**3. Analízis modell kidolgozása**

15 – jgoldfisch

Konzulens:

Bodó Zsófia

Csapattagok

|  |  |  |
| --- | --- | --- |
| Mátyás Gergely | IL21NI | matyasg97@gmail.com |
| Horváth Ákos | DKILK6 | horvath.akos1997@gmail.com |
| Gurubi Barnabás | DXEXVR | gurubibarni@gmail.com |
| Tolnai Márk | ID61MK | tolesz11@windowslive.com |
| Bertalan Bálint | HNN9GA | blintber@gmail.com |

2018. február 26.

# Analízis modell kidolgozása

## Objektum katalógus

[Minden, a feladatban szereplő objektum rövid, egy-két bekezdés hosszú ismertetése. Meg kell jelenjen minden objektumhoz, hogy mi a felelőssége. Informális leírás, ezért nem szabad még foglalkozni az örökléssel, az interfészekkel, az absztrakt osztályokkal, a segédosztályokkal.]

### Box

A box osztály minden példánya egy-egy különböző dobozt tárol. Amennyiben egy dobozt a célmezőre tolnak ellehetetleníti önmaga mozgatását.

### EndField

A célmezőt képviseli, amire a **Box-**t kell tolni, hogy pontot kapjunk. Amennyiben egy **Box** rátolódik, az ott mozgathatatlanná válik, és a **Worker,** aki az eltolást végeztepontot kap érte.

### Game

A játék alapvető logikáját és mechanizmusait tárolja. Ezen felül referenciát tartalmaz a **Mapre** és tárolja a **Worker-ök** és a szabad dobozok (még mozgatható) számát.

### Hole

Egy lyukat jelenít meg. Amennyiben egy **Box**, vagy **Worker** áll rajta, megsemmisíti a rajta álló objektumot.

### Map

A pályát képviseli, mely több **Fieldből** áll. A menüben a pálya kiválasztását követően az osztály képes betölteni azt egy külső fájlból.

### Recorder

Nyilvántartást vezet a statikus mezőkről (**Field**), mint a falak (**WallField**), illetve a mozdíthatatlan ládák(**Box**). Szerepe van a beszorult ládák felfedezésében is.

### SimpleField

A pálya egyes mezőit képviselik a játékban. Lekérdezhető tőle a megfelelő irányban lévő szomszédos **Field**. Egy cellán egyidőben állhat egy **Box,** vagy egy **Player**, valamint tartalmazhat referenciát egy speciális elemre amely lehet **Hole** vagy **Switch**.

### Switch

Egy a pályán található kapcsolót valósít meg. Minden kapcsolóhoz tartozik egy referencia egy **Hole** osztályra. Amennyiben egy **Box** áll a **Switchen**, akkor aktiválódik, más esetben nem.

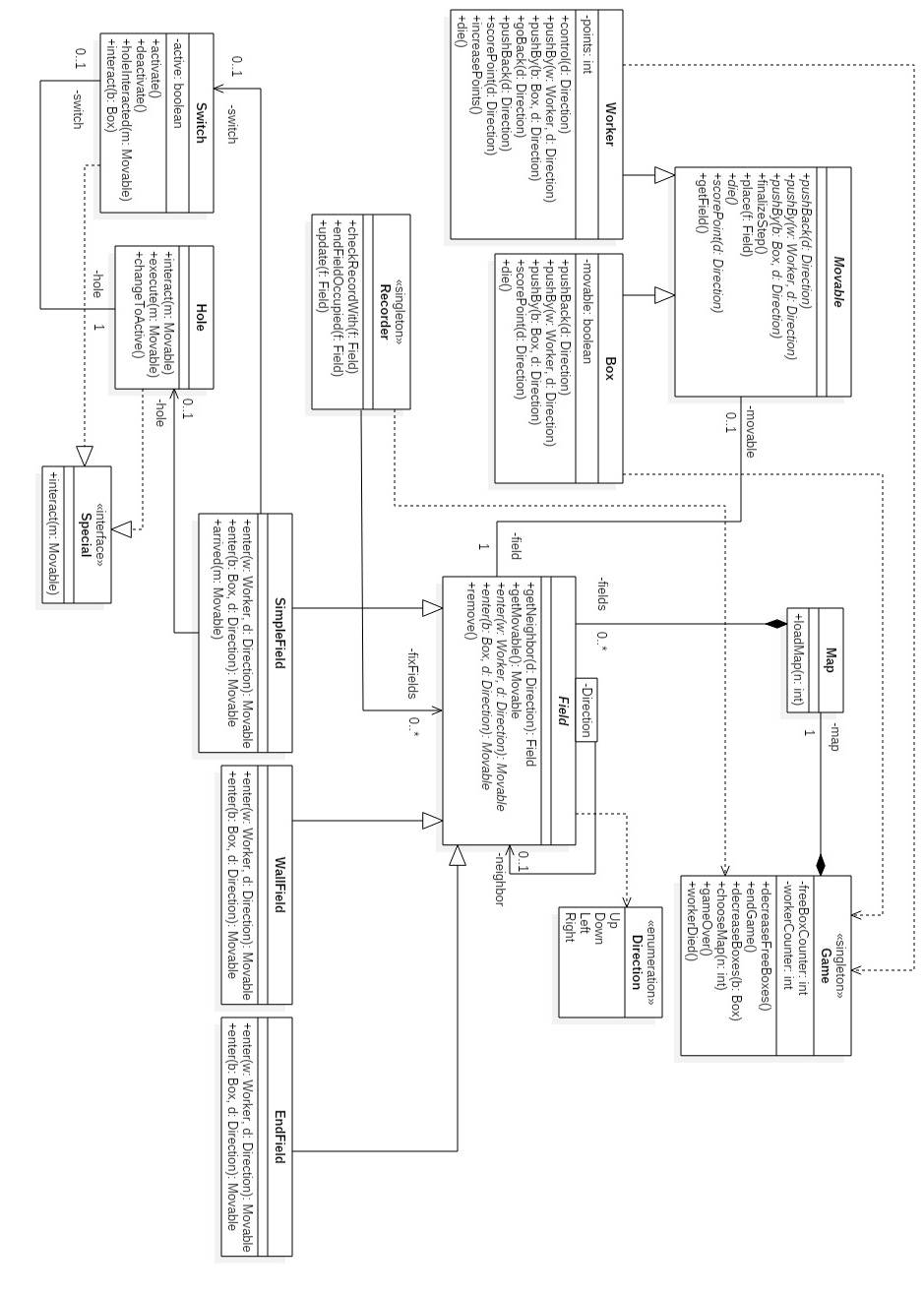
### WallField

Egy oszlopot, vagy falszeletet ábrázol a pályán. Megakadályozza, hogy dobozt toljanak rá, vagy hogy rálépjen egy játékos.

### Worker

Egy játékost képvisel, aki pálya területén belül szabadon mozoghat. Segítségével lehetséges a ládák tologatása és tárolja, hogy hány ládát tolt már a helyére.

## Statikus struktúra diagramok



## Osztályok leírása

### Box

#### Felelősség

Ennek az osztálynak az egyes példányai fogják reprezentálni a dobozokat a játékban. Őket kell majd a célmezőre eljuttatni.

#### Ősosztályok

Movable

#### Attribútumok

* **boolean movable:** ezzel jelezzük, hogy mozgatható-e a láda.
* **Field field:** az a mező, ahol éppen tartózkodik a láda.

