**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 25.

# 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.]

### SimpleField

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

### Box

A box osztály minden példánya egy-egy különböző dobozt reprezentál. Worker-ek által eltolható, de a **Wall**-on nem tolható át. Amennyiben egy dobozt a célmezőre tolnak ellehetetleníti önmaga mozgatását.

### Game

A játék alapvető logikáját és mechanizmusait tárolja. Ezen felül referenciát tartalmaz a **Mapre** és tárolja a **Worker-öket**. Számolja a pályán lévő **Worker**-öket, továbbá a mozdítható dobozokat. Az ő feladata a **Map** betöltése is.

### Hole

Egy lyukat jelenít meg. Amennyiben egy **Box**, vagy **Worker** áll rajta, megsemmisíti a rajta álló objektumot. Tartozhat hozzá kapcsoló. Amikor a hozzá tartozó kapcsoló nem aktív akkor nem semmisíti meg a rajta álló említett objektumokat.

### Map

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

### 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.

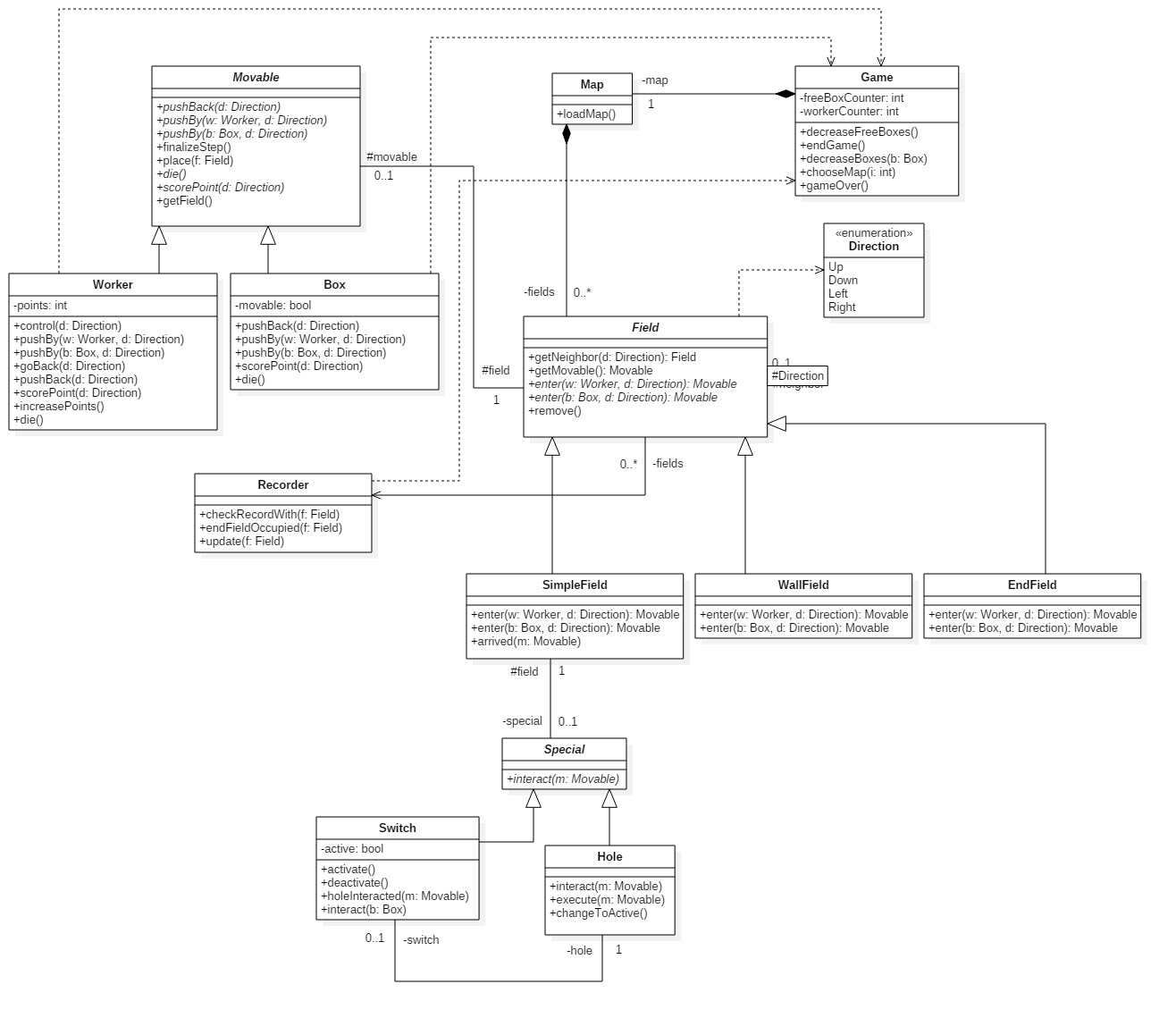
### Worker

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

[02. 22 – 19:00 -  
Balu – 1,5 óra, Márk – 0,5 óra]

[Az objektumkatalógus alapján kiindulva kell megalkotni az objektumorientált analízis modellt. A 3.2, 3.3, 3.4, 3.5 alfejezetek ugyanannak a modellnek a különböző nézetei, ezért egyidőben, egymással összefüggésben készülnek. Megtörténik az objektumkatalógusban tárgyalt objektumok felelősségének formalizálása osztályokká, attribútumokká, metódusokká . Csak publikus metódusok szerepelhetnek. Megjelennek az interfészek, az öröklés, az absztrakt osztályok. Segédosztályokra még nincs szükség.]

## Statikus struktúra diagramok



## Osztályok leírása

[A diagramon szereplő osztályok bemutatása. Az osztályok ABC sorrendben kövessék egymást. Interfészek esetén az Interfészek, Attribútumok pontok kimaradnak.]

### Box

#### Felelősség

Ennek az osztálynak az egyes példányai fogják reprezentálni a dobozokat a játékban. Őket kell majd a célmezőre eljuttatni. Egy mezőt foglalnak el. Képesek törlődni a pályáról, ha lyukba esnek

#### Ősosztályok

Movable

#### Attribútumok

* **Field field:** az a mező, ahol éppen tartózkodik a láda.

