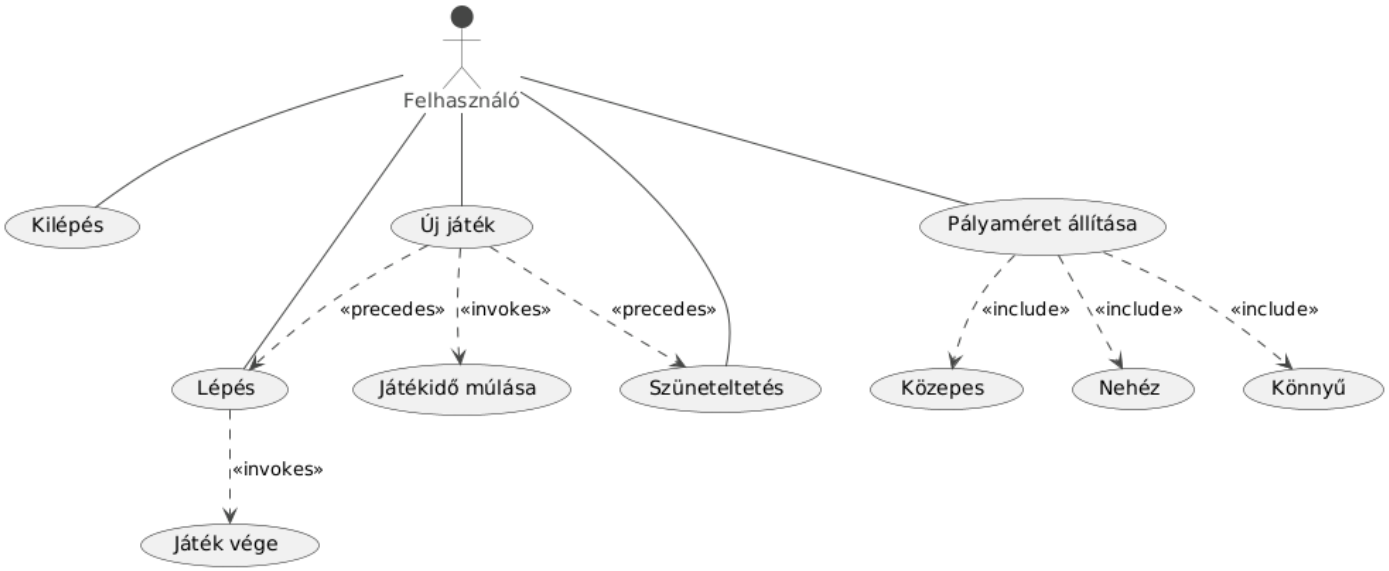
1. Beadandó feladat dokumentáció

Készítette:
Ferenczy Kata
Neptun kód: G09BOX

Feladat:
Készítsünk programot, amellyel a következő játékot játszhatjuk. Adott egy $n \times n$ elemből álló játékpálya, amely labirintusként épül fel, azaz fal, illetve padló mezők találhatóak benne, illetve egy kijárat a jobb felső sarokban. A játékos célja, hogy a bal alsó sarokból indulva minél előbb kijusson a labirintusból. A labirintusban nincs világítás, csak egy fáklyát visz a játékos, amely a 2 szomszédos mezőt világítja meg (azaz egy 5 × 5-ös négyzetet), de a falakon nem tud átvilágítani. A játékos figurája kezdetben a bal alsó sarokban helyezkedik el, és vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán. A pályák méretét, illetve felépítését (falak, padlók) tároljuk fájlban. A program legalább 3 különböző méretű pályát tartalmazzon. A program biztosítson lehetőséget új játék kezdésére a pálya kiválasztásával, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet a játékos), továbbá ismerje fel, ha vége a játéknak. A program játék közben folyamatosan jelezze ki a játékidőt.