#### Metódusok

* **void die():** Ez a függvény felel az entitás elpusztításáért, illetve csökkenti a szabad dobozok számát.
* **void finalizeStep():** Ez az a függvény, ami az egyes lépések véglegesítésért felel. Meghívja annak a **SimpleField**-nek az *arrive(m: Movable)* függvényét, ahová a Movable leszármazott majd megérkezne, ami pedig, ha tárolja a **Special** osztály egy példányát (jellemzően kapcsolót (**Switch**), vagy lyukat (**Hole**), akkor meghívja annak az *interact(Movable m)* függvényét, ami pedig megteszi a kellő intézkedést (bekapcsolja a kapcsolót vagy megsemmisíti a **Movable** leszármazottat).
* **void place(Field f):** Ez a függvény átállítja a **Movable** mező referenciáját a kapott mezőre.
* **void pushBack(Direction d):** Ennek a függvénynek a segítségével a tolás irányába visszaléptethetjük a dobozt. (Erre szükség van pl. amikor a falba bele akarnánk tolni a dobozt.)
* **void pushBy(Box b, Direction d):** Függvény arra az esetre, amennyiben a ládánkat (**Box**) egy másik doboz tolja meg, intézi, hogy a láda a megfelelő helyre kerüljön.
* **void pushBy(Worker w, Direction d):** A láda (**Box**) egy játékos által való megtolásákor hívjuk meg, intézi hogy a láda a megfelelő helyre kerüljön.
* **void scorePoint(Direction d):** Amennyiben a célmezőre tolódott a láda, ennek a függvénynek a segítségével végezhetjük el a megfelelő játékos pontjainak növelését.

### EndField

#### Felelősség

Célmező, amire a dobozokat (**Box**)juttatni kell. Amennyiben rálép egy doboz, akkor az mozdíthatatlanná válik ott.

#### Ősosztályok

Field

#### Attribútumok

* **Field neighbor[Direction]:** tárolja a szomszédos mezőket a lehetséges irányokban.
* **Movable movable:** referencia egy **Movable** leszármazottra.

#### Metódusok

* **Field getNeighbor(Direction d):** Visszaadja az átadott irányban lévő szomszédos mezőt.
* **Movable enter(Box b, Direction d):** Fogadja a mezőre érkező **Box**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Végezetül pedig mozdíthatatlanná teszi a mezőre érkezett **Box-**t. Továbbá elindítja a pontszerzési folyamatot a megfelelő munkás számára.
* **Movable enter(Worker w, Direction d):** Fogadja a mezőre érkező **Worker**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Végezetül pedig megnézi, hogy mi maradt azon a mezőn, ahonnan a **Worker** jött, és visszaadja annak a referenciáját (üres mező esetén null).
* **Movable getMovable():** Visszaadja a mezőn lévő **Movable** leszármazott referenciáját.
* **void remove():** A mezőhöz tartozó (tárolt) **Movable** referenciát eltávolítja.

### Field

#### Felelősség

Absztrakt ősosztály, amiből a mező objektumok származnak.

#### Attribútumok

* **Field neighbor[Direction]:** tárolja a szomszédos mezőket a lehetséges irányokban.
* **Movable movable:** referencia egy **Movable** leszármazottra.

#### Metódusok

* **Field getNeighbor(Direction d):** Visszaadja az átadott irányban lévő szomszédos mezőt.
* **Movable enter(Box b, Direction d):** Absztrakt függvény, ami a leszármazottakban kerül implementálásra.
* **Movable enter(Worker w, Direction d):** Absztrakt függvény, ami a leszármazottakban kerül implementálásra.
* **Movable getMovable():** Visszaadja a mezőn lévő **Movable** leszármazott referenciáját.
* **void remove():** A mezőhöz tartozó (tárolt) **Movable** referenciát eltávolítja.

### Game

#### Felelősség

A játék alapvető mechanizmusaiért felel és tárolja a nyeréshez, vagy vesztéshez szükséges adatokat.

#### Attribútumok

* **int freeBoxCounter:** a még mozgatható dobozok számát tárolja.
* **int workerCounter:** a még játékban lévő munkások számát tárolja.
* **Map map:** referencia a pályát tartalmazó osztályra.

#### Metódusok

* **void chooseMap(int i):** Meghívja az i-edik pálya betöltését.
* **void decreaseBoxes(Box b):** Abban az esetben hívódik meg, amennyiben egy doboz megsemmisül. Csökkenti a még mozgathatódobozok számát egyel.
* **void decreaseFreeBoxes():** Csökkenti a még mozgatható dobozok számát 1-el, amennyiben egy doboz elakad.
* **void endGame():** Akkor hívódik meg, ha a mozgatható dobozok száma nullára csökken. Ekkor a legtöbb pontot elért játékos nyer.
* **void gameOver():** Akkor hívódik meg, ha minden játékos meghal. Ebben az esetben senki sem kerül ki nyertesként a játékból.
* **void workerDied():** Amennyiben meghal a **Worker,** csökkenti a játékban lévő munkások számát.

### Hole

#### Felelősség

Megsemmisíti a rajta álló **Movable** osztályt.

#### Interfészek

Special

#### Attribútumok

* **Switch switch:** referencia a hozzá tartozó kapcsolóhoz.

#### Metódusok

* **void changeToActive():** A kapcsoló megváltoztatása esetén hívjuk meg, hogyha áll valami abban a pillanatban a lyukon, akkor azt semmisítse meg.
* **void execute(Movable m):** Függvény arra, ha meg akarjuk semmisíteni a mezőn álló osztályt. Meghívja a kellő metódusokat a objektum megsemmisítéséhez.
* **void interact(Movable m):** Egy a mezőn lévő lyukkal való interakció (rálépés) esetén hívjuk meg. Ellenőrzi, hogy tartozik-e hozzá kapcsoló, továbbá meghívja a reagáláshoz tartozó további metódusokat.

### Map

#### Felelősség

Szerepe a pálya betöltése.

#### Attribútumok

* **ArrayList<Field>fields:** referenciák az egyes mezőkre.

#### Metódusok

* **void loadMap(int n):** A pálya betöltését intézi.

### Movable

#### Felelősség

Egy absztrakt ősosztály, amiből a mozgatható objektumok származnak.

#### Attribútumok

* **Field field:** a mező, amin tartózkodik.

#### Metódusok

* **void die():** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* **void finalizeStep():** Ez az a függvény, ami az egyes lépések véglegesítésért felel. Meghívja annak a **SimpleField**-nek az *arrive(m: Movable)* függvényét, ahová a Movable leszármazott majd megérkezne, ami pedig, ha tárolja a **Special** osztály egy példányát (jellemzően kapcsolót (**Switch**), vagy lyukat (**Hole**), akkor meghívja annak az *interact(Movable m)* függvényét, ami pedig megteszi a kellő intézkedést (bekapcsolja a kapcsolót vagy megsemmisíti a **Movable** leszármazottat).
* **void place(Field f):** Ez a függvény átállítja a **Movable** mező referenciáját a kapott mezőre.
* **void pushBack(Direction d):** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* **void pushBy(Box b, Direction d):** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* **void pushBy(Worker w, Direction d):** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* **void scorePoint(Direction d):** Absztrakt függvény, leszármazottakban kerül megvalósításra.

### Recorder

#### Felelősség

Nyilvántartás vezetése a mozdíthatatlan mezőkről. Segítségével lehetséges a beragadt (pl. sarokban lévő) dobozok detektálása. Ide kerülnek a mozdíthatatlan dobozok is.

#### Attribútumok

* **ArrayList<Field>fields:** referenciák azokra a mezőkre, amikre nem lehet rálépni.

