2. Követelmény, projekt, funkcionalitás

2.1 Bevezetés

2.1.1 Cél

A dokumentáció ismerteti a projekt alapvető felépítését és követelményeit. Meghatározza a munka során követendő irányelveket.

2.1.2 Szakterület

Az elkészítendő szoftver a híres Sokoban nevű játékprogram, így kifejezett szakterülete nincs, bárki játszhat vele. Egyetlen célja a végfelhasználók szórakoztatása. Ez egy logikai játék - ráadásul elég nehéz, NP komplexitású - kevés akcióval, gondolkodni szerető játékosokat céloz meg.

2.1.3 Definíciók, rövidítések

aktor: a használati esetekben külső félként résztvevő szereplők által játszott összetartozó szerepek együttese
BME: Budapesti Műszaki és Gazdaságtudományi Egyetem
Git: egy verziókezelő rendszer, ami arra szolgál, hogy fájlok (programok, dokumentációk stb.) különböző verzióit kordában tartsa, elkönyvelje, tárolja és megossza
GitHub: egy Git-re épülő internetes szolgáltatás
IIT: Irányítástechnika és Informatika Tanszék
JDK: Java Development Kit - Java fejlesztőeszköz
JRE: Java Runtime Environment - Java futtatókörnyzet
NP komplexitás: egy probléma megoldásának nehézségét jellemzi, ez a kategória kifejezetten bonyolult
proto: prototípus - Olyan állapota a programnak, amikor amikor a grafikus felületen kívül minden belső működés meg van valósítva
szoftver: számítógépen futtatható program
szkeleton: olyan állapota a programnak, amikor még csak a belső felépítése van kész
UML: Unified Modeling Language - modellező eszköz
Use-case: leírja a rendszer és az őt felhasználó külső szereplők (aktorok) közötti akciók és reakciók (válaszok) sorozatát, az interakciókat

2.1.4 Hivatkozások

Szoftver projekt laboratórium - https://www.iit.bme.hu/targyak/BMEVIIIAB02

2.1.5 Összefoglalás

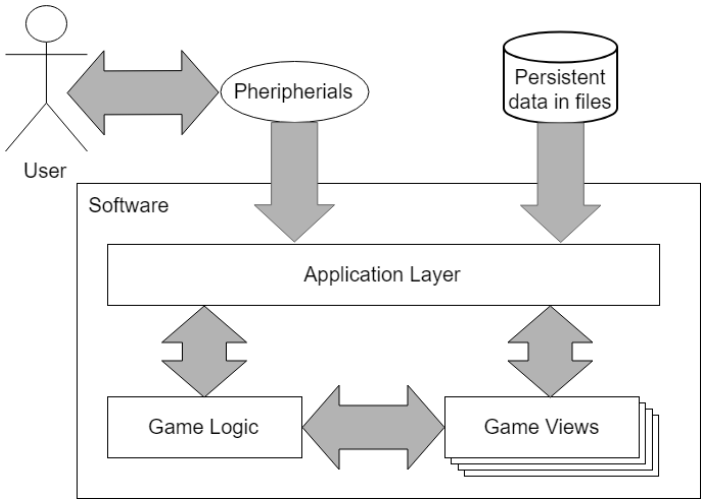
A dokumentum további részeiben található:

- Áttekintés, ami nagyvonalakban bemutatja a szoftvert
- Követelmények leírása, melyek alapján később a szoftver elkészül
- Use-case-ek felsorolása
- Szótár a nem közérthető szavak, illetve nem hagyományos értelemben használt szavak jelentésével
- Projekt megvalósításának a terve
- Napló a végzett munkáról

2.2 Áttekintés

2.2.1 Általános áttekintés

A szoftver legmagasabb szintű képe, mely esetünkben követi a Game Coding Complete c. könyvben (Mike McShaffry, David Graham, ISBN: 978-1133776574) - és rengeteg játékban és játékmotorban - prezentált architektúrát. Ez a fajta felbontás nagy rugalmasságot nyújt a rendszernek.



Megjegyzés: Az esemény és üzenet ezen kontextusban ekvivalens az architektúra szempontjából.

User: Játékos, játékosok, akik a szoftvert használják.

Pheripherials (perifériák): A számítógéphez kapcsolódó eszközök, melyek audiovizuálisan prezentálják a játékot (monitor, hangszoró) vagy a játék irányítását teszik lehetővé (billentyűzet).

Persistent data in files: Minden játékadat, amelyet fájl(ok)ban tárol a szoftver, mint a pályák vagy grafikus elemek.

Application Layer: A szoftver azon része, mely a perifériákkal és a game view-kkal tartja a kapcsolatot. Betölt file-okat, billentyűzetről olvas és a rajzolás legalacsonyabb szintjét végzi (operációs rendszer szinten). A perifériáktól kapott jelzéseket, például billentyű lenyomást elküldi a Game View-knak üzenetként.

Game Views: A játékra egy adott nézet. Ilyen nézet lehet a grafikai megjelenítés, a játékosok irányítása stb. Ha például az irányítás nézet megkapja az első játékos balra mozgató billentyűjének az üzenetét, akkor az átfordítja “Játékos 1 balra lép” üzenetté, és elküldi a Game Logicnak.

Elméletileg, ha készítenénk egy View-t, ami nem csinál mást, mint rögzíti a perifériális eseményeket, és egy másik View pedig ezeket időrendben lekezeli, létrehoztunk egy egyszerű visszajátszási rendszert, amely hibajavításnál például jól jöhet.

Game Logic: A játék funkcionalitása, logikája. A Game View-któl kapott események alapján alakítja a játékot (lépteti a játékost stb.) illetve ha valami vizuális frissítésre szorul, üzenettel jelzi

a Game View-nak. Az Application Layerhez üzenettel is fordulhat, ha például perzisztens adat betöltésére van szükség, mert véget ért a pálya és a következőre kell lépni.

2.2.2 Funkciók

A játék a híres Sokoban továbbgondolt változata. Egy raktárépületben ládákat tárolnak. A raktárépület padlója négyzet alakú mezőkre van osztva, ezeken állnak a mezőkkel megegyező alapterületű ládák. A padlón kívül kétféle építőelemet találunk még, falat és oszlopot.

A raktárban többen dolgoznak, egy játékos egy adott dolgozót irányít. Pontosan annyi dolgozó kezd a raktárban, mint ahány játékos játssza a játékot. Játékosból 2-4 lehet. Egy dolgozó mezőről mezőre haladhat az oldalak mentén, de kizárólag a padló típusú mezőkön, falon, oszlopon vagy másik dolgozó kiindulási mezőjén nem haladhat át. Egy mezőn egyszerre csak egy dolgozó tartózkodhat.

Egyes padló típusú mezőkön a játék kezdetén ládákat találunk, melyek szintén mezőről mezőre mozoghatnak az oldalak mentén, a padló típusú mezőkön. Mozgatásukhoz a dolgozóknak kell eltolniuk ezeket: ha egy dolgozó egy olyan padlóra akar lépni, melyen láda van és a láda képes ugyanazon irányban továbbmozogni (azaz nincs az abban az irányban lévő szomszédos mezőn fal, oszlop vagy egy játékos kiindulási mezője), amerre a dolgozó lépne, a ládát lépés közben eltolja arra a mezőre, különben a dolgozó nem léphet arra. A ládák egy ilyen eltolással mindig csak egy szomszédos mezőre kerülhetnek. A ládákat húzni nem lehet, így előfordulhat, hogy egy láda beragad és azt többé mozgatni nem lehet.

A cél a ládák előírt helyre tologatása. Az előírt helyek olyan padló típusú mezők, melyek meg vannak jelölve. Ha egy dolgozóra ládát tolunk, akkor a dolgozó automatikusan a szomszédos mezőre tolódik. Ha nem tud eltolódni (a tolás a dolgozót falba, oszlopba vagy másik játékos kiindulási mezőjére léptetné), a dolgozó életet veszít és visszakerül a kiindulási helyére. A ládák nem nyomhatók össze.

Ha egy vonalban közvetlen egymás után több láda vagy dolgozó - akár mindkettő vegyesen - áll és a legtávolabbi eltolható az adott irányba, akkor az egész lánc eltolható. A dolgozók közvetlenül nem tolhatják el egymást.

A padlón egyes helyeken lyukak találhatók, amelyekre ládát tolva a láda leesik (eltűnik), és nem kerül vissza a játékba. Ha dolgozó lép rá, vagy rátolják, életet veszít és visszakerül a kiindulási helyére. Némelyik lyuk csak akkor viselkedik lyukként, ha egy kapcsolón láda áll, egyébként padlónak tűnik. Ha egy kapcsolóra láda kerül, a kapcsoló kinyit egy vagy több lyukat. Amint a láda lekerül a kapcsolóról, a kapcsoló által kinyitott lyukak bezáródnak. Ha munkás áll a kapcsolóra, akkor nem kapcsol.

Minden dolgozó rendelkezik egy számára kijelölt kiindulási mezővel. A dolgozók itt kezdik a játékot, valamint innen folytatják életvesztés esetén is. Minden kiindulási mezőn csak az ahhoz tartozó dolgozó tartózkodhat és láda sem tolható rá.

Minden dolgozó három élettel kezd, ha valakinek ez elfogyott, az kikerül a játékból és az őt irányító játékos veszít. Egy játékos egy láda helyére tolásával egy pontot szerez. Ha egy játékos eltol a helyéről egy ládát, azzal a helyére juttató játékos - akár önmaga - pontszámát eggyel csökkenti. Ha egy játékos megmozgat úgy egy láncot, hogy a láncban valahol egy vagy több

dolgozót tol meg, ezáltal a helyére tol egy vagy több ládát, akkor az így szerzett pontokat az egész láncot mozgató játékos kapja, így lehet pontot lopni.

Egyes ládák szívecskével vannak megjelölve. Ezek a ládák ugyanúgy használhatók pontszerzésre és a játék céljának elérésére, mint a hagyományos ládák, de lyukba lökésük eggyel megnöveli a mozgató dolgozó életének számát. Az életszerzés elve azonos a pontszerzés elvével, vagyis, ha a láda lyukba lökése egy láncban és nem közvetlenül történik, akkor is a láncot mozgató dolgozó szerzi az életet.

Ha az összes játékban lévő láda (ami előfordulhat, hogy több, de az is, hogy kevesebb, mint amennyi előírt hely van) egy-egy előírt helyre került, vagy már nincsen mozgatható láda, a játék azonnal véget ér. Ilyenkor az a játékos nyer, aki a legtöbb pontot szerezte. Amennyiben több játékos (akár az összes) is azonos pontszámmal rendelkezik a játék végén, közöttük holtverseny alakul ki. A játék akkor is véget ér, ha már csak egy dolgozó marad, mert a többi meghalt. Ilyenkor automatikusan az utolsónak megmaradt dolgozó játékosa nyer, a ládák helyzetétől és a többi játékos pontszámától függetlenül.

A játék a kezdés pillanatától folyamatosan zajlik, vagyis nincsen körökre osztva, a dolgozók tetszőleges időpillanatban mozoghatnak. A játékban időkorlát nincsen, akár a végtelenségig is játszható.

2.2.3 Felhasználók

Használatához nincs szükség szakértelemre vagy előképzettségre, a játék lényege percek alatt megtanulható, és a gondolkodni - vagy egymást szétnyomni - vágyókat pedig rövid ideig elszórakoztathatja.

2.2.4 Korlátozások

A szoftvernek a JDK beépített könyvtárait kell használnia a funkciók megvalósításához és az elkészült programnak futnia kell a sztenderd JRE futtatókörnyezetben.

2.2.5 Feltételezések, kapcsolatok

Szoftver projekt laboratórium tárgyhonlap, ahol megtalálható a feladatkiírás, a dokumentáció sablonjai, továbbá a leadás ütemezése.
https://www.iit.bme.hu/targyak/BMEVIIIAB02

Szoftver projekt laboratórium ideiglenes oldal, ahol elérhetőek az imént említett segédanyagok a tárgyholnap elérhetetlensége esetén.
http://devil.iit.bme.hu/~balage/projlab/

2.3 Követelmények

2.3.1 Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
1.01	Vannak ládák	bemutatáskor	alapvető	megrendelő	View warehouse	
1.02	A pálya négyzet alakú mezőkre van osztva	bemutatáskor	alapvető	megrendelő	View warehouse	

1.03	Vannak falak és oszlopok	bemutatáskor	alapvető	megrendelő	View warehouse	
1.04	A játékosok dolgozókat irányítanak	bemutatáskor	alapvető	megrendelő	Move worker	
1.05	A játékot egyszerre több játékos játssza	bemutatáskor	alapvető	megrendelő	Move worker	
1.06	A dolgozók a padlókón mozoghatnak	bemutatáskor	alapvető	megrendelő	Move worker	
1.07	Egy mezőn egyszerre csak egy dolgozó állhat	bemutatáskor	fontos	csapat	Move worker	
1.08	A dolgozók nem tolhatják el közvetlenül egymást	bemutatáskor	fontos	megrendelő	Move worker	
1.09	A ládák falra, oszlopra és kiindulási mezőkre nem tolhatók	bemutatáskor	alapvető	megrendelő	Push crates	
1.10	A dolgozók eltolhatják a ládákat	bemutatáskor	alapvető	megrendelő	Move worker, Push crates	
1.11	A ládák beragadhatnak	bemutatáskor	fontos	csapat	Push crates	
1.12	Cél a ládák előírt helyre mozgatásával minél több pontot szerezni	bemutatáskor	alapvető	megrendelő	Push crates	
1.13	Egy eltolt láda másik ládát és dolgozót is tovább tolhat	bemutatáskor	fontos	megrendelő	Push workers, Push crates	
1.14	A dolgozók falra, oszlopra vagy másik dolgozó kiindulási mezőjére tolva életet veszítenek	bemutatáskor	fontos	megrendelő	Move worker, Push crates, Push workers	A megerendelő esetében meghalnak
1.15	A ládák nem nyomhatók össze	bemutatáskor	alapvető	megrendelő	Push crates	
1.16	A dolgozók ládákat és más dolgozókat láncban is eltolhatnak	bemutatáskor	fontos	csapat	Move worker, Push crates, Push workers	
1.17	Vannak lyukak	bemutatáskor	fontos	megrendelő	View warehouse	

1.18	A ládák lyukra tolva leesnek	bemutatáskor	fontos	megrendelő	Push crates	
1.19	A dolgozók lyukra lépve életet vesztenek	bemutatáskor	fontos	megrendelő	Move worker	A megrendelő esetében meghalnak
1.20	Életvesztés esetén a dolgozók visszakerülnek a kiindulási mezőjükre	bemutatáskor	fontos	csapat	Move worker	
1.21	A ládák kapcsolóra tolva kinyitnak egy vagy több lyukat	bemutatáskor	fontos	megrendelő	Push crates, Control switches	
1.22	A ládák kapcsolóról eltolva becsuknak egy vagy több, általuk kinyitott lyukat	bemutatáskor	fontos	megrendelő	Push crates, Control switches	
1.23	Minden dolgozó saját kiindulási mezőről indul	bemutatáskor	alapvető	csapat	Start game	
1.24	A dolgozók három élettel kezdenek	bemutatáskor	opcionális	csapat	Start game	
1.25	A játékos veszít, ha a dolgozójának nincs több élete	bemutatáskor	alapvető	csapat	Finish game	
1.26	A játékosok pontot szereznek egy láda helyre tolásával	bemutatáskor	alapvető	megrendelő	Move worker, Push crates	
1.27	A játékosok pontot veszíthetnek	bemutatáskor	opcionális	csapat	Move worker, Push crates	
1.28	A játékosok pontot lophatnak	bemutatáskor	fontos	csapat	Move worker, Push crates	
1.29	Egyes ládák szívecskével vannak megjelölve	bemutatáskor	opcionális	csapat	View warehouse	

1.30	A dolgozók életet szerezhetnek szívecskével jelölt láda lyukba tolásával	bemutatáskor	opcionális	csapat	Move worker, Push crates	
1.31	Ha minden láda előírt helyen van, a játéknak vége	bemutatáskor	alapvető	megrendelő	Push crates	
1.32	Ha nincs mozgatható láda, a játéknak vége	bemutatáskor	alapvető	megrendelő	Push crates	
1.33	Ha már csak egy dolgozó van életben, ő nyer és a játéknak vége	bemutatáskor	fontos	csapat	Move worker	
1.34	Ha a játék végére több dolgozó marad életben, a legtöbb pontot szerző játékos(ok) nyer(nek).	bemutatáskor	alapvető	megrendelő	Move worker	
1.35	Azonos pontszám esetén döntetlen van több játékos között.	bemutatáskor	fontos	csapat		
1.36	A játék folyamatos, nincsenek körök	bemutatáskor	opcionális	csapat		
1.37	A játéknak nincs időkorlátja	bemutatáskor	opcionális	csapat		
1.38	A játékot el lehet indítani	bemutatáskor	alapvető	megrendelő	Start game	
1.39	A játékból ki lehet lépni	bemutatáskor	alapvető	megrendelő	Exit game	

2.3.2 Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
2.01	Számítógép (PC)	bemutatáskor	alapvető	megrendelő	Célhardver
2.02	Eclipse	nincs	opcionális	csapat	JAVA IDE
2.03	JRE	bemutatáskor	alapvető	megrendelő	JAVA futtatókörnyezet
2.04	Slack	nincs	opcionális	csapat	Kommunikációs platform
2.05	WhiteStarUML	nincs	opcionális	csapat	UML modellező
2.06	Google Drive	nincs	alapvető	csapat	Webes tárhely és dokumentumkezelő
2.07	Microsoft Office	nincs	opcionális	csapat	Dokumentumkezelő
2.08	Git	nincs	alapvető	csapat	Verziókezelő
2.09	Github	nincs	alapvető	csapat	Webes Git platform

2.10	GIMP	nincs	opcionális	csapat	Raszteres képszerkesztő
2.11	Inkscape	nincs	opcionális	csapat	Vektoros képszerkesztő
2.12	Egér	nincs	alapvető	megrendelő	Periféria
2.13	Billentyűzet	nincs	alapvető	megrendelő	Periféria
2.14	Monitor	nincs	alapvető	megrendelő	Periféria

2.3.3 Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
3.01	Szkeleton átadása	bemutatáskor	alapvető	megrendelő	márc. 12.
3.02	Prototípus átadása	bemutatáskor	alapvető	megrendelő	ápr. 23.
3.03	Grafikus változat átadása	bemutatáskor	alapvető	megrendelő	máj. 14.
3.04	Teljes program átadása	bemutatáskor	alapvető	megrendelő	máj. 18.
3.05	Installálási útmutatást és kezelési leírás biztosítása	bemutatáskor	alapvető	megrendelő	

2.3.4 Egyéb nem funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
4.01	A forrásprogramnak a laboratóriumban rendszeresített JDK alatt lefordíthatónak és futtathatónak kell lennie	bemutatáskor	alapvető	megrendelő	
4.02	A programnak működnie kell a laboratórium számítógépein	bemutatáskor	alapvető	megrendelő	

2.4 Lényeges use-case-ek

2.4.1 Use-case leírások

Use-case neve	View warehouse
Rövid leírás	A játékosok megtekintik a pályát.
Aktorok	Player
Forgatókönyv	1. A játékosok megfigyelik a pályát. 2. A rendszer kirajzolja a pályát a ládákkal ill. a játékosokat azok kiindulási helyeivel együtt, továbbá kijelöli a ládák előírt helyeit.

Use-case neve	Move worker
Rövid leírás	A játékosok az általuk használt dolgozót irányítják.
Aktorok	Player
Forgatókönyv	1. Egy játékos a saját dolgozójával lép egy vele szomszédos padlóra, ha az nem más dolgozó kiindulási helye.
Alternatív forgatókönyv	1.A. A szomszédos mezőn láda tartózkodik, amely eltolható, ekkor a dolgozó eltolja azt a lépés irányába szomszédos mezőre.
Alternatív forgatókönyv	1.B A szomszédos mező lyuk, a dolgozó életet veszít és visszakerül a kiindulási helyére vagy ha nics több élete, akkor a játékosa veszít.

Use-case neve	Push crates
Rövid leírás	A dolgozók ládákat tologatnak.
Aktorok	Player
Forgatókönyv	1. Egy dolgozó eltol egy ládát, amely az eltolás irányába szomszédos mezőre kerül, ha az nem fal, oszlop vagy kiindulási mező.
Alternatív forgatókönyv	1.A A láda egy lánc vége, ekkor az egész lánc eltolódik, ha a másik végén játékos vagy eltolható láda van.
Alternatív forgatókönyv	1.B A szomszédos mező egy kijelölt hely, az eltolást végző játékos pontot kap.
Alternatív forgatókönyv	1.B.1 A láda egy láncban szerepel, a pontot a lánc végén álló játékos kapja.
Alternatív forgatókönyv	1.C A szomszédos mező egy lyuk, a láda leesik.
Alternatív forgatókönyv	1.C.1 A leesett láda szívecskével volt megjelölve, az eltolást végző dolgozó életet kap.
Alternatív forgatókönyv	1.D Az egyik kijelölt helyről eltolják a ládát, az érte pontot kapó játékos pontot veszít.

Use-case neve	Push workers
Rövid leírás	A dolgozók közvetetten más dolgozókat tologatnak.
Aktorok	Player

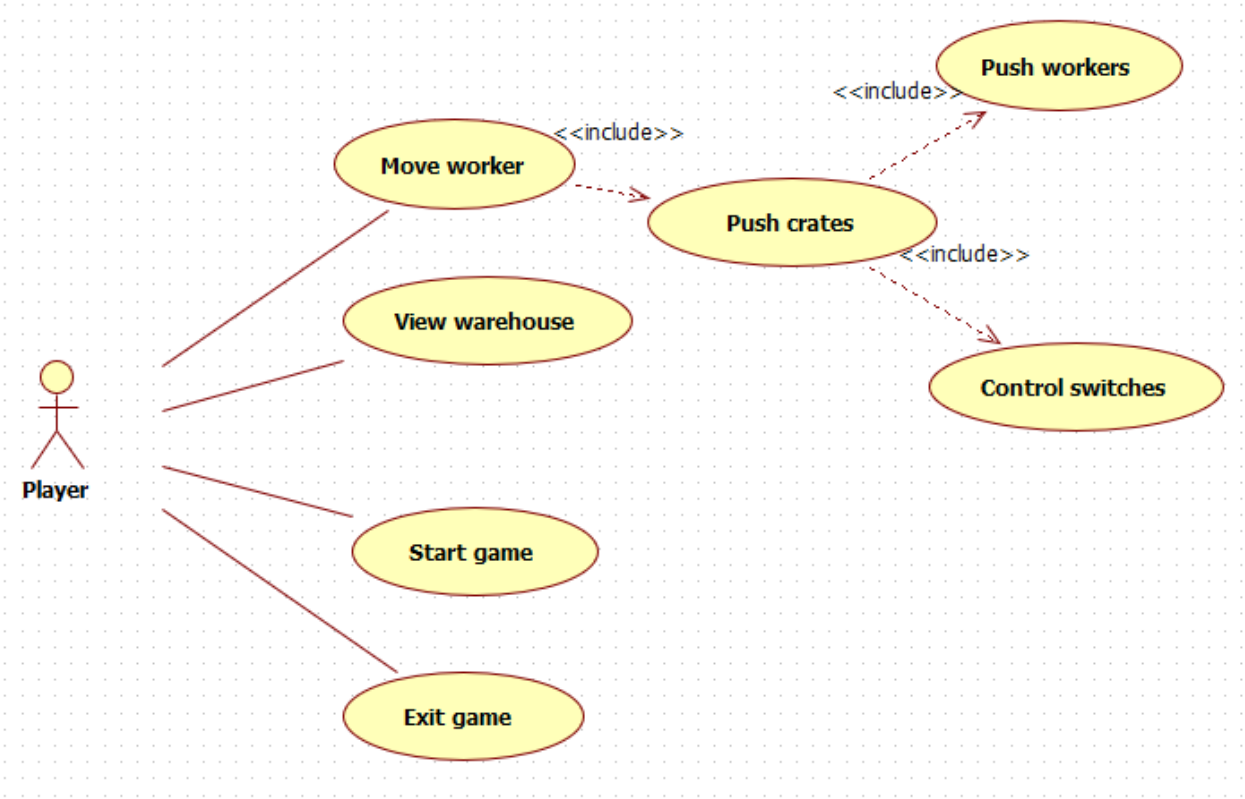
Forgatókönyv	1. Egy dolgozó eltol egy ládát, amely láncot alkot egy vagy több dolgozóval, ekkor a dolgozók is eltolódnak.
Alternatív forgatókönyv	1.A Ha a lánc végén egy dolgozó áll és az eltolás irányában fal, oszlop, lyuk vagy más kiindulási hely található, a lánc eltolódik és az eltolt dolgozó életet veszít.

Use-case neve	Control switches
Rövid leírás	A dolgozók a kapcsolót irányítják.
Aktorok	Player
Forgatókönyv	1. Egy dolgozó egy ládát tol egy kapcsolóra, ezzel aktiválva bizonyos lyukakat. 2. Egy dolgozó egy ládát eltol egy kapcsolóról, ezzel bizonyos lyukakat padlókká alakítva.

Use-case neve	Start game
Rövid leírás	A játékosok elindítják a játékot.
Aktorok	Player
Forgatókönyv	1. A játékosok megadják a létszámukat és elindítják a játékot.

Use-case neve	Exit game
Rövid leírás	A játékosok bezárják a programot.
Aktorok	Player
Forgatókönyv	1. Egy játékos bezárja a játékot a kilépés gombra kattintva.

2.4.2 Use-case diagram



2.5 Szótár

dolgozó: Egy entitás, amely a játéktérben található és egy játékos által irányítható. A dolgozó pontokkal és életekkel rendelkezik, továbbá ládákat ill. egy láncban más dolgozókat is el tud tolni. Egy dolgozó egy mező méretű. A dolgozók és a játékosok kölcsönösen megfeleltethetők egymásnak.
élet: Egy szám, ami az egyes dolgozókhoz van rendelve. Ha ez eléri a nullát, az őt irányító játékos veszít és nem folytathatja a játékot. A játék kezdetén minden dolgozónak három élete van.
előírt hely: ld. láda (előírt) helye.
eltol: Egy lépés, aminek keretében egy dolgozó, vagy egy láda a szomszédos mezőre mozdít egy vele szomszédos mezőn tartózkodó dolgozót vagy ládát az eltolás irányába.
eltolható láda: Egy láda eltolható egy lánc végén, ha a tolás irányában szomszédos mező padló.
fal: A játéktér határait jelző mezőtípus, amelyen ládák nem tolhatók keresztül, és amelynek egy dolgozót ládával nekitolva az életet veszít.
holtverseny: Egy helyzet, ami akkor következik be, ha a játék végén több játékos pontszáma megegyezik. Holtverseny esetén ily módon több nyertese is lehet a játéknak.
időkorlát: Egy számláló, amelynek értéke az idő előrehaladásával folyamatosan csökken, és ha eléri a nullát, a játék véget ér.
játék kezdete: Egy esemény, ami akkor következik be, ha játéktér és a benne elhelyezkedő objektumok teljesen betöltöttek.
játékos: Egy valós személy, aki egy dolgozót irányít.

játéktér: A tér, amelyben a játékosok játszanak. A játéktér négyzet alakú mezőkből áll, amelynek különböző típusai vannak. Egy mező legfeljebb négy másik mezővel lehet szomszédos az oldalai mentén.
játék vége: Egy esemény, ami akkor következik be, ha az összes láda a helyére kerül, ha már egyetlen ládát sem lehet eltolni vagy egy kivétellel az összes dolgozó életeinek száma eléri a nullát.
kapcsoló: Olyan padlótípus, amelyre ládát rátolva az lyukká változtat egy játék által kijelölt padlót vagy padlókat. Ha a láda lekerül erről a mezőről, a lyukak visszállnak padlóvá. Ha munkás áll a kapcsolóra, akkor nem kapcsol.
kiindulási mező: Minden dolgozóhoz pontosan egy ilyen mező tartozik, ami a játéktér szélén található és három oldalról fallal határolt. Egy kiindulási mezőn csak az ahhoz tartozó dolgozó tartózkodhat, más dolgozó vagy láda nem. Egy dolgozó szempontjából másik dolgozó kiindulási mezője falnak vagy oszlopnak tekinthető, azaz őt arra rátolva a dolgozó életet veszít. A dolgozók ide kerülnek a játék elején, illetve életvesztést követően.
láda: Egy entitás, amely a játéktérben található, egy mező méretű és dolgozók által eltolható.
láda beragad: Egy láda beragadtnak tekinthető, ha azt már egyetlen irányba se lehet eltolni, mivel legalább két szomszédos irányban kiindulási mező**,** fal, oszlop vagy egy beragadt láda található.
láda (előírt) helye: A játék elején kijelölt mező, amire egy ládát rátolva a tolást végző játékos pontot kap.
láda leesik: Egy esemény, ami akkor következik be, ha egy láda egy lyukra kerül. Ennek keretében a láda eltűnik a játéktérből.
lánc: Ládák és/vagy dolgozók sorozata, amelyek egy bizonyos irányba szomszédos mezőkön helyezkednek el egymáshoz képest, közvetlen egymás után. Egy láncban szereplő ládát vagy játékost eltolva, az eltolja a vele szomszédos ládát vagy játékost, ugyanabban az irányban.
lánc vége: A lánc első, illetve utolsó eleme, lehet dolgozó vagy láda.
lépés: Egy esemény, ami alatt egy dolgozó eltol egy vagy több ládát és/vagy dolgozót egy mezővel.
lyuk: Egy padlótípus, amelyre, ha dolgozó lép, az életet veszít, ha pedig ládát tolnak rá, az leesik.
mező: A játékteret felépítő egység, több típusa van, melyek különböző tulajdonságokkal jellemezhetők.
nyer: Egy játékos nyer, ha a játék végén neki van a legtöbb pontja, vagy egyedül az ő dolgozója maradt életben.
oszlop: Egy mezőtípus, amely a játéktér belsejében található. Az oszlopon ládák nem tolhatók keresztül és egy dolgozót egy oszlopnak nekitolva, az életet veszít.
padló: Egy általános mezőtípus, amelyen dolgozó vagy láda tartózkodhat és ami nincs semmilyen hatással a rajta álló játékosra vagy ládára. Ez a mező alap típusa.
pont: Egy szám, ami az egyes játékosokhoz van rendelve. Egy játékos pontjainak száma eggyel nő, ha a játékos a helyére tol egy ládát az általa irányított dolgozóval. Ha egy helyén lévő ládát eltolnak, a ládát helyére toló játékos pontjainak száma eggyel csökken.
pontlopás: Egy esemény, ami akkor következik be, ha egy játékos a helyére tol egy vagy több ládát úgy, hogy közötte és a ládák között más dolgozók is voltak. Ekkor a tolást kezdeményező játékos kapja a pontokat.
raktár: ld. játéktér.
raktárépület: ld. játéktér
Sokoban: Egy népszerű játék, amelyen a projektben tárgyalt játék alapul.
szívecske: Egy szimbólum, amellyel ládák meg lehetnek jelölve a játék által. Az ilyen ládák lyukba tolása eggyel megnöveli az eltoló dolgozó életeinek a számát.
szomszédos mező: Egy mezőnek az oldalain túl lévő mezők. Egy mezőnek 2-4 szomszédja lehet, attól függően, hogy az a játéktér sarkában, szélén vagy belsejében tartózkodik.

veszít: Egy játékos veszít, ha dolgozójának életeinek száma eléri a nullát, vagy a játék végén nem neki van a legtöbb pontja. Ekkor az ő dolgozója eltűnik a raktárból.

2.6 Projekt terv

2.6.1 A végrehajtás lépései

A feladatot három lépcsőben valósítjuk meg. A folyamatos munkát részhatáridők is segítik, melyeket az alábbi táblázat foglal össze.

Feladat	**Leadás módja***	**Határidő**
Követelmény, projekt, funkcionálitás	beadás	febr. 19.
Analízis modell kidolgozása 1.	beadás	febr. 26.
Analízis modell kidolgozása 2.	beadás	márc. 5.
Szkeleton tervezése	beadás	márc. 12.
Szkeleton	beadás + feltöltés	márc. 19.
Prototípus koncepciója	beadás	márc. 26.
Részletes tervek	beadás	ápr. 9.
Prototípus készítése, tesztelése	-	ápr. 16.
Prototípus	beadás + feltöltés	ápr. 23.
Grafikus felület specifikációja	beadás	máj. 2.
Grafikus változat készítése	-	máj. 7.
Grafikus változat	beadás + feltöltés	máj. 14.
Összefoglalás	beadás + feltöltés	máj. 18.

*A leadási módok magyarázata:

- beadás: A dokumentum leadása nyomtatott formában a konzulens részére.
- feltöltés: A forráskód, illetve az esetleges tesztbemenetek és elvárt kimenetek feltöltése az erre kialakított Hercules feladatbeadó rendszerbe.

