**EVA**

**3.beadandó**

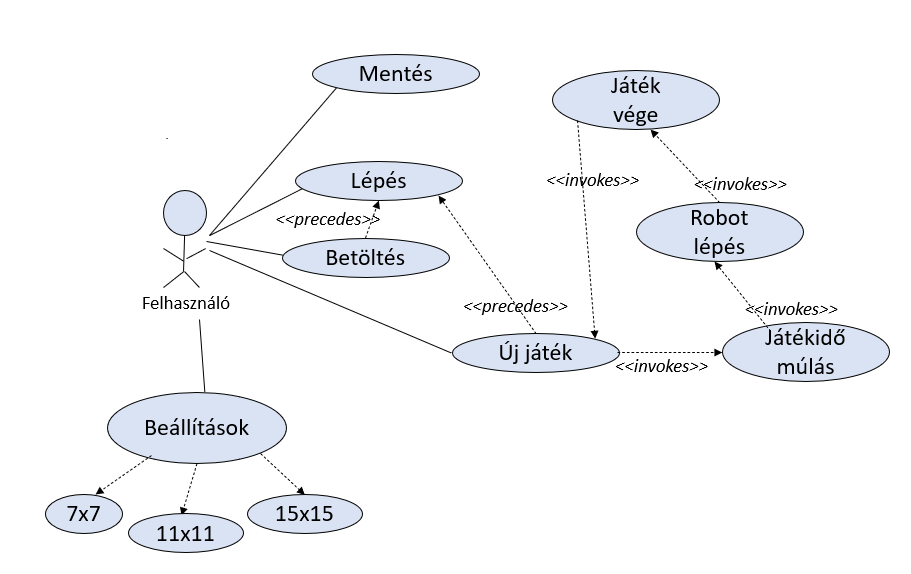
|  |  |
| --- | --- |
| **Batizi Eszter**  XQW70Y | *2020.11.24* |
| [*eszter379@gmail.com*](mailto:eszter379@gmail.com)  **7.csoport** |  |

**Feladat** *(3-as)*

Készítsünk programot, amellyel a következő játékot játszhatjuk. Adott egy 𝑛 × 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 egy személy játszhatja a gép ellen. Három táblaméretből tudunk választani: 7x7, 11x11, 15x15 (a játék elindulásakor a 11x11-es tábla az alapértelmezett).
* A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
* Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (New Game, Load Game, Pause/Resume), Settings (7x7,11x11,15x15). Az ablak alján megjelenítünk egy státuszsort, amely az eltelt időt jelzi.
* A játéktáblát egy n × n nyomógombokból álló rács reprezentálja. A nyomógomb egérkattintás hatására falat emel az adott mezőre. Amennyiben egy falnak már neki ütközött a robot, a fal eldől és oda kattintva már nem emelhető fal.
* A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (a robot belép a mágnes mezőjébe). Szintén dialógusablakokkal végezzük el a mentést, illetve betöltést, a fájlneveket a felhasználó adja meg.
* 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.
* Prezisztencia:

Az adatkezelés feladata a táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.

A GameTable osztály egy játéktáblát biztosít melynek értékei Field típusúak lehetnek (Robot, Empty, Magnet, Wall, DestroyedWall). A tábla alapértelmezés szerint 11 × 11 -es, de ez a konstruktorban paraméterezhető. A tábla lehetőséget ad egy adott mező típusának lekérdezésére (GetValue), illetve direkt beállítás (SetValue) elvégzésére.

