# Beadandó feladat dokumentáció

## Készítette:

Szabo Adorja n Otto E-mail:

[s4i92x@inf.elte.hu](mailto:s4i92x@inf.elte.hu)

## Feladat:

Ke szí tsu nk programot, amellyel a ko vetkezo ja te kot ja tszhatjuk. Adott egy 𝑛 × 𝑛

elembo l a llo ja te kpa lya, amely labirintuske nt e pu l fel, azaz fal, illetve padlo mezo k tala lhato ak benne, illetve egy kija rat a jobb felso sarokban. A ja te kos ce lja, hogy a bal also sarokbo l indulva mine l elo bb kijusson a labirintusbo l. A labirintusban nincs

vila gí ta s, csak egy fa klya t visz a ja te kos, amely a 2 szomsze dos mezo t vila gí tja meg (azaz egy 5 × 5-o s ne gyzetet), de a falakon nem tud a tvila gí tani. A ja te kos figura ja kezdetben a bal also sarokban helyezkedik el, e s ví zszintesen, illetve fu ggo legesen mozoghat (egyese vel) a pa lya n. A pa lya k me rete t, illetve fele pí te se t (falak, padlo k) ta roljuk fa jlban. A program legala bb 3 ku lo nbo zo me retu pa lya t tartalmazzon. A

program biztosí tson leheto se get u j ja te k kezde se re a pa lya kiva laszta sa val, valamint ja te k szu neteltete se re (ekkor nem telik az ido , e s nem le phet a ja te kos), tova bba

ismerje fel, ha ve ge a ja te knak. A program ja te k ko zben folyamatosan jelezze ki a ja te kido t.

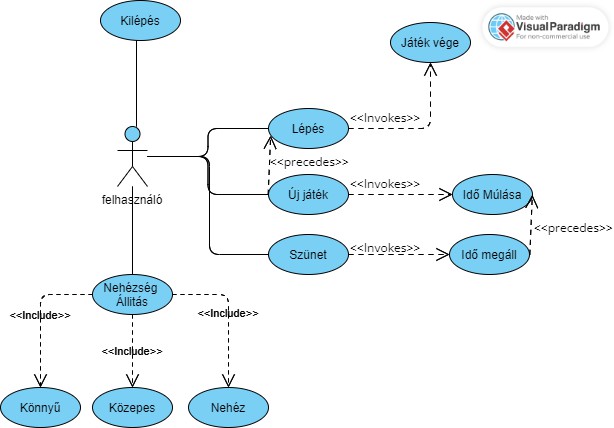
# Elemzés:

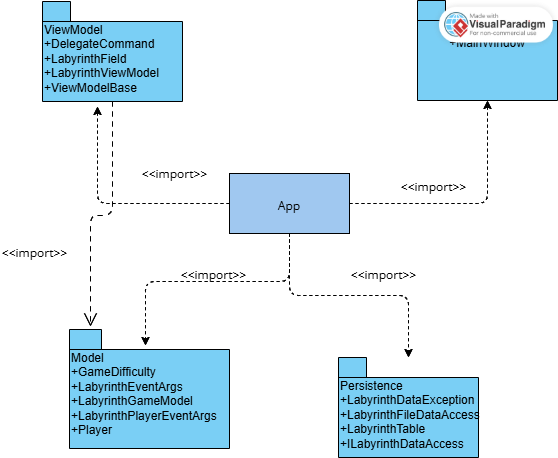
* A játékot három nehézségi szinttel játszhatjuk: könnyű (10x10es pálya), közepes (11x11es pálya), nehéz (12x12es pálya). A program indításkor könnyű nehézséget állít be.
* A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
* Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (Új játék, Szünet, Kilépés), Beállítások (Könnyű, Közepes,

Nehéz). Az ablak alján megjelenítünk egy státuszsort, amely a hátralévő időt jelzi. priossal a fal,zöldel az út és kékkel a játékos van jelölve.

* A játéktáblát egy n × n Label-ekből álló rács reprezentálja. A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak

(a jobb felső sarokba értünk). Szintén dialógusablakokkal történik a szünet is.

* 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 Felhasználó Betöltés Mentés Új játék Kilépés Lépés Nehézség állítása Könnyű Közepes Nehéz Játék v ége Játékidő múlása «precedes» «precedes» «invokes» «include» «invokes» «invokes» «include» «include» Eseményvezérelt alkalmazások 2024/2025 őszi félév 3 felületfüggetlen projektjében, míg a ViewModel és View csomagok a WPF függő projektjében kapnak helyet.  
Perzisztencia:

* + Az adatkezele s feladata a Labyrinth ta bla val kapcsolatos informa cio k ta rola sa, valamint a beto lte s/mente s biztosí ta sa.
  + A **LabyrinthTable** oszta ly egy Boolean-okal kito lto tt labirintust biztosit. A pa lya majd egy txt file-bo l lesz beto ltve. A konstruktorba megkell adni a pa lya me rete t amit automatikusan felto lt *false*-al.A ta bla leheto se get az

e rte kek va ltoztata sa ra/leke rdeze se re.