#### Metódusok

* **void checkRecordWith(Field f):** Ezzel a függvénnyel lehetséges annak az ellenőrzése, hogy a paraméterben kapott mező benne-e van már a nyilvántartásban
* **void endFieldOccupied(Field f):** Ennek a függvénynek a segítségével jelezhető, ha egy célmezőre (**EndField**) beérkezett egy láda (**Box**). A paraméter átadásával jelezzük, hogy melyik mezőről van szó, majd ez a mező bekerül a nyilvántartásba.
* **void update(Field f):** Ezt a függvényt mindig egy doboz (**Box**) mozgatása után hívjuk meg. A paraméterben átadott mező az a mező, ahová a doboz újonnan került. A nyilvántartás alapján pedig el tudja dönteni, hogy a dobozunk beszorult-e. Amennyiben igen, akkor azt is felveszi a nyilvántartásba.

### SimpleField

#### Felelősség

Közönséges mező, amire szabadon rá lehet lépni. Ezeken közlekednek a munkások és a ládák.

#### Ősosztályok

Field

#### Attribútumok

* **Field neighbor[Direction]:** tárolja a szomszédos mezőket a lehetséges irányokban.
* **Hole hole:** tárolja a mezőn lévő lyukat (ha van).
* **Movable movable:** referencia egy **Movable** leszármazottra.
* **Switch switch:** tárolja a mezőn lévő kapcsolót (ha van).

#### Metódusok

* **Field getNeighbor(Direction d):** Visszaadja az átadott irányban lévő szomszédos mezőt.
* **Movable enter(Box b, Direction d):** Fogadja a mezőre érkező **Box**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy mi maradt azon a mezőn, ahonnan a **Box** jött, és visszaadja annak a referenciáját (üres mező esetén null).
* **Movable enter(Worker w, Direction d):** Fogadja a mezőre érkező **Worker**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy mi maradt azon a mezőn, ahonnan a **Worker** jött, és visszaadja annak a referenciáját (üres mező esetén null).
* **Movable getMovable():** Visszaadja a mezőn lévő Movable leszármazott referenciáját.
* **void arrived(Movable m):** A *finalizeStep(),* azaz a lépés véglegesítésnél van szerepe ennek a függvénynek. Átadja az adott mezőre lépő **Movable** referenciát és ekkor kerül majd végrehajtásra (amennyiben van) a mezőn lévő **Special** interfésszel rendelkező entitás *interact(Movable m)* függvénye.
* **void remove():** A mezőhöz tartozó (tárolt) **Movable** referenciát eltávolítja.

### Special

#### Felelősség

Interfész

#### Metódusok

* **void interact(Movable m)**

### Switch

#### Felelősség

Egy kapcsolót és hozzá tartozó állapotot tárolja.

#### Interfészek

Special

#### Attribútumok

* **Hole hole:** referencia a kapcsolóhoz tartozó lyukhoz.
* **boolean active:** változó, hogy a kapcsoló jelen pillanatban aktiválva van-e.

#### Metódusok

* **void activate():** Aktiválja a kapcsolót.
* **void deactivate():** Deaktiválja a kapcsolót.
* **void holeInteracted(Movable m):** Akkor hívódik meg, ha a kapcsolóhoz tartozó mezőre valaki rálép. Eldönti, hogy a kapcsoló éppen aktív-e én ennek függvényében meghívja a szükséges metódusokat.
* **void interact(Box b):** Lekezeli, ha doboz érkezik a mezőre. Aktiválja a kapcsolót és szól a lyuknak, hogy megváltozott az állapota.

### WallField

#### Felelősség

Olyan mező objektum, mely egy falat reprezentál.

#### Attribútumok

* **Field neighbor[Direction]:** tárolja a szomszédos mezőket a lehetséges irányokban.
* **Movable movable:** referencia egy Movable leszármazottra.

#### Metódusok

* **Field getNeighbor(Direction d):** Visszaadja az átadott irányban lévő szomszédos mezőt.
* **Movable enter(Box b, Direction d):** Megakadályozza, hogy egy **Box** rálépjen a mezőre.
* **Movable enter(Worker w, Direction d):** Megakadályozza, hogy egy **Worker** rálépjen a mezőre.
* **Movable getMovable():** Visszaadja a mezőn lévő Movable leszármazott referenciáját.
* **void remove():** A mezőhöz tartozó (tárolt) **Movable** referenciát eltávolítja.

### Worker

#### Felelősség

A játékosok által irányított munkást reprezentáló osztály. Ő tudja a ládákat (**Box**) eltolni, illetve a pontokat is ő kapja majd.

#### Ősosztályok

Movable

#### Attribútumok

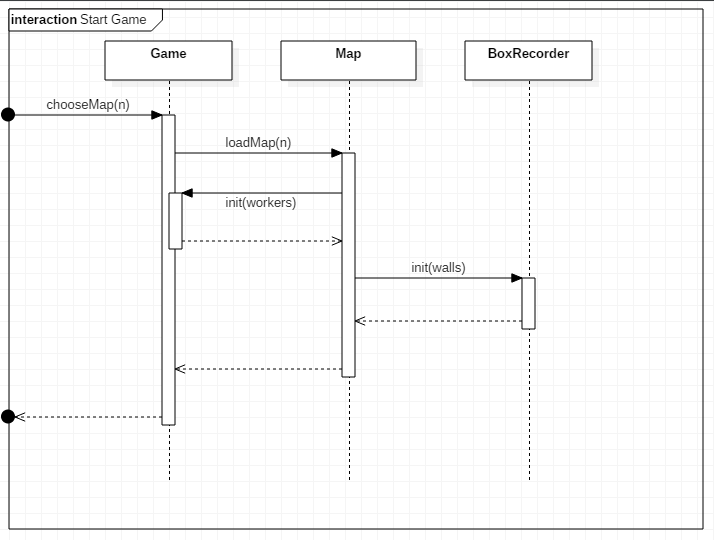
* **Field field:** referencia a mezőre, amin a munkás épp tartózkodik.
* **int points:** a játékos pontjai, amiket az egyes ládák célmezőre juttatásáért kap.

