** és **Model** csomagok a program felületfüggetlen projektjében, míg a **View** csomag a Windows Formstól függő projektjében kap helyet.

**Perzisztencia**:

* Az adatkezelés feladata a Tetris rácsával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
* A **GameGrid** osztály egy érvényes Tetris játékhoz szükséges rácsot biztosít kezdetben mindenhol üreset. **Rows** illetve **Columns** tulajdonságaival amiket a konstruktor paraméterben kap, tároljuk a rács sor és oszlopainak a számát.

(**IsEmpty**, **IsInside**, **IsRowFull**, **IsSmall,**) metódusokkal tudunk adatokat lekérni a rácsról. (**ClearFullRows**, **MoveRowDown, ClearRow**) műveletekkel pedig a rács sorait tudjuk módosítani.

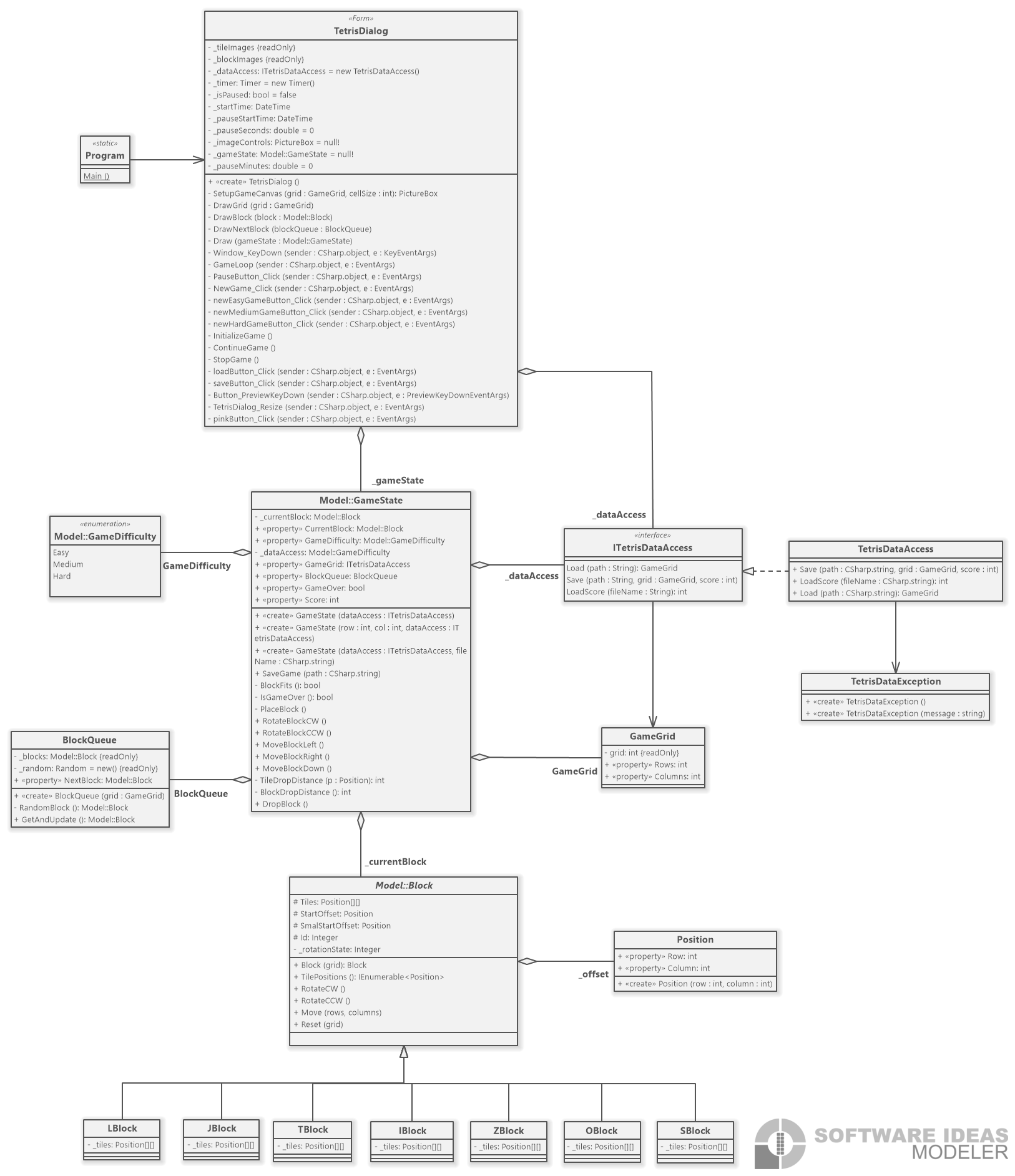
* A hosszú távú adattárolás lehetőségeit az **ITetrisDataAccess** interfész adja meg, amely lehetőséget ad a rács betöltésére (Load), valamint mentésére (Save). Illetve a pontszám betöltésére (LoadScore)
* Az interfészt szöveges fájl alapú adatkezelésre a **TetrisDataAccess** osztály valósítja meg. A fájlkezelés során fellépő hibákat a **TetrisDataException** kivétel jelzi.
* A program az adatokat szöveges fájlként tudja eltárolni, melyek az **txt** kiterjesztést kapják. Ezeket az adatokat a program első indításakor lehet betölteni, illetve a játék szüneteltetésekor ki lehet menteni az aktuális állapotot.
* A fájl első sora megadja az elmentett pontszámunkat. A következő sor a rács mentett állapotát adja meg, amely állhat 72, 144, 216 számból, az elmentett játék nehézségétől függően. A számok 0-7 közöttiek lehetnek, ahol 0 reprezentálja a még üres mezőt, a többi pedig az egyes blokkok típusát.