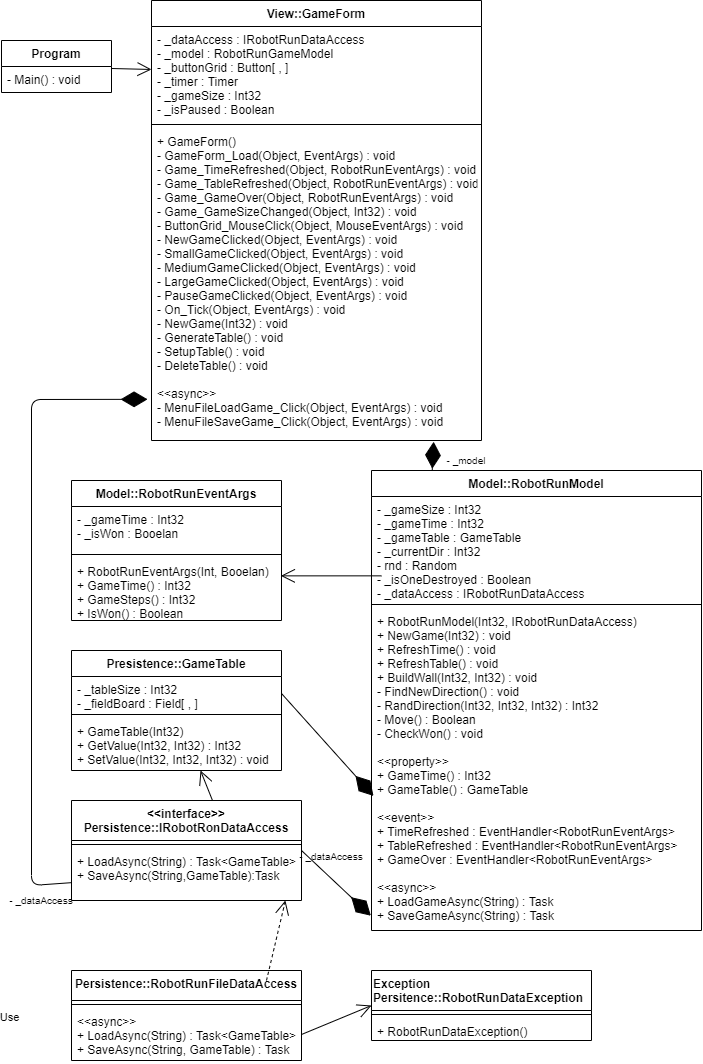
A hosszú távú adattárolás lehetőségeit az IRobotRunDataAccess interfész adja meg, amely lehetőséget ad a tábla betöltésére (LoadAsync), valamint mentésére (SaveAsync). A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg.

Az interfészt szöveges fájl alapú adatkezelésre a RobotRunFileDataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a RobotRunDataException 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 tábla méretét. A fájl többi része izomorf leképezése a játéktáblának, azaz összesen n sor következik, és minden sor n számot tartalmaz szóközökkel választva. A számok 0-4 közöttiek lehetnek (0-Robot, 1-Üres, 2-Mágnes, 3-Fal, 4-Ledöntött fal).

* Modell:
* A modell lényegi részét a RobotRunModel osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit. A típus lehetőséget ad új játék kezdésére (NewGame), valamint lépésre (BuildWall). Az idő múlásával meghívja a robot mozgatásáért felelős függvényt.
* A játékállapot változásáról a RefreshTable esemény, az idő múlásáról a RefreshTime míg a játék végéről a GameOver esemény tájékoztat. Az események argumentuma (RobotRunEventArgs) tárolja a győzelem állapotát valamint a játékidőt.
* A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (LoadGameAsync) és mentésre (SaveGameAsync)
* 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 (GenerateTable), illetve az értékek beállítását (SetupTable) külön metódusok végzik.
  + A játék időbeli kezelését egy időzítő végzi (\_timer), amelyet mindig aktiválunk játék során, illetve inaktiválunk, amennyiben bizonyos menüfunkciók futnak.



**Tesztelés:**

* A modell működését egységtesztekkel ellenőrizzük a RobotRunModelTest osztályban
* Tesztesetek:
  + NewGameSmallTest() – új játék indítása, a mágnes középre kerül, a játéktábla átméreteződik
  + NewGameMediumTest() -||-
  + NewGameLargeTest() -||-
  + BuildWallMethod() – új játékot indítunk, falat emelünk, ledönött falat emelünk -> próbálunk ráépíteni de nem engedheti a program
  + RobotLoadTest() – a játékmodell betöltésének tesztelése mockolt perzisztencia réteggel
  + Model\_GameOver(Object sender, RobotRunEventArgs e)