# 3. Beadandó feladat dokumentáció

**Készítette**:

Szalai Patrik

**E-mail:**

[szalai.patrik14@gmail.com](mailto:szalai.patrik14@gmail.com)

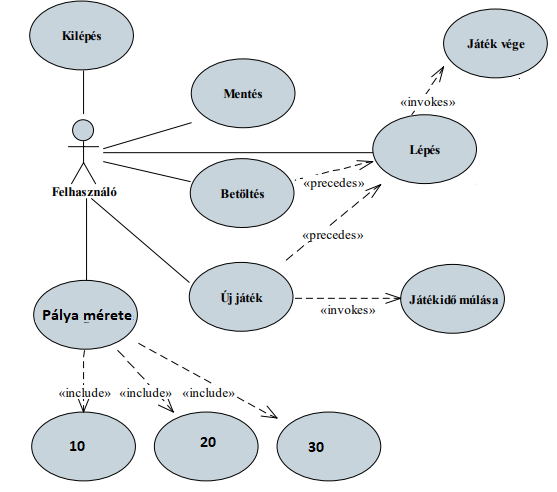
**Feladat:**

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:**

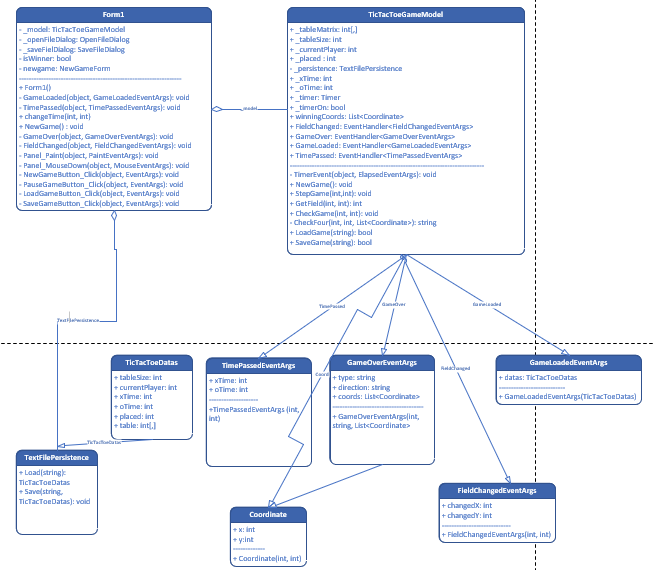
* A játékot 3 féle méretű pályán játszhatjuk
  + 10x10, 20x20, 30x30 mező
* A feladatot Windows Form grafikus felülettel valósítottam meg
* Az ablak tetején találhatók gombok, melyeknek funkciói:
  + Új játék, mentés, betöltés és szünet
* Fent található még a játékosok gondolkodási idejét mutató szövegdobozok, valamint az éppen következő játékost is mutatja
* A játéktábla a beépített rajzoló függvénnyel jött létre, ahogy a bábuk is
  + egér kattintásra figyelnek (event listener)
* A játék feldob egy dialógusablakot, ha vége a játéknak (valaki nyert/ betelt a pálya), majd új játékot indít
* Felhasználói esetek:



**Tervezés:**

* Programszerkezet:
  + A programot háromrétegű architektúrában valósíjuk meg. A megjelenítés a **View**, a modell a **Model**, míg a perzisztencia a **Persistence** névtérben helyezkedik el.
* Perzisztencia:
  + Betöltésnél tetszőleges méretű pályát tölthetünk be, ennek függvényében változik az adott méret
  + Eltároljuk a pálya méretét, a pályán elhelyezett bábukat, az aktuális játékost, a lerakott bábuk számát, valamint az egyes játékosok gondolkodási idejét
    - a pálya egészeket tartalmaz:
      * 0 - senki
      * 1 - X
      * 2 – O
      * mérete akkora, amekkora a pályaméret négyzete
  + Szöveges állományban tároljuk az értékeket *space*-ekkel elválasztva
  + Ha nem sikerült az írás/olvasás, akkor ennek megfelelő kivételt (Exception) dobunk, majd lekezeljük a hibát, így védve a programot
  + A fájl felépítése sorban:
    - táblaméret
    - aktuális játékos
    - X játékos gondolkodási ideje
    - O játékos gondolkodási ideje
    - lerakott bábuk száma
    - maga a pálya, és a rajta levő bábuk/helyek
* Modell:
  + A modell lényegi részét a **TicTacToeGameModel** publikus osztály valósítja meg
    - ez szabályozza a gondolkodási időket
      * **\_xTime, \_oTime**
    - lerakott bábuk számát
      * **\_placed**
  + Lehetőséget ad új játék kezdésére
    - **NewGame()**
  + Bábu lerakására
    - **StepGame()**
  + Valamint minden lépés után megvizsgálja, hogy van-e még hely a pályán, valamint hogy valakinek kigyűlt-e már a győzelemhez szükséges 4 bábu egymás mellett
    - ha valaki nyert, elküldi a **GameOverEventArgs** segítségével, hogy ki nyert, ezt a **Nézet** lekezeli
    - ha megtelt a pálya, akkor hasonló esemény történik
  + Itt hívjuk meg a **Perzisztencia** függvényeit
    - **SaveGame, LoadGame**
* Nézet:
  + A nézetet **Form1** osztály biztosítja, amely tárolja a modell példányát (**\_model)**, valamint az adatelérés példányát is átküldi a modellnek (**TextFilePersistence**)
  + Beépített rajzolófüggvény segítségével valósítottam meg a pálya kirajzolását, valamit a bábuk kinézetét
    - egy egér kattintást figyelő EventHandler figyeli és küldi el a modellnek az adott eseményt

**UML:**



**Tesztelés:**

* A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **UnitTestProject1** osztályban
* Az alábbi tesztek kerültek megvalósításra:
  + **InitTest()**
    - inicializálás
  + **NewGameTest()**
    - új játék ellenőrzése, tábla törlése
  + **StepTest()**
    - következő játékosok váltakozása
    - bábu pályára rakása
  + **WinTest()**
    - 4 azonos bábu egy sorban/átlóban
  + **SaveTest()**
    - sikeres mentés
  + **LoadTest()**
    - sikeres betöltés
  + **NoMoreSpaceInColumnTest()**
    - ha nincs hely, ahova rakni szeretnénk, akkor nem számít lépésnek
    - marad az aktuális játékos a következő
    - pályán változás nincs