Készítette

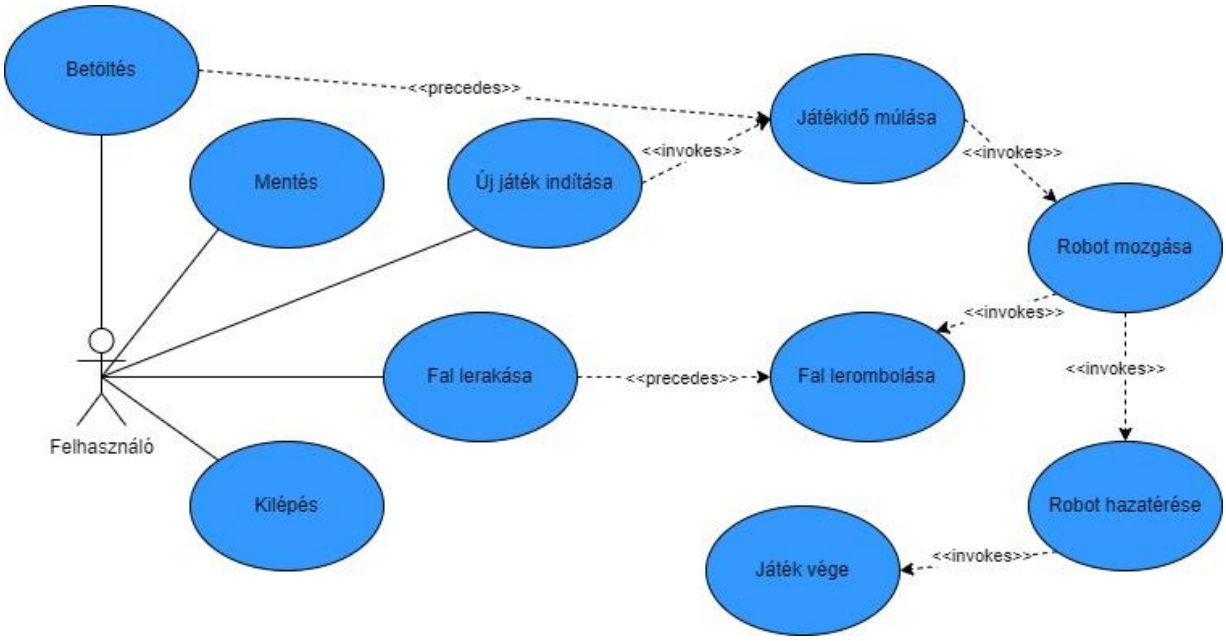
Orosz Zoltán
E-mail: ditp5v@inf.elte.hu
Csoportszám: 13

Feladat

Készítsünk programot, amellyel a következő játékot játszhatjuk.
Adott egy $n \times n$ mezőből álló játékpálya, amelyben egy elszabadult robot bolyong, és a feladatunk az, hogy betereljük a pálya közepén található mágnes alá, és így elkapjuk.
A robot véletlenszerű pozícióban kezd, és adott időközönként lép egy mezőt (vízszintesen, vagy függőlegesen) úgy, hogy általában folyamatosan előre halad egészen addig, amíg falba nem ütközik. Ekkor véletlenszerűen választ egy új irányt, és arra halad tovább. Időnként még jobban megkergül, és akkor is irányt vált, amikor nem ütközik falba.
A játékos a robot terelését úgy hajthatja végre, hogy egy mezőt kiválasztva falat emelhet rá. A felhúzott falak azonban nem túl strapabíróak. Ha a robot ütközik a fallal, akkor az utána eldől. A ledőlt falakat már nem lehet újra felhúzni, ott a robot később akadály nélkül áthaladhat.
A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (7 × 7, 11 × 11, 15 × 15), valamint játék szüneteltetésére (ekkor nem telik az idő, nem lép a robot, és nem lehet mezőt se kiválasztani). Ismerje fel, ha vége a játéknak, és jelenítse meg, hogy milyen idővel győzött a játékos. A program játék közben folyamatosan jelezze ki a játékidőt. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére.

