

2. Követelmény, projekt, funkcionalitás

2.1 Bevezetés

2.1.1 Cél

Jelen dokumentum a „Single Point of Failure” csapat „Killer Sokoban” implementációjával kapcsolatos általános tudnivalókat foglalja össze.

2.1.2 Szakterület

A készítendő szoftver célja a Szoftver projekt laboratórium tárgy oktatói által megállmodott „Killer Sokoban” játék megvalósítása, az oktatók és a játék fejlesztőinek minél nagyobb megelégedésére és élményére koncentrálva. Másodlagos célkitűzés a gondolkodtatónak játékok újabb példányának létrehozása minden, illesmire éhező embertársunk igényeinek kielégítésére.

2.1.3 Definíciók, rövidítések

ill.: illetve

2.1.4 Hivatkozások

Szoftvertechnológia órai jegyzet és előadásdiák

Prog. 3 órai jegyzet és előadásdiák

2.1.5 Összefoglalás

A továbbiakban részletesebben is ismertetésre kerülnek a készítendő szoftver sajátosságai, mind a leendő felhasználók, mind a fejlesztők szempontjából, a számukra érdekes aspektusokat részletezve (pl.: funkciók, ill. követelmények), valamint a szoftver megvalósításával kapcsolatos információk, tervezetek.

2.2 Áttekintés

2.2.1 Általános áttekintés

A program struktúrája alapvetően két részből áll: a pálya elemeiből és a pályán lévő dolgokból.

A pálya elemei alatt blokkok értendők, melyek lehetnek falak (vagy oszlopok) részei, járható padlódarabok, lyukak, ki/bekapcsolható lyukak illetve az ezekhez tartozó kapcsolók. A pályán lévő dolgok lehetnek tologatható lánkok vagy a játékosok által irányított munkások. Természetesen a pályán való mozgáshoz a pályaelemeknek és a pályán lévő dolgoknak kommunikálniuk kell egymással, ehhez mindenki egy interfész valósít meg.

A felhasználó a játékban szereplő munkások közül egyet irányíthat, így részt vehet a játékban.

2.2.2 Funkciók

A szoftver egy számítógépes játékprogram, amelyen egyszerre több játékos tud játszani egymás ellen. A játék világa egy raktárépület, ahol a játékosok rakodómunkások bőrébe bújhatnak. A raktárban lánkok helyezkednek el, amiket a munkások tologathatnak az épület padlóján. Fontos, hogy a lánkokat csak előre lehet tolni, ha más irányba is szeretnénk

elmozdítani azt, akkor másik oldalról kell mellé állnunk (pl. oldalra nyomni vagy húzni nem lehet őket.) A látások egymást is tolhatják, így a munkások egyszerre egész látásorokat pakolhatnak.

A raktár padlója egységes, négyzet alakú blokkokból áll, ezek a játékban felülről látszanak. minden látás és munkás is pontosan egy ilyen blokkot foglalhat el, annak teljes egészét lefed, és a szomszédos blokkokba nem lóg bele. A munkások a pályán azonos sebességgel sétálhatnak, azaz adott időközönként a megfelelő szomszédos blokkra kerülhetnek. A tolás művelete egyébként úgy működik, hogy közvetlenül a teher mellé állunk, és elkezdünk az irányába gyalogolni.

A játékosoknak vigyázniuk kell, mert az épületben többféle akadály is az útjukba kerülhet. Bizonyos blokkok falak vagy oszlopok részei, amikre természetesen sem a munkások nem tudnak rálépni, sem látás nem toltható. Más blokkok a padlóban lévő lyukak, amelyekre ha látás kerül, az megsemmisül és eltűnik, ha pedig munkás lép, akkor az meghal. Néhány lyuk aktiválható és deaktiválható, ezekhez a padlón blokknyi kapcsolók tartoznak. Ha a kapcsolón látás van, akkor a lyuk lyukként viselkedik, egyébként járható padlódarabként.

További zavaró tényező, hogy a munkások egymást is tudják akadályozni, esetleg megölni. Egy játékos eltolhatja egy másik passzív játékos karakterét, akár látásával keresztül is. Ha egy embert falhoz préselünk, az ember meghal.

A padlón ki van jelölve néhány speciális járható blokk, az úgynevezett célblokkok. Ezek különleges tulajdonsága, hogy a rájuk látását toló játékosnak jóváírnak egy pontot, a róluk látás eltolónak pedig minusz egyet. A játékosok célja, hogy ök szerezzék a legtöbb pontot.

A játékmenet véget ér, ha minden látás célblokkon van, vagy a munkások már nem tudnak többet oda juttatni. A program futása során a pontokat folyamatosan számolja és játékosonkénti bontásban kijelzi. A játék körökre osztott, minden játékos minden körben pontosan egyet léphet a saját karakterével.

2.2.3 Felhasználók

A felhasználóknak a szoftver használatához nincs szükségük különösebb előképzettségre. A programot egyszerre többen is használhatják.

2.2.4 Korlátozások

A programnak egy játéktól elvárható mértékben stabilan és helyesen kell működnie: nem fagyhat le, nem léphet ki a felhasználó akarata nélkül és a követelményekben meghatározott módon kell játszania és a felhasználókat játszani engednie.

2.2.5 Feltételezések, kapcsolatok

Az előadások anyagait és a jegyzeteinket a projekt szakszerűbb átgondolásához használtuk.

2.3 Követelmények

2.3.1 Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment

R00	Létezik egy pálya (raktárépület).	bemutatás	alapvető	feladatkírás	View Game	
R01	A pálya négyzetes mezőkből áll.	bemutatás	alapvető	feladatkírás	View Game	
R02	A raktárban munkások vannak, akiket a játékosok irányítanak.	bemutatás	alapvető	feladatkírás	View Game, Move	
R03	A munkások pontosan egy négyzetet foglalnak el.	bemutatás	alapvető	feladatkírás	View Game	
R04	A raktárban lágak vannak.	bemutatás	alapvető	feladatkírás	View Game	
R05	A lágak pontosan egy négyzetet foglalnak el.	bemutatás	alapvető	feladatkírás	View Game	
R06	A lágak a munkások által eltolhatók a szomszédos mezőkre.	bemutatás, kiértékelés	alapvető	feladatkírás	Move	
R07	A mezők típusa lehet egyszerű, fal, oszlop, lyuk, irányítható lyuk, cél, kapcsolót tartalmazó.	bemutatás	fontos	feladatkírás	View Game	
R08	Egy mező pontosan egy fajta lehet.	bemutatás	fontos	csapat	View Game	Egyszerűség kedvéért.
R09	A fal és oszlop típusú mezőkre nem lehet rálépni, sem lágát rátolni.	bemutatás, kiértékelés	alapvető	feladatkírás	View Game, Move	
R10	Lyuk típusú mezőre lépve a munkás meghal.	bemutatás, kiértékelés	alapvető	feladatkírás	Move	
R11	Eltűnik a lág, ha lyukra tolják.	bemutatás, kiértékelés	alapvető	feladatkírás	Move	
R12	Egy lyukba több lág is eshet egymás után.	bemutatás	fontos	csapat	Move, View Game	

R13	A cél a láták előírt helyre (célmézőkre) tologatása.	bemutatás	alapvető	feladatkírás	Move	
R14	A kapcsoló típusú mezők akkor, és csak akkor aktívak, ha láda van rajtuk.	bemutatás, kiértékelés	alapvető	feladatkírás	View Game	
R15	Az irányítható lyukak akkor, és csak akkor viselkednek lyukként, ha a hozzájuk tartozó kapcsoló aktív.	bemutatás, kiértékelés	alapvető	feladatkírás	View Game	
R16	Egy kapcsoló egy irányítható lyukhoz tartozhat és fordítva.	bemutatás	fontos	csapat	View Game	
R17	A láták egymást el tudják tolni.	bemutatás, kiértékelés	alapvető	feladatkírás	Move	A láták nem nyomhatók össze.
R18	Ha egy munkásra látát tolunk, akkor a munkás automatikusan szomszédos négyzetre tolódik, és tolja maga előtt az esetleges látákat, játékosokat is.	bemutatás, kiértékelés	alapvető	feladatkírás, csapat	Move	
R19	Egy munkást egy másik munkás közvetlenül is el tud tolni. (Az R18-hoz hasonlóan)	bemutatás, kiértékelés	fontos	csapat	Move	
R20	Ha egy munkás nem tud eltolódni, akkor meghal.	bemutatás, kiértékelés	alapvető	feladatkírás	Move	pl. fal van mellette, vagy fal miatt eltolhatatlan láda.
R21	Egy munkás közvetlenül “agyonnyomhat” egy másik fal mellett várakozót.	R19 és R20-ból következik	fontos	csapat	Move	

R22	A játék véget ér, ha az összes láda a helyén van, vagy ha már nem lehet többet tolni.	bemutatás, kiértékelés	alapvető	feladatkírás	View Game	
R23	Az nyer, aki ekkorra a legtöbb pontot szerezte.	bemutatás, kiértékelés	alapvető	feladatkírás	View Game	
R24	A játékos egy pontot szerez, minden egy célmezőre tolta ládáját.	bemutatás, kiértékelés	opcionális	csapat	View Game	
R25	A játékos egy pontot veszít, minden egy célmezőről eltolt ládájáról.	bemutatás, kiértékelés	opcionális	csapat	View Game	

2.3.2 Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
R101	A program Java nyelven készül.	bemutatás	alapvető	feladatkiadó	
R102	Az elkészült programkódnak fordíthatónak kell lenni Java SE Developer Kit 8-on.	bemutatás, kiértékelés	fontos	feladatkiadó	
R103	A termékhez ajánlott operációs rendszer a Windows 10.	bemutatás	opcionális	csapat	Bár az elkészült termék a programnyelv választása miatt elvileg crossplatform, az egyszerűség kedvéért, csak Windowsra garantáljuk a helyes működést.
R104	A programnak szüksége lehet fájlkezelési jogokra.	bemutatás	fontos	csapat	Csak abban az esetben, ha fájlból történő pálya beolvasás implementálásra kerül.

R105	A termék fő beviteli eszközei az egér és a billentyűzet.	bemutatás	fontos	csapat	
------	--	-----------	--------	--------	--

2.3.3 Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás
R111	Minden alapvető követelménynek teljesülnie kell.	bemutatás, kiértékelés	alapvető	feladat
R112	A termék futtatásához szükséges a Java Runtime Environment 8 megléte.	bemutatás	alapvető	feladat

2.4 Lényeges use-case-ek

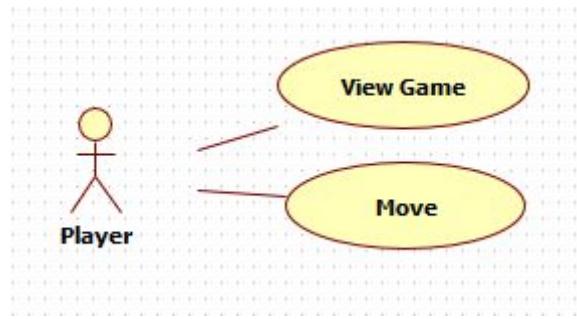
2.4.1 Use-case leírások

Use-case neve	View Game
Rövid leírás	A játékos megtekinti a játékot.
Aktorok	Player
Főforgatókönyv	1. A rendszer kirajzolja a játék aktuális állapotát.
Főforgatókönyv	2. A játékos megtekinti a játék aktuális állapotát.
Alternatív forgatókönyv	2. A A játékos megtekinti a munkások helyét.
Alternatív forgatókönyv	2. B A játékos megtekinti a falak, oszlopok, lyukak, kapcsolók helyét.
Alternatív forgatókönyv	2. C A játékos megtekinti a lánchídak és a célmezők helyét.
Alternatív forgatókönyv	2. D A játékos megtekinti a ponttábla aktuális állapotát.

Use-case neve	Move
Rövid leírás	A játékos a munkást irányítja.
Aktorok	Player
Főforgatókönyv	1. A játékos a munkást a szomszédos mezőre lépteti.

Főforgatókönyv	1. A Ha egy munkás lyukba esik, meghal, és az adott játékos számára vége a játéknak.
Főforgatókönyv	1. B A munkás a lánát, és más munkásokat tolthat odébb.
Alternatív forgatókönyv	1. B. 1 Amennyiben egy eltolt munkás nem fér el, akkor meghal.

2.4.2 Use-case diagram



2.5 Szótár

blokk: lásd mező

célmező: erre a mezőre kell tolni a lánát, szinonima: speciális járható blokk, célblokk

célblokk: lásd célmező

egyszerű mező: nincs rajta se lyuk, se kapcsoló, nem is célmező, szinonima: járható padlódarab

fal: lásd fal típusú mező

fal típusú mező: a munkás nem léphet rá, és nem tolhat rá lánát.

irányítható lyuk: lyuk típusú mezőként viselkedik, ha a hozzá tartozó kapcsoló aktív, egyébként egyszerű mező, szinonima: ki/bekapcsolható lyuk.

járható padlódarab: lásd egyszerű mező

játék: lásd program

játékos: a munkást irányító ember, a program felhasználója

kapcsolót tartalmazó mező: az irányítható lyukat lehet vele irányítani, akkor aktív, ha látát tol rá a játékos.

ki/bekapcsolható lyuk: irányítható lyuk

láda: ezeket tudja a munkás tologatni.

lép: a munkás egy blokkal odébb kerül

lyuk: lásd lyuk típusú mező

lyuk típusú mező: ha a munkás rálép, meghal, ha rátol egy látát, a láda eltűnik.

meghal: a munkást többé nem lehet mozgatni.

megöl: a munkást vagy látát ráveszi, hogy meghaljon

mező: négyzetes mezőkből áll a pálya, ezeken mozoghatnak a munkások, szinonima: blokk, pályaelem

munkás: a játékost reprezentálja, a látákat tudja tologatni a pályán, szinonima: rakodómunkás

nyer: az adott munkás (és az öt irányító játékos) nyer, ha neki van a legtöbb pontja, amikor a játék véget ér

oszlop: lásd fal típusú mező

pálya: a játék helyszíne, szinonima: raktár, raktárépület

pályaelem: lásd mező

program: az elkészített termék

rakodómunkás: lásd munkás

raktár: lásd pálya

raktárépület: lásd pálya

speciális járható blokk: lásd célmező

tol: a munkás a látát vagy egy másik munkást a szomszédos szabad mezőre mozdít

veszít: nem nyer, amikor vége a játéknak

2.6 Projekt terv

2.6.1 Ütemterv

Határidő	Feladat	Felelős
febr. 26.	Analízis modell kidolgozása 1. - beadás	Schulcz

márc. 5.	Analízis modell kidolgozása 2. - beadás	Schulcz
márc. 12.	Szkeleton tervezése - beadás	Zalavári
márc. 19.	Szkeleton - beadás és a forráskód herculesre való feltöltése	Zalavári
márc. 26.	Prototípus koncepciója - beadás	Benkő
ápr. 9.	Részletes tervezek - beadás	Benkő
ápr. 16.	Prototípus készítése, tesztelése	Benkő
ápr. 23.	Prototípus - beadás és a forráskód, a tesztbemenetek és az elvárt kimenetek herculesre való feltöltése	Benkő
máj. 2.	Grafikus felület specifikációja - beadás	Takács
máj. 7.	Grafikus változat készítése	Takács
máj. 14.	Grafikus változat - beadás és a forráskód herculesre való feltöltése	Takács
máj. 18.	Összefoglalás - beadás és feltöltés	Schulcz

2.6.2 Erőforrások

Dokumentálásra használt eszközök: Microsoft Word, Google Docs

Kommunikáció: Személyesen, Messenger

Modellező eszköz: WhiteStarUML

Fejlesztőeszköz: Eclipse/IntelliJ

Dokumentumok megosztása: Google Drive

Forráskód megosztása, verziókezelése: Git

2.7 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.02.14. 12:15	1 óra	Teljes csapat	Konzultáció
2018.02.14. 13:15	45 perc	Teljes csapat	Értekezlet: Nem specifikált követelmények meghatározása, Use-case-ek

2018.02.14. 17:15	1 óra	Teljes csapat	Értekezlet: Tisztázatlan kérdések megbeszélése, elérhetőségek cseréje, Döntés: 2.1 Benkő, hat. idő: 02.16. 17:00 2.2 Schulcz, 02.18.-ig 2.3 Zalavári, 02.18.-ig 2.4, 2.5 Takács, 02.18.-ig
2018. 02. 15. 13:30	30 perc	Benkő	Tevékenység: 2.1.1, 2.1.2 alszakaszok megírása
2018. 02. 15. 14:15	25 perc	Teljes csapat	Értekezlet Döntés: Az analízismodellt Schulcz és Zalavári a hétvégén egymástól függetlenül kidolgozza nagy vonalakban, a csapat 02. 20.-án (kedd) reggel megvitatja.
2018. 02. 15. 20:30	30 perc	Schulcz	Tevékenység: 2.2.1, 2.2.3, 2.2.4 alszakaszok megírása.
2018. 02. 16. 8:00	1 óra	Zalavári	Tevékenység: 2.3 szakasz elkészítése
2018. 02. 16. 9:20	20 perc	Benkő	Tevékenység: 2.1.4, 2.1.5 alszakaszok megírása
2018. 02. 16. 17:00	50 perc	Schulcz	Tevékenység: 2.2.2 alszakasz megírása
2018. 02. 16. 22:10	5 perc	Schulcz	Tevékenység: 2.2.5 alszakasz elkészítése
2018. 02. 17. 13:00	30 perc	Zalavári	Tevékenység: 2.3 szakasz pontosítása, javítása
2018. 02.17. 18:30	45 perc	Takács	Tevékenység: 2.4.1, 2.4.2 alszakasz elkészítése.
2018. 02.18. 11:00	60 perc	Takács	Tevékenység: 2.5 alszakasz elkészítése
2018. 02.18. 21:00	90 perc	Takács, Zalavári	Tevékenység: 2.6 alszakasz elkészítése, dokumentum formázása

3. Analízis modell kidolgozása

3.1 Objektum katalógus

3.1.1 Láda

A raktár életének egykedvű szemlélői. Biztosítják a saját mozgathatóságukat, de önhatalmúlag nem mozognak. Emellett a pontozási rendszerben van közreműködő szerepük.

3.1.2 Mező

Rajtuk tartózkodnak a munkások és a lánkok. Az összes mező halmaza alkotja a tulajdonképpeni pályát. Felelősségeik a szomszédos mezők irányonkénti ismerete (pályareprezentáció), ill. a rajtuk található dolgok mozgatásának biztosítása.

3.1.3 Munkás

A játék aktív mozgatói, formálói. Ők a belépési pontok a játékosok számára a raktár világába, a felhasználók rajtuk keresztül tudnak a játon belül cselekedni. Ők indítják a raktárban található lánkok mozgatását a mezőkkel kommunikálva. Az egyes játékosok pillanatnyi pontjait is ők tartják számon.

3.1.4 Lyuk

A belekerülő (rá lépő) tárgyakat vagy embereket megsemmisíti. A “megsemmisítéshez” tartozó járulékos feladatak tartoznak felelősségeik közé.

3.1.5 Célmező

A mező összes sajátosságával és felelősséggel rendelkezik. Emellett a pontszámításban játszik közre.

3.1.6 Kapcsoló

A mező összes sajátosságával és felelősséggel rendelkezik. Annak függvényében, hogy mi áll rajta, utasításokat ad bizonyos speciális mezőknek (kapcsolható lyukak).

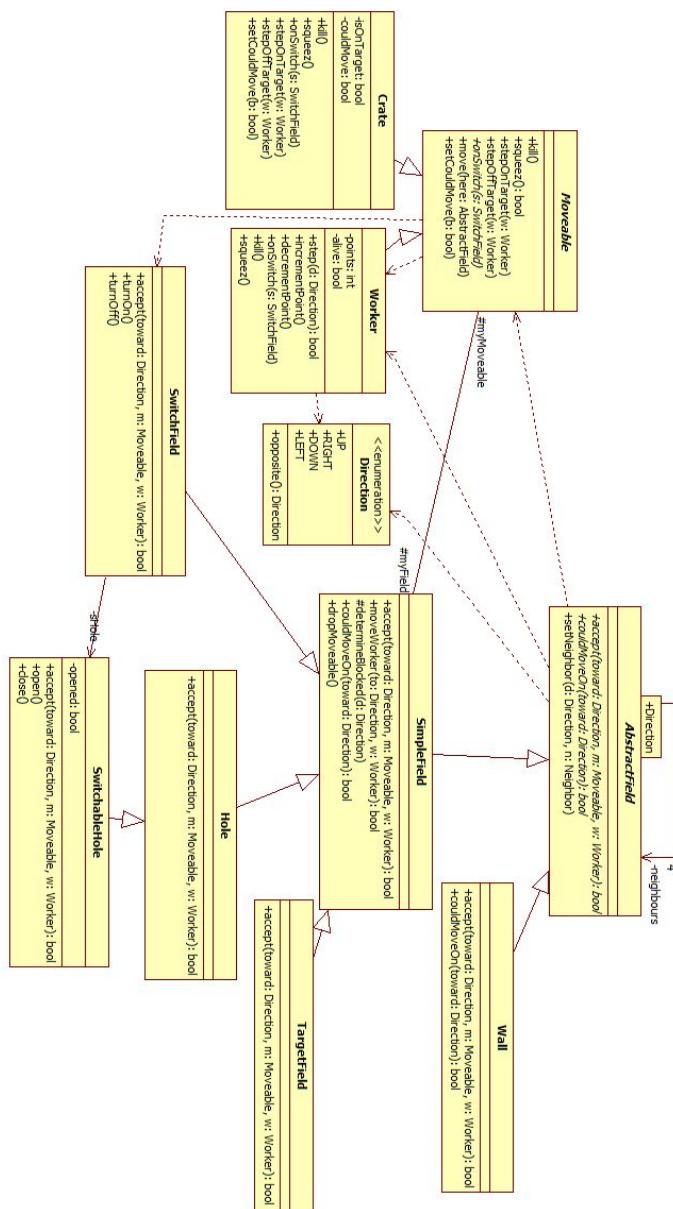
3.1.7 Kapcsolható lyuk

A hozzá tartozó kapcsoló utasításainak megfelelően a lyuk vagy (xor) a normális mező viselkedésének megfelelően működik. A kettő közötti döntés a legfőbb feladata.

3.1.8 Fal

A mezők között helyenént előforduló, átváratlan elem. Felelőssége, hogy a rálépni kívánó dolgoknak ezt (ti. a rálépést) ne engedje meg, valamint, hogy ez a meg-nem-engedés hibamentes legyen.

3.2 Statikus struktúra diagramok



3.3 Osztályok leírása

Az örökölt metódusokat (és attribútumokat) csak indokolt esetben (jellemzően felüldefiniálás esetén) tüntettem fel.

Mivel a modellünkben nincsenek interfések, ezért azon pontokat mindenhol kihagytam. Az attribútumok általában privát vagy védett láthatóságúak, de egy-két fontosabbat a könnyebb érthetőség miatt feltüntettem a leírásnál - Zalavári

3.3.1 AbstractField

• Felelősség

Egy mezőt reprezentál a játékban. Absztrakt alaposztály, konkrét példányai lehet például egy egyszerű járható mező, oszlop, lyuk, stb.

- **Ősosztályok**
- **Attribútumok**
 - **AbstractField neighbours:** A mező ismeri a szomszédait.
- **Metódusok**
 - **abstract bool accept(Direction toward, Moveable m, Worker w):** Ez a függvény azt jelzi a mező számára, hogy szeretnék rálépni. Ekkor a mező, ha szabad, vagy azzá tudja tenni magát, befogadja a paraméterül kapott Moveable-t. Ellenkező esetben nem csinál semmit, és hamis értékkel tér vissza.
 - **abstract bool couldMoveOn(Direction toward):** Megkérdezi a mezőtől, hogy ha egy tolás következne be a paraméterként átadott irányba, akkor ez mennyire lenne “könnyű”. A pontos algoritmusa még nem ismert, de csak a játék végének detektálása miatt szükséges függvény.
 - **void setNeighbor(Direction d, Neighbor n):** Az adott irányba beállítja, hogy ki a mező szomszédja.

3.3.2 Crate

- **Felelősség**
Egy ládat reprezentál a játékban. A láda tologatható, bizonyos mezőkre kerülve különlegesen viselkedhet.
- **Ősosztályok**
Moveable → Crate
- **Attribútumok**
 - **bool isOnTarget:** A láda tudja, hogy éppen célmezőn tartózkodik-e.
 - **bool couldMove:** A láda könnyen eltolható-e.
- **Metódusok**
 - **void kill():** Az láda elveszik, és nem vesz részt a további játékban. Például lyukba esés esetén hívható függvény.
 - **void onSwitch(SwitchField s):** Ennek a függvénynek a meghívásakor a Láda meghívja az s objektum turnOn() függvényét, majd visszatér.
 - **void stepOnTarget(Worker w):** Ha ezt a függvényt meghívják, akkor a láda ad egy pontot a játékosnak (w.incrementPoint()), és beállítja saját isOnTarget attribútumát igazra.
 - **void stepOffTarget(Worker w):** Ha ezt a függvényt meghívják, akkor a láda elvesz egy pontot a játé kostól (w.decrementPoint()), és beállítja saját isOnTarget attribútumát hamisra.
 - **void setCouldMove(bool b):** A függvény beállítja a couldMove attribútum értékét.
 - **bool couldNotMoveOrIsOnTarget():** A láda ennek a függvénynek a hívására válaszolja, hogy felőle vége legyen-e a játéknak. Tehát, ha a láda nehezen mozgatható helyre került, vagy célmezőn van.

3.3.3 Direction

- **Felelősség**

Enumeration osztály, amely a négy irány közül vehet fel egy értéket.

- **Ősosztályok**

- **Literálok**

- **UP**: Fel.
- **DOWN**: Le.
- **RIGHT**: Jobbra.
- **LEFT**: Balra.

- **Metódusok**

- **Direction opposite()**: Visszaadja a szemközti irányt. Lehetséges, hogy ebben a modellben kihagyható lesz, de személyes tapasztalat szerint hasonló feladatok esetén, ez egy hasznos metódus.

3.3.4 Hole

- **Felelősség**

A lyukakat reprezentálja. Amennyiben rákerül egy objektum, az azonnal meghal.

- **Ősosztályok**

AbstractField→SimpleField→ Hole

- **Attribútumok**

- **Metódusok**

- **bool accept(Direction toward, Moveable m, Worker w)**: Hasonló a működése a közvetlen őséhez képest, azonban nem fog továbbhívni egy újabb acceptet, hanem rögtön elfogadja az érkező p paramétert, majd meg is öli a kill() függvényével. Ezután visszatér egy igaz értékkel.

3.3.5 Moveable

- **Felelősség**

Egy, a pályán mozgó/mozgatható dolgot reprezentál. Igaz rá, hogy minden pontban egy mezőn van rajta, ahol rajta kívül nem lehet senki. Absztrakt alaposztály, konkrét példányai lehetnek a láda (Crate) és a munkás (Worker).

- **Ősosztályok**

- **Attribútumok**

- **AbstractField myField**: Az objektum ismeri a mezőt, amin tartózkodik.

- **Metódusok**
 - **void kill()**: Az objektum meghal, és nem vesz részt a további játékban; a referenciáját kitörli az elfoglalt mezőről. Például lyukba esés esetén hívható függvény.
 - **bool squeez()**: Az objektum összenyomódásra kényszerül. Amennyiben az összenyomás végbejött, elvárjuk, hogy az objektum meghívja saját kill() függvényét. Alapértelmezetten nem csinál semmit, csak visszatér. További elvárás, hogy akkor térjen vissza igazzal, ha a kill() meg volt hívva.
 - **void move(AbstractField here)**: Az objektum másik mezőre került. Ekkor ez a függvény állítja be az új mező értékét, és az esetleges egyéb adminisztrációkat végzi. (Alapvetően egy setter függvény a myField-re.)
 - **abstract void onSwitch(SwitchField s)**: Ha az objektum egy kapcsoló mezőre kerül, akkor a kapcsoló meghívja rajta ezt a függvényt, és átadja az önnön referenciáját, hogy az objektum saját belátása szerint kapcsolhassa a kapcsolót. Absztrakt függvény, felüldefiniálásra.
 - **void stepOnTarget(Worker w)**: Ha egy mozgó objektum rákerül egy célmezőre, akkor az meghívhatja rajta ezt a függvényt, hogy az objektum saját belátása szerint esetleg pontot adhasson a paraméterként átadott játékosnak. A paraméterként átadott játékos jellemzően az lehet, aki az egész tolást elindította.
 - **void stepOffTarget(Worker w)**: A stepOnTarget ellenkezője. Ha egy mozgó objektum lekerül egy célmezőről, akkor az meghívhatja rajta ezt a függvényt, hogy az objektum saját belátása szerint esetleg pontot vonjon le a paraméterként átadott játékosról. A paraméterként átadott játékos jellemzően az lehet, aki az egész tolást elindította.
 - **void setCouldMove(bool b)**: Ha a Moveable olyan helyre jutott, ahonnan a mozgatása nehézkes, akkor ezzel a setter függvénytelével lehet ezt számára jelezni. Jelentősége a játék vége detektálásánál lesz.

3.3.6 SimpleField

- **Felelősség**
Az egyszerű járható mezőt reprezentálja, nem rendelkezik semmi különlegesssel.
- **Ősosztályok**
AbstractField → SimpleField
- **Attribútumok**
 - **Moveable myMoveable**: A mező ismeri annak a referenciáját, aki rajta tartózkodik. Ez természetesen lehet NULL értékű is. (Egy lyuk esetében például nem is lehet más.)
- **Metódusok**
 - **bool moveWorker(Direction toward, Worker w)**: Ha egy munkás a saját mezőjéről szeretne ellépni, akkor meghívja rajta ezt a függvényt. Erre ez a mező kiválasztja a megfelelő szomszédját, és továbbadja neki a rajta lévő elem referenciáját annak reményében, hogy ő majd befogadja. A függvény hívója a visszatérési értékből fogja tudni, hogy az akció sikeres volt-e.

- **bool accept(Direction toward, Moveable m, Worker w):** Ez a függvény azt jelzi a mező számára, hogy szeretnék rálépni. Ekkor a mező, minden megróbál, hogy befogadhassa az érkező Moveable-t. Ezt úgy teszi, hogy...
 - a. amennyiben szabad, az elfogadásnak nincs akadálya.
 - b. Ha a mezőn már van valaki, akkor először megpróbálja tovább passzolni a rajta lévő elemet egy újabb accept függvényhívással, a megfelelő irányban lévő szomszédos mezőjének.
 - c. Ha ez nem sikerült, akkor megkéri a rajta lévő elemet az összenyomódásra.
 - d. Ha még mindig foglalt maradt a mező, akkor kénytelen visszatérni egy hamis értékkel, jelezve, hogy a befogadás nem volt sikeres.
- Ezt a függvényt a leszármazottak felüldefiniálhatják, és akár további speciális függvényeket is hívhatnak az elemen, például egy sikeres befogadás esetén.
- **void dropMoveable():** A myMoveable adattagot null-ra állítja.
 - **bool couldMoveOn(toward: Direction):** Az ōsosztály felüldefiniált függvénye a konkrét mezőre.
 - **void determineBlocked(d: Direction):** A szomszédos mezők couldMoveOn() függvényeinek segítségével ellenőrzi, hogy van-e legalább két nem szemközti szomszédja, ahonnan egy esetleg rajta lévő láda nem lenne közvetlenül tolható. Természetesen a d-vel ellentétes irányba vett szomszédot nem ellenőrzi.

3.3.7 SwitchableHole

- **Felelősség**

Egy kapcsolgatható lyukat reprezentál. Ha be van kapcsolva, Hole-ként viselkedik, ha nincs, akkor úgy, mint egy SimpleField.

- **Ősosztályok**

AbstractField → SimpleField → Hole → SwitchableHole

- **Attribútumok**

- **bool opened:** Eltárolja magáról, hogy a lyuk nyitva van-e.

- **Metódusok**

- **bool accept(Direction toward, Moveable m, Worker w):** Amennyiben az opened értéke igaz, viselkedése megegyezik a Hole osztály azonos nevű függvényével. Amennyiben az opened hamis, viselkedése megegyezik a SimpleField azonos nevű függvényével.
- **void open():** Beállítja az opened értékét igazra, és ha valaki tartózkodna a mezőn, akkor azt megöli.
- **void close():** Beállítja az opened értékét hamisra.

3.3.8 SwitchField

- **Felelősség**

Kapcsolót tartalmazó mezőt reprezentál. Amennyiben láda van a mezőn a hozzá tartozó lyukat nyitva, minden egyéb esetben zárva tartja

- **Ősosztályok**

AbstractField→ SimpleField→ SwitchField

- **Attribútumok**

- **SwitchableHole sHole:** Az a kapcsolható lyuk, akit a kapcsoló irányít.

- **Metódusok**

- **bool accept(Direction toward, Moveable m, Worker w):** Megegyezik az ősosztály azonos nevű függvényével. Csupán annyiban egészíti ki azt, hogy amennyiben a befogadás sikeres, meghívja a ráérkező Moveable-nek az onSwitch függvényét, saját magát átadva paraméterként. Ennek értelme, hogy a ráérkező Moveable eldöntheti, hogy a kapcsolót kívánja-e bekapcsolni.
- **void turnOn():** Meghívja az sHole open() függvényét.
- **void turnOff():** Meghívja az sHole close() függvényét.

3.3.9 TargetField

- **Felelősség**

A célmezőket reprezentálja. Legtöbb funkciójukban egy egyszerű járható mezővel egyeznek meg, ám a pontozást elősegítő rendeltetésük is van.

- **Ősosztályok**

AbstractField→ SimpleField→ TargetField

- **Attribútumok**

- **Metódusok**

- **bool accept(Direction toward, Moveable m, Worker w):** Az ősosztály funkcióján felül, amennyiben a befogadás sikeres, meghívja a kapott Moveable paraméter StepOnTarget(w) függvényét, és az eltávozott Moveable típusú objektum stepOffTarget(w) függvényét.

3.3.10 Wall

Felelősség

Egy fal vagy oszlop miatt járhatatlan mezőt reprezentál.

Ősosztályok

AbstractField→ Wall

Attribútumok

Metódusok

- **bool accept(Direction toward, Moveable m, Worker w):** Konstans hamis értéket ad vissza, mert semmit nem hajlandó elfogadni.

- **bool couldMoveOn(toward: Direction):** Az ōsosztály felüldefiniált függvénye a fal típusú mezőre. Konstans hamis értékkel tér vissza.

3.3.11 Worker

- **Felelősség**

Egy játékos által irányított munkást reprezentál a játékban. A munkáson hívható egy léptető függvény, ami elindítja a modellezett folyamatokat.

- **Ősosztályok**

Moveable → Worker

- **Attribútumok**

- **bool alive:** A munkásról feljegyezzük, hogy éppen játékban van-e még.
- **int point:** A munkás saját változójában jegyzi, hogy eddig hány pontot gyűjtött össze a játék során.

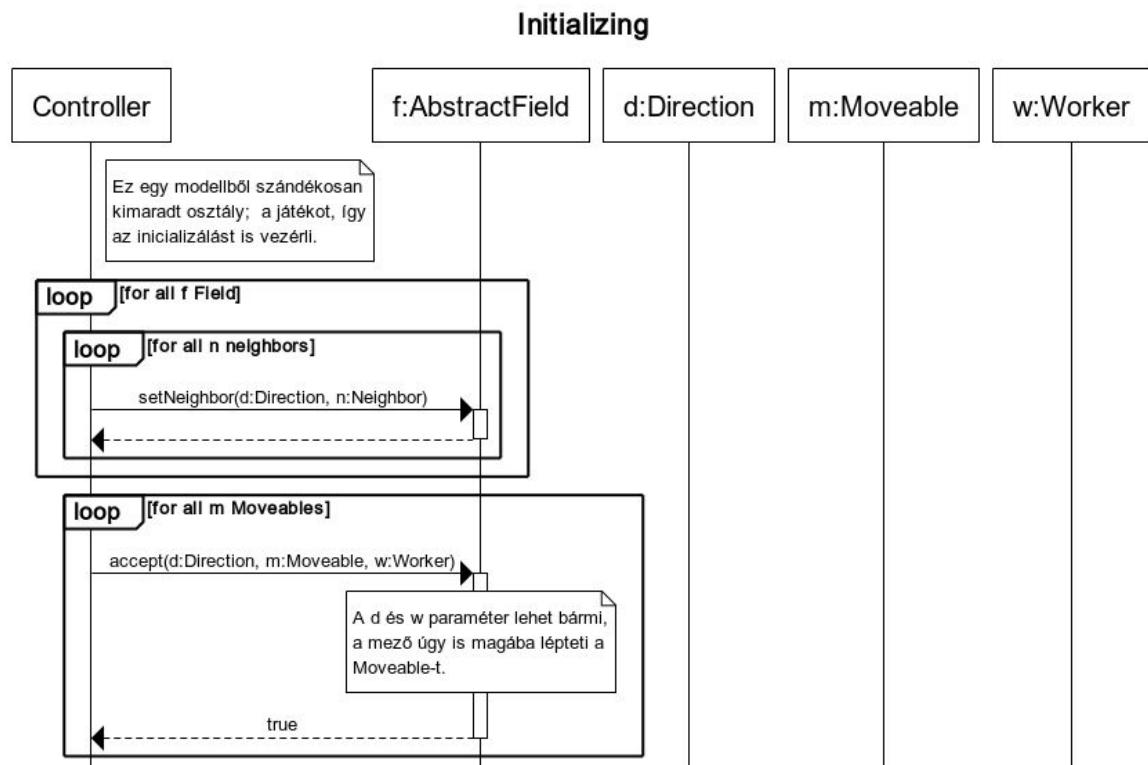
- **Metódusok**

- **void kill():** A munkás meghal, és nem vesz részt a további játékban. Például lyukba esés, vagy összenyomódás esetén hívható függvény.
- **void squeez():** Az ōsosztály felüldefiniált függvénye. Ennek hatására meghívja a kill függvényt saját magán.
- **bool step(Direction toward):** Jellemzően valamely külső vezérlő osztály hívja meg a játékosnak ezt a függvényét, amely közli vele, hogy a játékos szeretné léptetni valamely irányba. Visszatérési értéke mutatja a vezérlőnek, hogy a lépés sikeres volt-e.
- **void incrementPoint():** Pontot ad a játékosnak.
- **void decrementPoint():** Pontot von le a játékostól.
- **void onSwitch(SwitchField s):** Kikapcsolja a paraméterként kapott kapcsolót.

3.4 Szekvencia diagramok

A szekvenciadiagramokon lévő szintaktikai hibák, mint a lifeline-ok vonalainak folytonossága és a return hívások nyilai a modellező eszköz hiányosságai miatt vannak.

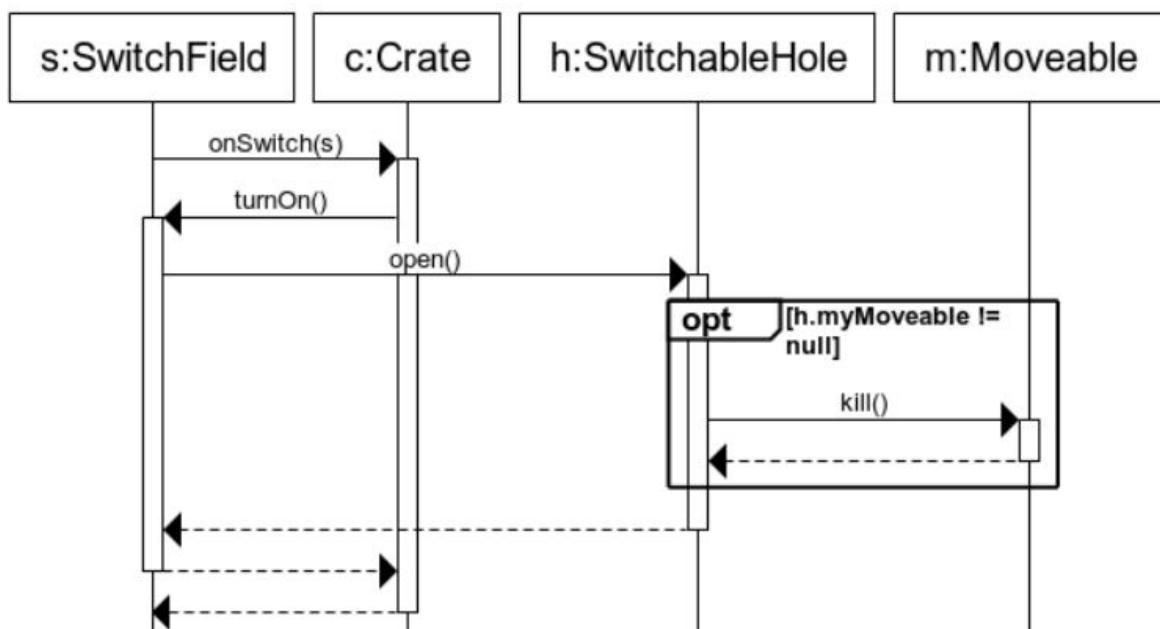
3.4.1 Initializing



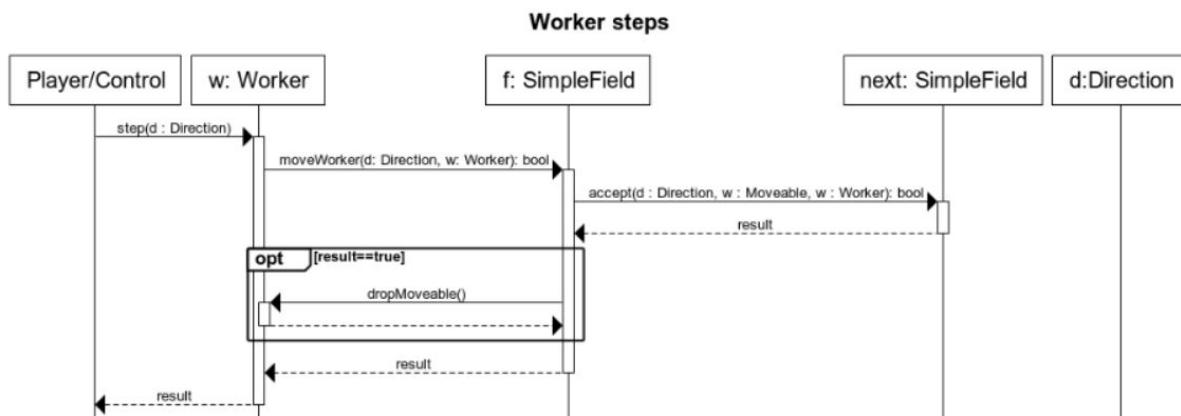
3.4.2 Crate on Switch

Megjegyzés: A h.myMoveable és az m ugyanazt az objektumot jelölik, tehát a h ismeri m-et kezdettől fogva.