#### Metódusok

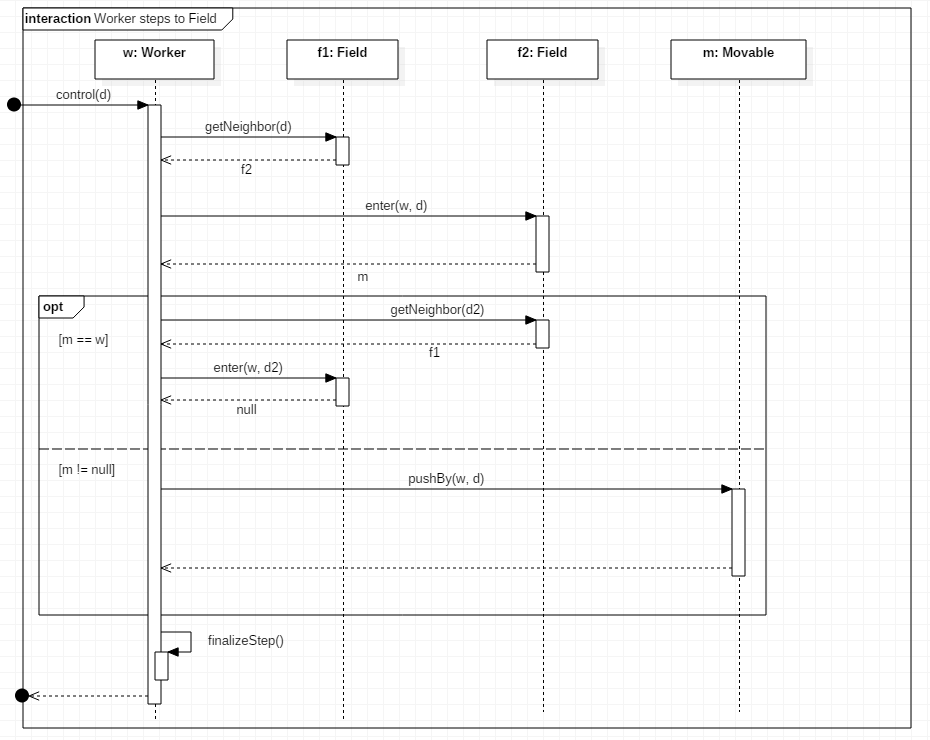
* **void control(Direction d):** A munkás (**Worker**) irányításáért felelős függvény. A paraméterben kapott irányban fogja megpróbálni elmozdítani a munkást a szomszédos mezők figyelembe vételével (Mindez önakaratból történik, tehát a felhasználó beavatkozására).
* **void die():** Ez a függvény felel az entitás elpusztításáért, illetve csökkenti a munkások számát.
* **void finalizeStep():** Ez az a függvény, ami az egyes lépések véglegesítésért felel. Meghívja annak a **SimpleField**-nek az *arrive(m: Movable)* függvényét, ahová a **Movable** leszármazott majd megérkezne, ami pedig, ha tárolja a **Special** osztály egy példányát (jellemzően kapcsolót (**Switch**), vagy lyukat (**Hole**), akkor meghívja annak az *interact(Movable m)* függvényét, ami pedig megteszi a kellő intézkedést (bekapcsolja a kapcsolót vagy megsemmisíti a **Movable** leszármazottat).
* **void goBack(Direction d):** Szintén a munkás -> munkás ütközés / eltolás megfelelő lekezelésében van szerepe, mégpedig elindít egy visszafele való lökést, hiszen nem szabad hagyni, hogy az egyik munkás a másikat közvetlenül eltolja, tehát úgymond visszarendezi a dolgokat.
* **void increasePoints():** A munkás pontjainak növelése.
* **void place(Field f):** Ez a függvény átállítja a **Movable** mező referenciáját a kapott mezőre.
* **void pushBack(Direction d):** Ennek a függvénynek ott van szerepe, amikor egy munkás a dobozt a falba bele akarná tolni. Ekkor ezt nem engedi majd, hanem az egész sor (pl.: munkás->doboz->fal) egyet visszafele fog lépni.  
  Ennek a megvalósításáért részben felelős ez a metódus is.
* **void pushBy(Box b, Direction d):** Ez a függvény arra az esetre szolgál, hogyha a munkást egy dobozzal (**Box**) próbálnának meg eltolni.
* **void pushBy(Worker w, Direction d):** Ez a függvény arra az esetre szolgál, ha a munkást egy másik munkás próbálná meg eltolni.
* **void scorePoint(Direction d):** A pontok növeléséért felelős, méghozzá az *increasePoints()* függvény segítségével.