A *szkeleton* változat célja annak bizonyítása, hogy az objektum és dinamikus modellek a definiált feladat egy modelljét alkotják. A szkeleton egy program, amelyben már valamennyi, a végső rendszerben is szereplő business objektum szerepel. Az objektumoknak csak az interfésze definiált. Valamennyi metódus az indulás pillanatában az ernyőre szöveges változatban kiírja a saját nevét, majd meghívja azon metódusokat, amelyeket a szolgáltatás végrehajtása érdekében meg kell hívnia. Amennyiben a metódusból valamely feltétel fennállása esetén hívunk meg más metódusokat, akkor a feltételre vonatkozó kérdést interaktívan az ernyőn fel kell tenni és a kapott válasz alapján kell a továbbiakban eljárni. A szkeletonnak alkalmasnak kell lenni arra, hogy a különböző forgatókönyvek és szekvencia diagramok ellenőrizhetők legyenek. Csak karakteres ernyőkezelés fogadható el, mert ez biztosítja a rendszer egyszerűségét.

A *prototípus* program célja annak demonstrálása, hogy a program elkészült, helyesen működik, valamennyi feladatát teljesíti. A prototípus változat egy elkészült program kivéve a kifejlett grafikus interfészt. Ez a program is parancssorból futtatható és karakteres ernyőkezelést alkalmaz. A változat tervezési szempontból elkészült, az ütemezés, az aktív objektumok kezelése megoldott. A business objektumok - a megjelenítésre vonatkozó részeket kivéve - valamennyi metódusa a végleges algoritmusokat tartalmazza. A megjelenítés és működtetés egy alfanumerikus ernyőn vezérelhető és követhető, ugyanakkor a vezérlés fájlból is történhet és a megjelenítés fájlba is logolható, ezzel megteremtve a rendszer tesztelésének lehetőségét. Különös figyelmet kell fordítani a parancssori interfész logikájára, felépítésére, valamint arra, hogy az mennyiben tükrözi és teszi láthatóvá a program működését, a beavatkozások hatásait.

A teljes (*grafikus*) változat a prototípustól elvileg csak a kezelői felület minőségében különbözhet. Ennek változatnak az értékelésekor a hangsúlyt sokkal inkább a megvalósítás belső szerkezetére, semmint a külalakra kell helyezni.

2.6.2 A csapat

A főbb feladatköröket igyekszünk úgy elosztani, hogy mindenki a számára legkézenfekvőbb témával tudjon foglalkozni. További fontos szempont a közel egységes terhelés, így szükség szerint egyes tagok az elsődleges feladatkörükön kívül eső témával is foglalkozhatnak.

Név	**Elsődleges feladatkör**
Csanády Máté Bende	Dokumentáció, UML
Jani Balázs Gábor	Programozás, Monitoring
Lakatos Dániel (CSK)	Menedzsment, Dokumentáció, Programozás
Lenkefi Péter	Programozás, Grafika
Szakállas Gergely	Dokumentáció, UML, Programozás

2.6.3 Kapcsolattartás

Kommunikáció: Az elsődleges kommunikációs csatornánkat a Slack nevű program jelenti. Itt lehetőség van különböző beszélgetéseket létrehozni az eltérő feladatokhoz, valamint csatlakoztatni a verzió- és dokumentumkezelő rendszerünket, ami lehetőséget ad egy átfogó kép kialakítására.
Meeting: Igyekszünk hetente legalább egyszer személyes meetinget tartani, ahol összefoglaljuk a történéseket, felvetjük és megvitatjuk az éppen aktuális problémákat. Ezen kívül részt veszünk a kijelölt konzultációkon, minél több csapattaggal.
Dokumentumkezelés: A dokumentumok kezelésére Google Docsot használunk. Ennek segítségével közösen szerkeszthetjük az állományt, követhetjük egymás változtatásait, valamint hozzászólhatunk, kiegészíthetjük egymás terveit, írásait.
Verziókezelés: Egy ilyen szintű projektnél már fontosnak tartjuk egy verziókezelő használatát, hogy mindenki átfogó képet kapjon a munka aktuális állapotáról, és a kódintegráció is zökkenőmentes legyen. A Git nevű szoftvert választottuk erre a feladatra, és hostként a Githubot használjuk.

2.6.4 Programok

Dokumentáció: A tervezési fázisban Google Docs dokumentumokat használunk, a dokumentációt később Microsoft Wordben véglegesítjük.
Verziókezelés: A Git nevű szoftvert választottuk erre a feladatra.
Verziókezelő host: A GitHub-on tároljuk a git repository-t.
Fejlesztőkörnyezet: A programozás az Eclipse szoftverben valósul meg.
UML: Az UML diagramokat a WhiteStarUML, valamint az Eclipse szoftverekkel készítjük.
Grafika: A grafikai elemek elkészítéséhez GIMP-et illetve Inkscape-et használunk.

2.7 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.02.16. 16:00	0,5 óra	LENKEFI	Cél (2.1.1) és szakterület (2.1.2) specifikálása, felhasználók meghatározása (2.2.3)
2018.02.16. 16:30	1 óra	LENKEFI	Funkciók meghatározása (2.2.2)
2018.02.16. 18:00	1 óra	JANI	Funkciók kiegészítése (2.2.2)
2018.02.16. 19:00	1,5 óra	SZAKÁLLAS	Szótár előzetes megírása a funkciók alapján (2.5)
2018.02.16. 20:00	1 óra	LAKATOS	Projekt terv elkészítése (2.6)
2018.02.17. 13:00	1 óra	LAKATOS	Funkciók átfogalmazása, pontosítása (2.2.2)
2018.02.17. 14:00	0,5 óra	CSANÁDY	Bevezetés kiegészítése (2.1)
2018.02.17. 15:00	1 óra	SZAKÁLLAS	Funkciók áttekintése, pontosítása (2.2.2), szótár frissítése (2.5)
2018.02.17. 15:00	1 óra	LAKATOS	Funkcionális, átadással kapcsolatos és egyéb nem funkcionális követelmények meghatározása (2.3.1, 2.3.3, 2.3.4)
2018.02.17. 18:30	2 óra	SZAKÁLLAS	Lényeges use-casek meghatározása (2.4)
2018.02.17. 18:30	2,5 óra	CSANÁDY	Szótár frissítése (2.5), a dokumentum általános felülvizsgálata
2018.02.17. 19:00	0,5 óra	LENKEFI	Funkciók és követelmények felülvizsgálata (2.2.2, 2.3)
2018.02.17. 19:00	2 óra	JANI	A dokumentum általános felülvizsgálata, esztétikai és grammatikai hibák javítása
2018.02.17. 21:00	1 óra	LENKEFI	Általános áttekintés hozzáadása (2.2.1)
2018.02.17 23:00	0,5 óra	LAKATOS	Definíciók és rövidítések magyarázatának hozzáadása (2.1.3)

2018.02.17 23:30	1 óra	LAKATOS	Dokumentum véglegesítése, esztétikai korrekciók

3. Analízis modell kidolgozása

3.1 Objektum katalógus

3.1.1 Dolgozó

A Dolgozó típus példányai azok, akiket a játékosok közvetlenül irányítanak a játéktérben. Minden Dolgozóhoz pontosan egy játékos tartozik. A Dolgozóknak több életük lehet (legfeljebb három), tudnak Ládákat tologatni és közvetetten el is lehet őket tolni más Dolgozók által.

3.1.2 Láda

A Dolgozók Láda objektumokat tologatnak túlnyomó részt a játékban. A Ládák másik Ládát vagy Dolgozót is letolhatnak, ők azonban csak tolás hatására tudnak mozogni, mást mozgatni. Összenyomhatatlanok, a játéktérből pedig csak úgy kerülhetnek ki, ha Lyukba kerülnek.

3.1.3 Szívecskés láda

A Szívecskés Láda a Láda egy olyan típusa, amit aktív Lyuk mezőre tolva, a tolást végző Dolgozó egy életet kap.

3.1.4 Mező

A Mező objektumok képezik a játéktér teljes részét, amiben egy mező jelent egy egységet. Négyzet alakúak és egyenként legfeljebb négy másik mezővel lehetnek szomszédosok, akiket tárolnak. A Mező objektum a játéktér alapegysége. Négyzet alakjukból kifolyólag legfeljebb négy Mezővel szomszédosok, melyeket ismernek, tárolnak.

3.1.5 Padló

A Padló a Mező objektum egy fajtája. A Padló objektumokon Dolgozók és Ládák egyaránt közlekedhetnek, egyéb ráhatások nélkül. Egy Padlón egyszerre csak egy Dolgozó vagy Láda állhat, akiket csak mozgatással vagy eltolással lehet onnan eltávolítani.

3.1.6 Lyuk

A Lyuk egyfajta Padló, amelyhez tartozhat egy viselkedését szabályozó kapcsoló. A kapcsolható lyukak lehetnek aktívak vagy inaktívak. Ha egy Lyuk aktív és egy Dolgozó rálép, a Dolgozó egy életet veszít, ha pedig Láda kerül a Lyukra, az eltűnik a játéktérből és nem kerül többé vissza. Inaktív esetben a Lyuk Padlóként funkcionál, a nem kapcsolható Lyukak pedig állandóan aktívak.

3.1.7 Kapcsoló

A Kapcsoló az egy fajta Padló. Egy vagy több Lyuk tartozik hozzá, amelyeket aktiválni vagy deaktiválni tud. A Kapcsoló állását egy Láda rátolásával lehet megváltoztatni. Ha a Láda lekerül a Kapcsolóról, a hozzá tartozó Lyukak ismét inaktívvá válnak.

3.1.8 Kiindulási mező

A Kiindulási mező objektumok speciális típusai a Padlónak. Egy Kiindulási Mezőhöz pontosan egy Dolgozó tartozik, aki erre mezőre kerül a játék kezdetén vagy életvesztés esetén. Kiindulási mezőn más Dolgozó vagy Láda objektum nem állhat, az ő számukra ez a mező Falként viselkedik.

3.1.9 Előírt hely

Az Előírt helyek olyan Padlók, amelyeket a Raktár jelöl ki. Ilyen típusú Padlóra tolva egy Ládát a tolást végző Dolgozó játékosa pontot kap. Ha egy Ládát eltolnak egy Előírt helyről, az érte pontot kapó játékos elveszíti a kapott pontot.

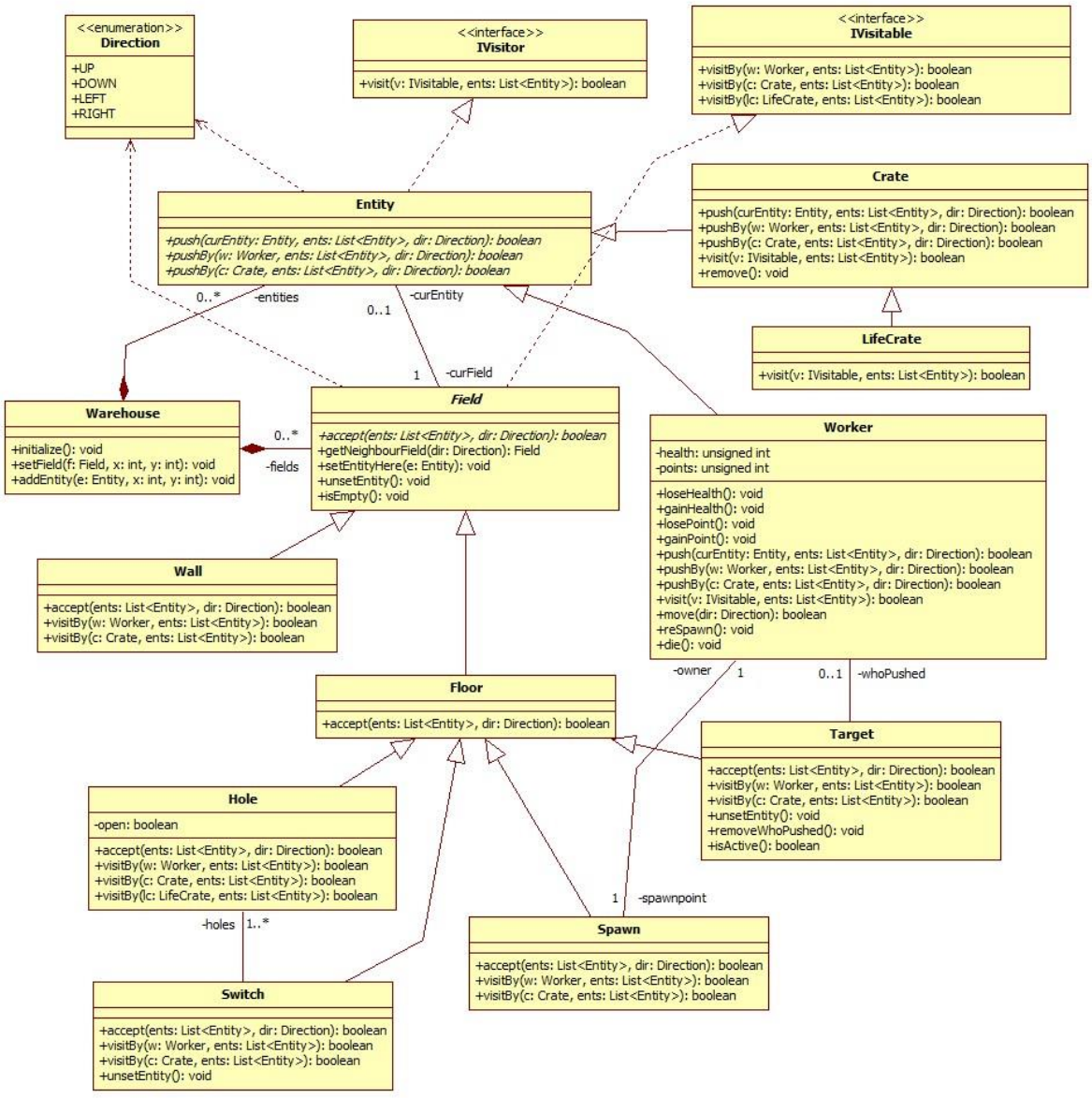
3.1.10 Fal

A Fal objektumok olyan Mezőket valósítanak meg, amelyeken nem tartózkodhat Dolgozó, illetve Láda, azaz Dolgozók nem léphetnek rájuk és Ládákat sem lehet rájuk tolni. A játéktér széleit ilyen objektumok határolják, a játéktéren belül pedig akadályt jelentenek. Ezzel szemben Dolgozót rá lehet tolni Falra, aminek hatására az adott Dolgozó egy életet veszít.

3.1.11 Raktár

A Raktár tárolja a játékteret alkotó összes Mezőt, betölti a játék kezdetén azokat, kijelöli a Kiindulási mezőket, az Előírt helyeket és elhelyezi a Dolgozókat, illetve a Ládákat.

3.2 Statikus struktúra diagramok



Megjegyzés: a nevesített asszociációvégekhez és attribútumokhoz implicit getter és setter függvények tartoznak, amelyeket a diagram az olvashatóság kedvéért nem jelöl.

3.3 Osztályok leírása

3.3.1 Crate

- **Felelősség**

Egy ládát szimbolizál. Nyilvántartja a láda pozícióját.

- **Interfészek**

IVisitor

- **Ősosztályok**

Entity

- **Attribútumok**
  - **-**
- **Metódusok**
  - **boolean push(Entity curEntity, List<Entity> ents, Direction dir):** Megpróbál eltolni egy megadott irányban lévő mezőn lévő entitást, szintén a megadott irányban. Átveszi egy listában az eddig tolni próbáló entitásokat. Igazzal tér vissza, amennyiben sikerült eltolnia az entitást, egyébként hamissal.
  - **boolean pushBy(Worker w, List<Entity> ents, Direction dir):** A ládát megpróbálja eltolni egy dolgozó a megadott irányban. Átveszi egy listában az eddig tolni próbáló entitásokat, és berakja a lista végére saját magát. Igazzal tér vissza, amennyiben a tolás sikeres volt, egyébként hamissal.
  - **boolean pushBy(Crate c, List<Entity> ents, Direction dir):** A ládát megpróbálja eltolni egy láda a megadott irányban. Átveszi egy listában az eddig tolni próbáló entitásokat, és berakja a lista végére saját magát. Igazzal tér vissza, amennyiben a tolás sikeres volt, egyébként hamissal.
  - **boolean visit(IVisitable v, List<Entity> ents):** A láda meglátogat egy látogatható objektumot. Átveszi egy listában az eddig tolni próbáló entitásokat. A látogatás eredményével tér vissza.
  - **void remove():** A láda kikerül a játékból.

3.3.2 Entity

- **Felelősség**

Absztrakt osztály, ami a mozgatható dolgokat képviseli.

- **Interfészek**

IVisitor

- **Ősosztályok**

-

- **Attribútumok**
  - **Field curField**: A mező, amelyen jelenleg az entitás tartózkodik.

- **Metódusok**
  - **abstract boolean push(Entity curEntity, List<Entity> ents, Direction dir):** Megpróbál eltolni egy megadott irányban lévő mezőn tartózkodó entitást, szintén a megadott irányban. Átveszi egy listában az eddig tolni próbáló entitásokat. Igazzal tér vissza, amennyiben sikerült eltolnia az entitást, egyébként hamissal.
  - **abstract boolean pushBy(Worker w, List<Entity> ents, Direction dir):** Az entitást megpróbálja eltolni egy dolgozó a megadott irányban. Átveszi egy listában az eddig tolni próbáló entitásokat, és berakja a lista végére saját magát. Igazzal tér vissza, amennyiben a tolás sikeres volt, egyébként hamissal.
  - **abstract boolean pushBy(Crate c, List<Entity> ents, Direction dir):** Az entitást megpróbálja eltolni egy láda a megadott irányban. Átveszi egy listában az eddig tolni próbáló entitásokat, és berakja a lista végére saját magát. Igazzal tér vissza, amennyiben a tolás sikeres volt, egyébként hamissal.
  - **void setCurField(Field f):** Beállítja az entitás mezőjét.
  - **void getCurField():** Lekérdezi az entitás mezőjét.

3.3.3 Field

- **Felelősség**

Absztrakt osztály, ami a játéktér (raktár) egy mezőjét szimbolizálja.

- **Interfészek**

IVisitable

- **Ősosztályok**

-

- **Attribútumok**
  - **Entity curEntity**: Az az entitás, ami jelenleg ezen a mezőn tartózkodik.

- **Metódusok**
  - **abstract boolean accept(List<Entity> ents, Direction dir)**: Megpróbál ráléptetni egy entitást saját magára. Átveszi egy listában az eddig tolni próbáló entitásokat. A lista utolsó eleme az az entitás, akit megpróbál ráléptetni saját magára. Igazzal tér vissza, amennyiben sikerült a léptetés, egyébként hamissal.
  - **Field getNeighbourField(Direction dir):** Az mező megkeresi az adott irányban lévő szomszédját.
  - **void setEntityHere(Entity e):** A mezőre helyez egy entitást, levéve onnan, ahol eddig volt.
  - **void setCurEntity(Entity e):** Beállítja a mező entitását.
  - **void unsetEntity():** A mezőn lévő entitás lekerül a mezőről, ezek után a mező üres lesz.
  - **boolean isEmpty():** Megadja, hogy tartózkodik-e entitás a mezőn. Igazzal tér vissza, ha tartózkodik entitás a mezőn, egyébként hamissal.

3.3.4 Floor

- **Felelősség**

A játéktér egy padlóját képviseli. Egy padlón tartózkodhat entitás, így rá is lehet lépni.

- **Interfészek**

IVisitable

- **Ősosztályok**

Field

- **Attribútumok**
  - **Entity curEntity**: Az az entitás, ami jelenleg ezen a mezőn tartózkodik.
- **Metódusok**
  - **boolean accept(List<Entity> ents, Direction dir)**: Megpróbál ráléptetni egy entitást saját magára. Átveszi egy listában az eddig tolni próbáló entitásokat. A lista utolsó eleme az az entitás, akit megpróbál ráléptetni saját magára. Ha nincs a mezőn entitás, a léptetés automatikusan sikerül. Ha van a mezőn egy entitás, azt megpróbálja rekurzívan továbbléptetni a megadott irányban. Igazzal tér vissza, amennyiben sikerült a léptetés, egyébként hamissal.

3.3.5 Hole

- **Felelősség**

A játéktér egy lyukas padlóját képviseli, ami lehet nyitott, illetve zárt állapotban. Zárt állapotban ugyanúgy viselkedik, mint egy egyszerű padló. Nyitott állapotban mind ha dolgozó, mind ha láda lép rá, az leesik.

- **Interfészek**

IVisitable

- **Ősosztályok**

Field ⇨ Floor

- **Attribútumok**
  - **boolean open:** Igaz, amennyiben a lyuk nyitott állapotban van, egyébként hamis.
- **Metódusok**
  - **boolean accept(List<Entity> ents, Direction dir)**: Megpróbál ráléptetni egy entitást saját magára. Átveszi egy listában az eddig tolni próbáló entitásokat. A lista utolsó eleme az az entitás, akit megpróbál ráléptetni saját magára. Ha nyitott állapotban van, a rálépni próbáló entitás leesik, és hamissal tér vissza. Ha zárt állapotban van és nincs a mezőn entitás, a léptetés automatikusan sikerül. Ha van a mezőn egy entitás, azt

megpróbálja rekurzívan továbbléptetni a megadott irányban. Igazzal tér vissza, amennyiben sikerült a léptetés, egyébként hamissal.
- **boolean visitBy(Worker w, List<Entity> ents)**: Leejt egy dolgozót a mélybe. A dolgozó ekkor életet veszít. Átveszi egy listában az eddig tolni próbáló entitásokat. Mindig igazzal tér vissza.
- **boolean visitBy(Crate c, List<Entity> ents)**: Leejt egy ládát a mélybe. A láda ekkor kikerül a játékból. Átveszi egy listában az eddig tolni próbáló entitásokat. Mindig igazzal tér vissza. Mindig igazzal tér vissza.
- **boolean visitBy(LifeCrate ls, List<Entity> ents)**: Leejt egy szívecskés ládát a mélybe. A láda ekkor kikerül a játékból és az őt leejtő dolgozó életeinek száma eggyel nő. Átveszi egy listában az eddig tolni próbáló entitásokat. Mindig igazzal tér vissza.
- **void setOpen(boolean o):** Beállítja a lyuk állapotát.

3.3.6 IVisitable

- **Felelősség**

Visitor viselkedés látogatható részét megvalósító interfész. Az őt implementáló osztályok objektumai meglátogathatók az IVisitor interfészt implementáló osztályok objektumai által.

- **Ősosztályok**

-

- **Metódusok**
  - **boolean visitBy(Worker w, List<Entity> ents):** Egy dolgozó meglátogat egy objektumot. Átveszi egy listában az eddig tolni próbáló entitásokat. A visszatérési értéke a látogatott objektumtól függ.
  - **boolean visitBy(Crate c, List<Entity> ents):** Egy láda meglátogat egy objektumot. Átveszi egy listában az eddig tolni próbáló entitásokat. A visszatérési értéke a látogatott objektumtól függ.
  - **boolean visitBy(LifeCrate lc, List<Entity> ents):** Egy szívecskés láda meglátogat egy objektumot. Átveszi egy listában az eddig tolni próbáló entitásokat. A visszatérési értéke a látogatott objektumtól függ.

3.3.7 IVisitor

- **Felelősség**

Visitor viselkedés látogató részét megvalósító interfész. Az őt implementáló osztályok objektumai képesek az IVisitable interfészt implementáló osztályok objektumait meglátogatni.

- **Ősosztályok**

-

- **Metódusok**
  - **boolean visit(IVisitable v, List<Entity> ents):** Egy objektum meglátogat egy látogatható objektumot. Átveszi egy listában az eddig tolni próbáló entitásokat. A látogatás eredményével tér vissza.

3.3.8 LifeCrate

- **Felelősség**

Egy szívecske szimbólummal rendelkező ládát szimbolizál. Ez a láda egy hagyományos ládaként viselkedik, de ha leesik egy lyukba, akkor az őt leejtő dolgozó kap egy életet.

- **Interfészek**

IVisitor

- **Ősosztályok**

Entity

- **Attribútumok**
  - **-**

- **Metódusok**
  - **boolean visit(IVisitable v, List<Entity> ents):** A szívecskés láda meglátogat egy látogatható objektumot. Átveszi egy listában az eddig tolni próbáló entitásokat. A látogatás eredményével tér vissza.

3.3.9 Spawn

- **Felelősség**

A játéktér egy olyan padlóját képviseli, ami egy dolgozó kiindulási mezője. Minden dolgozó saját kiindulási mezővel rendelkezik, innen kezdik a játékot, és innen is folytatják életvesztés után. Erre a mezőre kizárólag a hozzá tartozó dolgozó léphet, más entitás nem.

- **Interfészek**

IVisitable

- **Ősosztályok**

Field ⇨ Floor

- **Attribútumok**
  - **Worker owner:** Az a dolgozó, melynek ez a kiindulási mezője.

- **Metódusok**
  - **boolean accept(List<Entity> ents, Direction dir)**: Megpróbál ráléptetni egy entitást saját magára. Átveszi egy listában az eddig tolni próbáló entitásokat. A lista utolsó eleme az az entitás, akit megpróbál ráléptetni saját magára. Ha ez az a dolgozó, akinek ez a kiindulási mezője, akkor igazzal tér vissza, egyébként hamissal.
  - **boolean visitBy(Worker w, List<Entity> ents):** Egy munkás ütközik a kiindulási mezővel. Átveszi egy listában az eddig tolni próbáló entitásokat. A munkást falnak toló entitás a lista utolsó eleme. Ha a tolás közvetetten történik, akkor életet veszít. Ebben az esetben igazzal tér vissza, egyébként hamissal.
  - **boolean visitBy(Crate c, List<Entity> ents):** Egy dolgozó ütközik a kiindulási mezővel. Ekkor nem történik semmi, hamissal tér vissza. Átveszi egy listában az eddig tolni próbáló entitásokat.

3.3.10 Switch

- **Felelősség**

A játéktér egy olyan padlóját képviseli, melyen egy kapcsoló található. Amennyiben erre a kapcsolóra egy láda lép, egy vagy több lyuk nyitott állapotba kerül. Ha a láda lekerül a mezőről, a kapcsoló által kinyitott lyukak bezáródnak. Ha dolgozó lép a kapcsolóra, nem kapcsol.

- **Interfészek**

IVisitable

- **Ősosztályok**

Field ⇨ Floor

- **Attribútumok**
  - **List<Hole> holes**: Azon lyukak, melyek ehhez a kapcsolóhoz vannak rendelve.

- **Metódusok**
  - **boolean accept(List<Entity> ents, Direction dir)**: Megpróbál ráléptetni egy entitást saját magára. Átveszi egy listában az eddig tolni próbáló entitásokat. A lista utolsó eleme az az entitás, akit megpróbál ráléptetni saját magára. Ha nincs a mezőn entitás, a léptetés automatikusan sikerül. Ha van a mezőn egy entitás, azt megpróbálja továbbléptetni a megadott irányban. Ha sikerül a léptetés, a mezőre kerülő entitás kapcsolja a kapcsolót. Igazzal tér vissza, amennyiben sikerült a léptetés, egyébként hamissal.
  - **boolean visitBy(Worker w, List<Entity> ents)**: Kapcsol dolgozó által. Ilyenkor nem történik semmi, mert egy dolgozó hatására nem kapcsol. Átveszi egy listában az eddig tolni próbáló entitásokat. Mindig igazzal tér vissza.
  - **boolean visitBy(Crate c, List<Entity> ents)**: Kapcsol egy láda által. Ilyenkor a kapcsolóhoz rendelt lyukak nyitott állapotba kerülnek. Átveszi egy listában az eddig tolni próbáló entitásokat. Mindig igazzal tér vissza.
  - **void unsetEntity():** A mezőn lévő entitás lekerül a mezőről, ezek után a mező üres lesz. Ilyenkor a kapcsolóhoz rendelt lyukak zárt állapotba kerülnek.

3.3.11 Target

- **Felelősség**

A játéktér egy olyan padlóját képviseli, ami egy előírt hely a ládák számára. Ha egy dolgozó erre a mezőre tol egy ládát, akkor pontot kap érte, azonban ha később valaki eltolja innen a ládát, az érte pontot kapó dolgozó elveszti a kapott pontot.

- **Interfészek**

IVisitable

- **Ősosztályok**

Field ⇨ Floor

- **Attribútumok**
  - **Worker whoPushed:** Az a dolgozó, amelyik pontot kapott egy láda ide tolásával.

- **Metódusok**
  - **boolean accept(List<Entity> ents, Direction dir)**: Megpróbál ráléptetni egy entitást saját magára. Átveszi egy listában az eddig tolni próbáló entitásokat. A lista utolsó eleme az az entitás, akit megpróbál ráléptetni saját magára. Ha nincs a mezőn entitás, a léptetés automatikusan sikerül. Ha van a mezőn egy entitás, azt megpróbálja továbbléptetni a megadott irányban. Ha sikerül a léptetés, a mezőre kerülő entitás előírt helyre lépett. Igazzal tér vissza, amennyiben sikerült a léptetés, egyébként hamissal.
  - **void visitBy(Worker w, List<Entity> ents):** Egy munkás elérte az előírt helyet, ekkor nem történik semmi. Átveszi egy listában az eddig tolni próbáló entitásokat. Mindig igazzal tér vissza.
  - **void visitBy(Crate c, List<Entity> ents):** Egy láda elérte az előírt helyet. Ekkor a ládát ide juttató dolgozó pontot kap. Átveszi egy listában az eddig tolni próbáló entitásokat. A ládát ide juttató dolgozó az átvett lista első eleme. Mindig igazzal tér vissza.
  - **void unsetEntity():** A mezőn lévő entitás lekerül a mezőről, ezek után a mező üres lesz. Ha egy láda került el a mezőről (azaz van olyan dolgozó, aki ide juttatott egy ládát), az ide juttató dolgozó elveszti az érte kapott pontot.
  - **void removeWhoPushed():** Beállítja, hogy jelenleg nincsen olyan dolgozó, aki ide ládát tolt volna.
  - **boolean isActive():** Megadja, hogy van-e láda az előírt helyen. Igazzal tér vissza, amennyiben igen, egyébként hamissal.

3.3.12 Wall

- **Felelősség**

A játéktér egy olyan mezőjét képviseli, amire nem lehet lépni. Ha munkás próbálna meg közvetetten rálépni (tehát úgy, hogy egy láda tolja őt), akkor a munkás életet veszít.

- **Interfészek**

IVisitable

- **Ősosztályok**

Field

- **Attribútumok**
  - **-**

- **Metódusok**
  - **boolean accept(List<Entity> ents, Direction dir)**: Ütköztet egy entitást saját magával. Átveszi egy listában az eddig tolni próbáló entitásokat. A lista utolsó eleme az az entitás, akit megpróbál ütköztetni. Ha az entitás egy munkás közvetetten tolva, akkor igazzal tér vissza, egyébként hamissal.
  - **boolean visitBy(Worker w, List<Entity> ents):** Egy munkás ütközik a fallal. Átveszi egy listában az eddig tolni próbáló entitásokat. A munkást falnak toló entitás a lista utolsó eleme. Ha a tolás közvetetten történik, akkor életet veszít. Ebben az esetben igazzal tér vissza, egyébként hamissal.
  - **boolean visitBy(Crate c, List<Entity> ents):** Egy láda ütközik a fallal. Ekkor nem történik semmi, hamissal tér vissza. Átveszi egy listában az eddig tolni próbáló entitásokat.

3.3.13 Warehouse

- **Felelősség**

A játékteret szimbolizálja, tartalmazza az ott lévő mezőket.

- **Interfészek**

-

- **Ősosztályok**

-

- **Attribútumok**
  - **List<Field> fields:** A raktárépületben található mezők.
  - **List<Entity> entities:** A raktárépületben található entitások.

- **Metódusok**
  - **void initialize():** Inicializálja a játékteret, azzal felépíti a mezőket, és létrehozza az entitásokat.
  - **void setField(Field f, int x, int y):** Beállítja a pálya adott helyére a paraméterként kapott mezőt.
  - **void addEntity(Entity e, int x, int y):** Hozzáad egy entitást az adott pozíción levő mezőre.

3.3.14 Worker

- **Felelősség**

Egy dolgozót szimbolizál. Nyilvántartja a dolgozó pozícióját, életét, valamint pontszámát.