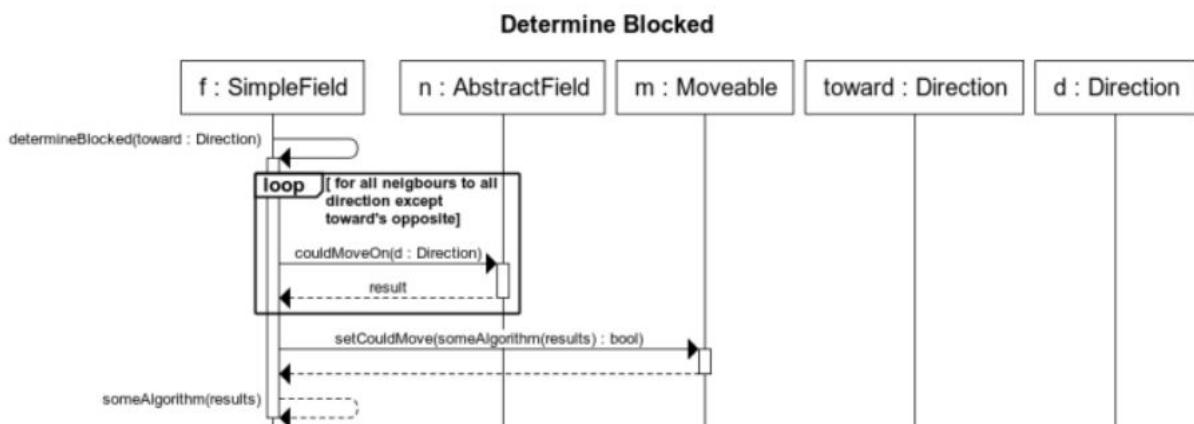
Crate on Switch



3.4.3 Worker steps

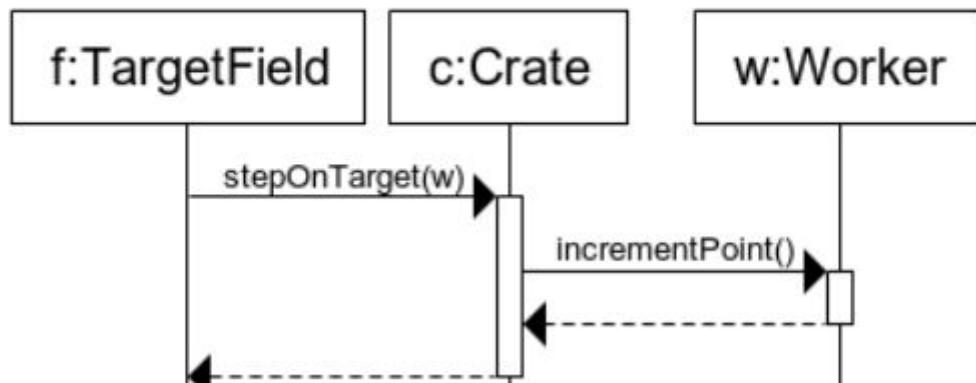


3.4.4 Determine Blocked



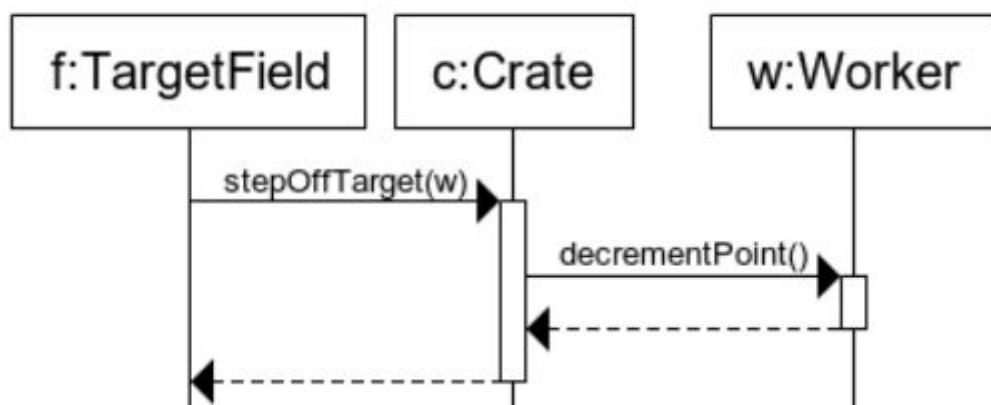
3.4.5 Crate steps on TargetField

Crate steps on TargetField



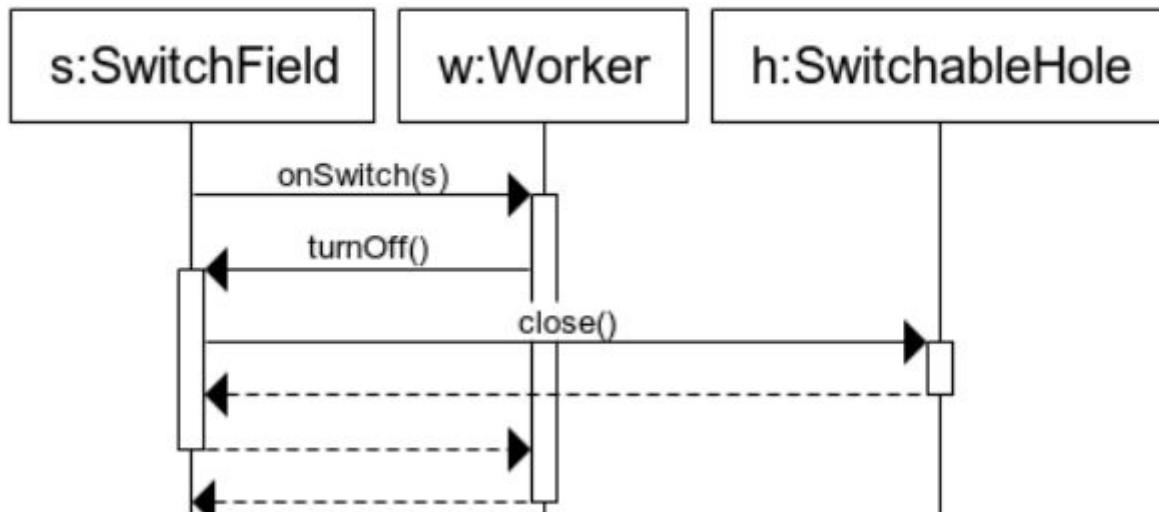
3.4.6 Crate steps off TargetField

Crate steps off TargetField



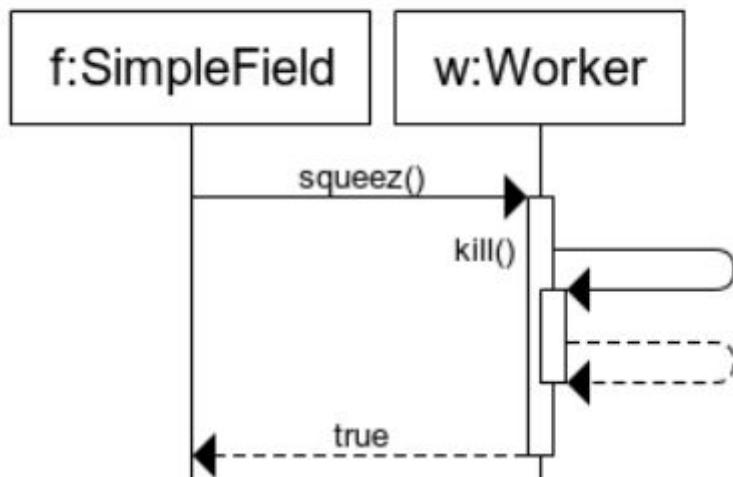
3.4.7 Worker on Switch

Worker on Switch



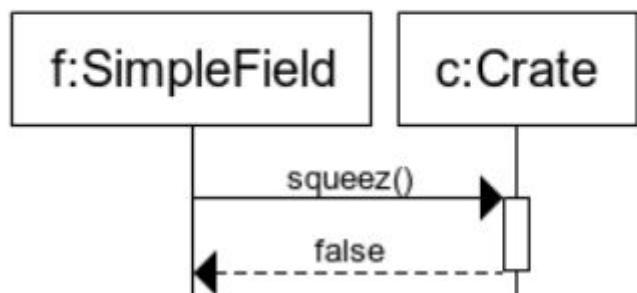
3.4.8 Worker squeezes

Worker squeezes



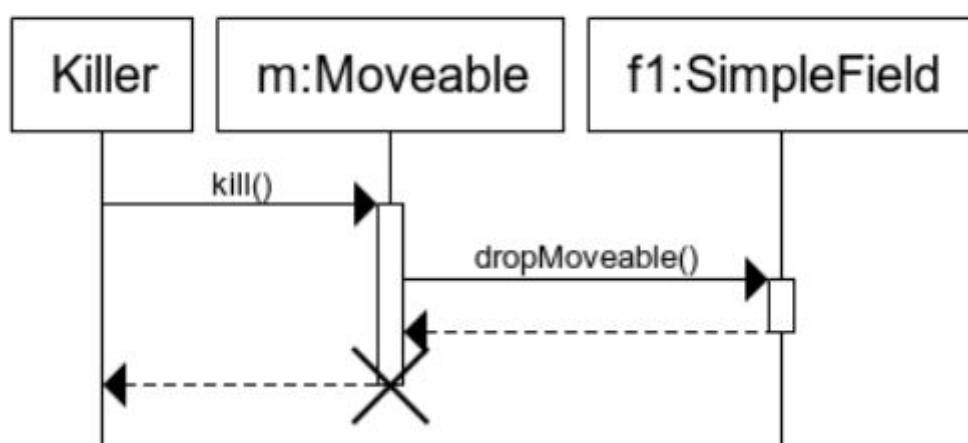
3.4.9 Crate squeezes

Crate squeezes



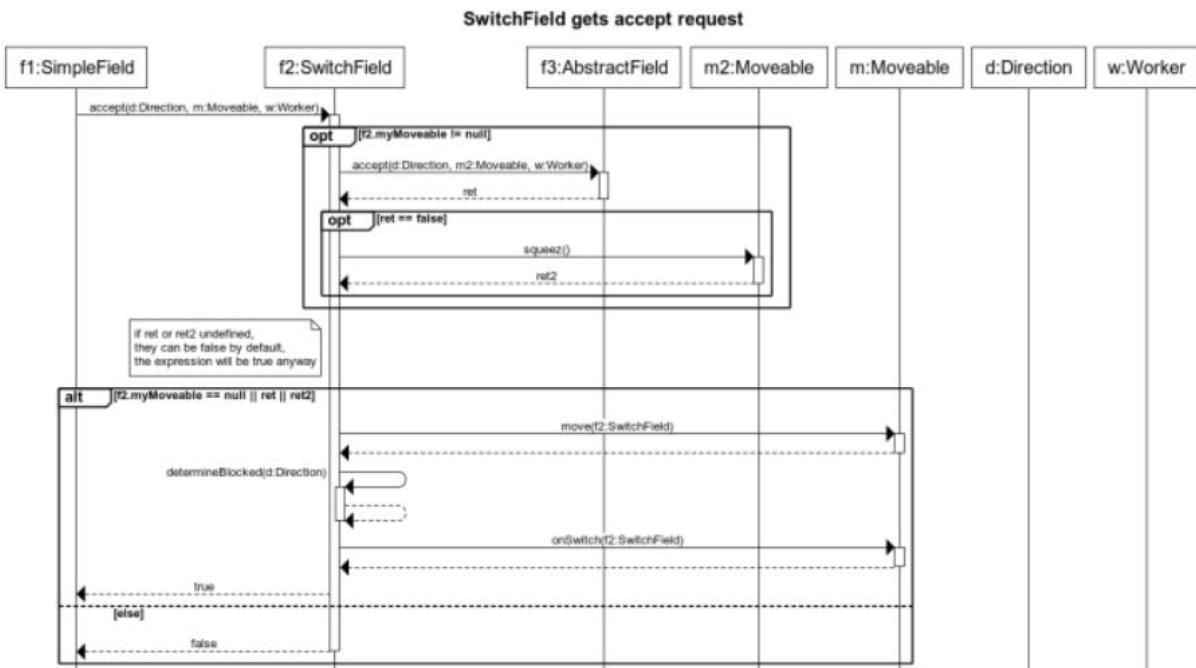
3.4.10 Moveable dies

Moveable dies



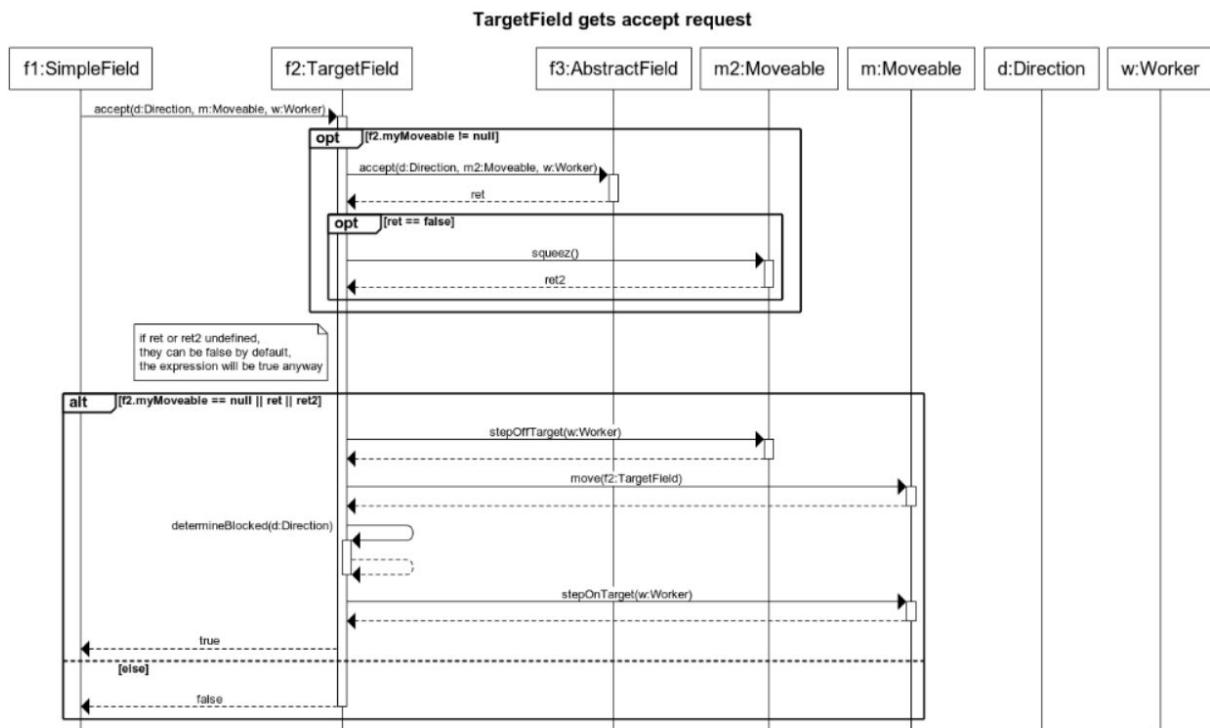
3.4.11 SimpleField gets accept request

Megjegyzés: Az f2.myMoveable és az m2 ugyanazt az objektumot jelölik, tehát a f2 ismeri m2-et kezdettől fogva. f3 pedig f2 megfelelő irányú szomszédja, tehát azt is ismeri.



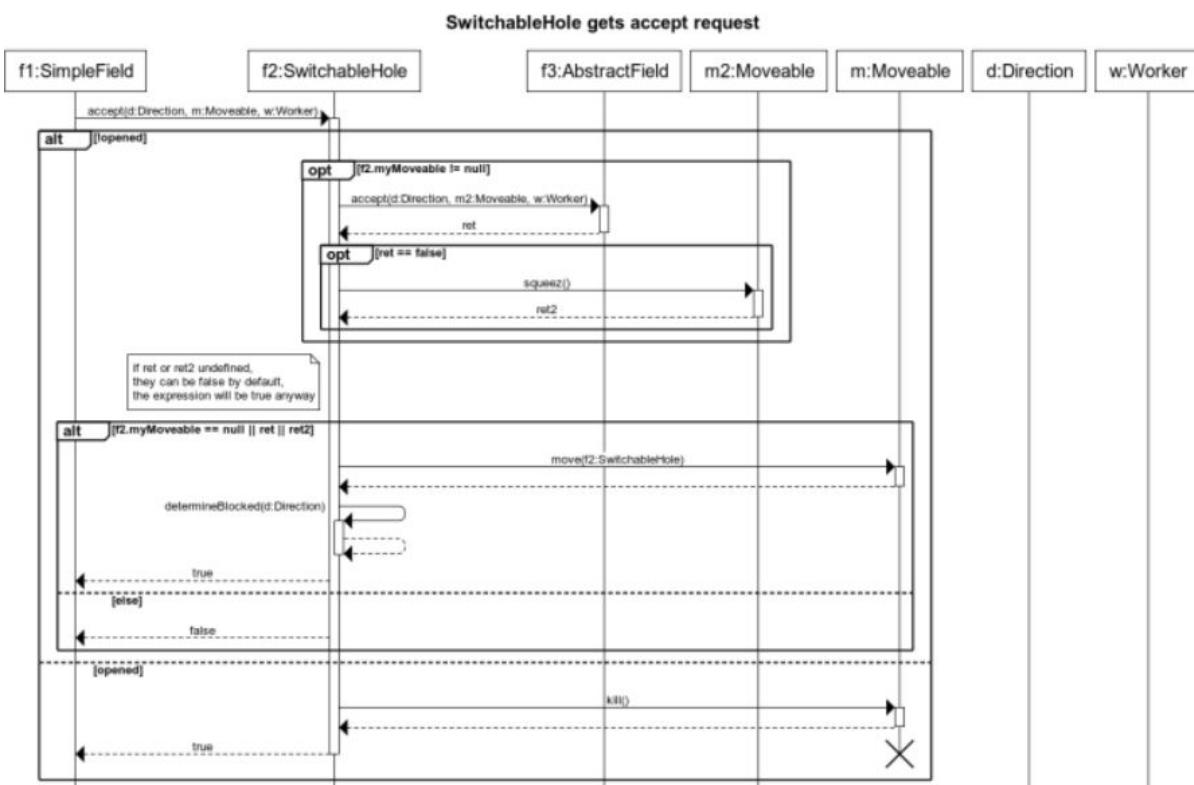
3.4.12 TargetField gets accept request

Megjegyzés: Az f2.myMoveable és az m2 ugyanazt az objektumot jelölik, tehát a f2 ismeri m2-et kezdettől fogva. f3 pedig f2 megfelelő irányú szomszédja, tehát azt is ismeri.



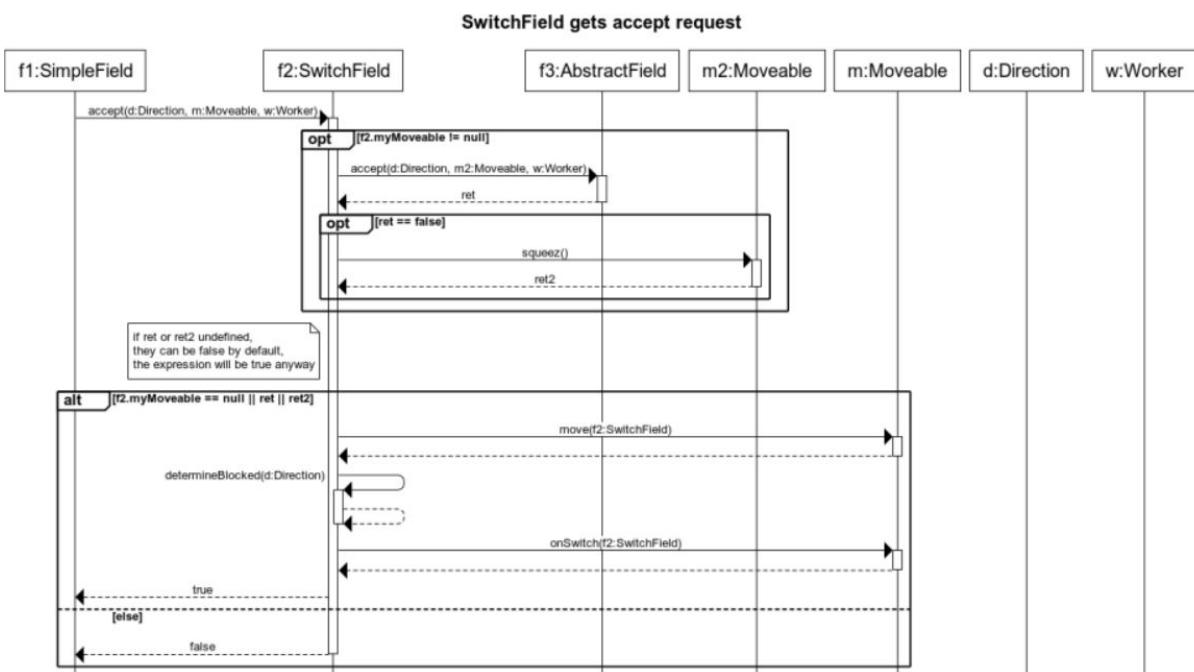
3.4.13 SwitchableHole gets accept request

Megjegyzés: Az f2.myMoveable és az m2 ugyanazt az objektumot jelölik, tehát a f2 ismeri m2-et kezdettől fogva. f3 pedig f2 megfelelő irányú szomszédja, tehát azt is ismeri.



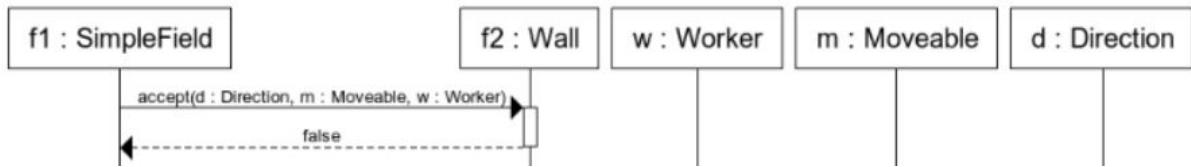
3.4.14 SwitchField gets accept request

Megjegyzés: Az f2.myMoveable és az m2 ugyanazt az objektumot jelölik, tehát a f2 ismeri m2-et kezdettől fogva. f3 pedig f2 megfelelő irányú szomszédja, tehát azt is ismeri.



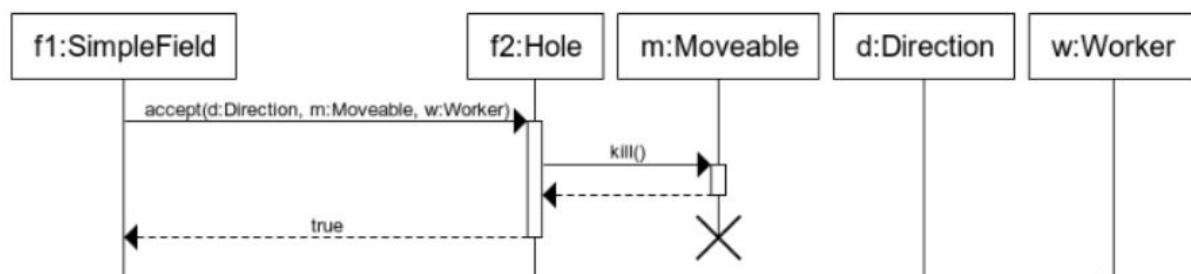
3.4.15 Wall gets accept request

Wall gets accept request



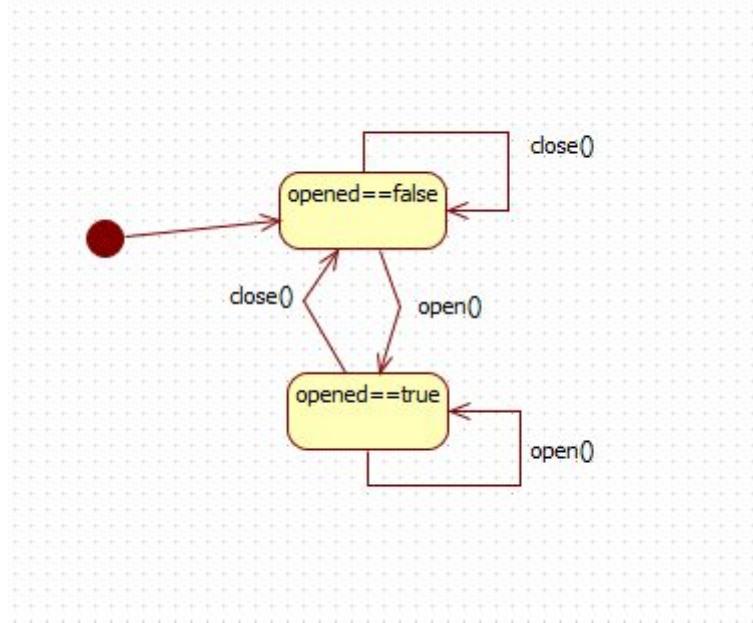
3.4.16 Hole accepts Moveable

Hole accepts Moveable



3.5 State-chartok

SwitchableHole



3.6 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.02.18. 17:00	2 óra	Zalavári	Analízis modell nagyvonalú kidolgozása
2018.02.19. 19:00	1 óra	Zalavári	Analízis modell nagyvonalú kidolgozása
2018.02.20 8:00	1 óra	Csapat	Melegszendvics sütés és értekezlet: A különböző analízismodellek összehasonlítása, ötleteletés
2018.02.21 12:00	2 óra	Csapat	A konzulens meghallgatása, releváns kérdések egyeztetése
2018.02.22. 15:00	0,5 óra	Zalavári	Modellvariációk vázlatos készítése.
2018.02.22. 17:30	45 perc	Csapat	Értekezlet: Modell kiválasztása, tevékenységek felosztása Döntés: 3.1 Benkő, 3.2 és 3.3 Zalavári, Schulcz ellenőrzi, 3.4 Takács, Schulcz ellenőrzi, 3.5 Schulcz. Határidő: 02.25.
2018.02.22. 22:30	30 perc	Zalavári	3.3
2018.02.23. 17:00	1 óra	Zalavári	3.3
2018.02.24. 12:30	2 óra	Zalavári	3.2 és 3.3
2018.02.24. 14:00	2 óra	Takács	3.4
2018.02.24. 17:00	1 óra	Takács	3.4
2018.02.24. 18:00	1 óra	Zalavári, Takács	Konzultáció
2018.02.24. 19:00	2 óra	Takács	3.4
2018.02.24. 21:00	0.5 óra	Schulcz, Zalavári	Modellbeli elemek szemantikai ellenőrzése, R22-es követelmény modellbe építése.
2018.02.25. 9:00	1 óra	Zalavári	3.1 átnézése.

			Módosítások elvégzése a 3.2, 3.3 részekben.
2018.02.25. 13:30	2 óra 45 perc	Schulcz	Szekvenciadiagramok rajzolása
2018.02.25. 18:30	5 óra 40 perc	Schulcz	Szekvenciadiagramok rajzolása
2018.02.25. 21:00	3 óra	Zalavári	Szekvenciadiagramok rajzolása
2018.02.25. 21:00	3 óra	Takács	Szekvenciadiagramok rajzolása, Állapotgép
2018.02.25. 22:00	30 perc	Benkő	Objektum katalógus elkészítése
2018.02.26. 0:15	20 perc	Schulcz	Pontosítások az osztályleírásokban
2018.02.26. 08:30	30 perc	Csapat	Értekezlet Az eddigi munka áttekintése, a pontosítások egységesítése, feladatok kiosztása a leadásig: Schulcz: osztálydiagram és osztályleírások pontosítása, kiegészítése Zalavári: szekvenciadiagramok pontosítása, kiegészítése Benkő: sajtó alá rendezés, nyomtatás
2018.02.26 12:15	1 óra	Zalavári	Szekvenciadiagram mokon végső simítások, többiek munkájának ellenőrzése
2018.02.26. 12:45	30 perc	Schulcz	Diagramok pontosítása, apróbb javítások a dokumentumban
2018.02.26. 12:30	1 óra 30 perc	Benkő	sajtó alá rendezés, nyomtatás

4. Analízis modell kidolgozása

4.1 Objektum katalógus

4.1.1 Láda

A raktár életének egykedvű szemlélői. Biztosítják a saját mozgathatóságukat, de önhatalmúlag nem mozognak. Emellett a pontozási rendszerben van közreműködő szerepük.

4.1.2 Mező

Rajtuk tartózkodnak a munkások és a lánkok. Az összes mező halmaza alkotja a tulajdonképpeni pályát. Felelősségeik a szomszédos mezők irányonkénti ismerete (pályareprezentáció), ill. a rajtuk található dolgok mozgatásának biztosítása.

4.1.3 Munkás

A játék aktív mozgatói, formálói. Ők a belépési pontok a játékosok számára a raktár világába, a felhasználók rajtuk keresztül tudnak a játon belül cselekedni. Ők indítják a raktárban található lánkok mozgatását a mezőkkel kommunikálva. Az egyes játékosok pillanatnyi pontjait is ők tartják számon.

4.1.4 Lyuk

A belekerülő (rá lépő) tárgyakat vagy embereket megsemmisíti. A “megsemmisítéshez” tartozó járulékos feladatak tartoznak felelősségeik közé.

4.1.5 Célmező

A mező összes sajátosságával és felelősséggel rendelkezik. Emellett a pontszámításban játszik közre.

4.1.6 Kapcsoló

A mező összes sajátosságával és felelősséggel rendelkezik. Annak függvényében, hogy mi áll rajta, utasításokat ad bizonyos speciális mezőknek (kapcsolható lyukak).

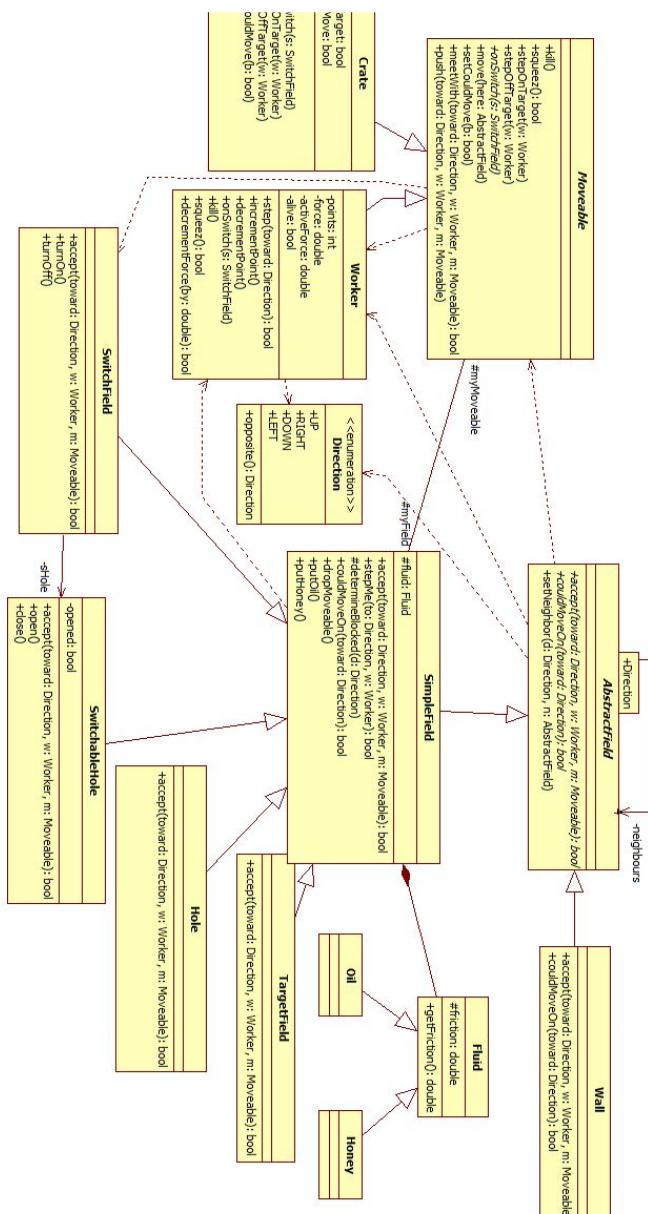
4.1.7 Kapcsolható lyuk

A hozzá tartozó kapcsoló utasításainak megfelelően a lyuk vagy (xor) a normális mező viselkedésének megfelelően működik. A kettő közötti döntés a legfőbb feladata.

4.1.8 Fal

A mezők között helyenként előforduló, átjárhatatlan elem. Felelőssége, hogy a rálépni kívánó dolgoknak ezt (ti. a rálépést) ne engedje meg, valamint, hogy ez a meg-nem-engedés hibamentes legyen.

4.2 Statikus struktúra diagramok



4.3 Osztályok leírása

Az örökölt metódusokat (és attribútumokat) csak indokolt esetben (jellemzően felüldefiniálás esetén) tüntettem fel.

Mivel a modellünkben nincsenek interfészek, ezért azon pontokat mindenhol kihagytam.

Az attribútumok általában privát vagy védett láthatóságúak, de egy-két fontosabbat a könnyebb érthetőség miatt feltüntettem a leírásnál

A turnOff() függvényt nem hívjuk meg kívülről, de a szimmetria és bővíthetőség érdekében publikus függvény maradt.

A determineBlocked(..) metódus szintén nem hívódik kívülről, így bár nem publikus, (tehát halálfejes hiba...) de más mezőkkel kommunikál, és más publikus függvényeket csak ő hívna meg. (és ha ezeket nem tüntetjük fel, akkor meg az halálfejes hiba..)

4.3.1 AbstractField

- **Felelősség**

Egy mezőt reprezentál a játékban. Absztrakt alaposztály, konkrét példányai lehet például egy egyszerű járható mező, oszlop, lyuk, stb.

- **Ősosztályok**

- **Attribútumok**

- **AbstractField neighbours:** A mező ismeri a szomszédait.

- **Metódusok**

- **abstract bool accept(Direction toward, Worker w, Moveable m):** Ez a függvény azt jelzi a mező számára, hogy szeretnének rálépni. Ekkor a mező, ha szabad, vagy azzá tudja tenni magát, befogadja a paraméterül kapott Moveable-t, és visszatér igazzal. Ellenkező esetben nem csinál semmit, és hamis értékkel tér vissza.
- **abstract bool couldMoveOn(Direction toward):** Megkérdezi a mezőtől, hogy ha egy tolás következne be a paraméterként átadott irányba, akkor ez lehetséges lenne-e. A pontos algoritmusa még nem ismert, de csak a játék végének detektálása miatt szükséges függvény.
- **void setNeighbor(Direction d, AbstractField n):** Az adott irányba beállítja, hogy ki a mező szomszédja.

4.3.2 Crate

- **Felelősség**

Egy lánát reprezentál a játékban. A láda tologatható, bizonyos mezőkre kerülve különlegesen viselkedhet.

- **Ősosztályok**

Moveable → Crate

- **Attribútumok**

- **bool isOnTarget:** A láda tudja, hogy éppen célmezőn tartózkodik-e.
- **bool couldMove:** A láda eltolható-e.

- **Metódusok**

- **void kill():** A láda elveszik, és nem vesz részt a további játékban. Például lyukba esés esetén hívható függvény.
- **void onSwitch(SwitchField s):** Ennek a függvénynek a meghívásakor a Láda meghívja az s objektum turnOn() függvényét, majd visszatér.
- **void stepOnTarget(Worker w):** Ha ezt a függvényt meghívják, akkor a láda ad egy pontot a játékosnak (w.incrementPoint()), és beállítja saját isOnTarget attribútumát igazra.

- **void stepOffTarget(Worker w):** Ha ezt a függvényt meghívják, akkor a láda elvesz egy pontot a játékostól (w.decrementPoint()), és beállítja saját isOnTarget attribútumát hamisra.
- **void setCouldMove(bool b):** A függvény beállítja a couldMove attribútum értékét.

4.3.3 Direction

- **Felelősség**

Enumeration osztály, amely a négy irány közül vehet fel egy értéket.

- **Ősosztályok**

- **Literálok**

- **UP:** Fel.
- **DOWN:** Le.
- **RIGHT:** Jobbra.
- **LEFT:** Balra.

- **Metódusok**

- **Direction opposite():** Visszaadja a szemközti irányt. Lehetséges, hogy ebben a modellben kihagyható lesz, de személyes tapasztalat szerint hasonló feladatok esetén, ez egy hasznos metódus.

4.3.4 Hole

- **Felelősség**

A lyukakat reprezentálja. Amennyiben rákerül egy objektum, az azonnal meghal.

- **Ősosztályok**

AbstractField→SimpleField → Hole

- **Attribútumok**

- **Metódusok**

- **bool accept(Direction toward, Worker w, Moveable m):** Hasonló a működése a közvetlen őséhez képest, azonban rögtön elfogadja az érkező m paramétert, majd meg is öli a kill() függvényével. Ezután visszatér egy igaz értékkel.

4.3.5 Moveable

- **Felelősség**

Egy, a pályán mozgó/mozgatható dolgot reprezentál. Igaz rá, hogy minden pontosan egy mezőn van rajta, ahol rajta kívül nem lehet senki. Absztrakt alaposztály, konkrét példányai lehetnek a láda (Crate) és a munkás (Worker).

- **Ősosztályok**
- **Attribútumok**
 - **AbstractField myField:** Az objektum ismeri a mezőt, amin tartózkodik.
- **Metódusok**
 - **void kill():** Az objektum meghal, és nem vesz részt a további játékban; a referenciáját kitörli az elfoglalt mezőről. Például lyukba esés esetén hívható függvény.
 - **bool squeez():** Az objektum összenyomódásra kényszerül. Amennyiben az összenyomás végbement, elvárjuk, hogy az objektum meghívja saját kill() függvényét. Alapértelmezetten nem csinál semmit, csak visszatér. További elvárás, hogy akkor térjen vissza igazzal, ha a kill() meg volt hívva.
 - **void move(AbstractField here):** Az objektum másik mezőre került. Ekkor ez a függvény állítja be az új mező értékét, miután az előző mezővel eldobatta magát a dropMoveable()-lel. (Alapvetően egy setter függvény a myField-re.)
 - **abstract void onSwitch(SwitchField s):** Ha az objektum egy kapcsoló mezőre kerül, akkor a kapcsoló meghívja rajta ezt a függvényt, és átadja az önnön referenciáját, hogy az objektum saját belátása szerint kapcsolhassa a kapcsolót. Alapvetően nem csinál semmit, csak ha a leszármazott felüldefiniálja.
 - **void stepOnTarget(Worker w):** Ha egy mozgó objektum rákerül egy célmezőre, akkor az meghívhatja rajta ezt a függvényt, hogy az objektum saját belátása szerint esetleg pontot adhasson a paraméterként átadott játékosnak. A paraméterként átadott játékos jellemzően az lehet, aki az egész tolást elindította.
 - **void stepOffTarget(Worker w):** A stepOnTarget ellenkezője. Ha egy mozgó objektum lekerül egy célmezőről, akkor az meghívhatja rajta ezt a függvényt, hogy az objektum saját belátása szerint esetleg pontot vonjon le a paraméterként átadott játé kostól. A paraméterként átadott játékos jellemzően az lehet, aki az egész tolást elindította.
 - **void setCouldMove(bool b):** Ha a Moveable olyan helyre jutott, ahonnan a mozgatása nehézkes, akkor ezzel a setter függvénytelhet ezt számára jelezni. Jelentősége a játék vége detektálásánál lesz.
 - **bool meetWith(Direction toward, Worker w, Moveable m):** Akkor hívandó, ha az m Moveable helyére jelen Moveable akar kerülni. A projekt jelen állapotában egy push()-t hív a paraméterül kapott Moveable-n. Akkor tér vissza igazzal, ha sikerült eltolnia m-et.
 - **bool push(Direction toward, Worker w, Moveable m):** Akkor hívandó, ha a Moveable-t meg akarjuk tolni. Hatására a Moveable szól a mezőjének, hogy a paraméterül kapott irányba szeretne menni. (A w paraméter a mozgást indító Worker, az m az a Moveable, aki minket éppen megtol.) Ha ez nem sikerül, akkor squeez()-t hív magán. Akkor tér vissza igazzal, ha valahogyan távozni tudott a mezőjéről.

4.3.6 SimpleField

- **Felelősség**

Az egyszerű járható mezőt reprezentálja, nem rendelkezik semmi különlegesssel.

- Ősosztályok

AbstractField → SimpleField

- Attribútumok

- **Moveable myMoveable:** A mező ismeri annak a referenciáját, aki rajta tartózkodik. Ez természetesen lehet NULL értékű is. (Egy lyuk esetében például nem is lehet más.)

- Metódusok

- **bool accept(Direction toward, Worker w, Moveable m):** Ez a függvény azt jelzi a mező számára, hogy szeretnék rálépni.
 - a. Amennyiben szabad, az elfogadásnak nincs akadálya.
 - b. Ha a mezőn már van valaki, akkor meghívja a belépést kérőn a meetWith() függvényét, hátha az odébb tudja tolni vagy valami mást tud csinálni vele.
 - c. Ha ez nem sikerült, akkor kénytelen visszatérni egy hamis értékkal, jelezve, hogy a befogadás nem volt sikeres.

Ezt a függvényt a leszármazottak felüldefiniálhatják, és akár további speciális függvényeket is hívhatnak az elemen, például egy sikeres befogadás esetén.

A befogadás sikerességét követnie kell egy move függvényhívásnak az érkező moveable-re.

Megjegyzés: Ez a függvény egy setterként is használható a myMoveable attribútumra, ha biztosak vagyunk benne, hogy az adott mező üres. Ekkor toward és w paraméter értéke lényegtelen, bizonyos esetekben lehet null is, de ha speciális mezőről van szó, jobb ha egy konkrét létrehozott dummy objektumot adunk oda neki.

- **void dropMoveable():** A myMoveable adattagot null-ra állítja.
- **bool couldMoveOn(toward: Direction):** Az Ősosztály felüldefiniált függvénye a konkrét mezőre.
- **void determineBlocked(d: Direction):** A szomszédos mezők couldMoveOn() függvényeinek segítségével ellenőrzi, hogy van-e legalább két nem szemközti szomszédja, ahonnan egy esetleg rajta lévő láda nem lenne közvetlenül tolható. Természetesen a d-vel ellentétes irányba vett szomszédot nem ellenőrzi.
- **bool stepMe(Direction d, Worker w):** Akkor hívja a mezőn a rajta álló Moveable, ha egy másik mezőre szeretne lépni. A d paraméter által meghatározott szomszéd mezőn accept()-et hív, továbbadva az ezen mezőn található Moveable-t, valamint a mozgatást kezdeményező w Worker-t.

4.3.7 SwitchableHole

- Felelősség

Egy kapcsolgatható lyukat reprezentál. Ha be van kapcsolva, Hole-ként viselkedik, ha nincs, akkor úgy, mint egy SimpleField.

- Ősosztályok

AbstractField → SimpleField → Hole → SwitchableHole

- Attribútumok

- **bool opened:** Eltárolja magáról, hogy a lyuk nyitva van-e.

- **Metódusok**

- **bool accept(Direction toward, Worker w, Moveable m)**: Amennyiben az opened értéke igaz, viselkedése megegyezik a Hole osztály azonos nevű függvényével. Amennyiben az opened hamis, viselkedése megegyezik a SimpleField azonos nevű függvényével.
- **void open()**: Beállítja az opened értékét igazra, és ha valaki tartózkodna a mezőn, akkor azt megöli.
- **void close()**: Beállítja az opened értékét hamisra.

4.3.8 SwitchField

- **Felelősség**

Kapcsolót tartalmazó mezőt reprezentál. Amennyiben láda van a mezőn a hozzá tartozó lyukat nyitva, minden egyéb esetben zárva tartja

- **Ősosztályok**

AbstractField → SimpleField → SwitchField

- **Attribútumok**

- **SwitchableHole sHole**: Az a kapcsolható lyuk, akit a kapcsoló irányít.

- **Metódusok**

- **bool accept(Direction toward, Worker w, Moveable m)**: Megegyezik az ősosztály azonos nevű függvényével. Csupán annyiban egészíti ki azt, hogy amennyiben a befogadás sikeres, meghívja a ráérkező Moveable-nek az onSwitch függvényét, saját magát átadva paraméterként. Ennek értelme, hogy a ráérkező Moveable eldöntheti, hogy a kapcsolót kívánja-e bekapsolni.
- **void turnOn()**: Meghívja az sHole open() függvényét.
- **void turnOff()**: Meghívja az sHole close() függvényét.
-

4.3.9 TargetField

- **Felelősség**

A célmezőket reprezentálja. Legtöbb funkciójukban egy egyszerű járható mezővel egyeznek meg, ám a pontozást elősegítő rendeltetésük is van.

- **Ősosztályok**

AbstractField → SimpleField → TargetField

- **Attribútumok**

- **Metódusok**

- **bool accept(Direction toward, Worker w, Moveable m)**: Az ősosztály funkcióján felül, amennyiben a befogadás sikeres, meghívja a kapott Moveable paraméter

StepOnTarget(w) függvényét, és az eltávozott Moveable típusú objektum stepOffTarget(w) függvényét.

4.3.10 Wall

Felelősség

Egy fal vagy oszlop miatt járhatatlan mezőt reprezentál.

Ősosztályok

AbstractField → Wall

Attribútumok

Metódusok

- **bool accept(Direction toward, Worker w, Moveable m,):** Konstans hamis értéket ad vissza, mert semmit nem hajlandó elfogadni.
- **bool couldMoveOn(toward: Direction):** Az ősosztály felüldefiniált függvénye a fal típusú mezőre. Konstans hamis értékkel tér vissza.

4.3.11 Worker

● Felelősség

Egy játékos által irányított munkást reprezentál a játékban. A munkáson hívható egy léptető függvény, ami elindítja a modellezett folyamatokat.

● Ősosztályok

Moveable → Worker

● Attribútumok

- **bool alive:** A munkásról feljegyezzük, hogy éppen játékban van-e még.
- **int points:** A munkás saját változójában jegyzi, hogy eddig hány pontot gyűjtött össze a játék során.

● Metódusok

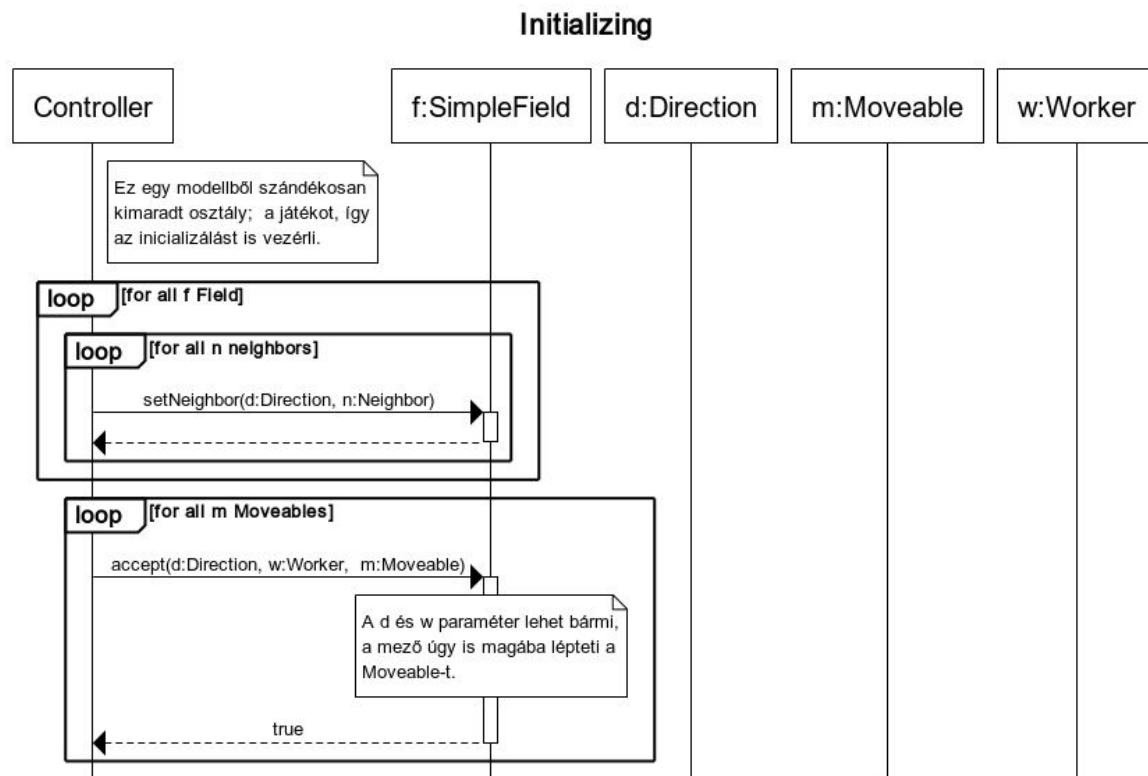
- **void kill():** A munkás meghal, és nem vesz részt a további játékban. Például lyukba esés, vagy összenyomódás esetén hívható függvény.
- **bool squeez():** Az ősosztály felüldefiniált függvénye. Ennek hatására meghívja a kill függvényt saját magán.
- **bool step(Direction toward):** Jellemzően valamely külső vezérlő osztály hívja meg a játékosnak ezt a függvényét, amely közli vele, hogy a játékos szeretné léptetni valamely irányba. Visszatérési értéke mutatja a vezérlőnek, hogy a lépés sikeres volt-e.
- **void incrementPoint():** Pontot ad a játékosnak.
- **void decrementPoint():** Pontot von le a játékosról.
- **void onSwitch(SwitchField s):** Kikapcsolja a paraméterként kapott kapesolót.

4.4 Szekvencia diagramok

A szekvenciadiagramokon lévő szintaktikai hibák, mint a lifeline-ok vonalainak folytonossága és a return hívások nyilai a modellező eszköz hiányosságai miatt vannak.

4.4.1 Initializing

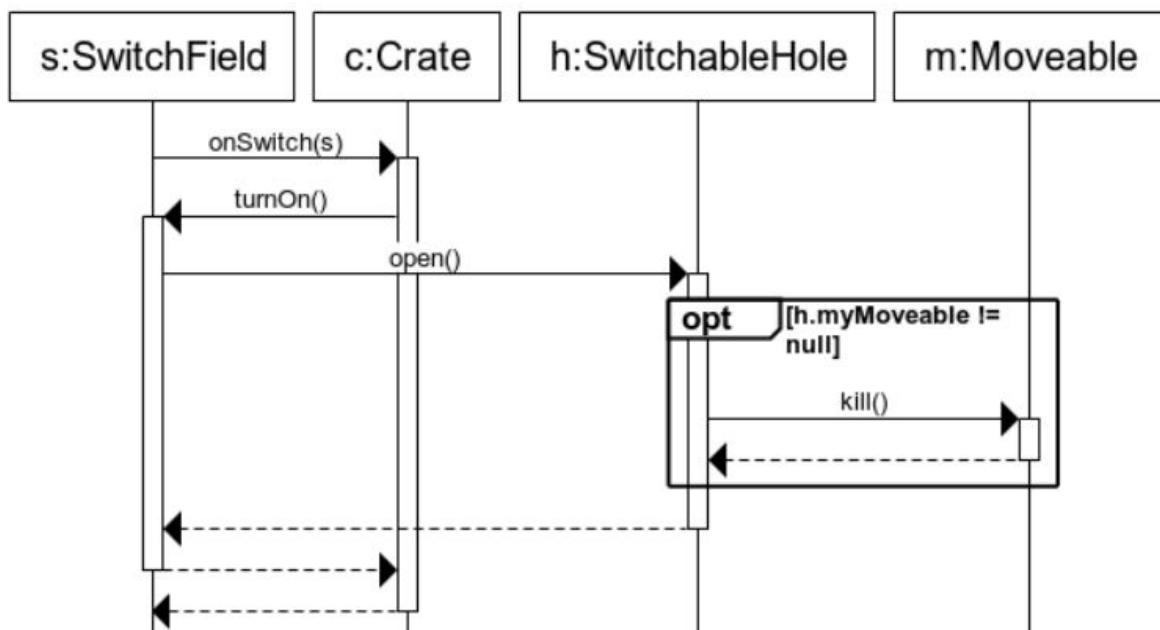
AbstractField helyett SimpleField



4.4.2 Crate on Switch

Megjegyzés: A h.myMoveable és az m ugyanazt az objektumot jelölik, tehát a h ismeri m-et kezdettől fogva.