## Szekvencia diagramok

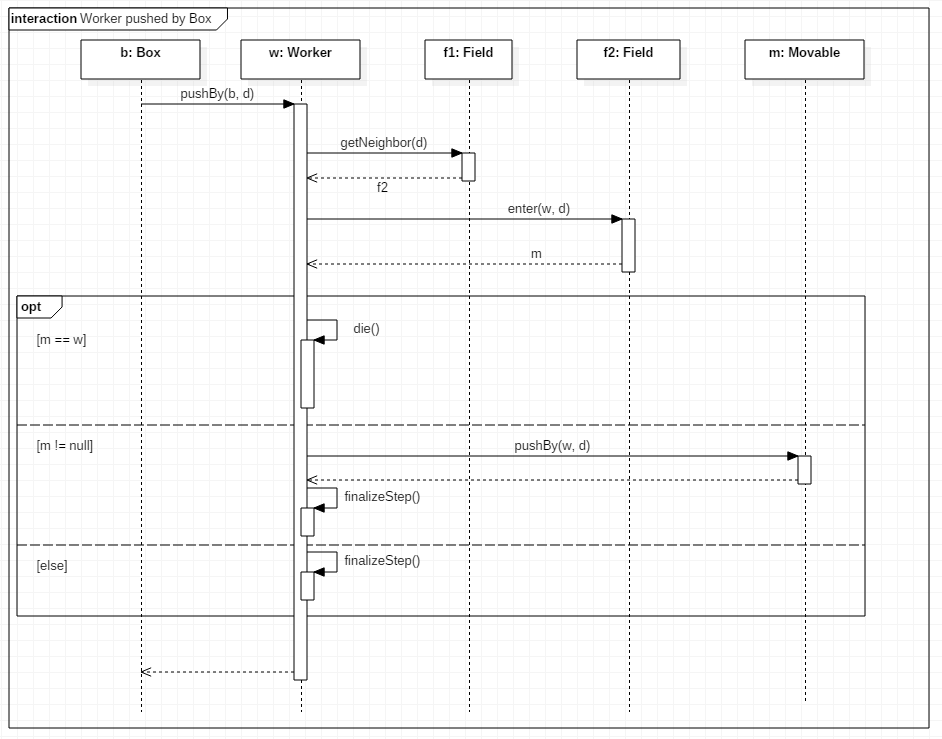
### Start Game



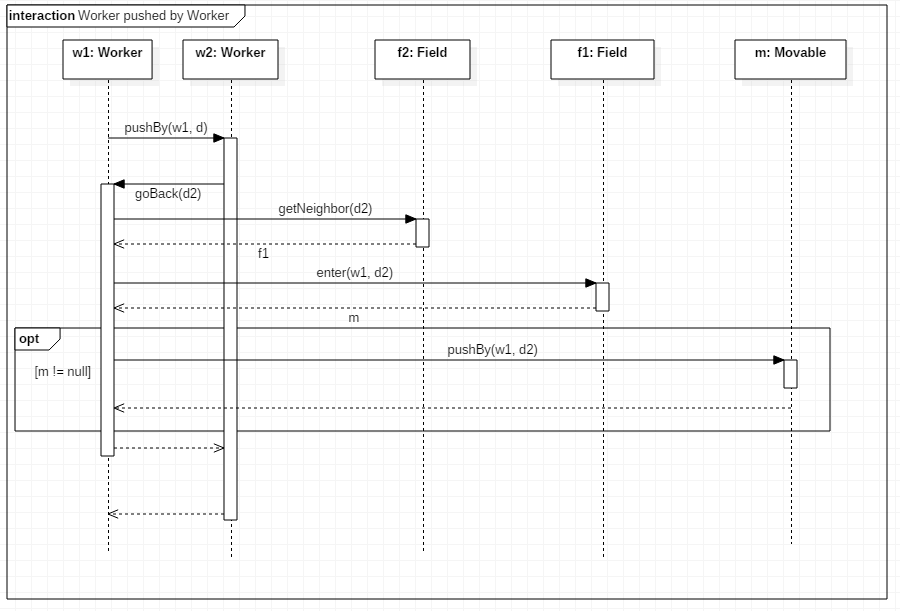
### Worker steps to Field



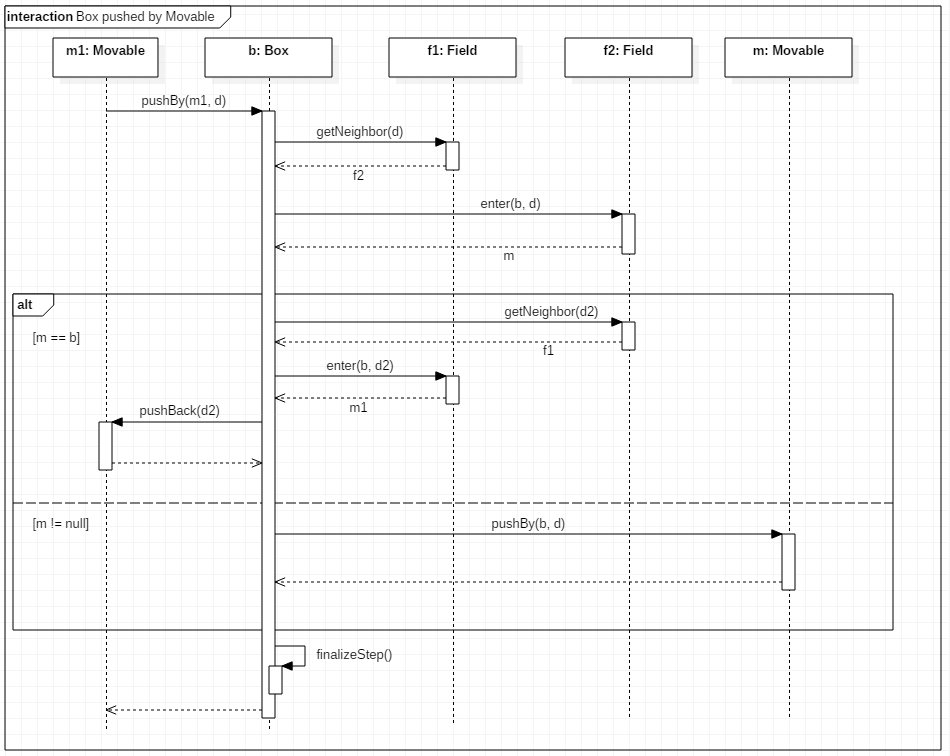
### Worker pushed by Box



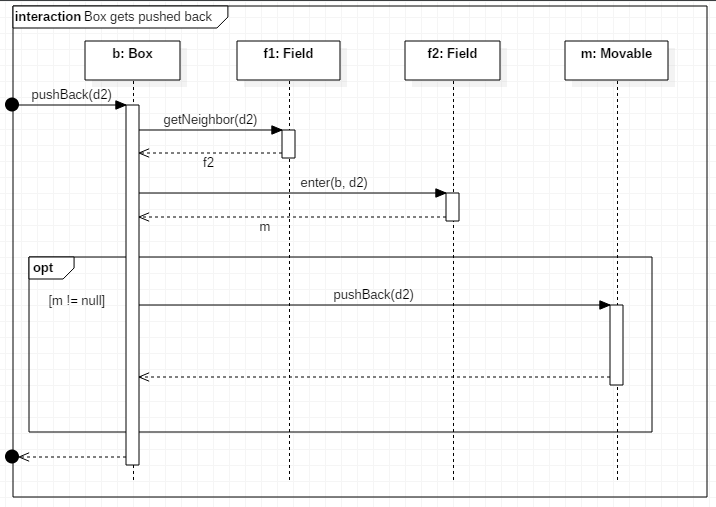
### Worker pushed by Worker



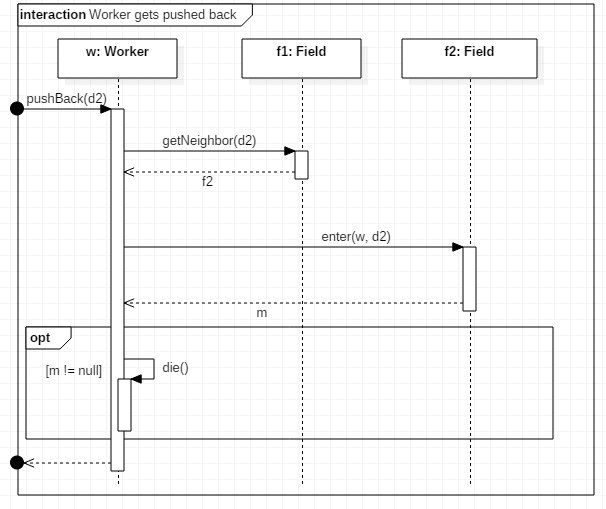
### Box pushed by Movable



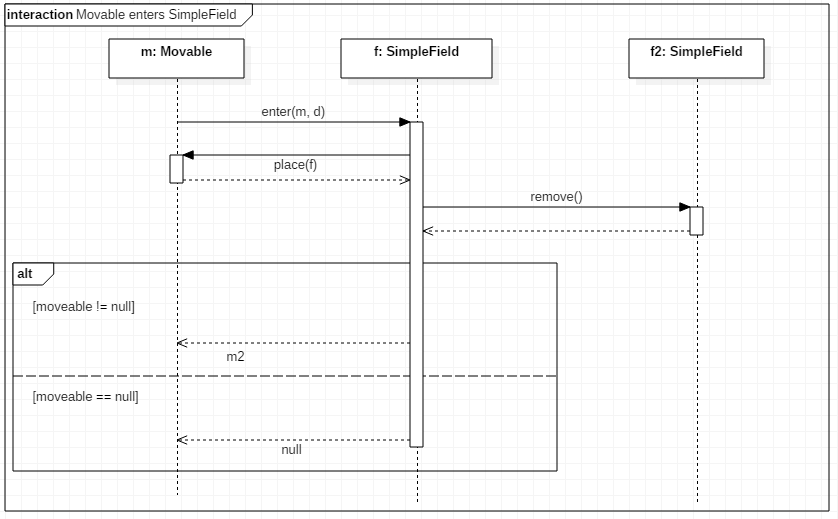