Elemzés:

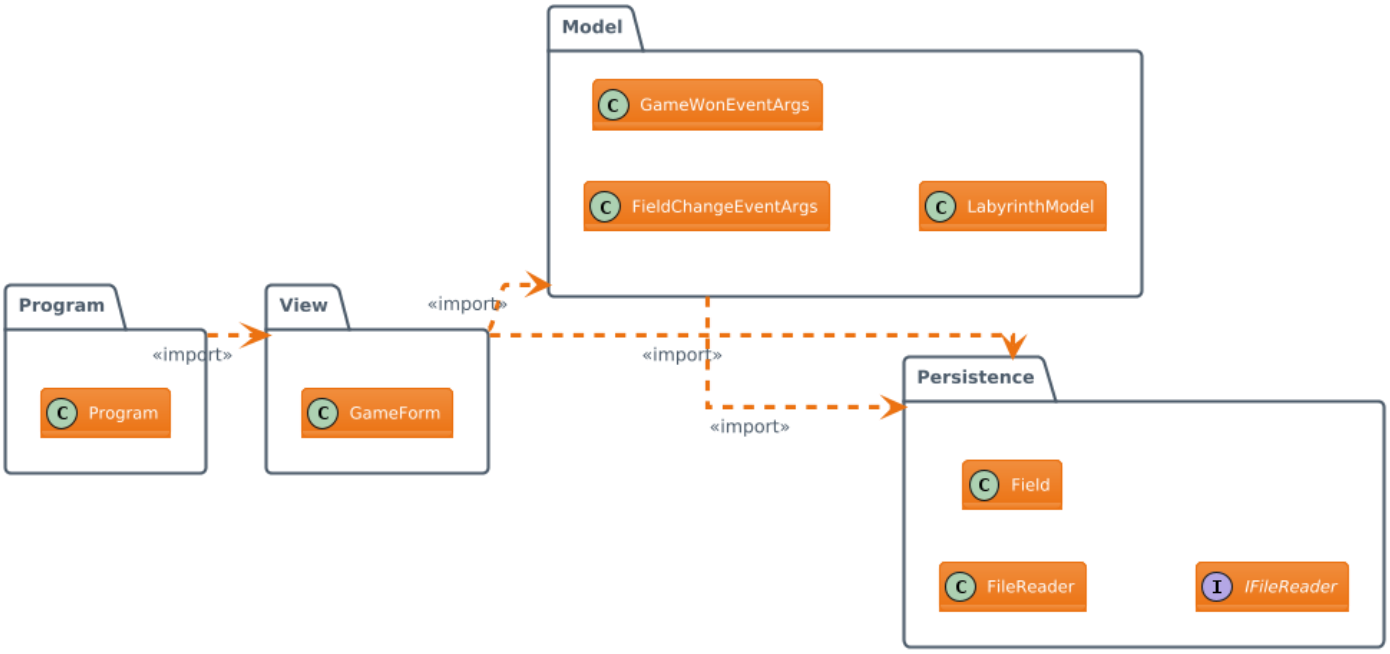
- A játékot három pályamérettel játszhatjuk: könnyű (5x5), közepes (10x10) és nehéz (15x15). A program indításakor a felhasználó egy Combo Boksz segítségével választja ki a nehézséget és a Start gombra kattintva elindítja a játékot.
- A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt egyetlen File (Új játék) menüponttal. Az ablakban megjelenik egy státuszsor, ami az eltelt időt jelzi, valamint egy szüneteltető gomb.
- A játéktáblát egy, a kiválasztott méretnek megfelelő, nyomógombokból álló rács reprezentálja. A nyomógomb egérkattintás hatására megváltoztatja a játékos karakterének pozícióját. A változtatást csak akkor engedjük, ha a megnyomott gomb közvetlenül a karakter addigi pozíciója mellett van és a megnyomott gomb mezője padló.
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak, azaz elértük a trófea mezőjét (jobb felső sarok). Szintén dialógusablakokkal jelezzük, ha a felhasználó rossz mezőre akar lépni.
- A felhasználói esetek az első ábrán láthatóak.



1. ábra: Felhasználói esetek diagramja

Tervezés:

- Programszerkezet:
  - A programot háromrétegű architektúrában valósítjuk meg. A megjelenítés a **View**, a modell a **Model**, míg a perzisztencia a **Persistence** névtérben helyezkedik el. A program csomagszerkezete a 2. ábrán látható.
  - A program szerkezetét két projektre osztjuk implementációs megfontolásból: a **Persistence** és a **Model** csomagok a program felületfüggetlen projektjében, míg a **View** csomag a Windows Formstól függő projektjében kap helyet.