#### Metódusok

* **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.) Ha a mezőn ahova visszatolódik, van ,ár valami akkor továbbhívja önmagát.
* **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. Ha a továbbkerülés helyén található további Movable, akkor továbbhívja önmagát.
* **void pushBy(Worker w, Direction d):** A láda (**Box**) egy játékos által való megtolásákor hívjuk meg, intézi hogy a láda a megfelelő helyre kerüljön. Ha a továbbkerülés helyén található további Movable, akkor továbbhívja önmagát.
* **void scorePoint(Direction d):** Amennyiben a célmezőre tolódott a láda, ennek a függvénynek a segítségével végezhetjük el a megfelelő játékos pontjainak növelését. A Box esetén mindössze annyit tesz, hogy a megfelelő irányban álló Movable-nek továbbadja a hívást.
* **void die():** Ez a függvény felel az entitás elpusztításáért, illetve csökkenti a szabad dobozok számát. Fontos, hogy itt figyelembe vesszük, hogy az elpusztított doboz szabad volt e.

### EndField

#### Felelősség

Célmező, amire a dobozokat (**Box**)juttatni kell. Amennyiben rálép egy doboz, akkor az statikussá válik ott. **Worker** esetén **SimpleField**-ként viselkedik.

#### Ősosztályok

Field

#### Attribútumok

* **void arrived(Worker w):** Véglegesíti a mezőre való érkezést. Beállítja a movable referenciáját a kapott objektumra.
* **void arrived(Box w):** Működése a az előzővel megegyező.
* **Movable movable:** referencia egy **Movable** leszármazottra.
* **Field neighbor[Direction]:** Tárolja a szomszédos mezőket a lehetséges irányokban.

#### Metódusok

* **Field getNeighbor(Direction d):** Visszaadja az átadott irányban lévő szomszédos mezőt.
* **Movable getMovable():** Visszaadja a mezőn lévő **Movable** leszármazott referenciáját.
* **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 található önmagán, és visszaadja annak a referenciáját (ha semmi akkor esetén null-t ad).
* **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). Ha nem található rajta semmi, akkor elindítja a pontszerzését a játékosnak, frissíti a **BoxRecorder**-t és visszatér null-al. Ha található rajta valami, akkor visszaadja a hívó referenciáját.

### Field

#### Felelősség

Absztrakt ősosztály, amiből a mező objektumok származnak. Tárolja a rajta álló **Movable**-t, és ismeri a szomszédait.

#### Attribútumok

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

#### Metódusok

* **void arrived(Box b):** Absztrakt függvény, leszármazottakban kerül

megvalósításra.

* **void arrived(Worker w):** Absztrakt függvény, leszármazottakban kerül

megvalósításra.

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

### 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 decreaseFreeBoxes():** csökkenti a még mozgatható dobozok számát 1-el, amennyiben egy doboz elakad.
* **void gameOver():** Akkor hívódik meg, ha minden játékos meghal. Ebben az esetben senki sem kerül ki nyertesként a játékból.
* **void endGame():** Akkor hívódik meg, ha a mozgatható dobozok száma nullára csökken. Ekkor a legtöbb pontot elért játékos nyer.
* **chooseMap(int n):** Meghívja az n-edik pálya betöltését.
* **decreaseBoxes(Box b):** Abban az esetben hívódik meg, amennyiben egy doboz megsemmisül. A **BoxRecorder** alapján ha ez a megsemmisült doboz mozgatható volt, akkor továbbhív a *decreaseFreeBoxes()* függvényre.

### Hole

#### Felelősség

Megsemmisíti a rajta álló **Movable** osztálból származó példányt.

#### Interfészek

Special

#### Attribútumok

* **SimpleField field:** Referencia a mezőre, melyen lyuk áll
* **Switch switch:** Referencia a hozzá tartozó kapcsolóhoz

#### Metódusok

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

### Map

#### Felelősség

A pálya betöltése a szerepe.

#### Attribútumok

* **ArrayList<Field>fields:** Az egyes mezők tárolása egy ArrayList-ben.

#### Metódusok

* **void loadMap():** a pálya betöltését intézi.

### Movable

#### Felelősség

Egy absztrakt ősosztály, amiből a mozgatható objektumok származnak. Tárolja, hogy éppen melyik mezőn áll. Kezeli az ütközéseket más hasonló objektumokkal. Továbbá a pontozás elkezdését lebonyolítja.

#### 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.
* **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.

### BoxRecorder

#### Felelősség

A mozgatható dobozok nyilvántartásának irányítása. Eldönti egy mezőről, hogy az azon lévő doboz beszorult e. Ismeri az éppen ilyen mezőket.

#### Attribútumok

* **ArrayList<Field> fixFields**

#### Metódusok

* **void endFieldOccupied(Field f)**: Az átadott mezőt beleteszi a fix mezők nyilvántartásába, hiszen ez akkor hívódik ha egy célmezőre doboz került.
* **void update(Field f)**: A kapott mezőről megállapítja a nyilvántartás alapján, hogy a rajta lévő doboz beszorult e, vagyis, hogy a mező „fix”. (Ha a mezőnek van pl. három már a nyilvántartásban szereplő szomszédja, akkor biztosan fi, továbbá ha két nem szemközti szomszédja benne van, akkor is).
* **checkRecordWith(Field f)**: Az adott mezőt keresi a nyilvántartásban, és ha nincs benne, akkor csökkenti a szabad mezők számát a **Game**-ben.

### SimpleField

#### Felelősség

Közönséges mező, amire szabadon rá lehet lépni. Ezeken közlekednek a munkások és a ládák. Lehet rajta **Special** interfészt implementáló osztály objektuma. Ismeri a szomszédait.

#### Ősosztályok

Field

#### Attribútumok

* **Movable movable:** referencia egy **Movable** leszármazottra.
* **Field neighbor[Direction]:** Tárolja a szomszédos mezőket a lehetséges irányokban.
* **Special special:** Tárolja a mezőn lévő **Special** ősosztályú objektumot.