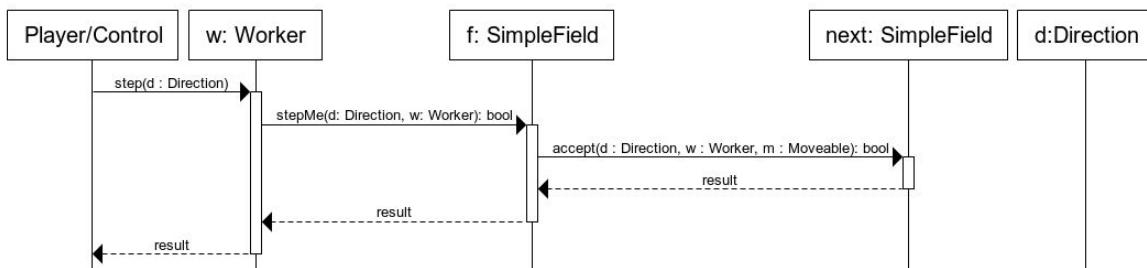
Create on Switch



4.4.3 Worker steps

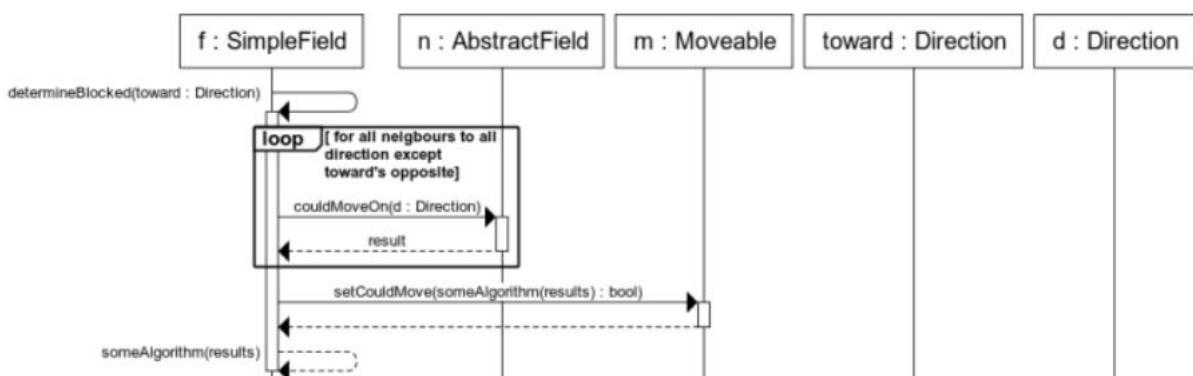
DropMoveable nem kill ide

Worker steps



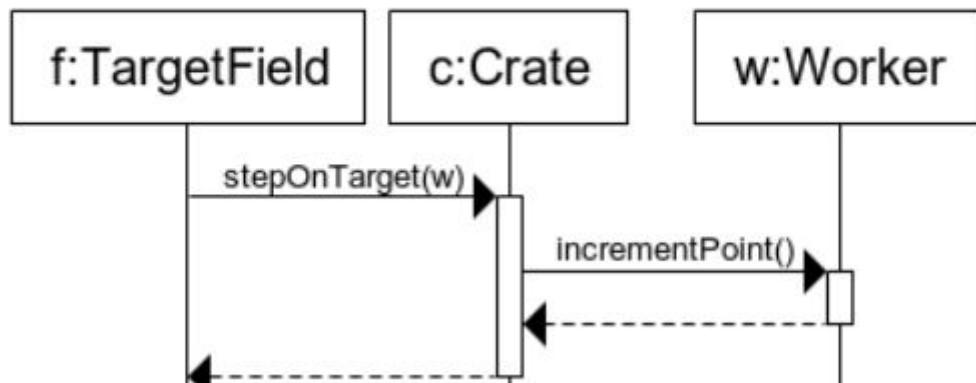
4.4.4 Determine Blocked

Determine Blocked



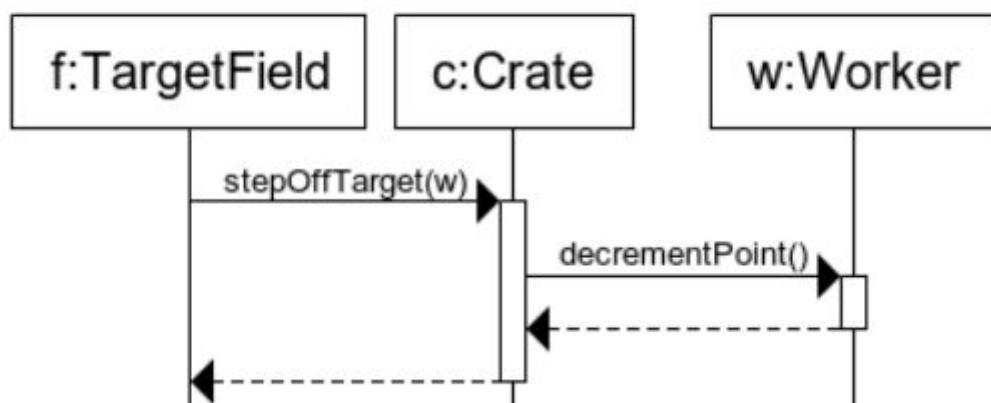
4.4.5 Crate steps on TargetField

Crate steps on TargetField



4.4.6 Crate steps off TargetField

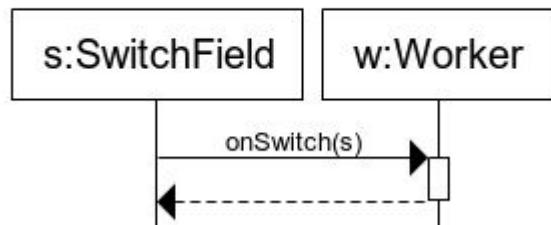
Crate steps off TargetField



4.4.7 Worker on Switch

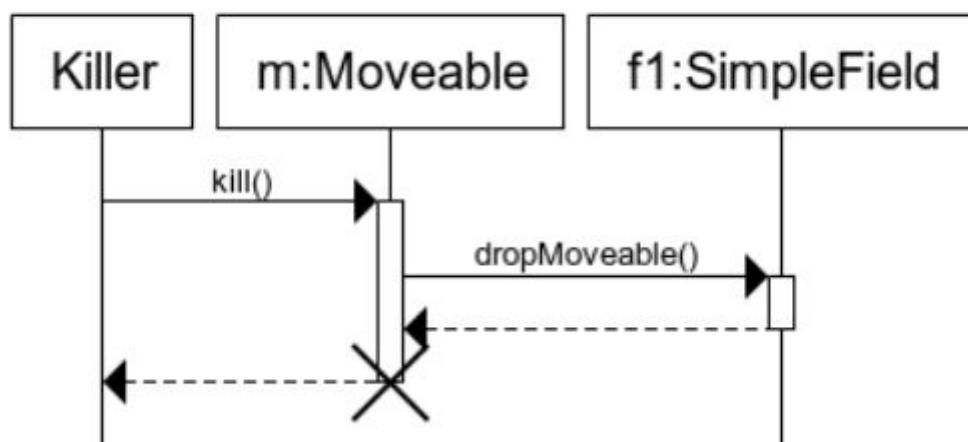
Worker nem csinál semmit a Switccsel

Worker on Switch



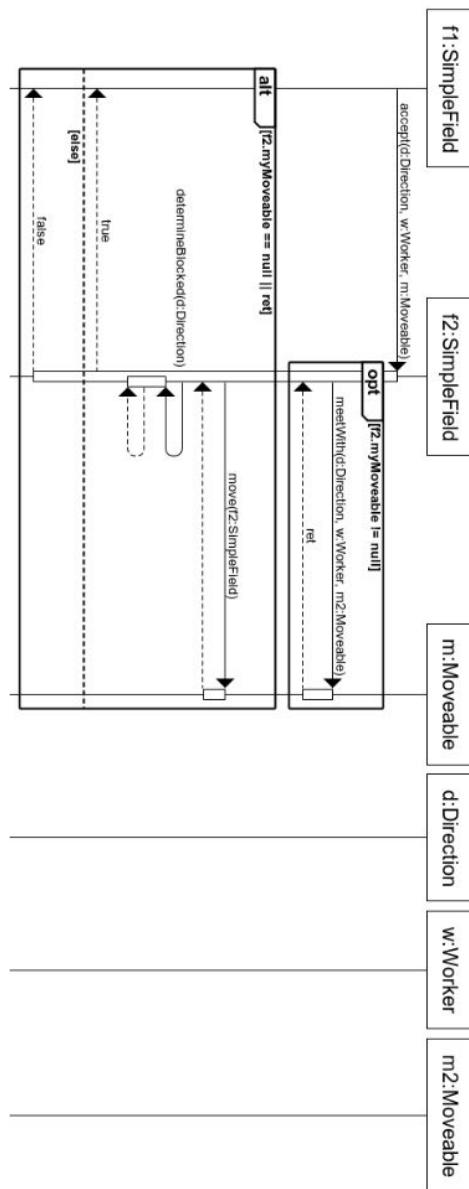
4.4.8 Moveable dies

Moveable dies



4.4.9 SimpleField gets accept request

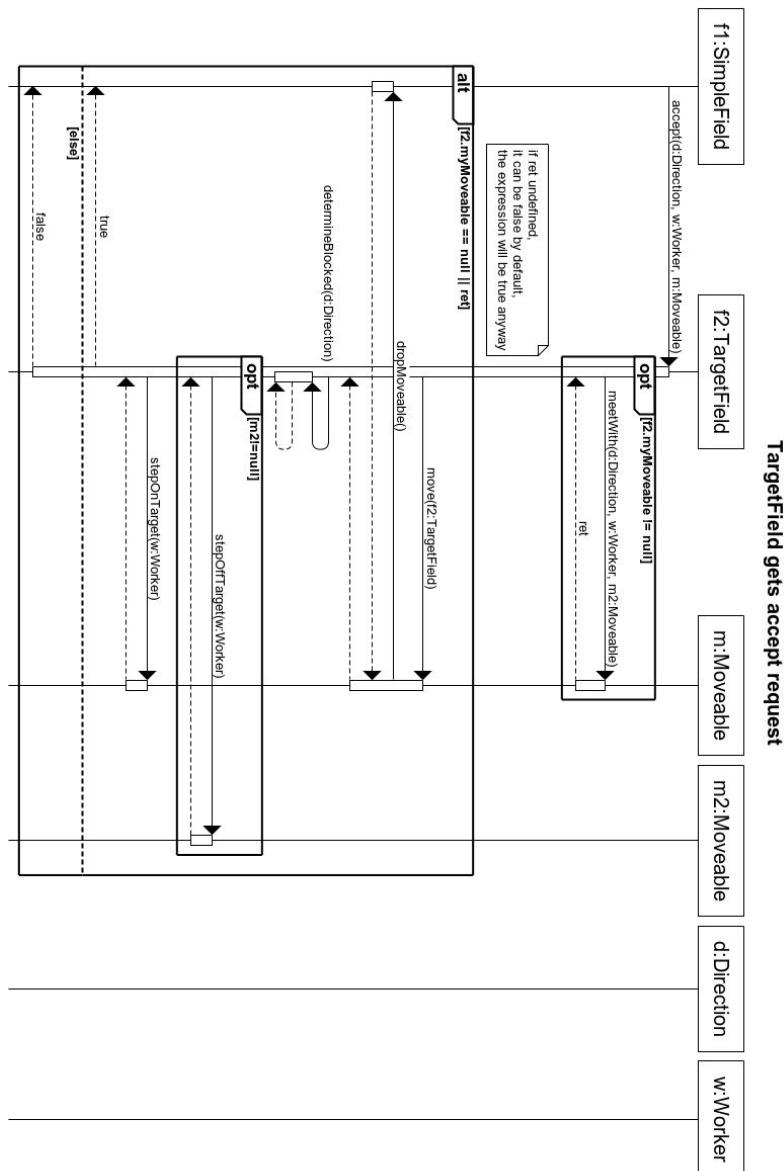
Megjegyzés: Az f2.myMoveable és az m2 ugyanazt az objektumot jelölik, tehát a f2 ismeri m2-et kezdtettől fogva. f3 pedig f2 megfelelő irányú szomszédja, tehát azt is ismeri.



4.4.10 TargetField gets accept request

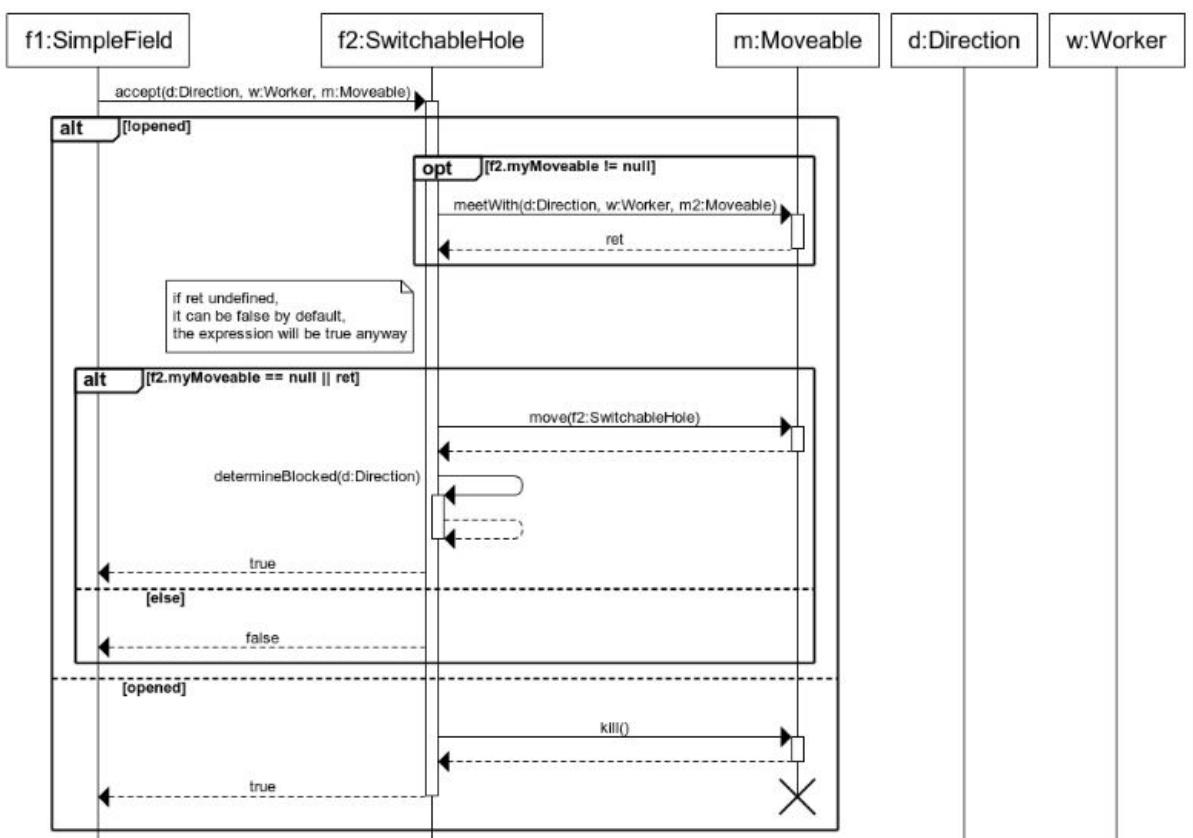
A `stepOffTarget` később hívódik, és egy `opt`-ba tettek, hogy csak akkor hívódjon, ha nem null objektumon akarjuk meghívni.

Megjegyzés: Az f2.myMoveable és az m2 ugyanazt az objektumot jelölik, tehát a f2 ismeri m2-et kezdettől fogva. f3 pedig f2 megfelelő irányú szomszédja, tehát azt is ismeri.



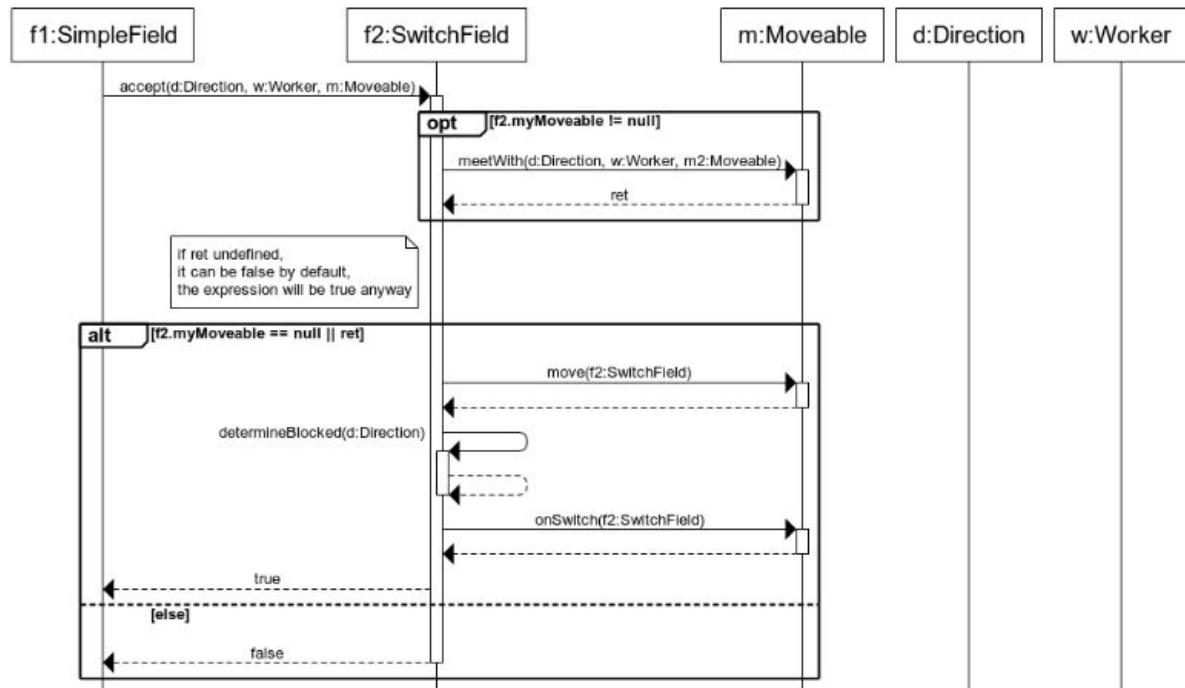
4.4.11 SwitchableHole gets accept request

Megjegyzés: Az f2.myMoveable és az m2 ugyanazt az objektumot jelölik, tehát a f2 ismeri m2-et kezdettől fogva. f3 pedig f2 megfelelő irányú szomszédja, tehát azt is ismeri.

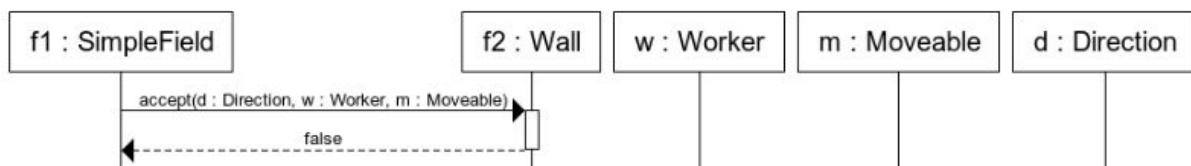


4.4.12 SwitchField gets accept request

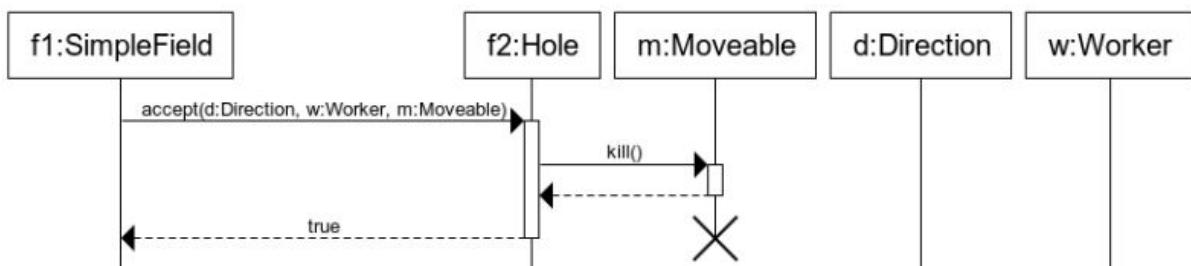
Megjegyzés: Az f2.myMoveable és az m2 ugyanazt az objektumot jelölik, tehát a f2 ismeri m2-et kezdettől fogva. f3 pedig f2 megfelelő irányú szomszédja, tehát azt is ismeri.



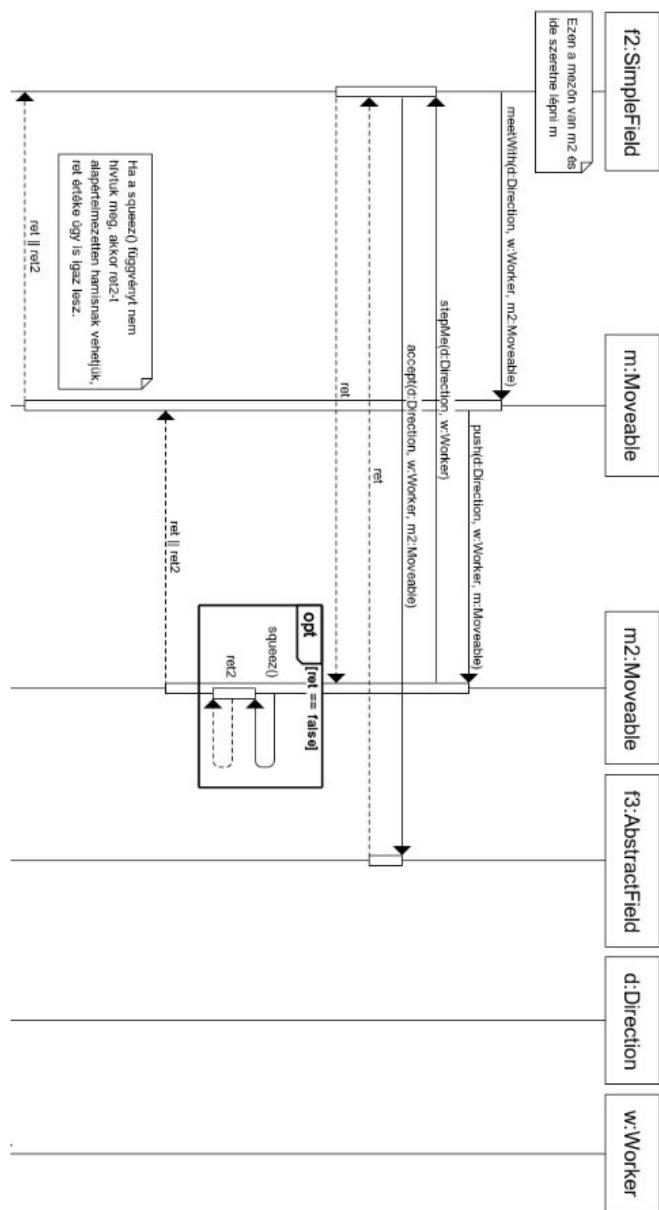
4.4.13 Wall gets accept request



4.4.14 Hole accepts Moveable

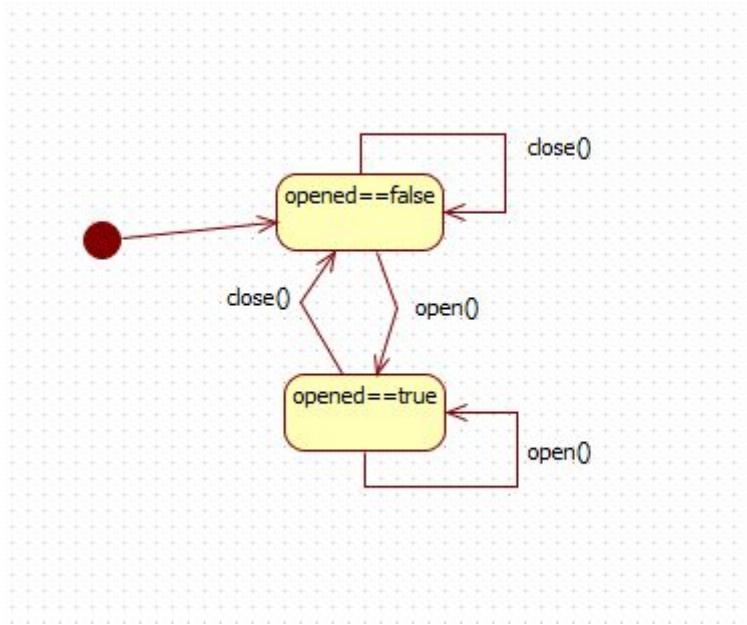


4.4.15 Moveable meets Moveable



4.5 State-chartok

SwitchableHole



4.6 Napló

Kezdet	Időtartam	Résznevők	Leírás
2018.02.27. 21:00	5 perc	Benkő	A hátralévő feladatok fedlapjainak kinyomtatása és bugyiba tétele
2018.02.28. 13:00	45 perc	Csapat	Ötletelés a kifogásolt részek módosítását illetően
2018.02.28. 14:00	5 perc	Schulcz, Zalavári	Konzultáció Simon Balázzsal a kiötlött módosítási előirányzatról
2018.02.28. 17:45	10 perc	Csapat	Megegyezés a módosítandó metódusokkal kapcsolatban. Feladatok nagyvonalú kiosztása és megegyezés egy péntek reggeli értekezletben
2018.02.28. 21:50	15 perc	Benkő	A változatlan (4.1, 4.5) és megváltoztatandó (4.3, 4.4) részek átemelése az előző feladatból

2018.02.28. 22:00	20 perc	Schulcz	Minta szekvenciadiagram elkészítése a léptetés újfajta menetéről
2018.03.01. 13:00	1 óra	Benkő	Az osztályleírások (4.3) átírása az új léptetésstruktúrának megfelelően.
2018.03.01. 14:00	10 perc	Benkő	A Schulcz által elkészített szekvenciadiagramok beillesztése, különös tekintettel azok olvashatóságára
2018.03.01. 13:50	35 perc	Schulcz	Frissítendő szekvenciadiagramok megrajzolása, osztálydiagram frissítése és beillesztése; a frissített Worker steps szekvencia beszúrása
2018. 03. 02. 14:00	1 óra	Zalavári	Dokumentum és diagramok ellenőrzése és javítása
2018. 03. 03. 12:00	50 perc	Schulcz	Lépést megvalósító szekvenciadiagramokon a csapat által talált hibák kijavítása
2018. 03. 04. 16:15	5 perc	Schulcz	Osztálydiagram frissítése az új egységes paramétersorrenddel
2018.03.04. 17:50	45 perc	Takács	Ellenőrzés, javítás, új szekvenciadiagramok beszúrása.
2018.03.11. 10:00	0.5 óra	Zalavári	Utólagos módosítások

5. Szkeleton tervezése

A modellben az eddigiekhez képest a következő apró változtatásokat hajtottuk végre:

A `dropMoveable()` függvényhívás rossz helyen volt, áttettük a `Moveable move()` függvényébe. Ez azért is logikusabb, mert minden meghívódik, ha egy `moveable` elhagy egy mezőt.

Ezt kihasználva a `SwitchField` definiálja felül saját `dropMoveable()` függvényét, és hívja meg benne saját `turnOff()` metódusát. Ezzel kiküszöböltük, hogy a munkásoknak expliciten kell kikapcsolni a kapcsolót, és nem használjuk ki, hogy egy mezőről biztosan munkás fog először lelépni.

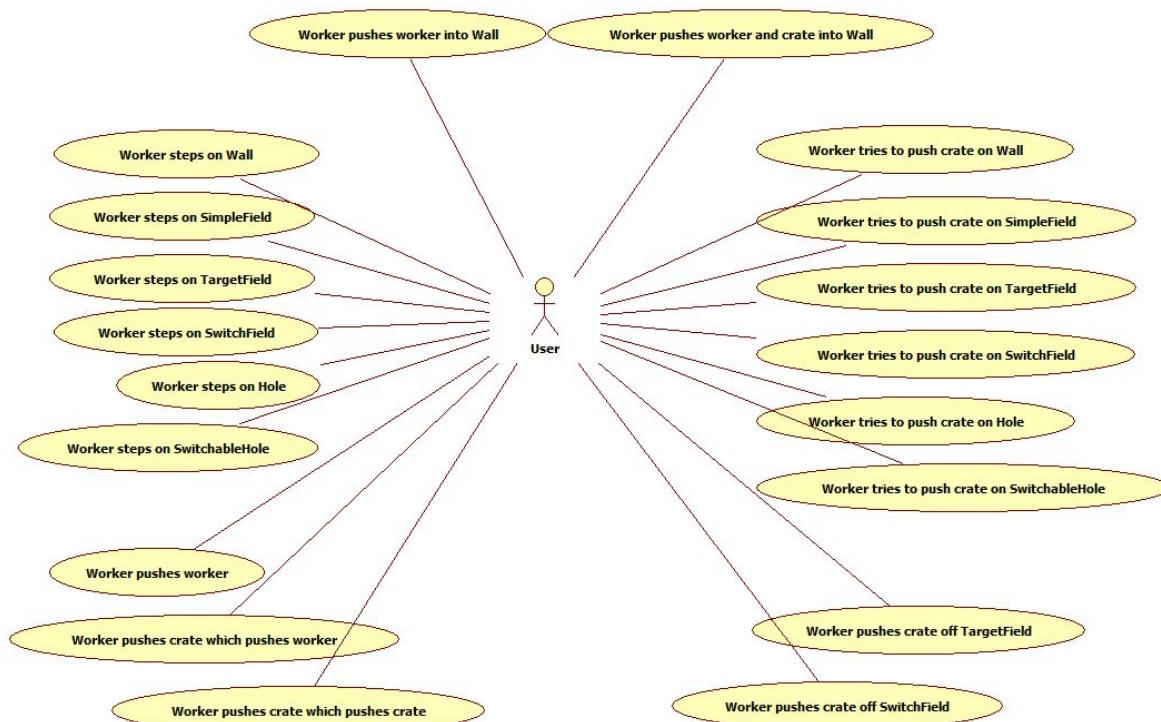
Ez további egyszerűsítéseket eredményezett. A munkásnak nem kell már felüldefiniálnia az `onSwitch(..)` örökolt függvényt, ha azt az űsben absztrakt helyett egy üres metódussal valósítjuk meg.

(Ezzel a `turnOff()` egy olyan függvényé változott, amit kívülről jelenleg senki nem hív meg, de ennek ellenére a szimmetria és a bővíthetőség érdekében ezt mégis publikus függvénynek hagyjuk meg.)

(Korábbi dokumentumokban frissítve - 2018. 05. 17.)

5.1 A szkeleton modell valóságos use-case-ai

5.1.1 Use-case diagram



5.1.2 Use-case leírások

Use-case neve	Worker steps on Wall
Rövid leírás	A munkás egy falra próbál lépni.
Aktorok	User

Forgatókönyv	1. A program a képernyőre írja a lépés megpróbálásával és meghiúsulásával járó függvényhívásokat.
---------------------	--

Use-case neve	Worker steps on SimpleField
Rövid leírás	Lemodelezük azt a folyamatot, ahogy egy munkás egy szomszédos, üres SimpleField típusú mezőre lép.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a lépéssel járó függvényhívásokat.

Use-case neve	Worker steps on TargetField
Rövid leírás	Lemodelezük azt a folyamatot, ahogy egy munkás egy szomszédos, üres TargetField típusú mezőre lép.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a lépéssel járó függvényhívásokat.

Use-case neve	Worker steps on SwitchField
Rövid leírás	Lemodelezük azt a folyamatot, ahogy egy munkás egy szomszédos, üres SwitchField típusú mezőre lép.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a lépéssel járó függvényhívásokat.

Use-case neve	Worker steps on Hole
Rövid leírás	Lemodelezük azt a folyamatot, ahogy egy munkás egy szomszédos lyukra lép és meghal.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a lépéssel és halállal járó függvényhívásokat.

Use-case neve	Worker steps on SwitchableHole
Rövid leírás	Lemodelezük azt a folyamatot, ahogy egy munkás egy szomszédos kapcsolható lyukra lép és esetleg meghal.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a lépéssel járó függvényhívásokat, majd megkérdezi a felhasználót, hogy a lyuk nyitva volt-e.
Alternatív forgatókönyv	1.A. Ha a lyuk nyitva volt, kirajzoljuk a halállal járóakat is.

Use-case neve	Worker tries to push crate on Wall
Rövid leírás	Lemodelezük azt a folyamatot, ahogy egy munkás falra akar tolni egy lánát.
Aktorok	User

Forgatókönyv	1. A program a képernyőre írja a tolás meghívásával járó függvényhívásokat.
---------------------	---

Use-case neve	Worker tries to push crate on SimpleField
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy ládát egy azzal szomszédos, üres SimpleField típusú mezőre tol.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a tolással járó függvényhívásokat.

Use-case neve	Worker tries to push crate on TargetField
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy ládát egy azzal szomszédos, üres TargetField típusú mezőre tol.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a tolással és pontszámítással járó függvényhívásokat.

Use-case neve	Worker tries to push crate on SwitchField
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy ládát egy azzal szomszédos, üres SwitchField típusú mezőre tol.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a tolással és kapcsoló átkapcsolásával járó függvényhívásokat.

Use-case neve	Worker tries to push crate on Hole
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy ládát egy azzal szomszédos Hole típusú mezőre tol.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a tolással és láda megsemmisülésével járó függvényhívásokat.

Use-case neve	Worker tries to push crate on SwitchableHole
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy ládát egy azzal szomszédos, üres SwitchableHole típusú mezőre tol.
Aktorok	User

Forgatókönyv	1. A program a képernyőre írja a tolással járó függvényhívásokat, majd megkérdezi a felhasználót, hogy a lyuk nyitva volt-e.
Alternatív forgatókönyv	1.A. Ha a lyuk nyitva volt, kirajzoljuk a láda megsemmisülésével járóakat is.

Use-case neve	Worker pushes worker
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy SimpleField-ekből álló pályán egy munkás megtol egy másikat.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a tolással járó függvényhívásokat.

Use-case neve	Worker pushes crate which pushes worker
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás által tolta a láda megtol egy másik munkást.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a tolással járó függvényhívásokat.

Use-case neve	Worker pushes crate which pushes crate
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás által tolta a láda megtol egy másik ládát.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a tolással járó függvényhívásokat.

Use-case neve	Worker pushes crate off TargetField
Rövid leírás	Lemodellezük azt a folyamtot, ahogy egy munkás letol egy ládat egy célmezőről.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a tolással és pontszámítással járó függvényhívásokat.

Use-case neve	Worker pushes crate off SwitchField
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás letol egy ládat egy kapcsolóról, ezzel kikapcsolva azt.

Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a tolással és kikapcsolással járó függvényhívásokat.

Use-case neve	Worker pushes worker into wall
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás felnak tol egy másikat, ezzel összenyomva őt.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a tolás megpróbálásával, meghiúsulásával és összenyomással járó függvényhívásokat.

Use-case neve	Worker tries to push worker which pushes worker on Wall
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás megtol egy másikat, ami megtol egy harmadikat, ezzel összenyomva őt.
Aktorok	User
Forgatókönyv	1. A program a képernyőre írja a tolás megpróbálásával, és összenyomással járó függvényhívásokat.

5.2 A szkeleton kezelői felületének terve, dialógusok

A program indításkor egy listát ír ki a konzolra, melyben előre definiált forgatókönyvek közül lehet választani. Bemenetként ekkor a szcenáriókhoz rendelt számok egyikét várja. A szám begépelése után kiírja a szcenárió nevét, majd sorra az annak elvégzése alatt vérehajtott függvényhívásokkal kapcsolatos információkat, a következőképpen:

{tabulálás} {az objektum neve, amelyen a függvényt hívtuk}.{a függvény neve} {soremelés}

Az objektumok a szcenárióban résztvevő objektumok, a függvények ezek publikus függvényei, a tabulálás mértékét pedig az határozza meg, hogy a hívási láncban minden mélyen hívtuk a függvényt. Utóbbinak megfelelően, ha pl. az *a()* függvény belsejében meghívja a *b()* függvényt, a *b()* az *a()* alatt és annál valamivel beljebb kezdődik a konzolon.

a()
b()

Viszont ha a *c()* függvény meghívja *a()*-t, majd annak visszatérése után *b()*-t, akkor *b()* az *a()* alatt és vele azonos tabulálással lesz látható, valamint mindenketten *c()* alatt és annál beljebb.

c()
a()
b()

A main függvény által először hívott függvény tabulálása 0-s.

Az egyes függvényeknek esetenként felhasználói beavatkozásra lesz szükségük, ilyenkor a kimenetre kiírnak egy kérdést (pl.: Nyitva van-e az s1 kapcsolható lyuk?). Miután a felhasználó meghozta a döntést, beírja azt a konzolon, a program pedig tovább futhat. Pl.:

s1.accept()
Nyitva van-e az s1 kapcsolható lyuk? (I/N) I
c1.kill()

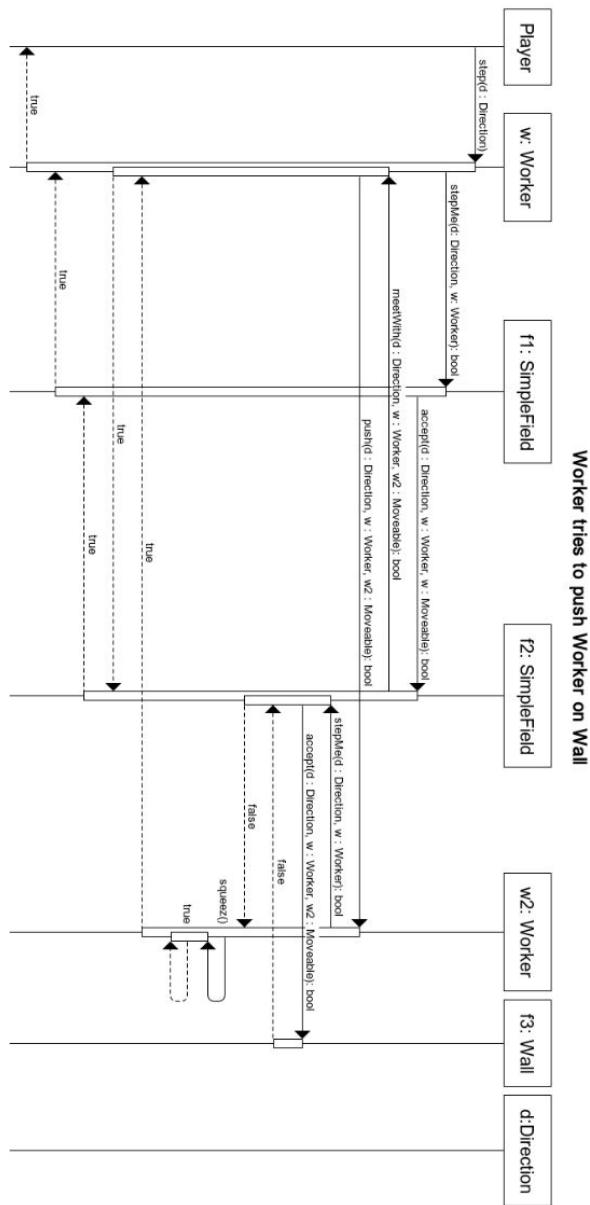
(Megj.: A kérdést a jobb olvashatóság érdekében nem tabuláljuk. A második sor végén álló I a felhasználó válasza, melyet helytakarékkossági okokból a kérdéssel egy sorba várunk.)

5.3 Szekvencia diagramok a belső működésre

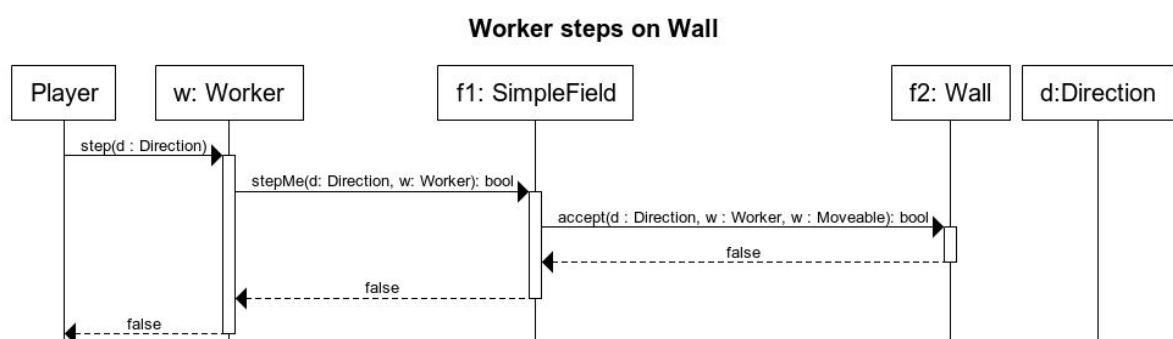
A szekvenciadiagramokról hiányzik a determineblocked függvényhívás. Ennek okai a következők:

- Ez egy olyan protected függvény, amelyet a mezőnek minden move után meg kell hívnia. Ezért csak tovább bonyolítaná a diagrammokat, de érdemi működést nem adna hozzá. (Hiszen ez csak a játék vége detektálásához szükséges.)
- A függvény minden esetben a saját mezőjének három szomszédjával is kommunikál. Ez azt jelentené, hogy a minipályákon, amiket a use-case-ekhez készítünk, kb. 3-szor több mezőt kéne létrehozni, a megfelelő szomszédságokat beállítani. Ez átláthatatlanná tenné a kommunikációs diagramokat is. Továbbá a szekvenciadiagramokon is nehézkes/hosszadalmas lenne ennek a működésnek a bemutatása.

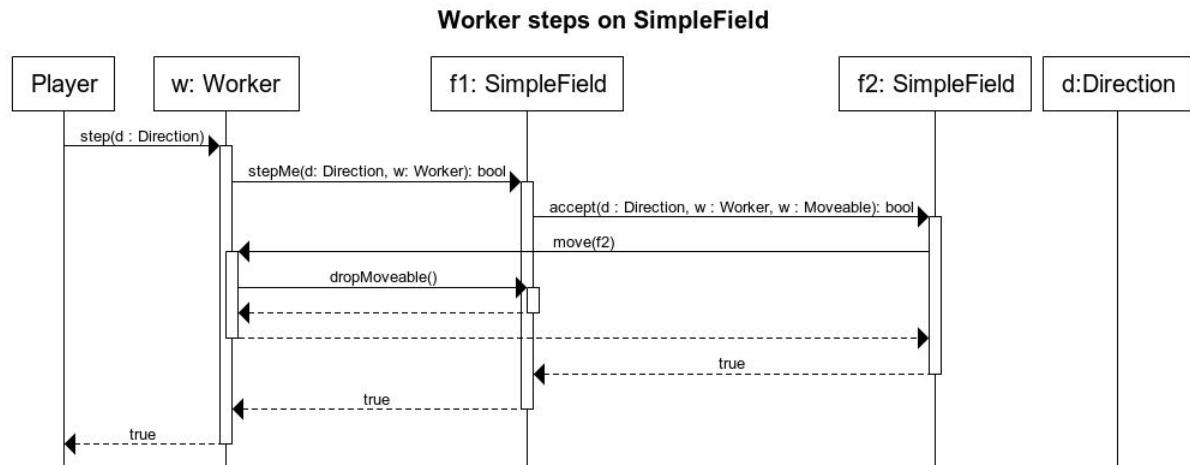
5.3.1 Worker tries to push Worker into Wall



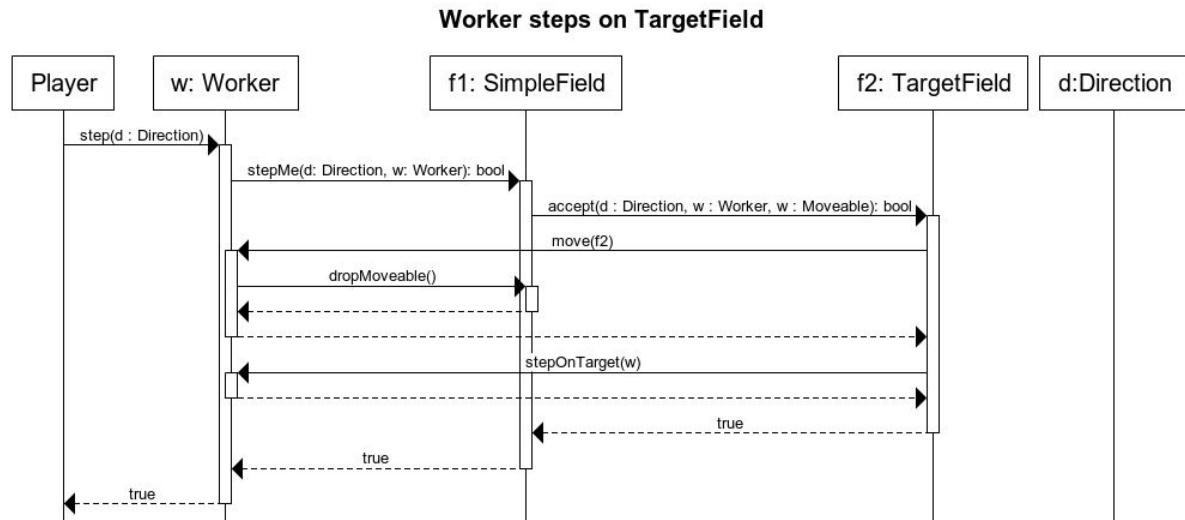
5.3.2 Worker steps on Wall



5.3.3 Worker steps on SimpleField

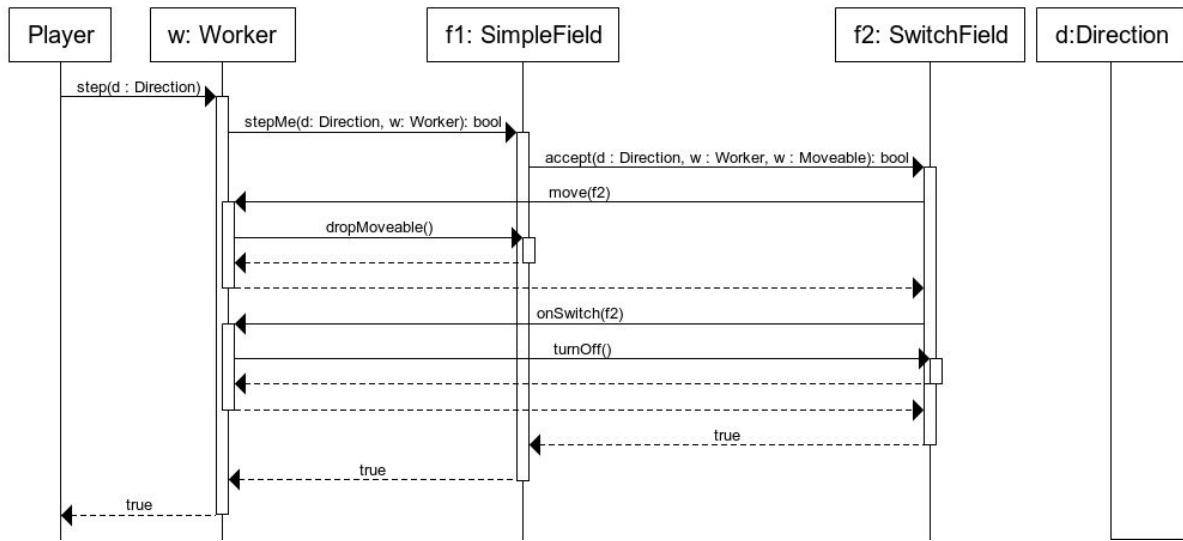


5.3.4 Worker steps on TargetField



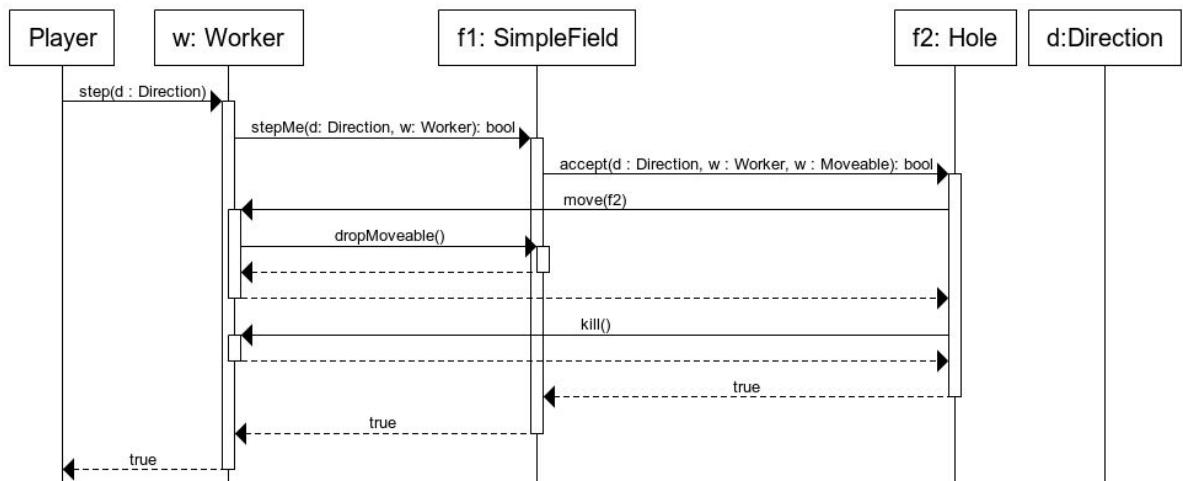
5.3.5 Worker steps on SwitchField

Worker steps on SwitchField



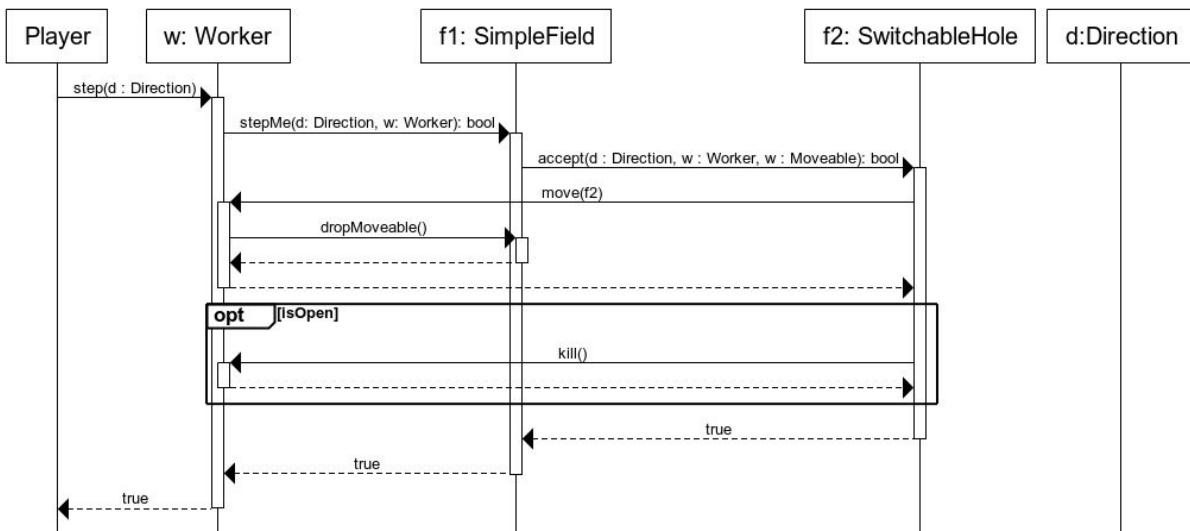
5.3.6 Worker steps on Hole

Worker steps on Hole

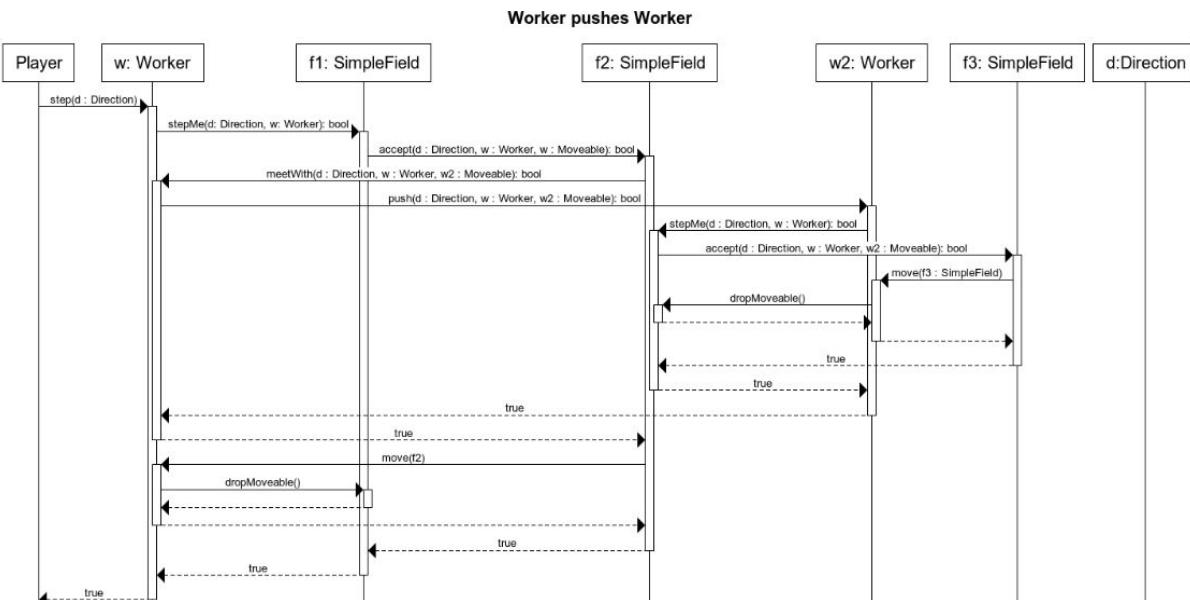


5.3.7 Worker steps on SwitchableHole

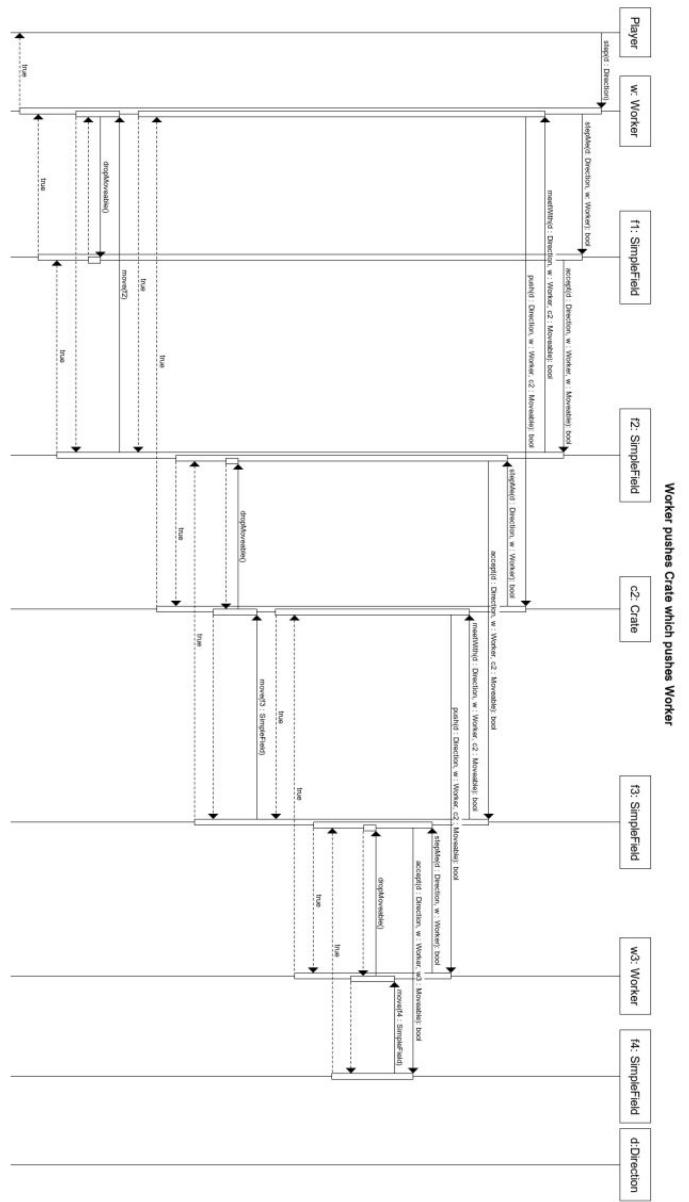
Worker steps on SwitchableHole



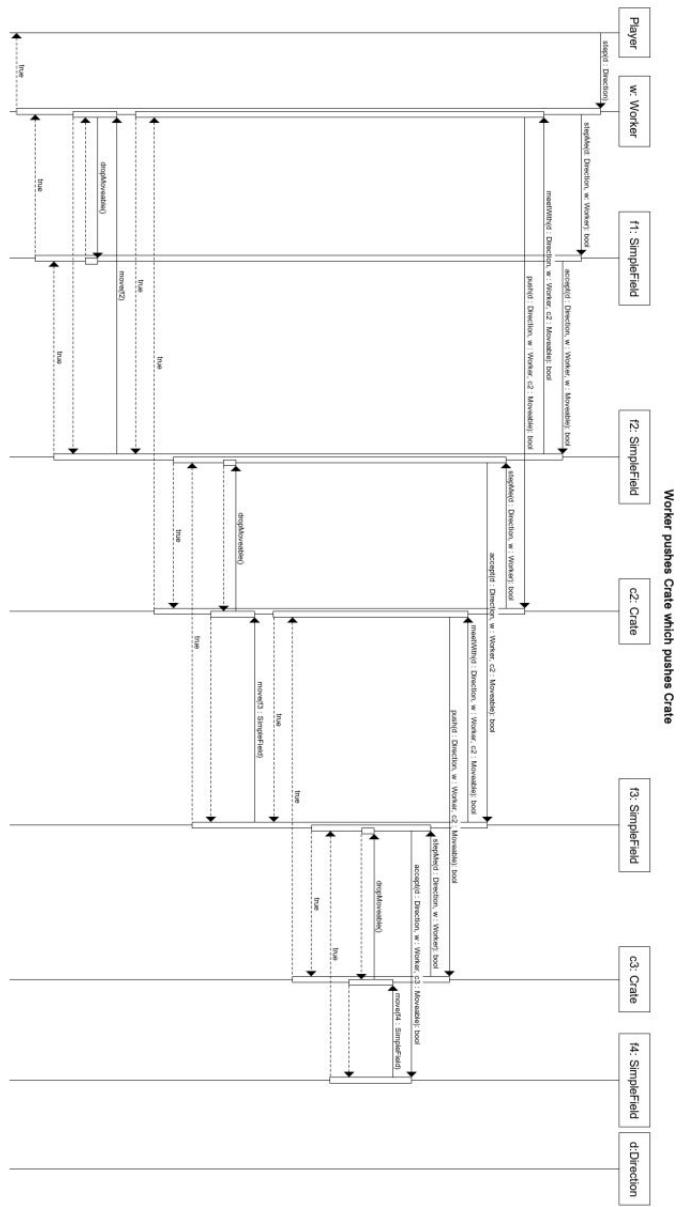
5.3.8 Worker pushes Worker



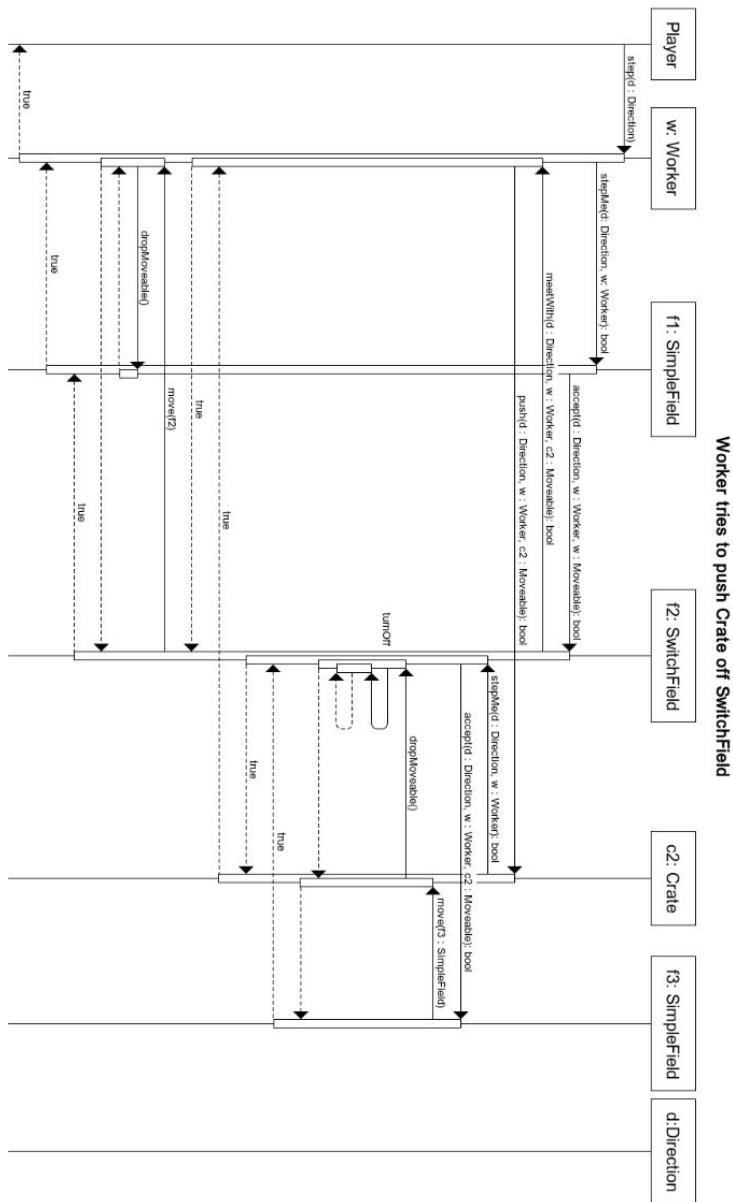
5.3.9 Worker pushes Crate which pushes Worker