- **Interfészek**

IVisitor

- **Ősosztályok**

Entity

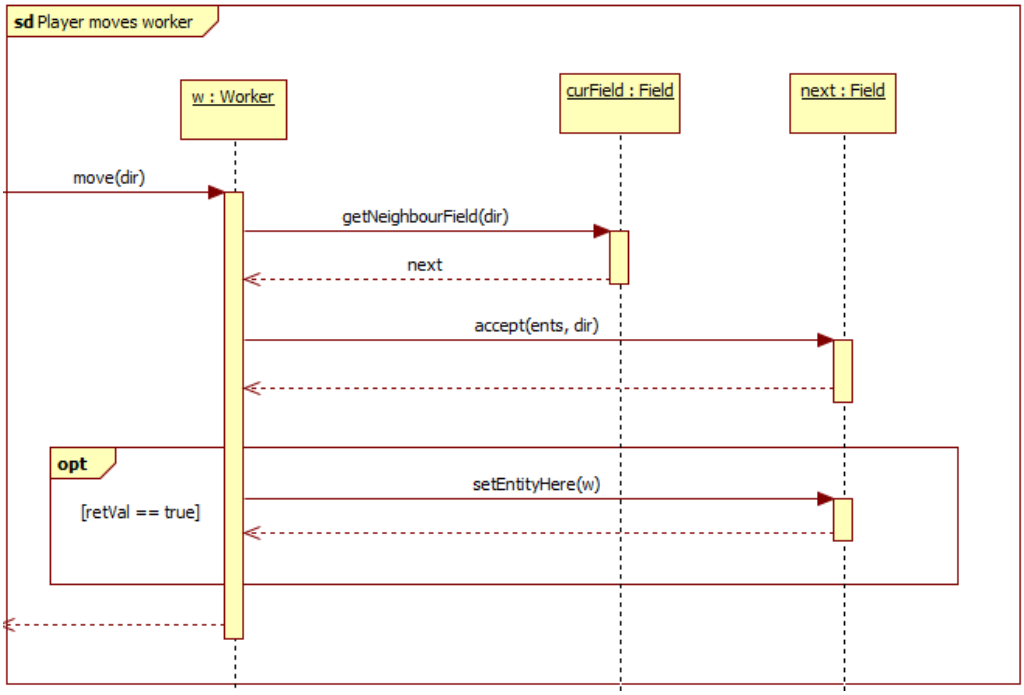
- **Attribútumok**
  - **unsigned int health**: A dolgozó élete. Ha eléri a nullát, a dolgozó meghal és kikerül a játékból.
  - **unsigned int points**: A dolgozó pontszáma. Minden egyes előírt helyre tolt ládával egyel nő, ám amennyiben azok a ládák később elmozdulnak az előírt helyekről, a pontok elvesznek.
  - **Spawn spawnpoint:** A munkás kiinduló mezője.

- **Metódusok**
  - **boolean push(Entity curEntity, List<Entity> ents, Direction dir)**: Megpróbál eltolni egy megadott irányban lévő mezőn tartózkodó entitást, szintén a megadott irányban. Átveszi egy listában az eddig tolni próbáló entitásokat. Igazzal tér vissza, amennyiben sikerült eltolnia az entitást, egyébként hamissal.
  - **boolean pushBy(Worker w, List<Entity> ents, Direction dir)**: Az dolgozót megpróbálja eltolni egy dolgozó a megadott irányban. Átveszi egy listában az eddig tolni próbáló entitásokat, és berakja a lista végére saját magát. Igazzal tér vissza, amennyiben a tolás sikeres volt, egyébként hamissal.
  - **boolean pushBy(Crate c, List<Entity> ents, Direction dir)**: Az dolgozót megpróbálja eltolni egy láda a megadott irányban. Átveszi egy listában az eddig tolni próbáló entitásokat, és berakja a lista végére saját magát. Igazzal tér vissza, amennyiben a tolás sikeres volt, egyébként hamissal.
  - **boolean visit(IVisitable v, List<Entity> ents):** A dolgozó meglátogat egy látogatható objektumot. Átveszi egy listában az eddig tolni próbáló entitásokat. A látogatás eredményével tér vissza.
  - **void move(Direction dir):** A játékos lépteti a dolgozót a megadott irányban lévő szomszédos mezőre.
  - **void loseHealth():** A dolgozó egy életet veszít.
  - **void gainHealth():** A dolgozó egy életet kap.
  - **void losePoint():** A dolgozó egy pontot veszít.
  - **void gainPoint():** A dolgozó egy pontot kap.
  - **void reSpawn():** A dolgozó átkerül jelenlegi mezőjéről a számára kijelölt kiinduló mezőre.
  - **void die():** A dolgozó meghal és kikerül a játékból.

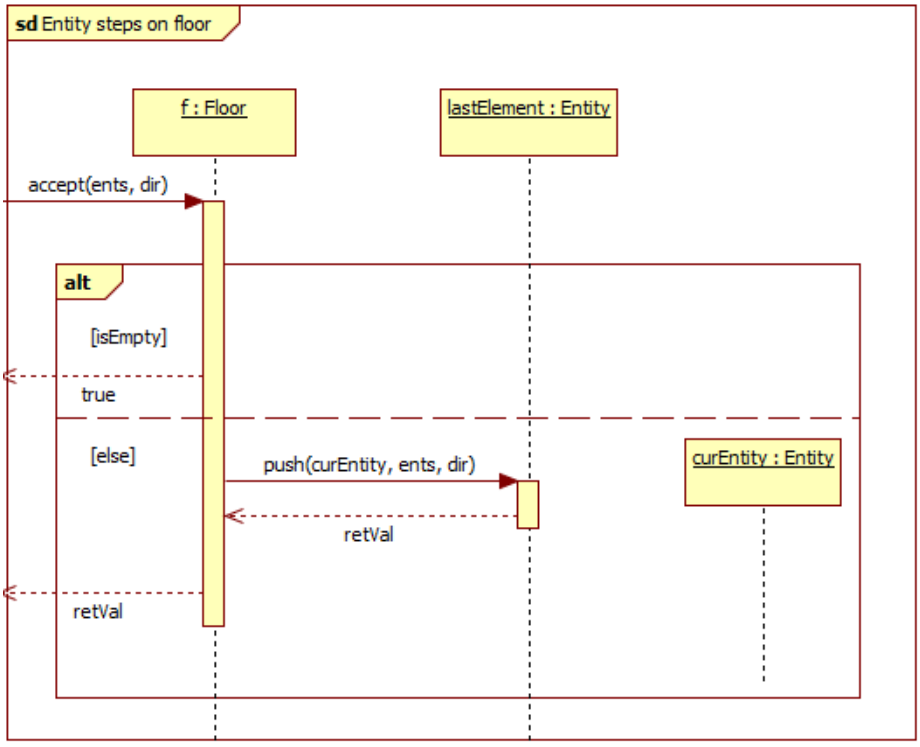
3.4 Szekvencia diagramok

Megjegyzés: A „dobozokban” lévő aláhúzásokat nem lehetséges eltüntetni a WhiteStarUML-ben, ezért úgy tekintjük, mintha azok ott sem lennének, hiszen aláhúzva Object-eket reprezentálnának. Továbbá mivel az UML eszköz nem támogatja a szabványos create jelölést, így az metódushívásként van modellezve.

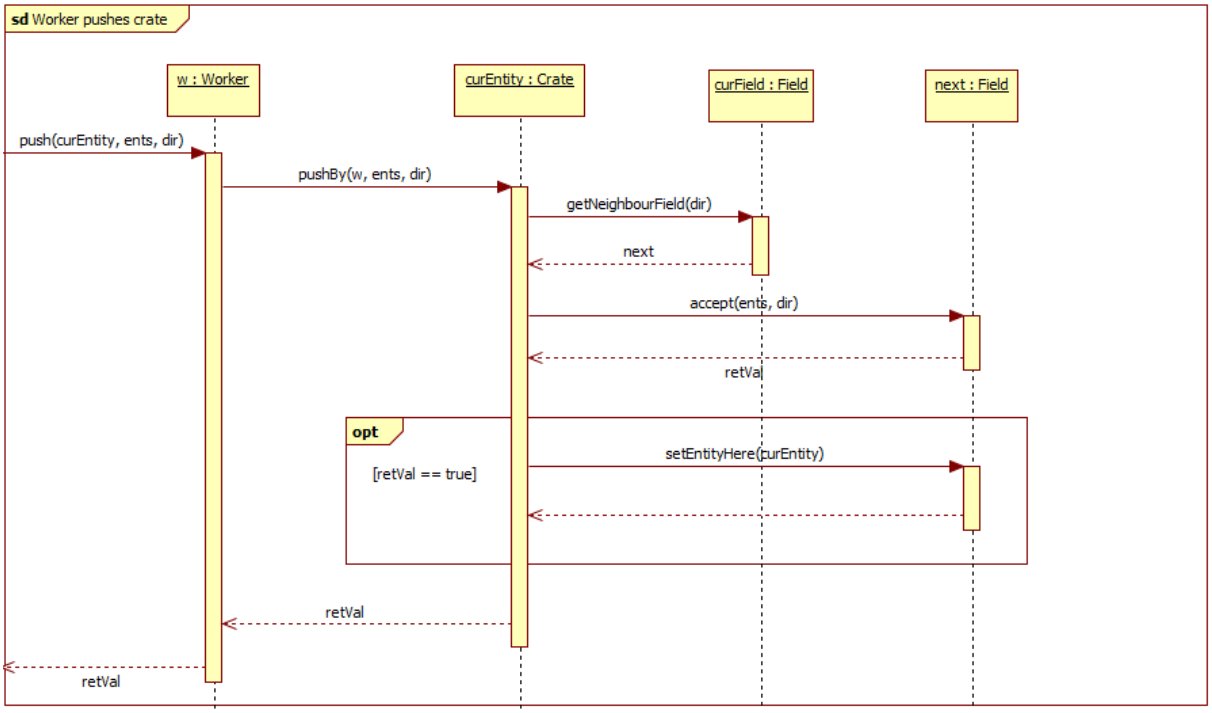
3.4.1 A játékos lépteti a dolgozót



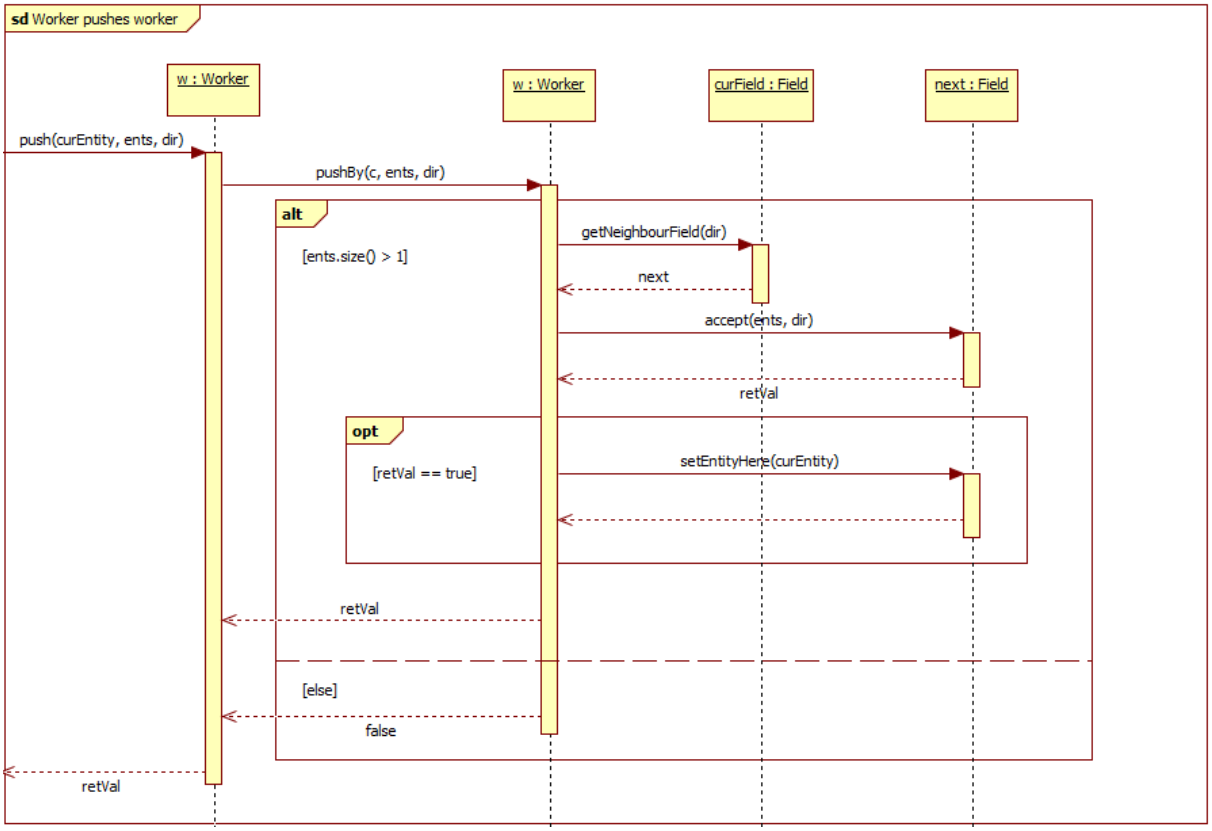
3.4.2 Egy entitás egy padlóra lép



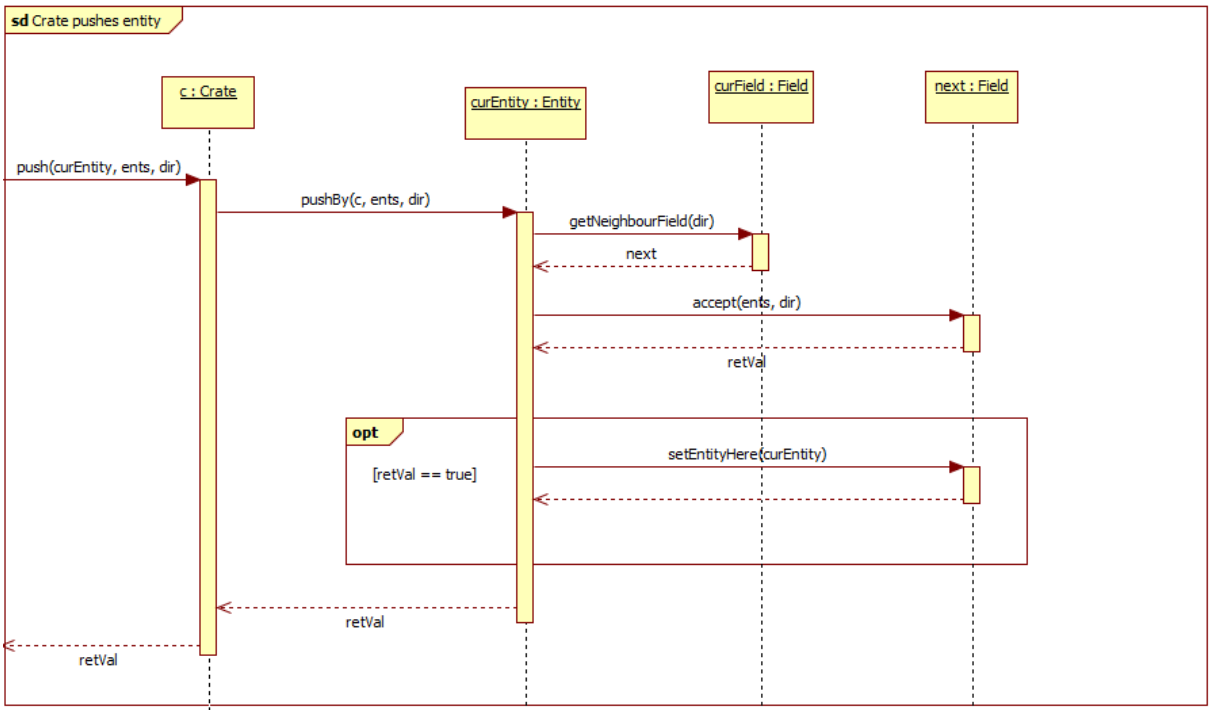
3.4.3 Egy dolgozó ládát tol



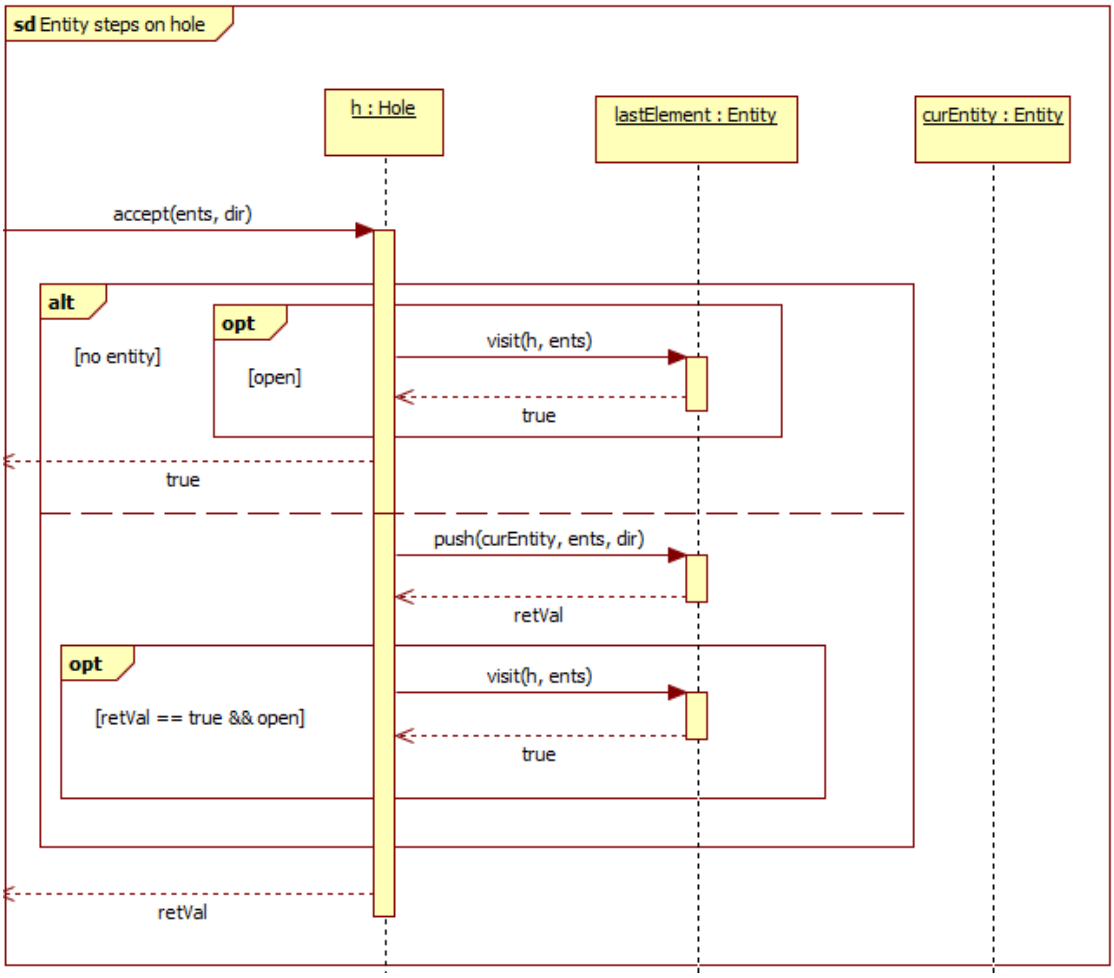
3.4.4 Egy dolgozó dolgozót tol



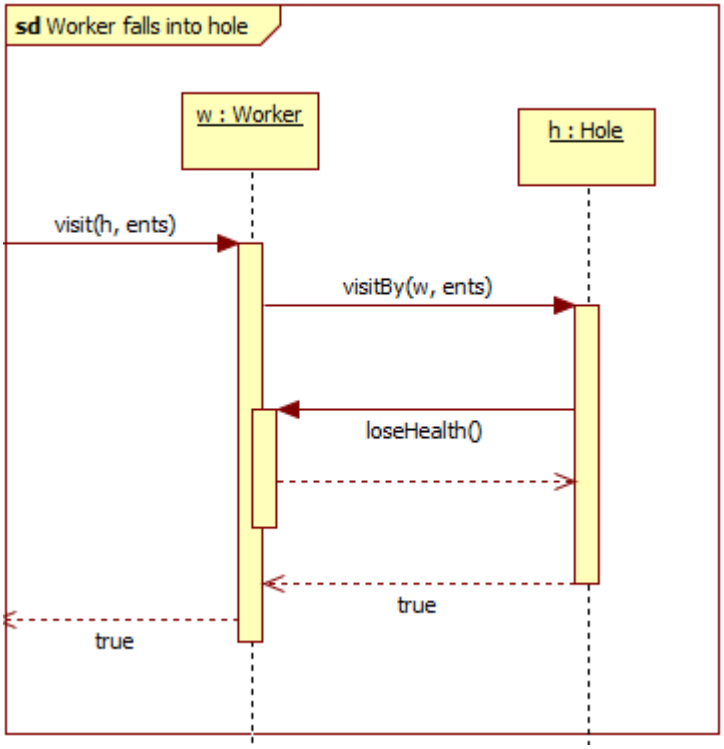
3.4.5 Egy láda entitást tol



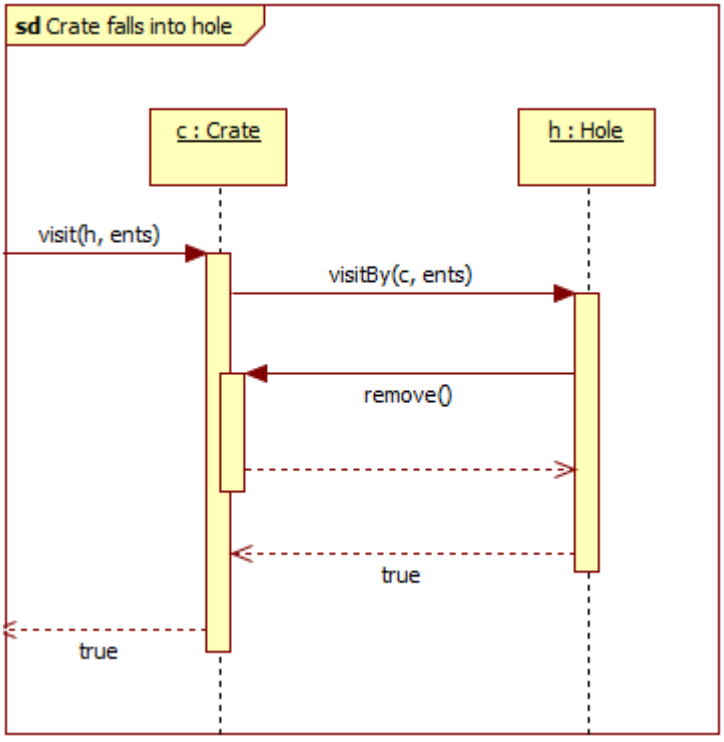
3.4.6 Entitás lyukra lép



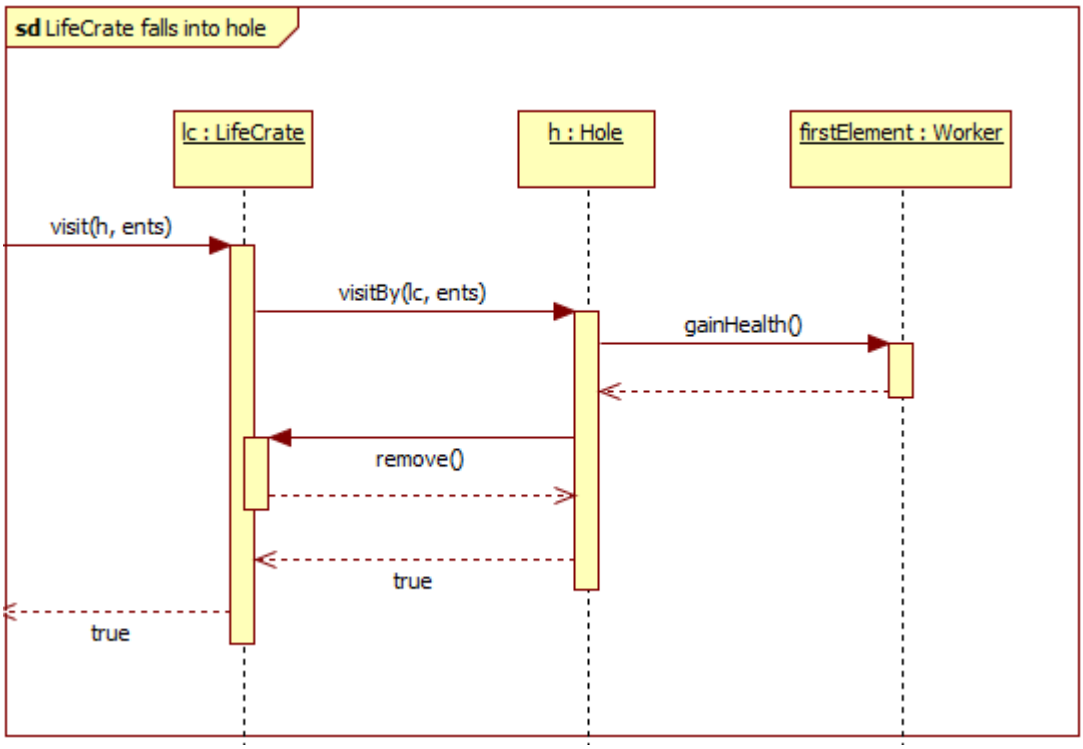
3.4.7 Dolgozó lyukba esik



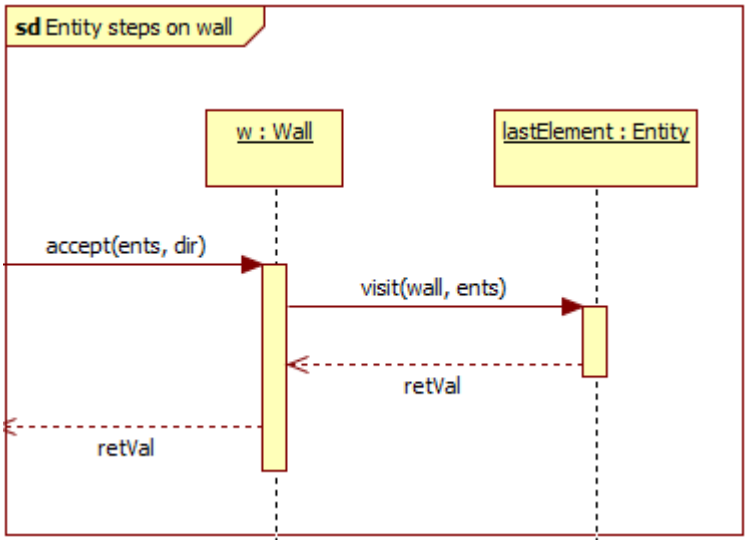
3.4.8 Láda lyukba esik



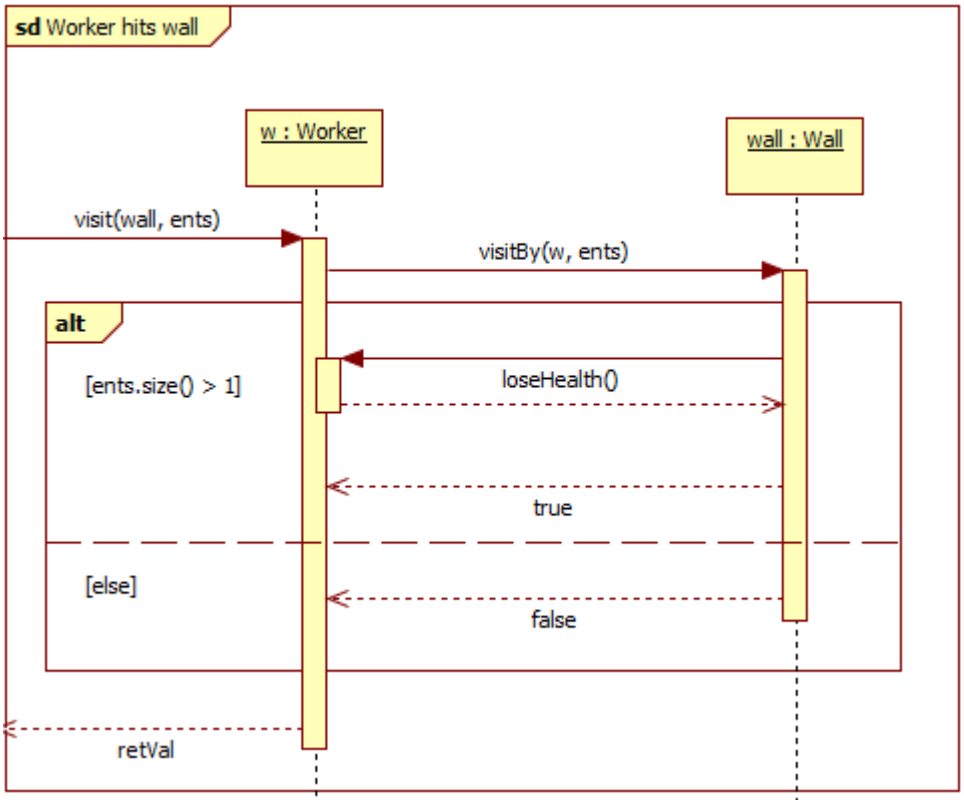
3.4.9 Szívecskés láda lyukba esik



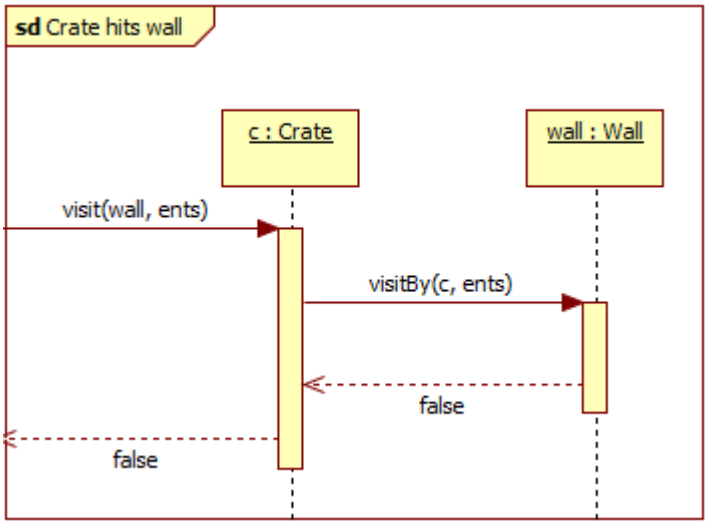
3.4.10 Entitás falnak megy



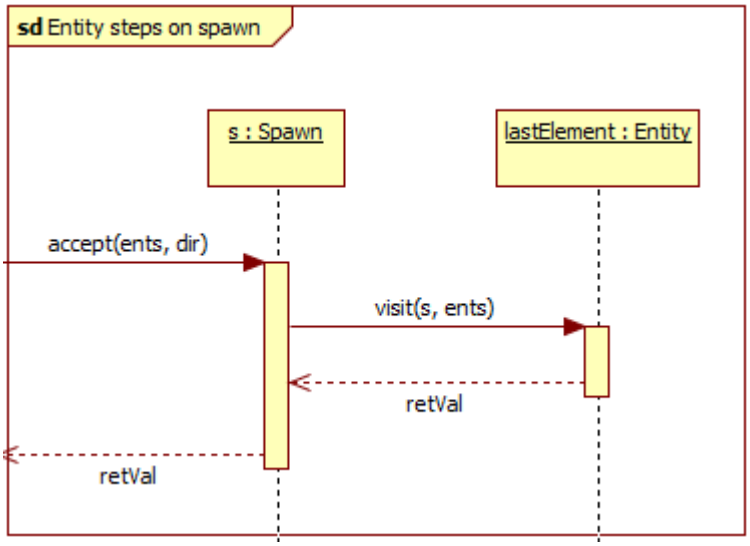
3.4.11 Dolgozó falnak megy



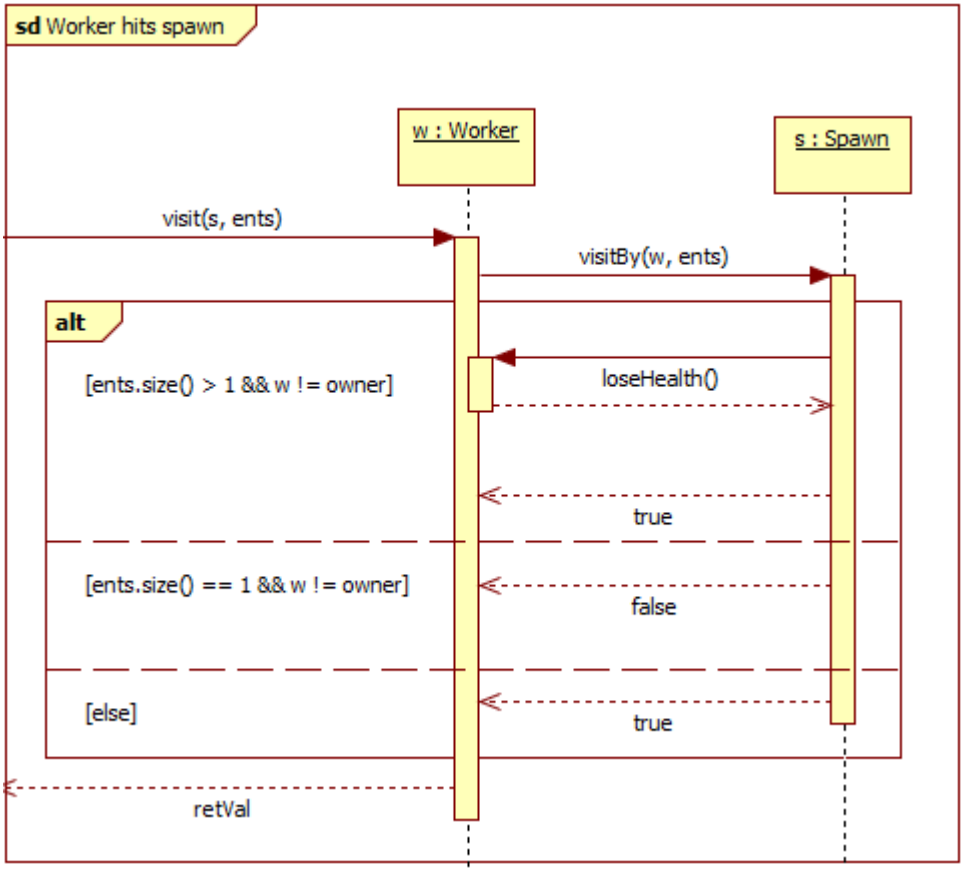
3.4.12 Láda falnak megy



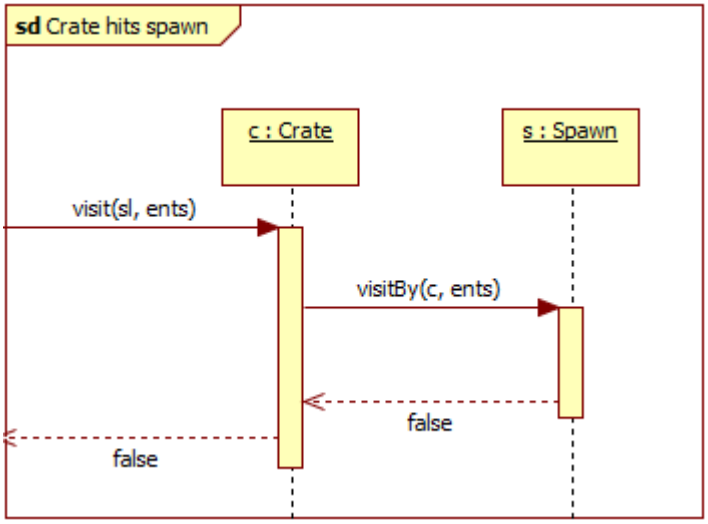
3.4.13 Entitás kiindulási mezőnek ütközik