#### Metódusok

* **void arrived(Box b):** 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** ősosztályú entitás *interact(Movable m)* függvénye. Továbbá beállítja a mező movable referenciáját a kapott objektuméra és frissíti a **BoxRecord**-ert
* **void arrived(Box b):** A működése az előzővel megegyező, de nem frissíti a **BoxRecorder**-t.
* **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 van e rajta valami (movable), és visszaadja annak a referenciáját (ha nincs semmi akkor null-t).
* **Movable enter(Box b, Direction d):** Működése az előző függvénnyel megegyező.
* **Field getNeighbor(Direction d):** Visszaadja az átadott irányban lévő szomszédos mezőt.
* **Movable getMovable():** Visszaadja a mezőn lévő Movable leszármazott referenciáját.

### Special

#### Felelősség

Interfész. A **SimpleField**-eken esetlegesen lévő speciális objektumot reprezentálja (**Hole** vagy **Switch**).

#### Metódusok

* **void interact(Worker w):** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* **void interact(Box b):** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* **void erease():** A movable objektum mezőről való lelépésekor hívódik.

### Switch

#### Felelősség

Egy kapcsolót és hozzá tartozó állapotot tárolja. Továbbá ismeri a hozzátartozó lyukat, azt szabályozza.

#### Interfészek

Special

#### Attribútumok

* **SimpleField field:** referencia a mezőre, melyen a kapcsoló található
* **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(Worker w):** Lekezeli, ha munkás érkezik a kapcsolóra, ekkor nem vált állapotot.
* **void interact(Box b):** Lekezeli, ha doboz érkezik a mezőre. Aktiválja a kapcsolót és szól a lyuknak, hogy megváltozott az állapota.
* **void erease():** Meghívja a kapcsolót deaktiváló függvényt.

### WallField

#### Felelősség

Olyan mező objektum, mely egy falat reprezentál. Nem engedi átlépni rajta a **Movable-**öket, azokat visszalépteti (az ellenkező irányba) ha rálépnek.

#### Attribútumok

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

#### Metódusok

* **Field getNeighbor(Direction d):** Visszaadja az átadott irányban lévő szomszédos mezőt.
* **Movable getMovable():** Visszaadja a mezőn lévő Movable leszármazott referenciáját.
* **Movable enter(Worker w, Direction d):** Megakadályozza, hogy egy **Worker** rálépjen a mezőre. Ezt azzal éri el, hogy annak a referenciáját adja vissza, aki rálépett.
* **Movable enter(Box b, Direction d):** Megakadályozza, hogy egy **Box** rálépjen a mezőre. Ezt azzal éri el, hogy annak a referenciáját adja vissza, aki rálépett.

### 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

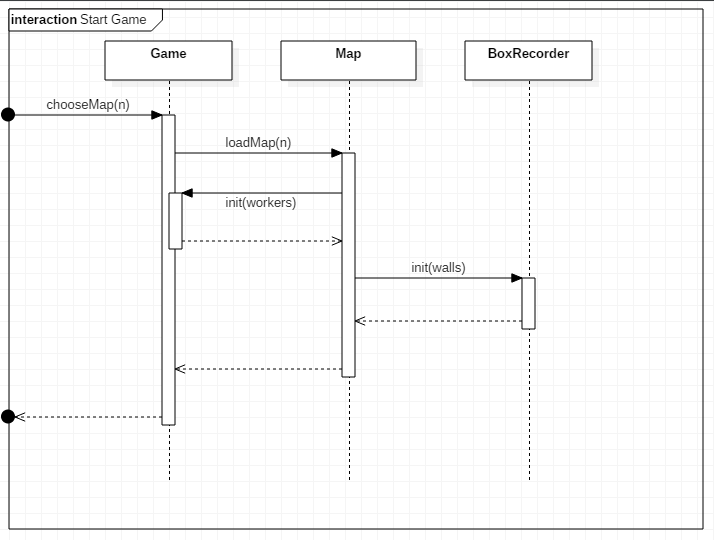
* **Field field:** referencia a mezőre, amin a munkás épp tartózkodik.
* **int points:** a játékos pontjai, amiket az egyes ládák célmezőre juttatásáért kaphat.

#### Metódusok

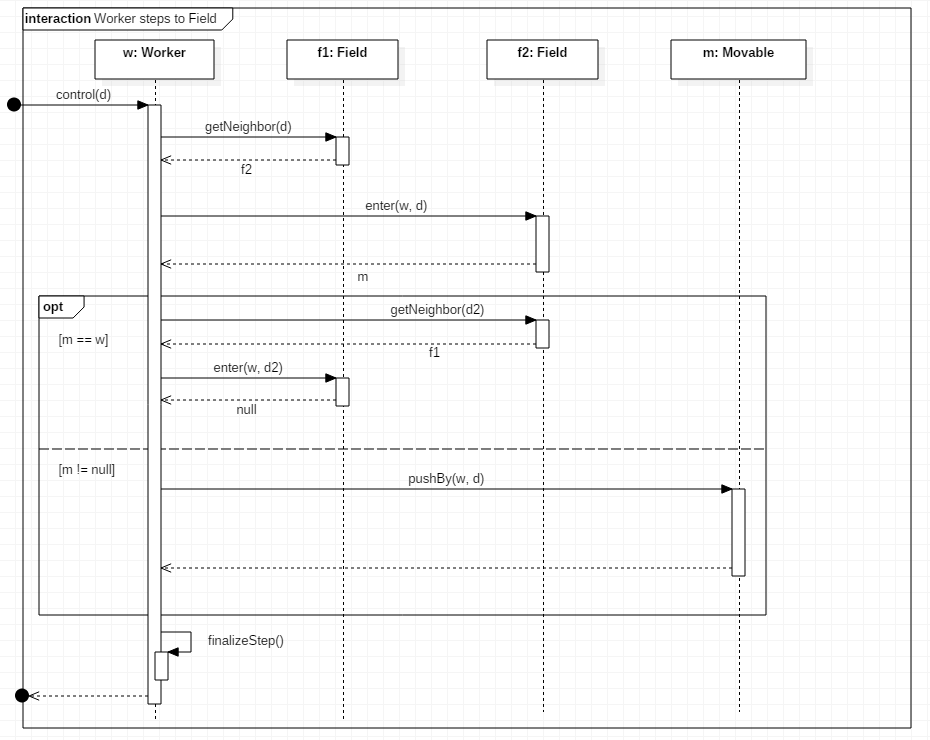
* **void control(Direction d):** A munkás (**Worker**) irányításáért felelős függvény. A paraméterben kapott irányban fogja megpróbálni elmozdítani a munkást a szomszédos mezők figyelembe vételével (Mindez önakaratból történik, tehát a felhasználó beavatkozására).
* **void pushBy(Box b, Direction d):** 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. Ekkor ez megakadályozódik, és a munkás a másikat visszalépteti.
* **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 pushBack(Direction d):** Ennek a függvénynek ott van szerepe, amikor egy munkás a dobozt a falba bele akarná tolni. Ha a visszatolás irányába, nem található semmi, akkor odalép. Ellenkező esetben meghal a munkás.
* **void scorePoint(Direction d):** A pontok növeléséért felelős, méghozzá az *increasePoints()* függvény segítségével.
* **void die():** Ez a függvény felel az entitás elpusztításáért, illetve csökkenti a munkások számát a **Game**-ben.
* **void increasePoints():** A munkás pontjainak növelése.

