# 2. Beadandó feladat

## Készítette:

**Szigetközi Zsolt**

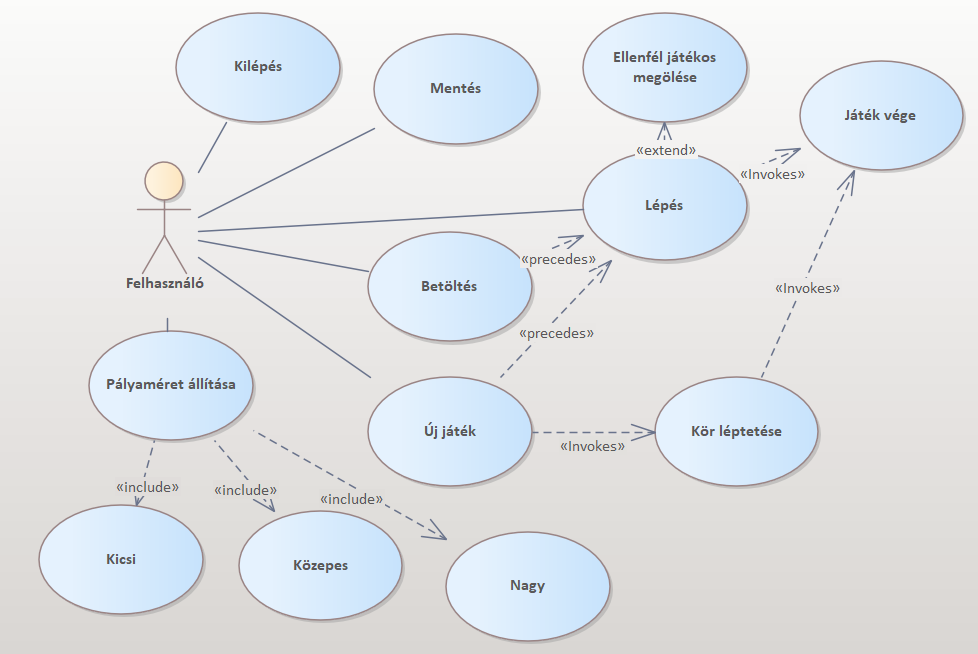
**E-mail:** [**szigetkozizsolt9@gmail.com**](mailto:szigetkozizsolt9@gmail.com)

## Feladat:

Készítsünk programot, amellyel a következő két személyes játékot játszhatjuk. Adott egy 𝑛 × 𝑛 mezőből álló tábla, amelyen a mezők két színt vehetnek fel spirális alakban (tradicionálisan pirosat, illetve zöldet), továbbá a középső mező szürke. Minden mezőn, kivéve a középsőn egy kaméleon helyezkedik el, amelynek színe megegyezik a mezővel, így minden játékos (𝑛 2 − 1)/2 kaméleonnal rendelkezik. A játékosok felváltva léphetnek. Egy kaméleonnal léphetünk egy szomszédos üres mezőre (vízszintesen, illetve függőlegesen), illetve átugorhatjuk az ellenfél kaméleonját (vízszintesen, illetve függőlegesen), amennyiben a rákövetkező mező üres. Az átugrott kaméleon lekerül a tábláról. A játék célja, hogy a másik játékos elveszítse az összes kaméleonját. A játékban a csavar, hogy a kaméleonok alkalmazkodnak a környezetükhöz. Amennyiben egy kaméleon egy másik színű mezőre ugrott, vagy lépett, akkor további 1 kör elteltével átszíneződik a másik színre (tehát a másik játékosé lesz). Ez alól kivétel a középső mező.  
A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (3 × 3, 5 × 5, 7 × 7), valamint játék mentésére és betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