### Box gets pushed back



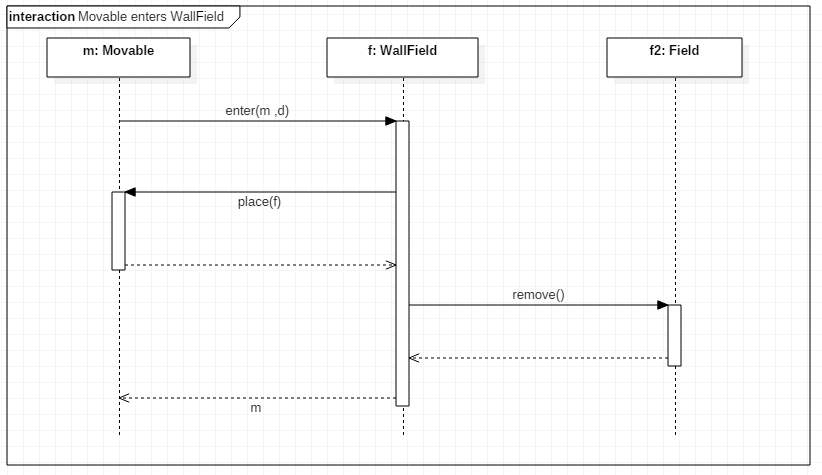
### Worker gets pushed back



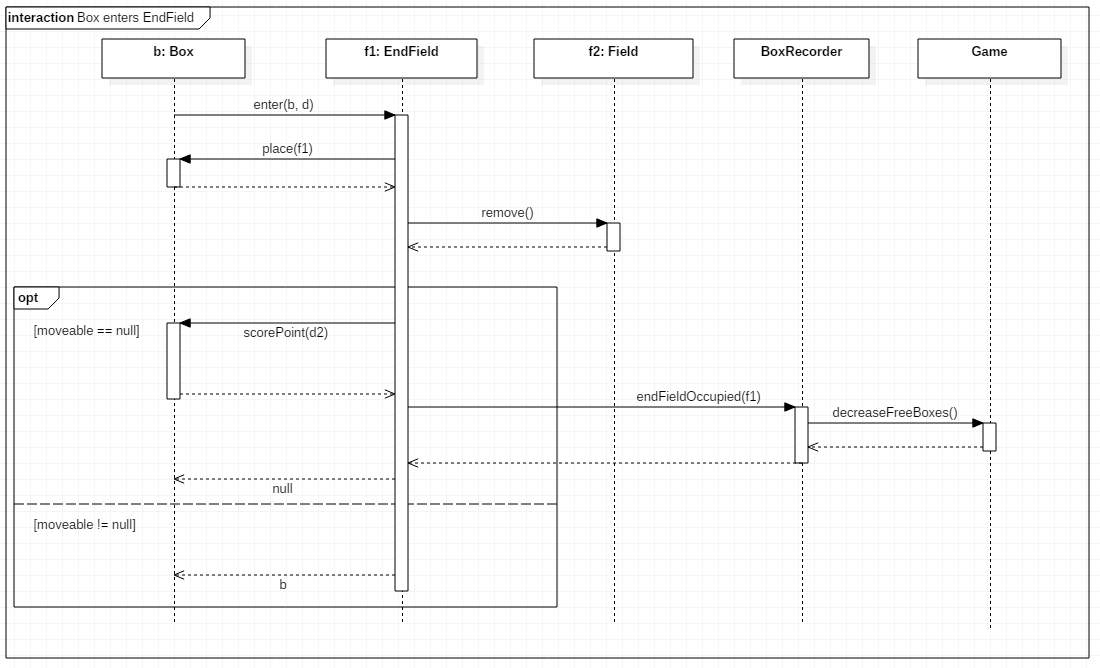
### Movable enters SimpleField



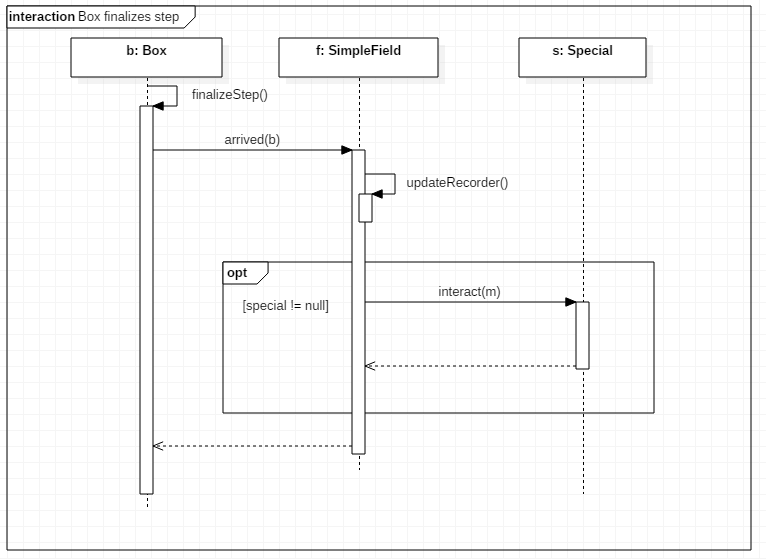
### Movable enters WallField



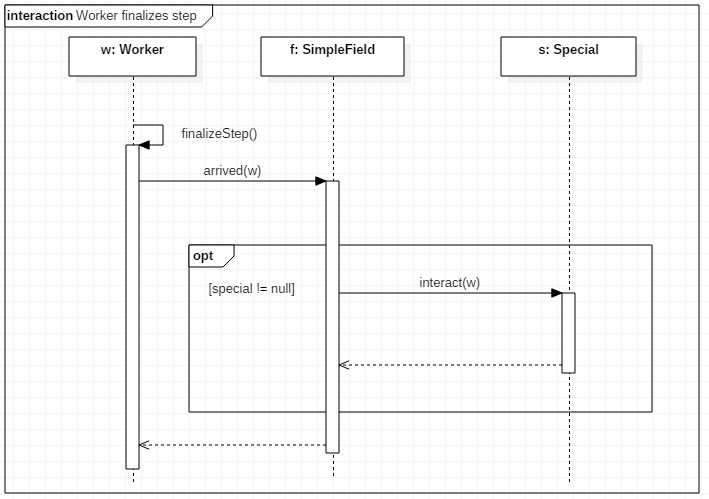
### Box enters EndField



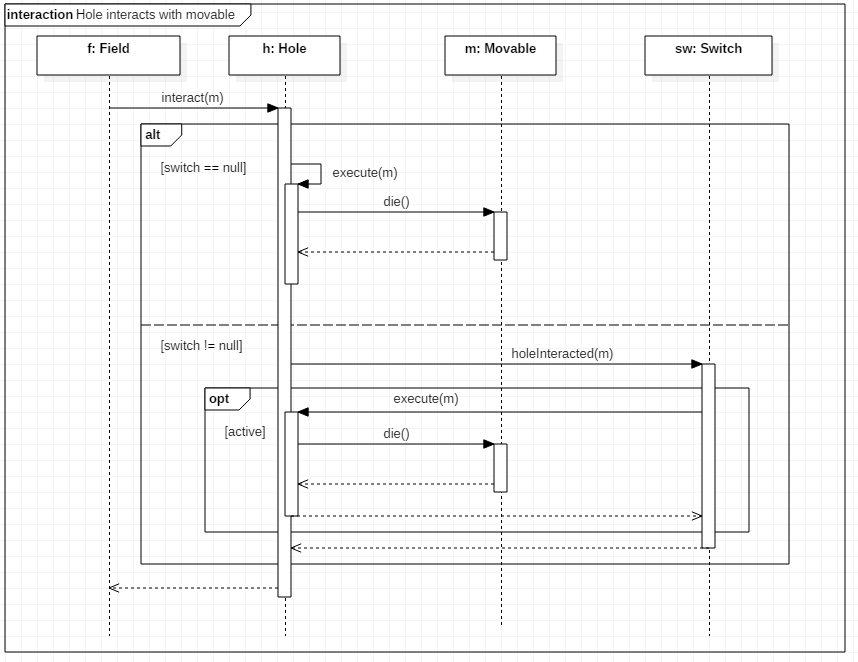
### Box finalizes step



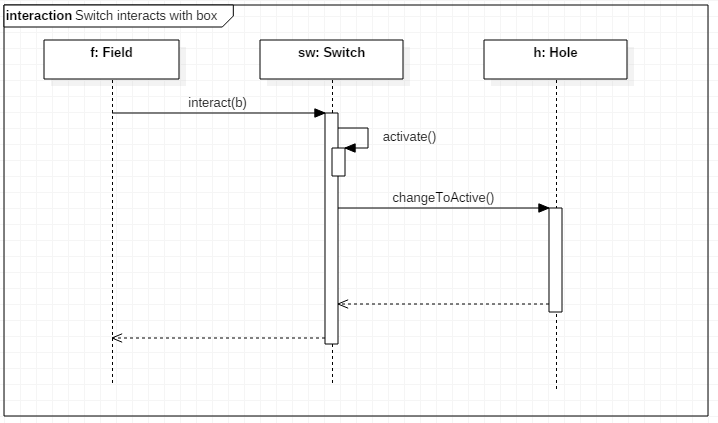