## Szekvencia diagramok

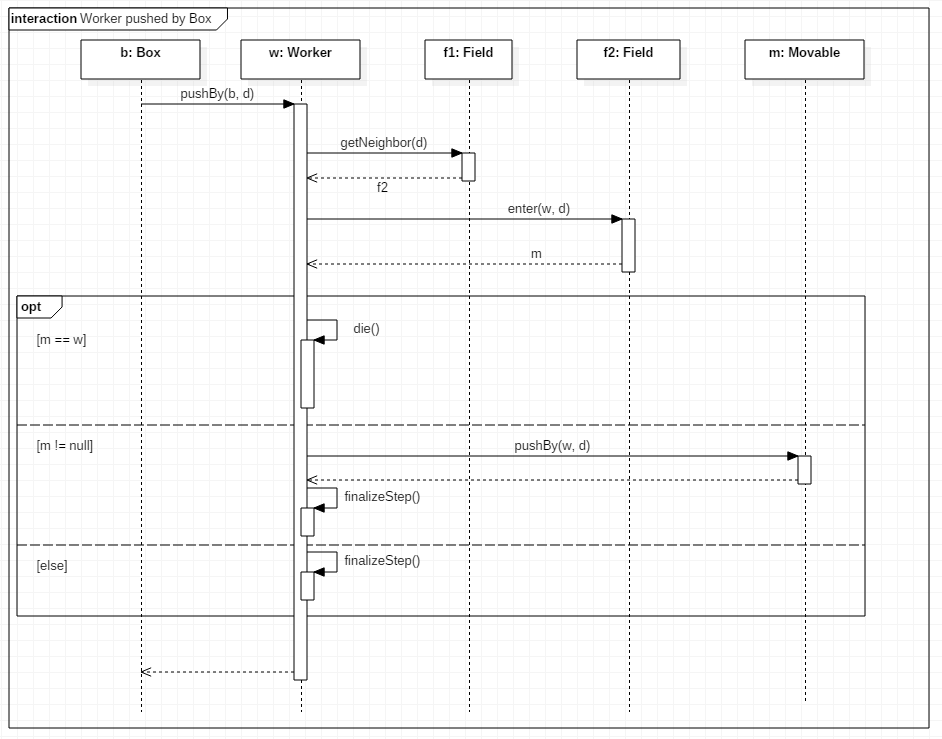
### Start Game



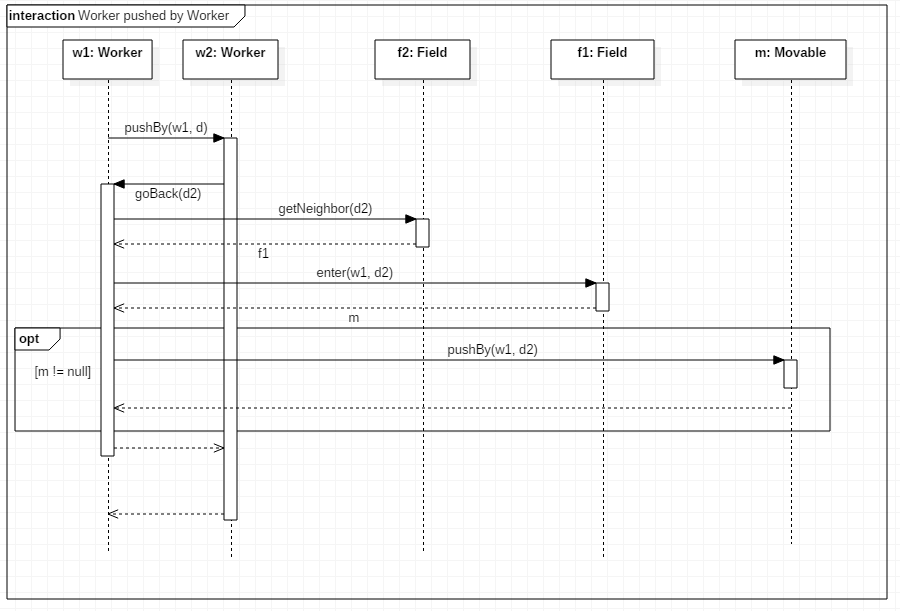
### Worker steps to Field



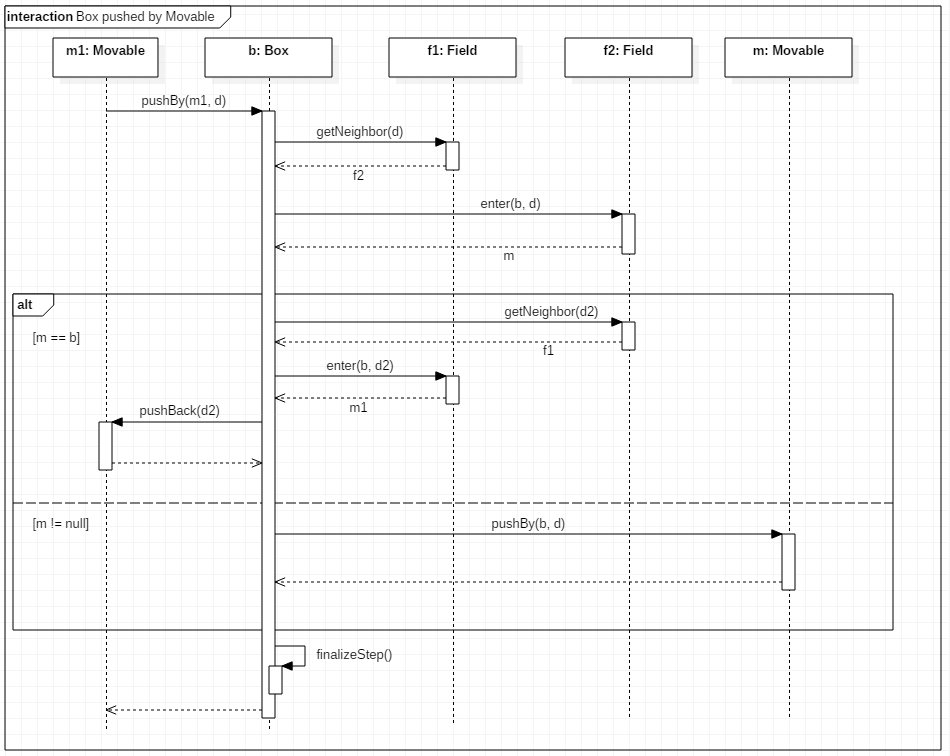
### Worker pushed by Box



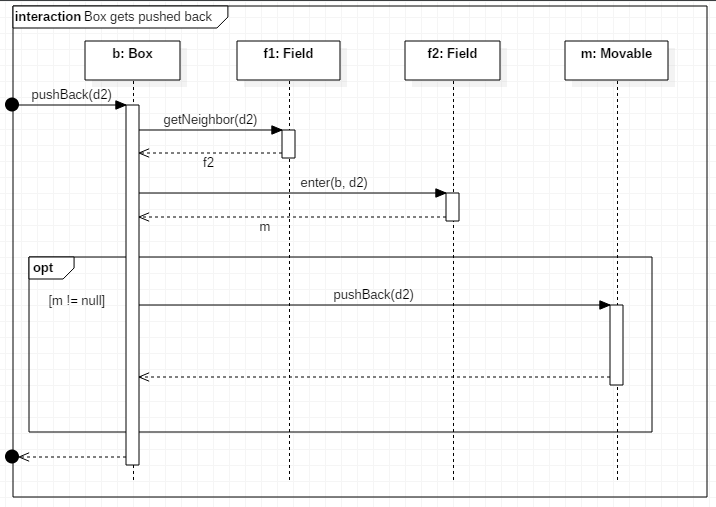