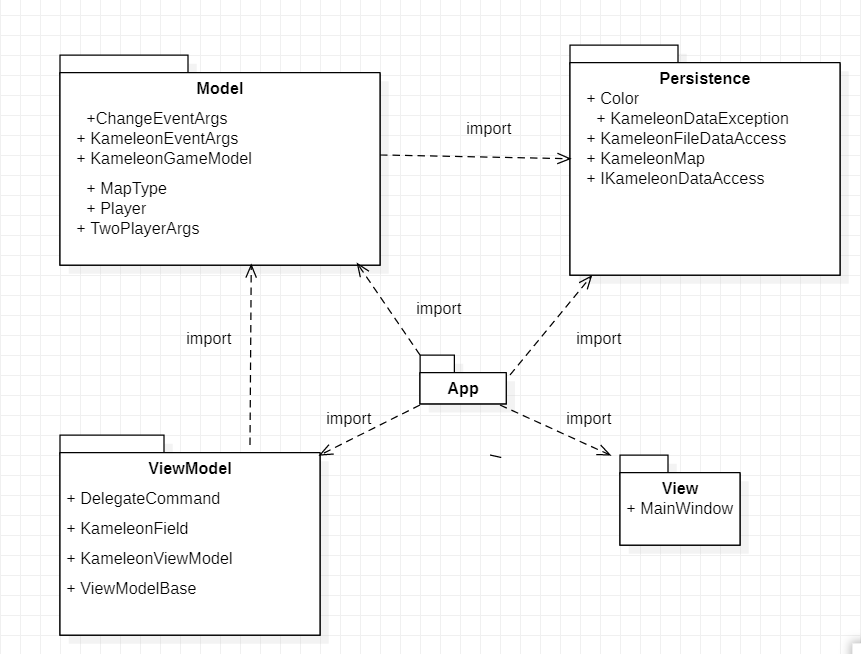
## Elemzés:

* A játékot három féle pályán lehet játszani: kicsi (3 x 3), közepes (5 x 5), nagy (7 x 7). A program indításkor a közepeset állítja be és automatikusan új játékot indít.
* A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
* Az ablabkan elhelyezünk egy menüt a következő pontokkal: File (Új játék, Játék mentése, Játék betöltése, Kilépés), Beállítások (3 x 3, 5 x 5, 7 x 7). Az ablak alján megjelenítünk egy státuszsort, amely az aktuális játékos körét mutatja.
* A játékbtáblát egy 5 x 5 (3 x 3, 7 x 7) nyomógombokból álló rács reprezentálja. A nyomógomb egérkattintás hatására kijelöli az adott kaméleont, a második kattintásnál pedig az adott üres mezőre lépteti azt. A program csak az aktuális játékos kaméleonjait engedi léptetni.
* A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak a nyertes játékos nevével (valamelyik játékosnak elfogytak a kaméleonjai). Szintén dialógusablakokkal végezzük el a mentést, illetve betöltést, a fájlneveket a felhasználó adja meg.
* A felhasználói esetek az 1. ábrán láthatóak.