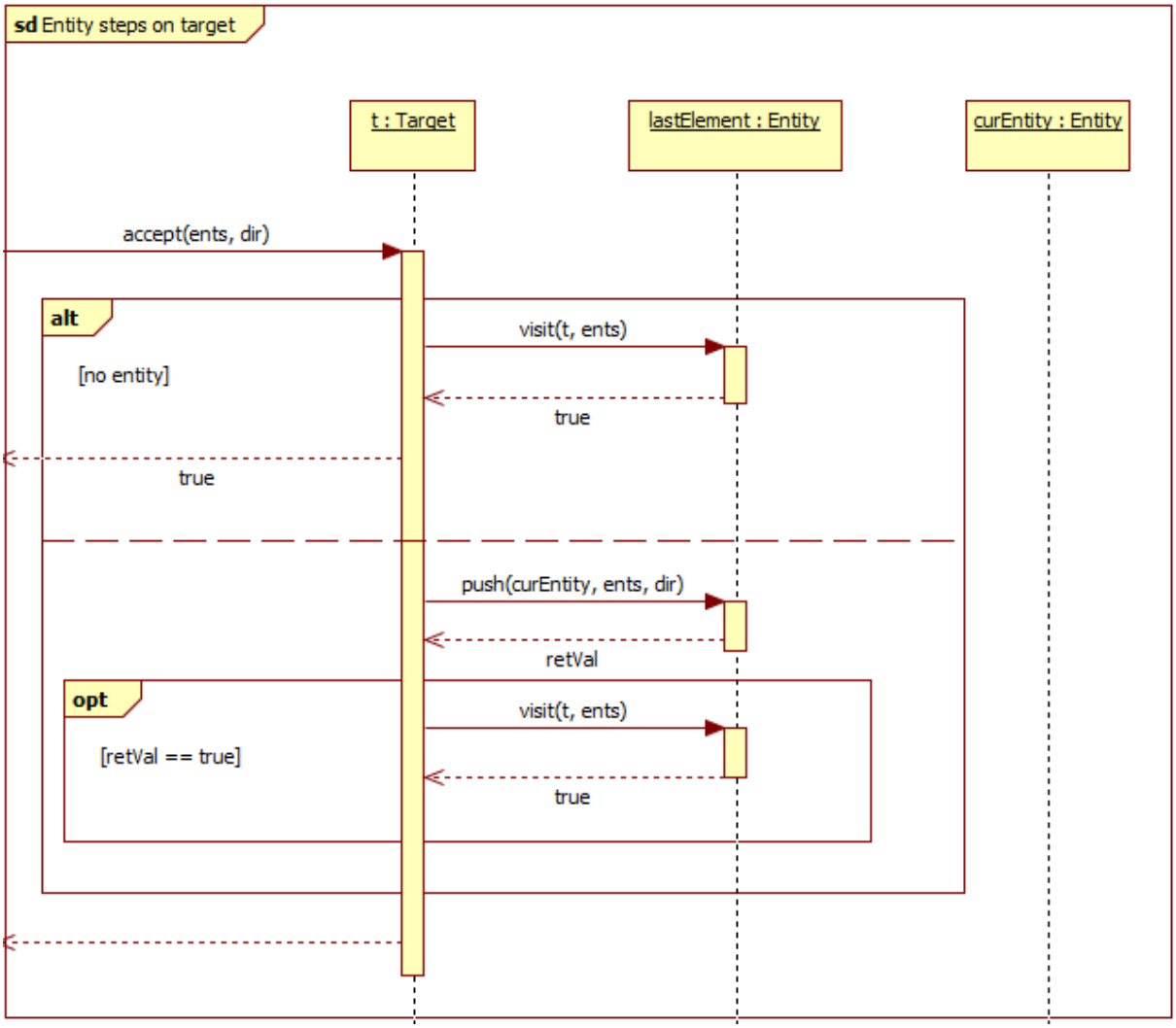
3.4.14 Dolgozó kiindulási mezőnek ütközik



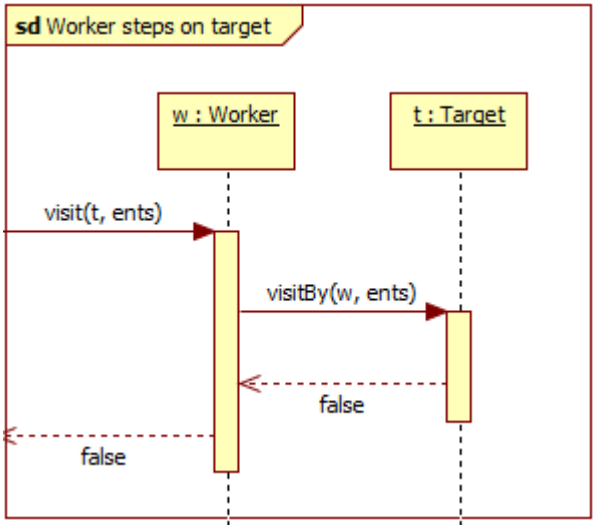
3.4.15 Láda kiindulási mezőnek ütközik



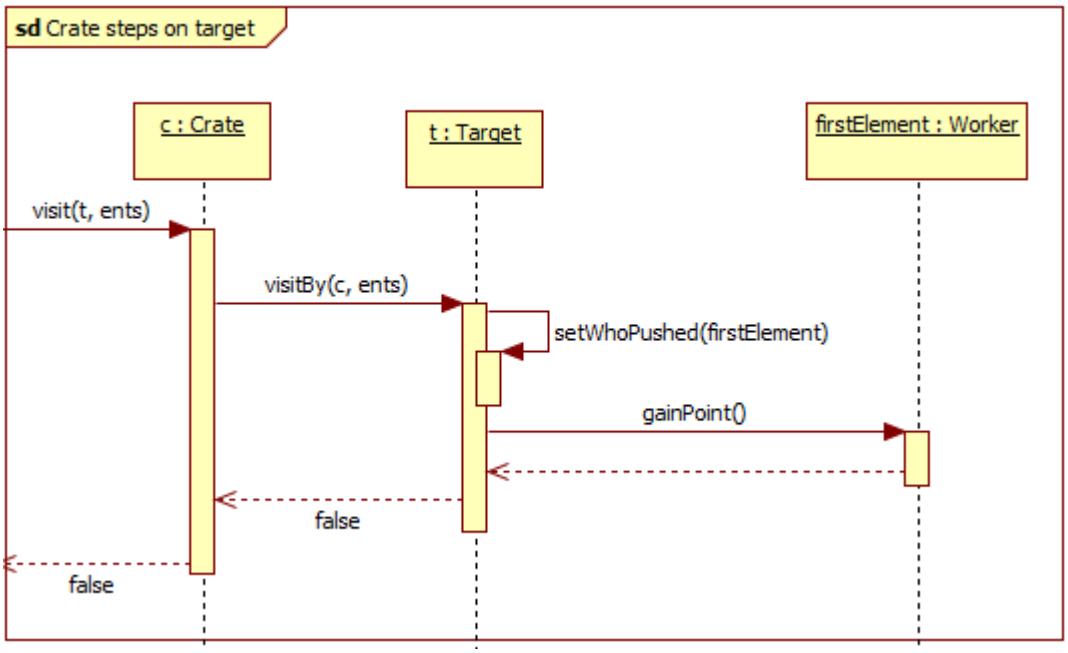
3.4.16 Entitás előírt helyre lép



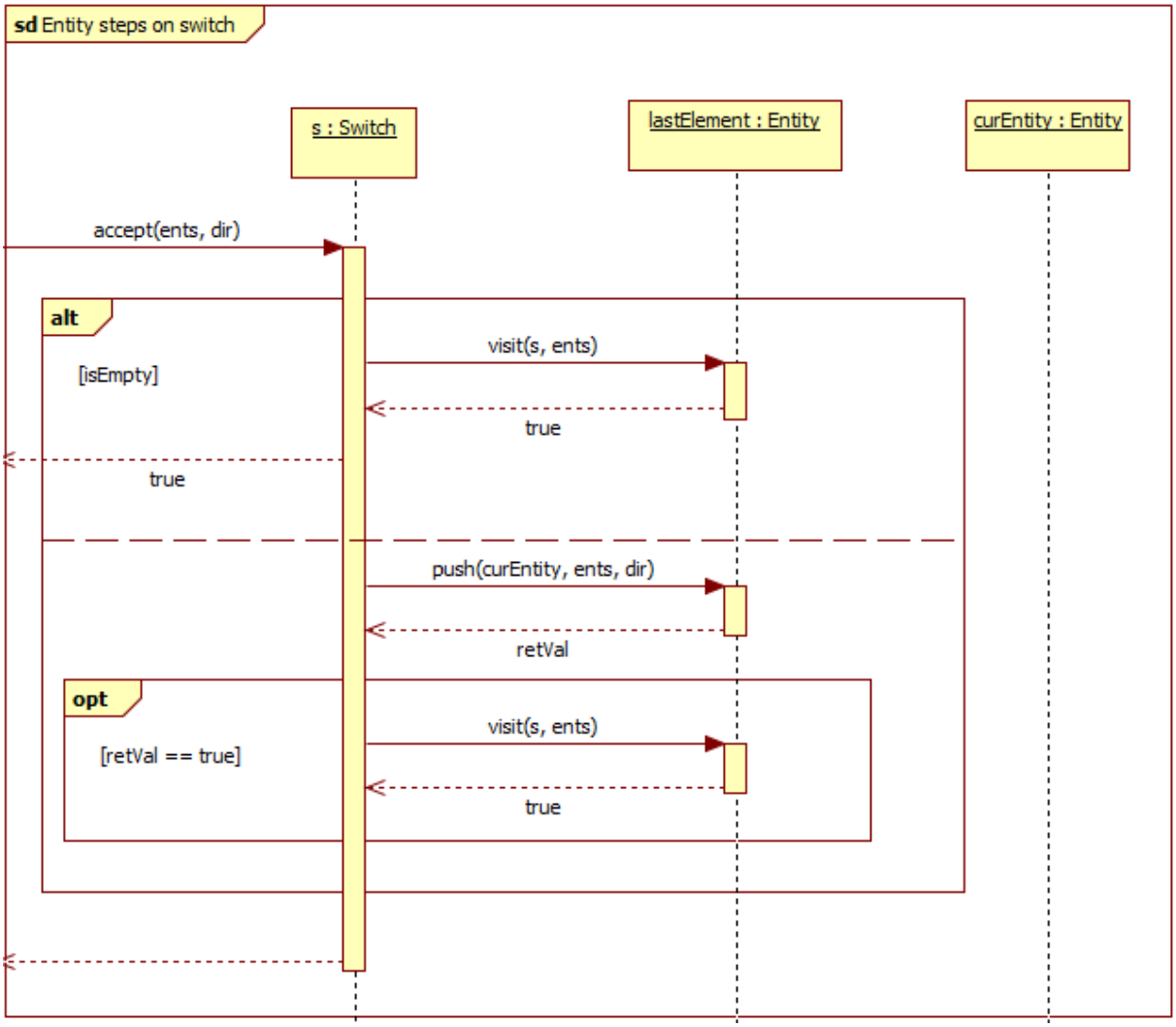
3.4.17 Dolgozó előírt helyre lép



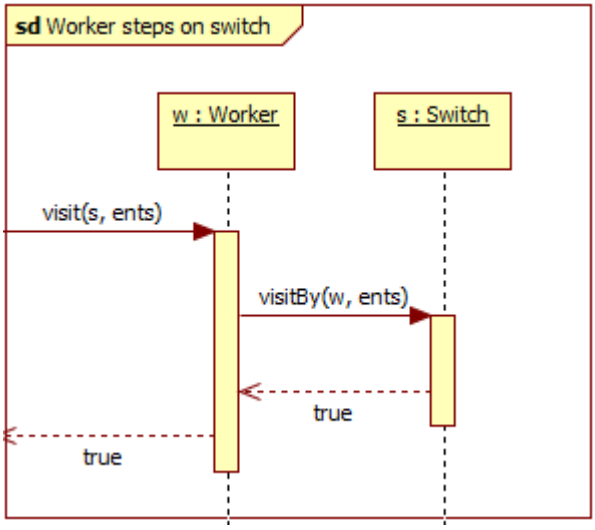
3.4.18 Láda előírt helyre lép



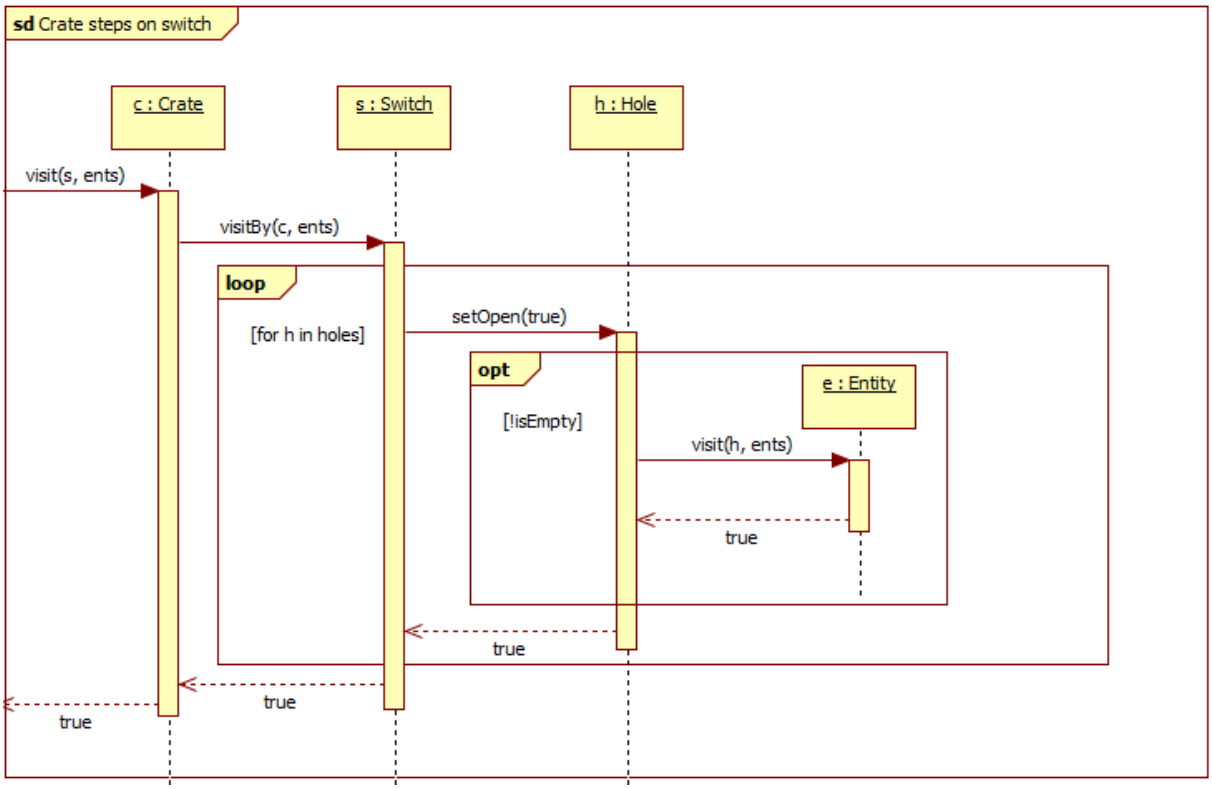
3.4.19 Entitás kapcsolóra lép



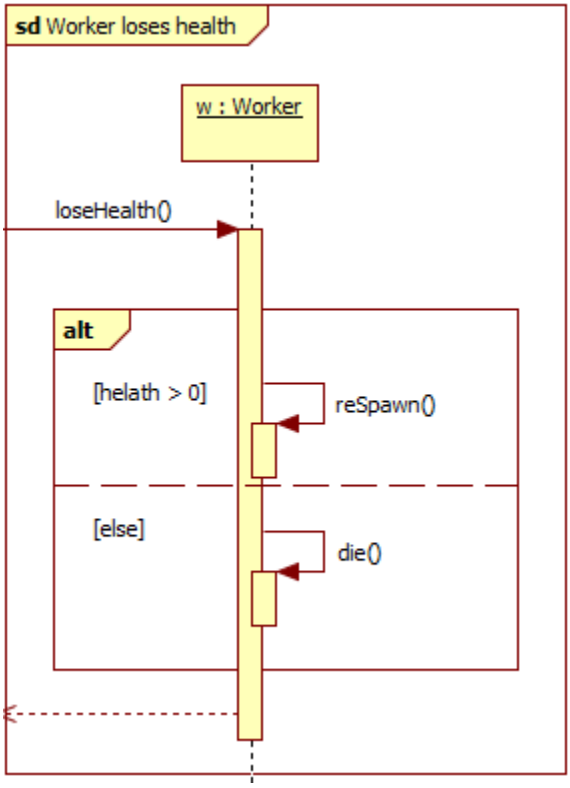
3.4.20 Dolgozó kapcsolóra lép



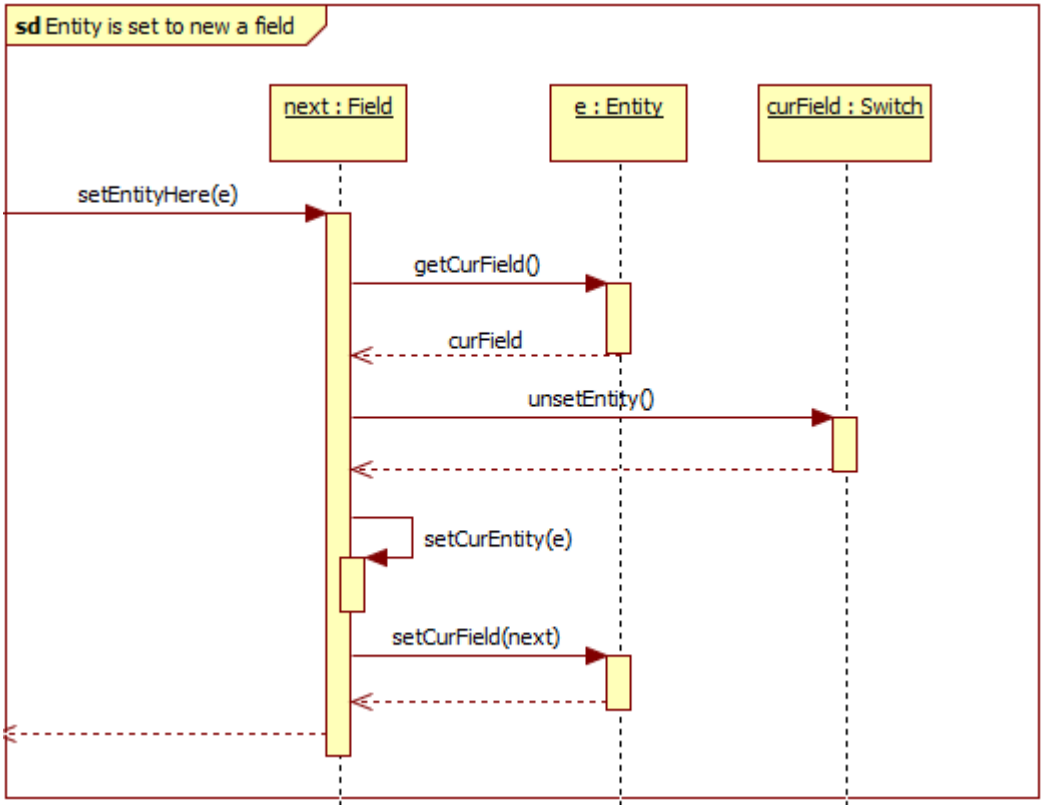
3.4.21 Láda kapcsolóra lép



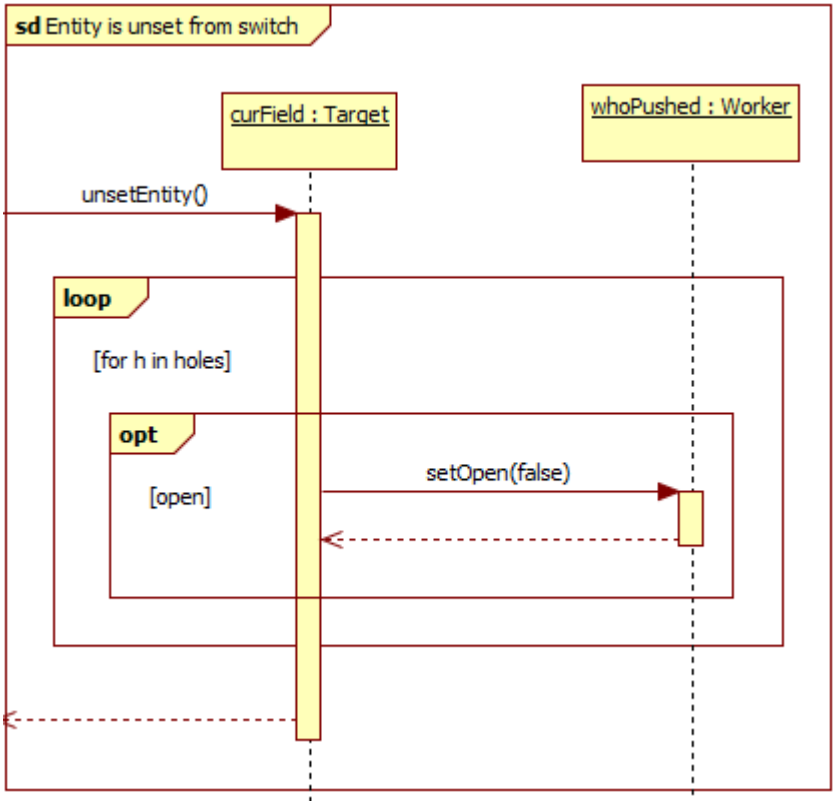
3.4.22 Dolgozó életet veszít



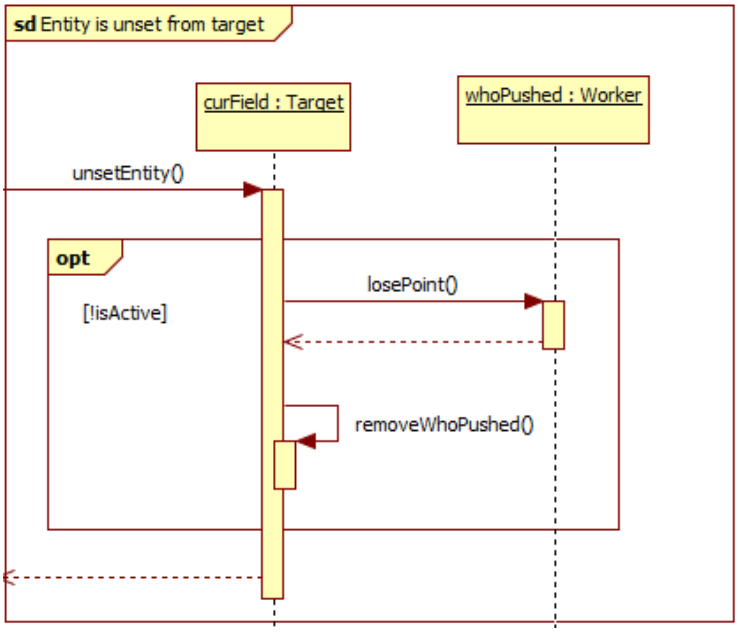
3.4.23 Entitás új mezőre kerül



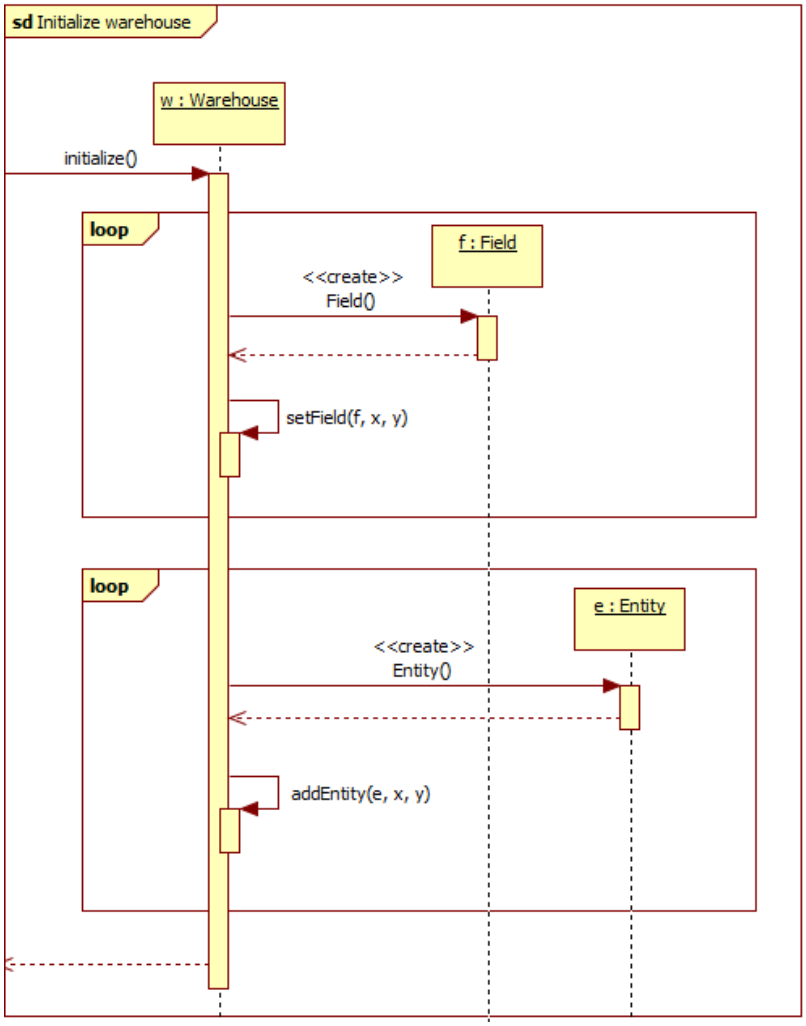
3.4.24 Entitás új mezőre kerül egy kapcsolóról



3.4.25 Entitás új mezőre kerül egy előírt helyről

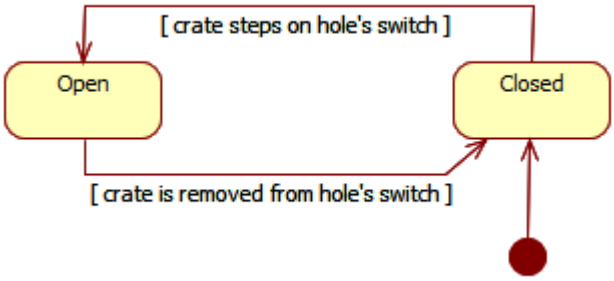


3.4.26 Raktárépület inicializálása



3.5 State-chartok

3.5.1 Lyuk állapota változik



3.6 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.02.19. 12:00	1 óra	CSANÁDY JANI LAKATOS LENKEFI	Értekezlet, melynek döntései: SZAKÁLLAS elkészíti az objektum katalógust CSANÁDY ellenőrzi, kiegészíti az objektum katalógust
2018.02.22. 14:00	1 óra	SZAKÁLLAS	Objektum katalógus pontjainak előzetes elkészítése (3.1)
2018.02.22. 19:00	1 óra	CSANÁDY	Objektum katalógus ellenőrzése, kiegészítése (3.1)
2018.02.23 14:00	2 óra	JANI LAKATOS LENKEFI	Értekezlet, melynek döntései: JANI elkészíti az osztálydiagramot. LAKATOS elkészíti az osztályok leírását LENKEFI ellenőrzi, és teszteli a statikus struktúra tervezése során felmerült ötleteket SZAKÁLLAS elkészíti a szekvencia diagramokat
2018.02.23 16:00	2 óra	LENKEFI	A statikus struktúra tervének ellenőrzése és tesztelése
2018.02.23. 21:00	1 óra	LAKATOS	Osztályok leírásának elkészítése (3.3)
2018.02.24. 10:00	2 óra	LAKATOS	Osztályok leírásának elkészítése (3.3)
2018.02.24. 16:00	4 óra	JANI	Osztálydiagram elkészítése (3.2)
2018.02.24. 21:00	7 óra	SZAKÁLLAS	Szekvencia diagramok készítése (3.4)
2018.02.25. 12:00	3 óra	LAKATOS	Szekvencia diagramok ellenőrzése, kiegészítése (3.4)
2018.02.25. 15:00	1 óra	LAKATOS	Konzisztencia ellenőrzése, apróbb módosítások. (3.2, 3.3, 3.4)
2018.02.25. 15:00	1 óra	SZAKÁLLAS	Szekvencia diagramok javítása (3.4)
2018.02.25. 18:00	1 óra	JANI	Konzisztencia ellenőrzése

2018.02.25. 18:00	1 óra	LENKEFI	State-chart diagram elkészítése (3.5)
2018.02.25. 19:00	1 óra	CSANÁDY	Általános ellenőrzés, apróbb javítások
2018.02.26 00:00	1 óra	LAKATOS	Dokumentum véglegesítése

4. Analízis modell kidolgozása

4.1 Objektum katalógus

4.1.1 Dolgozó

A Dolgozó típus példányai azok, akiket a játékosok közvetlenül irányítanak a játéktérben. Minden Dolgozóhoz pontosan egy játékos tartozik. A Dolgozóknak több életük lehet (legfeljebb három), tudnak Ládákat tologatni és közvetetten el is lehet őket tolni más Dolgozók által.

4.1.2 Láda

A Dolgozók Láda objektumokat tologatnak túlnyomó részt a játékban. A Ládák másik Ládát vagy Dolgozót is letolhatnak, ők azonban csak tolás hatására tudnak mozogni, mást mozgatni. Nem összenyomhatók, a játéktérből pedig csak úgy kerülhetnek ki, ha Lyukba esnek.

4.1.3 Szívecskés láda

A Szívecskés Láda a Láda egy olyan típusa, amit aktív Lyuk mezőre tolva, a tolást végző Dolgozó egy életet kap.

4.1.4 Mező

A Mező objektumok képezik a játéktér teljes részét, amiben egy mező jelent egy egységet. Négyzet alakúak és egyenként legfeljebb négy másik mezővel lehetnek szomszédosok, akiket tárolnak. A Mező objektum a játéktér alapegysége. Négyzet alakjukból kifolyólag legfeljebb négy Mezővel lehetnek szomszédosok, amelyeket ismernek, tárolnak.

4.1.5 Padló

A Padló a Mező objektum egy fajtája. A Padló objektumokon Dolgozók és Ládák egyaránt közlekedhetnek, egyéb ráhatások nélkül. Egy Padlón egyszerre csak egy Dolgozó vagy Láda állhat, akiket csak mozgatással vagy eltolással lehet onnan eltávolítani.