### Worker finalizes step



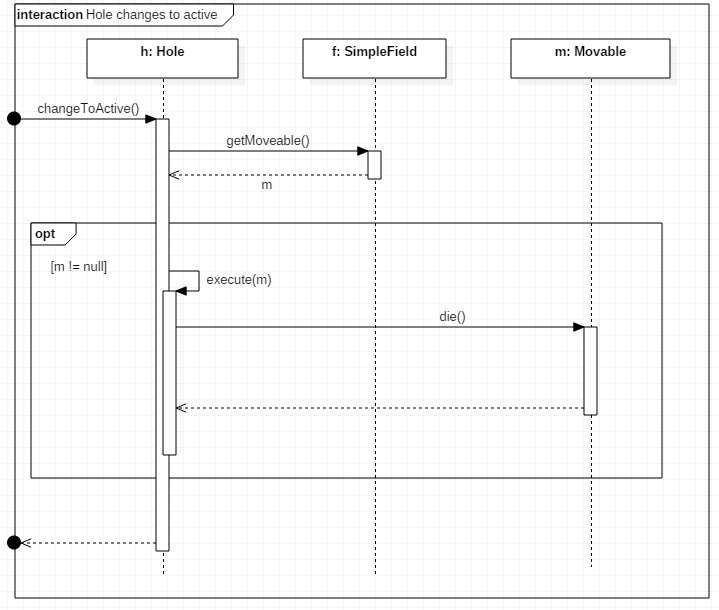
### Hole interacts with movable



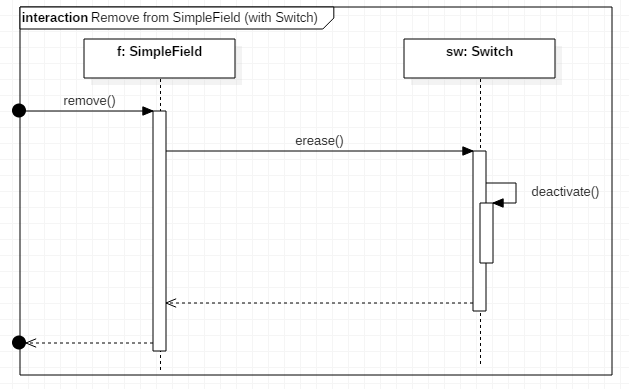
### Switch interacts with box



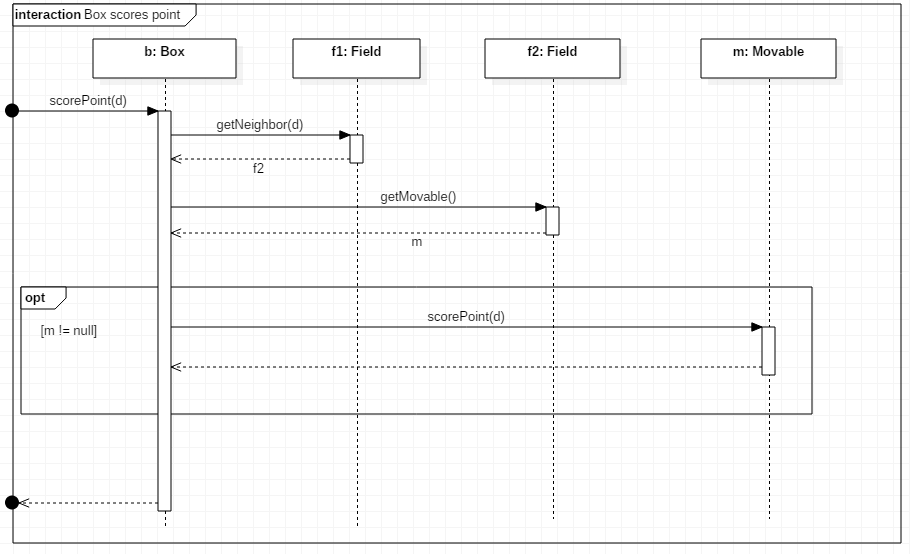
### Hole chnages to active



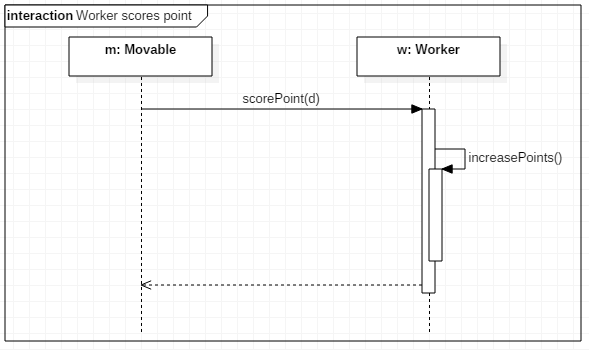
### Remove from SimpleField (with Switch)



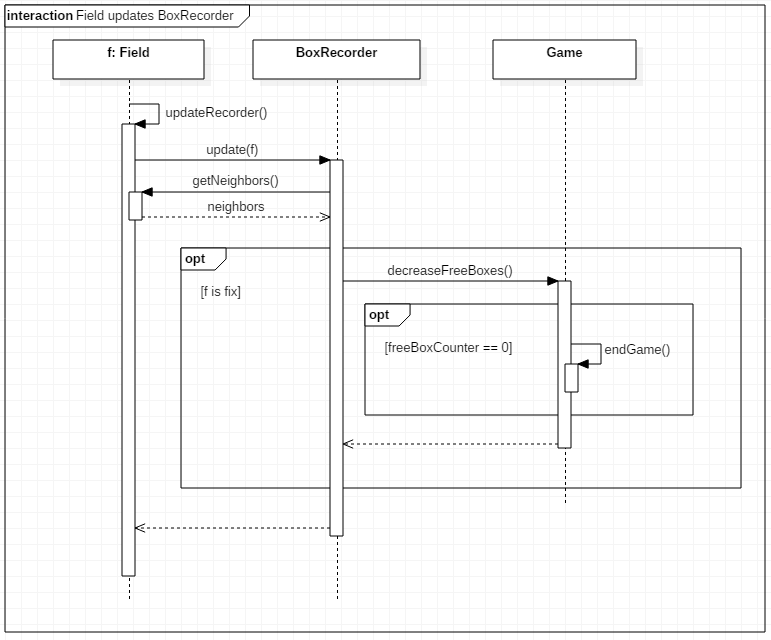
### Box „scores point”