### Worker pushed by Worker



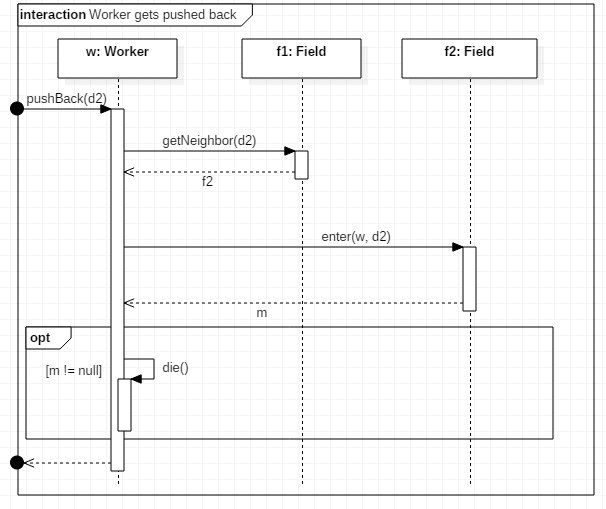
### Box pushed by Movable



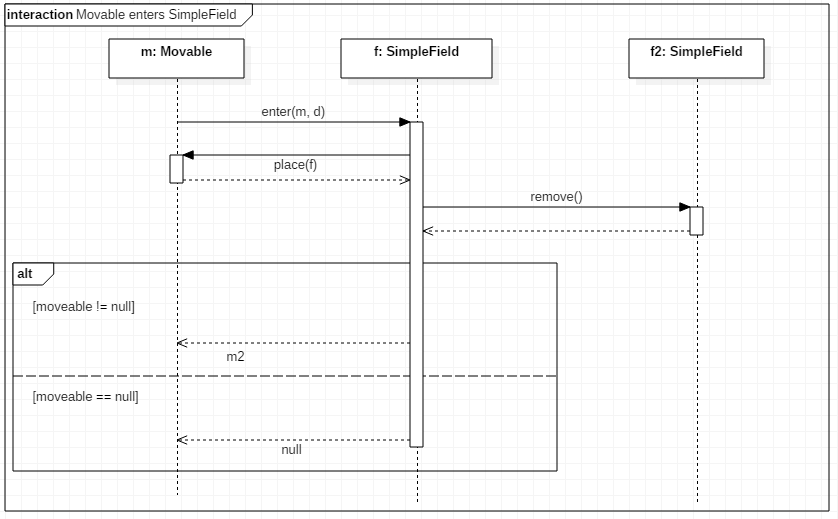
### Box gets pushed back



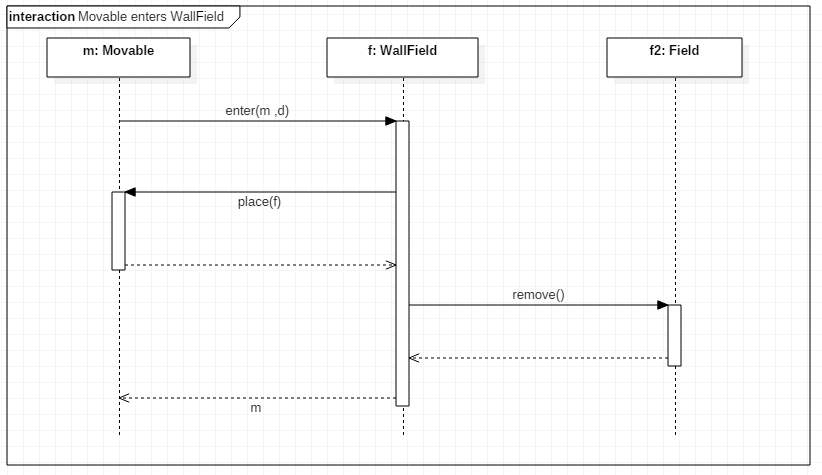
### Worker gets pushed back



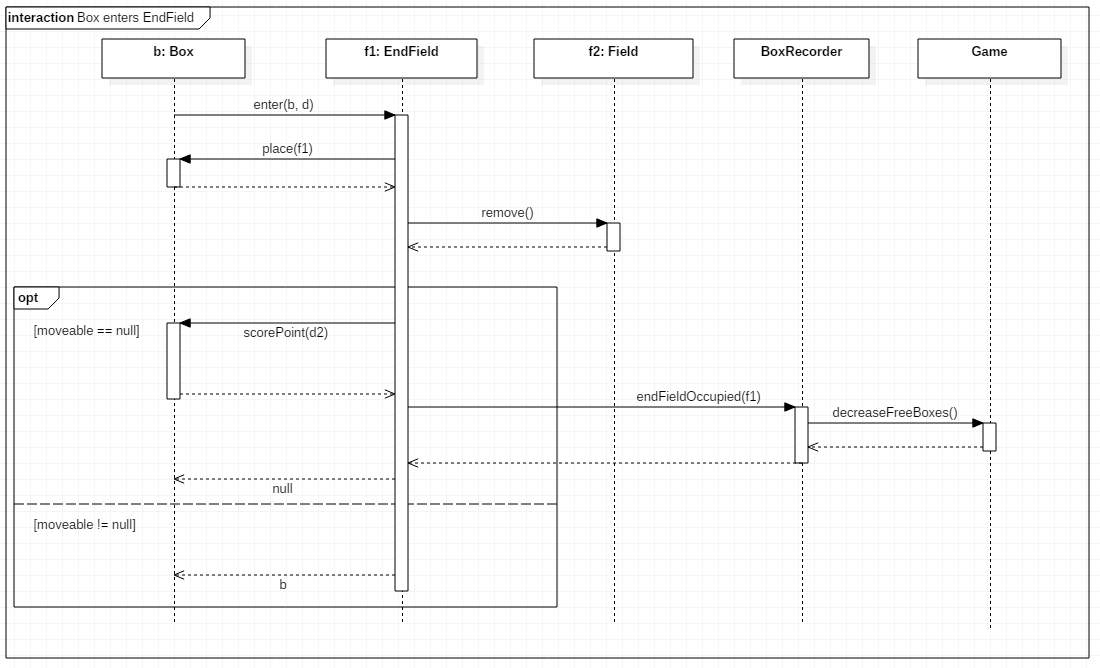