4.1.6 Lyuk

A Lyuk egyfajta Padló, amelyhez tartozhat egy viselkedését szabályozó kapcsoló. A kapcsolható lyukak lehetnek aktívak vagy inaktívak. Ha egy Lyuk aktív és egy Dolgozó rálép, a Dolgozó egy életet veszít, ha pedig Láda kerül a Lyukra, az eltűnik a játéktérből és nem kerül többé vissza. Inaktív esetben a Lyuk Padlóként funkcionál, a nem kapcsolható Lyukak pedig állandóan aktívak.

4.1.7 Kapcsoló

A Kapcsoló az egy fajta Padló. Egy vagy több Lyuk tartozik hozzá, amelyeket aktiválni vagy deaktiválni tud. A Kapcsoló állását egy Láda rátolásával lehet megváltoztatni. Ha a Láda lekerül a Kapcsolóról, a hozzá tartozó Lyukak ismét inaktívvá válnak.

4.1.8 Kiindulási mező

A Kiindulási mező objektumok speciális típusai a Padlónak. Egy Kiindulási Mezőhöz pontosan egy Dolgozó tartozik, aki erre mezőre kerül a játék kezdetén vagy életvesztés esetén. Kiindulási mezőn más Dolgozó vagy Láda objektum nem állhat, az ő számukra ez a mező Falként viselkedik.

4.1.9 Előírt hely

Az Előírt helyek olyan Padlók, amelyeket a Raktár jelöl ki. Ilyen típusú Padlóra tolva egy Ládát a tolást végző Dolgozó játékosa pontot kap. Ha egy Ládát eltolnak egy Előírt helyről, az érte pontot kapó játékos elveszíti a kapott pontot.

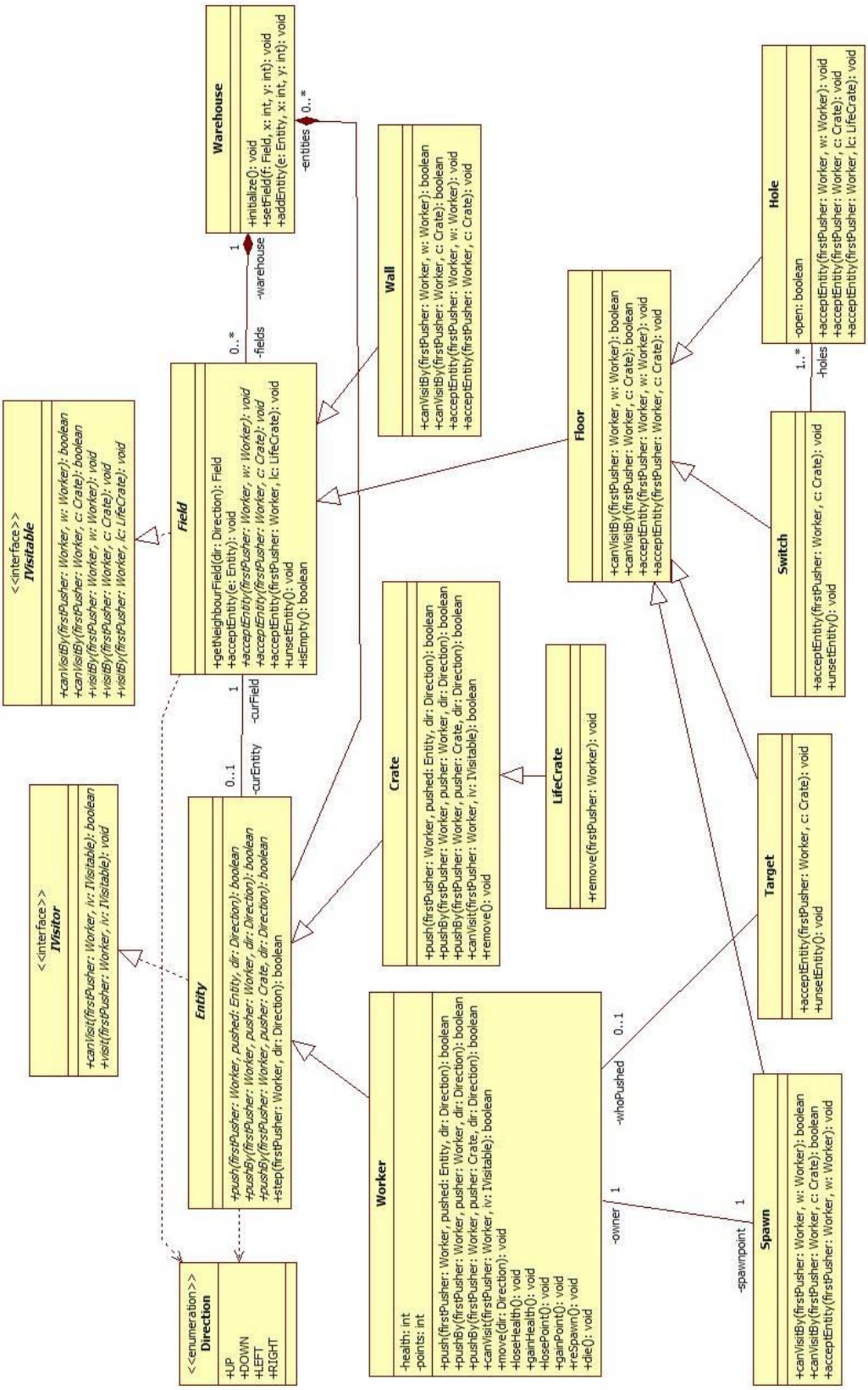
4.1.10 Fal

A Fal objektumok olyan Mezőket valósítanak meg, amelyeken nem tartózkodhat Dolgozó illetve Láda, azaz Dolgozók nem léphetnek rájuk és Ládákat sem lehet rájuk tolni. A játéktér széleit ilyen objektumok határolják, a játéktéren belül pedig akadályt jelentenek. Ezzel szemben Dolgozót rá lehet tolni Falra, aminek hatására az adott Dolgozó egy életet veszít.

4.1.11 Raktár

A Raktár tárolja a játékteret alkotó összes Mezőt, betölti a játék kezdetén azokat, kijelöli a Kiindulási mezőket, az Előírt helyeket és elhelyezi a Dolgozókat illetve a Ládákat.

4.2 Statikus struktúra diagramok


Warehouse
+initialize(): void
+setField(f: Field, x: int, y: int): void
+addEntity(e: Entity, x: int, y: int): void
-entities
0..*
1
-warehouse
0..*
-fields
Field
+getNeighbourField(dir: Direction): Field
+acceptEntity(e: Entity): void
+acceptEntity(firstPusher: Worker, w: Worker): void
+acceptEntity(firstPusher: Worker, c: Crate): void
+acceptEntity(firstPusher: Worker, lc: LifeCrate): void
+unsetEntity(): void
+isEmpty(): boolean
Wall
+canVisitBy(firstPusher: Worker, w: Worker): boolean
+canVisitBy(firstPusher: Worker, c: Crate): boolean
+acceptEntity(firstPusher: Worker, w: Worker): void
+acceptEntity(firstPusher: Worker, c: Crate): void
Hole
-open: boolean
+acceptEntity(firstPusher: Worker, w: Worker): void
+acceptEntity(firstPusher: Worker, c: Crate): void
+acceptEntity(firstPusher: Worker, lc: LifeCrate): void
1..*
-holes
Floor
+canVisitBy(firstPusher: Worker, w: Worker): boolean
+canVisitBy(firstPusher: Worker, c: Crate): boolean
+acceptEntity(firstPusher: Worker, w: Worker): void
+acceptEntity(firstPusher: Worker, c: Crate): void
Switch
+acceptEntity(firstPusher: Worker, c: Crate): void
+unsetEntity(): void
<<interface>>
IVisitable
+canVisitBy(firstPusher: Worker, w: Worker): boolean
+canVisitBy(firstPusher: Worker, c: Crate): boolean
+visitBy(firstPusher: Worker, w: Worker): void
+visitBy(firstPusher: Worker, c: Crate): void
+visitBy(firstPusher: Worker, lc: LifeCrate): void
<<interface>>
IVisitor
+canVisit(firstPusher: Worker, iv: IVisitable): boolean
+visit(firstPusher: Worker, iv: IVisitable): void
Entity
+push(firstPusher: Worker, pushed: Entity, dir: Direction): boolean
+pushBy(firstPusher: Worker, pusher: Worker, dir: Direction): boolean
+pushBy(firstPusher: Worker, pusher: Crate, dir: Direction): boolean
+step(firstPusher: Worker, dir: Direction): boolean
0..1
-curEntity
1
-curField
Crate
+push(firstPusher: Worker, pushed: Entity, dir: Direction): boolean
+pushBy(firstPusher: Worker, pusher: Worker, dir: Direction): boolean
+pushBy(firstPusher: Worker, pusher: Crate, dir: Direction): boolean
+canVisit(firstPusher: Worker, iv: IVisitable): boolean
+remove(): void
LifeCrate
+remove(firstPusher: Worker): void
Target
+acceptEntity(firstPusher: Worker, c: Crate): void
+unsetEntity(): void
Worker
-health: int
-points: int
+push(firstPusher: Worker, pushed: Entity, dir: Direction): boolean
+pushBy(firstPusher: Worker, pusher: Worker, dir: Direction): boolean
+pushBy(firstPusher: Worker, pusher: Crate, dir: Direction): boolean
+canVisit(firstPusher: Worker, iv: IVisitable): boolean
+move(dir: Direction): void
+loseHealth(): void
+gainHealth(): void
+losePoint(): void
+gainPoint(): void
+reSpawn(): void
+die(): void
0..1
-whoPushed
1
-owner
Spawn
+canVisitBy(firstPusher: Worker, w: Worker): boolean
+canVisitBy(firstPusher: Worker, c: Crate): boolean
+acceptEntity(firstPusher: Worker, w: Worker): void
1
-spawnpoint
<<enumeration>>
Direction
+UP
+DOWN
+LEFT
+RIGHT

Megjegyzés: a nevesített asszociációvégekhez és attribútumokhoz implicit getter és setter függvények tartoznak, amelyeket a diagram az olvashatóság kedvéért nem jelöl.

4.3 Osztályok leírása

4.3.1 Crate

- **Felelősség**

Egy ládát szimbolizál. Nyilvántartja a láda pozícióját.

- **Interfészek**

IVisitor

- **Ősosztályok**

Entity

- **Attribútumok**
  - **-**

- **Metódusok**
  - **boolean canVisit(Worker firstPusher, IVisitable iv)**: Láda megkérdezi, hogy van-e lehetősége átlépni *iv*-re, azaz képes-e *iv* ládát fogadni. Igazzal tér vissza, ha igen, egyébként hamissal.
  - **boolean push(Worker firstPusher, Entity pushed, Direction dir):** A láda eltol egy entitást a megadott irányban lévő szomszédos mezőre. Igazzal tér vissza, ha sikerült eltolni, egyébként hamissal.
  - **boolean pushBy(Worker firstPusher, Worker pusher, Direction dir):** Egy dolgozó tolja a ládát a megadott irányban lévő szomszédos mezőre, így a láda oda lép. Igazzal tér vissza, ha sikerült lépni, egyébként hamissal.
  - **boolean pushBy(Worker firstPusher, Crate pusher, Direction dir):** Egy láda tolja a ládát a megadott irányban lévő szomszédos mezőre, így a láda oda lép. Igazzal tér vissza, ha sikerült lépni, egyébként hamissal.
  - **void remove():** A láda kikerül a raktárból.

4.3.2 Entity

- **Felelősség**

Absztrakt osztály, ami a mozgatható dolgokat képviseli.

- **Interfészek**

IVisitor

- **Ősosztályok**

-

- **Attribútumok**
  - **Field curField**: A mező, amelyen jelenleg az entitás tartózkodik.

- **Metódusok**
  - **abstract boolean push(Worker firstPusher, Entity pushed, Direction dir):** Megpróbál eltolni egy megadott irányban lévő mezőn tartózkodó entitást, szintén a megadott irányban. Átveszi a dolgozót, aki a láncot tolja. Igazzal tér vissza, amennyiben sikerült eltolnia az entitást, egyébként hamissal.
  - **abstract boolean pushBy(Worker firstPusher, Worker pusher, Direction dir):** Az entitást megpróbálja eltolni egy dolgozó a megadott irányban. Átveszi a dolgozót, aki a láncot tolja és az entitást tolni próbáló dolgozót. Igazzal tér vissza, amennyiben a tolás sikeres volt, egyébként hamissal.
  - **abstract boolean pushBy(Worker firstPusher, Crate pusher, Direction dir):** Az entitást megpróbálja eltolni egy láda a megadott irányban. Átveszi a dolgozót, aki a láncot tolja és az entitást tolni próbáló ládát. Igazzal tér vissza, amennyiben a tolás sikeres volt, egyébként hamissal.
  - **boolean step(Worker firstPusher, Direction dir):** Az entitás lép a megadott irányban lévő szomszédos mezőre. Ha léphet a következő mezőre, de ott tartózkodik egy entitás, akkor megpróbálja őt eltolni. Igazzal tér vissza, ha sikerült lépni, egyébként hamissal.
  - **void setCurField(Field f):** Beállítja az entitás mezőjét.
  - **Field getCurField():** Lekérdezi az entitás mezőjét.

4.3.3 Field

- **Felelősség**

Absztrakt osztály, ami a játéktér (raktár) egy mezőjét szimbolizálja.

- **Interfészek**

IVisitable

- **Ősosztályok**

-

- **Attribútumok**
  - **Entity curEntity**: Az az entitás, ami jelenleg ezen a mezőn tartózkodik.

- **Metódusok**
  - **void acceptEntity(Entity e)**: Egy entitás a mezőre lép.
  - **abstract void acceptEntity(Worker firstPusher, Worker w)**: Egy dolgozó a mezőre lép. Átveszi a dolgozót, aki a láncot tolja.
  - **abstract void acceptEntity(Worker firstPusher, Crate c)**: Egy láda a mezőre lép. Átveszi a dolgozót, aki a láncot tolja.
  - **void acceptEntity(Worker firstPusher, LifeCrate lc)**: Egy szívecskés láda a mezőre lép. Átveszi a dolgozót, aki a láncot tolja.
  - **Field getNeighbourField(Direction dir):** Az mező megadja az adott irányban lévő szomszédját.

- **void setCurEntity(Entity e):** Beállítja a mezőn lévő entitást.
- **Entity getCurEntity(Entity e):** Lekérdezi a mezőn lévő entitást.
- **void unsetEntity():** A mezőn lévő entitás lekerül a mezőről, ezek után a mező üres lesz.
- **boolean isEmpty():** Megadja, hogy van-e épp entitás a mezőn. Ha van, hamissal tér vissza, egyébként igazzal.

4.3.4 Floor

- **Felelősség**

A játéktér egy padlóját képviseli. Egy padlón tartózkodhat entitás, így rá is lehet lépni.

- **Interfészek**

IVisitable

- **Ősosztályok**

Field

- **Attribútumok**
  - **Entity curEntity**: Az az entitás, ami jelenleg ezen a mezőn tartózkodik.
- **Metódusok**
  - **void acceptEntity(Worker firstPusher, Worker w)**: Egy dolgozó a padlóra lép.
  - **void acceptEntity(Worker firstPusher, Crate c)**: Egy láda a padlóra lép.
  - **boolean canVisitBy(Worker firstPusher, Worker w):** Megadja, hogy egy dolgozó rá tud-e lépni a padlóra. Ez mindig teljesül.
  - **boolean canVisitBy(Worker firstPusher, Crate c):** Megadja, hogy egy láda rá tud-e lépni a padlóra. Ez mindig teljesül.

4.3.5 Hole

- **Felelősség**

A játéktér egy lyukas padlóját képviseli, ami lehet nyitott, illetve zárt állapotban. Zárt állapotban ugyanúgy viselkedik, mint egy egyszerű padló. Nyitott állapotban mind ha dolgozó, mind ha láda lép rá, az leesik.

- **Interfészek**

IVisitable

- **Ősosztályok**

Field ⇨ Floor

- **Attribútumok**
  - **boolean open:** Igaz, amennyiben a lyuk nyitott állapotban van, egyébként hamis.

- **Metódusok**
  - **void acceptEntity(Worker firstPusher, Worker w)**: Egy dolgozó a lyukra lép. Ha nyitva van, a dolgozó leesik és életet veszít.
  - **void acceptEntity(Worker firstPusher, Crate c)**: Egy láda a lyukra lép. Ha nyitva van, a láda leesik.
  - **void acceptEntity(Worker firstPusher, LifeCrate lc)**: Egy szívecskés láda a lyukra lép. Ha nyitva van, a szívecskés láda leesik.
  - **void setOpen(boolean o):** Beállítja a lyuk állapotát.

4.3.6 IVisitable

- **Felelősség**

Visitor pattern egyik fele, ami az átlépés lehetőségét adja vissza. Ez a fél a lehetőség meglétének eldöntéséért felel.

- **Ősosztályok**

-

- **Metódusok**
  - **abstract boolean canVisitBy(Worker firstPusher, Worker w)**: Egy dolgozó átlépési lehetőségét adja vissza.
  - **abstract boolean canVisitBy(Worker firstPusher, Crate c)**: Egy láda átlépési lehetőségét adja vissza.
  - **abstract void visitBy(Worker firstPusher, Worker w)**: Egy visitálható objektumot meglátogat egy dolgozó.
  - **abstract void visitBy(Worker firstPusher, Crate c)**: Egy visitálható objektumot meglátogat egy láda.
  - **abstract void visitBy(Worker firstPusher, LifeCrate lc)**: Egy visitálható objektumot meglátogat egy szívecskés láda.

4.3.7 IVisitor

- **Felelősség**

Visitor pattern egyik fele, ami az átlépés lehetőségét adja vissza. Ez a fél a dinamikus típussal való visszahívásért felel.

- **Ősosztályok**

-

- **Metódusok**
  - **abstract boolean canVisit(Worker firstPusher, IVisitable iv)**: Dinamikus típussal való visszahívást tartalmazza, a visszatérési érték az átlépési lehetőséget fejezi ki.

- **abstract void visit(Worker firstPusher, IVisitable iv):** Egy visitálni képes objektum meglátogat egy visitálható objektumot.

4.3.8 LifeCrate

- **Felelősség**

Egy szívecske szimbólummal rendelkező ládát szimbolizál. Ez a láda egy hagyományos ládaként viselkedik, de ha leesik egy lyukba, akkor az őt leejtő dolgozó kap egy életet.

- **Interfészek**

IVisitor

- **Ősosztályok**

Entity

- **Attribútumok**
  - **-**
- **Metódusok**
  - **void remove(Worker firstPusher):** A szívecskés láda kikerül a raktárból, miközben a paraméterként átvett dolgozónak életet ad.

4.3.9 Spawn

- **Felelősség**

A játéktér egy olyan padlóját képviseli, ami egy dolgozó kiindulási mezője. Minden dolgozó saját kiindulási mezővel rendelkezik, innen kezdik a játékot, és innen is folytatják életvesztés után. Erre a mezőre kizárólag a hozzá tartozó dolgozó léphet, más entitás nem.

- **Interfészek**

IVisitable

- **Ősosztályok**

Field ⇨ Floor

- **Attribútumok**
  - **Worker owner:** Az a dolgozó, melynek ez a kiindulási mezője.
- **Metódusok**
  - **boolean canVisitBy(Worker firstPusher, Worker w):** Megadja, hogy egy dolgozó rá tud-e lépni a kiindulási mezőre. Ez csak akkor teljesül, ha a dolgozónak ez a kiindulási mezője vagy ha a dolgozót láncban tolják.

- **boolean canVisitBy(Worker firstPusher, Crate c):** Megadja, hogy egy láda rá tud-e lépni a kiindulási mezőre. Ez sosem teljesül.
- **void acceptEntity(Worker firstPusher, Worker w)**: Egy dolgozó a lyukra lép. Ha a dolgozónak nem ez a kiindulási mezője, akkor ő nekiütközik a kiindulási mezőnek és életet veszít.

4.3.10 Switch

- **Felelősség**

A játéktér egy olyan padlóját képviseli, melyen egy kapcsoló található. Amennyiben erre a kapcsolóra egy láda lép, egy vagy több lyuk nyitott állapotba kerül. Ha a láda lekerül a mezőről, a kapcsoló által kinyitott lyukak bezáródnak. Ha dolgozó lép a kapcsolóra, nem kapcsol.

- **Interfészek**

IVisitable

- **Ősosztályok**

Field ⇨ Floor

- **Attribútumok**
  - **List<Hole> holes**: Azon lyukak, melyek ehhez a kapcsolóhoz vannak rendelve.
- **Metódusok**
  - **void acceptEntity(Worker firstPusher, Crate c)**: Egy láda a kapcsolóra lép. Ilyenkor a kapcsolóhoz rendelt lyukak nyitott állapotba kerülnek.
  - **void unsetEntity():** A mezőn lévő entitás lekerül a mezőről, ezek után a mező üres lesz. Ilyenkor a kapcsolóhoz rendelt lyukak zárt állapotba kerülnek.

4.3.11 Target

- **Felelősség**

A játéktér egy olyan padlóját képviseli, ami egy előírt hely a ládák számára. Ha egy dolgozó erre a mezőre tol egy ládát, akkor pontot kap érte, azonban, ha később valaki eltolja innen a ládát, az érte pontot kapó dolgozó elveszti a kapott pontot.

- **Interfészek**

IVisitable

- **Ősosztályok**

Field ⇨ Floor

- **Attribútumok**
  - **Worker whoPushed:** Az a dolgozó, amelyik pontot kapott egy láda ide tolásával.
- **Metódusok**
  - **void acceptEntity(Worker firstPusher, Crate c)**: Egy láda az előírt helyre lép. Ilyenkor a láncot toló dolgozó pontot kap.
  - **void unsetEntity():** A mezőn lévő entitás lekerül a mezőről, ezek után a mező üres lesz. Ilyenkor a dolgozó, aki eredetileg a helyére tolta a ládát, pontot veszít.
  - **void unsetWhoPushed():** Beállítja, hogy jelenleg nincs dolgozó aki ládát tolt volna az előírt helyre.
  - **void setWhoPushed(Worker w):** Beállítja, hogy melyik dolgozó tolta a ládát az előírt helyre.

4.3.12 Wall

- **Felelősség**

A játéktér egy olyan mezőjét képviseli, amire nem lehet lépni. Ha munkás próbálna meg közvetetten rálépni (tehát úgy, hogy egy láda tolja őt), akkor a munkás életet veszít.

- **Interfészek**

IVisitable

- **Ősosztályok**

Field

- **Attribútumok**
  - **-**
- **Metódusok**
  - **void acceptEntity(Worker firstPusher, Worker w)**: Egy dolgozó a falnak ütközik. Ilyenkor a dolgozó életet veszít.
  - **void acceptEntity(Worker firstPusher, Crate c)**: Egy láda a falnak ütközik. Ez jelenleg nem lehetséges.
  - **boolean canVisitBy(Worker firstPusher, Worker w)**: Megadja, hogy egy dolgozó rá tud-e lépni a falra. Ez csak akkor teljesül, ha a dolgozót láncban tolják.
  - **boolean canVisitBy(Worker firstPusher, Crate c)**: Megadja, hogy egy láda rá tud-e lépni a falra. Ez sosem teljesül.

4.3.13 Warehouse

- **Felelősség**

A játékteret szimbolizálja, tartalmazza az ott lévő mezőket.

- **Interfészek**

-

- **Ősosztályok**

-

- **Attribútumok**
  - **List<Field> fields:** A raktárépületben található mezők.
  - **List<Entity> entities:** A raktárépületben található entitások.

- **Metódusok**
  - **void initialize():** Inicializálja a játékteret, azzal felépíti a mezőket, és létrehozza az entitásokat.
  - **void setField(Field f, int x, int y):** Beállítja a pálya adott helyére a paraméterként kapott mezőt.
  - **void addEntity(Entity e, int x, int y):** Hozzáad egy entitást az adott pozíción levő mezőre.

4.3.14 Worker

- **Felelősség**

Egy dolgozót szimbolizál. Nyilvántartja a dolgozó pozícióját, életét, valamint pontszámát.

- **Interfészek**

IVisitor

- **Ősosztályok**

Entity

- **Attribútumok**
  - **unsigned int health**: A dolgozó élete. Ha eléri a nullát, a dolgozó meghal és kikerül a játékból.
  - **unsigned int points**: A dolgozó pontszáma. Minden egyes előírt helyre tolt ládával egyel nő, ám amennyiben azok a ládák később elmozdulnak az előírt helyekről, a pontok elvesznek.
  - **Spawn spawnpoint:** A munkás kiinduló mezője.

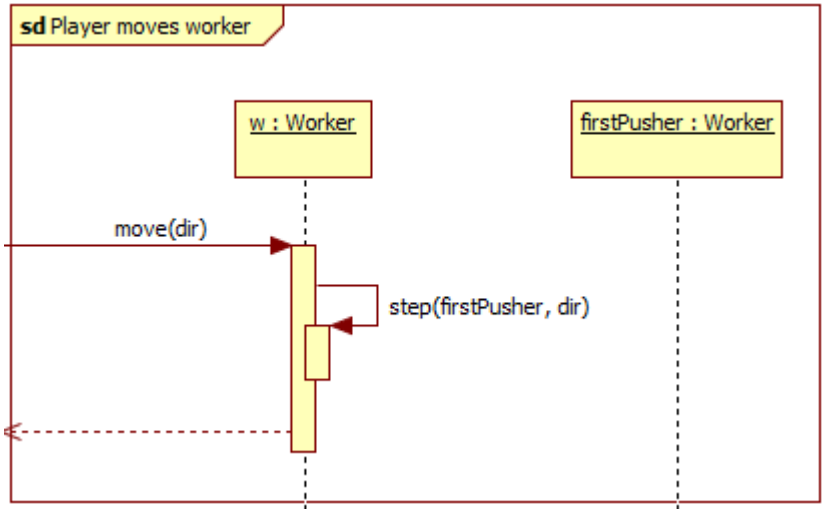
- **Metódusok**
  - **boolean canVisit(Worker firstPusher, IVisitable iv)**: Dolgozó megkérdezi, hogy van-e lehetősége átlépni *iv*-re, azaz képes-e *iv* dolgozót fogadni. Igazzal tér vissza, ha igen, egyébként hamissal.
  - **void move(dir Direction):** A játékos lépteti a dolgozót a megadott irányban.

- **boolean push(Worker firstPusher, Entity pushed, Direction dir)**: A dolgozó eltol egy entitást a megadott irányban lévő szomszédos mezőre. Igazzal tér vissza, ha sikerült eltolni, egyébként hamissal.
- **boolean pushBy(Worker firstPusher, Worker pusher, Direction dir)**: Egy dolgozó tolja a dolgozót a megadott irányban lévő szomszédos mezőre, így a dolgozó oda lép. Igazzal tér vissza, ha sikerült lépni, egyébként hamissal.
- **boolean pushBy(Worker firstPusher, Crate pusher, Direction dir)**: Egy láda tolja a dolgozót a megadott irányban lévő szomszédos mezőre, így a dolgozó oda lép. Igazzal tér vissza, ha sikerült lépni, egyébként hamissal.
- **void loseHealth():** A dolgozó egy életet veszít.
- **void gainHealth():** A dolgozó egy életet kap.
- **void losePoint():** A dolgozó egy pontot veszít.
- **void gainPoint():** A dolgozó egy pontot kap.
- **void reSpawn():** A dolgozó átkerül jelenlegi mezőjéről a számára kijelölt kiinduló mezőre.
- **void die():** A dolgozó meghal és kikerül a játékból.

4.4 Szekvencia diagramok

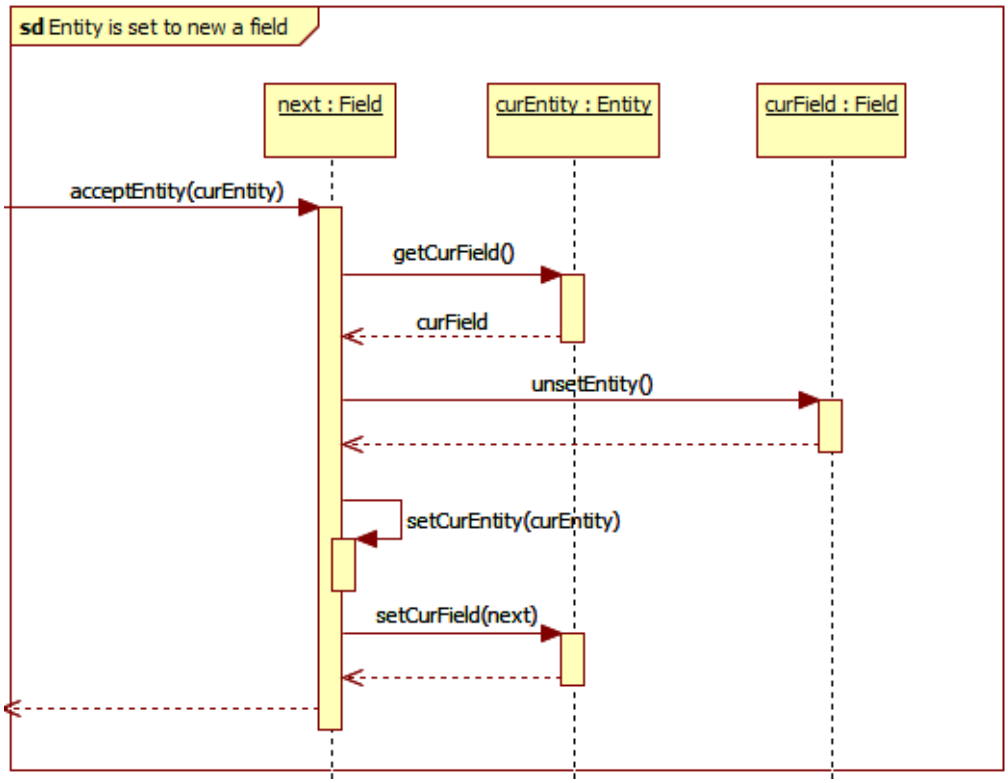
Megjegyzés: A „dobozokban” lévő aláhúzásokat nem lehetséges eltüntetni a WhiteStarUML-ben, ezért úgy tekintjük, mintha azok ott sem lennének, hiszen aláhúzva Object-eket reprezentálnának. Továbbá mivel az UML eszköz nem támogatja a szabványos create jelölést, így az metódushívásként van modellezve.

4.4.1 A játékos lépteti a dolgozót



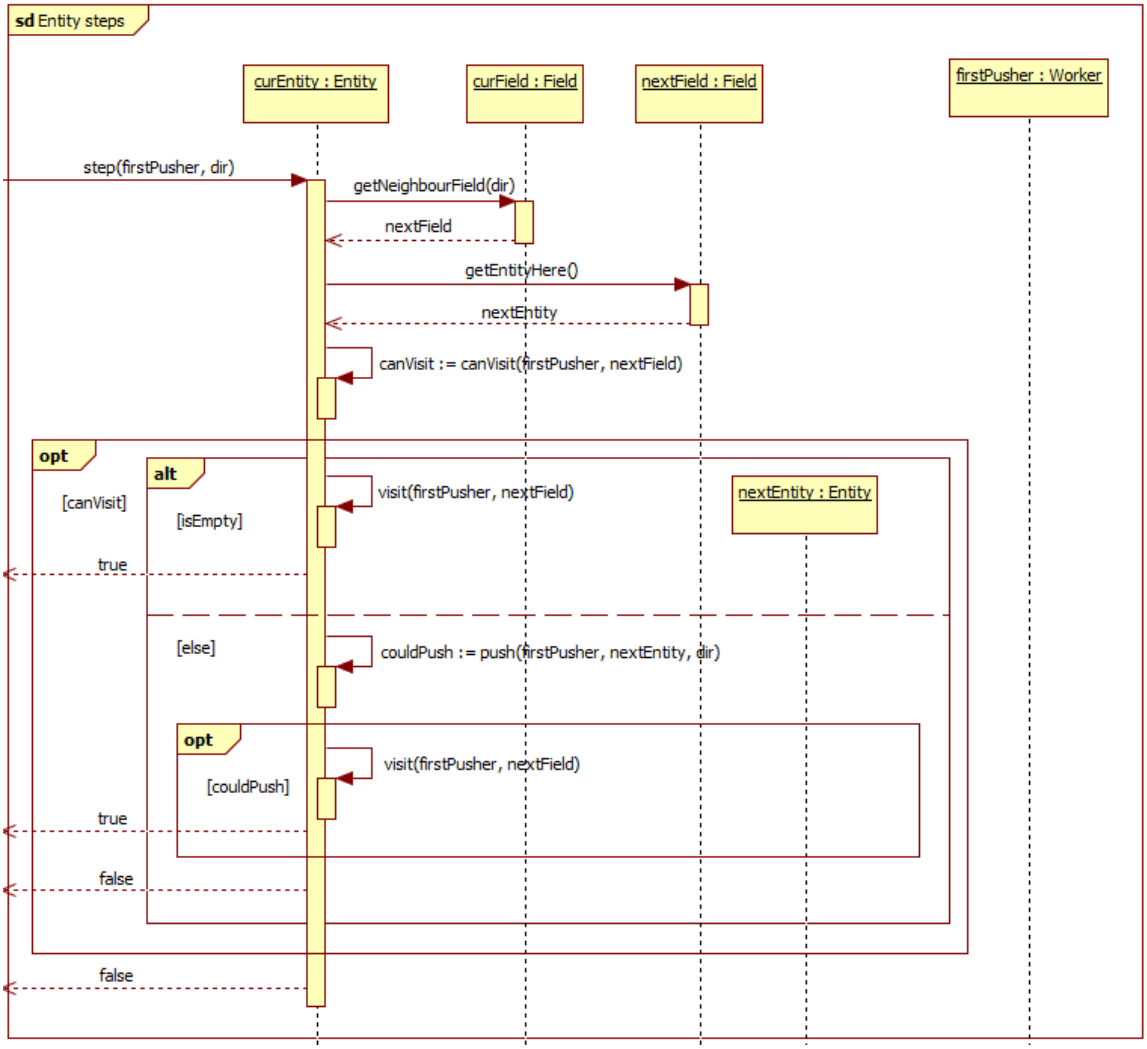
Ilyenkor megpróbálunk ellépni az adott irányban.