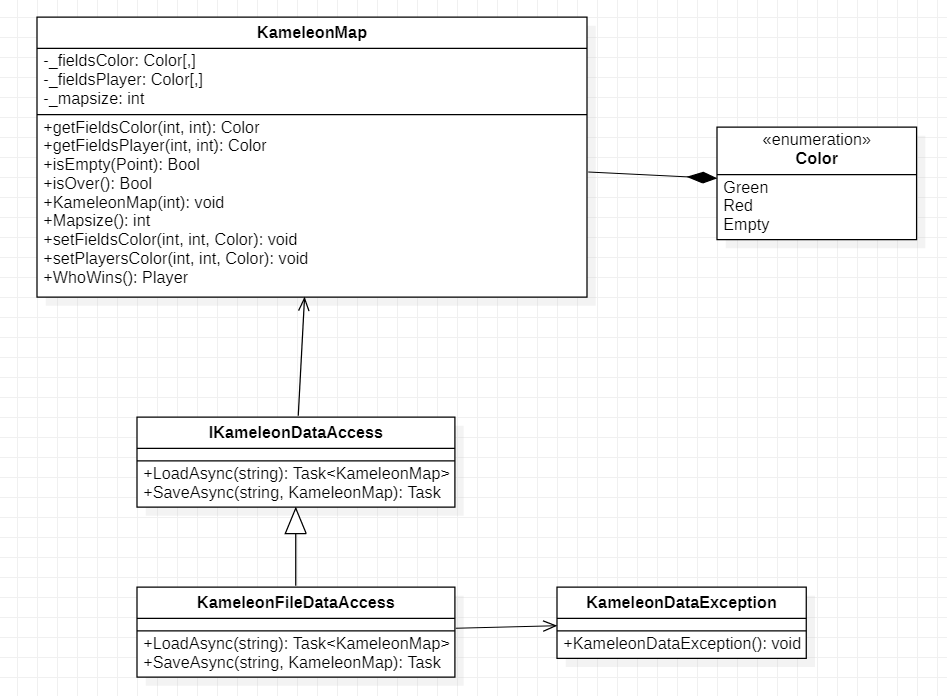


**1. ábra: Felhasználói esetek diagramja**

## Tervezés:

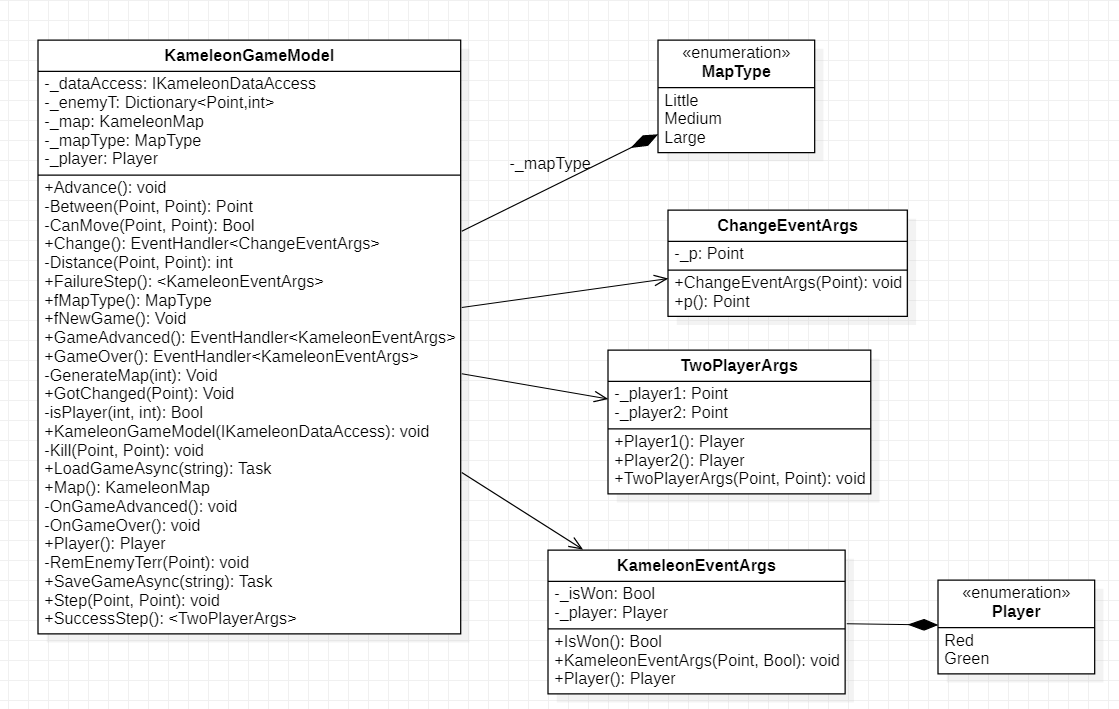
* Programszerkezet:
  + A programot MVVM architektúrában valósítjuk meg, ennek megfelelően **View**, **Model**, **ViewModel** és **Persistence** névtereket valósítunk meg az alkalmazáson belül. A program környezetét az alkalmazás osztály (**App**) végzi, amely példányosítja a modellt, a nézetmodell és 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.