### Movable enters SimpleField



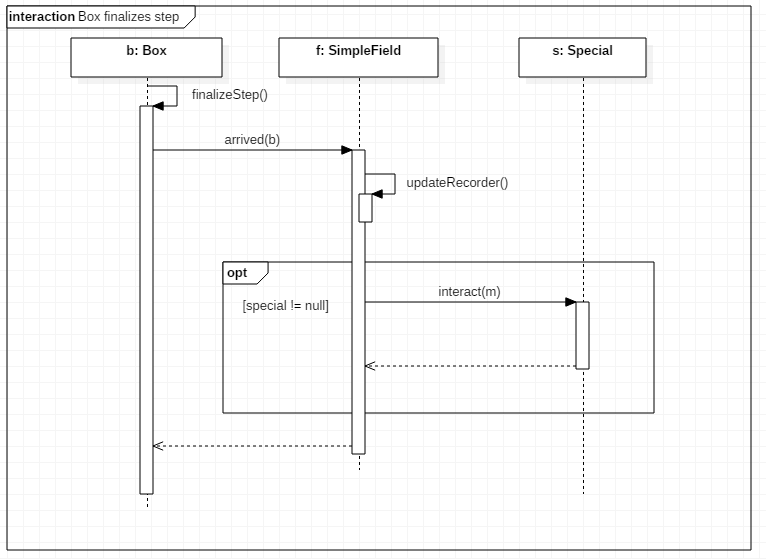
### Movable enters WallField



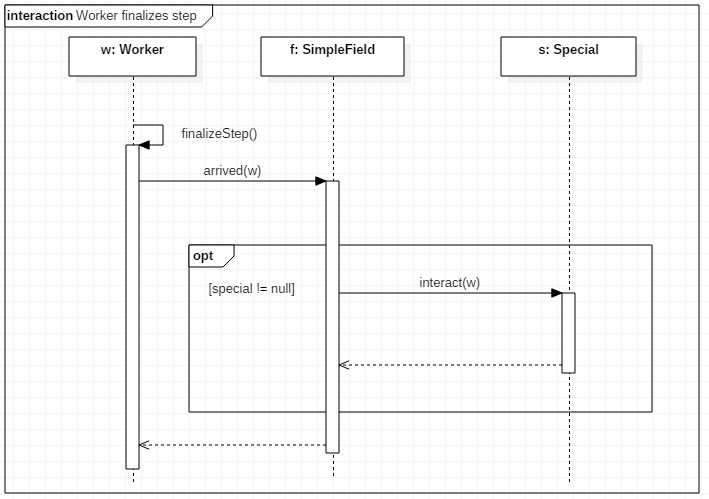
### Box enters EndField



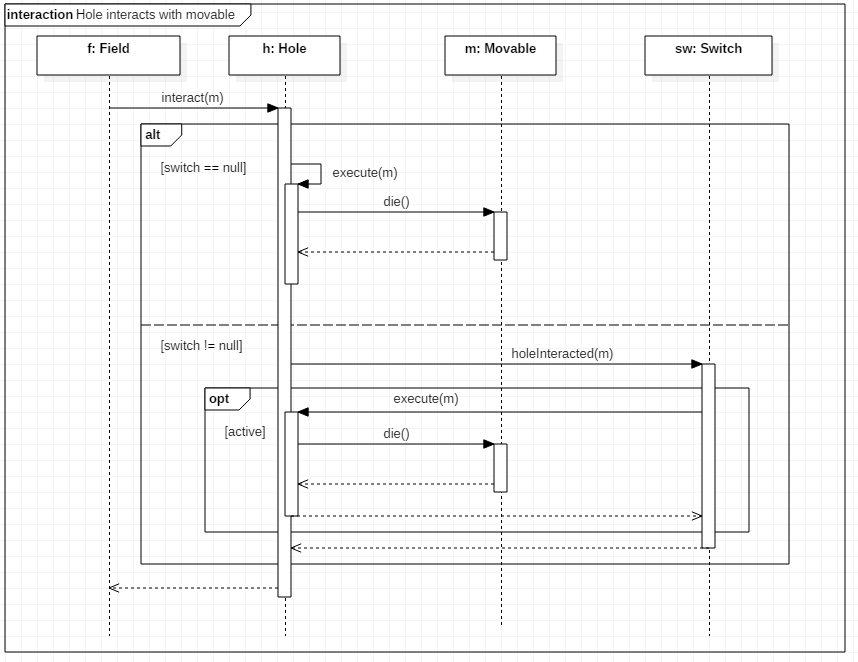
### Box finalizes step



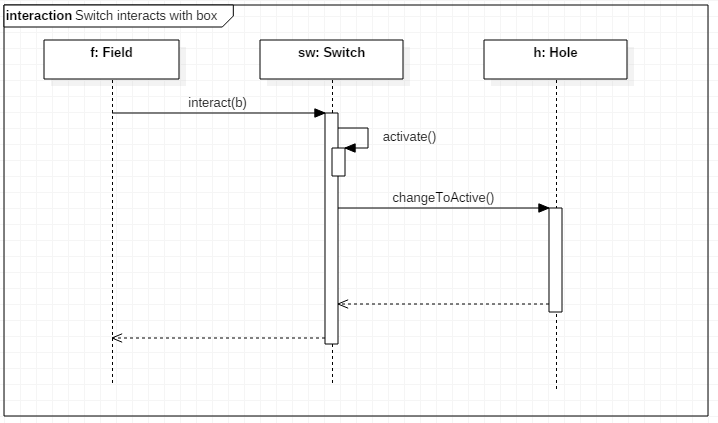
### Worker finalizes step



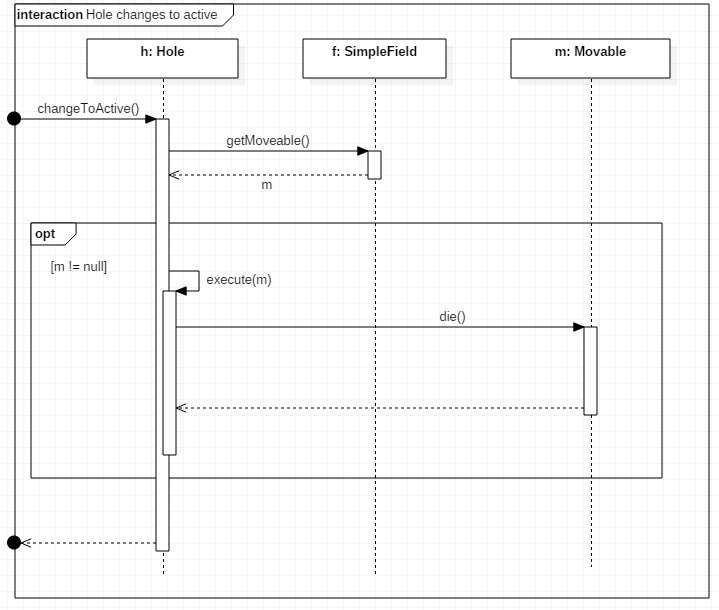