4.4.2 Entitás új mezőre kerül



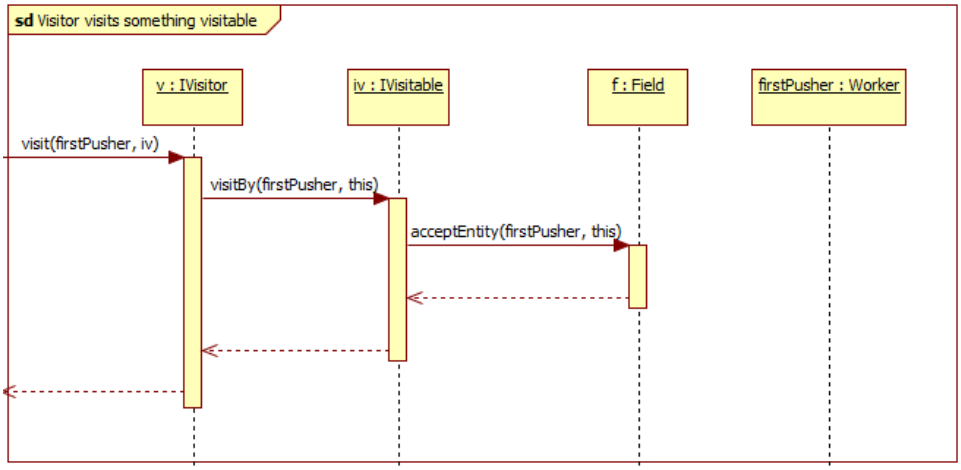
A belépéskor adminisztráljuk mind az entitás mezőre vonatkozó, mind a mező entitásra vonatkozó referenciájának megváltozását.

4.4.3 Entitás lép



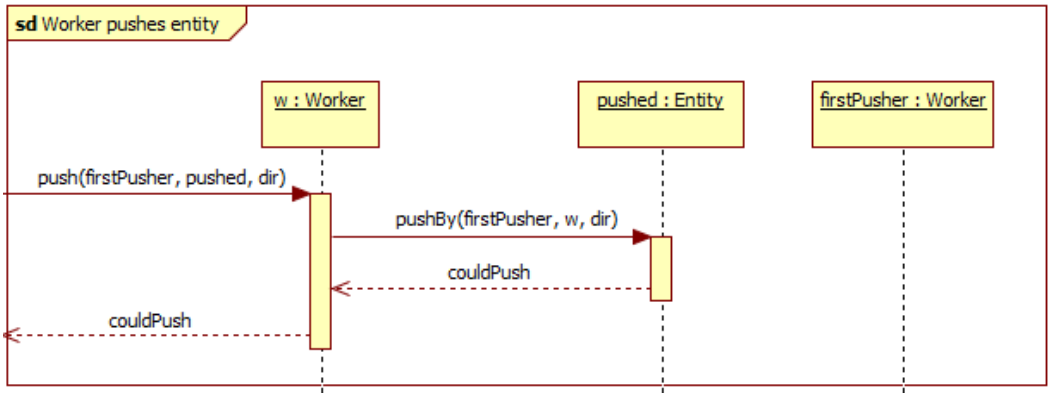
Léptetéskor két dolgot kell ellenőriznünk. Elsőnek, hogy a célmező egyáltalán képes-e fogadni minket (pl.: Spawn csak a saját játékosát fogadja be), és ha ez fennáll, akkor azt, hogy jelenleg szabad-e, ugyanis, ha másik entitás már ott tartózkodik, akkor meg kell próbálnunk eltolni. Ha sikerült a tolás, vagy eleve üres volt, akkor odalépünk, egyébként helyben maradunk.

4.4.4 Visitáló objektum visitálhatót látogat



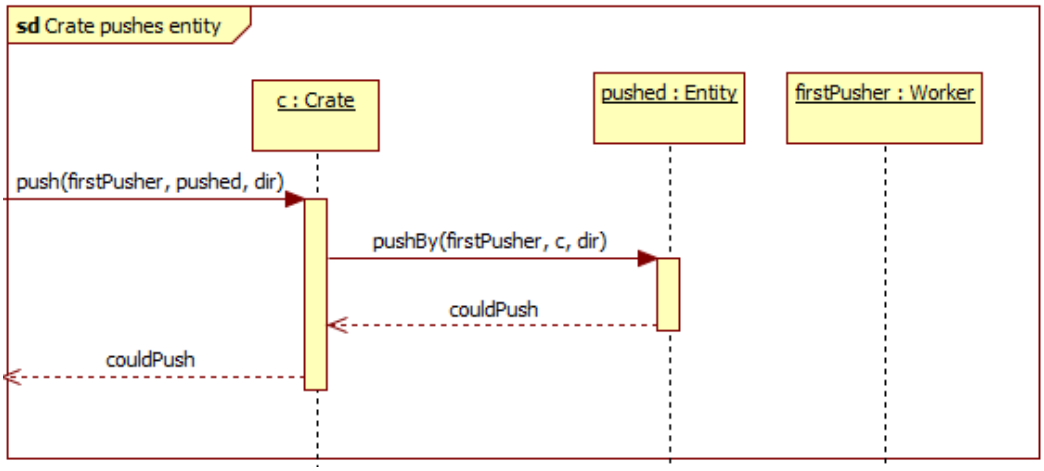
A visitálás megtörténik minden lépés „valós” lépés előtt, vagyis amikor már biztos, hogy az adott entitás a mezőre lép, annak érdekében, hogy kiderüljön, milyen típusú entitás érkezik a mezőre.

4.4.5 Dolgozó entitást tol



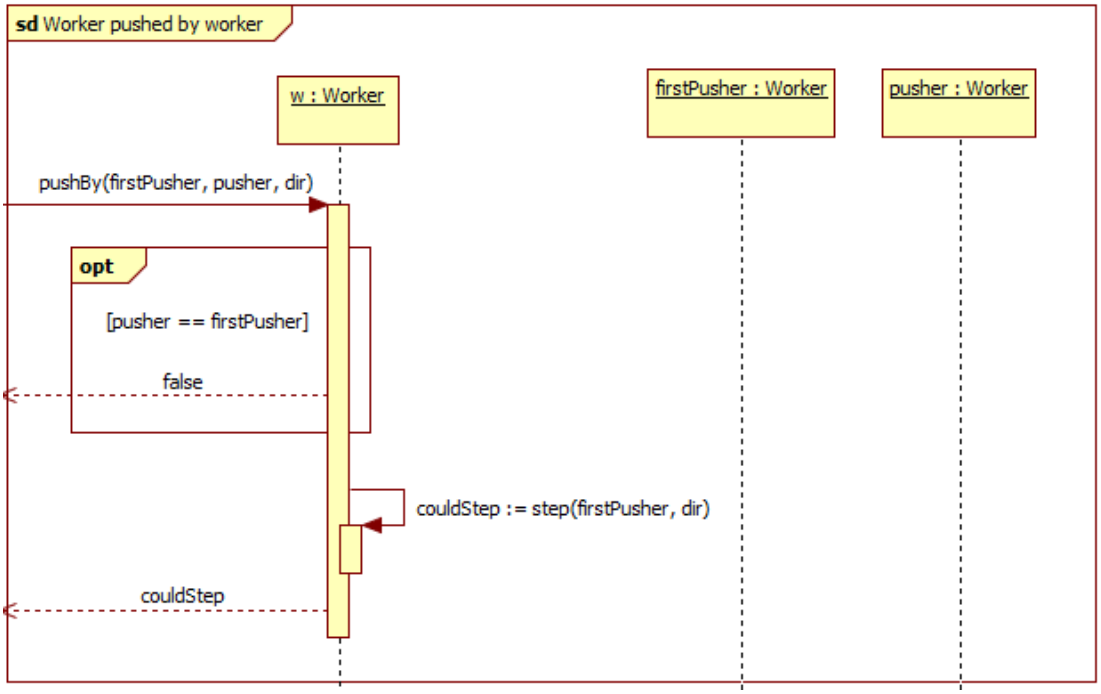
Mini “Visitor” -> Dinamikus típussal való hívás miatt szükséges függvény.

4.4.6 Láda entitást tol



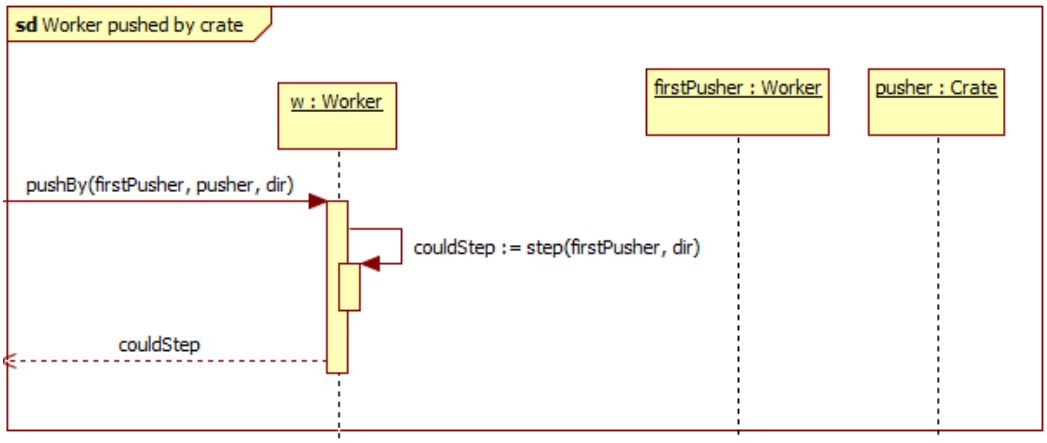
Mini "Visitor" -> Dinamikus típussal való hívás miatt szükséges függvény.

4.4.7 Dolgozót dolgozó tol



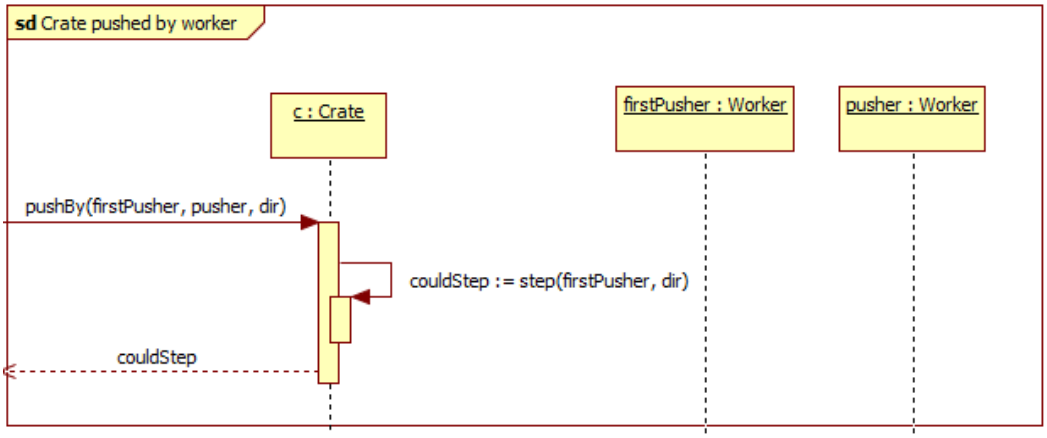
Dolgozó dolgozót csak akkor tolhat, ha láncba van, ezért ellenőrizzük, hogy aki itt tol az nem az első-e. Ha az első akkor, visszatérünk hamissal, ezzel megtagadva a tolást, ha nem akkor megpróbálunk ellépni a megfelelő irányba és ennek sikerességével térünk vissza.

4.4.8 Dolgozót láda tol



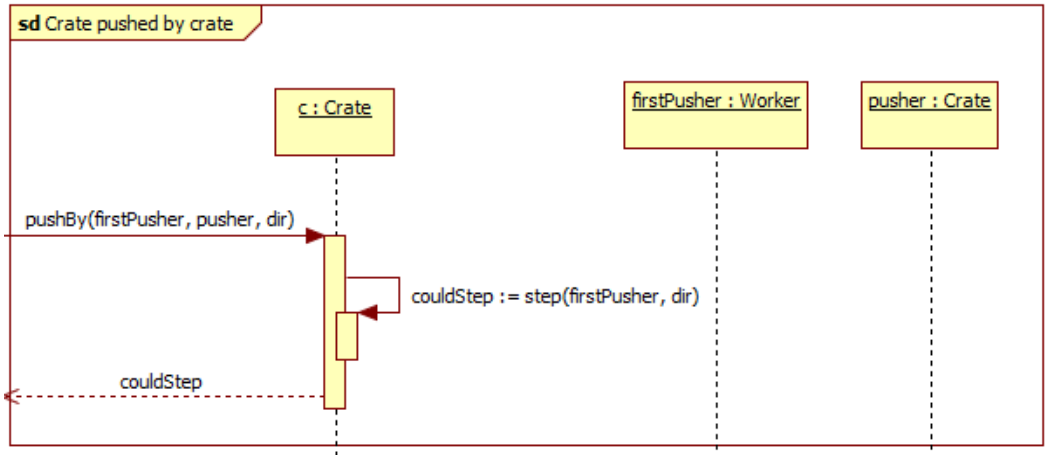
Mivel láda mindig képes dolgozót tolni, ezért megpróbálunk ellépni a megfelelő irányba és ennek sikerességével térünk vissza.

4.4.9 Ládát dolgozó tol



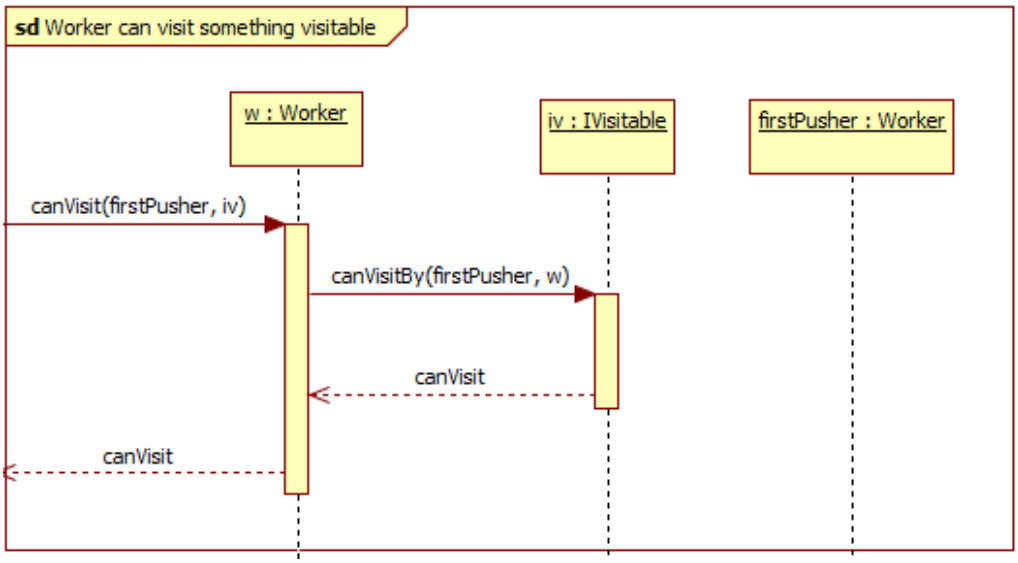
Mivel dolgozó mindig képes ládát tolni, ezért megpróbálunk ellépni a megfelelő irányba és ennek sikerességével térünk vissza.

4.4.10 Ládát láda tol



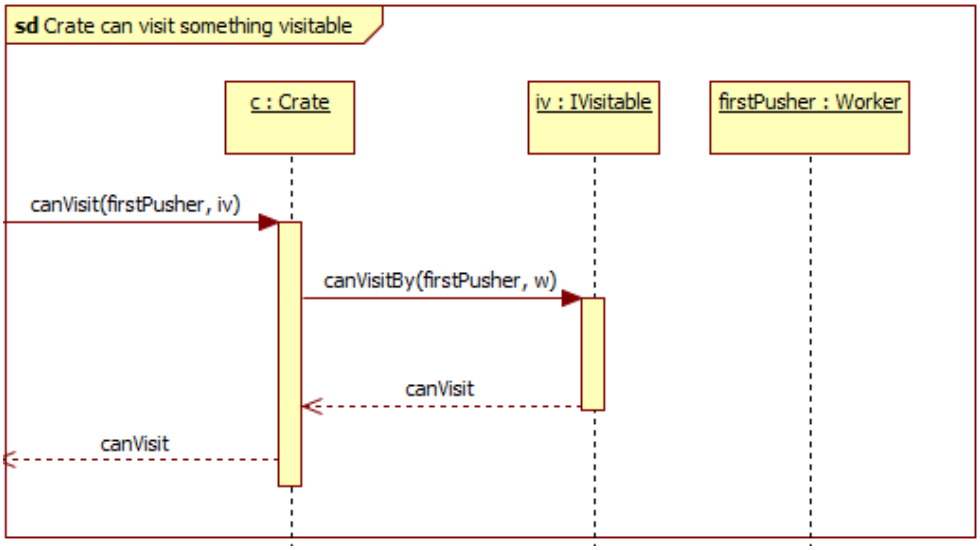
Mivel láda mindig képes ládát tolni, ezért megpróbálunk ellépni a megfelelő irányba és ennek sikerességével térünk vissza.

4.4.11 Dolgozó léphet-e valahova



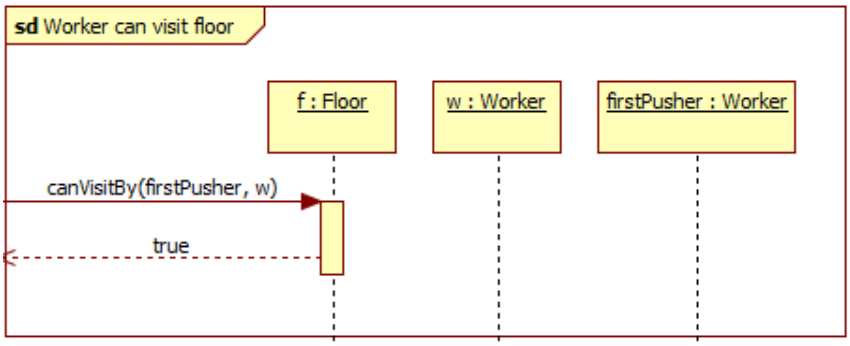
"Visitor" -> Dinamikus típussal való hívás megvalósítása.

4.4.12 Láda léphet-e valahova



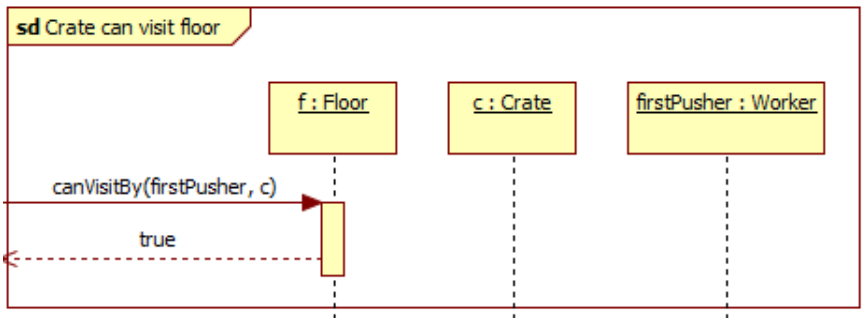
"Visitor" -> Dinamikus típussal való hívás megvalósítása.

4.4.13 Dolgozó léphet-e padlóra



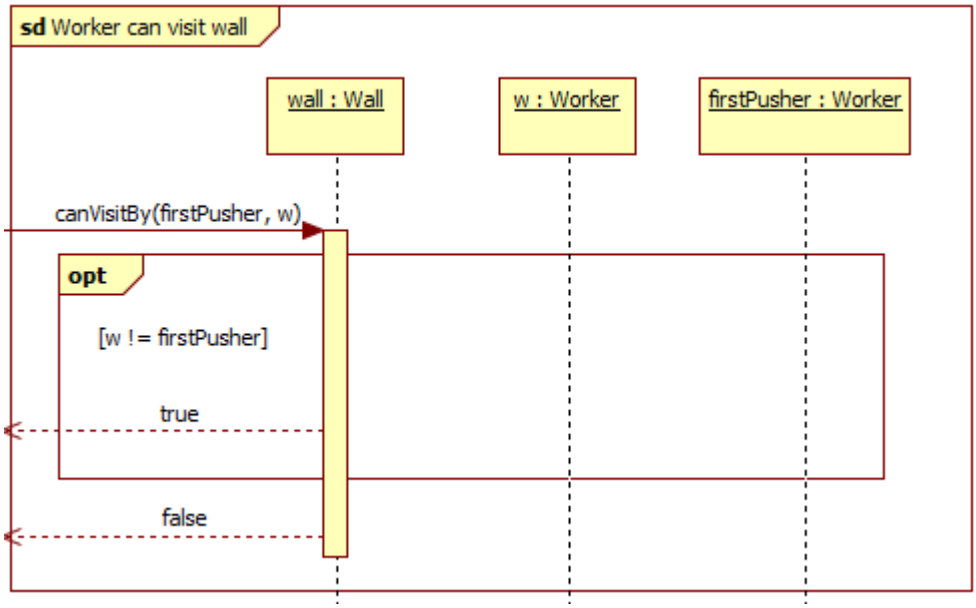
Dolgozónak mindig megvan a lehetősége, hogy padlóra lépjen.

4.4.14 Láda léphet-e padlóra



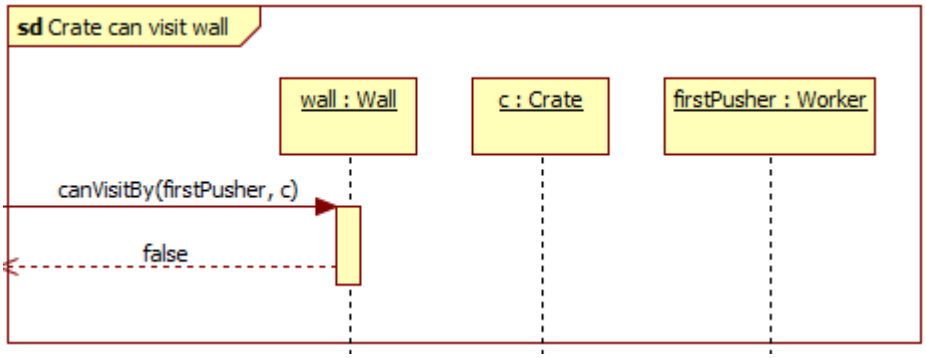
Ládának mindig megvan a lehetősége, hogy padlóra lépjen.

4.4.15 Dolgozó léphet-e falra



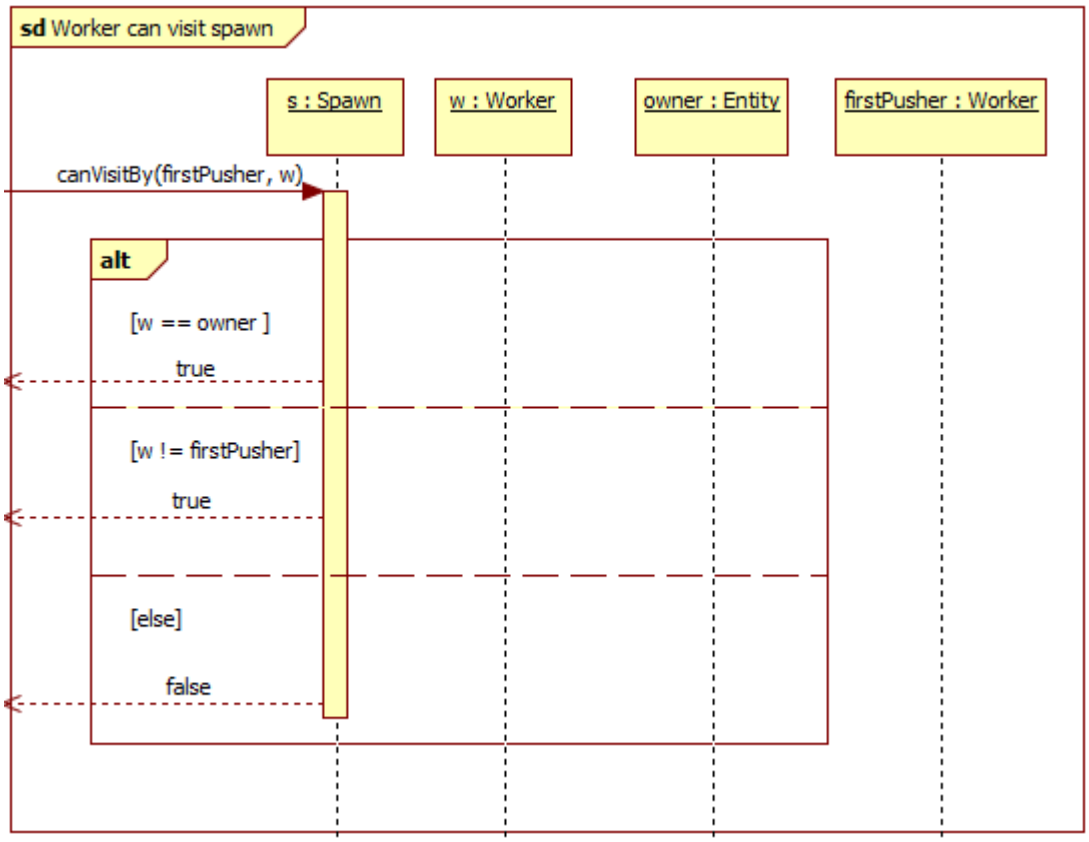
Csak akkor léphet falra (azaz nyomódhat szét a falon a) dolgozó, ha láncban van, így ezt ellenőrizzük azzal, hogy nem a dolgozó a firstPusher, és ennek eredményét adjuk vissza.

4.4.16 Láda léphet-e falra



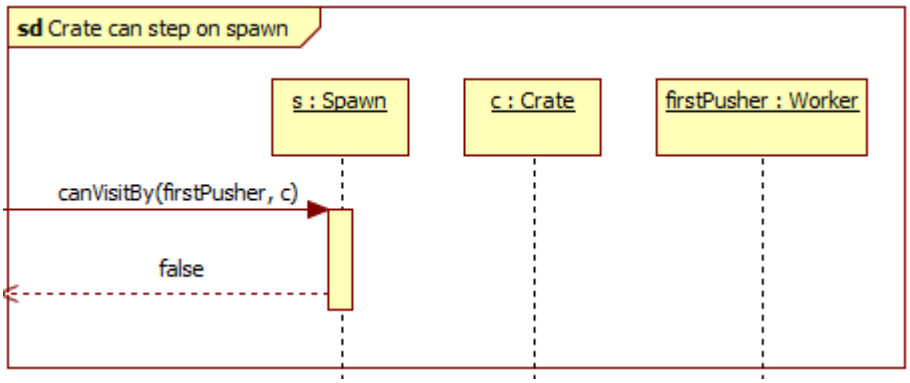
Láda soha nem léphet falra.

4.4.17 Dolgozó léphet-e kiindulási helyre



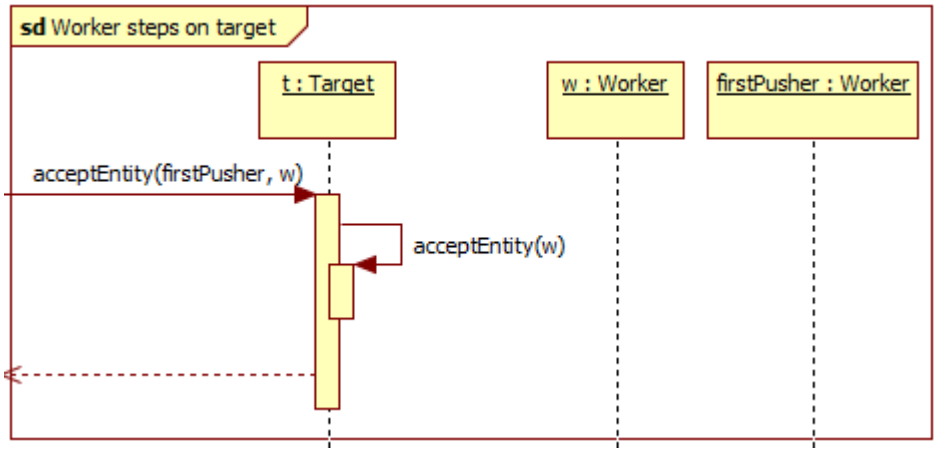
Dolgozók közül csak annak van lehetősége a kiindulási helyre lépni, ami vagy annak a tulajdonosa, vagy láncban tolják (ekkor felnyomódik a mezőre, mint egy falra). Ezt a feltételt kell kiértékelnünk és visszatérnünk a válasszal.

4.4.18 Láda léphet-e kiindulási helyre



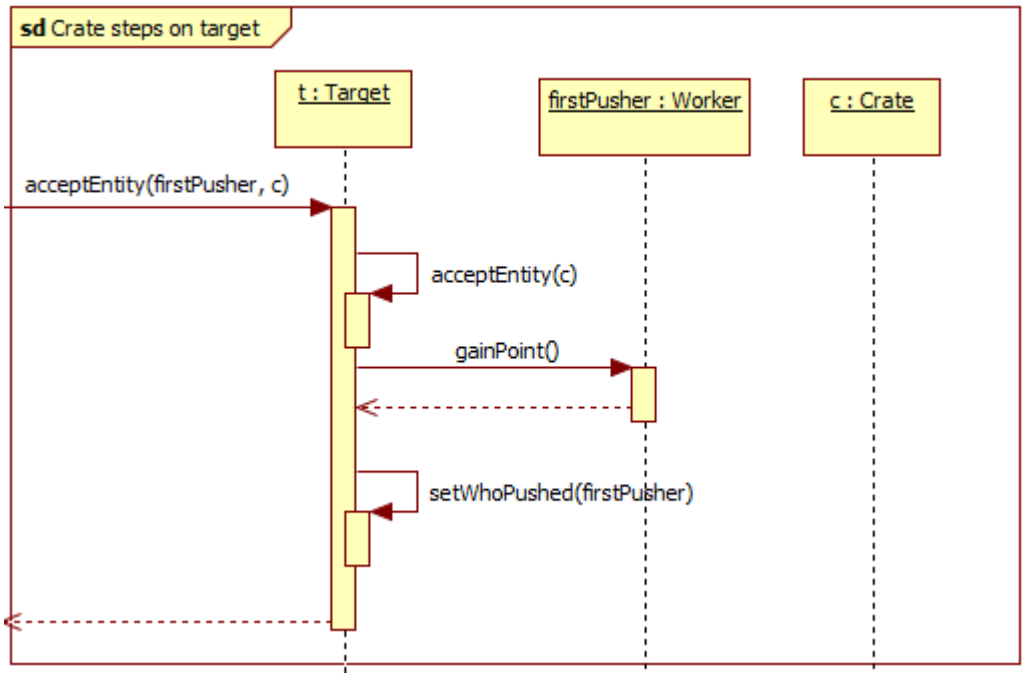
Láda soha nem léphet Spawnra.