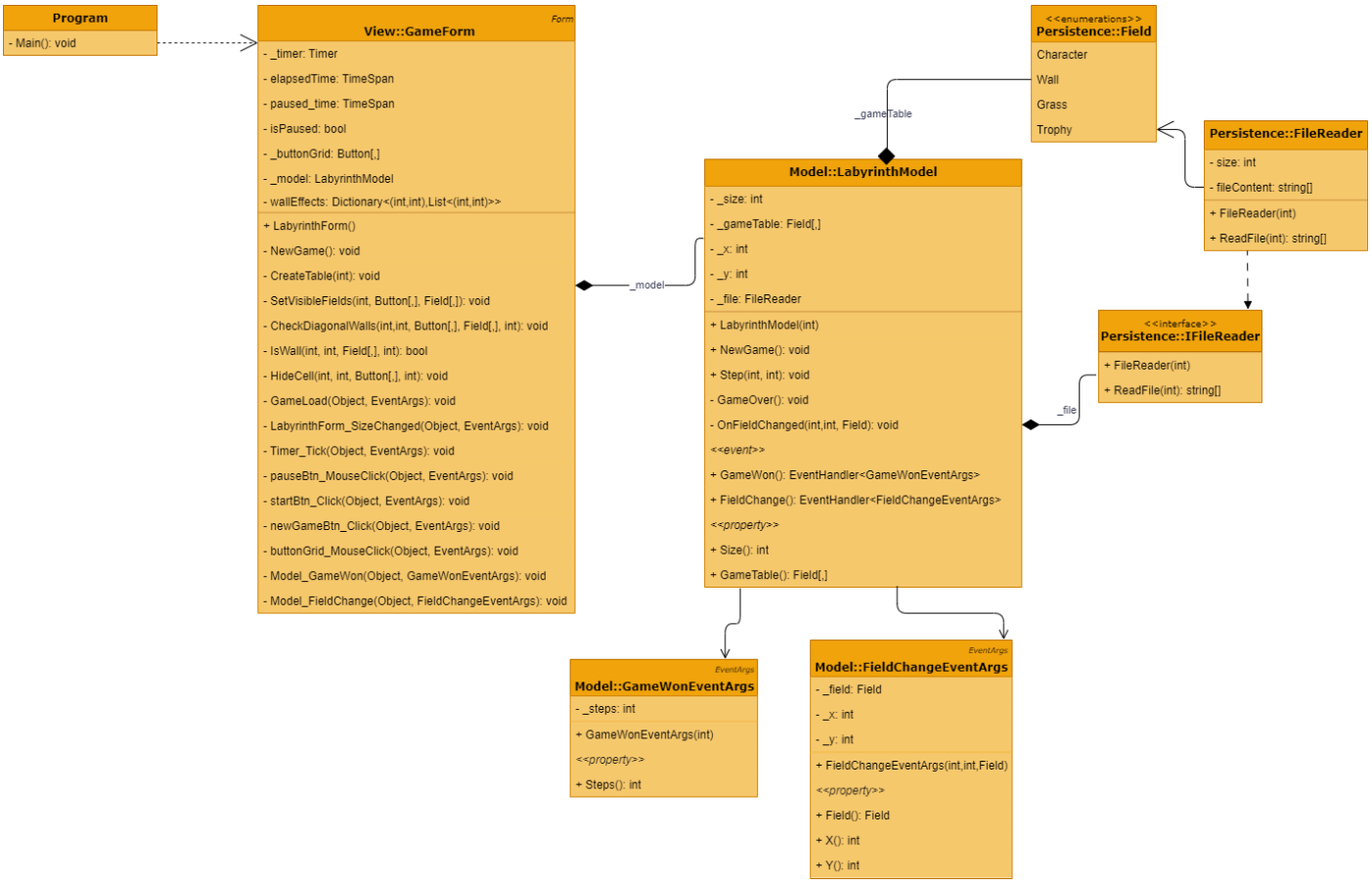
2. ábra Az alkalmazás csomagdiagramja

- Perzisztencia:
  - Az adatkezelés feladata a Labirintus táblával kapcsolatos információk tárolása.