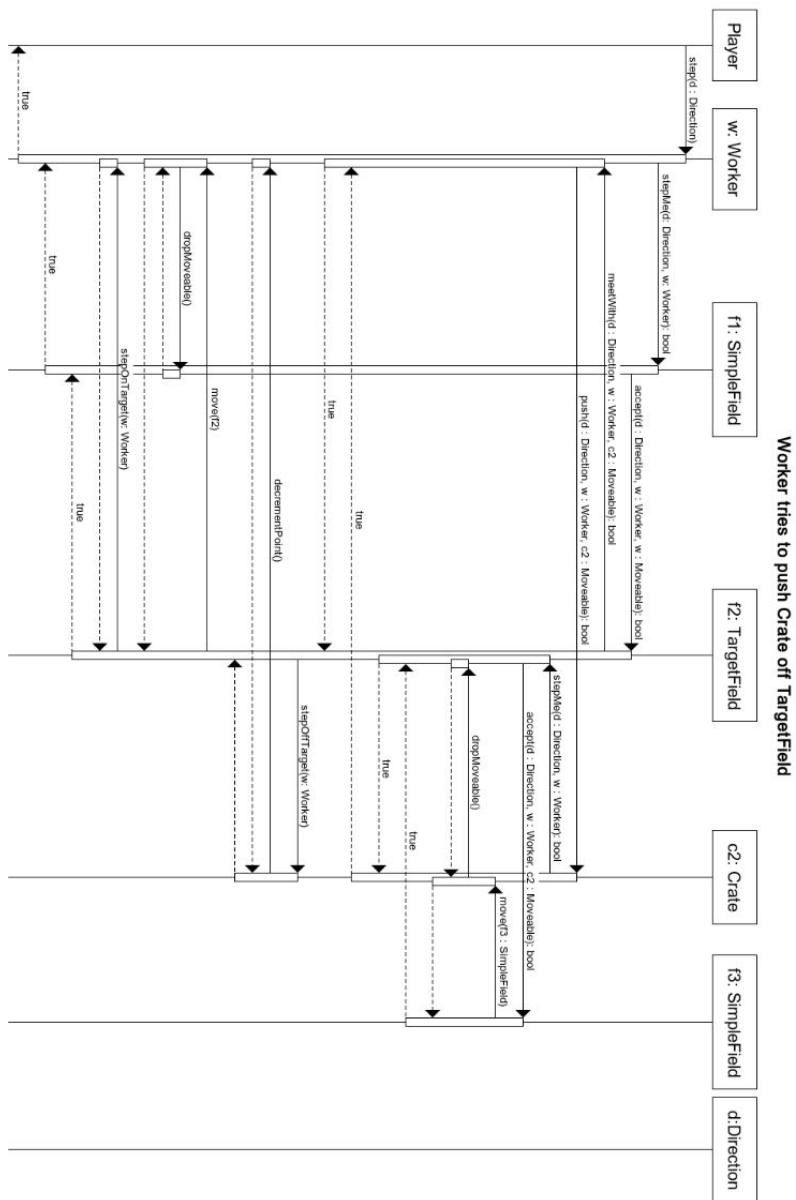
5.3.10 Worker pushes Crate which pushes Worker



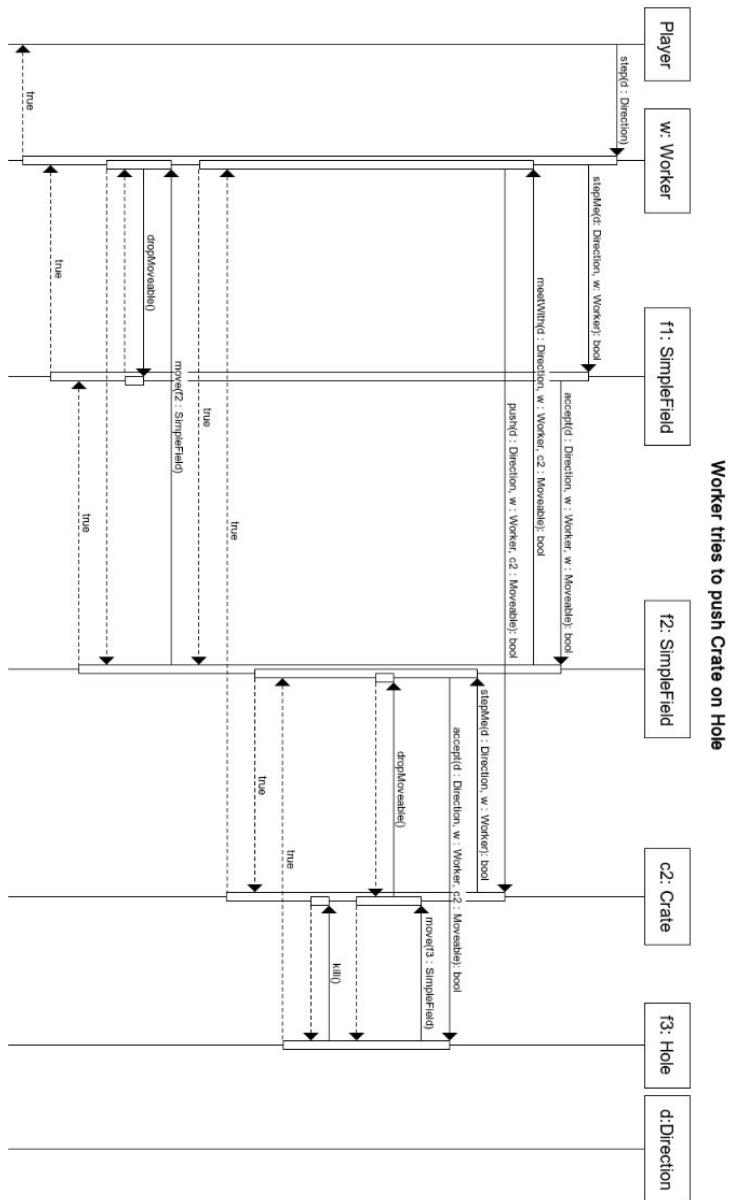
5.3.11 Worker pushes Crate off SwitchField



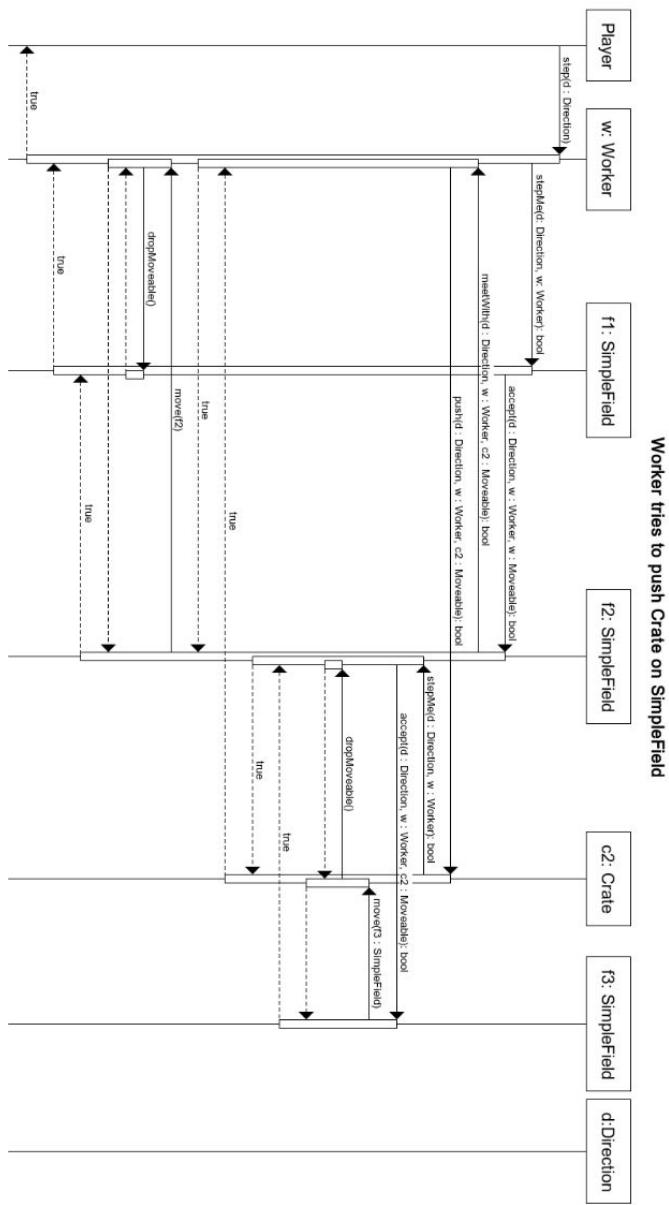
5.3.12 Worker pushes Crate off TargetField



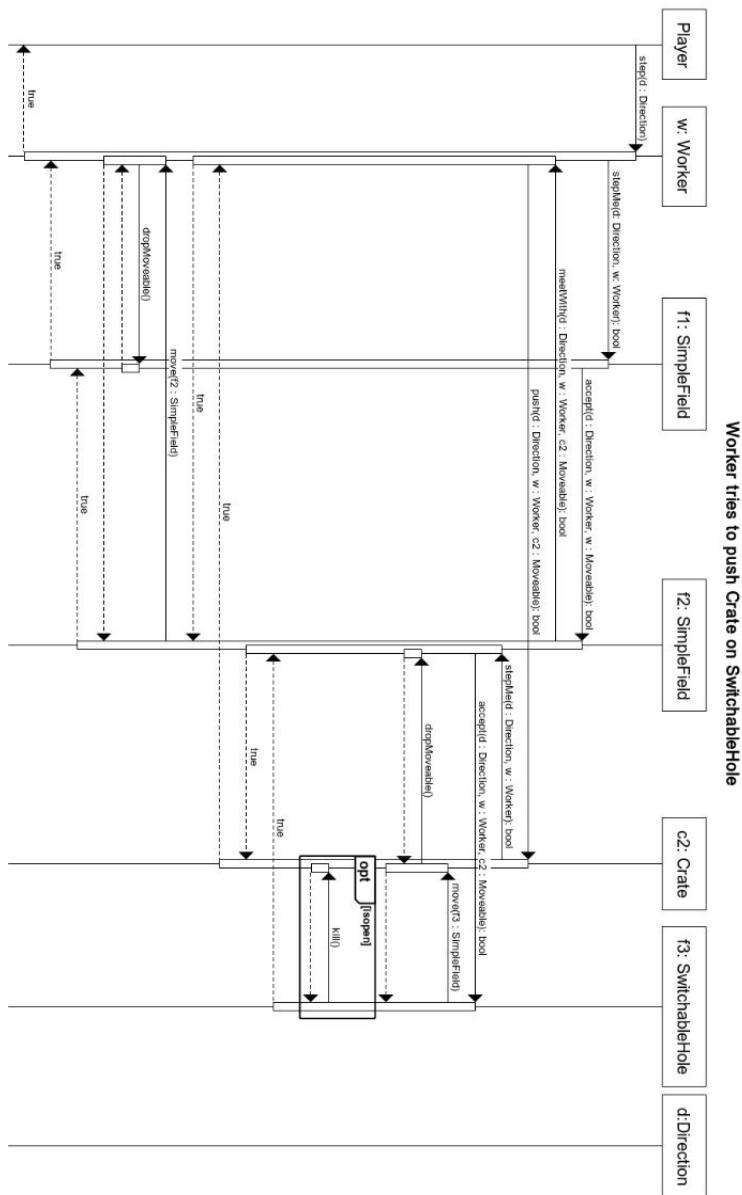
5.3.13 Worker tries to push Crate on Hole



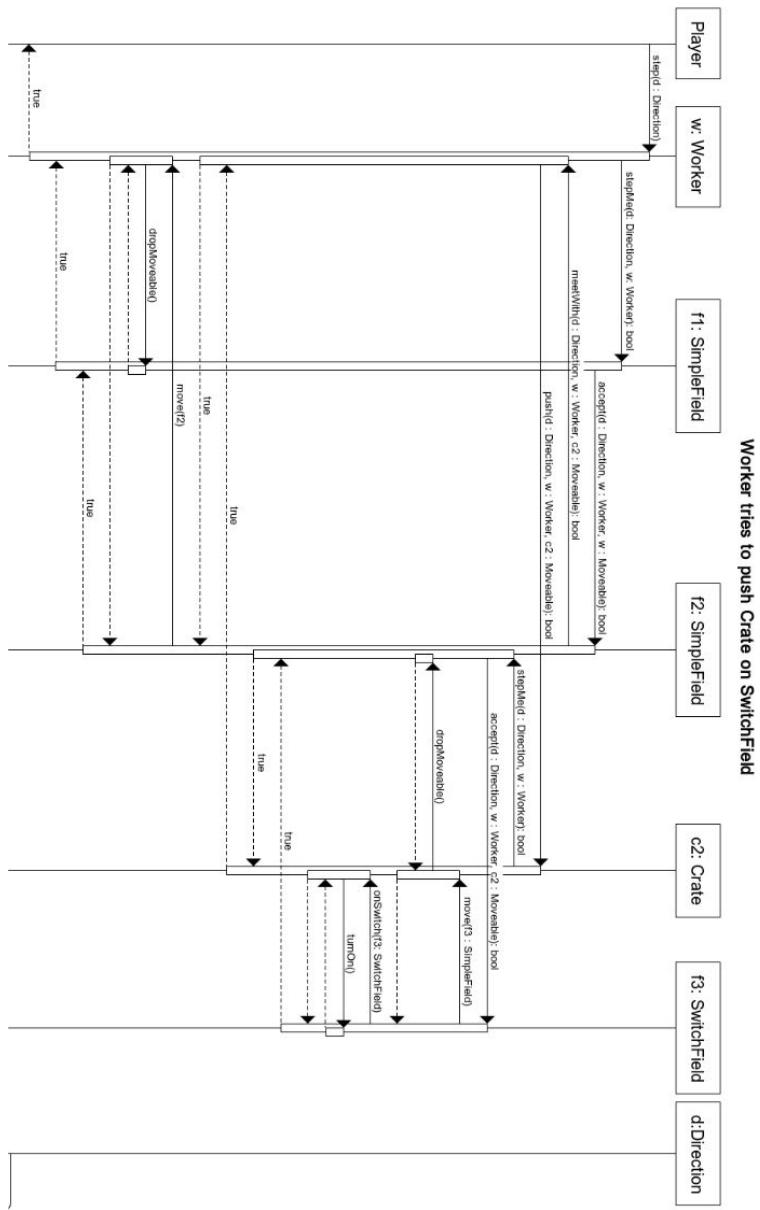
5.3.14 Worker tries to push Crate on SimpleField



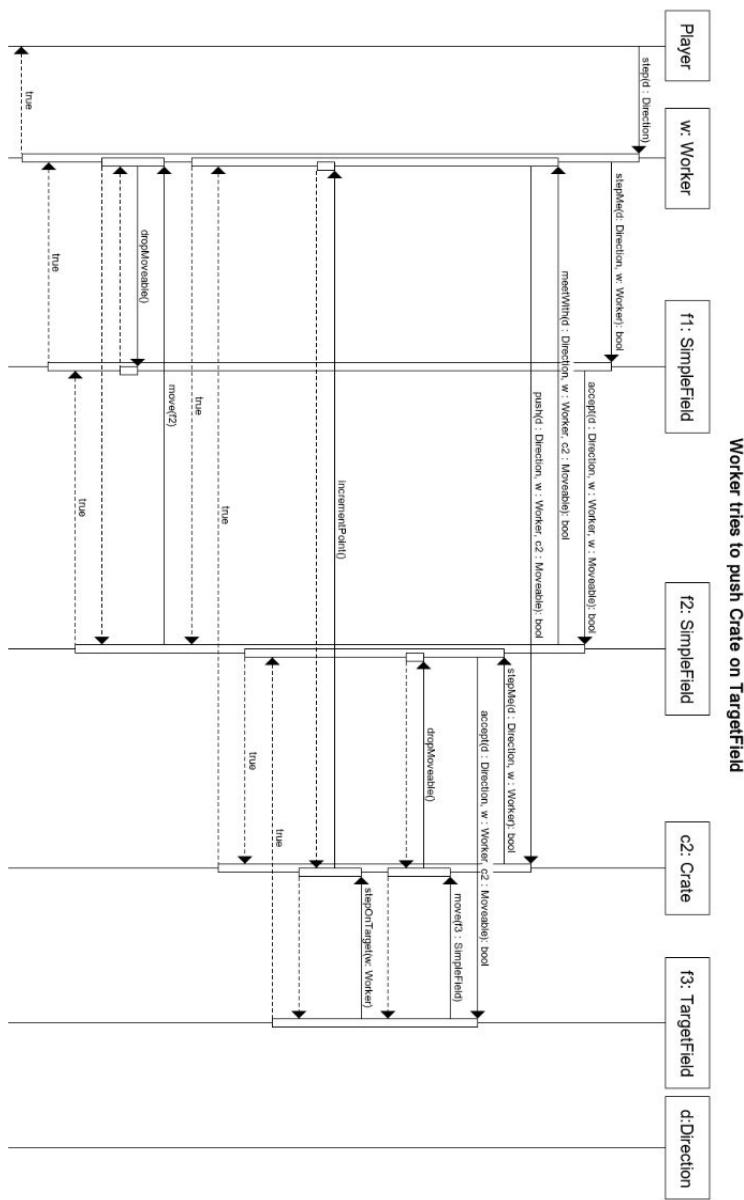
5.3.15 Worker tries to push Crate on SwitchableHole



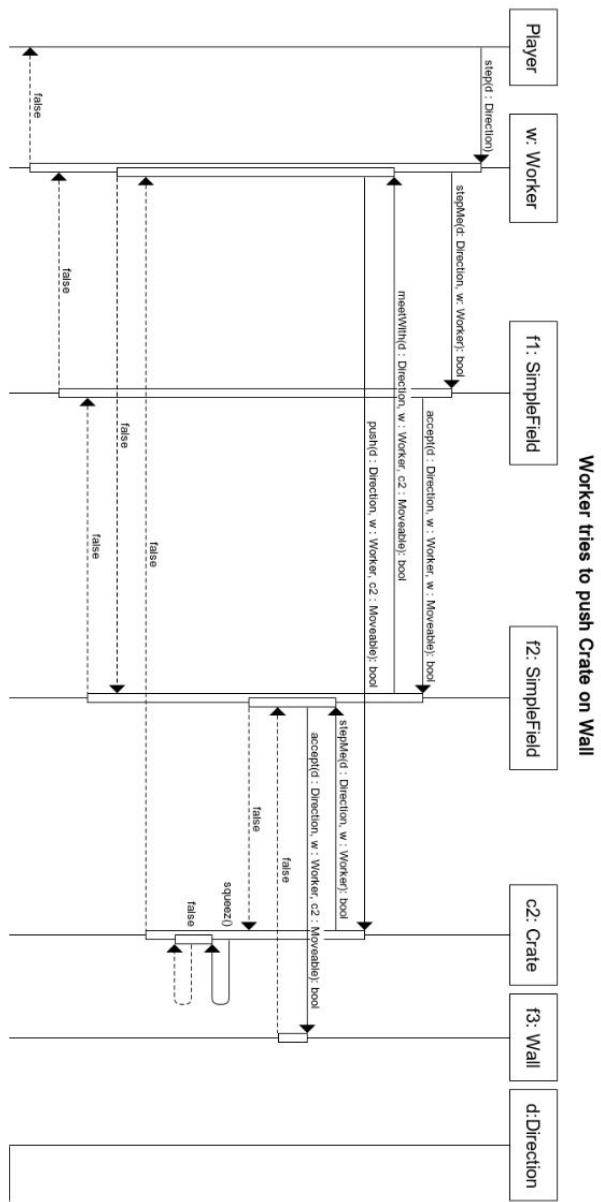
5.3.16 Worker tries to push Crate on SwitchField



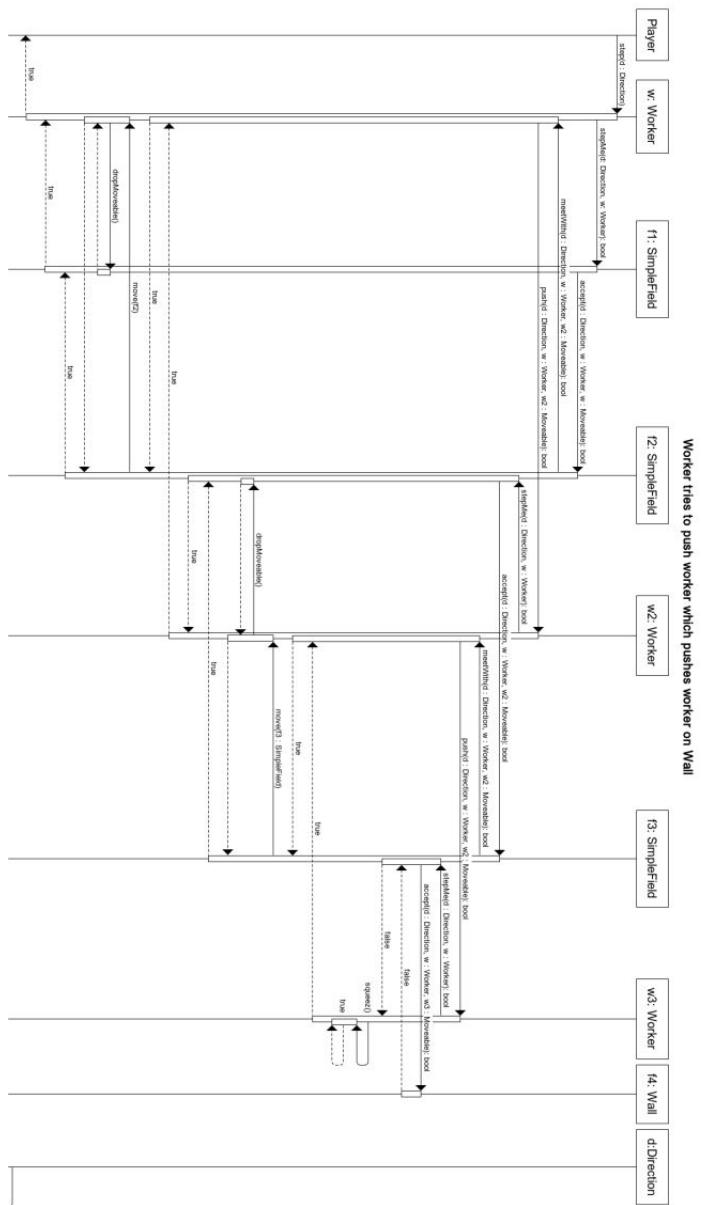
5.3.17 Worker tries to push Crate on TargetField



5.3.18 Worker tries to push Crate on Wall



5.3.19 Worker tries to push worker which pushes worker on Wall

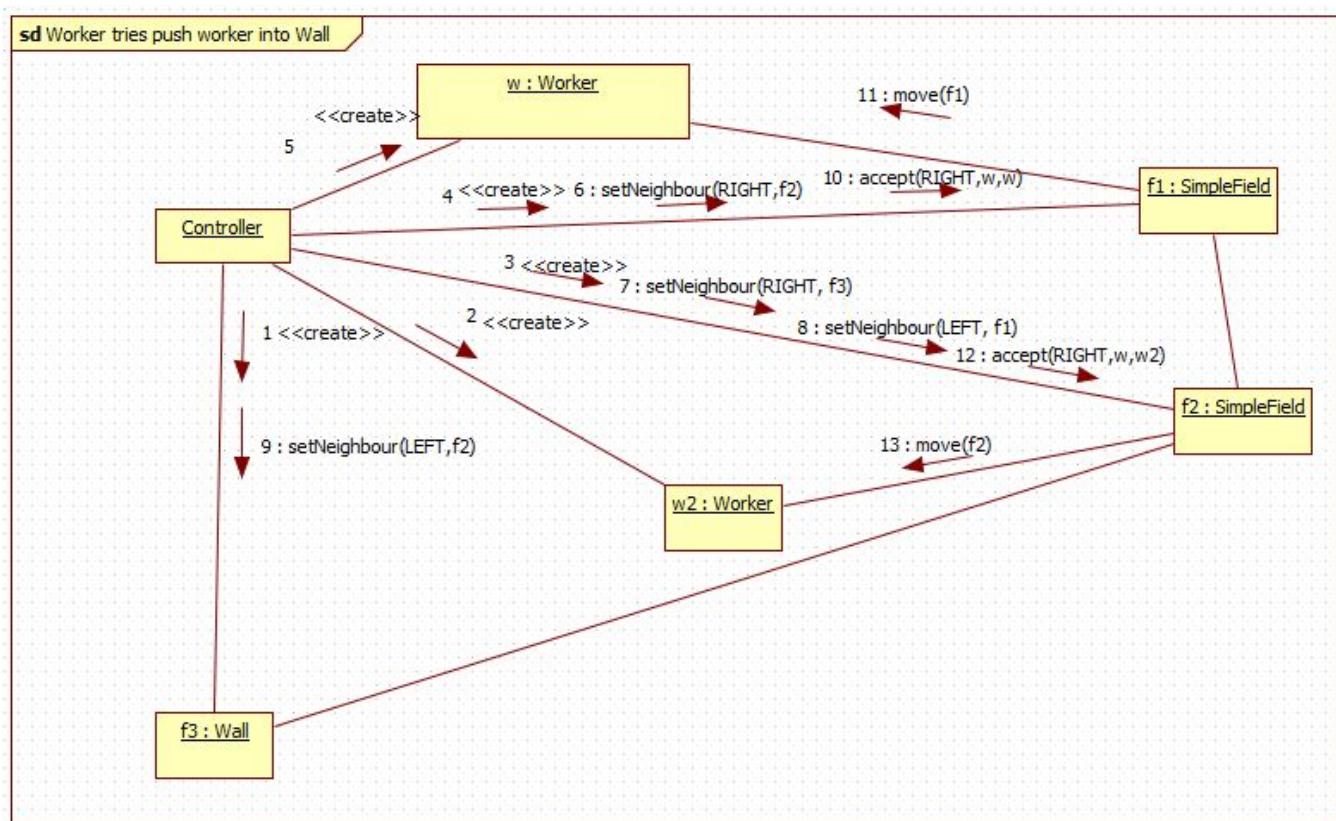


5.4 Kommunikációs diagramok

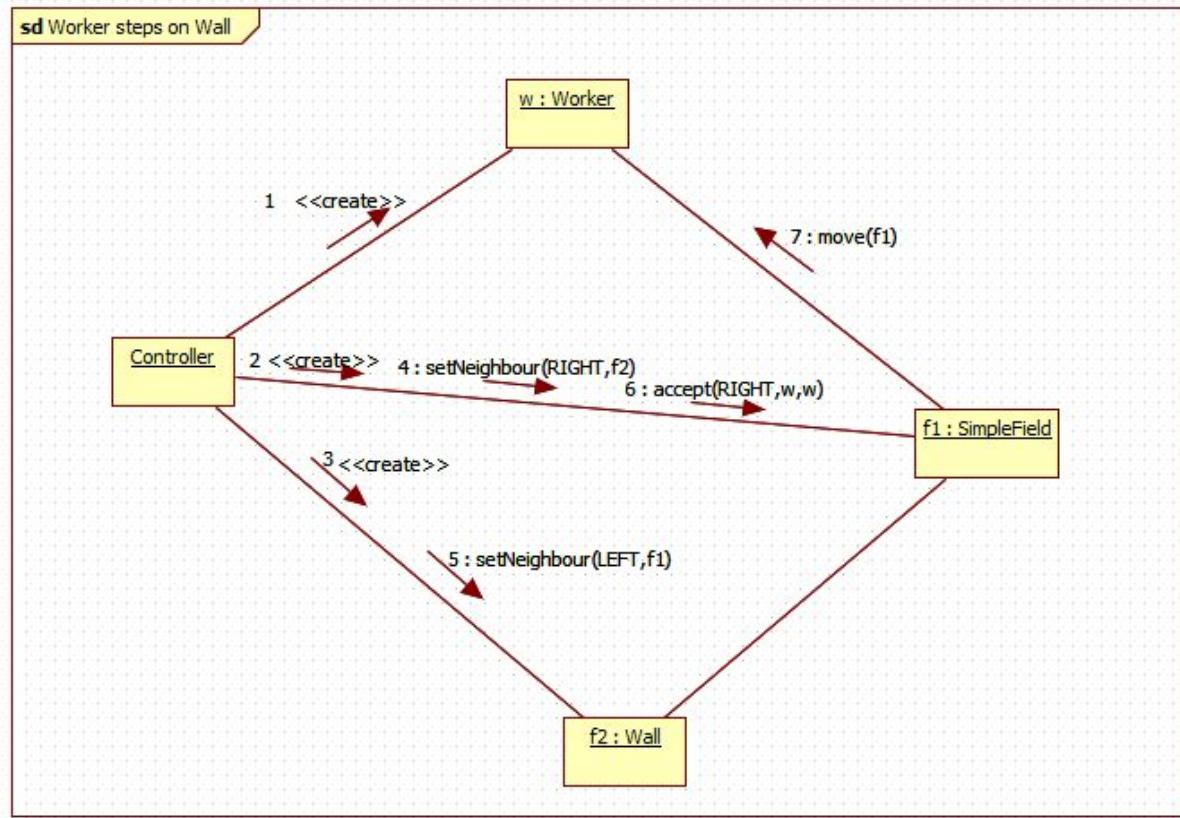
Megjegyzés:

1. *a <<create>> hívások nyilai teli fejűek üres fejűek helyett, ezeket a modellező eszköz nem támogatta.*
2. *A visszatérési értékeket a könnyebb olvashatóság érdekében nem jelöltük. Ezek értéke vagy void vagy true.*
3. *A modellező eszköz a hierarchikus számozást nem támogatja. Ennek csak az accept függvény hatására meghívódó move-nál van jelentősége.*

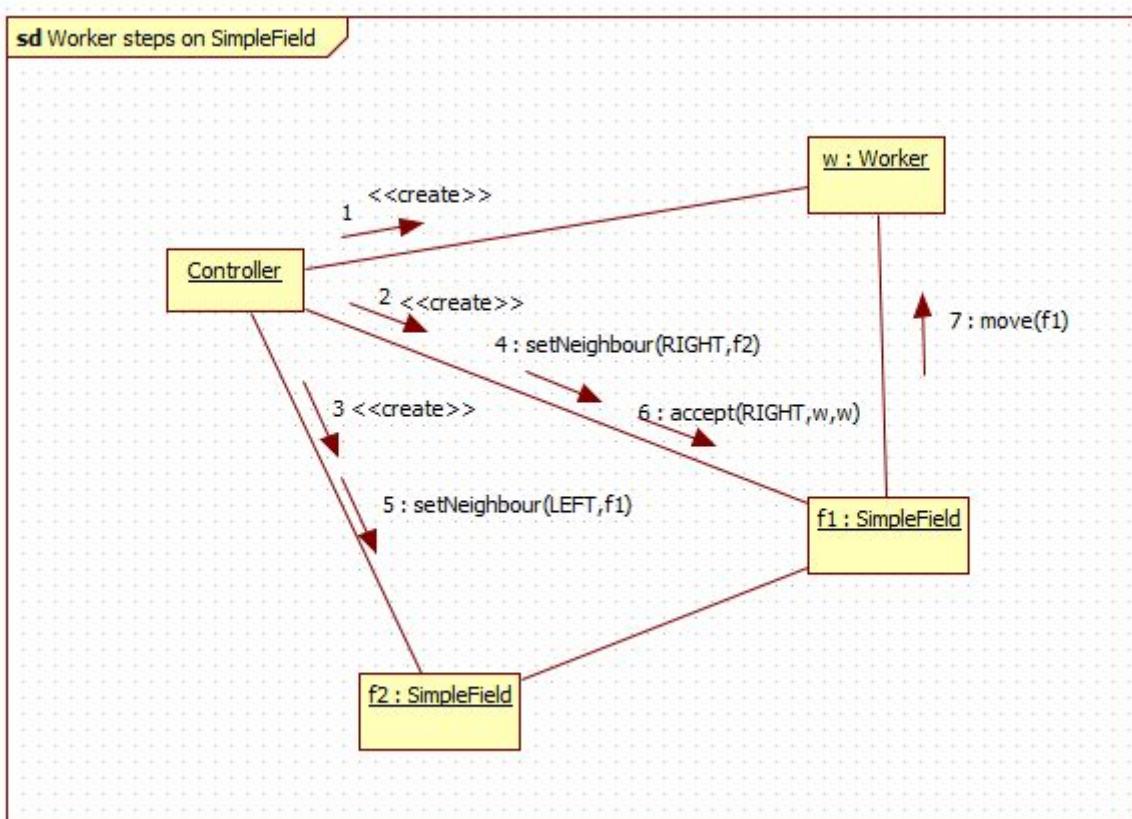
5.4.1 Worker tries to push Worker into Wall