**Modell:**

* A modell lényegi részét a **GameState** osztály valósítja meg, amely szabályozza a játék tevékenységeit, valamint egyéb paramétereit, úgymint a pontszám (**Score**) és a játék végét (**GameOver**). Új játék kezdésére a konstruktor felel, valamint blokk megjelenésére a (**MoveBlockDown**). Új játéknál megadható a kiinduló rács mérete is, különben a legnagyobb rács generálódik.
* A jelenlegi blokk mozgatásáról és manipulálásáról a **MoveBlockLeft, MoveBlockRight, RotateBlockCW, DropBlock** metódusok felelnek.
* A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (**Konstruktor**) és mentésre (**SaveGame**).
* A játék nehézségét a **GameDifficulty** felsorolási típuson át kezeljük, és a **GameGrid** osztályban a rács méreteivel tároljuk az egyes nehézségeket.

**Nézet:**

* A nézetet a **TetrisDialog** osztály biztosítja, amely tárolja a modell egy példányát (**\_gameState**), valamint az adatelérés konkrét példányát (**\_dataAccess**).
* A játéktáblát egy dinamikusan létrehozott képmező (**\_imageControls**) reprezentálja. A felületen létrehozzuk a megfelelő gombokat, illetve feliratot, valamint dialógusablakokat, és a hozzájuk tartozó eseménykezelőket. A játéktábla generálását (**SetupGameCanvas**), illetve az értékek beállítását (**Draw**) külön metódusok végzik.
* A játék időbeli kezelését és a blokkok esését egy időzítő végzi (**\_timer**), amelyet mindig aktiválunk játék során, illetve inaktiválunk, amennyiben bizonyos menüfunkciók futnak.
* A program teljes statikus szerkezete a 2. ábrán látható.



**2. ábra: Az alkalmazás osztálydiagramja**

**Tesztelés:**

* A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **TetrisUnitTest** osztályban.
* Az alábbi tesztesetek kerültek megvalósításra:
* **SmallGridTest**, **MediumGridTest**, **LargeGridTest**: A rácsok méretei és tartalma ellenörzése.
* **NewSmallGameTest, NewMediumGameTest, NewHardGameTest**: Új játék kezdő tulajdonságainak tesztelése, pontszám, nehézség, rács állapota, betöltés.
* **CreateAndPlaceBlockTest**: Új blokk létrehozása és alulra lehelyezése. A rács új állapotának ellenörzése.
* **PlaceBlockLeft, PlaceBlockRight**: Egy új blokk teljesen bal vagy jobb szélre helyezése és annak tesztelése, hogy jó helyen állt meg a blokk.
* **GameOverTest**: Blokkok egymásra helyezése, hogy elérjük a játék végét. Annak tesztelése, hogy a játék leáll-e.