### Hole interacts with movable



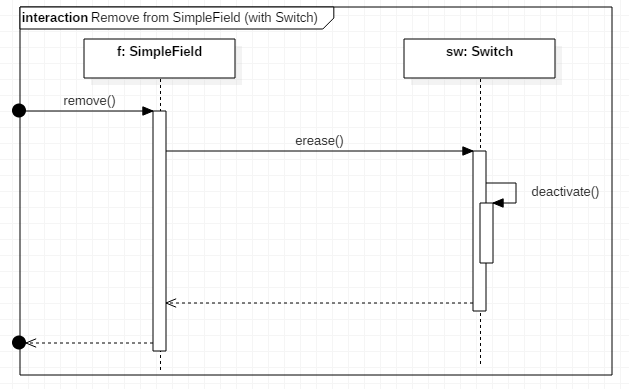
### Switch interacts with box



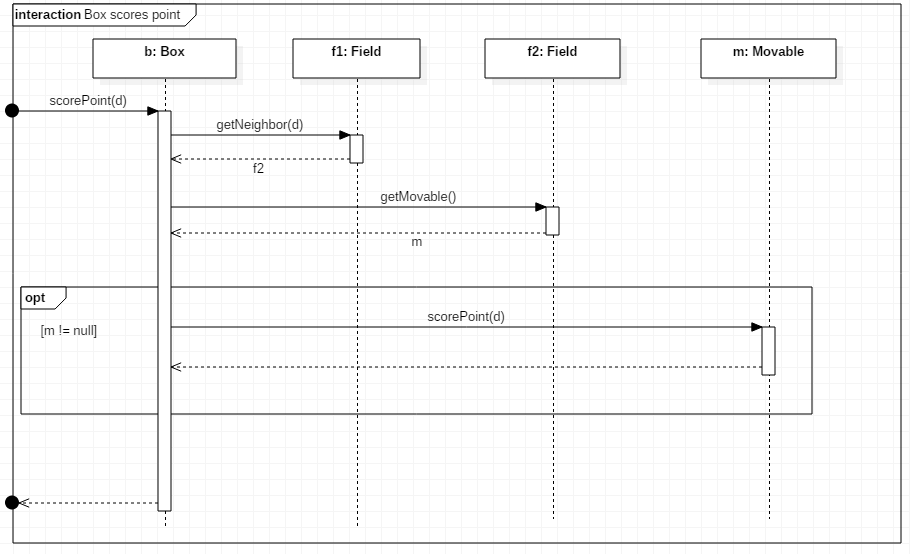
### Hole chnages to active



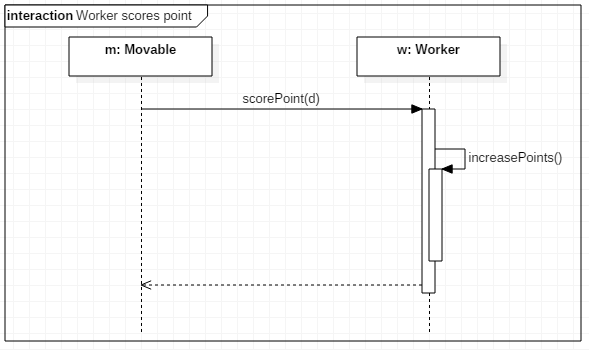
### Remove from SimpleField (with Switch)