A feladat elemzése

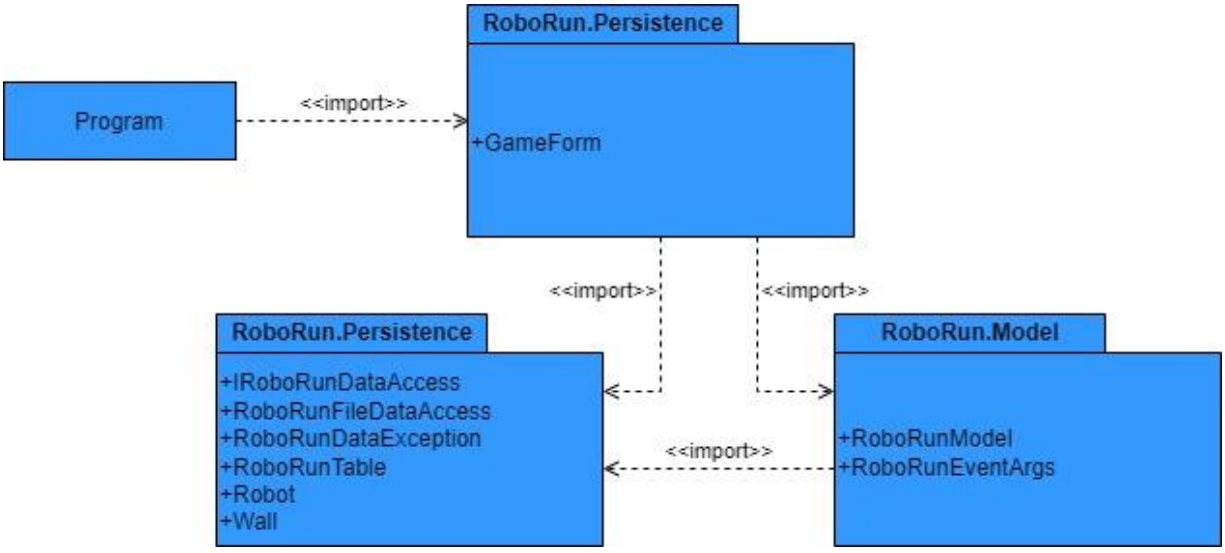
- A játékot három pályamérettel játszhatjuk: (7 × 7), (11 × 11), (15 × 15). A program indításkor (11 x 11)-es méretet állít be, és automatikusan új játékot indít.
- A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (Új játék, Játék betöltése, Játék mentése, Kilépés), Beállítások ((7 × 7), (11 × 11), (15 × 15)). Az ablak alján megjelenítünk egy státuszsort, amely a hátralévő időt jelzi.
- A játéktáblát a beállításoknak megfelelő méretű nyomógombokból álló rács reprezentálja. A nyomógomb egérkattintás hatására lerak egy falat az adott területre. Falat a robot vagy egy már lent lévő fal pozíciójára nem lehet lerakni. Lent lévő fal pozíciójára később, esetleges ledőlés esetén sem lehet már új falat lerakni.
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (sikerült beterelni a robotot a pálya közepén található mágnes alá). Szintén dialógusablakkal végezzük el a mentést, illetve betöltést is, a fájlneveket a felhasználó adja meg.