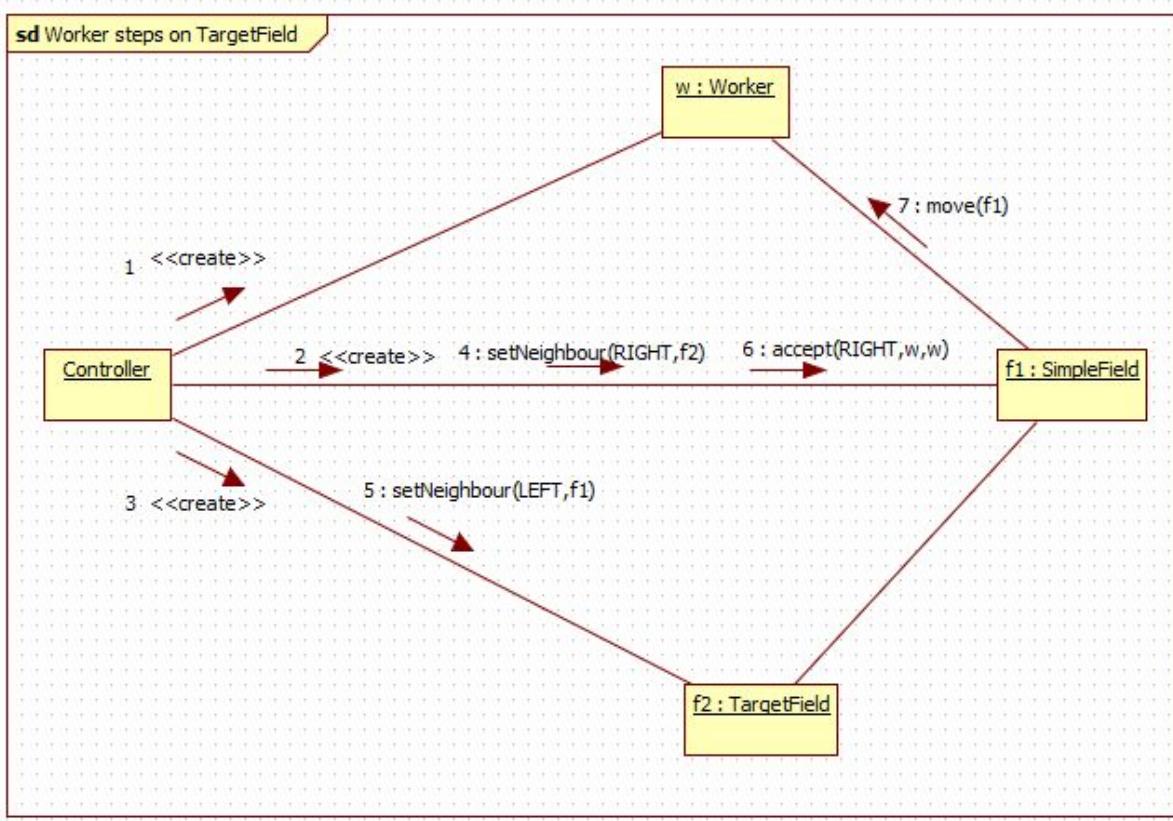
5.4.2 Worker steps on Wall



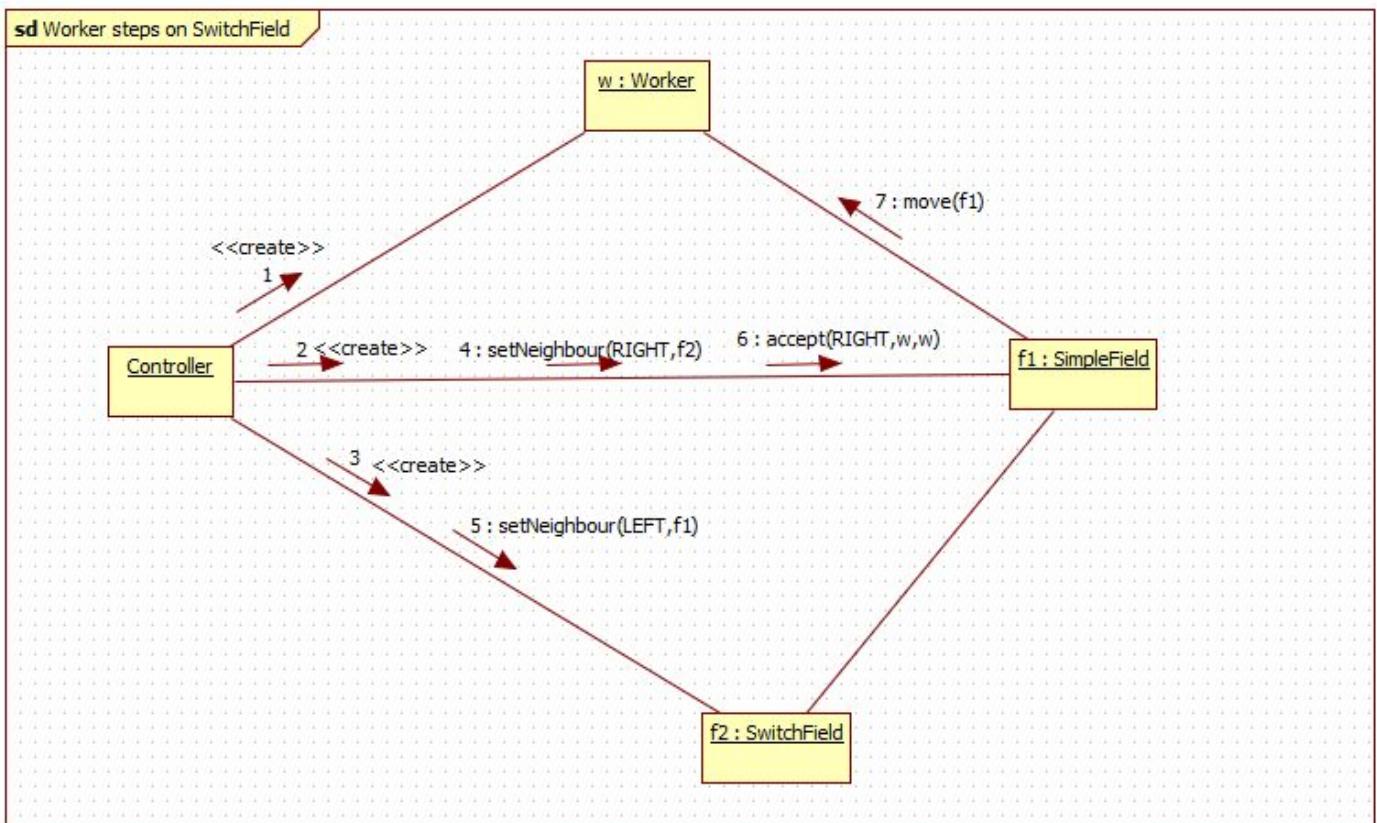
5.4.3 Worker steps on SimpleField



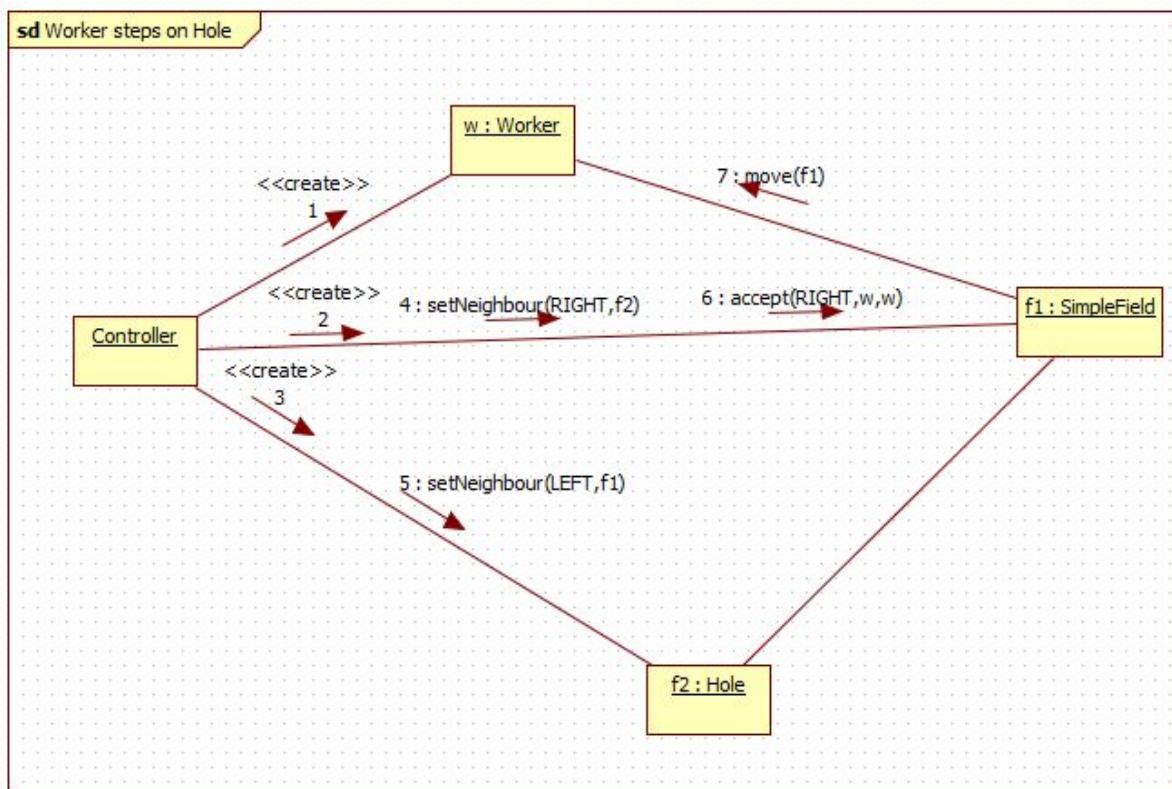
5.4.4 Worker steps on TargetField



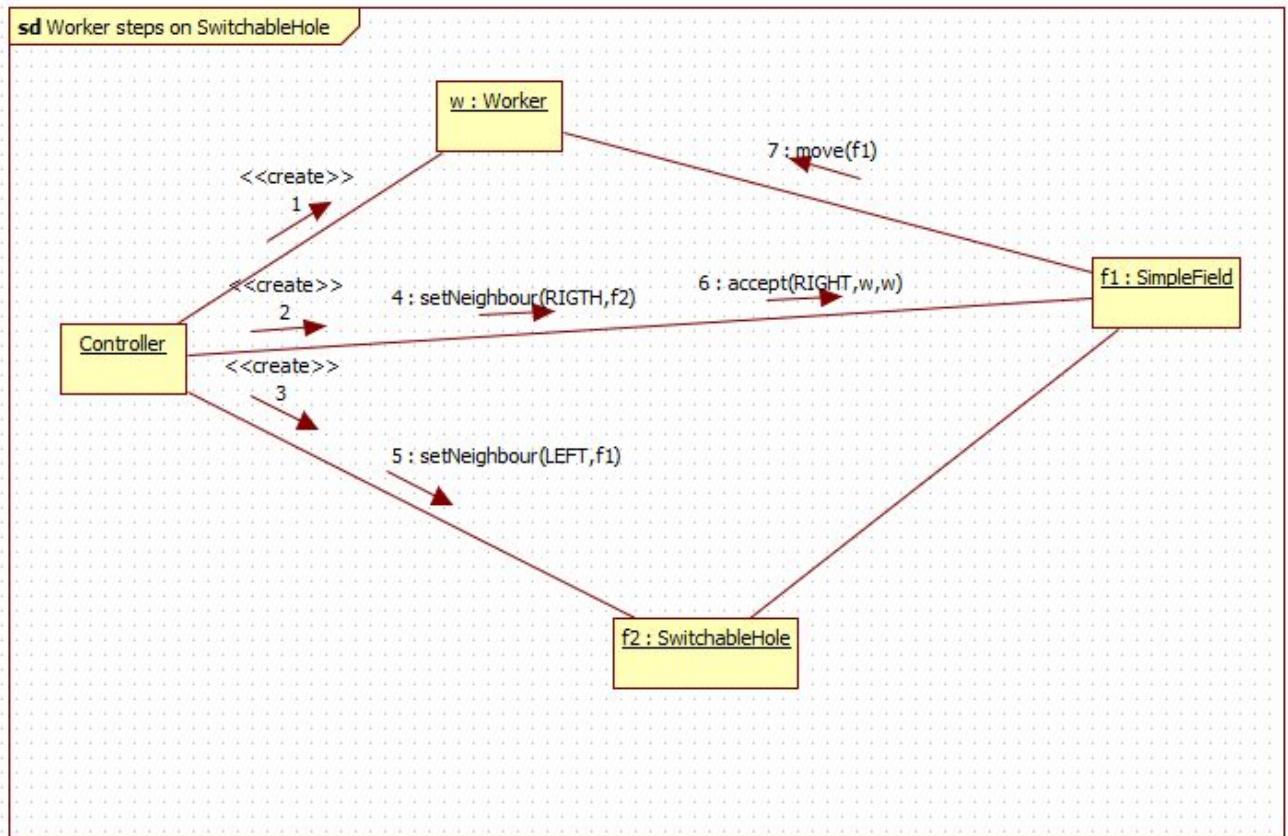
5.4.5 Worker steps on SwitchField



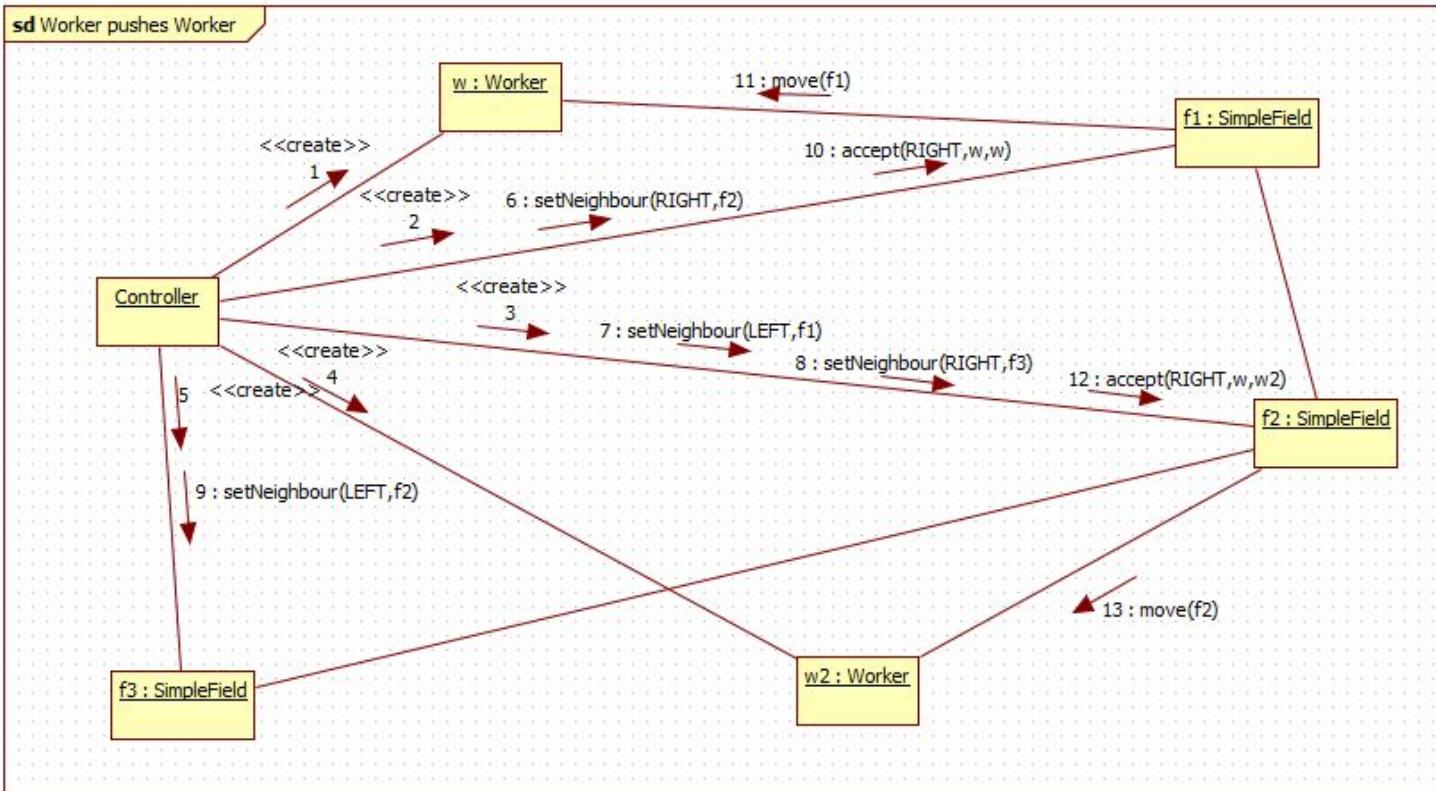
5.4.6 Worker steps on Hole



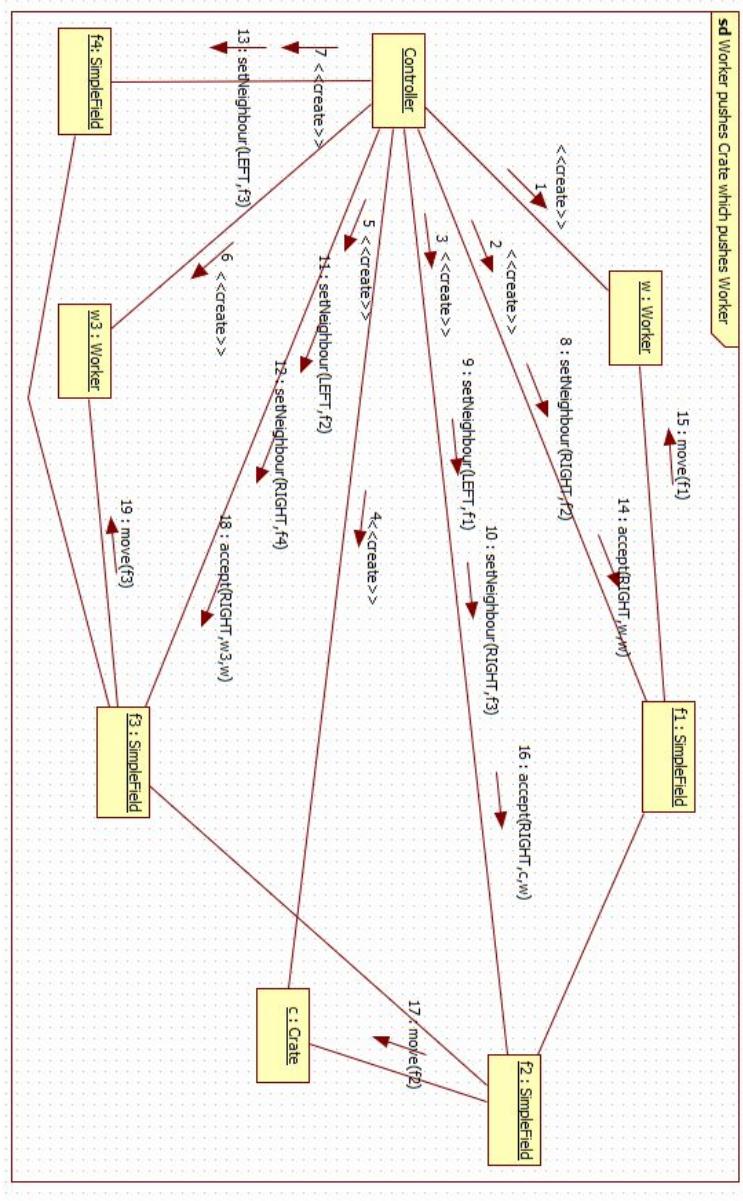
5.4.7 Worker steps on SwitchableHole



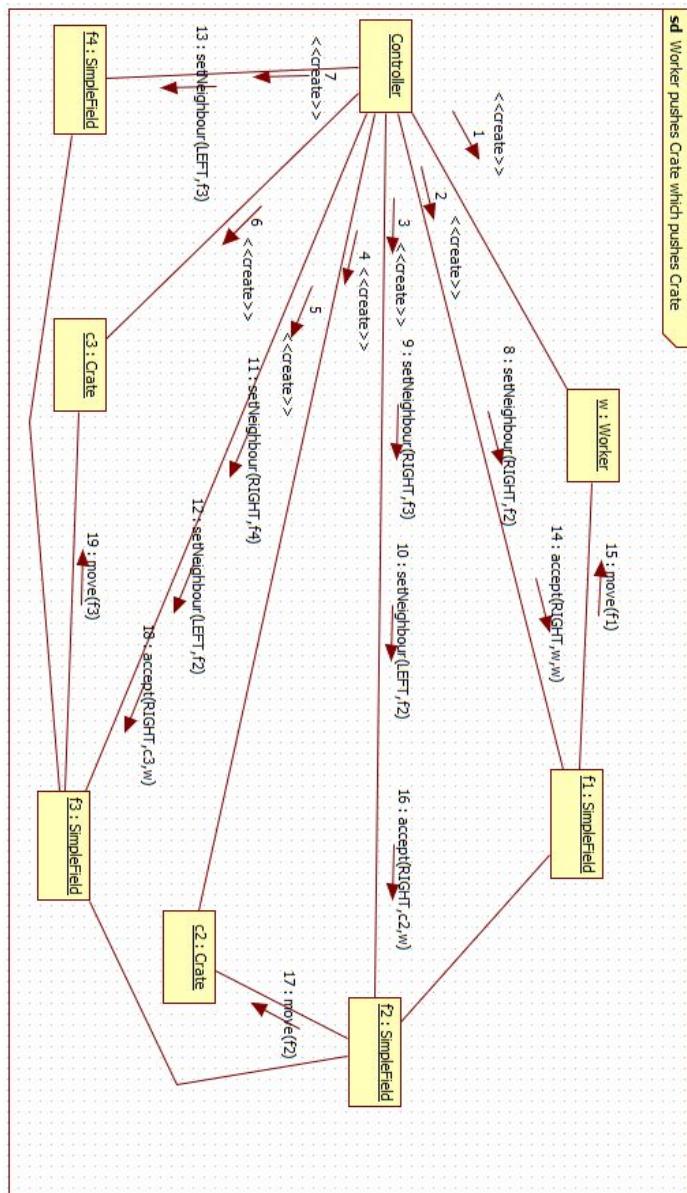
5.4.8 Worker pushes Worker



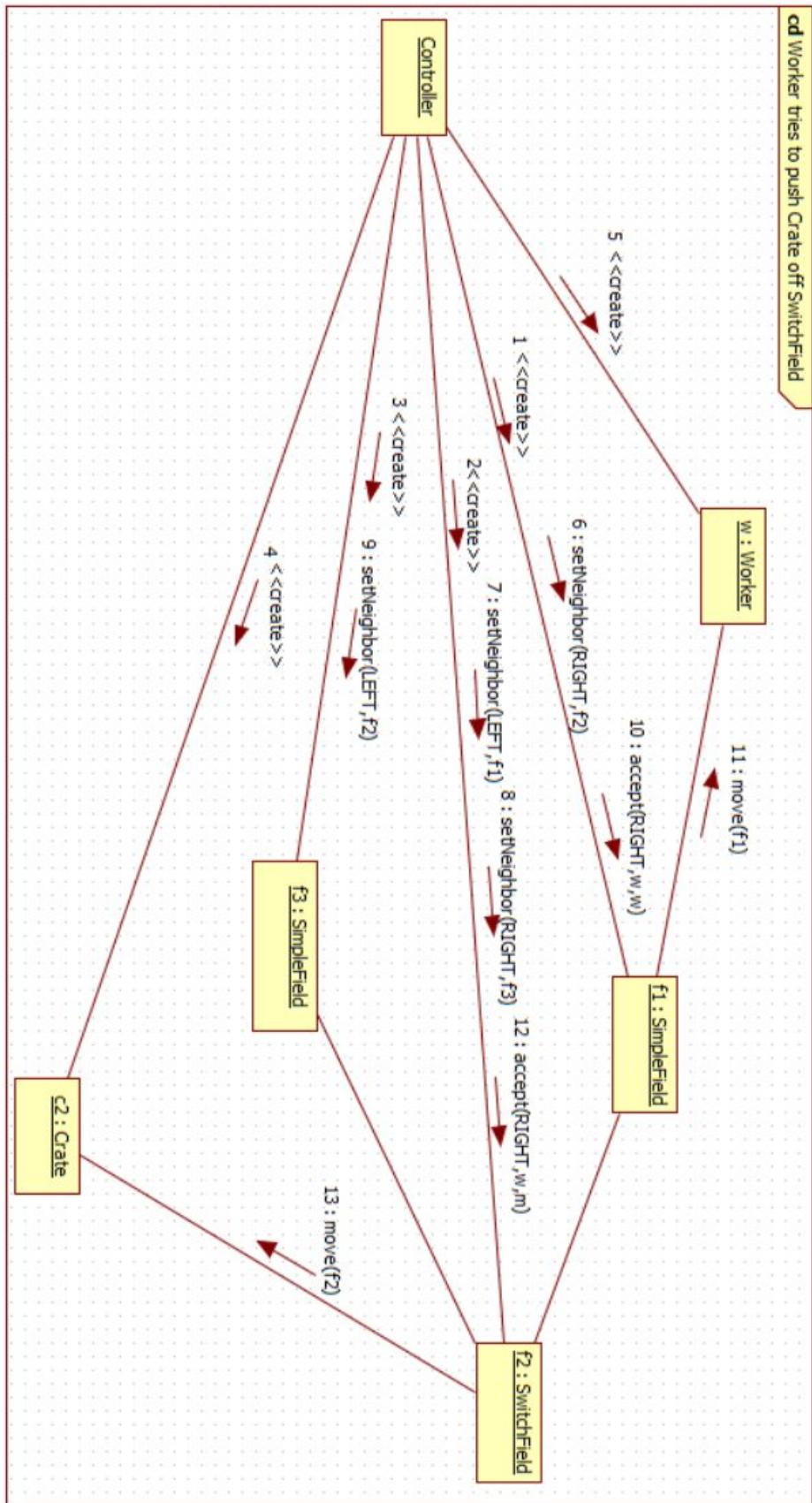
5.4.9 Worker pushes Crate which pushes Worker



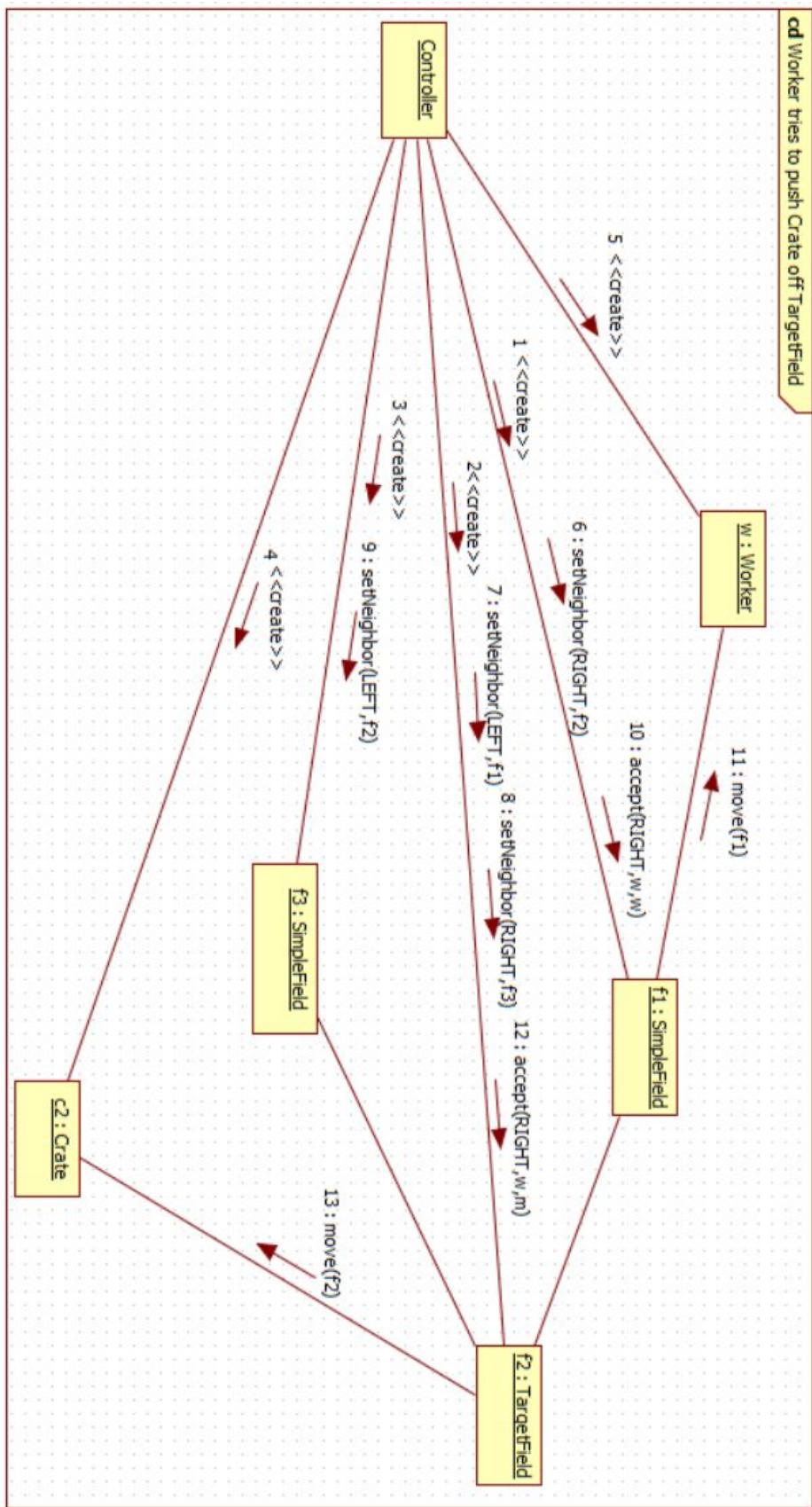
5.4.10 Worker pushes Crate which pushes Crate



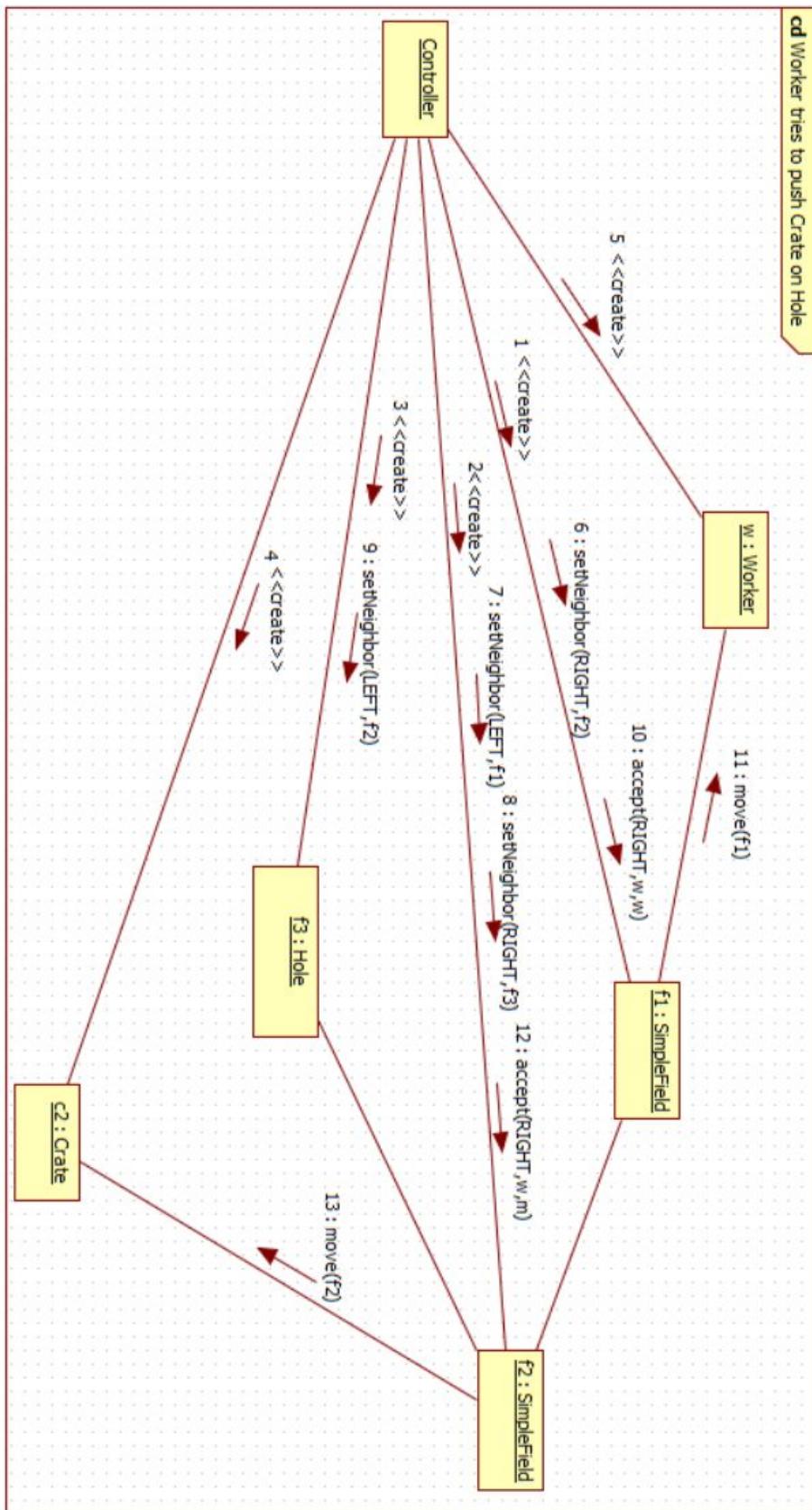
5.4.11 Worker pushes Crate off SwitchField



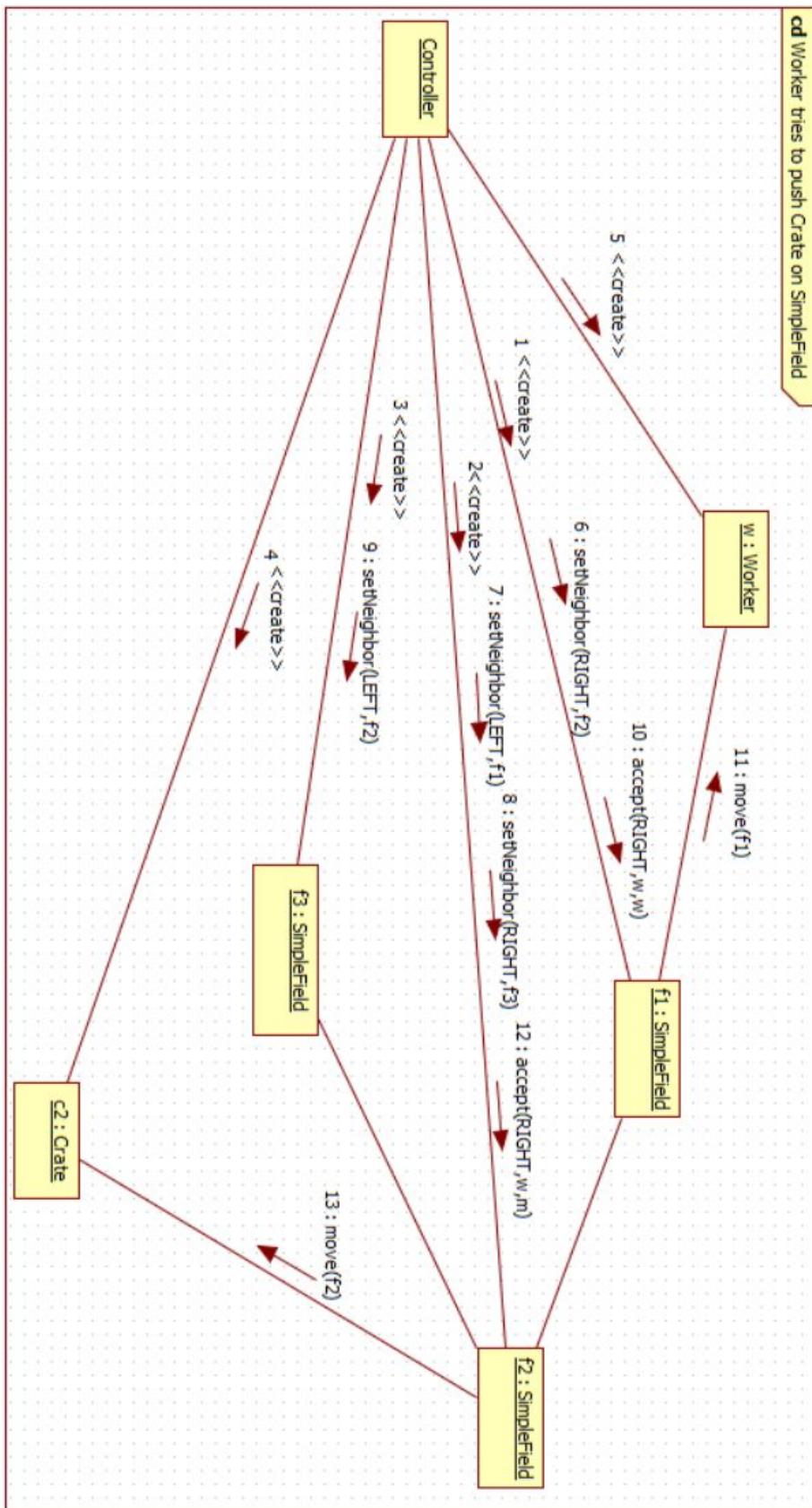
5.4.12 *Worker pushes Crate off TargetField*



5.4.13 Worker tries to push Crate on Hole

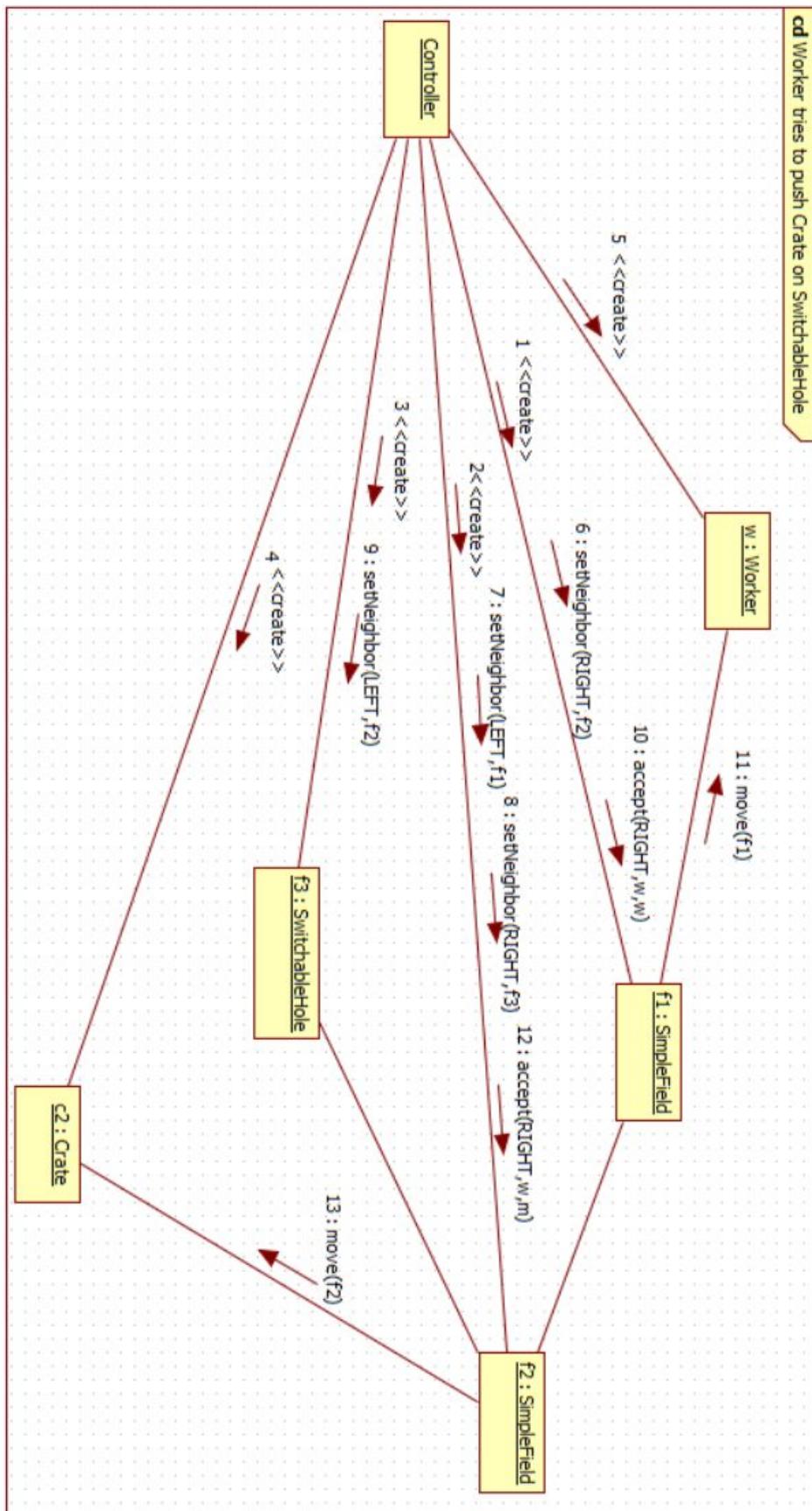


5.4.14 Worker tries to push Crate on SimpleField



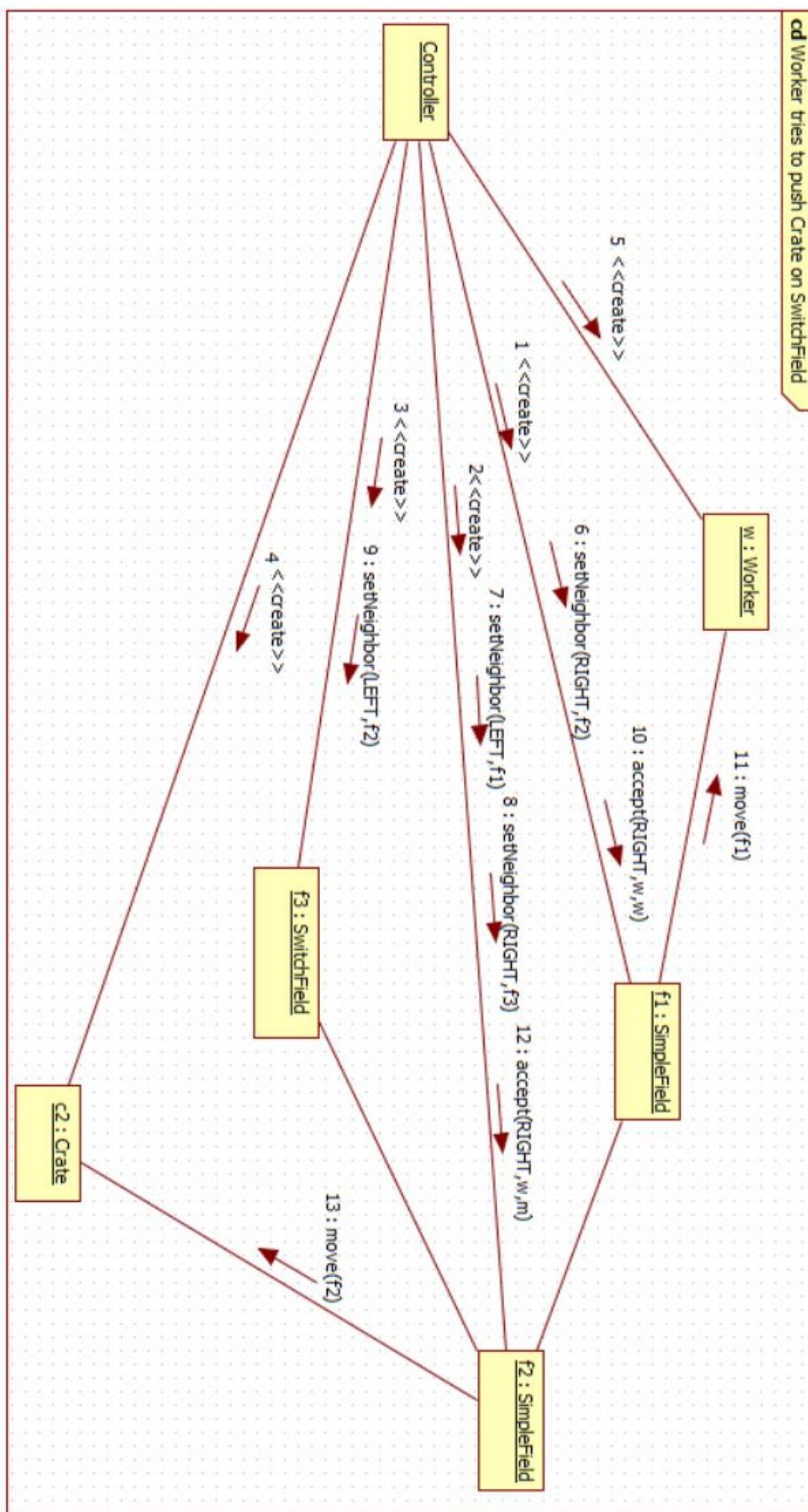
5.4.15 *Worker tries to push Crate on SwitchableHole*

cd Worker tries to push Crate on SwitchableHole



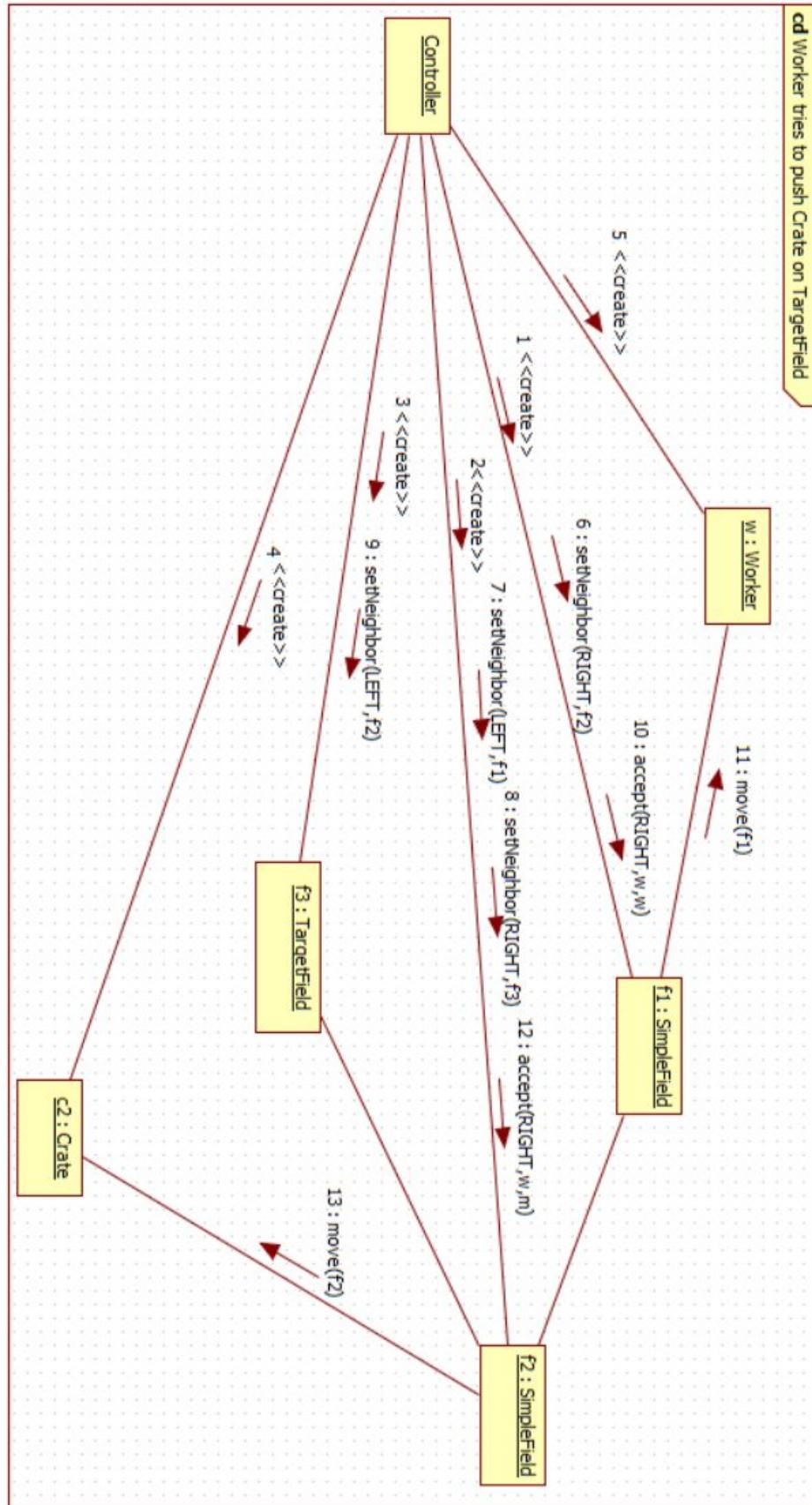
5.4.16 *Worker tries to push Crate on SwitchField*

cd Worker tries to push Crate on SwitchField



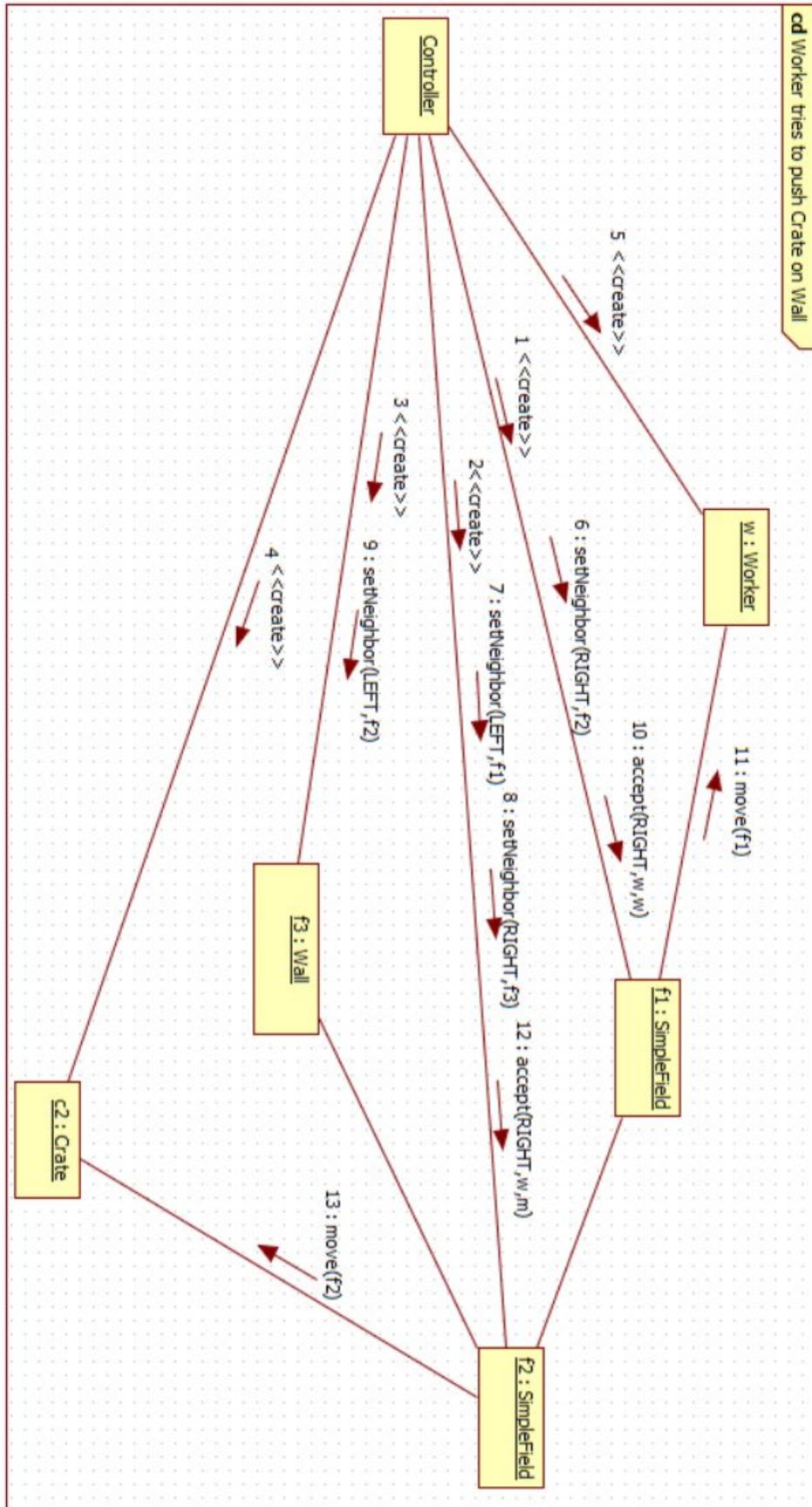
5.4.17 *Worker tries to push Crate on TargetField*

cd Worker tries to push Crate on TargetField

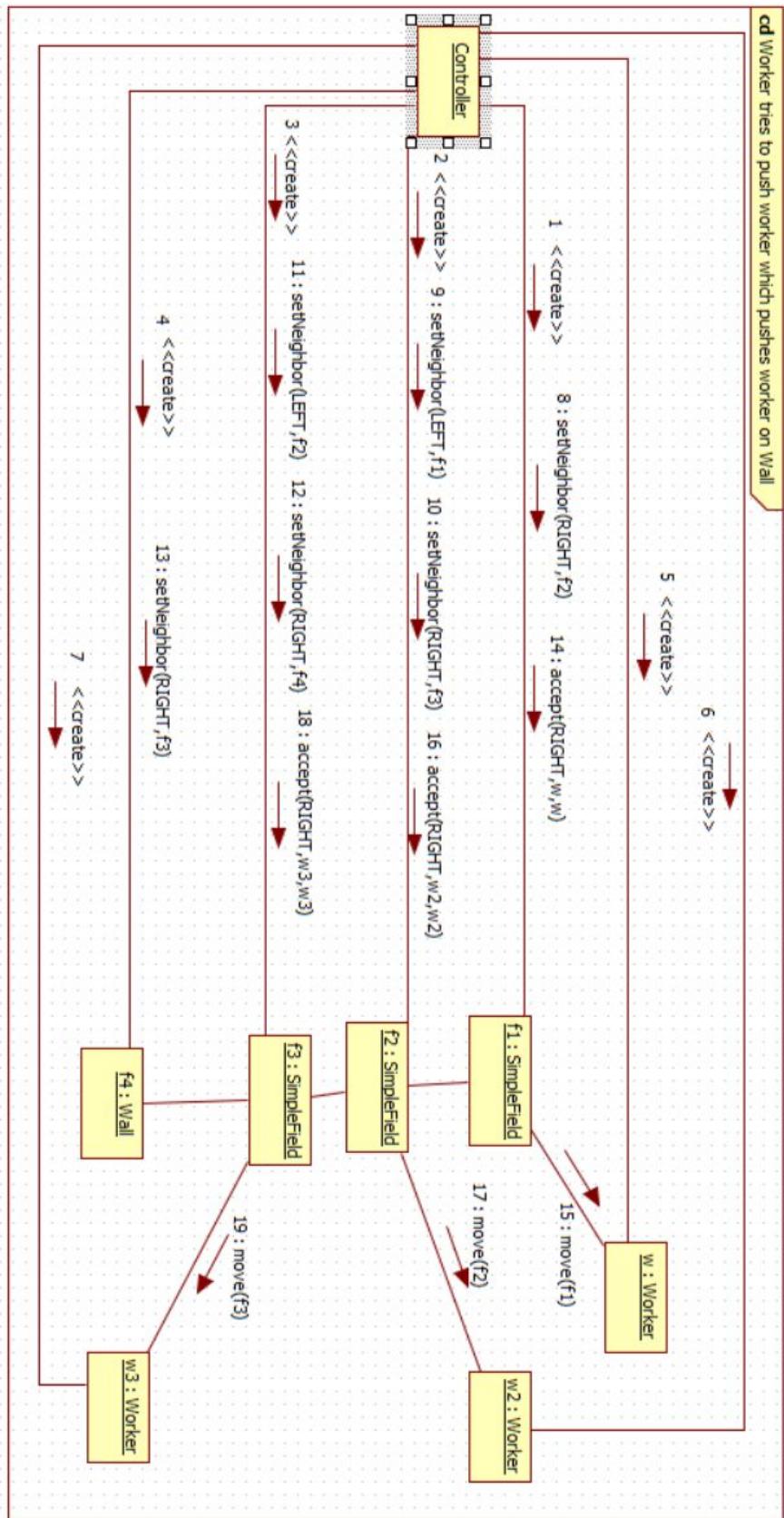


5.4.18 *Worker tries to push Crate on Wall*

cd Worker tries to push Crate on Wall



5.4.19 Worker tries to push worker which pushes worker on Wall



5.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018. 03. 07. 13:20	20 perc	csapat	Értekezlet; Döntés: feladatok kiosztása 5.1: Schulcz 5.2: Benkő 5.3: Zalavári, Benkő 5.4: Takács
2018. 03. 08. 14:15	30 perc	Schulcz	Szükséges use-case-k kitalálása, use-case diagram megrajzolása
2018. 03. 09 8:45	30 perc	Benkő	A kezelői felület érvényes in- és outputjainak specifikálása
2018. 03. 09 18:00	1.5 óra	Zalavári	5.3
2018. 03. 10. 14:30	30 perc	Schulcz	Apró hibajavítás a use-case diagramban, use-case-k kifejtése
2018. 03. 09 15:00	2.5 óra	Zalavári	5.3
2018. 03. 10 16:00	2 óra	Benkő	Kommunikációs diagramok rajzolása
2018.03.10. 17:30	2.5 óra	Takács	Kommunikációs diagramok rajzolása
2018. 03. 10 18:30	1 óra	Benkő	Kommunikációs diagramok rajzolása, formázása, beszúrása
2018. 03. 11. 10:30	0.5 óra	Zalavári	Szekvenciadiagram mok rajzolása. Megjegyzések írása a dokumentumba.
2018. 03. 11. 18:00	0.5 óra	Zalavári	Szekvenciadiagram mok. beszúrása
2018.03.11.19:30	10 perc	Takács	Kommunikációs diagramok beszúrása

2018.03.11.19:10	20 perc	Benkő	Kommunikációs diagramok javítása, beszúrása
------------------	---------	-------	---

6. Szkeleton beadás

Megjegyzés: A különféle mezők accept() függvényei a Don't Repeat Yourself elv miatt meghívják az ősök accept() függvényeit is. Ezt a kiírásnál Mezőtípus::accept()-tel jelöltük, és természetesen külön jelöljük az osztály és őse függvényét is.