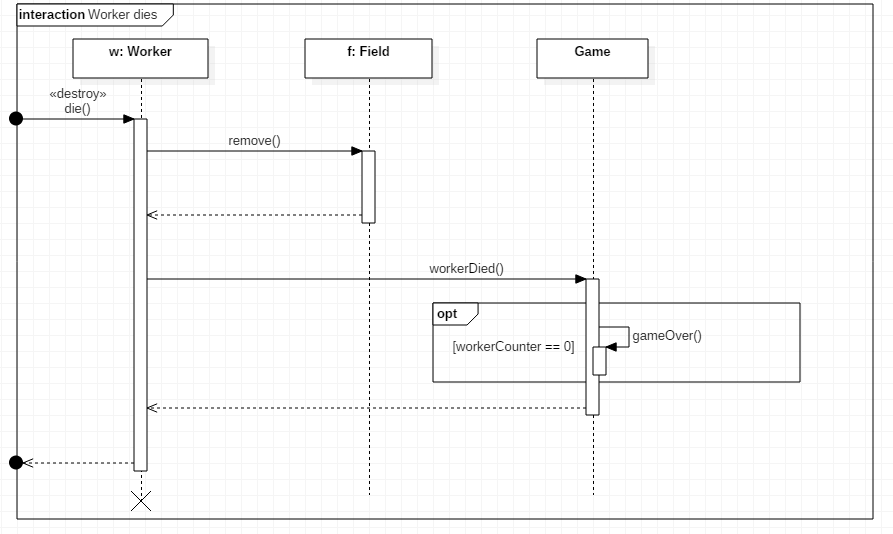
### Worker scores point



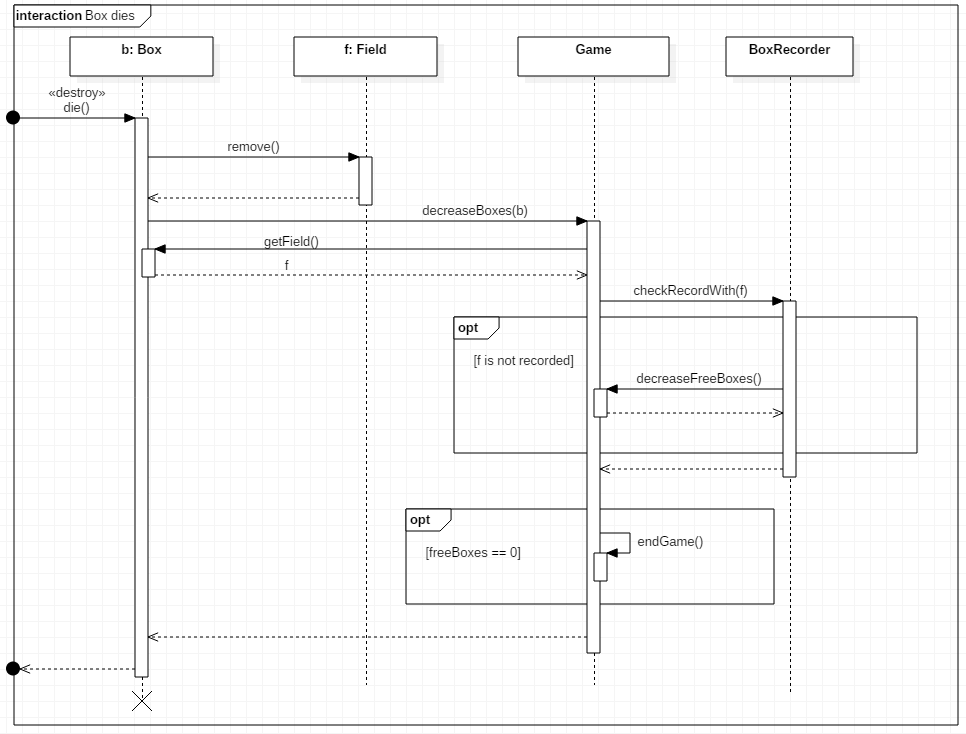
### Field updates BoxRecorder



### Worker dies



### Box dies



[Inicializálásra, use-case-ekre, belső működésre. Konzisztens kell legyen az előző alfejezettel. Minden metódus, ami ott szerepel, fel kell tűnjön valamelyik szekvenciában. Minden metódusnak, ami szekvenciában szerepel, szereplnie kell a valamelyik osztálydiagramon. Fontos, hogy az aktor use-case-eitől bármelyik diagramhoz el lehessen jutni a metódushívások követésével.]

## State-chartok

[Csak azokhoz az osztályokhoz, ahol van értelme. Egyetlen állapotból álló state-chartok ne szerepeljenek. A játék működését bemutató state-chart-ot készíteni tilos.]

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2018.02.20. 18:00 | 2 óra | Horváth  Tolnai  Bertalan  Mátyás  Gurubi | **Értekezlet**.  **Döntés**: Alapvető működés első átgondolása. Mátyás elkészíti 02.21-re, a konzultációra a próba szekvencia diagramot. |
| 2018.02.21. 09:00 | 2 óra | Mátyás | **Tevékenység**: Mátyás elkészíti a próba szekvencia diagramot. |
| 2018.02.21 19:00 | 2 óra | Horváth  Tolnai  Bertalan  Mátyás  Gurubi | **Értekezlet**.  **Döntés**: Közös csapatmegbeszélés, részek kiosztása. Határidő: 02.24. 18:00. Tolnai, Bertalan: 3.1 – 3.3. Horváth, Mátyás, Gurubi: 3.4. Mátyás: 3.6 |
| 2018.02.22. 19:00 | 2 óra | Bertalan | **Tevékenység**: 3.1 Kidolgozása 3.2 elkezdése. |
| 2018.02.22. 20:00 | 1 óra | Tolnai | **Tevékenység**: 3.1 ellenőrzése és 3.2 kidolgozásának átgondolása. |
| 2018.02.23. 13:00 | 2 óra | Horváth | **Tevékenység**: Szekvencia diagramok alap tervezése |
| 2018:02:23. 15:00 | 2 óra | Gurubi | **Tevékenység**: Szekvencia diagramok rajzolása: 3.4.1 – 3.4.5 |
| 2018.02.23. 16:00 | 2 óra | Mátyás | **Tevékenység**: Szekvencia diagramok rajzolása: 3.4.8-3.4.12 |

|  |  |  |  |
| --- | --- | --- | --- |
| 2018. 02. 23. 18:00 | 2 óra | Tolnai | **Tevékenység**: Tolnai kidolgozza a 3.3.1 – 3.3.4 |
| 2018.02.23. 21:00 | 1 óra | Bertalan | **Tevékenység**: Bertalan kidolgozza a 3.3.5 – 3.3.7 |
| 2018. 02. 24. 13:00 | 2 óra | Bertalan | **Tevékenység**: Bertalan kidolgozza a 3.3.8 – 3.3.11 |
| 2018. 02. 24. 14:00 | 2 óra | Tolnai | **Tevékenység**: Tolnai kidolgozza a 3.3.12 – 13. Továbbá javít az eddig elkészült Osztály leírásokban. (3.3) |
| 2018. 02. 24. 16:00 | 2 óra | Hotváth | **Tevékenység**: Szekvencia diagramok rajzolása: 3.4.6 – 3.4.7 és javítás az eddigi elkészült szekvencia diagramokon. |
| 2018.02.24. 18:00 | 2 óra | Gurubi | **Tevékenység**: Szekvencia diagramok rajzolása: 3.4.13 – 3.4.16 |
| 2018.02.24. 19:00 | 1 óra | Mátyás | **Tevékenység**: Szekvencia diagramok rajzolása: 3.4.17 – 3.4.18 |
| 2018.02.25. 09:00 | 2 óra | Tolnai | **Tevékenység**: Osztály diagram (3.2) kiegészítése, befejezése (első) |
| 2018.02.25. 10:00 | 2 óra | Bertalan | **Tevékenység**: Osztály leírások (3.3) javítása, összehangolása az elkészült diagramokkal. |
| 2018.02.25. 13:00 | 2 óra | Horváth | **Tevékenység**: Szekvencia diagramok rajzolása: 3.4.19 – 3.4.21 |
| 2018.02.25 15:00 | 2 óra | Gurubi | **Tevékenység**: Javítás a szekvencia diagramokon. Főleg a végén lévőkőn. |
| 2018.02.25. 16:00 | 2 óra | Tolnai | **Tevékenység**: Osztály diagram (3.2) kiegészítése, javítása |
| 2018.02.25. 17:00 | 2 óra | Bertalan | **Tevékenység**: Osztály leírások (3.3) javítása, összehangolása a diagramokkal. |
| 2018.02.25 |  |  |  |