1. ábra: Felhasználói esetek diagramja

Tervezés

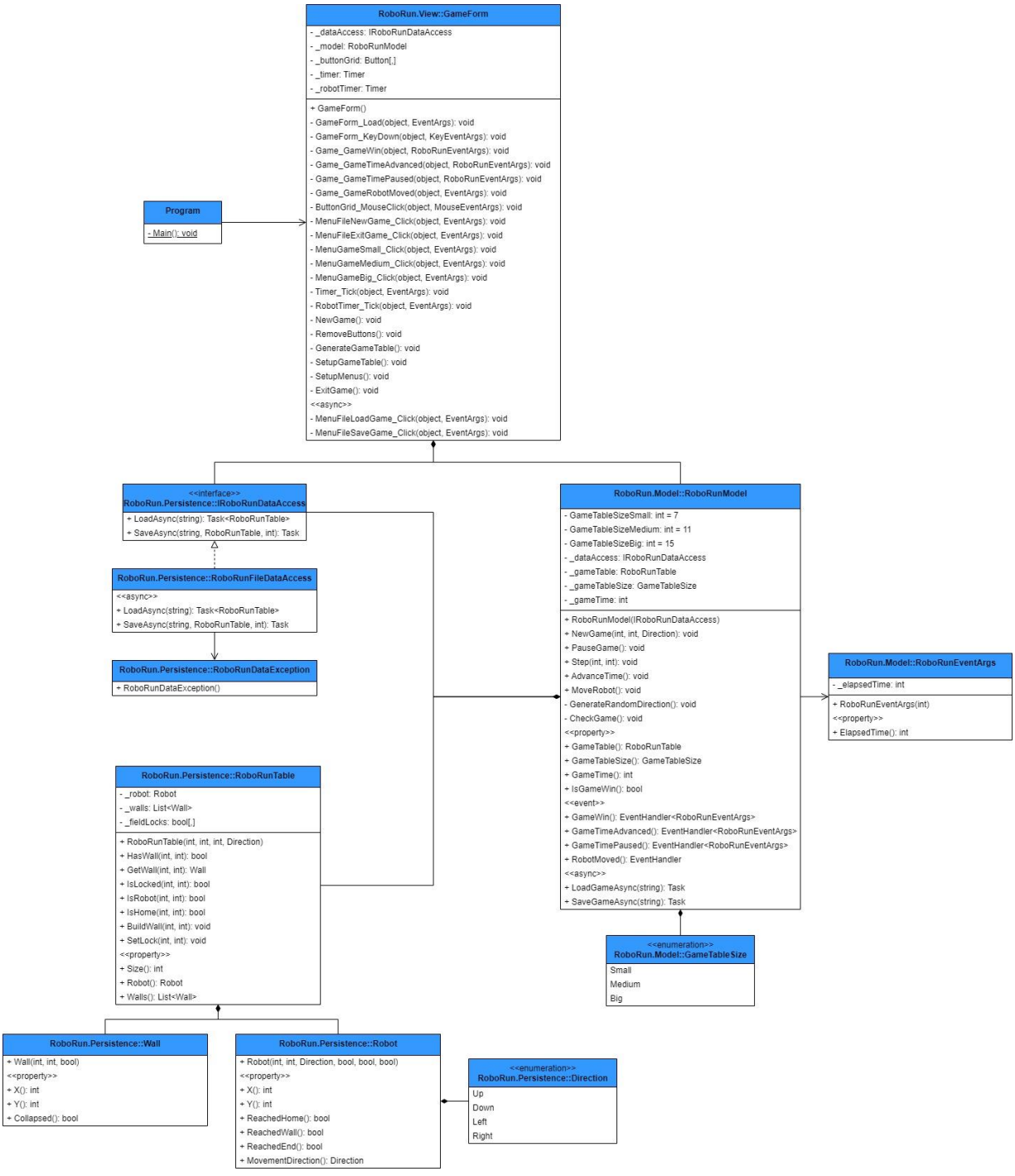
- Programszerkezet:
  - A programot háromrétegű architektúrában valósítjuk meg. A megjelenítés a **RoboRun.View**, a modell a **RoboRun.Model**, míg a perzisztencia a **RoboRun.Persistence** névtérben helyezkedik el. Továbbá a rétegeket külön projektként adjuk hozzá az újrafelhasználhatóság érdekében.