6.1 Fordítási és futtatási útmutató

6.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
AbstractField.java	2 KB	2018.03. 14. 21:32	A mezők ősosztálya
Crate.java	2 KB	2018.03. 14. 21:32	A láda osztály
Direction.java	1 KB	2018.03. 14. 21:32	Az irányok enum-ja
Hole.java	1 KB	2018.03. 14. 21:32	A lyukak osztály
Logger.java	2 KB	2018.03. 14. 21:32	A logger osztály
Main.java	15 KB	2018.03. 14. 21:32	A szimulációt irányító osztály
Moveable.java	4 KB	2018.03. 14. 21:32	A mobok ősosztálya
SimpleField.java	4 KB	2018.03. 14. 21:32	Az egyszerű mező osztály
SwitchableHole.java	3 KB	2018.03. 14. 21:32	A kapcsolható lyuk osztály
SwitchField.java	2 KB	2018.03. 14. 21:32	A kapcsoló osztály
TargetField.java	2 KB	2018.03. 14. 21:32	A célmező osztály
Wall.java	2 KB	2018.03. 14. 21:32	A fal osztály
Worker.java	2 KB	2018.03. 14. 21:32	A munkások osztály

6.1.2 Fordítás

1. módszer

1. Navigálunk valamilyen parancsossal (bash, PowerShell, stb.) a kitömörített fájlok tartalmazó mappába
2. Miután meggyőződtünk róla, hogy a JDK bin mappája hozzá van adva a PATH-hoz, adjuk ki a következő parancsokat (Windows alatt:)

```
mkdir .\class
```

```
javac -d .\class .\AbstractField.java .\Crate.java  
.\\Direc  
tion.java .\\Hole.java .\\SwitchableHole.java .\\Logger.java  
.\\Main.java .\\Moveable.java .\\SimpleField.java  
.\\SwitchField.j  
ava .\\TargetField.java .\\Wall.java .\\Worker.java
```

(Linux alatt értelemszerűen ugyanez, csak “\” helyett “/” karakterekkel.)

2. módszer

1. Importáljuk a forrásfájlokat tetszőleges Java IDE-be, és hozzunk létre belőlük egy projektet
2. Fordítsuk az IDE beépített szolgáltatásával.

6.1.3 Futtatás

Windows alatt adjuk ki a következő parancsot:

```
java -cp .\class killersokoban1.Main
```

(Linux alatt értelemszerűen ugyanez, csak “\” helyett “/” karakterekkel.)

6.2 Értékelés

Tag neve	Munka százalékban
Benkő	24
Schulcz	25
Takács	24
Zalavári	27

6.3 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018. 03. 14.		Teljes csapat	Értekezlet, feladatok kiosztása. Döntés: Zalavári: szkeleton alapjainak megírása Schulcz: Logger osztály megírása Benkő: SwitchableHole beolvasásának elkészítése Takács: Függvényhívások Kommentelés: teljes csapat 6.1: Schulcz
2018. 03. 14. 18:00	3 óra	Zalavári	Szkeleton szkeletonjának elkészítése
2018. 03. 15. 8:30	2 óra	Schulcz	Logger osztály, logolás beleírása

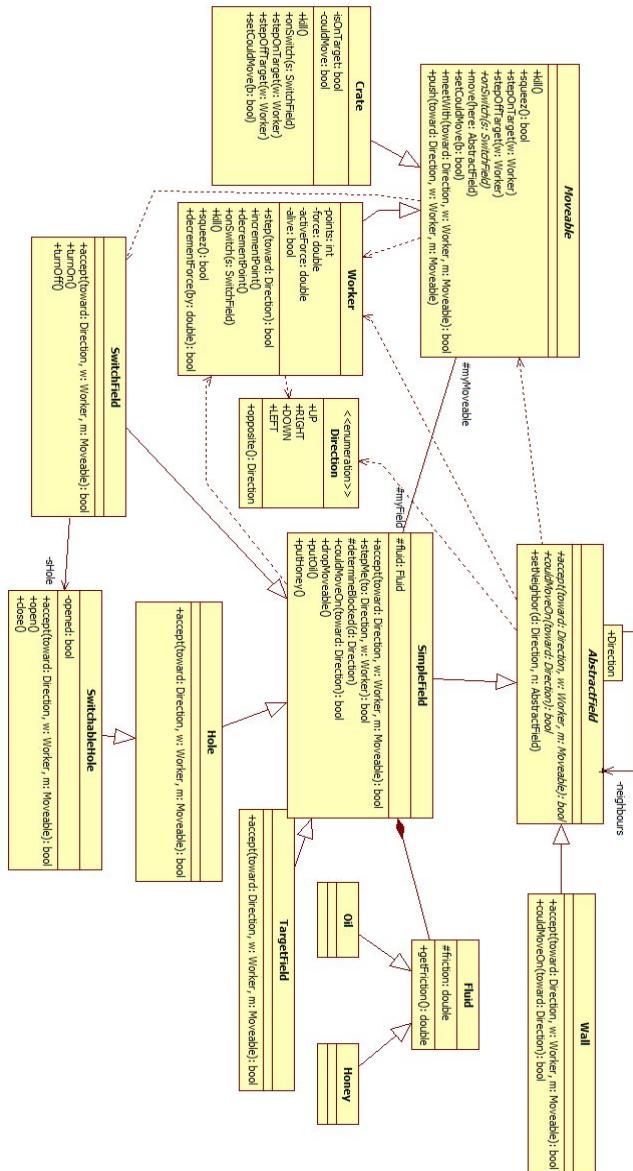
			minden meglévő függvénybe
2018. 03. 15. 10:00	1 óra	Benkő	„Főmenü” elkészítése
2018. 03.15. 13:00	15 perc	Takács	Függvényhívások
2018. 03. 15. 17:00	30 perc	Benkő	Felhasználói döntésbekérés elkészítése a SwitchableHole osztályban, szcenáriók inicializálása
2018. 03.16. 16:00	20 perc	Takács	JavaDoc
2018. 03. 16. 17:00	30 perc	Benkő	Szcenáriók inicializálása
2018.03.17. 11:00	1 óra	Takács	JavaDoc
2018. 03. 17. 18:30	30 perc	Benkő	Szcenáriók inicializálása
2017. 03. 18. 10:00	2 óra	Zalavári	Kód ellenőrzése, javítások a kódban és a kommentekben
2018. 03. 18. 13:00	1 óra	Schulcz	Elkészült kód ellenőrzése, bugfixek
2018. 03. 18. 18:00	15 perc	Schulcz	Jelen dokumentum megírása
2018. 03. 18. 21:00	2 óra	Teljes csapat	Értekezlet; értékelés megbeszélése, szemantikai ellenőrzések és véglegesítések, további tervekről való ötletelés

7. Prototípus koncepciója

7.0 Változás hatása a modellre

7.0.1 Módosult osztálydiagram

Megjegyzés: a modellező eszköz nem tud inner class-t ábrázolni, így azt kompozícióként jelöltük.



7.0.2 Új vagy megváltozó metódusok

SimpleField

`void putHoney()`: mézessé teszi a mező felszínét, függetlenül attól, hogy addig milyen volt.
`void putOil()`: olajossá teszi a mező felszínét, függetlenül attól, hogy addig milyen volt.

bool stepMe(Direction toward, Worker w): levonja a toló munkás a még éppen kifejthető erejéből azt a mennyiséget, ami ahoz kell, hogy továbbléptesse a mezőn álló dolgot, és ha ez nem lett negatív, akkor megpróbálja továbbléptetni. A léptetés sikerességével tér vissza.

Worker

bool decrementForce(double by): a munkás az aktuálisan még kifejthető erejéből levonja a paraméterül kapott értéket, és igazzal tér vissza, ha ez nem lett negatív.

Új/módosított osztályok:

Fluid

Azon folyadékok ősosztálya, amik a mezőre kerülhetnek. Alkalmazható egy placeholderként is, ha ez van a mezőn, úgy viselkedik, mintha nem lenne ott semmi (csak az alap súrlódás.)
double getFriction(): a folyadék által okozott súrlódást (tolóerő-csökkentést) adja vissza.

Honey

A fluid leszármazottja, magasabb súrlódással.

Oil

A fluid leszármazottja, alacsonyabb súrlódással.

SwitchableHole

Az osztály már nem a Hole osztályból öröklödik, hanem a SimpleField-ből. Ez a döntés a feladat meg változásától függetlenül született, refaktorálás okán.

Megj.: Bár a Honey és az Oil látszólag üres leszármazottak (csak a konstruktorkat definiálják felül), és a modellben talán valóban nem lenne szükség ezen osztályok között ilyen nagy különbséget tenni, azonban ezt a bővíthetőség miatt mégis megelőlegeztük. Például hasznos lehet, ha máshogyan akarjuk majd kirajzolni őket a grafikus verzióinál.

7.0.3 Szekvencia-diagramok

Munkás-láda-láda

Megj.: a két nagy diagram valamivel jobb felbontásban elérhető itt:

<https://drive.google.com/file/d/1Isn2jKTqoZR6cWK4ubCD94D9rYVrESq2/view?usp=sharing>

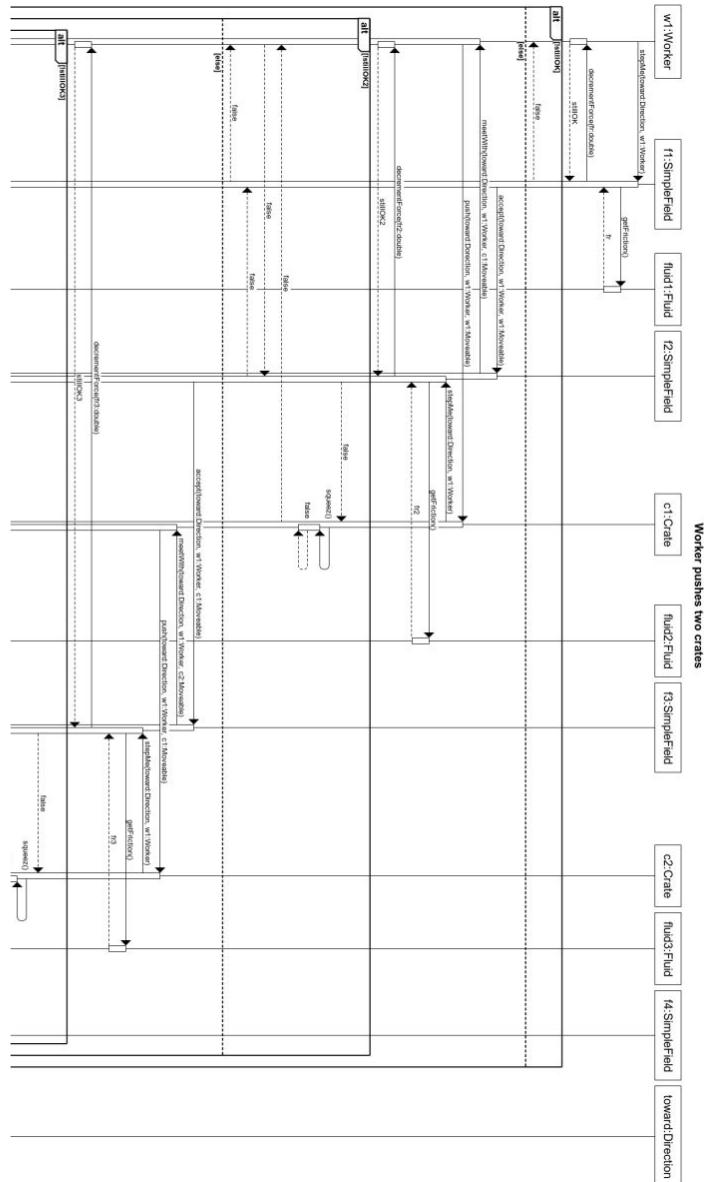
rövidítve: <https://goo.gl/EnpJcQ>

<https://drive.google.com/file/d/1giofOl9EytALGaOf3yb2i3Ux7m4-3LIP/view?usp=sharing>

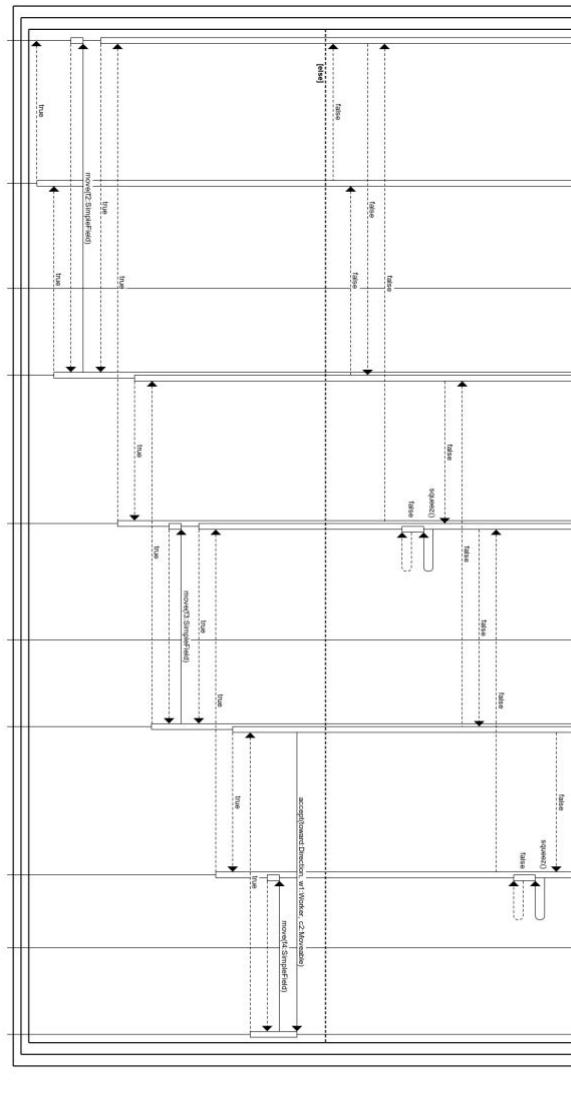
rövidítve: <https://goo.gl/DruUSV>

Megj.: mivel a kért diagramok nagyon nagyméretűek, az egyszerűbb megértés kedvéért elkészítettük a munkás egyetlen látását tol szekvenciát is. Ez a két kötelező után található.

1. rész

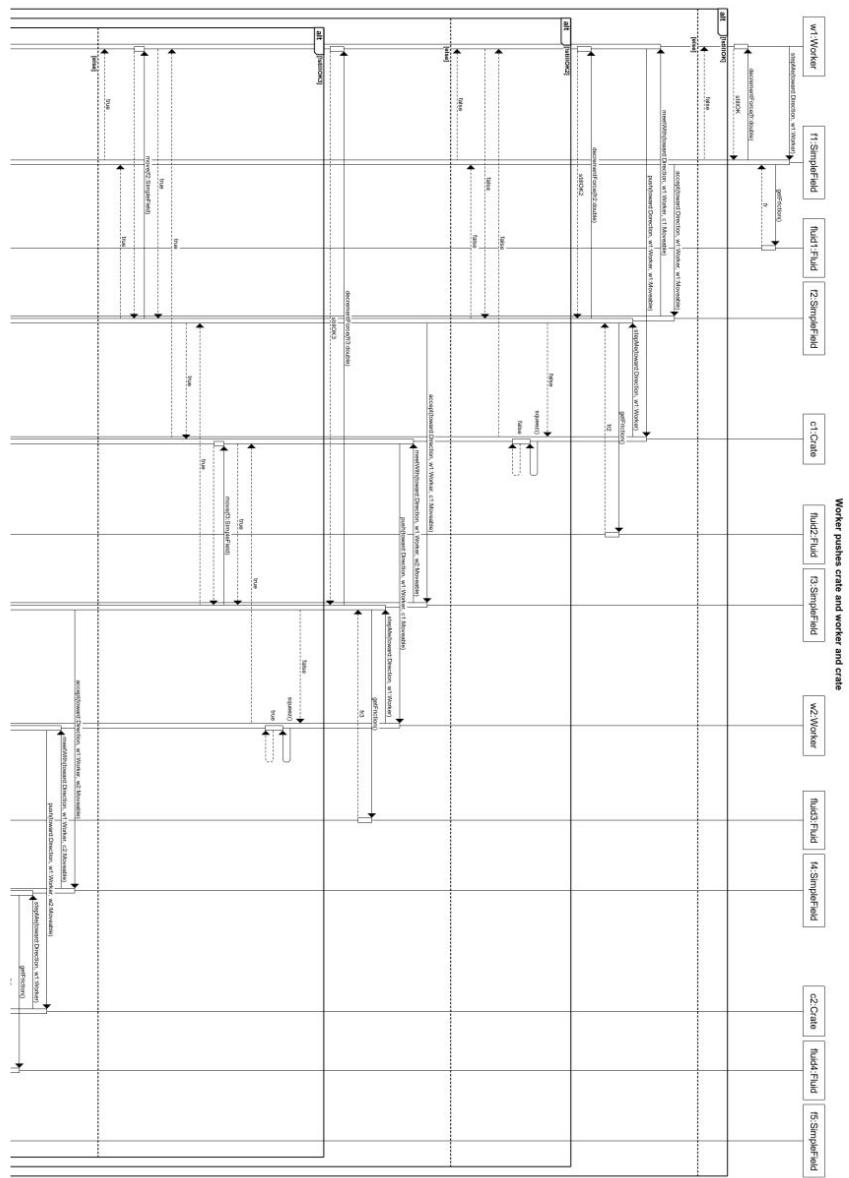


2. rész

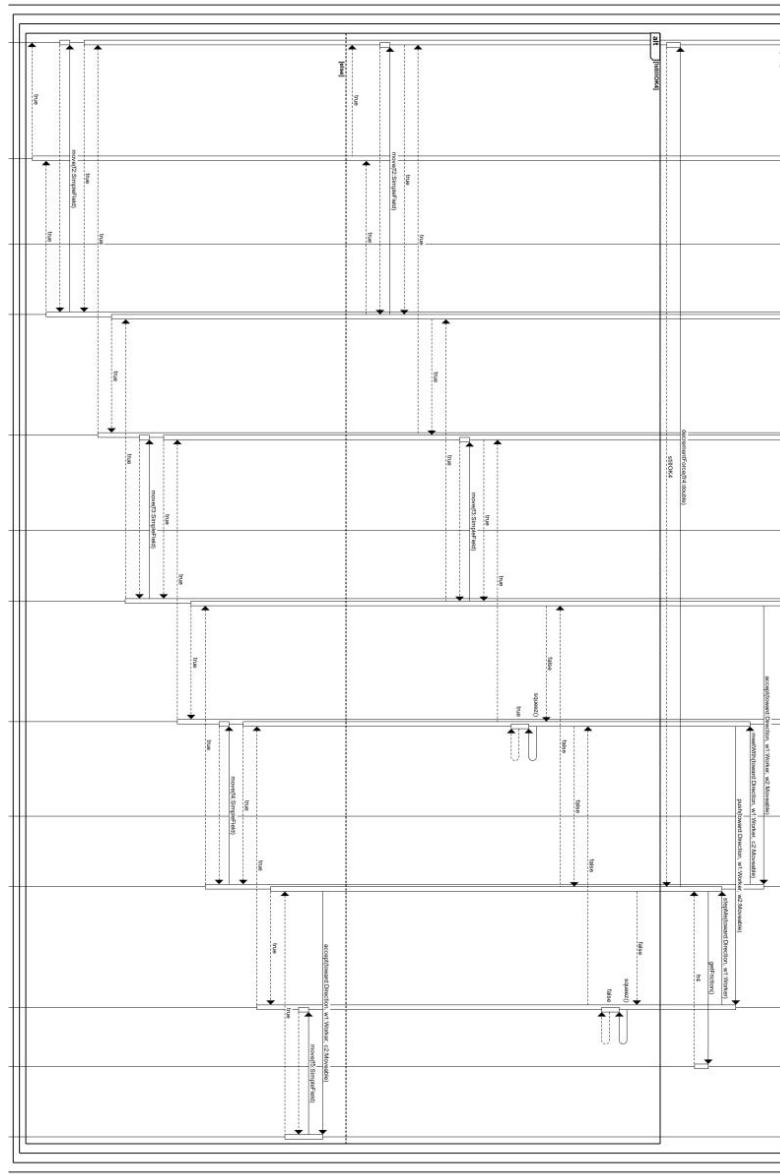


Munkás-láda-munkás-láda

1. rész (Megj: a toward:Direction direkt maradt ki, úgy már tényleg nem lett volna olvasható.)

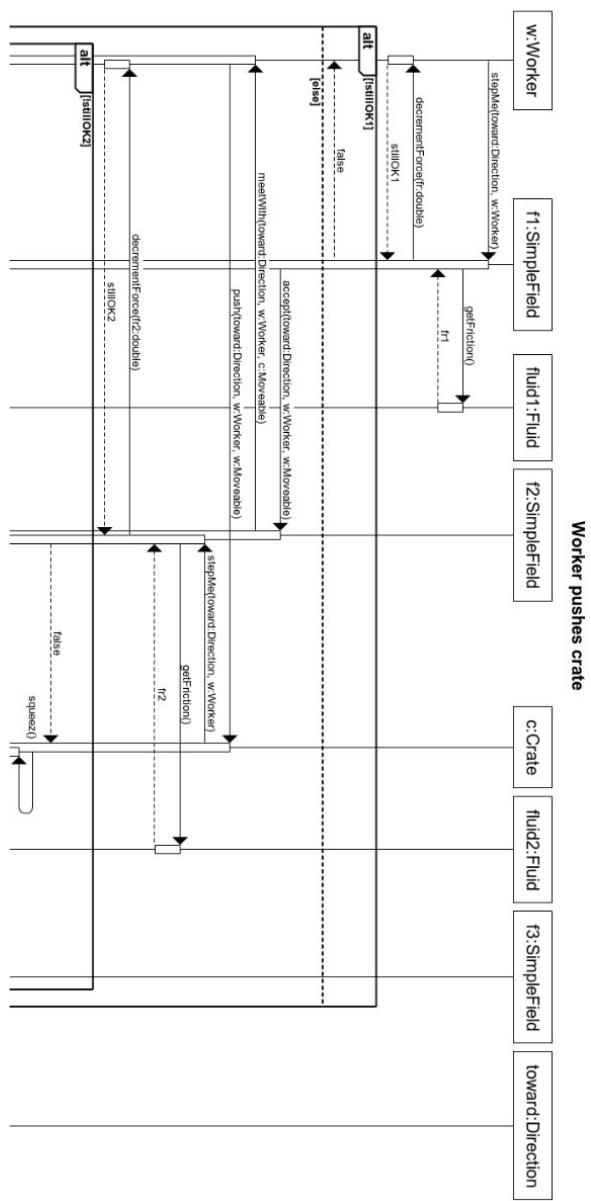


2. rész

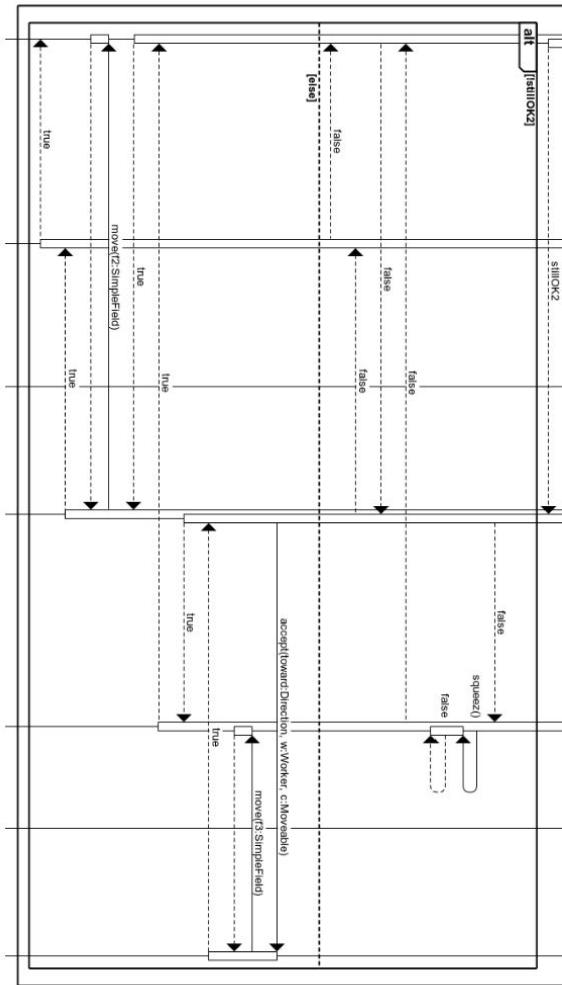


Munkás-láda (kötelezőn felüli)

1. rész



2. rész



7.1 Prototípus interface-definíciója

7.1.1 Az interfész általános leírása

A protó program működése a parancssoron keresztül vezérelhető. Ezt megkönnyítendő, a program képes lesz előre elkészített, utasításokat tartalmazó fájlok alapján is működni, batch jelleggel.

Megj.: A mezők létrehozását és mezők szomszédságának egyéni beállítását nem támogatjuk a bemeneti nyelvben. Erről hosszú értekezlet folyt, végül azért döntöttünk így, mert a modellünk esetleg kihasználja, hogy a pályáról nem lehet lemenni. Tehát legalább fallal, vagy lyukakkal körül van véve. Amennyiben ez nem teljesülne, könnyen előállítható lenne olyan pálya, ahol hamar NullPointerException-t kapnánk.

Továbbá nem támogatjuk a lyukak egyéni kapcsolását sem, ez csak egy hozzákötött kapcsolóval oldható meg.

7.1.2 Bemeneti nyelv

Parsemap

Leírás: A paraméterben megadott sor- és oszlopszámú, ill.elrendezésű pálya mezőit létrehozza és összeköti szomszédaikkal. A paraméter szintaxisát ld. lentebb.

Opciók: parsemap <sorok száma> <oszlokok száma>

```
<f_1_1><f_1_2>...<f_1_m>
<f_2_1><f_2_2>...<f_2_m>
..... .
<f_n_1><f_n_2><f_n_m>
```

pl.: parsemap 4 3

```
www
wmw
wmw
www
```

A sorok közben nincs szóköz, a sor végén enter van

A kívánt pálya felépítését egy vele megegyező struktúrájú rendezett karakterhalmaz írja majd le, mind a konzolra íráskor, mind a fájlból olvasáskor. Ennek felépítését az alábbi sorok szemléltetik:

```
wwwww
wmhmw
wkmmw
wsmtw
wwwww
```

Ahol az egyes betűk jelentése: w: Wall, m: SimpleField, h: Hole, k: SwitchField, s: SwitchableHole, t:TargetField.

Worker

Leírás: Létrehoz egy Worker-t a paraméterben megadott névvel, és a paraméterben megadott mezőn elhelyezi

Opciók: worker <név> <erő> <célmező>

pl.: worker w1 5 f_2_3

Crate

Leírás: Létrehoz egy Crate-t a paraméterben megadott névvel, és a paraméterben megadott mezőn elhelyezi

Opciók: crate <név> <célmező>

pl.: crate c1 f_3_2

Setswitch

Leírás: Összeköt egy már létrehozott SwitchField-et egy már létrehozott SwitchableHole-lal. A SwitchableHole ettől kezdve az adott SwitchField állapotának függvényében lesz nyitva vagy csukva.

Opciók: setswitch <lyuk neve> <kapcsoló neve>

pl.: setswitch sh_4_2 sf_3_2

ahol sh_4_2: SwitchableHole és sf_3_2: SwitchField

Putfluid

Leírás: A paraméterben megadott mező a paraméterben megadott típusú folyadékot helyez el a mezőn, amint áll.

Opciók: putfluid <mező> <folyadék>

pl.: putfluid f_2_2 o

//olaj elhelyezése az f_2_2 mezőre

pl.: putfluid f_2_2 h

//mész elhelyezése az f_2_2 mezőre

//A második paraméter értéke csak ,o' vagy ,h' lehet.

Step

Leírás: A paraméterben megadott munkást a paraméterben megadott irányba lépteti.

Opciók: step <munkás> <irány>

pl.: step w1 right

Script

Leírás: A paraméterben megkapott, a bemeneti nyelv kifejezéseit tartalmazó fájlból beolvásott utasításokat hajtja végre.

Opciók: script <forrásfájl>

pl.: script eztcsinald.txt

Stat

Leírás: A paraméterben kapott entitás tulajdonságait írja a képernyőre.

Opciók: stat <entitásnév>

pl.: stat w1

stat f_8_5

7.1.3 Kimeneti nyelv

A stat parancsot kivéve minden parancs csak hiba esetén ír valamit a kimenetre, az a valami pedig egy tömör hibaüzenet.

A stat parancs az entitás adattagjait írja ki, az alábbi formában:

tulajdonság1: érték

tulajdonság2: érték

...

Alesetek:

SimpleField:

```
neighborUp: f_0_3
neighborRight: f_1_4
neighborDown: f_2_3
neighborLeft: f_1_2
moveable: null
fluid: honey
```

SwitchField:

```
neighborUp: f_0_3
neighborRight: f_1_4
neighborDown: f_2_3
neighborLeft: f_1_2
moveable: null
fluid: honey
sHole: f_4_5
```

TargetField:

```
neighborUp: f_0_3
neighborRight: f_1_4
neighborDown: f_2_3
neighborLeft: f_1_2
moveable: null
fluid: honey
```

SwitchableHole:

```
neighborUp: f_0_3
neighborRight: f_1_4
neighborDown: f_2_3
neighborLeft: f_1_2
moveable: null
fluid: honey
opened: false
```

Hole:

```
neighborUp: f_0_3
neighborRight: f_1_4
neighborDown: f_2_3
neighborLeft: f_1_2
fluid: honey
```

Wall:

```
neighborUp: f_0_3
neighborRight: f_1_4
neighborDown: f_2_3
neighborLeft: f_1_2
fluid: honey
```

Crate:

```

myField: f_0_3
couldMove: true
isOnTarget: false

```

Worker:

```

myField: f_0_3
force: 3
points: 2

```

7.2 Összes részletes use-case

Use-case neve	Step
Rövid leírás	Az egyik munkás léptetése
Aktorok	Player
Forgatókönyv	<ol style="list-style-type: none"> A munkás lép a kiválasztott irányba Elvégezzük a lépés következetében végbemenő tolásokat.

Use-case neve	Put Fluid
Rövid leírás	Olajat vagy mézet helyez el a kiválasztott mezőn.
Aktorok	Controller
Forgatókönyv	<ol style="list-style-type: none"> Folyadék létrehozása A folyadék elhelyezése a mezőn

Use-case neve	Set Switch
Rövid leírás	Kapcsoló és kapcsolható lyuk összekötése
Aktorok	Controller
Forgatókönyv	<ol style="list-style-type: none"> A mezők összekötése

Use-case neve	Create Crate
Rövid leírás	Láda letétele adott mezőre
Aktorok	Controller
Forgatókönyv	<ol style="list-style-type: none"> Láda létrehozása A láda elhelyezése a mezőn

Use-case neve	Create Worker
Rövid leírás	Munkás letétele adott mezőre
Aktorok	Controller
Forgatókönyv	<ol style="list-style-type: none"> Munkás létrehozása A munkás elhelyezése a mezőn

Use-case neve	Parse Map
Rövid leírás	A megadott pálya létrehozása
Aktorok	Controller
Forgatókönyv	<ol style="list-style-type: none"> 1. A mezők létrehozása 2. A mezők szomszédsági viszonyainak beállítása

Use-case neve	See stat
Rövid leírás	Információ megtekintése az adott entitásról
Aktorok	Controller, Player
Forgatókönyv	<ol style="list-style-type: none"> 1. Információ kiírása 2. A játékos megnézi az információt

Use-case neve	Run script
Rövid leírás	A program lefuttat egy szkriptet
Aktorok	Controller
Forgatókönyv	<ol style="list-style-type: none"> 1. A program lefuttatja a szkriptet

7.3 Tesztelési terv

Az első 22 (1-től 20-ig) teszesetben feltételezzük, hogy a munkásnak elég ereje van a tevékenységek végrehajtására, így ez nem lehet okozója a mozgások meghiúsulásának.

Teszt-eset neve	1.Worker steps on Wall
Rövid leírás	A munkás egy falra próbál lépni.
Teszt célja	A munkás lépése meghiúsul. (Nem tud falra lépni.)

Teszt-eset neve	2. Worker steps on SimpleField
Rövid leírás	Lemodelezük azt a folyamatot, ahogy egy munkás egy szomszédos, üres SimpleField típusú mezőre lép.
Teszt célja	A munkás sima mezőre lép.

Teszt-eset neve	3. Worker steps on TargetField
Rövid leírás	Lemodelezük azt a folyamatot, ahogy egy munkás egy szomszédos, üres TargetField típusú mezőre lép.
Teszt célja	A munkás célmezőre lép.

Teszt-eset neve	4. Worker steps on SwitchField
Rövid leírás	Lemodelezük azt a folyamatot, ahogy egy munkás egy szomszédos, üres SwitchField típusú mezőre lép.

Teszt célja	A munkás kapcsoló típusú mezőre lép.
Teszt-eset neve	5. Worker steps on Hole
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy szomszédos lyukra lép és meghal.
Teszt célja	A munkás lyukra lép, és meghal.
Teszt-eset neve	6. a Worker steps on opened SwitchableHole
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy szomszédos kapcsolható lyukra lép és meghal.
Teszt célja	A munkás kapcsolható lyukra lép és meghal.
Teszt-eset neve	6. b Worker steps on closed SwitchableHole
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy szomszédos kapcsolható lyukra lép.
Teszt célja	A munkás kapcsolható lyukra lép.
Teszt-eset neve	7. Worker tries to push crate on Wall
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás falra akar tolni egy lánát.
Teszt célja	A munkásnak nem sikerül eltolni a lánát.
Teszt-eset neve	8. Worker tries to push crate on SimpleField
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy lánát egy azzal szomszédos, üres SimpleField típusú mezőre tol.
Teszt célja	A munkás sima mezőre tol egy lánát.
Teszt-eset neve	9. Worker tries to push crate on TargetField
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy lánát egy azzal szomszédos, üres TargetField típusú mezőre tol.
Teszt célja	A munkás célmezőre tol egy lánát.
Teszt-eset neve	10. Worker tries to push crate on SwitchField
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy lánát egy azzal szomszédos, üres SwitchField típusú mezőre tol.
Teszt célja	A munkás kapcsoló típusú mezőre tol egy lánát.
Teszt-eset neve	11. Worker tries to push crate on Hole
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy lánát egy azzal szomszédos Hole típusú mezőre tol.
Teszt célja	A munkás lyukba tol egy lánát, és az eltűnik.
Teszt-eset neve	12. a Worker tries to push crate on opened SwitchableHole

Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy látát egy azzal szomszédos, üres SwitchableHole típusú mezőre tol, ami nyitva volt, így a láda eltűnik.
Teszt célja	A munkás kapcsolható lyukra tolja a látát, és az eltűnik.

Teszt-eset neve	12. b Worker tries to push crate on closed SwitchableHole
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás egy látát egy azzal szomszédos, üres SwitchableHole típusú mezőre tol, ami zárvá volt.
Teszt célja	A munkás kapcsolható lyukra tolja a látát.

Teszt-eset neve	13. Worker pushes worker
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy SimpleField-ekből álló pályán egy munkás megtol egy másikat.
Teszt célja	A munkás megtol egy másik munkást.

Teszt-eset neve	14. Worker pushes crate which pushes worker
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás által tolta láda megtol egy másik munkást.
Teszt célja	A munkás megtol egy látát, ami megtol egy másik munkást.

Teszt-eset neve	15. Worker pushes crate which pushes crate
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás által tolta láda megtol egy másik látát.
Teszt célja	A munkás megtol egy látát, ami megtol egy másik látát.

Teszt-eset neve	16. Worker pushes crate off TargetField
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás letol egy látát egy célmezőről.
Teszt célja	A munkás letol egy látát a célmezőről.

Teszt-eset neve	17. Worker pushes crate off SwitchField
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás letol egy látát egy kapcsolóról, ezzel kikapcsolva azt.
Teszt célja	A munkás letol egy látát a kapcsoló típusú mezőről.

Teszt-eset neve	18. Worker pushes worker into wall
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás falnak tol egy másikat, ezzel összenyomva őt.
Teszt célja	A munkás falnak tol egy másik munkást, és az meghal.

Teszt-eset neve	19. Worker tries to push worker which pushes worker on Wall
------------------------	---

Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás megtol egy másikat, ami megtol egy harmadikat, ezzel összenyomva őt.
Teszt célja	A munkás megtol egy másik munkást, ami egy harmadikat a falnak tol. A falnak tolta munkás meghal.

Teszt-eset neve	20. Worker pushes crate on SwitchableHole, and opens.
Rövid leírás	Lemodellezük azt a folyamatot, ahogy egy munkás rátol egy lántható lyukra, majd kinyitja a lyukat.
Teszt célja	A munkás rátol egy lántható lyukra, majd kinyitja a lyukat. A lántható lyuk elérhető lesz.

Teszt-eset neve	21. Worker puts Honey on SimpleField
Rövid leírás	A játékos összekeni a mezőt mézzel.
Teszt célja	A játékos sikeresen összekente a mezőt mézzel.

Teszt-eset neve	22. Worker puts Oil on SimpleField then moves
Rövid leírás	A játékos összekeni a olajjal és odébb áll.
Teszt célja	A mező összekentsége megmarad, akkor is, ha utána odébb lép.

Teszt-eset neve	23. Worker puts Honey, then Oil on SimpleField
Rövid leírás	A játékos összekeni a mezőt előbb mézzel, majd olajjal.
Teszt célja	Annak az anyagnak kell a mezőn maradnia, amit utoljára rátettek.

Teszt-eset neve	24. Worker puts Honey on SwitchableHole, then opens and closes the Hole
Rövid leírás	A játékos összekeni egy zárható lyukat (zárt állapotában természetesen), majd bezárja, és kinyitja azt.
Teszt célja	A lyuk kinyitása majd újra bezárása nem tisztíthatja meg a mezőt, azaz annak az anyagnak kell rajta maradnia, amit utoljára rátettek.

Teszt-eset neve	25. Just enough force
Rövid leírás	A munkásnak éppen elég ereje van, hogy az előtte lévő lántható lyukat eltolja.
Teszt célja	A tolás sikeresen a munkás erőssége miatt.

Teszt-eset neve	26. Not enough force
Rövid leírás	A munkásnak nincsen elég ereje, hogy az előtte lévő lántható lyukat eltolja.
Teszt célja	A tolás meghiúsul a munkás erő hiánya miatt.

Teszt-eset neve	27. Not enough force because of Honey
------------------------	---------------------------------------

Rövid leírás	A munkásnak éppen elég ereje lenne, hogy az előtte lévő ládákat eltolja, de mivel van egy bemézezett mező, ez mégsem fog sikerülni neki.
Teszt célja	A tolás méz miatt meghiúsul.

Teszt-eset neve	28. Enough force because of Oil
Rövid leírás	A munkásnak nem lenne elég ereje, hogy az előtte lévő ládákat eltolja, de mivel van egy beolajozott mező, ez mégsem fog sikerülni neki.
Teszt célja	A tolás olaj miatt sikerül.

Teszt-eset neve	29. Enough force because more Oil than Honey
Rövid leírás	A munkásnak nem lenne elég ereje, hogy az előtte lévő ládákat eltolja, de mivel több beolajozott mező van, mint mézezett, ez mégsem fog sikerülni neki.
Teszt célja	A tolás több olaj, mint méz miatt sikerül.

Teszt-eset neve	30. Not enough force because more Honey than Oil
Rövid leírás	A munkásnak lenne elég ereje, hogy az előtte lévő ládákat eltolja, de mivel több mézezett mező van, mint olajozott, ez mégsem fog neki sikerülni.
Teszt célja	A tolás több méz, mint olaj miatt nem sikerül.

7.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása

A prototípus tesztelése legkönnyebben egy PowerShell szkript segítségével lesz elvégezhető. A szkript indítható paraméter nélkül vagy egy teszteset nevét paraméterül megadva - előbbi esetben az összes, vele egy mappában található tesztesetet lefuttatja majd.

Egy teszteset két fájlból áll: tesztnév.tst és tesztnév.exp. A TST a programra szánt bemenetet, az EXP pedig az elvárt kimenetet tartalmazza, szöveges formában.

A szkript minden tesztesetre vagy a teszt sikereségét megerősítő üzenetet fog visszaadni, vagy a bukás tényét írja ki, a valós és elvárt kimenet közti különbség megmutatásával.

7.5 Napló

Kezdet	Időtartam	Résznevők	Leírás
2018.03.22. 21:00	2 óra	Schulcz, Takács, Zalavári	Értekezlet.
2018.03.22. 23:00	1 óra	Teljes csapat	Értekezlet. Döntés: Bemenet és kimenet specifikálása, új modellbeli elemek modellbe illesztése, munka felosztása

2018.03.23. 7:00	1 óra 30 perc	Benkő	Be- és kimeneti nyelv leírásának kidolgozása I.
2018.03.23. 12:00	40 perc	Benkő	Be- és kimeneti nyelv leírásának kidolgozása II. use-case-k
2018. 03. 23. 16:00	5 óra	Schulcz	Osztálydiagram frissítése, változások implementálása, változások szöveges leírása (frissülő és új metódusok és osztályok), szekvenciadiagramok elkészítése.
2018. 03. 25. 11:00	1 óra	Schulcz	PowerShell alapjainak átnézése, tesztelést támogató program specifikálása
2018. 03. 25. 14:00	10 perc	Schulcz	Be- és kimeneti nyelvben, use-case-leírásokban módosítások
2018. 03. 25. 14:30	1 óra	Takács	Tesztesetek megírása
2018. 03. 25. 15:00	1 óra	Zalavári	Tesztesetek megírása
2018. 03. 25. 17:00	1 óra	Zalavári	Ellenőrzés, apró javítások

8. Részletes tervez

8.1. Osztályok és metódusok tervei.

8.1.1. Direction

- **Felelősség**

Egy enumeration, ami a mezőn értelmezett négy lehetséges irányt tudja tárolni. Képes továbbá azt is megmondani, hogy egy adott iránnyal melyik irány ellentétes.

- **Ősosztályok**

Nem leszármazott osztály.

- **Interfészek**

Nem valósít meg interfészt.

- **Literálok**

- **UP:** a felfelé irány
- **DOWN:** a lefelé irány
- **LEFT:** a balra irány
- **RIGHT:** a jobbra irány

- **Metódusok**

[Milyen publikus, protected és privát metódusokkal rendelkezik. Metódusonként precíz leírás, ha szükséges, activity diagram is a metódusban megvalósítandó algoritmusról. minden olyan metódusnak szerepelnie kell, amelyiket az osztály megvalósít vagy felüldefiniál.]

- **static Direction Dir(int num):** A paraméterként kapott számhoz visszaad egy Direction-t: 0 esetén LEFT, 1 esetén UP, 2 esetén RIGHT, egyébként pedig DOWN értékkel. Láthatóság: +. Statikus függvény.
- **Direction opposite():** A Direction a saját magával ellentétes irányt jelképező Direction-nel tér vissza. RIGHT esetén LEFT-tel, UP esetén DOWN-nal, és vice versa. Láthatóság: +
- **static Direction fromString(String dir):**
- **String toString():**

8.1.2. Crate

- **Felelősség**

Egy lánát jelképez a játékban, amit a munkások tologathatnak. Tolás hatására megpróbál a megfelelő mezőre lépni, de kezeli azt is, ha ez nem sikerül. Ha lyukba esik, megsemmisül, ha célmezőre kerül, pontot adat az őt oda toló játékosnak, ha onnan lekerül, elvesz egy pontot. Kapcsolóra érve aktiválja azt.

- **Ősosztályok**

Moveable

- **Interfészek**

Nem valósít meg interfészt.

- **Attribútumok**

- **boolean couldMove**: Tárolja, hogy a láda közvetlenül tolható lenne-e, azaz bármely oldalról mellé tudna-e állni egy munkás és onnan el tudná-e tolni. A játék végének (elakadásának) detektálásához használjuk. Láthatóság: -
- **boolean isOnTarget**: Tárolja, hogy a láda épp egy célmezőn áll-e. A játék végének detektálásához használjuk. Láthatóság: -

- **Metódusok**

- **void kill()**: Megsemmisíti a látát, kitörölteti magát a Main adathalamzból, és eldobatja magát az alatta lévő mezővel. Meghívja az alatta lévő mező dropMoveable() függvényét. Láthatóság: +
- **void onSwitch(SwitchField s)**: Jelzi a láda számára, hogy kapcsolóra érkezett. Meghívja a kapcsoló turnOn() függvényét. Láthatóság: +
- **void stepOnTarget(Worker w)**: Jelzi a láda számára, hogy az adott munkás egy célmazőre tolta. Meghívja a munkás incrementPoint() függvényét. Láthatóság: +
- **void stepOffTarget(Worker w)**: Jelzi a láda számára, hogy az adott munkás letolta egy célmazőről. Meghívja a munkás decrementPoint() függvényét. Láthatóság: +
- **void setCouldMove(boolean b)**: Beállítja a couldMove adattag értékét. Láthatóság: +
- **boolean gameFinishable()**: Azt adja vissza, hogy a láda szempontjából befejeződhet-e a játék, azaz hogy nem közvetlenül tolható vagy célmezőn van-e. Láthatóság: +
- **void printStat()**: meghívja az ōse printStat függvényét, majd kiírja a saját adattagjait az alábbi formában:
 - couldMove: érték
 - isOnTarget: értékahol a két érték "true" vagy "false".
Láthatóság: +

8.1.3. Fluid

- **Felelősség**

A mezőn lévő folyadékot reprezentálja a belőle leszármazó osztályok által. Önmagában azt az esetet jelenti, amikor a mezőn nincs folyadék (felfoghatjuk egy alapfolyadéknak is.) Fő szerepe, hogy súrlódással nehezíti a mezőn zajló mozgásokat.

A SimpleField által tartalmazott osztály.

- **Ősosztályok**

Nem leszármazott osztály.

- **Interfészek**

Nem valósít meg interfészt.

- **Attribútumok**
 - **double friction:** A súrlódás nagysága, értéke (ha nem definiáljuk felül) konstans 1.0. Láthatósága: #
- **Metódusok**
 - **double getFriction():** Visszaadja a súrlódás értékét. Láthatósága: +
 - **konstruktor:** Beállítja a súrlódás értékét. Láthatóság: +
 - **String getType():** Visszaadja a “no fluid” sztringet. A stat parancs miatt van. Láthatóság: +

8.1.4. Honey

- **Felelősség**
Az egyikfajta folyadékot, a mézet szimbolizálja. Feladata, hogy megnehezítse a mezőn való mozgást.
- **Ősosztályok**
Fluid
- **Interfészek**
Nem valósít meg interfészrt.
- **Metódusok**
 - **konstruktor:** Beállítja a súrlódás értékét 2-re. Láthatóság: +
 - **String getType():** Visszaadja a “honey” sztringet. A stat parancs miatt van. Láthatóság: +

8.1.5. Main

- **Felelősség**
EZ az osztály felelős a játék menetéért. Tartalmazza a game loopot, előkészíti a pályát és a felhasználói utasítások alapján mozgatja a mobokat. Megállapítja, mikor van vége a játéknak és ilyenkor lezárja azt. Statikus osztály, minden függvénye és adattagja külön jelzés nélkül is statikus.
- **Ősosztályok**
Nem leszármazott osztály.
- **Interfészek**
Nem valósít meg interfészrt.
- **Attribútumok**
 - **boolean gameRunning:** azt tárolja, hogy épp folyamatban van-e a játék. A game loop vizsgálja, és ez alapján lép ki. Láthatóság: -

- **HashMap<String, AbstractField> map**: A pályán lévő mezőket tároló szótár, melynek kulcsa a mező (létérehozáskor generált) neve, értéke maga a mező. Főként a bejövő parancsok végrehajtásakor használjuk. Láthatóság: -
- **HashMap<String, Worker> workers**: A munkásokat tároló szótár, név-munkás párokból. Főként a bejövő parancsok végrehajtásakor használjuk. Láthatóság: -
- **HashMap<String, Crate> crates**: A lánkokat tároló szótár, név-lánkokat párokból. Főként a bejövő parancsok végrehajtásakor használjuk. Láthatóság: -
- **Metódusok**
 - **void main(String[] args)**: Tartalmazza a game loopot. minden ciklusban beolvassa a konzolról a felhasználó következő parancsát, és végrehajtja azt. Ez azt jelenti, hogy a beolvasott sort lefuttat a runCommand() függvényel. Láthatóság: +
 - **String getFieldName(AbstractField field)**: a map-ben történő lineáris kereséssel megállapítja és visszaadja a mező nevét. Ha nincs ilyen mező (akár nullpointer) akkor a "null" sztringet. Láthatóság: +
 - **String getMoveableName(Moveable m)**: A workers és crates adattagokban történő kereséssel megállapítja és visszaadja a mob nevét. Ha nincs ilyen moveable (akár nullpointer) akkor a "null" sztringet. Láthatóság: +
 - **void deleteWorker(Worker w)**: Kitörli a megadott munkást a munkások közül, így az többé nem vehet részt a játékban. Láthatóság: +
 - **void deleteCrate(Crate c)**: Kitörli a lánkat a lánkok közül. Láthatóság: +
 - **void endGame()**: Kiírja a képernyőre a játékosok pontjait, minden sorba egy játékosét, játékosnév: pontszám formában, majd véget vet a játéknak a gameRunning adattag beállításával. Láthatóság: -
 - **void runCommand(String cmd)**: A paraméterül kapott parancsokat szétszedi parancsnévre és paraméterekre, és ezzel meghívja a megfelelő függvényt az alábbiak közül. Láthatóság: -
 - **void parseMap(int lines, int columns)**: Beolvashat lines darab, columns számú karakterből álló sort, majd ez alapján létrehozza a pálya mezőit, és egymáshoz láncolja őket. Létrehozás után hozzáadja őket névvel együtt a map adattaghoz. A névadás és a bemenet értelmezésének pontos módját ld. a prototípus tervezeti közt, az előző dokumentumban! Láthatóság: -
 - **void spawnWorker(String name, double force, String field)**: Létrehoz egy munkást a megadott paraméterekkel, hozzáadja workers adattaghoz, és ráállítja a megfelelő mezőre. Láthatóság: -
 - **void spawnCrate(String name, String field)**: Létrehoz egy lánkat, hozzáadja a crates adattaghoz, és ráállítja a megfelelő mezőre. Láthatóság: -
 - **void setSwitch(String shole, String mySwitch)**: A megadott SwitchField-et ráállítja a megadott SwitchableHole-ra, annak setSHole() függvényével. Láthatóság: -
 - **void putFluid(String field, char fluid)**: A megadott mezőre mézet helyez, ha a fluid 'h' és olajat, ha az 'o'. Ehhez a mező putHoney() vagy putOil() függvényét használja. Láthatóság: -
 - **void step(String worker, String toward)**: A megadott munkást a megadott irányba próbálja léptetni. A toward paraméter értéke lehet: "right", "left", "up" vagy "down", egyébként kivételt dob. Lépés után végignézi a lánkokat, hogy felőlük vége lehet-e a játéknak (azaz célmezőn vannak-e és lehet-e őket közvetlenül tolni.) Ha minden lánka a játék végére szavaz, véget vet annak az endGame() függvényel. Láthatóság: -

- **void stat(String object)**: Megkeresi, hogy milyen pályán lévő dolog nevét kapta paraméterül, és meghívja annak a printStat() függvényét. Láthatóság: -
- **void script(String file)**: Beolvassa a fájlnévvel megadott fájlt, és soronként lefuttatja azt a runCommand() függvénnnyel. Ha ez nem sikerül, kivételt dob. Láthatóság: -

8.1.6. Moveable

- **Felelősség**

Absztrakt alaposztály, egy pályán mozogni képes objektumot reprezentál. Lehetőséget ad a mezőknek és másoknak, hogy minden, mozgatható dolgon végrehajtható műveletet elvégezzenek.