4.4.19 Dolgozó előírt helyre lép



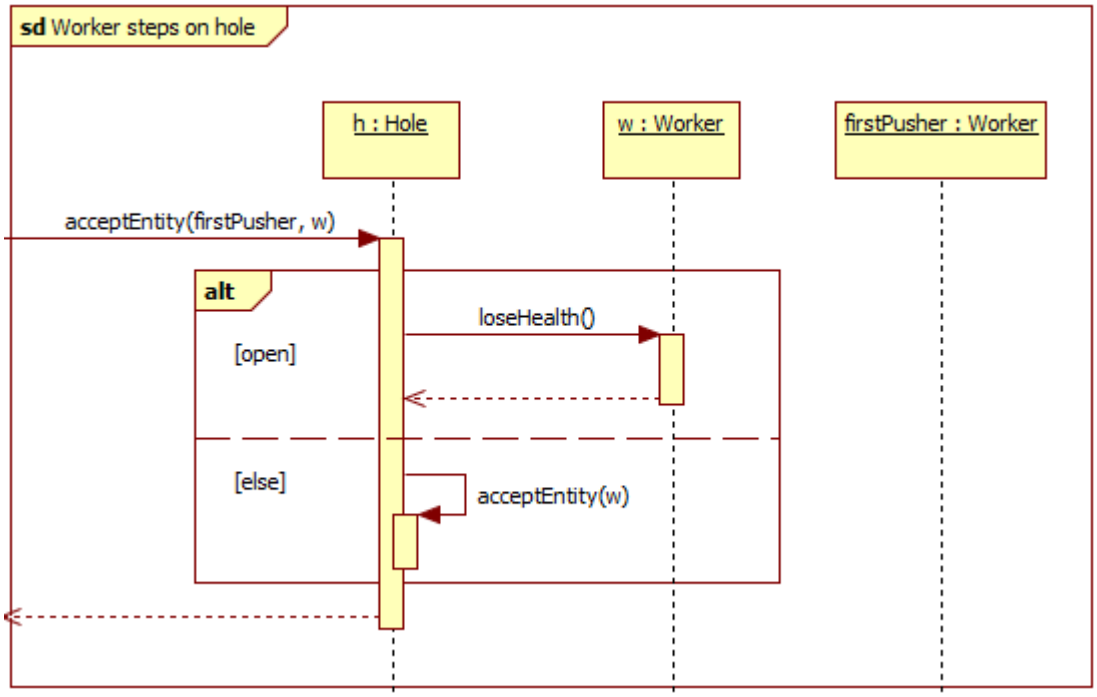
Olyan, mintha a dolgozó egy padlóra lépne.

4.4.20 Láda előírt helyre lép és pontot ad



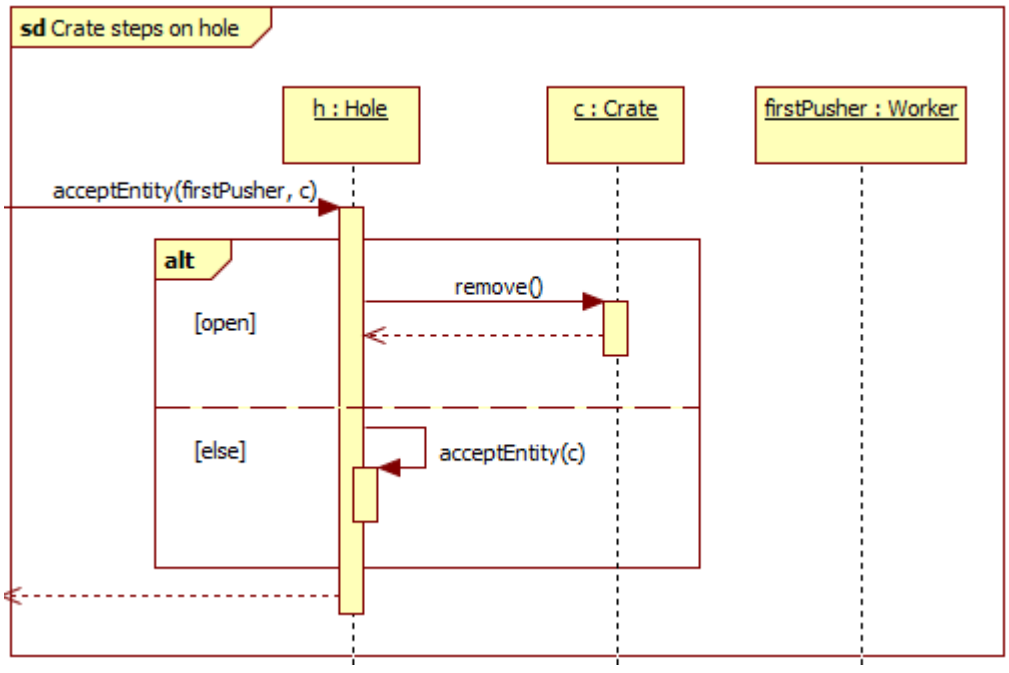
Mikor egy láda kilép innen, a szokásos adminisztráció során elvesszük a kiosztott pontot. Tehát beléptetéskor szokásos adminisztráción kívül, két dolgot kell tennünk, kiosztani a pontot annak, aki miatt idekerült a láda, valamint referenciáját eltárolni, hogy később, ha kell, eltudjuk venni a pontot.

4.4.21 Dolgozó lyukra lép és életet veszít



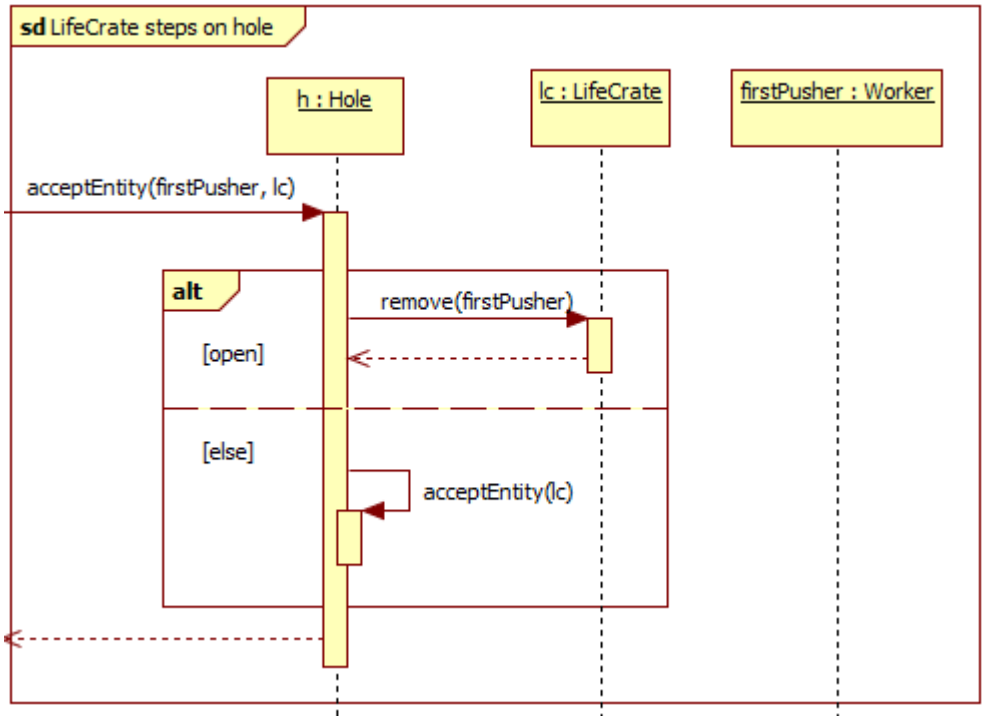
Ekkor, ha nyitva van a lyuk akkor a lépés hatására életet kell veszítenie a dolgozónak, ha zárva akkor a szokásos műveleteket kell elvégezni.

4.4.22 Láda lyukra lép és eltűnik



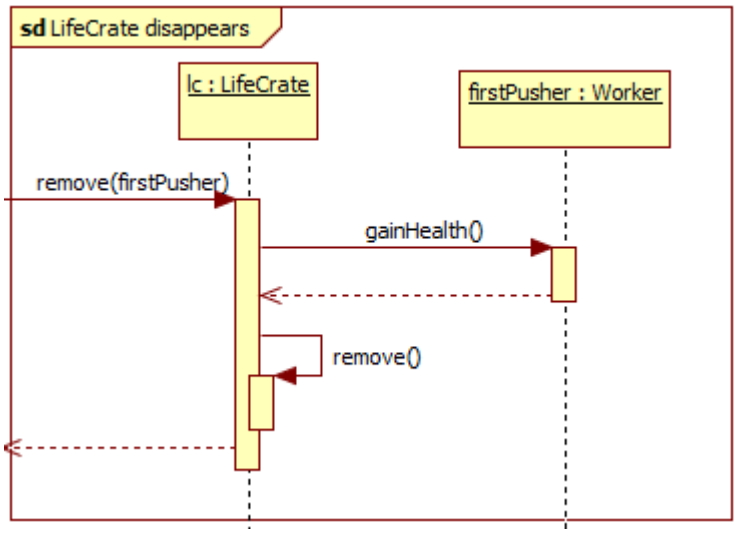
Ekkor, ha nyitva van a lyuk akkor a lépés hatására megsemmisül a láda, ha zárva akkor a szokásos műveleteket kell elvégezni.

4.4.23 Szívecskés láda lyukra lép és eltűnik



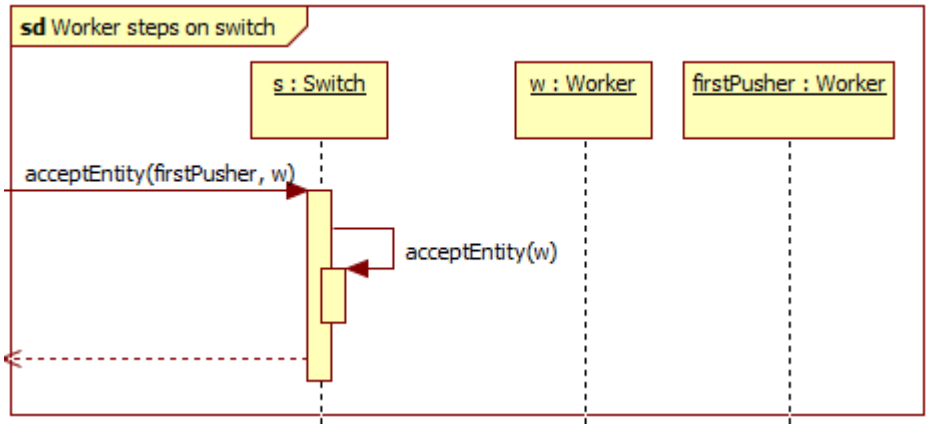
Ekkor, ha nyitva van a lyuk akkor a lépés hatására megsemmisül a láda és aki ezt okozta kap egy életet (remove függvényen belül), ha zárva akkor a szokásos műveleteket kell elvégezni.

4.4.24 Szívecskés láda eltűnik és életet ad



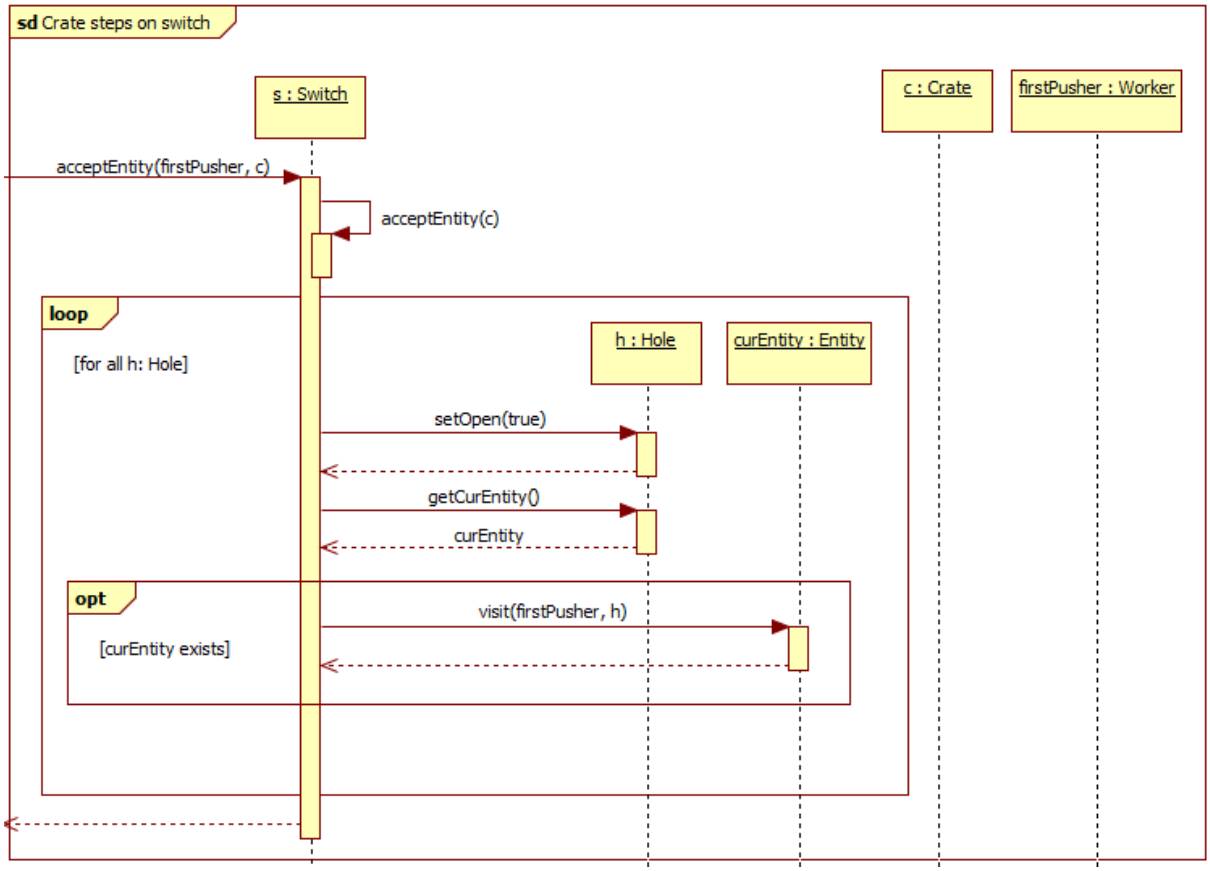
Szívecskés láda esetén a megsemmisüléskor az azt okozó kap egy életet, majd a szokásos módon eltűnik a láda.

4.4.25 Dolgozó kapcsolóra lép



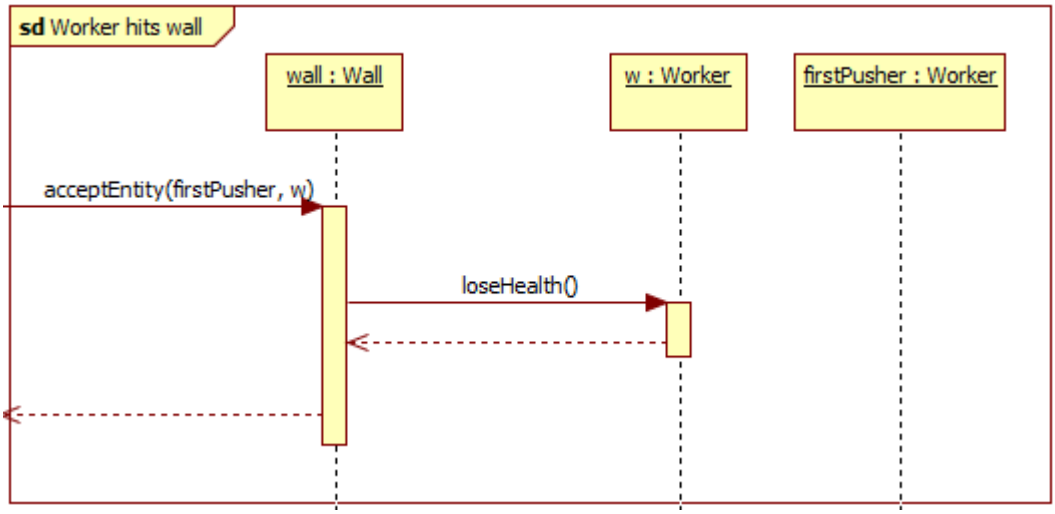
A dolgozó nem képes kapcsolni a kapcsolót, ezért csak a szokásos lépések történnek.

4.4.26 Láda kapcsolóra lép és kapcsol



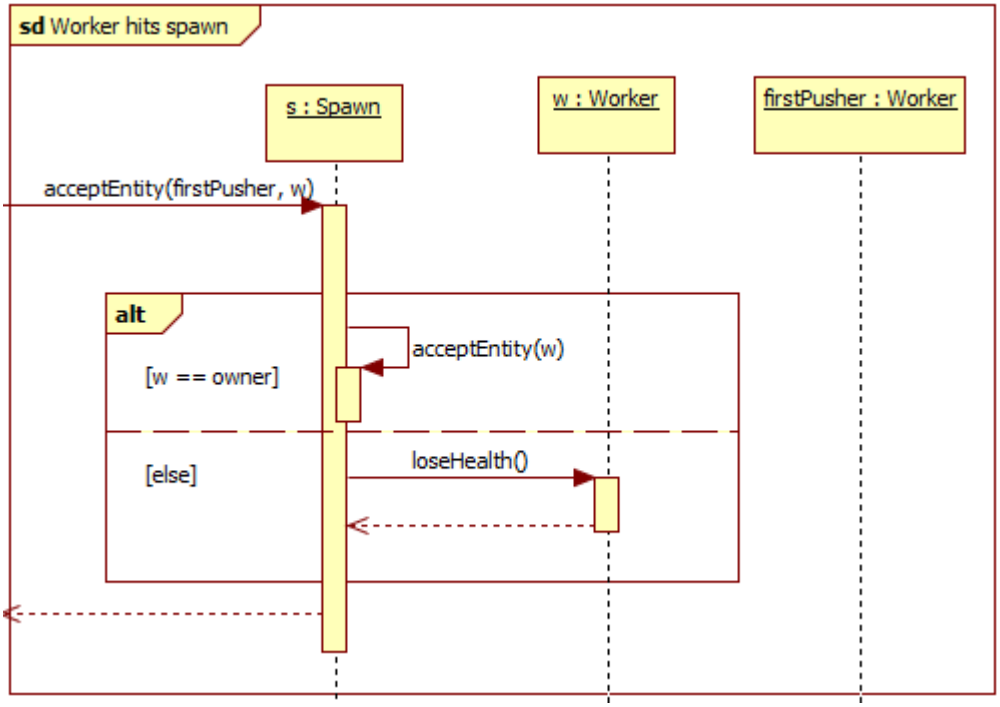
Ha láda egy kapcsolóra lép, akkor a szokásos adminisztráción kívül, ki kell nyitni az összes hozzá tartozó lyukat, valamint az ott álló entityket le kell ejteni. Mivel a leejtést az a dolgozó okozza, aki a kapcsolóra tolta a ládát, így a leejtés megfelel annak, mintha épp beletolta volna a lyukba az ott álló entitásokat.

4.4.27 Dolgozó falnak ütközik és életet veszít



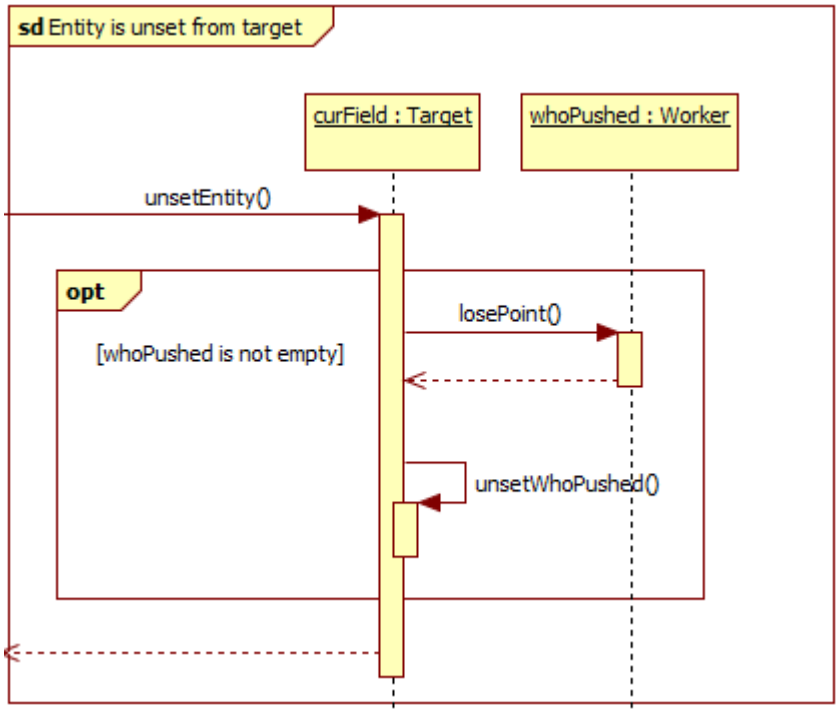
A dolgozó, ha láncban tolják felnyomódik a falra és életet veszít.

4.4.28 Dolgozó kiindulási helyre lép, vagy nekiütközik és életet veszít



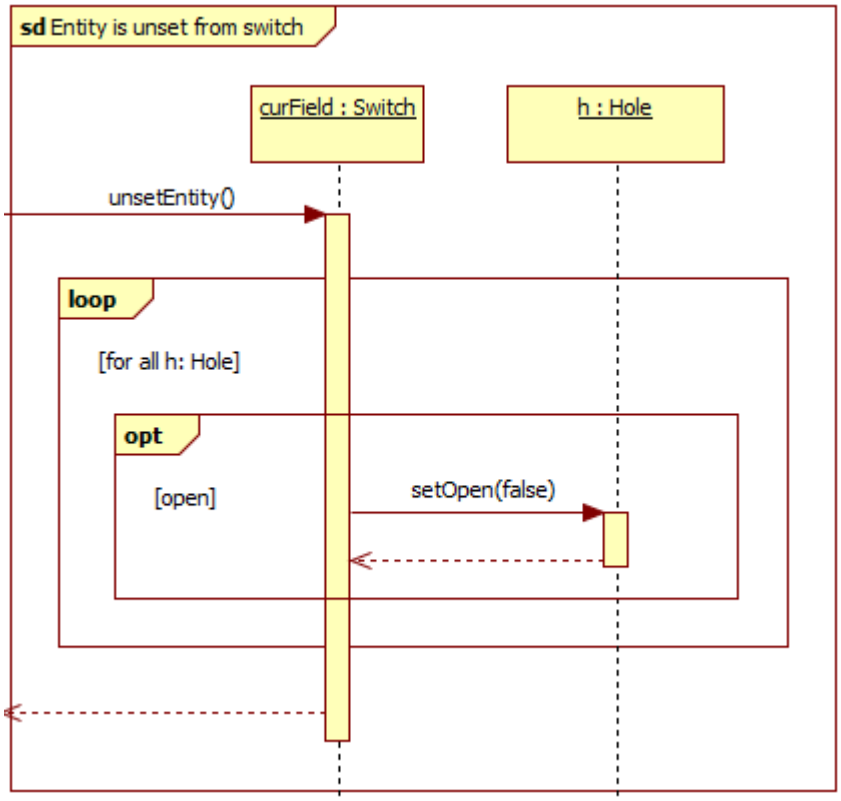
A spawnra csak a tulajdonosa léphet rá, minden más dolgozó csak láncban tolva kerülhet ide, amiért életet veszít.

4.4.29 Entitás lekerül egy előírt helyről



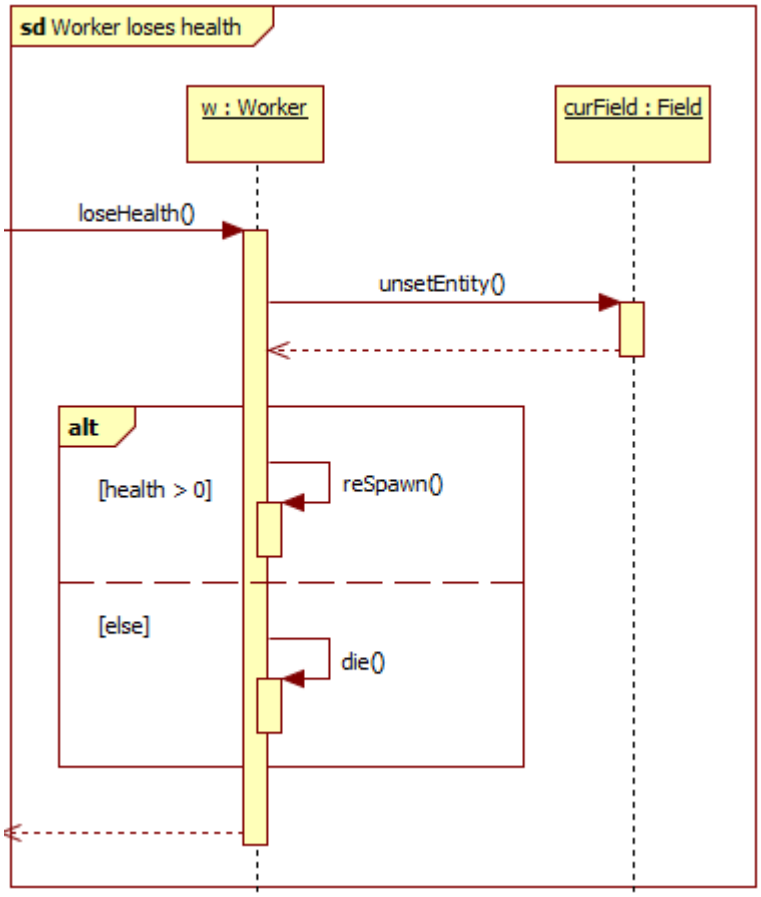
A whoPushed referencia csak akkor van beállítva, ha valaki kapott pontot, mert ládát tolt ide, ezért ebben az esetben, el kell venni tőle ezt a pontot és kinullázni a referenciát.

4.4.30 Entitás lekerül egy kapcsolóról



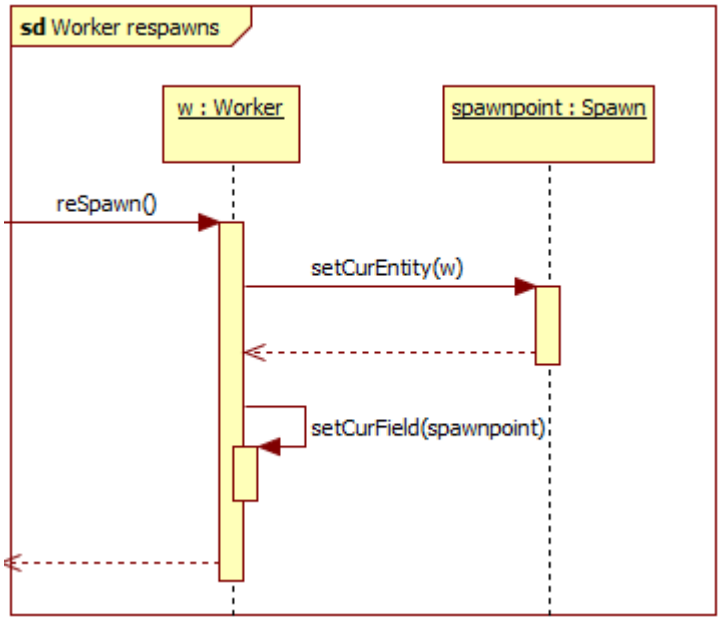
Ha üressé válik a kapcsoló mező, akkor minden hozzá tartozó lyukat be kell zárnunk.

4.4.31 Dolgozó életet veszít



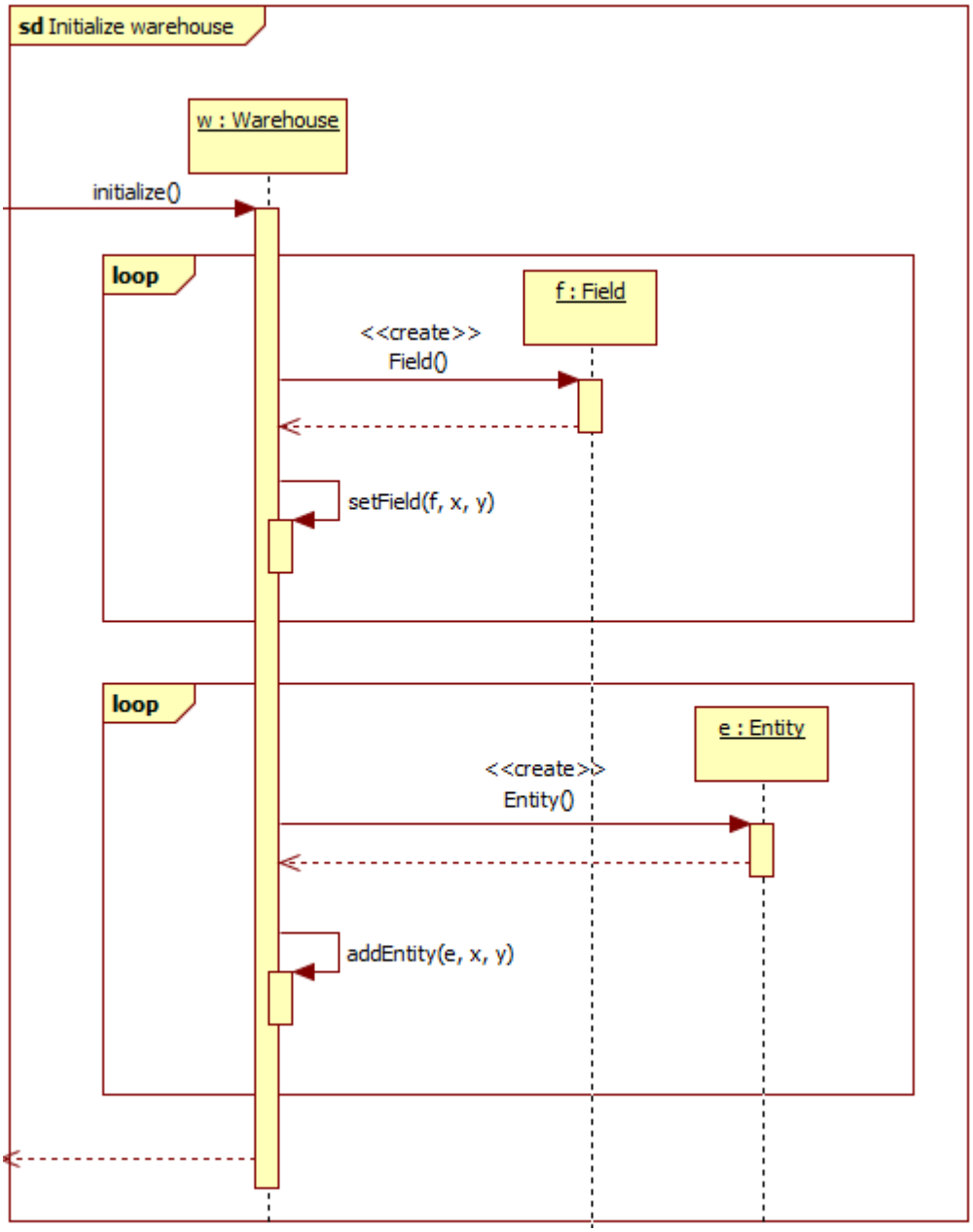
Ha a dolgozó életet veszít akkor le kell venni a mezőről, ahol állt, csökkenteni az életeinek számát, majd az élet mennyiségének megfelelően vagy újrateremteni, vagy megölni.

4.4.32 Dolgozó respawn-ol



Respawn esetén a dolgozót elhelyezzük a spanpointjára, ezt a dolgozó mező és a mező entitás referenciájának állításával tehetjük meg.

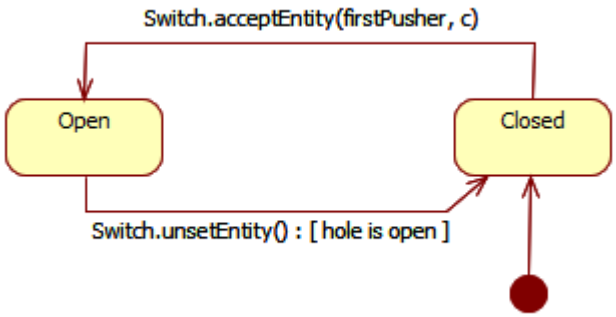
4.4.33 Raktárépület inicializálása



Ekkor jön létre kell hozni a pályát, ami az egyes mezőkből épül fel, majd el kell helyezni ezen a pályán a különböző entitásokat.

4.5 State-chartok

4.5.1 Lyuk állapota változik



Amikor láda lép a lyukhoz rendelt kapcsolóra, a lyuk nyitott állapotba kerül. Amint a láda lekerül a kapcsolóról (vagyis egy entitás lelép a kapcsolóról, és a lyuk korábban nyitva volt), a lyuk zárt állapotba kerül.