**2. ábra: Az alkalmazás csomagdiagramja**

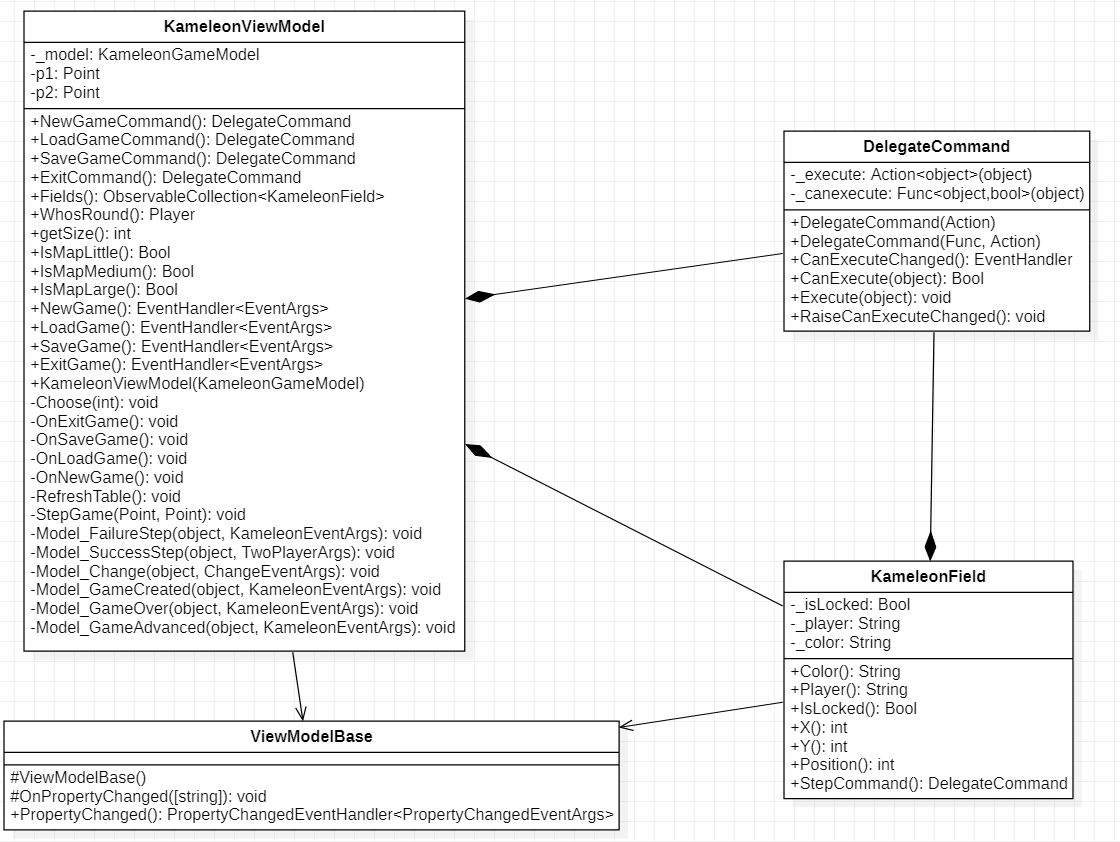
* Perzisztencia:
  + Az adatkezelés feladata a Kaméleon pályával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
  + A **KameleonMap** osztály egy érvényes Kaméleon pályát biztosít (azaz mindig ellenőrzi a beállított értékek), ahol minden mezőre ismert a színe (**Piros, Zöld, Üres**) (**\_fieldsColor**), illetve, hogy a mezőkön milyen játékos tartozkódik (**Piros, Zöld, Üres**) (**\_fieldsPlayer**). A konstruktorban tudjuk megadni a **\_mapSize** értékét paraméterként. Az pálya lehetőséget ad az állapotok lekérdezésére (**Mapsize, getFieldsColor, getFieldsPlayer, isOver, whoWins, isEmpty**), valamint a szabályos léptetésre (**setFieldsPlayer**), illetve a direkt beállítás elvégzésére (**\_setFieldsColor**).
  + A hosszú távú adattárolás lehetőségeit az **IKameleonDataAccess** 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 **KameleonFileDataAccess** osztály valósítja meg. A fájlkezelés során fellépő hibákat a **KameleonDataException** 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 pálya méretét (3, 5, 7). A fájl többi része kettő darab méret x méret mátrix, ahol az első mátrix a pálya színeit adja meg, a második pedig a mezőn lévő játékosokat (**Piros, Zöld, Üres**).