•

* + A hosszu ta vu adatta rola s leheto se geit az **IlabyrinthDataAccess** interfe sz adja meg, amely leheto se get ad a ta bla beto lte se re LoadEasy,LoadMedium,LoadHard ,Load.
  + Az interfe szt szo veges fa jl alapu adatkezele sre a **LabyrinthFileDataAccess** oszta ly valo sí tja meg. A fa jlkezele s sora n felle po hiba kat a

**LabyrinthDataException** kive tel jelzi.

* + A program az adatokat szo veges fa jlbo l nyeri ki.
    - A fa jl elso sora megadja a ta bla me rete t, majd nxn me retben Boolean e rte kek vannak.
  1. **ábra: Az alkalmazás csomagdiagramja**

Modell:

* + A modell le nyegi re sze t a **LabyrinthGameModel** oszta ly valo sí tja meg, amely szaba lyozza a ta bla teve kenyse geit, valamint a ja te k egye b parame tereit, u gymint az ido (**\_gameTime**) e s az hogy a ja te kos ce lba e rt-e **(\_isGameOver)**. A tí pus leheto se get ad u j ja te k kezde se re (NewGame), valamint le pe sre (**Step**). Ú j ja te kna l megadhato ,hogy melyik pa lya legyen.
  + A ja te ka llapot megva ltoza sa ro l (ve ge van-e, ha tra le vo ido ) a **GameAdvanced** eseme ny, mí g a ja te k ve ge ro l a **GameOver** eseme ny ta je koztat. Az eseme nyek argumentuma (**LabyrinthEventArgs**) ta rolja a gyo zelem a llapota t, valamint a ja te kido t.
  + A modell pe lda nyosí ta skor megkapja az adatkezele s felu lete t, amelynek segí tse ge vel leheto se get ad beto lte sre (**LoadGame**).

•

* + A ja te k nehe zse ge t a **GameDifficulty** felsorola si tí puson a t kezelju k.
  + Ne zet:
  + A ne zetet a Labyrinth**WinForm** oszta ly biztosí tja, amely ta rolja a modell egy pe lda nya t (**\_model**), valamint az adatele re s konkre t pe lda nya t (**\_dataAccess**).
  + A ja te kta bla t egy dinamikusan le trehozott (**\_LabelGrid**) reprezenta lja. A felu leten le trehozzuk a megfelelo menu pontokat, illetve sta tuszsort, valamint dialo gusablakokat, e s a hozza juk tartozo eseme nykezelo ket. A ja te kta bla genera la sa t (**GenerateTable)** ve gzi.
  + A ja te k ido beli kezele se t egy ido zí to ve gzi (**\_timer**), amelyet mindig aktiva lunk ja te k sora n, illetve inaktiva lunk, amennyiben bizonyos menu funkcio k futnak.
  + A program teljes statikus szerkezete a 3. a bra n la thato .

A képen szöveg, képernyőkép, Párhuzamos, Betűtípus látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, diagram, Párhuzamos látható

Automatikusan generált leírás

* 1. **ábra:Nézet model**• 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 (6. ábra):   
     • 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

**Tesztelés:**

* A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a

**LabyrinthGameModelTest** osztályban.

* Az alábbi tesztesetek kerültek megvalósításra:
* **LabyrinthGameModelNewGameMediumTest,LabyrinthGameModelNewGameEasyTest, LabyrinthGameModelNewGameHardTest,LabyrinthGameModelStepTest, LabyrinthGameModelAdvanceTimeTest,**

**LabyrinthGameModelNewGameMediumTest:** Új játék indítása, megfelelő tábla és nehézség ellenőrzése a nehézségi

fokozat

függvényében**(LabyrinthGameModelNewGameEasyTest,LabyrinthGameModelNewGameHar**

**dTest** ugyan ez).

* **LabyrinthGameModelStepTest**: Az üres pályán be megyünk a célba és leellenőrizzük, hogy a\_model IsWon adattagja megfelelően változik-e
* **LabyrinthGameModelAdvanceTimeTest**: A játékbeli idő kezelésének

ellenőrzése, beleértve azt hogy nem telik az idő hogy ha már beértünk a célba.