- **Ősosztályok**

Nem leszármazott osztály.

- **Interfészek**

Nem valósít meg interfészt.

- **Attribútumok**

- **SimpleField myField**: Tárolja, hogy éppen melyik mezőn van. Láthatóság: #

- **Metódusok**

- **void kill()**: A leszármazott osztályokban megöli az objektumot. Itt nem csinál semmit. Láthatóság: +
- **boolean squeez()**: Megpróbálja összenyomni az objektumot, és igazzal tér vissza, ha ez sikerült - egyébként hamissal. Az alaposztályban minden hamissal tér vissza. Láthatóság: +
- **void stepOnTarget(Worker w)**: Azt jelzi a Moveable számára, hogy célmezőre lépett. Nem csinál semmit. Láthatóság: +
- **void stepOffTarget(Worker w)**: Azt jelzi a Moveable számára, hogy a munkás letolta egy célmezőről. Nem csinál semmit. Láthatóság: +
- **void onSwitch(SwitchField s)**: Absztrakt függvény. Azt jelzi, hogy kapcsolóra léptünk. Láthatóság: +
- **void move(AbstractField here)**: Átlépteti a Moveable-t a megadott mezőre. Ehhez eldobatja magát az addigi mezőjével és frissíti a myField adattagot. Láthatóság: +
- **void setCouldMove(boolean b)**: Elmondja a Moveable-nek, hogy ő közvetlenül tolható-e. Nem csinál semmit, de felüldefiniálható. Láthatóság: +
- **boolean meetWith(Direction toward, Worker w, Moveable m)**: Azt jelzi, hogy w munkás egy toward irányú tolásának hatására ő m-nek ütközött. Megpróbálja m-et odébbtolni annak push függvényével, és azzal tér vissza, hogy ez sikerült-e. Láthatóság: +
- **boolean push(Direction toward, Worker w, Moveable m)**: w munkás lépésének hatására, m által, toward irányba megnyomódik. Megpróbálja szabaddá tenni a saját mezőjét először azzal, hogy odébblépteti magát toward irányba az alatta lévő mezővel, vagy ha ez nem sikerül, a saját squeez() függvényével. Azzal tér vissza, hogy ez sikerült-e.
- **void printStat()**: Kiírja az adattagjait az alábbi formában:

myField: mezőnév

ahol a mezőnév a Main.getFieldName() által visszaadott sztring, a mező beolvasásakor kapott neve. A pontos szintaktikát ld. az előző dokumentumban!

Láthatóság: +

8.1.7. Oil

- **Felelősség**

Az egyikfajta folyadékot, az olajat szimbolizálja. Feladata, hogy megkönnyítse a mezőn való mozgást.

- **Ősosztályok**

Fluid

- **Interfészek**

Nem valósít meg interfést.

- **Metódusok**

- **konstruktur:** Beállítja a súrlódás értékét 0-ra. Láthatóság: +
- **String getType():** Visszaadja az "oil" sztringet. A stat parancs miatt van. Láthatóság: +

8.1.8. Worker

- **Felelősség**

Egy munkást reprezentál. Képes magától mozogni a pályán, de tolni is lehet. Szükség esetén ő is megtol más mobokat, a rá jellemző erővel. Ha összenyomják, meghal. Tárolja és frissíti a maga által szerzett pontszámot.

- **Ősosztályok**

Moveable

- **Interfészek**

Nem valósít meg interfést.

- **Attribútumok**

- **int points:** A munkás által eddig szerzett pontszám; a játék végeredményének eldöntéséhez használjuk. Láthatóság: -
- **double force:** A munkás által kifejthető maximális erő. Láthatóság: -
- **double activeForce:** A tolás során használjuk; a tolásban még felhasználható erőt tárolja. Láthatóság: -
- **boolean alive:** Azt tárolja, hogy a munkás él-e még. Láthatóság: -

- **Metódusok**
 - **bool step(Direction toward)**: Átmásolja a force értékét az activeForce-ba, majd megpróbál lépni az adott irányba a mező stepMe() függvényét felhasználva. Azzal tér vissza, hogy sikerült-e. Láthatóság: +
 - **void incrementPoint()**: Megnöveli eggyel a játékos pontszámát. Láthatóság: +
 - **void decrementPoint()**: Csökkenti eggyel a játékos pontszámát. Láthatóság: +
 - **void kill()**: Megöli a munkást, és 0-ra állítja az erejét. Láthatóság: +
 - **boolean squeeze()**: A munkás összenyomódik. Meghívjuk a kill()-t és igazzal térünk vissza. Láthatóság: +
 - **bool decrementForce(double by)**: Azt jelzi tolás közben, hogy még by-nyi erőre szükség van a sikeres toláshoz. Ezt az értéket kivonjuk az activeForce adattagból, és azzal térünk vissza, hogy volt-e még ennyi erőnk. Láthatóság: +
 - **int getPoints()**: A munkás elmondja, hogy mennyi pontja van. A játék végénél használjuk. Láthatóság: +
 - **void printStat()**: meghívja az ōse printStat függvényét, majd kiírja a saját adattagjait az alábbi formában:
 - points: x
 - force: y
 - alive: érték

ahol x egész szám, a játékos pontszáma; y a munkás létrehozásakor megadott erő; érték pedig “true” vagy “false” attól függően, hogy a munkás él-e még.
- Láthatóság: +

8.1.9. AbstractField

- **Felelősség**
Egy mezőt reprezentál a játékban. Absztrakt alaposztály, konkrét példányai lehet például egy egyszerű járható mező, oszlop, lyuk, stb.
- **Ősosztályok**
Nem leszármazott osztály.
- **Interfészek**
Nem valósít meg interfészt.
- **Attribútumok**
 - **AbstractField neighbours**: A mező ismeri a szomszédait. Láthatóság: -
- **Metódusok**
 - **abstract bool accept(Direction toward, Worker w, Moveable m)**: Ez a függvény azt jelzi a mező számára, hogy szeretnék rálépni. Ekkor a mező, ha szabad, vagy azzá tudja tenni magát, befogadja a paraméterül kapott Moveable-t, és visszatér

igazzal. Ellenkező esetben nem csinál semmit, és hamis értékkel tér vissza.
Láthatóság: +

- **abstract bool couldMoveOn(Direction toward):** Megkérdezi a mezőtől, hogy ha egy tolás következne be a paraméterként átadott irányba, akkor ez lehetséges lenne-e. Ha falat vagy moveable-t tartalmaz a mező, akkor hamissal, üres mező esetén igazzal tér vissza. Láthatóság: +
- **void setNeighbor(Direction d, AbstractField n):** Az adott irányba beállítja, hogy ki a mező szomszédja. Láthatóság: +
- **void printStat():** Kiírja az adattagjait a következő formában:
neighborUp: n1
neighborRight: n2
neighborDown: n3
neighborLeft: n4
ahol n1..n4 a szomszédok neve, ami a Main.getFieldName() által visszaadott sztring
Láthatóság: +

8.1.10. SimpleField

- **Felelősség**

Az egyszerű járható mezőt reprezentálja, nem rendelkezik semmi különlegesssel.

- **Ősosztályok**

AbstractField

- **Interfészek**

Nem valósít meg interfész.

- **Attribútumok**

- **Moveable myMoveable:** A mező ismeri annak a referenciáját, aki rajta tartózkodik. Ez természetesen lehet NULL értékű is. (Egy lyuk esetében például nem is lehet más.)
Láthatóság: #
- **Fluid fluid:** A mezőn lévő folyadék. Láthatóság: #

- **Metódusok**

- **bool accept(Direction toward, Worker w, Moveable m):** Ez a függvény azt jelzi a mező számára, hogy szeretnének rálépni.
 - Amennyiben szabad, az elfogadásnak nincs akadálya.
 - Ha a mezőn már van valaki, akkor meghívja a belépést kérőn a meetWith() függvényét, hátha az odébb tudja tolni vagy valami mást tud csinálni vele.
 - Ha ez nem sikerült, akkor kénytelen visszatérni egy hamis értékkel, jelezve, hogy a befogadás nem volt sikeres.

Ezt a függvényt a leszármazottak felüldefiniálhatják, és akár további speciális függvényeket is hívhatnak az elemen, például egy sikeres befogadás esetén.

A befogadás sikerességét követnie kell egy move függvényhívásnak az érkező moveable-re.

Megjegyzés: Ez a függvény egy setterként is használható a myMoveable attribútumra, ha biztosak vagyunk benne, hogy az adott mező üres. Ekkor toward és w paraméter értéke lényegtelen, bizonyos esetekben lehet null is, de ha speciális mezőről van szó, jobb ha egy konkrét létrehozott dummy objektumot adunk oda neki.

Láthatóság: +

- **void dropMoveable()**: A myMoveable adattagot null-ra állítja. Láthatóság: +
- **bool couldMoveOn(toward: Direction)**: Az ōsosztály felüldefiniált függvénye a konkrét mezőre. Láthatóság: +
- **void determineBlocked(d: Direction)**: A szomszédos mezők couldMoveOn() függvényeinek segítségével ellenőrzi, hogy van-e legalább három szomszédja, ahonnan egy esetleg rajta lévő láda nem lenne közvetlenül tolható. Természetesen a d-vel ellentétes irányba vett szomszédot nem ellenőrzi. Láthatóság: #
- **void putHoney()**: mézessé teszi a mező felszínét, függetlenül attól, hogy addig milyen volt. Ezt úgy valósítja meg, hogy a fluid attribútum-ot Honey-ra állítja.

Láthatóság: +

- **void putOil()**: olajossá teszi a mező felszínét, függetlenül attól, hogy addig milyen volt. Ezt úgy valósítja meg, hogy a fluid attribútum-ot Oil-ra állítja. Láthatóság: +
- **bool stepMe(Direction toward, Worker w)**: levonja a toló munkás a még éppen kifejthető erejéből azt a mennyiséget, ami ahoz kell, hogy továbbléptesse a mezőn álló dolgot, és ha ez nem lett negatív, akkor megpróbálja továbbléptetni. A léptetés sikereségével tér vissza. Láthatóság: +
- **void printStat()**: Meghívja a super.printStat()-ot, majd kiírja az adattagjait a következő formában:

moveable: rajta álló neve, ami a Main.getMoveableName(myMoveable) által visszaadott sztring

fluid: a folyadék típusa, amit az állít magáról.

Láthatóság: +

8.1.11 Hole

- **Felelősség**

A lyukakat reprezentálja. Amennyiben rákerül egy objektum, az azonnal meghal.

- **Ősosztályok**

AbstractField→SimpleField

- **Interfészek**

Nem valósít meg interfészt.

- **Attribútumok**

Nincsen attribútuma.

- **Metódusok**

- **bool accept(Direction toward, Worker w, Moveable m):** Hasonló a működése a közvetlen őséhez képest, azonban rögtön elfogadja az érkező m paramétert, majd meg is öli a kill() függvényével. Ezután visszatér egy igaz értékkel. Láthatóság: +

8.1.12 SwitchableHole

- **Felelősség**

Egy kapcsolatható lyukat reprezentál. Ha be van kapcsolva, Hole-ként viselkedik, ha nincs, akkor úgy, mint egy SimpleField.

- **Ősosztályok**

AbstractField → SimpleField

- **Interfészek**

Nem valósít meg interfészt.

- **Attribútumok**

- **bool opened:** Eltárolja magáról, hogy a lyuk nyitva van-e. Láthatóság: -

- **Metódusok**

- **bool accept(Direction toward, Worker w, Moveable m):** Amennyiben az opened értéke igaz, viselkedése megegyezik a Hole osztály azonos nevű függvényével. Amennyiben az opened hamis, viselkedése megegyezik a SimpleField azonos nevű függvényével. Láthatóság: +
- **void open():** Beállítja az opened értékét igazra, és ha valaki tartózkodna a mezőn, akkor azt megöli. Láthatóság: +
- **void close():** Beállítja az opened értékét hamisra. Láthatóság: +
- **void printStat():** Meghívja a super.printStat()-ot, majd kiírja az adattagjait a következő formában:
opened: igaz/hamis
Láthatóság: +

8.1.13 SwitchField

- **Felelősség**

Kapcsolót tartalmazó mezőt reprezentál. Amennyiben láda van a mezőn a hozzá tartozó lyukat nyitva, minden egyéb esetben zárva tartja

- **Ősosztályok**

AbstractField → SimpleField

- **Interfészek**

Nem valósít meg interfészt.

- **Attribútumok**
 - **SwitchableHole sHole:** Az a kapcsolható lyuk, akit a kapcsoló irányít. Láthatóság: -
- **Metódusok**
 - **bool accept(Direction toward, Worker w, Moveable m):** Megegyezik az ōsosztály azonos nevű függvényével. Csupán annyiban egészíti ki azt, hogy amennyiben a befogadás sikeres, meghívja a ráérkező Moveable-nek az onSwitch függvényét, saját magát átadva paraméterként. Ennek értelme, hogy a ráérkező Moveable eldöntheti, hogy a kapcsolót kívánja-e bekapcsolni. Láthatóság: +
 - **void turnOn():** Meghívja az sHole open() függvényét. Láthatóság: +
 - **void turnOff():** Meghívja az sHole close() függvényét. Láthatóság: +
 - **void setShole(SwitchableHole sh):** A SwitchField-et ráállítja a megadott SwitchableHole-ra. Láthatóság: +
 - **void printStat():** Meghívja a super.printStat()-ot, majd kiírja az adattagjait a következő formában:
sHole: kapcsolható lyuk neve, ami a Main.getFieldName() által visszaadott sztring
Láthatóság: +

8.1.14 TargetField

- **Felelősség**
A célmezőket reprezentálja. Legtöbb funkciójukban egy egyszerű járható mezővel egyeznek meg, ám a pontozást elősegítő rendeltetésük is van.
- **Ősosztályok**
AbstractField → SimpleField
- **Interfészek**
Nem valósít meg interfészt.
- **Attribútumok**
Nincsen attribútuma.
- **Metódusok**
 - **bool accept(Direction toward, Worker w, Moveable m):** Az ōsosztály funkcióján felül, amennyiben a befogadás sikeres, meghívja a kapott Moveable paraméter StepOnTarget(w) függvényét, és az eltávozott Moveable típusú objektum stepOffTarget(w) függvényét. Láthatóság: +

8.1.15 Wall

- **Felelősség**
Egy fal vagy oszlop miatt járhatatlan mezőt reprezentál.

- Ősosztályok

AbstractField

- Interfészek

Nem valósít meg interfészt.

- Attribútumok

Nincsen attribútuma.

- Metódusok

- **bool accept(Direction toward, Worker w, Moveable m,):** Konstans hamis értéket ad vissza, mert semmit nem hajlandó elfogadni. Láthatóság: +
- **bool couldMoveOn(toward: Direction):** Az Ősosztály felüldefiniált függvénye a fal típusú mezőre. Konstans hamis értékkel tér vissza. Láthatóság: +

8.2. A tesztek részletes tervei, leírásuk a teszt nyelvén

8.2.1. Teszteset1: Pályainicializálás

- Leírás

Egy kisebb pálya létrehozása, melyen minden típusú mezőből van egy.

- Ellenőrzött funkcionális, várható hibahelyek

A pályát alkotó mezők létrehozása és szomszédsági viszonyaiak megfelelő beállítása. Hibás lehet: a Main osztályban parsemap, setswitch és stat parancsokat megvalósító függvények, vagy az ezek által hívottak.

- Bemenet

parsemap 4 5

wwwww

wmthw

wkmsw

wwwww

setswitch f_2_3 f_2_1

stat f_2_1

- Elvárt kimenet

neighborUp: f_1_1

neighborRight: f_2_2

neighborDown: f_2_1

neighborLeft: f_2_0

moveable: null

fluid: null

sHole: f_2_3

1. Teszteset2: 1.Worker steps on Wall

- **Leírás**

A munkás egy falra próbál lépni.

- **Ellenőrzött funkcionális, várható hibahelyek**

A munkás lépése meghiúsul. (Nem tud falra lépni.) Hibás lehet: a lépés során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmmmmmw

wmmmmmw

wwwwww

worker w1 5 f_1_1

step w1 left

stat f_1_1

stat w1

- **Elvárt kimenet**

neighborUp: f_0_1

neighborRight: f_1_2

neighborDown: f_2_1

neighborLeft: f_1_0

moveable: w1

fluid: no fluid

myField: f_1_1

force: 5

points: 0

alive: true

2. Teszteset3: 2.Worker steps on SimpleField

- **Leírás**

A munkás sima mezőre lép.

- **Ellenőrzött funkcionális, várható hibahelyek**

Leellenőrizzük azt a folyamatot, ahogy egy munkás egy szomszédos, üres SimpleField típusú mezőre lép. Hibás lehet: a lépés során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmmmmmw

wmmmmmw

wwwwww

worker w1 5 f_1_1

step w1 right

stat f_1_1

stat f_1_2

stat w1

- **Elvárt kimenet**

neighborUp: f_0_1

neighborRight: f_1_2

neighborDown: f_2_1

neighborLeft: f_1_0

moveable: null

fluid: no fluid

neighborUp: f_0_2

neighborRight: f_1_3

neighborDown: f_2_2

neighborLeft: f_1_1

moveable: w1

fluid: no fluid

myField: f_1_2

force: 5

points: 0

alive: true

3. Teszteset4: 3.Worker steps on TargetField

- **Leírás**

A munkás célmezőre lép.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Leellenőrzük azt a folyamatot, ahogy egy munkás egy szomszédos, üres TargetField típusú mezőre lép. Hibás lehet: a lépés során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmtmmw

wmmmmmw

wwwww

worker w1 5 f_1_1

step w1 right

stat f_1_1

stat f_1_2

stat w1

- **Elvárt kimenet**

neighborUp: f_0_1

neighborRight: f_1_2

neighborDown: f_2_1

neighborLeft: f_1_0

moveable: null

fluid: no fluid

neighborUp: f_0_2

neighborRight: f_1_3

neighborDown: f_2_2

neighborLeft: f_1_1

moveable: w1

fluid: no fluid

myField: f_1_2

force: 5

points: 0

alive: true

4. Teszteset5: 4. Worker steps on SwitchField

- **Leírás**

A munkás kapcsoló típusú mezőre lép.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Leellenőrzük azt a folyamatot, ahogy egy munkás egy szomszédos, üres SwitchField típusú mezőre lép. Hibás lehet: a lépés során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmkmmw

wmmsmw

wwwwww

setswitch f_2_3 f_1_2

worker w1 5 f_1_1

step w1 right

stat f_2_3

stat f_1_2

stat w1

- **Elvárt kimenet**

neighborUp: f_1_3

neighborRight: f_2_4

neighborDown: f_3_3

neighborLeft: f_2_2

moveable: null

fluid: no fluid

opened: false

neighborUp: f_0_2

neighborRight: f_1_3

neighborDown: f_2_2

neighborLeft: f_1_1

moveable: w1

fluid: no fluid

myField: f_1_2

force: 5

points: 0

alive: true

5. Teszteset6: 5. Worker steps on Hole

- **Leírás**

A munkás lyukra lép.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Leellenőrzük azt a folyamatot, ahogy egy munkás egy szomszédos lyukra lép és meghal. Hibás lehet: a lépés során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmhmmw

wmmmmw

wwwwww

worker w1 5 f_1_1

step w1 right

stat f_1_1

stat f_1_2

stat w1

- **Elvárt kimenet**

neighborUp: f_0_1

neighborRight: f_1_2

neighborDown: f_2_1

neighborLeft: f_1_0

moveable: null

fluid: no fluid

neighborUp: f_0_2

neighborRight: f_1_3

neighborDown: f_2_2

neighborLeft: f_1_1

moveable: null

fluid: no fluid

myField: f_1_2

force: 0
points: 0
alive: false

6. Teszteset7: 6.a Worker steps on opened SwitchableHole

- **Leírás**

A munkás kapcsolható lyukra lép és meghal.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Leellenőrizzük azt a folyamatot, ahogy egy munkás egy szomszédos kapcsolható lyukra lép és meghal. Hibás lehet: a lépés során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmsmmw

wmkmmw

wwwwww

setswitch f_1_2 f_2_2

worker w1 5 f_1_1

crate c1 f_2_2

step w1 right

stat f_1_1

stat f_1_2

stat w1

- **Elvárt kimenet**

neighborUp: f_0_1
neighborRight: f_1_2
neighborDown: f_2_1
neighborLeft: f_1_0
moveable: null
fluid: no fluid

neighborUp: f_0_2
neighborRight: f_1_3
neighborDown: f_2_2

neighborLeft: f_1_1

moveable: null

fluid: no fluid

myField: f_1_2

force: 0

points: 0

alive: false

7. Teszteset8: 6.b Worker steps on closed SwitchableHole

- **Leírás**

A munkás kapcsolható lyukra lép.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Leellenőrizzük azt a folyamatot, ahogy egy munkás egy szomszédos kapcsolható lyukra lép. Hibás lehet: a lépés során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmsmmw

wmkmmw

wwwwww

setswitch f_1_2 f_2_2

worker w1 5 f_1_1

step w1 right

stat f_1_1

stat f_1_2

stat w1

- **Elvárt kimenet**

neighborUp: f_0_1

neighborRight: f_1_2

neighborDown: f_2_1

neighborLeft: f_1_0

moveable: null

fluid: no fluid

```
neighborUp: f_0_2
neighborRight: f_1_3
neighborDown: f_2_2
neighborLeft: f_1_1
moveable: w1
fluid: no fluid
```

```
myField: f_1_2
force: 5
points: 0
alive: true
```

8. Teszteset9: 7. Worker tries to push crate on Wall

- **Leírás**

A munkásnak nem sikerül eltolni a lánát.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Leellenőriztük azt a folyamatot, ahogy egy munkás falra akar tolni egy lánát. Hibás lehet: a tolás során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

```
parsemap 4 6
```

```
wwwwww
```

```
wmmmmmw
```

```
wmmmmw
```

```
wwwwww
```

```
worker w1 5 f_1_3
```

```
crate c1 f_1_4
```

```
step w1 right
```

```
stat f_1_3
```

```
stat f_1_4
```

```
stat w1
```

- **Elvárt kimenet**

```
neighborUp: f_0_3
neighborRight: f_1_4
```

neighborDown: f_2_3

neighborLeft: f_1_2

moveable: w1

fluid: no fluid

neighborUp: f_0_4

neighborRight: f_1_5

neighborDown: f_2_4

neighborLeft: f_1_3

moveable: c1

fluid: no fluid

myField: f_1_3

force: 5

points: 0

alive: true

9. Teszteset10: 8. Worker tries to push crate on SimpleField

- **Leírás**

A munkás sima mezőre tol egy lánát.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Leellenőrzük azt a folyamatot, ahogy egy munkás egy lánát egy azzal szomszédos, üres SimpleField típusú mezőre tol. Hibás lehet: a tolás során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmmmmw

wmmmmw

wwwwww

worker w1 5 f_1_2

crate c1 f_1_3

step w1 right

stat f_1_3

stat f_1_4

stat w1

- **Elvárt kimenet**

```
neighborUp: f_0_3  
neighborRight: f_1_4  
neighborDown: f_2_3  
neighborLeft: f_1_2  
moveable: w1  
fluid: no fluid
```

```
neighborUp: f_0_4  
neighborRight: f_1_5  
neighborDown: f_2_4  
neighborLeft: f_1_3  
moveable: c1  
fluid: no fluid
```

```
myField: f_1_3  
force: 5  
points: 0  
alive: true
```

10. Teszteset11: 9. Worker tries to push crate on TargetField

- **Leírás**

A munkás célmezőre tol egy lánát.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Leellenőrzük azt a folyamatot, ahogy egy munkás egy lánát egy azzal szomszédos, üres TargetField típusú mezőre tol. Hibás lehet: a tolás során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmmtmw

wmmmmw

wwwww

worker w1 5 f_1_2

crate c1 f_1_3

stat c1

step w1 right

stat w1

stat c1

- **Elvárt kimenet**

```
myField: f_1_3
couldMove: false
isOnTarget: false
```

```
myField: f_1_3
force: 5
points: 1
alive: true
```

```
myField: f_1_4
couldMove: true
isOnTarget: true
```

11. Teszteset12: 10. Worker tries to push crate on SwitchField

- **Leírás**

A munkás kapcsoló típusú mezőre tol egy lánát.

- **Ellenőrzött funkcionálitás, várható hibahelyek**

Leellenőrizzük azt a folyamatot, ahogy egy munkás egy lánát egy azzal szomszédos, üres SwitchField típusú mezőre tol, és a hozzá tartozó lyuk kinyílik. Hibás lehet: a tolás során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmmkmw

wmmsmw

wwwwww

worker w1 5 f_1_2

crate c1 f_1_3

setswitch f_2_3 f_1_3

stat f_2_3

step w1 right

stat f_2_3

- **Elvárt kimenet**

```
neighborUp: f_1_3
neighborRight: f_2_4
neighborDown: f_3_3
neighborLeft: f_2_2
moveable: null
fluid: no fluid
opened: false
```

```
neighborUp: f_1_3
neighborRight: f_2_4
neighborDown: f_3_3
neighborLeft: f_2_2
moveable: null
fluid: no fluid
opened: true
```

12. Teszteset13: 11. Worker tries to push crate on Hole

- **Leírás**

A munkás lyukba tol egy lánát, és az eltűnik.

- **Ellenőrzött funkcionális, várható hibahelyek**

Leellenőriztük azt a folyamatot, ahogy egy munkás egy lánát egy azzal szomszédos Hole típusú mezőre tol. Hibás lehet: a tolás során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmmhmw

wmmmmmw

wwwwww

worker w1 5 f_1_2

crate c1 f_1_3

step w1 right

stat c1

- **Elvárt kimenet**

```
myField: f_1_4
couldMove: true
isOnTarget: false
```

13. Teszteset14: 12. a Worker tries to push crate on opened SwitchableHole

- **Leírás**

A munkás kapcsolható lyukra tolja a lánát, és az eltűnik.

- **Ellenőrzött funkcionális, várható hibahelyek**

Leellenőrizzük azt a folyamatot, ahogy egy munkás egy lánát egy azzal szomszédos, üres SwitchableHole típusú mezőre tol, ami nyitva volt, így a láná eltűnik. Hibás lehet: a tolás során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

```
parsemap 4 6
```

```
wwwwww
```

```
wmmsmw
```

```
wmmkmw
```

```
wwwwww
```

```
worker w1 5 f_1_1
```

```
crate c1 f_1_2
```

```
setswitch f_1_3 f_2_3
```

```
crate c2 f_2_3
```

```
step w1 right
```

```
stat c1
```

- **Elvárt kimenet**

```
myField: f_1_4
couldMove: true
isOnTarget: false
```

14. Teszteset15: 12. b Worker tries to push crate on closed SwitchableHole

- **Leírás**

A munkás kapcsolható lyukra tolja a lánát.

- **Ellenőrzött funkcionális, várható hibahelyek**

Leellenőrzük azt a folyamatot, ahogy egy munkás egy lánát egy azzal szomszédos, üres SwitchableHole típusú mezőre tol, ami zárva volt. Hibás lehet: a tolás során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmmmsmw

wmmkmw

wwwwww

worker w1 5 f_1_2

crate c1 f_1_3

setswitch f_1_3 f_2_3

step w1 right

stat c1

- **Elvárt kimenet**

myField: f_1_4

couldMove: true

isOnTarget: false

15. Tesztelés 16: 13. Worker pushes worker

- **Leírás**

A munkás megtol egy másik munkást.

- **Ellenőrzött funkcionális, várható hibahelyek**

Leellenőrzük azt a folyamatot, ahogy egy SimpleField-ekből álló pályán egy munkás megtol egy másikat. Hibás lehet: a tolás során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmmmmmw

wmmmmw

wwwwww

worker w1 5 f_1_1

worker w2 5 f_1_2

step w1 right

stat w2

- **Elvárt kimenet**

myField: f_1_3

points: 0

force: 5

alive: true

16. Teszteset17: 14. Worker pushes crate which pushes worker

- **Leírás**

A munkás megtol egy lánát, ami megtol egy másik munkást.

- **Ellenőrzött funkcionális, várható hibahelyek**

Leellenőrzük azt a folyamatot, ahogy egy munkás által tolta lánát megtol egy másik munkást. Hibás lehet: a tolás során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmmmmmw

wmmmmmw

wwwwww

worker w1 5 f_1_1

crate c f_1_2

worker w2 5 f_1_3

step w1 right

stat w2

- **Elvárt kimenet**

myField: f_1_4

points: 0

force: 5

alive: true

17. Teszteset18: 15. Worker pushes crate which pushes crate

- **Leírás**

A munkás megtol egy lánát, ami megtol egy másik lánát.

- **Ellenőrzött funkcionális, várható hibahelyek**

Leellenőrzük azt a folyamatot, ahogy egy munkás által tolta lánát megtol egy másik lánát. Hibás lehet: a tolás során használt függvények valamelyike. (Ld. korábbi szekvenciadiagramok.)

- **Bemenet**

parsemap 4 6

wwwwww

wmmmmmw

wmmmmmw

```
wwwwww  
worker w1 5 f_1_1  
crate c1 f_1_2  
crate c2 f_1_3  
step w1 right  
stat c2
```

- **Elvárt kimenet**

```
myField: f_1_4  
couldMove: false  
isOnTarget: false
```

18. Teszteset19: 16. Worker pushes crate off TargetField

- **Leírás**

A munkás letol egy lánát a célmezőről.

- **Ellenőrzött funkcionális, várható hibahelyek**

Leellenőrzük azt a folyamatot, ahogy egy munkás letol egy lánát egy célmezőről.

Hibás lehet: stepOffTarget() függvény.

- **Bemenet**

```
parsemap 4 6  
wwwwww  
wmtmmw  
wmffffw  
wwwwww  
worker w1 5 f_1_1  
crate c1 f_1_2  
step w1 right  
stat c1  
stat w1
```

- **Elvárt kimenet**

```
myField: f_1_3  
couldMove: true  
isOnTarget: false  
myField: f_1_2  
points: -1  
force: 5  
alive: true
```

19. Teszteset20: 17. Worker pushes crate off SwitchField

- **Leírás**

A munkás letol egy lánát a kapcsoló típusú mezőről.

- **Ellenőrzött funkcionális, várható hibahelyek**

Leellenőrizzük azt a folyamatot, ahogy egy munkás letol egy lánát egy kapcsolóról, ezzel kikapcsolva azt. Hibás lehet: az onSwitch() függvény.

- **Bemenet**

parsemap 4 6

wwwwww

wmkmmw

wmsmmw

wwwwww

setswitch f_2_2 f_1_2

worker w1 5 f_1_1

crate c1 f_1_2

step w1 right

stat f_2_2

- **Elvárt kimenet**

neighborUp: f_1_2

neighborRight: f_2_3

neighborDown: f_3_2

neighborLeft: f_2_1

moveable: null

fluid: no fluid

opened: false

20. Teszteset21: 18. Worker pushes worker into wall

- **Leírás**

A munkás falnak tol egy másik munkást, és az meghal.

- **Ellenőrzött funkcionális, várható hibahelyek**

Leellenőrizzük azt a folyamatot, ahogy egy munkás falnak tol egy másikat, ezzel összenyomva őt. Hibás lehet: a squeeze() és a kill() függvény.

- **Bemenet**

parsemap 4 6

wwwwww

wmmmmw

wmmmmw

wwwwww

worker w1 5 f_1_1

worker w2 5 f_1_2

step w2 left

stat w1

- **Elvárt kimenet**

```
myField: f_1_1
```

```
points: 0
```

```
force: 0
```

```
alive: false
```

21. Teszteset22: 19. Worker tries to push worker which pushes worker on Wall

- **Leírás**

A munkás megtol egy másik munkást, ami egy harmadikat a falnak tol. A falnakolt munkás meghal.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Leellenőrizzük azt a folyamatot, ahogy egy munkás megtol egy másikat, ami megtol egy harmadikat, ezzel összenyomva őt. Hibás lehet: a léptetés algoritmusa.

- **Bemenet**

```
parsemap 4 6
```

```
wwwwww
```

```
wmmmmw
```

```
wmmmmw
```

```
wwwwww
```

```
worker w1 5 f_1_1
```

```
worker w2 5 f_1_2
```

```
worker w3 5 f_1_3
```

```
step w3 left
```

```
stat w1
```

```
stat w2
```

- **Elvárt kimenet**

```
myField: f_1_1
```

```
points: 0
```

```
force: 0
```

```
alive: false
```

```
myField: f_1_1
```

```
points: 0
```

```
force: 5
```

```
alive: true
```

22. Teszteset23: 20. Worker pushes crate on SwitchableHole, and opens

- **Leírás**

A munkás rátol egy lántható lyukra, majd kinyitja a lyukat. A láda eltűnik.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Leellenőrizzük azt a folyamatot, ahogy egy munkás rátol egy lántható lyukra, majd kinyitja a lyukat. Hibás lehet: az onSwitch() és az open() függvény.

- **Bemenet**

```
parsemap 4 6
wwwwww
wmmsmw
wmmkmw
wwwwww
setswitch f_1_3 f_2_3
worker w1 5 f_1_1
crate c1 f_1_2
crate c2 f_2_2
step w1 right
step w1 left
step w1 down
step w1 right
stat f_1_3
```

- **Elvárt kimenet**

```
neighborUp: f_0_3
neighborRight: f_1_4
neighborDown: f_2_3
neighborLeft: f_1_2
moveable: null
fluid: no fluid
opened: true
```

23. Teszteset24: 21. Worker puts Honey on SimpleField

- **Leírás**

A játékos összekeni a mezőt mézzel.

- **Ellenőrzött funkcionális, várható hibahelyek**

A játékos sikeresen összekente a mezőt mézzel. Hibás lehet: a putHoney() függvény.

- **Bemenet**

```
parsemap 4 6
wwwwww
wmmmmmw
wmmmmmw
wwwwww
worker w1 5 f_1_1
putfluid f_1_1 h
stat f_1_1
```

- **Elvárt kimenet**

```
neighborUp: f_0_0
neighborRight: f_1_2
neighborDown: f_2_1
neighborLeft: f_1_0
moveable: w1
fluid: honey
```

24. Teszteset25: 22. Worker puts Oil on SimpleField then moves

- **Leírás**

A játékos összekeni a olajjal és odébb áll.

- **Ellenőrzött funkcionális, várható hibahelyek**

A mező összekentsége megmarad, akkor is, ha utána odébb lép. Hibás lehet: a putOil() függvény és a lépés során használt függvények.

- **Bemenet**

```
parsemap 4 6
```

```
wwwwww
```

```
wmmmmmw
```

```
wmmmmmw
```

```
wwwwww
```

```
worker w1 5 f_1_1
```

```
putfluid f_1_1 o
```

```
step w1 right
```

```
stat f_1_1
```

```
stat w1
```

- **Elvárt kimenet**

```
neighborUp: f_0_0
```

```
neighborRight: f_1_2
```

```
neighborDown: f_2_1
```

```
neighborLeft: f_1_0
```

```
moveable: null
```

```
fluid: oil
```

```
myField: f_1_2
```

```
points: 0
```

```
force: 5
```

```
alive: true
```

25. Teszteset26: 23. Worker puts Honey, then Oil on SimpleField

- **Leírás**

A játékos összekeni a mezőt előbb mézzel, majd olajjal.

- **Ellenőrzött funkcionális, várható hibahelyek**

Annak az anyagnak kell a mezőn maradnia, amit utoljára rátettek. Hibás lehet: a putHoney() és putOil() függvények.

- **Bemenet**

parsemap 4 6

wwwwww

wmmmmmw

wmmmmw

wwwww

worker w1 5 f_1_1

putfluid f_1_1 h

putfluid f_1_1 o

stat f_1_1

- **Elvárt kimenet**

neighborUp: f_0_0

neighborRight: f_1_2

neighborDown: f_2_1

neighborLeft: f_1_0

moveable: w1

fluid: oil

26. Teszteset27: 24. Worker puts Honey on SwitchableHole, then opens and closes the Hole

- **Leírás**

A játékos összezen egy zárható lyukat (zárt állapotában természetesen), majd bezárja, és kinyitja azt.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A lyuk kinyitása majd újra bezárása nem tisztíthatja meg a mezőt, azaz annak az anyagnak kell rajta maradnia, amit utoljára rákentek. Hibás lehet: az open() és a close() függvények.

- **Bemenet**

parsemap 4 6

wwwwww

wsmkmw

wmmmmw

wwwww

setswitch f_1_1 f_1_3

worker w1 5 f_1_1

crate c1 f_1_2

putfluid f_1_1 h

step w1 right

step w1 right

stat f_1_1

- **Elvárt kimenet**

neighborUp: f_0_0

neighborRight: f_1_2

neighborDown: f_2_1

neighborLeft: f_1_0

moveable: null

fluid: honey

27. Teszteset28: 25. Just enough force

- **Leírás**

A munkásnak éppen elég ereje van, hogy az előtte lévő lánchákat eltolja.

- **Ellenőrzött funkcionális, várható hibahelyek**

A tolás sikeres a munkás erőssége miatt. Hibás lehet: az erők levonásához és ellenőrzéséhez használt függvények egyike.

- **Bemenet**

parsemap 4 6

wwwwww

wmmmmw

wmmmmw

wwwwww

worker w1 3 f_1_1

crate c1 f_1_2

crate c2 f_1_3

step w1 right

stat w1

- **Elvárt kimenet**

myField: f_1_2

points: 0

force: 3

alive: true

28. Teszteset29: 26. Not enough force

- **Leírás**

A munkásnak nincsen elég ereje, hogy az előtte lévő lánchákat eltolja.

- **Ellenőrzött funkcionális, várható hibahelyek**

A tolás meghiúsul a munkás erő hiánya miatt. Hibás lehet: az erők levonásához és ellenőrzéséhez használt függvények egyike.

- **Bemenet**

parsemap 4 6

```
wwwwww  
wmmmmw  
wmmmmw  
wwwww  
worker w1 2 f_1_1  
crate c1 f_1_2  
crate c2 f_1_3  
step w1 right  
stat w1
```

- **Elvárt kimenet**

```
myField: f_1_1  
points: 0  
force: 2  
alive: true
```

29. Teszteset30: 27. Not enough force because of Honey

- **Leírás**

A munkásnak éppen elég ereje lenne, hogy az előtte lévő lánát eltolja, de mivel van egy bemézezett mező, ez mégsem fog sikerülni neki.

- **Ellenőrzött funkcionális, várható hibahelyek**

A tolás méz miatt meghiúsul. Hibás lehet: az erők levonásához és ellenőrzéséhez használt függvények egyike.

- **Bemenet**

```
parsemap 4 6  
wwwwww  
wmmmmw  
wmmmmw  
wwwww  
worker w1 3 f_1_1  
crate c1 f_1_2  
crate c2 f_1_3  
putfluid f_1_1 h  
step w1 right  
stat w1
```

- **Elvárt kimenet**

```
myField: f_1_1  
points: 0  
force: 3  
alive: true
```

30. Teszteset31: 28. Enough force because of Oil

- **Leírás**

A munkásnak nem lenne elég ereje, hogy az előtte lévő lánát eltolja, de mivel van egy beolajozott mező, ez mégis fog sikerülni neki.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A tolás olaj miatt sikerül. Hibás lehet: az erők levonásához és ellenőrzéséhez használt függvények egyike.

- **Bemenet**

```
parsemap 4 6
```

```
wwwwww
```

```
wmmmmw
```

```
wmmmmw
```

```
wwwwww
```

```
worker w1 2 f_1_1
```

```
crate c1 f_1_2
```

```
crate c2 f_1_3
```

```
putfluid f_1_1 o
```

```
step w1 right
```

```
stat w1
```

- **Elvárt kimenet**

```
myField: f_1_2
```

```
points: 0
```

```
force: 2
```

```
alive: true
```

31. Teszteset32: 29. Enough force because more Oil than Honey

- **Leírás**

A munkásnak nem lenne elég ereje, hogy az előtte lévő lánát eltolja, de mivel több beolajozott mező van, mint mézezett, ez mégis fog sikerülni neki.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A tolás több olaj, mint méz miatt sikerül. Hibás lehet: az erők levonásához és ellenőrzéséhez használt függvények egyike.

- **Bemenet**

```
parsemap 4 6
```

```
wwwwww
```

```
wmmmmw
```

```
wmmmmw
```

```
wwwwww
```

```
worker w1 2 f_1_1
```

```
crate c1 f_1_2
```

```
crate c2 f_1_3
putfluid f_1_1 o
putfluid f_1_2 o
putfluid f_1_3 h
step w1 right
stat w1
```

- **Elvárt kimenet**

```
myField: f_1_2
points: 0
force: 2
alive: true
```

32. Teszteset33: 30. Not enough force because more Honey than Oil

- **Leírás**

A munkásnak lenne elég ereje, hogy az előtte lévő lánchákat eltolja, de mivel több mézezett mező van, mint olajozott, ez mégsem fog neki sikerülni.

- **Ellenőrzött funkcionális, várható hibahelyek**

A tolás több méz, mint olaj miatt nem sikerül. Hibás lehet: az erők levonásához és ellenőrzéséhez használt függvények egyike.

- **Bemenet**

```
parsemap 4 6
wwwwww
wmmmmmw
wmwwwmw
wwwwww
worker w1 3 f_1_1
crate c1 f_1_2
crate c2 f_1_3
putfluid f_1_1 o
putfluid f_1_2 h
putfluid f_1_3 h
step w1 right
stat w1
```

- **Elvárt kimenet**

```
myField: f_1_1
points: 0
force: 3
alive: true
```

8.3. A tesztelést támogató programok tervei

Specifikáció pontosítása: A tesztszkript egy Powershell-szkript. Használatához - ha ez még nincs beállítva - engedélyeznünk kell a szkriptek futtatását a Windows rendszeren. Ezt a Powershell-t rendszergazdai jogokkal indítva, a "Set-ExecutionPolicy Unrestricted" parancs kiadásával tehetjük meg.

A szkript teszteseteket vár el, amik egy bemenetet és egy elvárt kimenetet tartalmazó fájlból állnak. Az előbbi neve x.in és az utóbbi neve x.out, ahol x a teszt neve. Ez lehet tetszőleges, de tesztesetenként különböző. A fájloknek a szkripttel azonos mappában kell lenniük. Ettől eltérni csak az egyetlen tesztesetet futtató módban (lásd lejjebb) szabad, ekkor viszont paraméterként megfelelő elérési utat kell megadni. Az input fájl tartalmazza a programnak standard inputon átadandó bemenetet, az output fájl pedig a program által standard outputon kiírandó elvárt kimenetet.

Indíthatjuk a tesztelést single-test módban, ezt úgy tehetjük meg, hogy a szkript paraméteréül megadjuk előbb az input majd az output fájl nevét. Ha a teszt sikерül, akkor a szkript a "Test with test file [inputfájl] passed." szöveget írja a képernyőre, ha elbukik, akkor előbb a "Test with test file [inputfájl] FAILED!" szöveget, majd a várt és kapott kimenet közötti különbségeket az alábbi formában:

Got:

[kapott kimenet]

Expected:

[elvárt kimenet]

Test-all módban úgy indíthatjuk, hogy nem adunk meg neki paramétert. Ekkor betűrendben végigmegy a vele egy mappában lévő összes teszteseten, és úgy viselkedik, mintha mindenre meghívták volna single-test módban.

Fontos, hogy a lefordított .class fájlok szkripthez képesti relatív elérési útvonala .\class\killer_sokoban*.class legyen.

Részletes tervezet: A szkript rendelkezni fog egy test() függvényel, amely paraméterként kapja az input fájl és az elvárt outputot tartalmazó fájl nevét, és a specifikációnak megfelelően összehasonítja őket, majd kiírja a képernyőn megjelenítendő szöveget.

Lesz egy testAll függvény is, ami listába gyűjti a vele egy mappában található teszteseteket, és mindenre meghívja a test() függvényt.

A szkript törzse a paraméterek számától függően vagy a test() vagy a testAll() függvényt hívja meg.

8.4. Napló

Kezdet	Időtartam	Résznevők	Leírás
2018. 04. 05. 9:00	6 óra	Schulcz	Ki- és bemeneti nyelv implementálásának, Main osztály

			működésének kigondolása és részbeni implementálása, nem mező típusú osztályok, adattagok pontos leírása. Néhány új függvény írása
2018. 04. 05. 19:00	1 óra	Zalavári	Tesztek írása
2018. 04. 06. 9:00	2 óra	Benkő	Tesztesetek kiötlése
2018. 04. 06. 19:00	2 óra	Zalavári	Tesztek írása
2018. 04. 06. 6:30	2 óra	Schulcz	Powershell nyelv részletesebb megismerése, tesztelést végző szkript megtervezése és kísérletképpen megírása
2018. 04. 06. 14:00	30 perc	Takács	8.3
2018. 04. 06. 16:45	30 perc	Takács	8.3
2018. 04. 07. 13:00	30 perc	Schulcz	Apróbb javítások
2018. 04. 07. 14:00	1 óra	Zalavári	Dokumentum átnézése
2018. 04. 07. 17:00	3 óra	Schulcz	Tesztesetek befejezése

10. Prototípus beadása

10.1 Fordítási és futtatási útmutató

10.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
AbstractField.java	2 KB	2018.03. 14. 21:32	A mezők ősosztálya
Crate.java	2 KB	2018.03. 14. 21:32	A láda osztály
Direction.java	1 KB	2018.03. 14. 21:32	Az irányok enum-ja
Hole.java	1 KB	2018.03. 14. 21:32	A lyukak osztálya
Main.java	7 KB	2018.03. 14. 21:32	A szimulációt irányító osztály
Moveable.java	3 KB	2018.03. 14. 21:32	A mobok ősosztálya
SimpleField.java	4 KB	2018.03. 14. 21:32	Az egyszerű mező osztály
SwitchableHole.java	2 KB	2018.03. 14. 21:32	A kapcsolható lyuk osztály
SwitchField.java	2 KB	2018.03. 14. 21:32	A kapcsoló osztály
TargetField.java	2 KB	2018.03. 14. 21:32	A célmező osztály
Wall.java	1 KB	2018.03. 14. 21:32	A fal osztály
Worker.java	2 KB	2018.03. 14. 21:32	A munkások osztály
tester.ps1	1 KB	2018. 04. 06. 12:25	A tesztelő szkript
TesztesetX.in	1 KB	2018. 04. 18. 12:47	Tesztesetek bemenete
TesztesetX.out	1KB	2018. 04. 18. 12:47	Tesztesetek elvárt kimenete

Megjegyzés: a TesztesetX.in és TesztesetX.out fájlnevekben X helyén tetszőleges, {1, 2, ..., 33} halmazbeli szám állhat, így ezek a sorok egyenként 33 fájlt jelölnek.

10.1.2 Fordítás

1. Navigálunk valamelyen parancsossal (bash, PowerShell, stb.) a kitömörített fájlokat tartalmazó mappába!
2. Miután meggyőződtünk róla, hogy a JDK bin mappája hozzá van adva a PATH-hoz, adjuk ki a következő parancsokat (Windows alatt:)

```
mkdir .\class  
  
javac -d .\class .\AbstractField.java .\Crate.java  
.\\Direc  
tion.java .\Hole.java .\SwitchableHole.java .\Main.java  
.\\Moveable.java .\\SimpleField.java .\\SwitchField.java  
.\\TargetField.java .\\Wall.java .\\Worker.java
```

(Linux alatt értelemszerűen ugyanez, csak “\” helyett “/” karakterekkel.)

10.1.3 Futtatás

Futtatáshoz:

Windows alatt adjuk ki a következő parancsot:

```
java -cp .\class killer_sokoban.Main
```

(Linux alatt értelemszerűen ugyanez, csak “\” helyett “/” karakterekkel.)

Automatikus teszteléshez:

0. Győződjünk meg róla, hogy a Windows-ban engedélyezett a szkriptek futtatása, ez alapértelmezetten le van tiltva. Ebben az esetben a Powershell-t rendszergazdai módban indítva adjuk ki a következő parancsot:

```
Set-ExecutionPolicy Unrestricted
```

1. Egyetlen teszeset futtatásához indítsuk el a tesztelőszkriptet a következő paraméterekkel:

```
.\\tester.ps1 inputfájl outputfájl
```

Például: .\\tester.ps1 .\\Teszteset1.in .\\Teszteset1.out

VAGY

Minden teszeset futtatásához indítsuk a szkriptet paraméterek nélkül:

```
.\\tester.ps1
```

10.2 Tesztek jegyzőkönyvei

Bizonyos teszteket még az első futtatás előtt javítottunk. A hibák a következők voltak:

- az elvárt kimenetek a specifikálthoz képest néha rossz sorrendben írták ki az adattagokat
- a Fluidra néhol "null" választ vártunk "no fluid" helyett
- a játékos ereje double típusú, ezért a kimenetben megjelennek tizedes értékek is
- a kimenetekben nem lehetnek üres sorok, ezeket törlni kellett
- a bemenetben lehetnek üres sorok, de azokban még whitespace karakter sem lehetett, de némelyikben voltak space-ek
- Némely bemenetben lekértük összetört lánckód állapotát, utólagosan abban egyeztük meg, hogy ekkor a program azt válaszolja, hogy "This doesn't exist!".