**3. ábra: Persistence csomag osztálydiagramja**

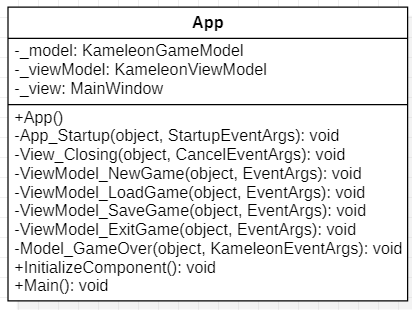
* Modell:
  + A modell lényegi részét a KameleonGameModel osztály valósítja meg, amely szabályozza a pálya tevékenységet, valamint a játék egyéb paramétereit, úgymint az aktuális játékos (**\_player**) (**Red, Green**), illetve a pálya típusa (**\_mapType**) (**Little, Medium, Large)** és azon kaméleonok helyzete, amik ellenséges mezőn tartozkódnak (**\_enemyT**). A típus lehetőséget ad az új játék kezdésére, valamint lépésre (**Step**). Új játéknál meg kell adnunk a pálya méretét. A kör végét és a játék előrehaladását is jelezzük (**Advance**).
  + A játékállapot változásáról a **GameAdvanced** esemény, míg a játék végéről a **GameOver** esemény tájékoztat. Az események argumentuma (**KameleonEventArgs**) tárolja a győzelem állapotát, a lépések számát, valamint a játékidőt.
  + Ha egy játékos színt vált akkor a Change esemény váltódik ki és egy **ChangeEventArgs** -et kapunk.
  + Sikeres lépés esetén a **SuccessStep**, érvénytelen lépés esetén a **FailureStep** váltódik ki.
  + A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (LoadGameAsync) és mentésre (SaveGameAsync).
  + A pálya méretét a **MapType** felsorolási típuson át kezeljük.

**4.ábra: Model csomag osztálydiagramja**

* Nézetmodell:
  + A nézetmodell megvalósításához felhasználunk egy általános utasítás (**DelegateCommand**), valamint egy ős változásjelző (**ViewModelBase**) osztályt.
  + A nézetmodell feladatait a **KameleonViewModel** 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.
  + A játékmező számára egy külön mezőt biztosítunk (**KameleonField**), amely eltárolja a pozíciót, szöveget, engedélyezettséget, valamint a lépés parancsát (**StepCommand**). A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (**Fields**).

**5.ábra: 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.

**6.ábra: A vezérlés osztálydiagramja**

## Tesztelés:

* A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **KameleonGameModelTest** osztályban.
* **KameleonGameModelNewGameMediumMapTest,  
  KameleonGameModelNewGameLittleMapTest,  
  KameleonGameModelNewGameLargeMapTest**: Új játék indítása, a mezők kitöltése és megnézi, hogy a megfelelő mennyiségű kaméleon van – e a pályán.
* **KameleonGameModelStepTest**: Lépés ellenőrzése, az aktuális játékos kaméleonjával akarunk – e lépni, illetve olyan mezőre, ahova lehet lépni.
* **KameleonGameModelAdvanceTest**: A játék előrehaladásának ellenőrzése, megváltozott – e az aktuális játékos.
* **KameleonGameModelLoadTest**: A játék modell betöltésének tesztelése mockolt perzisztencia réteggel.