    - Az enum típusú **Field** osztály négy különböző értéket definiál, amelyek a labirintus mezőit reprezentálják. A **Character** érték a játékos pozícióját jelöli, a **Wall** a falmezőket, amelyeken nem lehet áthaladni, a **Grass** érték a szabadon járható területet jelöli, míg a **Trophy** a célt, azaz a labirintus végét.
    - A fájlbeolvasást az **IFileReader** interfész adja meg.
    - Az interfész szöveges fájl alapú adatkezelését a **FileReader** osztály valósítja meg, amely a fájl tartalmát méret (**size**) alapján választja ki, és a fájl sorait egy string tömbben (**fileContent**) tárolja. A fájlok tartalmazzák a megfelelő méretű mátrixot, ahol a 0-ás értékek a padló mezőket, az 1-es értékek a falmezőket jelölik, míg a 2-es érték a kijáratot, a 3-as érték a karakter pozícióját jelöli. A fő metódus a **ReadFile**, ami a paraméter értéke alapján kiválasztja és beolvassa a megfelelő fájlt („easy.txt”, „medium.txt”, „hard.txt”). Ha a méret más, kivételt dob: **InvalidDataException**. Ha bármilyen hiba történik a fájl megnyitása során, egy általános kivételt dob, amely tartalmazza a hibaüzenetet.
- Modell:
    - A modell lényegi részét a **LabyrinthModel** osztály valósítja meg. A modell a játéktáblát (**_gameTable**), a játékos mozgását (**_x**, **_y** koordináta értékekkel) és a játékmenet eseményeit (**GameWon**, **FieldChange**) kezeli. Fő metódusai:
        - **NewGame**, ami a **FileReader** segítségével beolvassa a megfelelő fájlt és feltölti a játéktáblát **Field** értékekkel.
        - **Step**, ami kezeli a karakter lépéseit a paraméterben kapott koordinátájú mezőre. Ellenőrzi, hogy a lépés érvényes-e, majd frissíti a játéktáblát. Ha a karakter a trófea mezőre lép, véget ér a játék.
        - A **GameOver** metódus meghívja a **GameWon** eseményt, jelezve, hogy véget ért a játék.
        - Az **OnFieldChanged** meghívja a **FieldChange** eseményt, amikor egy mező értéke megváltozik (pl. a karakter új mezőre lép).
    - A **FieldChangeEventArgs** osztály egy esemény argumentum osztály, amely akkor hívódik meg, amikor egy mező megváltozik a játéktáblán. A mező változásához kapcsolódó adatokat továbbítja (**Field**, **X**, **Y**).
    - A **GameWonEventArgs** osztályt akkor használjuk, amikor a **LabyrinthModel** osztály meghívja a **GameWon** eseményt a játék befejezésekor, jelezve, hogy a játékos megnyerte a játékot.
- Nézet:
    - A nézetet a **GameForm** osztály biztosítja, amely tárolja a modell egy példányát (**_model**).
    - A játéktáblát egy dinamikusan létrehozott gombmező (**_buttonGrid**) reprezentálja. A felületen létrehozzuk a megfelelő menüpontokat, illetve státuszsort, valamint dialógusablakokat, és a hozzájuk tartozó eseménykezelőket. A játéktábla generálását (**CreateTable**), illetve a látható mezők beállítását (**SetVisibleFields**) külön metódusok végzik.

- A játék időbeli kezelését egy időzítő végzi (**_timer**), amelyet mindig aktiválunk játék során, illetve inaktiválunk, amennyiben bizonyos menüfunkciók futnak.

- A program teljes statikus szerkezete a 3. ábrán látható.



3. ábra: Az alkalmazás osztálydiagramja

Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **LabyrinthModelTest** osztályba.
- Az alábbi tesztesetek kerültek megvalósításra:
  - **LabyrinthStepTest**: Játékbeli lépés hatásainak ellenőrzése, játék megkezdése után. Érvénytelen lépések és esemény kiváltásának ellenőrzése.
  - **LabyrinthGameOverTest**: A játék végét jelző esemény ellenőrzése.
  - **LabyrinthFieldChangeTest**: A megváltozott mezőket jelző esemény ellenőrzése.