2. ábra: Az alkalmazás csomagdiagramja

- Perzisztencia:
  - Az adatkezelés feladata a RoboRun táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
  - A **RoboRunTable** osztály egy érvényes RoboRun táblát biztosít (azaz mindig ellenőrzi a beállított értékeket), ahol minden mezőről ismert a zároltsága (**_fieldLocks**). Ezt akkor alkalmazzuk, ha a mező a Home mező, esetleg robot tartózkodik rajta, vagy már épült rá fal. A tábla alapértelmezés szerint (11 x 11)-es, de ez a konstruktorban paraméterezhető, leírás szerint (7 x 7) vagy (15 x 15) méretűre változtatható. A tábla lehetőséget ad állapotok lekérdezésére (**HasWall**, **IsRobot**, **IsHome**, **GetWall**, **IsLocked**), illetve beállítására (**BuildWall**, **SetLock**).
  - A hosszú távú adattárolás lehetőségeit az **IRoboRunDataAccess** 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 RoboRunFileDataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a **RoboRunDataException** kivétel jelzi.
  - A program az adatokat szöveges fájlként tudja eltárolni, melyek az **rrt** kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
  - A fájl első sora megadja az eltelt játékidőt, valamint a tábla méretét. A fájl többi része izomorf leképezése a játéktáblának, azaz a tábla méretének megfelelő számú sor következik ugyanennyi számmal soronként. Ezek a számok a 0 és 1 értéket vehetik fel, a mező zároltságának megfelelően. Ezután a robot adatai következnek egy sorban, majd még egy sorban a falak száma. Az utóbbi számnak megfelelő mennyiségű sorban pedig a falak adatai következnek.
- Modell:
  - A modell lényegi részét a **RoboRunModel** osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, úgymint az idő (**_gameTime**). A típus lehetőséget ad új játék kezdésére (**NewGame**), valamint lépésre (**Step**). Új játéknál megadható a játéktábla mérete is, különben alapértelmezetten (11 x 11)-es méretű lesz. Az idő előreléptetését időbeli lépések végzésével (**AdvanceTime**) tehetjük meg.
  - Az idő múlásáról a **GameTimeAdvanced**, az idő megállításáról a **GameTimePaused**, a robot mozgásáról a **RobotMoved**, a játék végéről pedig a **GameWin** események tájékoztatnak. Az események argumentuma (a **RobotMoved** eseményen kívül) (**RoboRunEventArgs**) tárolja az eltelt játék időt.
  - A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (**LoadAsync**) és mentésre (**SaveAsync**).
  - A játéktábla méretét a **GameTableSize** felsorolási típuson át kezeljük, és a **RoboRunModel** osztályban konstansok segítségével tároljuk az egyes méretek paramétereit.

- Nézet:
  - A nézetet a **GameForm** osztály biztosítja, amely tárolja a modell egy példányát (**_model**), valamint az adatelérés konkrét példányát (**_dataAccess**).
  - 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 (**GenerateGameTable**), illetve az értékek beállítását (**SetupGameTable**) külön metódusok végzik.
  - A játék időbeli kezelését egy időzítő végzi (**_timer**), ahogy robot mozgását is ez vezérli (**_robotTimer**). Ezeket mindig aktiváljuk a játék során, illetve inaktiváljuk, amennyiben bizonyos menüfunkciók futnak.
- A program teljes statikus szerkezete:



Tesztelés

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **TestModel** osztályban.
- Az alábbi tesztesetek kerültek megvalósításra:
  - **RoboRunModelNewGameSmallTest, RoboRunModelNewGameMediumTest, RoboRunModelNewGameBigTest:**
    - Új játék indítása, a mezők kitöltése, valamint a tábla méretének ellenőrzése a beállított méret alapján
  - **RoboRunModelStepTest:**
    - Játékbeli lépés hatásainak ellenőrzése, játék megkezdése előtt, valamint után.
  - **RoboRunModelAdvanceTimeTest:**
    - A játékbeli idő kezelésének ellenőrzése.
  - **RoboRunModelLoadTest:**
    - A játék modell betöltésének tesztelése mockolt perzisztencia réteggel.