10.2.1 Teszteset1

Teszteleő neve	Zalavári Márton
Teszt időpontja	2018.04.20 17:32

10.2.2 Teszteset2

Teszteleő neve	Zalavári Márton
Teszt időpontja	2018.04.20 17:33

10.2.3 Teszteset3

Teszteleő neve	Zalavári Márton
Teszt időpontja	2018.04.20 17:33

10.2.4 Teszteset4

Teszteleő neve	Zalavári Márton
Teszt időpontja	2018.04.20 17:33

10.2.5 Teszteset5

Teszteleő neve	Zalavári Márton
Teszt időpontja	2018.04.20 17:48

Teszteleő neve	Zalavári Márton
Teszt időpontja	2018.04.20 17:34
Teszt eredménye	Hiba
Lehetséges hibaok	Kimeneti teszt hibás, nem írja ki a kapcsoló sHole adattagját
Változtatások	Kimeneti fájl javítása.

10.2.6 Teszteset6

Teszteleő neve	Zalavári Márton
----------------	-----------------

Teszt időpontja	2018.04.20 17:40
------------------------	------------------

10.2.7 Teszteset7

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 17:48

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 17:42
Teszt eredménye	Hiba
Lehetséges hibaok	Kimeneti teszt hibás, nem írja ki a lyuk opened adattagját
Változtatások	Kimeneti fájl javítása.

10.2.8 Teszteset8

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 17:48

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 17:44
Teszt eredménye	Hiba
Lehetséges hibaok	Kimeneti teszt hibás, nem írja ki a lyuk opened adattagját
Változtatások	Kimeneti fájl javítása.

10.2.9 Teszteset9

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 17:36

10.2.10 Teszteset10

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 17:47

10.2.11 Teszteset11

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 17:48
Teszt eredménye	Hiba
Lehetséges hibaok	A tesztbemenet rossz helyre akarja rakni a moveableöket.
Változtatások	A tesztek javítása.

10.2.12 Teszteset12

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:15

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:10

Teszt eredménye	Hiba
Lehetséges hibaok	A tesztbemenet rossz helyre akarja rakni a moveableöket.
Változtatások	A tesztek javítása.

10.2.13 Teszteset13

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:17

10.2.14 Teszteset14

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:17

10.2.15 Teszteset15

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:17

10.2.16 Teszteset16

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:17

10.2.17 Teszteset17

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:17

10.2.18 Teszteset18

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:28

10.2.19 Teszteset19

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:21

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:19
Teszt eredménye	Hiba
Lehetséges hibaok	Hiba a teszt kimenetében
Változtatások	Egy : helyett . szerepelt egy helyen. Ezt javítottam.

10.2.20 Teszteset20

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:22

10.2.21 Teszteset21

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:22

10.2.22 Teszteset22

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:22

10.2.23 Teszteset23

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:28

Tesztelő neve	Zalavári Márton
Teszt időpontja	2018.04.20 18:22
Teszt eredménye	Hiba
Lehetséges hibaok	A bemenet más objektum statját kérte le, mint ami a kimenetben benne volt.
Változtatások	A bemenetet kiegészítettem még egy stat parancsal, és a kimenetet másik már meglévő parancs kimenetével.

10.2.24 Teszteset24

Tesztelő neve	Schulcz Ferenc
Teszt időpontja	2018. 04. 18. 13:08

Tesztelő neve	Schulcz Ferenc
Teszt időpontja	2018. 04. 18. 13:06
Teszt eredménye	Hiba
Lehetséges hibaok	Az elvárt kimenetben rossz mezőnél szerepelt a felső szomszédnál
Változtatások	Az elvárt kimenet átírása

10.2.25 Teszteset25

Tesztelő neve	Schulcz Ferenc
Teszt időpontja	2018. 04. 18. 13:15

Tesztelő neve	Schulcz Ferenc
Teszt időpontja	2018. 04. 18. 13:12
Teszt eredménye	Hiba
Lehetséges hibaok	Az elvárt kimenetben rossz mezőnél szerepelt a felső szomszédnál
Változtatások	Az elvárt kimenet átírása

Tesztelő neve	Schulcz Ferenc
Teszt időpontja	2018. 04. 18. 13:08
Teszt eredménye	Hiba
Lehetséges hibaok	A putOil() függvény is mézet tett a pályára.
Változtatások	A függvényt kijavítottam.

10.2.26 Teszteset26

Tesztelő neve	Schulcz Ferenc
Teszt időpontja	2018. 04. 18. 13:15

Tesztelő neve	Schulcz Ferenc
Teszt időpontja	2018. 04. 18. 13:12
Teszt eredménye	Hiba
Lehetséges hibaok	Az elvárt kimenetben rossz mezőnél szerepelt a felső szomszédnál
Változtatások	Az elvárt kimenet átírása

10.2.27 Teszteset27

Tesztelő neve	Schulcz Ferenc
Teszt időpontja	2018. 04. 18. 13:44

Tesztelő neve	Schulcz Ferenc
Teszt időpontja	2018. 04. 18. 13:20
Teszt eredménye	Hiba
Lehetséges hibaok	A kapcsolható lyuk és a kapcsolója túl közel volt a tesztnél, a munkás egy lépés közben a kapcsolóra tolta a lánát és a lyukba esett.
Változtatások	Rövid konzultáció után úgy döntöttünk, hogy ez a viselkedés a természetesnek megfelelő, és a tesztesetben távolabb vittük a lyukat a kapcsolótól.

Tesztelő neve	Schulcz Ferenc
Teszt időpontja	2018. 04. 18. 13:12
Teszt eredménye	Hiba
Lehetséges hibaok	Az elvárt kimenetben rossz mezőnél szerepelt a felső szomszédnál
Változtatások	Az elvárt kimenet átírása

10.2.28 Teszteset28

Tesztelő neve	Benkő Csaba
Teszt időpontja	2018. 04. 21. 13:10

10.2.29 Teszteset29

Tesztelő neve	Benkő Csaba
Teszt időpontja	2018. 04. 21. 13:10

10.2.30 Teszteset30

Tesztelő neve	Benkő Csaba
Teszt időpontja	2018. 04. 21. 13:10

10.2.31 Teszteset31

Tesztelő neve	Benkő Csaba
Teszt időpontja	2018. 04. 21. 13:10

10.2.32 Teszteset32

Tesztelő neve	Benkő Csaba
Teszt időpontja	2018. 04. 21. 13:10

10.2.33 Teszteset33

Tesztelő neve	Benkő Csaba
Teszt időpontja	2018. 04. 21. 13:10

10.3 Értékelés

Tag neve	Munka százalékban
Benkő	23
Schulcz	26
Takács	25
Zalavári	26

10.4 Napló

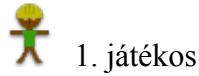
Kezdet	Időtartam	Résznevők	Leírás
2018.04.11. 11:00	1 óra	Teljes csapat	Értekezlet. Tennivalók felosztása, problémák megbeszélése
2018.04.14. 10:00	3 óra	Zalavári	Prototípus implementálása
2018.04.14. 10:00	30 perc	Takács	Logger osztály kiszedése
2018.04.14. 11:00	30 perc	Takács	printStat függvények megírása
2018.04.14. 15:00	45 perc	Takács	Tesztesetek txt-be szedése
2018.04.14. 15:00	1,5 óra	Zalavári	Prototípus implementálása, javítások
2018.04.17. 17:30	1 óra	Zalavári	Triviális javítások a tesztesetekben, azok átnevezése, és verziókövetés alá vetése
2018. 04 18. 13:00	1 óra	Schulcz	A folyadékokkal kapcsolatos tesztek

			ellenőrzése, működésre bírása
2018.04.20. 17:30	1 óra	Zalavári	Tesztelek
2018. 04. 20. 18:00	15 perc	Schulcz	Apró változtatások a dokumentumban, két nappal azelőtti teszteredmények felvitele
2018. 04. 21. 13:00	30 perc	Benkő	Maradék tesztek futtatása, jegyzőkönyvezése
2018. 04. 22. 13:00	30 perc	Schulcz	10.1 szakasz megírása

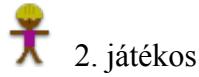
11. Grafikus felület specifikációja

11.1 A grafikus interfész

A pálya megrajzolása során a következő képeket fogjuk használni:



1. játékos



2. játékos



Láda



Méz



Olaj



Lyuk



Járható mező



Kapcsoló



Célmező



Fal

Ez alapján egy példa pálya így nézhet ki:



(Természetesen a pálya tényleges mérete változtatható.)

Az alkalmazás egyetlen ablakból áll majd, ebben a pályát úgy rajzoljuk ki, hogy minden mező 32x32 pixel méretű legyen, és a raktár középre legyen zárva.

A játék végén megtörténik a pontok kiírása, egy ehhez hasonló formában:



A pálya képe alatt lesz még egy bottombar, amin látszik a játékosok pontja, “w1: x points w2: y points” formában.

11.2 A grafikus rendszer architektúrája

A felület tervezése során az MVC tervezési mintát igyekeztünk követni.

A Controller szerepét a Main osztályunk tölti be, ami az alkalmazás motorja, a felhasználói interakciók kezelője, és a View és Modell létrehozója és otthonát adója is egyben. Kezeli az felhasználó által generált eseményeket, melyeket a modellbe továbbirányítva módosíthatja annak állapotát. minden ilyen módosítás után felszólítja a View osztályt, hogy frissítse a megjelenést.

A View részt a View osztály egy példánya testesíti meg. Kezeli a megjelenítéshez szükséges erőforrásokat, illetve a pálya méretének ismeretében frissítés esetén, minden rajzolásához a Main osztályhoz fordul, hogy lekérdezze a modellbeli elemektől az állapotukat, és függvény hívásokkal jelezzék számára, hogy most mi a kirajzolandó objektum.

A Modell az eddigi modell, azonban mostantól implementálja a IViewable interfész. Ezáltal lehet felszólítani, hogy eljött a rajzolás ideje, ezért jelezzen vissza, hogy mit/miket kell most rajzolni. Ez a visszajelzés a paraméterül kapott view-n tetszőleges számú (az értelmes keretek között) függvény hívásból állhat.

11.2.1 A felület működési elve

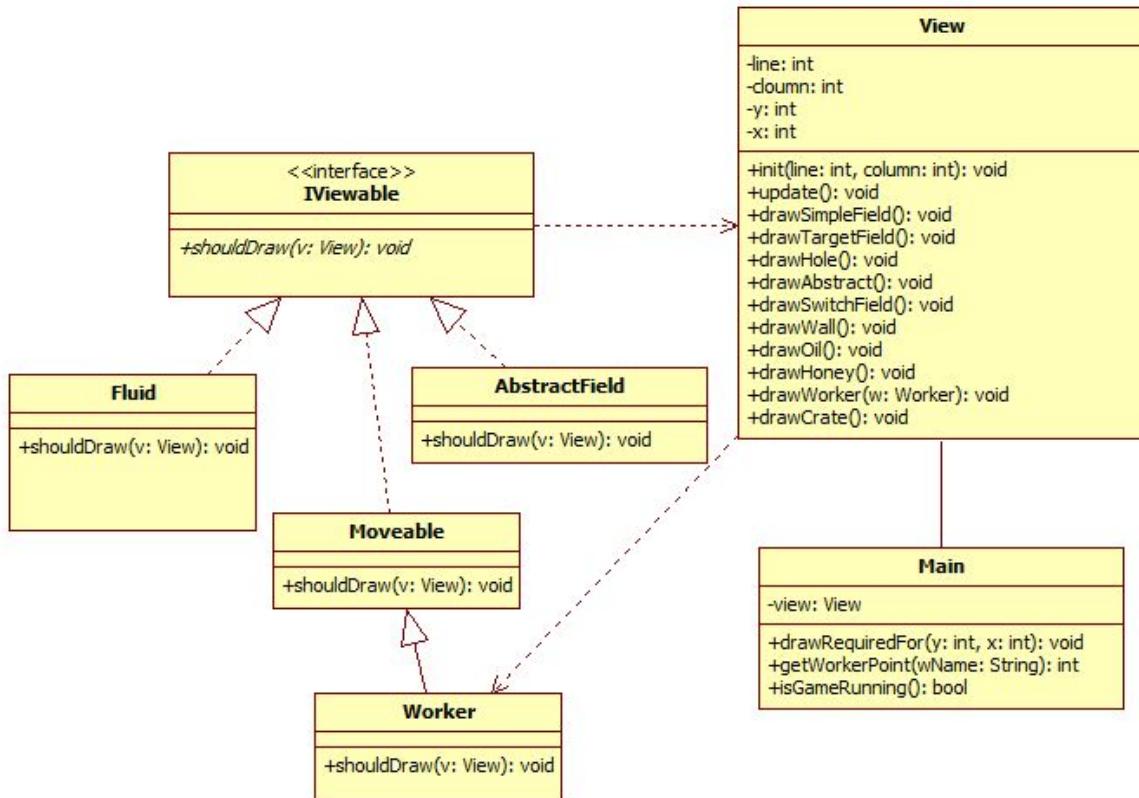
A megjelenítés alapelve pull alapú. Amikor a Controller funkciót betöltő osztály (Main) módosította a modellt, meghívja a View osztály update() metódusát.

Ennek hatására a View osztály a modellen, konkrétabban minden AbstractFielden meghívja a shouldDraw() függvényt az IViewable interfészen keresztül (pull), ezután pedig válaszokat vár, hogy miket kell neki kirajzolnia.

Ezután lekérdezi a pontokat a játékosuktól, és a megfelelő formában megjeleníti.

Végül, megkérdezi a Maintől, hogy vége van-e a játéknak, és ennek megfelelően módosítja még a kirajzolt állapotot.

11.2.2 A felület osztály-struktúrája



Ezen felül természetes minden AbstractField, Fluid és Moveable leszármazott kénye-kedve szerint felüldefiniálhatja a shouldDraw(..) függvényt.

11.3 A grafikus objektumok felsorolása

11.3.1 Main

- **Felelősség**

A program maga, és egyben a vezérlő. minden adat tagja és függvénye statikus.

- **Ősosztályok**

-

- **Interfészek**

-

- **Attribútumok**

- **-view:** View: a nézetért és rajzolásért felelős osztály példánya

- **Metódusok**
 - **+drawRequiredFor(int y, int x): void**: A paraméterként kapott koordinátákból egy mezőnevet parse-ol, majd az így kapott nevű mezőnek előkeresi a referenciáját, és meghívja rajta a shouldDraw() függvényt v paraméterrel.
 - **+getWorkerPoint(String wName): int**: Visszaadja a paraméterként kapott nevű munkás aktuális pontszámát.
 - **+isGameRunning(): bool**: Visszaadja, hogy éppen megy-e a játék, vagy már vége lett.

11.3.2 View

- **Felelősség**

A grafikus felület elemeinek megjelenítése.

- **Ősosztályok**
-

- **Interfészek**
-

- **Attribútumok**

- **-line: int**: a létrehozott pálya sorainak számát tárolja
- **-column: int**: a létrehozott pálya oszlopainak számát tárolja
- **-y: int**: Az aktuálisan kirajzolandó objektum sorának a száma
- **-x : int**: Az aktuálisan kirajzolandó objektum oszlopának a száma

- **Metódusok**

- **+init(line: int, column: int): void**: A megjelenítendő pálya paramétereinek beállítása.
- **+update(): void**: Ennek hatására a pálya minden egyes eleme újra kirajzolásra kerül.
- **+drawSimpleField(): void**: Az x, y koordinátákkal meghatározott mezőre a SimpleField-nek megfelelő grafikus elem kirajzolása
- **+drawTargetField(): void**: Az x, y koordinátákkal meghatározott mezőre a TargetField-nek megfelelő grafikus elem kirajzolása
- **+drawHole(): void**: Az x, y koordinátákkal meghatározott mezőre a Hole-nak megfelelő grafikus elem kirajzolása
- **+drawAbstract(): void**: Az x, y koordinátákkal meghatározott mezőre az Abstractfieldnek megfelelő grafikus elem kirajzolása
- **+drawSwitchField(): void**: Az x, y koordinátákkal meghatározott mezőre a SwitchField-nek megfelelő grafikus elem kirajzolása
- **+drawWall(): void**: Az x, y koordinátákkal meghatározott mezőre a Wall-nek megfelelő grafikus elem kirajzolása
- **+drawOil(): void**: Az x, y koordinátákkal meghatározott mezőre a Oil-nak megfelelő grafikus elem kirajzolása (a már ott lévő tetejére, amely még kilátszik alól)
- **+drawHoney(): void**: Az x, y koordinátákkal meghatározott mezőre a Honey-nak megfelelő grafikus elem kirajzolása (a már ott lévő tetejére, amely még kilátszik alól)

- **+drawWorker(w: Worker): void:** Az x, y koordinátákkal meghatározott mezőre a Worker-nek megfelelő grafikus elem kirajzolása (a már ott lévő tetejére, amely még kilátszik alól). A w paramétert azért kell átdannunk, hogy tudjuk, hogy pontosan melyik dolgozót kell most kirajzolni.
- **+drawCrate(): void:** Az x, y koordinátákkal meghatározott mezőre a Crate-nek megfelelő grafikus elem kirajzolása (a már ott lévő tetejére, amely még kilátszik alól)

11.3.3 IViewable

- **Felelősség**

Interfész, melyet a megtekinthető objektumoknak meg kell valósítaniuk, hogy a grafikát kezelő osztály kommunikálhasson velük.

- **Ősosztályok**
-

- **Interfészek**
-

- **Attribútumok**
-

- **Metódusok**

- **+shouldDraw(v: View): void:** Akkor hívandó, amikor az objektum újbóli kirajzolása szükséges. A paraméterül kapott View-n meghívja azt a függvényt, ami az adott objektum kirajzolását elvégzi.

11.3.4 AbstractField

- **Interfészek**

IViewable

- **Metódusok**

- **+shouldDraw(v: View): void:** A paraméterül kapott View-n meghívja annak drawAbstract() függvényét.

11.3.5 SimpleField

- **Interfészek**

IViewable

- **Metódusok**

- **+shouldDraw(v: View): void:** A paraméterül kapott View-n meghívja annak drawSimpleField() függvényét, majd a fluidján, aztán a Moveable-jén, ha az nem null, rendre meghívja a shouldDraw() függvényeket, a v paraméterrel.

Végül meghívja saját ősosztálya shouldDraw() függvényét v paraméterrel.

11.3.6 TargetField

- **Interfészek**

IViewable

- **Metódusok**

- **+shouldDraw(v: View): void**: A paraméterül kapott View-n meghívja annak drawTargetField() függvényét, majd az őosztálya shouldDraw() függvényét v paraméterrel.

11.3.7 Hole

- **Interfészek**

IViewable

- **Metódusok**

- **+shouldDraw(v: View): void**: A paraméterül kapott View-n meghívja annak drawHole() függvényét, majd az őosztálya shouldDraw() függvényét v paraméterrel.

11.3.8 SwitchableHole

- **Interfészek**

IViewable

- **Metódusok**

- **+shouldDraw(v: View): void**: Ha a lyuk nyitva van, meghívja a paraméterül kapott View-n annak drawHole() függvényét. Majd minden esetben meghívja saját őosztályának a shouldDraw() függvényét v paraméterrel.

11.3.9 SwitchField

- **Interfészek**

IViewable

- **Metódusok**

- **+shouldDraw(v: View): void**: A paraméterül kapott View-n meghívja annak drawSwitchField() függvényét, majd az őosztálya shouldDraw() függvényét v paraméterrel.

11.3.10 Wall

- **Interfészek**

IViewable

- **Metódusok**

- **+shouldDraw(v: View): void**: A paraméterül kapott View-n meghívja annak drawWall() függvényét, majd az őosztálya shouldDraw() függvényét v paraméterrel.

11.3.11 Fluid

- **Interfészek**

IViewable

- **Metódusok**

- **+shouldDraw(v: View): void:** Nem csinál semmit. (A leszármazottak miatt kell.)

11.3.12 Oil

- **Interfészek**

IViewable

- **Metódusok**

- **+shouldDraw(v: View): void:** A paraméterül kapott View-n meghívja annak drawOil() függvényét, majd az ősosztálya shouldDraw() függvényét v paraméterrel.

11.3.13 Honey

- **Interfészek**

IViewable

- **Metódusok**

- **+shouldDraw(v: View): void:** A paraméterül kapott View-n meghívja annak drawHoney() függvényét, majd az ősosztálya shouldDraw() függvényét v paraméterrel.

11.3.14 Worker

- **Interfészek**

IViewable

- **Metódusok**

- **+shouldDraw(v: View): void:** A paraméterül kapott View-n meghívja annak drawWorker() függvényét, “this” paraméterrel.

11.3.15 Crate

- **Interfészek**

IViewable

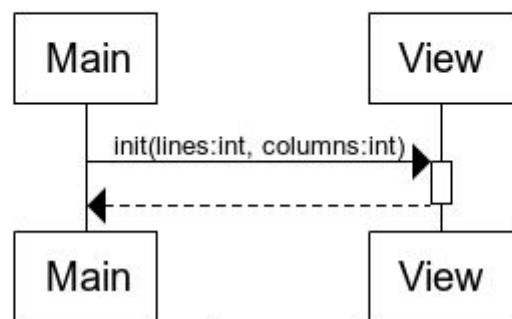
- **Metódusok**

- **+shouldDraw(v: View): void:** A paraméterül kapott View-n meghívja annak drawCrate() függvényét

11.4 Kapcsolat az alkalmazói rendszerrel

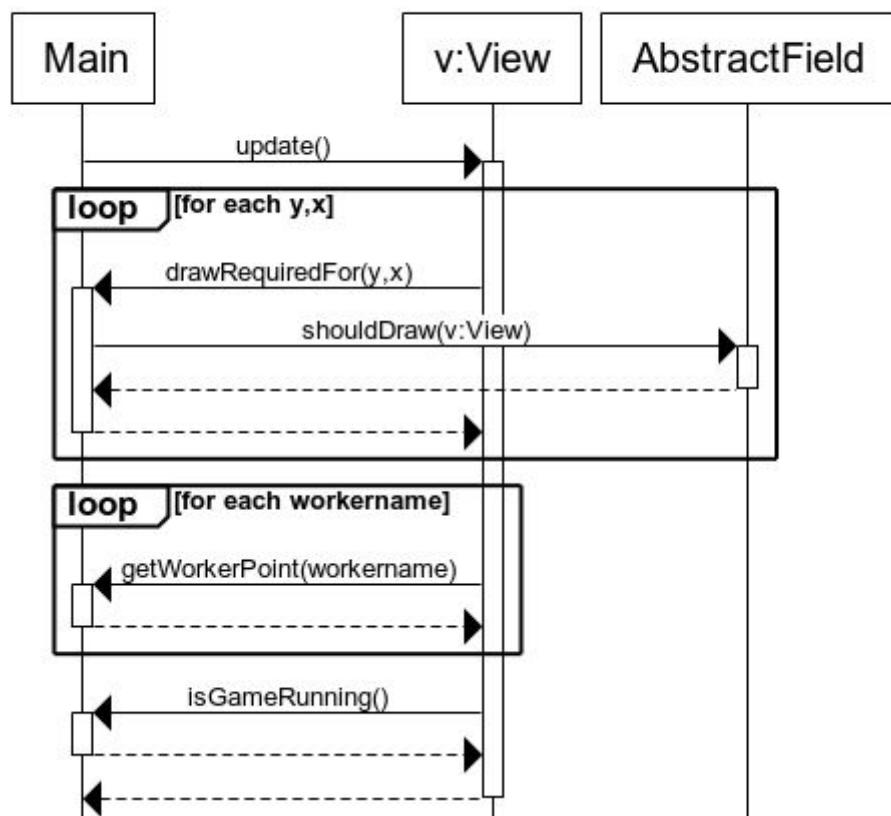
11.4.1 Initialization of View

Initialization of View

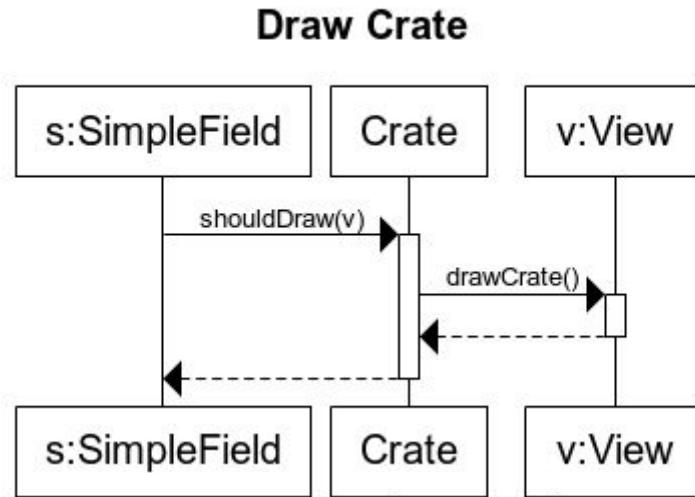


11.4.2 Update View

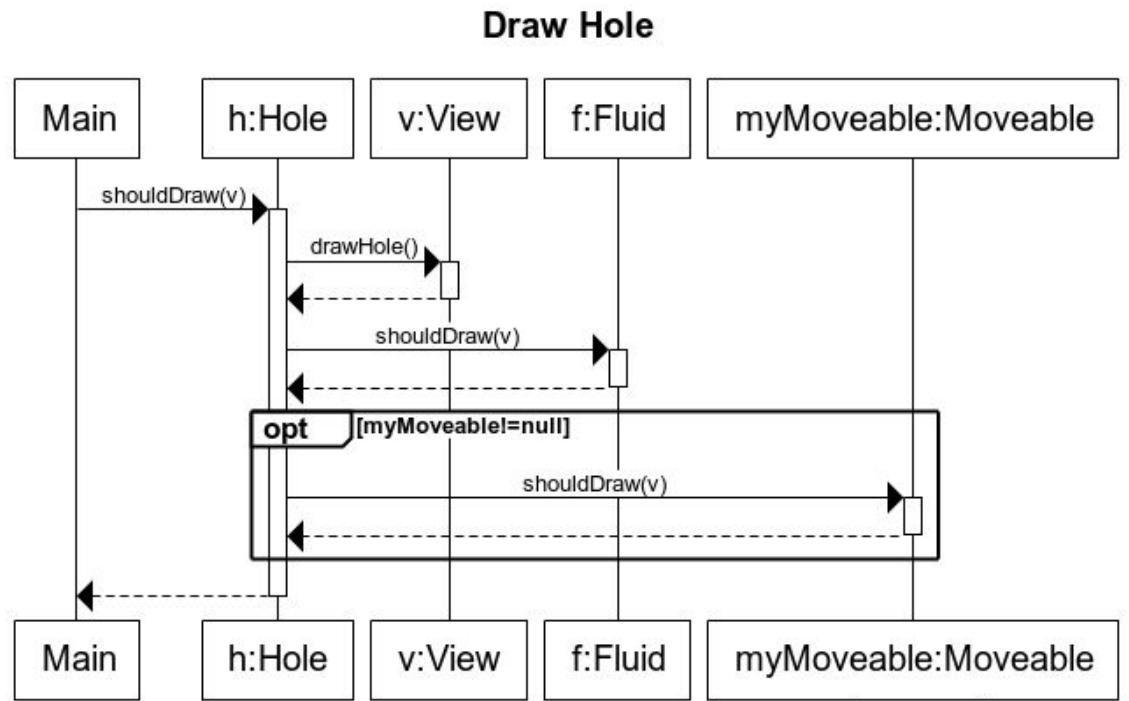
Update View



11.4.3 Draw Crate

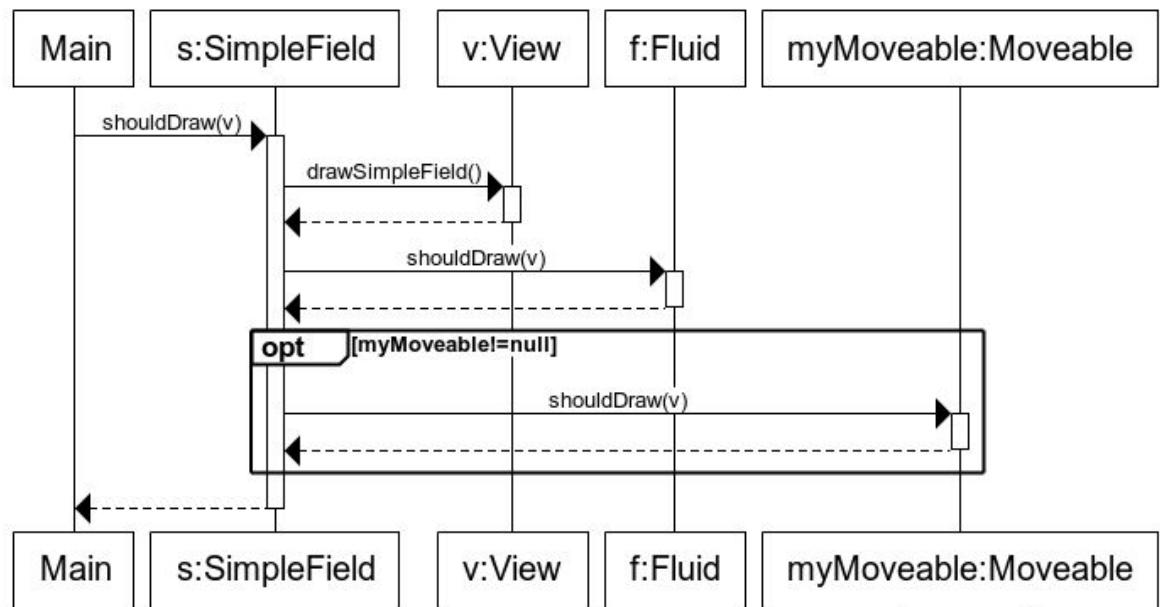


11.4.4 Draw Hole



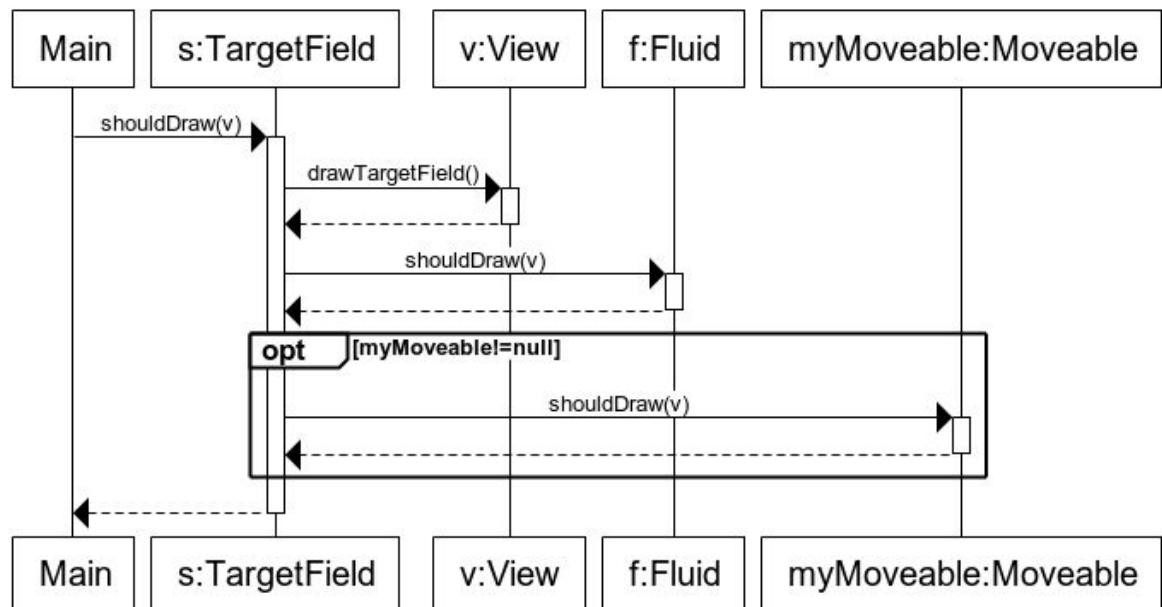
11.4.5 Draw SimpleField

Draw SimpleField



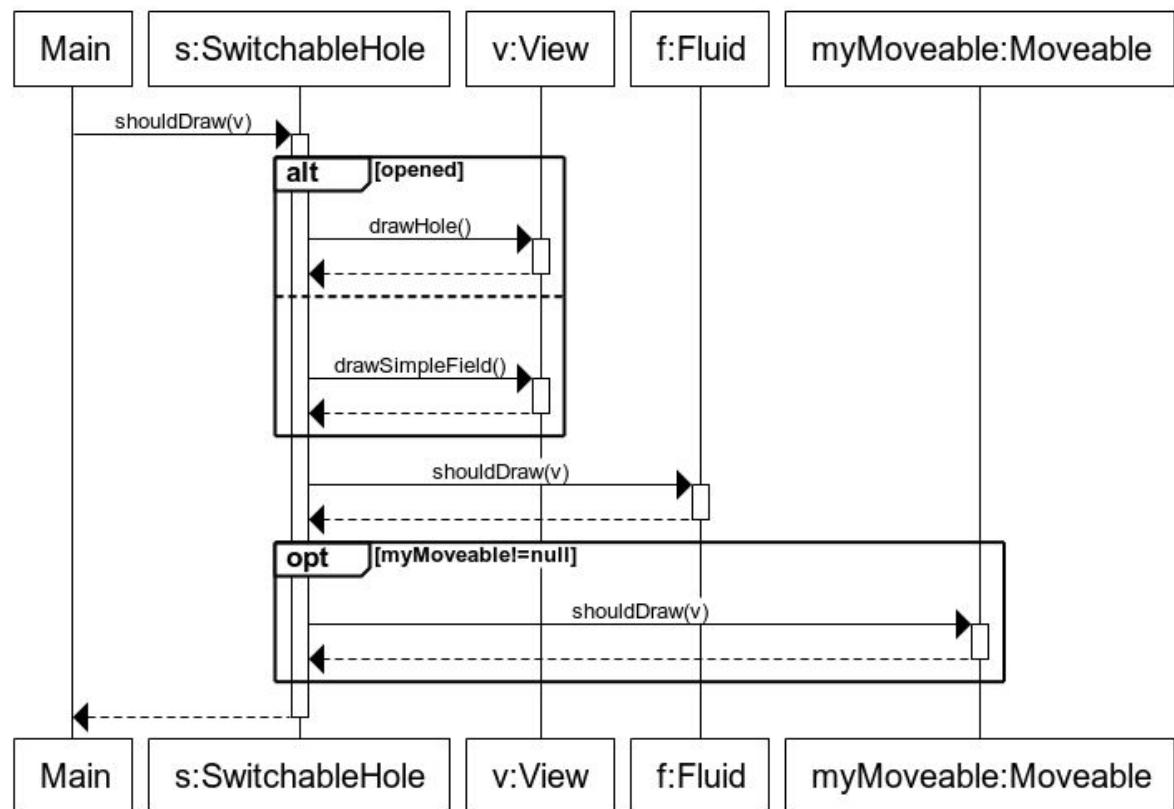
11.4.6 *Draw TargetField*

Draw TargetField



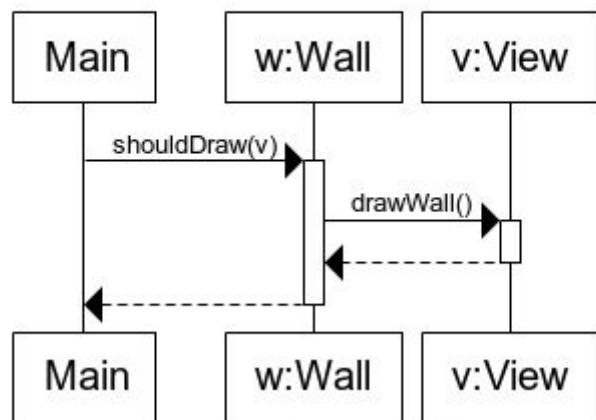
11.4.7 *Draw SwitchableHole*

Draw SwitchableHole



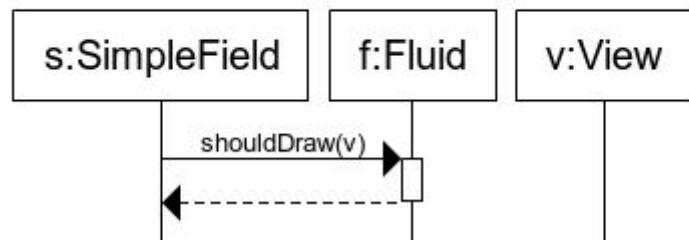
11.4.8 Draw Wall

Draw Wall



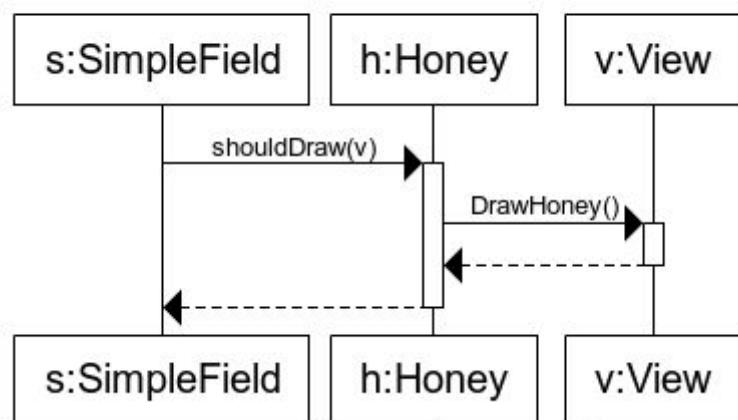
11.4.9 Draw Fluid

Draw Fluid



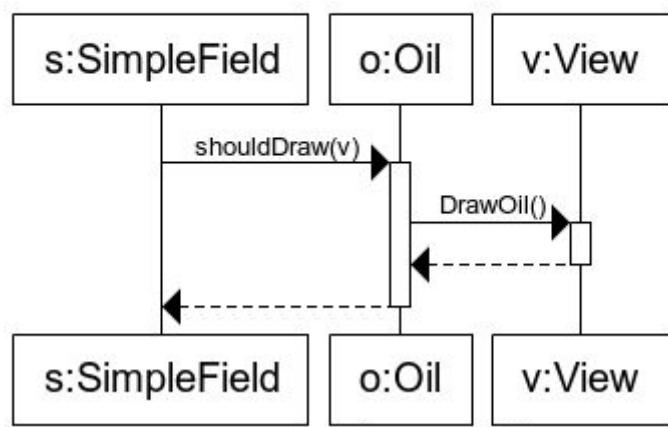
11.4.10 Draw Honey

Draw Honey



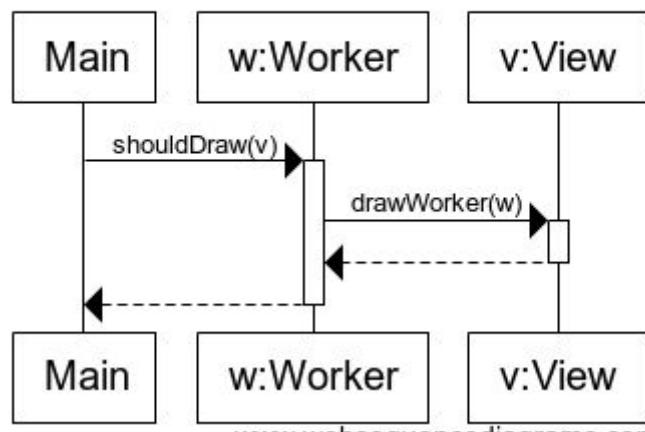
11.4.11 Draw Oil

Draw Oil



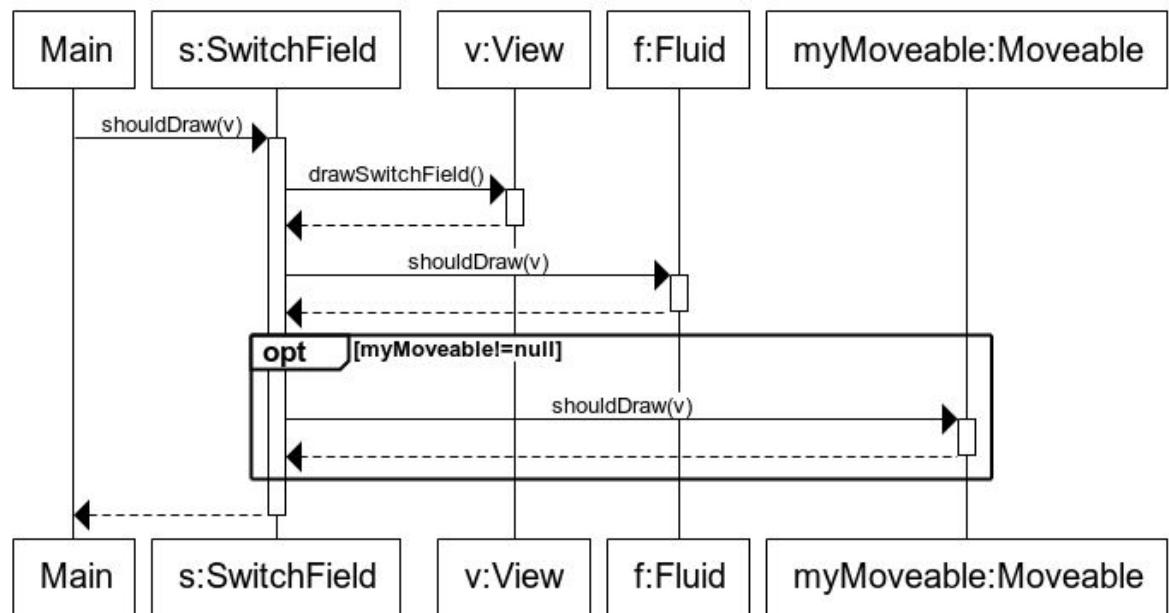
11.4.12 Draw Worker

Draw Worker



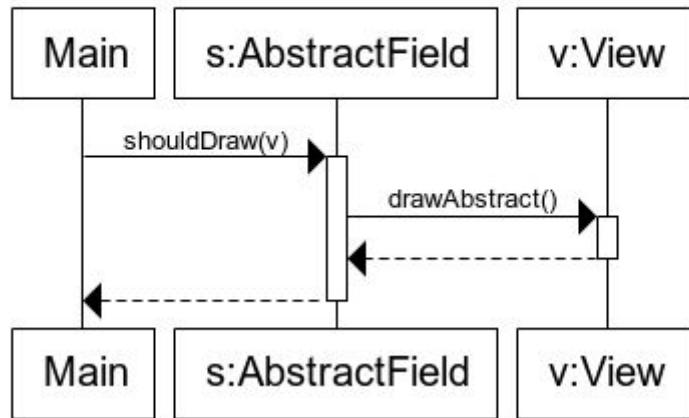
11.4.13 Draw SwitchField

Draw SwitchField



11.4.14 Draw AbstractField

Draw AbstractField



11.5 Napló

Kezdet	Időtartam	Résznevők	Leírás
2018.04.27. 14:10	50 perc	Csapat	Értekezlet. Döntés: A grafikával kapcsolatos feladatok és komponensek tervezése
2018.04.28. 14:00	1 óra	Benkő	Tevékenység: Az új és a módosult osztályok leírásainak elkészítése
2018.04.28. 19:30	10 perc	Benkő	Osztályleírások finomítása
2018.04.29. 11:00	2.5 óra	Zalavári	Az eddig megbeszéltek vázának kódbeli implementálása
2018.04.30.17:00	1.5 óra	Takács	Szekvenciadiagramok
2018. 04. 30. 22:30	1 óra	Schulecz	A játék során használt képfájlok elkészítése

2018. 04. 30. 23:00	1 óra	Schulcz	Finomítások a képi világban, illusztrációk elkészítése, dokumentálás
2018.05.01.10:15	2 óra	Takács	Szekvenciadiagramok javítása, beillesztése, osztálydiagram
2018.05.01.11:00	1.5 óra	Zalavári	Dokumentáció javítása, kiegészítése, 11.2-es rész elkészítése
2018.05.01. 19:30	1 óra	Schulcz	Apróbb javítások a tervezekben

13. Grafikus változat beadása

13.1 Fordítási és futtatási útmutató

13.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
AbstractField.java	2 KB	2018. 03. 14. 21:32	A mezők ősosztálya
Crate.java	2 KB	2018. 03. 14. 21:32	A láda osztály
Direction.java	1 KB	2018. 03. 14. 21:32	Az irányok enum-ja
Hole.java	1 KB	2018. 03. 14. 21:32	A lyukak osztálya
Main.java	7 KB	2018. 03. 14. 21:32	A szimulációt irányító osztály
Moveable.java	3 KB	2018. 03. 14. 21:32	A mobok ősosztálya
SimpleField.java	4 KB	2018. 03. 14. 21:32	Az egyszerű mező osztály
SwitchableHole.java	2 KB	2018. 03. 14. 21:32	A kapcsolható lyuk osztály
SwitchField.java	2 KB	2018. 03. 14. 21:32	A kapcsoló osztály
TargetField.java	2 KB	2018. 03. 14. 21:32	A célmező osztály
Wall.java	1 KB	2018. 03. 14. 21:32	A fal osztály
Worker.java	2 KB	2018. 03. 14. 21:32	A munkások osztály
Testmap1.in	1 KB	2018. 05. 05. 17:32	Egy teszteléshez való pálya
Crate.png	1 KB	2018. 05. 01. 12:16	Láda grafikus megjelenítése
Hole.png	1 KB	2018. 05. 01. 12:16	Lyuk grafikus megjelenítése
Honey.png	1 KB	2018. 05. 01. 12:16	Méz grafikus megjelenítése
Oil.png	1 KB	2018. 05. 01. 12:16	Olaj grafikus megjelenítése
SimpleField.png	1 KB	2018. 05. 01. 12:16	SimpleField grafikus megjelenítése
SwichField.png	2 KB	2018. 05. 01. 12:16	SwitchField grafikus megjelenítése
TargetField.png	1 KB	2018. 05. 01. 12:16	TargetField grafikus megjelenítése

w1.png	2 KB	2018. 05. 01. 12:16	Egyik munkás grafikus megjelenítése
w2.png	2 KB	2018. 05. 01. 12:16	Másik munkás grafikus megjelenítése
Wall.png	1 KB	2018. 05. 01. 12:16	Fal grafikus megjelenítése

13.1.2 Fordítás és telepítés

1. Navigálunk valamilyen parancssorral (bash, PowerShell, stb.) a kitömörített fájlokat tartalmazó mappába!
2. Miután meggyőződtünk róla, hogy a JDK bin mappája hozzá van adva a PATH-hoz, adjuk ki a következő parancsokat (Windows alatt):

```
mkdir .\class
```

```
javac -d .\class -encoding utf8 .\AbstractField.java
.\Crate.java .\Direction.java .\Hole.java
.\SwitchableHole.java .\Main.java .\Moveable.java
.\SimpleField.java .\SwitchField.java .\TargetField.java
.\Wall.java .\Worker.java .\View.java
```

(Linux alatt értelemszerűen ugyanez, csak “\” helyett “/” karakterekkel.)

13.1.3 Futtatás

Futtatáshoz:

Windows alatt adjuk ki a következő parancsot:

```
java -cp .\class killer_sokoban.Main
```

(Linux alatt értelemszerűen ugyanez, csak “\” helyett “/” karakterekkel.)

13.2 Értékelés

Tag neve	Munka százalékban
Schulcz	25
Zalavári	25
Takács	25
Benkő	25

13.3 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018. 05. 13. 10:00	1 óra	Takács	Kódolás
2018. 05. 13. 16:00	1 óra	Schulcz	Bugfix a játék vége detektálásban
2018. 05. 13., 21:06	3 perc	Benkő	Fájllista, fordítási és futtatási útmutató elkészítése az előző beadások alapján.

14. Összefoglalás

14.1 A projektre fordított összes munkaidő

Tag neve	Munkaidő (óra)
Schulcz	56 óra 15 perc
Zalavári	59 óra 55 perc
Takács	35 óra 5 perc
Benkő	28 óra 33 perc
Összesen	179 óra 48 perc

- A feltöltött programok forrássorainak száma**

Fázis	KódSOROK száma
Szkeleton	1355
Prototípus	907
Grafikus változat	1649
Összesen	3911

14.2 Projekt összegzés

14.2.1 Mit tanultak a projektből konkrétan és általában?

Jobban megtanultunk csapatban dolgozni, és gyakorolhattuk az objektumorientált tervezést és programozást.

14.2.2 Mi volt a legnehezebb és a legkönnyebb?

Legnehezebb volt megtalálni az időt, amit közös munkára tudtunk szánni, akár értekezlet formájában, akár úgy, hogy tudtuk, hogy más a mi munkánk eredményére vár. Legkönnyebb volt implementálni a már megtervezett programot.

Konkrét kihívás tekintetében a játék végének detektálása okozott nehéz perceket és szennedéssel teli módosításokat. Sok bosszúságot megspóroltunk volna, ha már a projekt elején eszünkbe jut a végül megvalósított módszer.

14.2.3 Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?

Nagyjából igen. Az elején talán többet dolgoztunk, mert kevésbé voltunk rutinosak, és tényleg a nulláról kellett indulni.

14.2.4 Ha nem, akkor hol okozott ez nehézséget?

Főleg a projekt elején. (Kis részben a végén, a ZH-k mellett.)

14.2.5 Milyen változtatási javaslatuk van?

Néhol kicsit inkonzisztensnek éreztük az értékelést. Megkérdeztük például még megvalósítás előtt azt, hogy adhatjuk-e mindenhol paraméterként a tolást végző munkást, és hiába egyeztünk meg ebben a megoldásban, a következő beadásnál mégis pontlevonást kaptunk érte.

Az értékelés is lehetne egységesebb. A laborvezetők között is nagy különbségek vannak. Előfordult, hogy más labveznél lévő csapatoknál az a megoldás is közel maximum pontos, amit nálunk külön kiemelnek, hogy ne úgy csináljunk.

14.2.6 Milyen feladatot ajánlanának a projektre?

Chat kliens.

14.2.7 Egyéb kritika és javaslat

-