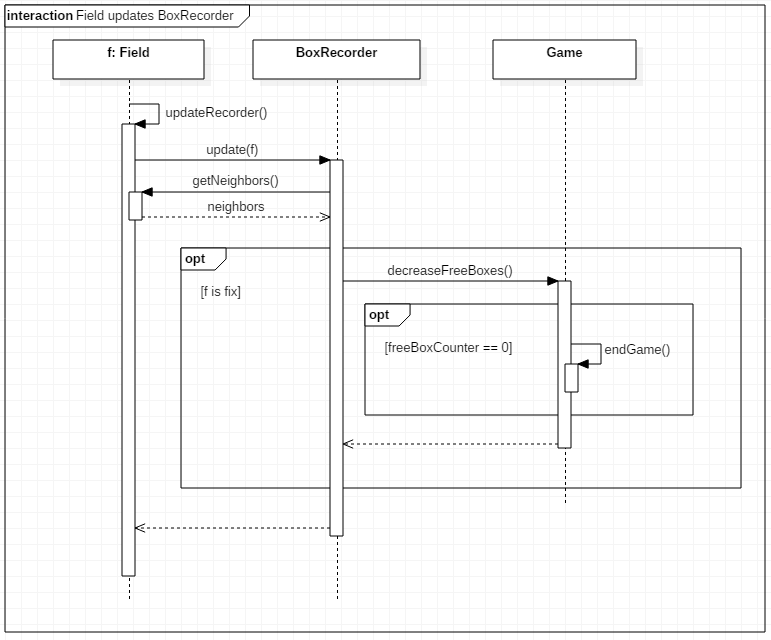
### Box „scores point”



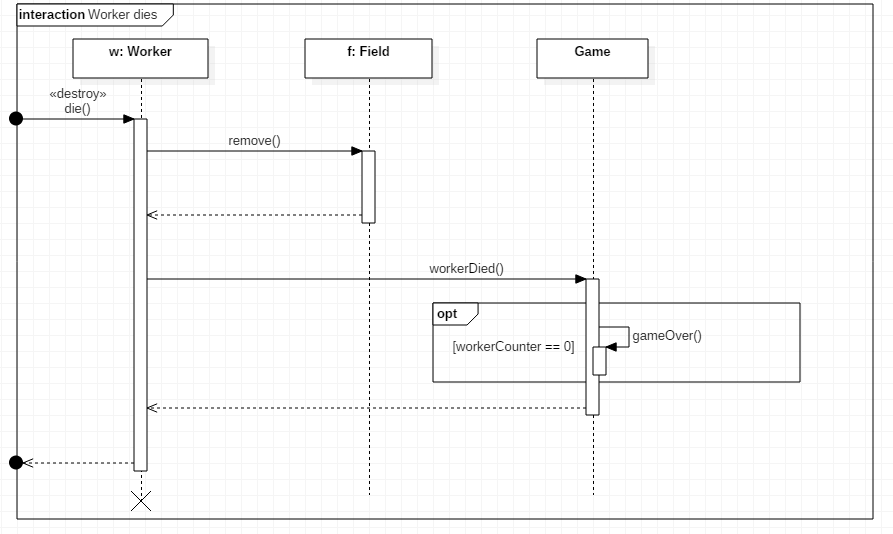
### Worker scores point



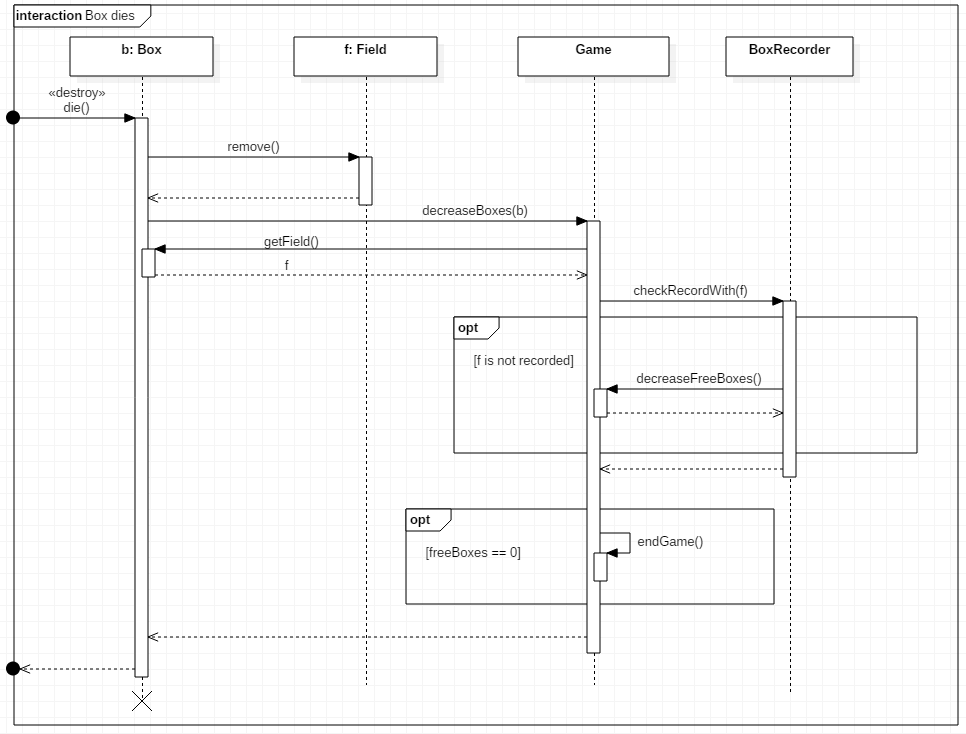
### Field updates BoxRecorder



### 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** |
| 2010.03.21. 18:00 | 2,5 óra | Horváth  Németh  Tóth  Oláh | Értekezlet.  Döntés: Horváth elkészíti az osztálydiagramot, Oláh a use-case leírásokat. |
| 2010.03.23. 23:00 | 5 óra | Németh | Tevékenység: Németh implementálja a tesztelő programokat. |
| … | … | … | … |