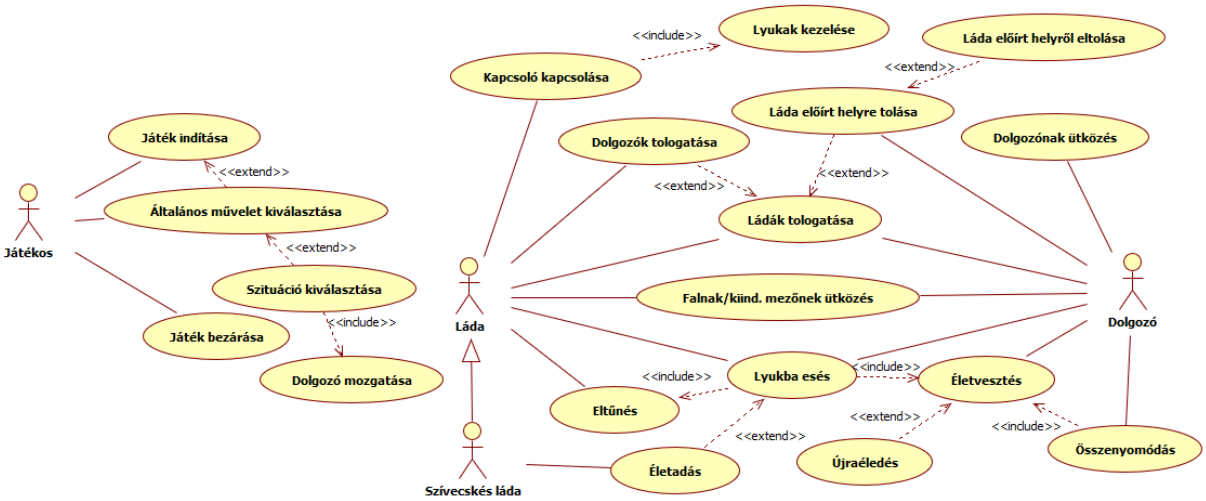
4.6 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.02.28 13:00	1 óra	LAKATOS LENKEFI	Értekezlet, a módosított analízis modell alapjának megtervezése.
2018.03.02. 12.00	2 óra	LAKATOS LENKEFI JANI SZAKÁLLAS CSANÁDY	Értekezlet, melynek döntései: LAKATOS és SZAKÁLLAS módosítja a szekvencia diagramokat. JANI módosítja az osztálydiagramot és ellenőrzi a konzisztenciát. CSANÁDY módosítja az osztályok leírását. LENKEFI ellenőrzi a teljes dokumentumot, esetleges korrekciókat végez.
2018.03.01. 10:00	2,5 óra	CSANÁDY	Osztályok leírásnak módosítása (3.3)
2018.03.01. 14:00	1,5 óra	CSANÁDY	Osztályok leírásnak módosítása (3.3)
2018.03.02. 14.00	1 óra	SZAKÁLLAS	Objektum katalógus (3.1) ellenőrzése, javítása
2018.03.02. 20.00	1 óra	JANI	Osztálydiagram javítása (3.2)
2018.03.02. 21.00	3 óra	SZAKÁLLAS	Szekvencia diagramok készítése (3.4)
2018.03.20. 21:00	1 óra	LAKATOS	Szekvencia diagramok módosítása (3.4)
2018.03.03. 16:00	3 óra	JANI	Szekvencia diagrammok leírása (3.4), osztályok leírásának módosítása (3.3)
2018.03.03. 20:00	2 óra	LAKATOS	Szekvencia diagramok (3.3) és osztálydiagram módosítása (3.4)
2018.03.04. 10:00	2 óra	JANI	Szekvencia diagramok leírása (3.4)
2018.03.04. 18:00	3 óra	LENKEFI	Teljes dokumentum ellenőrzése, konzisztencia vizsgálata, apróbb javítások
2018.03.04. 23:00	1 óra	LAKATOS	Dokumentum véglegesítése

5. Szkeleton tervezése

5.1 A szkeleton modell valóságos use-case-ei

5.1.1 Use-case diagram



5.1.2 Use-case leírások

Use-case neve	Falnak vagy kiindulási mezőnek ütközés
Rövid leírás	Az aktor falnak vagy kiindulási mezőnek ütközik.
Aktorok	Dolgozó, Láda
Forgatókönyv	1. A dolgozó falra vagy nem saját kiindulási mezőre próbál lépni, ekkor ez nem sikerül neki, helyben marad. 2. A ládát falnak vagy kiindulási mezőre próbálják tolni, ekkor ez nem sikeres, helyben marad.

Use-case neve	Életvesztés
Rövid leírás	Valamilyen hatás miatt a dolgozó életet veszít.
Aktorok	Dolgozó
Forgatókönyv	1. Ha a dolgozónak nem ez volt az utolsó élete, akkor ezután eggyel kevesebb élete lesz és átkerül a kiindulási mezőjére. 2. Ha a dolgozónak ez volt az utolsó élete, akkor ezután nulla élete lesz, lekerül a pályáról és ebben a játékban már nem vesz többé részt.

Use-case neve	Dolgozónak ütközés
Rövid leírás	A dolgozó másik dolgozónak ütközik.
Aktorok	Dolgozó
Forgatókönyv	1. A dolgozó közvetlenül olyan mezőre próbál lépni, ahol egy másik dolgozó áll, ekkor a lépés sikertelen és a dolgozó helyben marad.

Use-case neve	Játék indítása
Rövid leírás	A játékos elindítja a játékot.
Aktorok	Játékos
Forgatókönyv	1. A játékos elindítja a játékot.

Use-case neve	Általános művelet kiválasztása
Rövid leírás	A játékos kiválaszt egy általános műveletet.
Aktorok	Játékos
Forgatókönyv	1. A játékos a menüből kiválaszt egy általános műveletet.

Use-case neve	Szituáció kiválasztása
Rövid leírás	A játékos kiválaszt egy szituációt.
Aktorok	Játékos
Forgatókönyv	1. A játékos kiválaszt egy szituációt az általános műveleten belül.

Use-case neve	Dolgozó mozgatása
Rövid leírás	A játékos mozgat egy dolgozót.
Aktorok	Játékos
Forgatókönyv	1. A játékos a kiválasztott szituációban mozgatja a hozzá tartozó dolgozót.

Use-case neve	Játék bezárása
Rövid leírás	A játékos bezárja a játékot.
Aktorok	Játékos
Forgatókönyv	1. A játékos bezárja a játékot.

Use-case neve	Kapcsoló kapcsolása
Rövid leírás	Egy láda mozgásával a kapcsoló állapotot vált (aktív/inaktív).
Aktorok	Láda
Forgatókönyv	1. A kapcsoló mezőre egy láda kerül, így a kapcsoló aktív állapotba kerül. 2. A kapcsoló mezőről lekerül az eddig rajta lévő láda, így a kapcsoló inaktív állapotba kerül.

Use-case neve	Dolgozók tologatása
Rövid leírás	Az aktor egy dolgozót egy szomszédos mezőre tol.
Aktorok	Láda, Dolgozó

Forgatókönyv	1. A dolgozó megfelelő irányban lévő szomszéd mezője üres, ilyenkor egy láda egyszerűen megtolja a dolgozót, aki ennek hatására a szomszédos mezőre kerül. a. A dolgozó megfelelő irányban lévő szomszéd mezője már foglalt, ilyenkor az aktor megtolja a dolgozót, aki ennek hatására megtolja a szomszédos mezőn lévő entitást. 2. A dolgozó megfelelő irányban lévő szomszéd mezője üres, ilyenkor egy közvetve tolt dolgozó egyszerűen megtolja a dolgozót, aki ennek hatására a szomszédos mezőre kerül. a. A dolgozó megfelelő irányban lévő szomszéd mezője már foglalt, ilyenkor az aktor megtolja a dolgozót, aki ennek hatására megtolja a szomszédos mezőn lévő entitást.

Use-case neve	Ládák tologatása
Rövid leírás	Az aktor egy ládát egy szomszédos mezőre tol.
Aktorok	Láda, Dolgozó
Forgatókönyv	1. A láda megfelelő irányban lévő szomszéd mezője üres, ilyenkor az aktor egyszerűen megtolja a ládát, ami ennek hatására a szomszédos mezőre kerül. 2. A láda megfelelő irányban lévő szomszéd mezője már foglalt, ilyenkor az aktor megtolja a ládát, ami ennek hatására megtolja a szomszédos mezőn lévő entitást.

Use-case neve	Lyukba esés
Rövid leírás	Az aktor egy aktív lyuk mezőre kerül.
Aktorok	Láda, Dolgozó
Forgatókönyv	1. A láda egy aktív lyuk mezőre kerül, ezáltal beleesik és eltűnik. 2. A dolgozó egy aktív lyuk mezőre kerül, ezáltal beleesik és elveszít egy életet.

Use-case neve	Eltűnés
Rövid leírás	A Láda eltűnik aktív lyuk mezőre kerülve.
Aktorok	Láda
Forgatókönyv	1. A láda a lyukba esik, így lekerül a játéktérről.

Use-case neve	Életadás
Rövid leírás	A dolgozó életet kap.
Aktorok	Szívecskés Láda

Forgatókönyv	1. A dolgozó egy szívecskés ládát aktív lyuk mezőre tol, ezáltal kap egy életet. 2. A dolgozó egy láncban lévő szívecskés ládát aktív lyuk mezőre tol, ezáltal kap egy életet. 3. A dolgozó aktivál egy szívecskés láda alatt lévő lyukat, aminek hatására a láda leesik és a dolgozó kap egy életet.

Use-case neve	Lyukak kezelése
Rövid leírás	Lyuk aktiválása illetve deaktiválása.
Aktorok	Láda
Forgatókönyv	1. Egy láda egy kapcsolóra kerül, aminek hatására a kapcsolóhoz tartozó lyukak aktívvá válnak. 2. Egy láda lekerül egy kapcsolóról, aminek hatására a kapcsolóhoz tartozó lyukak inaktívvá válnak.

Use-case neve	Újraéledés
Rövid leírás	A dolgozó életvesztést követően újraéled.
Aktorok	Dolgozó
Forgatókönyv	1. A dolgozó életet veszít, és ha nem az volt az utolsó élete, újraéled és a saját kiindulási mezőjére kerül.

Use-case neve	Összenyomódás
Rövid leírás	A dolgozót összenyomják, aminek hatására életet veszít.
Aktorok	Dolgozó
Forgatókönyv	1. A dolgozót falnak tolják, aki ezáltal összenyomódik és életet veszít. 2. A dolgozót nem saját kiindulási mezőre tolják, aki ezáltal összenyomódik és életet veszít.

Use-case neve	Láda előírt helyre tolása
Rövid leírás	A ládát egy előírt helyre tolják.
Aktorok	Dolgozó
Forgatókönyv	1. A ládát egy dolgozó közvetlenül vagy láncban egy előírt helyre tol, ezáltal pontot szerez.

Use-case neve	Láda előírt helyről eltolása
Rövid leírás	A ládát eltolják egy előírt helyről.
Aktorok	Dolgozó

Forgatókönyv	1. A ládát egy dolgozó közvetlenül vagy láncban eltol egy előírt helyről, ezáltal a rátolásért pontot kapó játékos pontot veszít.

5.2 A szkeleton kezelői felületének terve, dialógusok

A szkeleton szoftver alapvetően egy konzolos menürendszer, melyben a fontosabb szenáriók játszhatóak végig. A menüelem általában meghatároz egy általánosabb műveletet (például: Dolgozó mozgatása), ezen belül konkrétabb szituációk választhatók ki. Például:

```
What do you want to do?
 1 - Move Worker
 2 - Push Crate
 3 - Create Own Sequence (Dev. Mode)
```

Ha például az első lehetőséget választjuk ki:

```
More specifically?
  1.1 - Move Worker On Empty Floor
  1.2 - Move Worker On Empty Hole
  1.3 - Move Worker On Empty Target
  1.4 - Move Worker On Empty Switch
  1.5 - Move Worker On Wall
  1.6 - Move Worker On Spawn
  1.7 - Worker Pushes Worker
  1.8 - Worker Pushes Worker In Chain On Wall
  1.9 - Worker Pushes Worker In Chain On Spawn
```

Esetleg ha a másodikat:

```
More specifically?
  2.1 - Worker Pushes Crate On Empty Floor
  2.2 - Worker Pushes Crate On Empty Hole
  2.3 - Worker Pushes LifeCrate On Empty Hole
  2.4 - Worker Pushes Crate On Empty Target
  2.5 - Worker Pushes Crate On Empty Switch
  2.6 - Worker Pushes Crate On Wall
  2.7 - Worker Pushes Crate On Spawn
  2.8 - Worker Pushes Crate From Switch
  2.9 - Worker Pushes Crate From Target
```

Ezek már konkrét szituációk, melyeket kiválasztva láthatjuk a lefutó interakciókat/függvényeket. Például az elsőből az ötödiket kiválasztva megnézhetjük mi történik akkor, ha a munkás megpróbál falra lépni:

```
1.5 Move Worker On Wall
      -> Worker::move(dir):
         -> Entity::step(firstPusher, dir):
           -> Field::getNeighbourField(dir):
           neighbor <- Field::getNeighbourField(dir)
           -> Field::isEmpty():
           true <- Field::isEmpty()
           -> Worker::visit(firstPusher, iv):
                 -> Field::visitByWorker(firstPusher, w):
                false <- Field::visitByWorker(firstPusher, w)
           false <- Worker::visit(firstPusher, iv)
      false <- Entity::step(firstPusher, dir)
      <- Worker::move(dir)
```

A szoftvernek van egy ún. developer módja, amikor saját szituációt gyárthatunk. Ekkor minden egyes dolgot magunknak kell kiválasztani. Például azon szituáció legyártása, mikor egy játékos egy másikat próbál közvetlen eltolni (amit nálunk nem lehet):

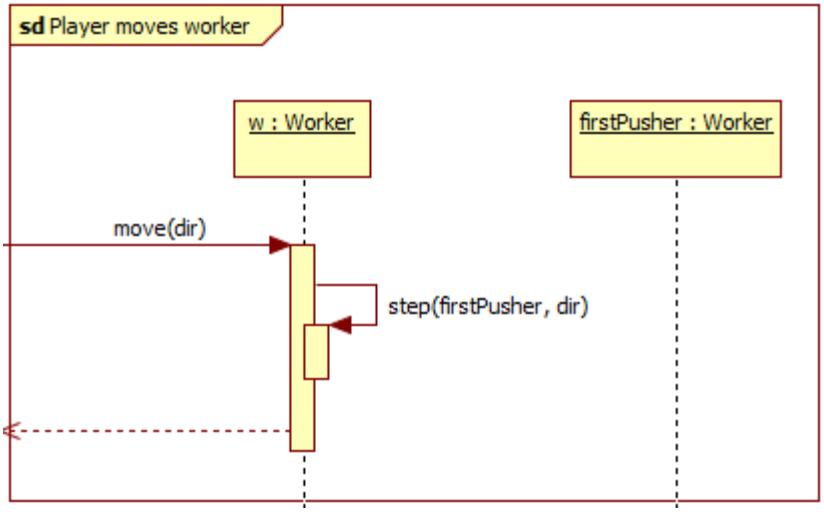
```
What direction to move? [L - Left, R - Right, U - Up, D - Down] : R
      -> Worker::move(dir):
      -> Entity::step(firstPusher, dir):
            -> Field::getNeighbourField(dir):
                  What kind of field is the neighbour? [F - Floor, W - Wall,
S - Spawn, T - Target, H - Hole] : F
            neighbor <- Field::getNeighbourField(dir)
            -> Field::isEmpty():
                  Is an entity already on this field? [Y - Yes, N - No] : Y
                  What kind of entity? [W - Worker, C - Crate, L - LifeCrate]
: W
            false <- Field::isEmpty()
              -> Worker::push(firstPusher, pushed, dir):
                  -> Worker::pushByWorker(firstPusher, pusher, dir):
                  false <- Worker::pushByWorker(firstPusher, pusher, dir)
            false <- Worker::push(firstPusher, pushed, dir)
        false <- Entity::step(firstPusher, dir)
      <- Worker::move(dir)
```

Ekkor a lefutás végén megkapjuk a lefutás közben beírt betű-szekvenciát (jelen esetben ez: RFYW). Az automatikus verzió lényegében önállóan adja a programnak ezt a bemenetet, így ezt kimásolva könnyen adhatunk hozzá új szituációt, lényegében egy új menüelemhez rendeljük hozzá a betűsorozatot.

5.3 Szekvencia diagramok a belső működésre

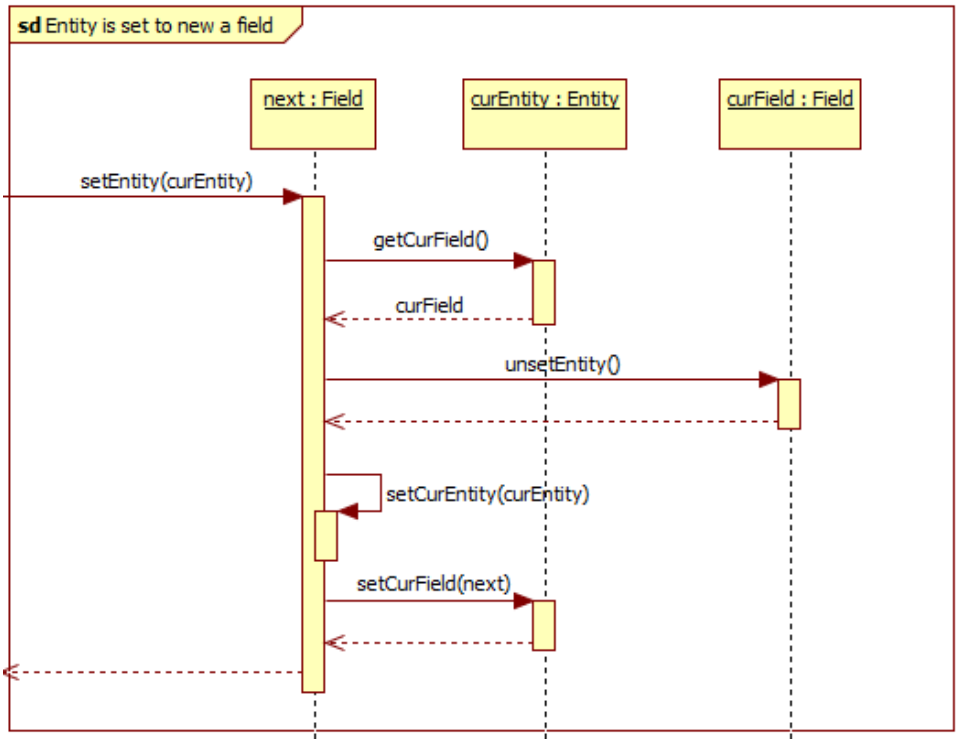
Megjegyzés: Mivel az UML eszköz nem támogatja a szabványos create jelölést, így az metódushívásként van modellezve.

5.3.1 A játékos lépteti a dolgozót



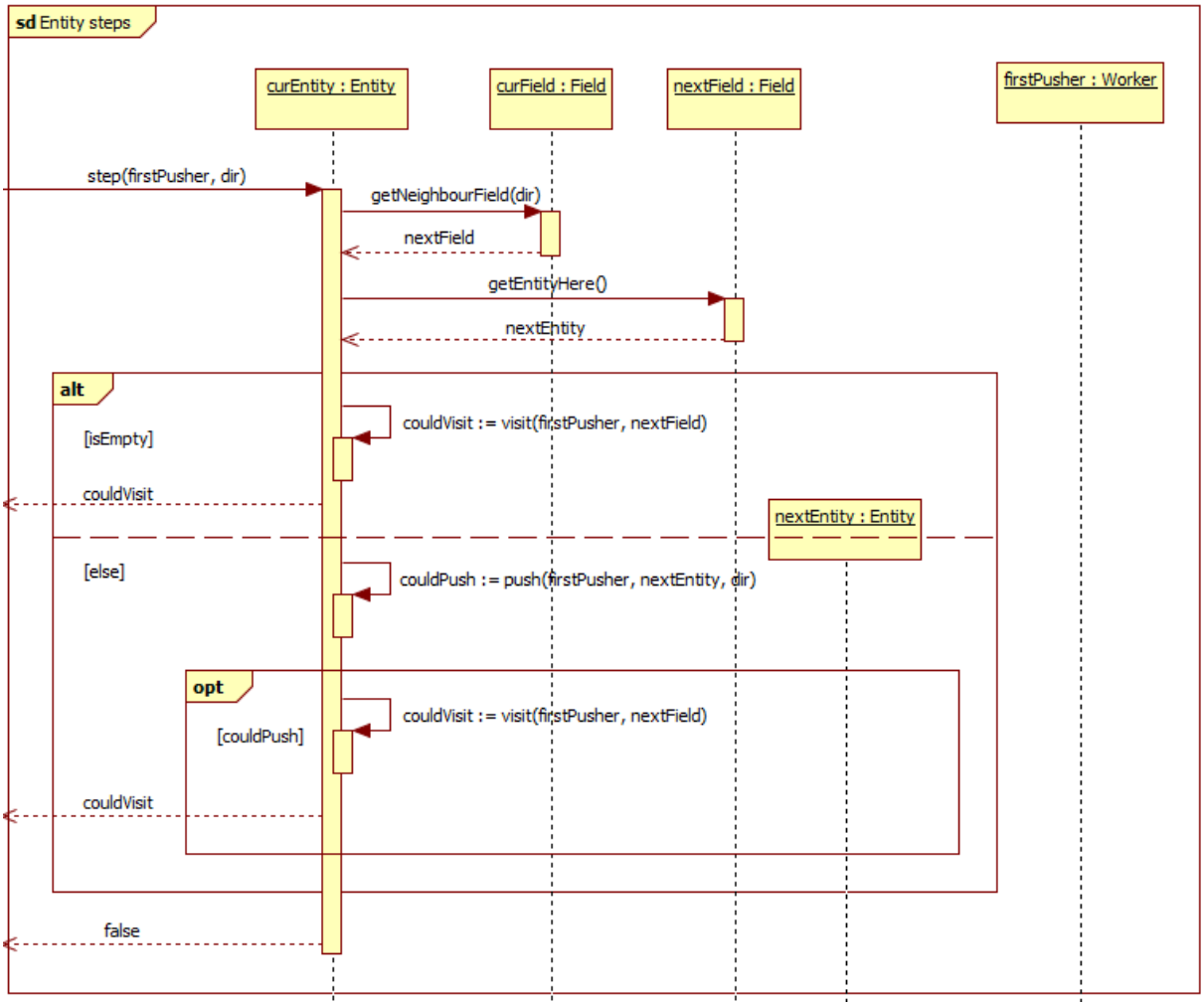
Ilyenkor megpróbálunk ellépni az adott irányban.

5.3.2 Entitás új mezőre lép



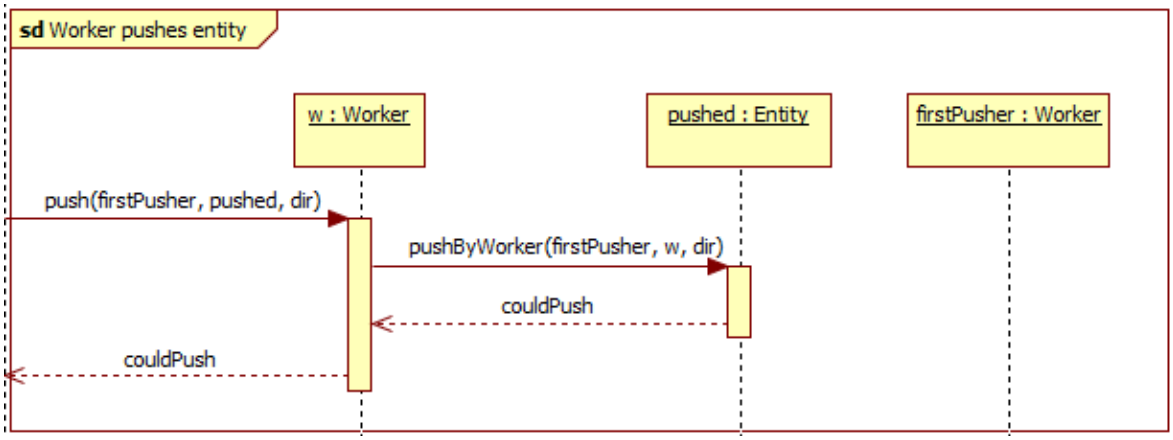
A belépéskor adminisztráljuk mind az entitás mezőre vonatkozó, mind a mező entitásra vonatkozó referenciájának megváltozását.

5.3.3 Entitás lép



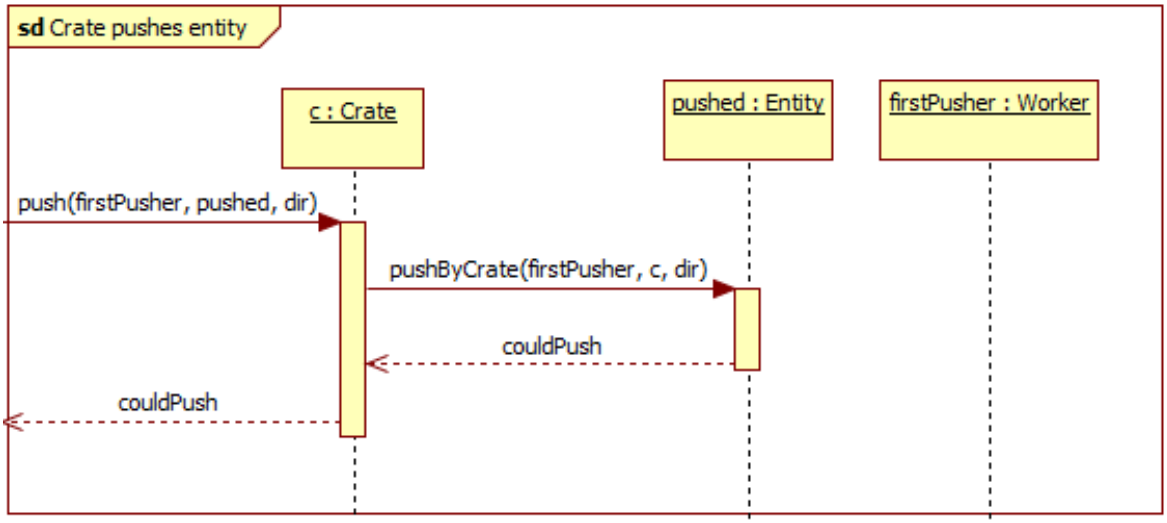
Léptetéskor két dolgot kell ellenőriznünk. Elsőnek, hogy a célmező egyáltalán képes-e fogadni minket (pl.: Spawn csak a saját játékosát fogadja be), és ha ez fennáll, akkor azt, hogy jelenleg szabad-e, ugyanis, ha másik entitás már ott tartózkodik, akkor meg kell próbálnunk eltolni. Ha sikerült a tolás, vagy eleve üres volt, akkor odalépünk, egyébként helyben maradunk.

5.3.4 Dolgozó entitást tol



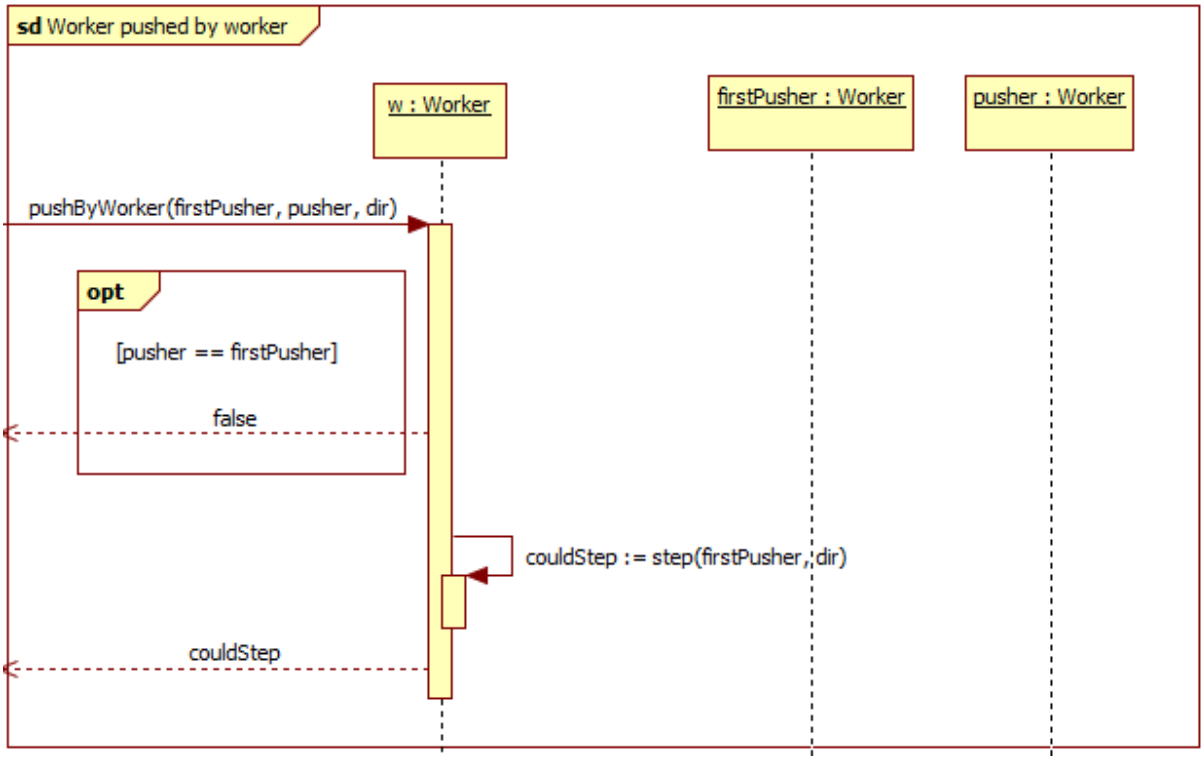
Mini "Visitor" -> Dinamikussal típussal való hívás miatt szükséges függvény.

5.3.5 Láda entitást tol



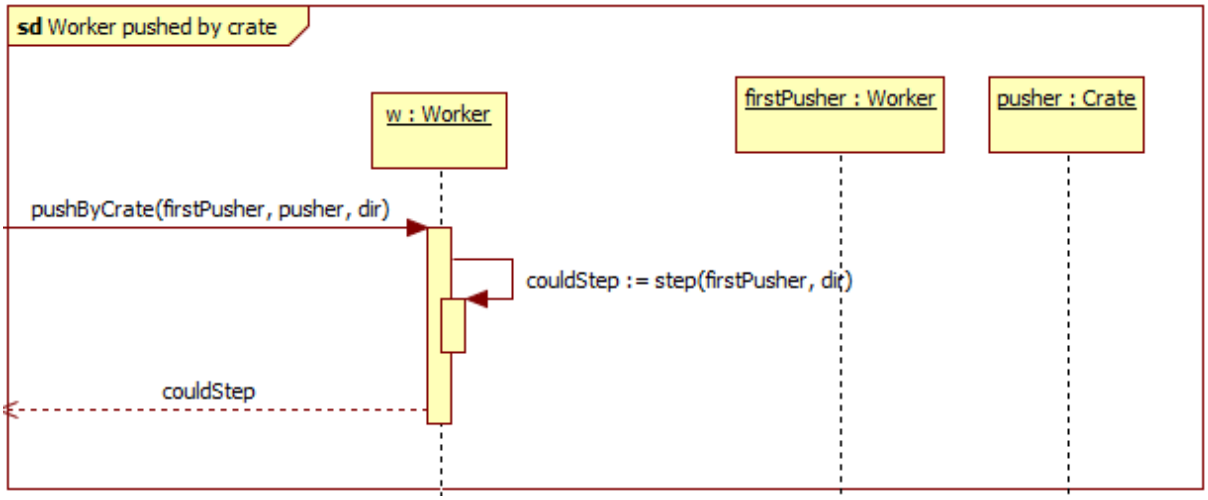
Mini "Visitor" -> Dinamikussal típussal való hívás miatt szükséges függvény.

5.3.6 Dolgozót dolgozó tol



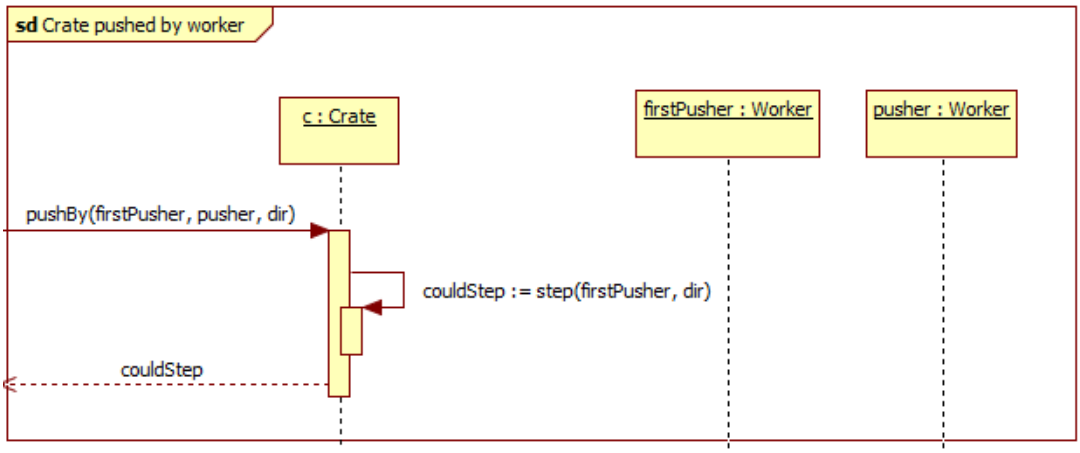
Dolgozó dolgozót csak akkor tolhat, ha láncba van, ezért ellenőrizzük, hogy aki itt tol az nem az első-e. Ha az első akkor, vissza térünk hammissal, ezzel megtagadva a tolást, ha nem akkor megpróbálunk ellépni a megfelelő irányba és ennek sikerességével térünk vissza.

5.3.7 Dolgozót láda tol



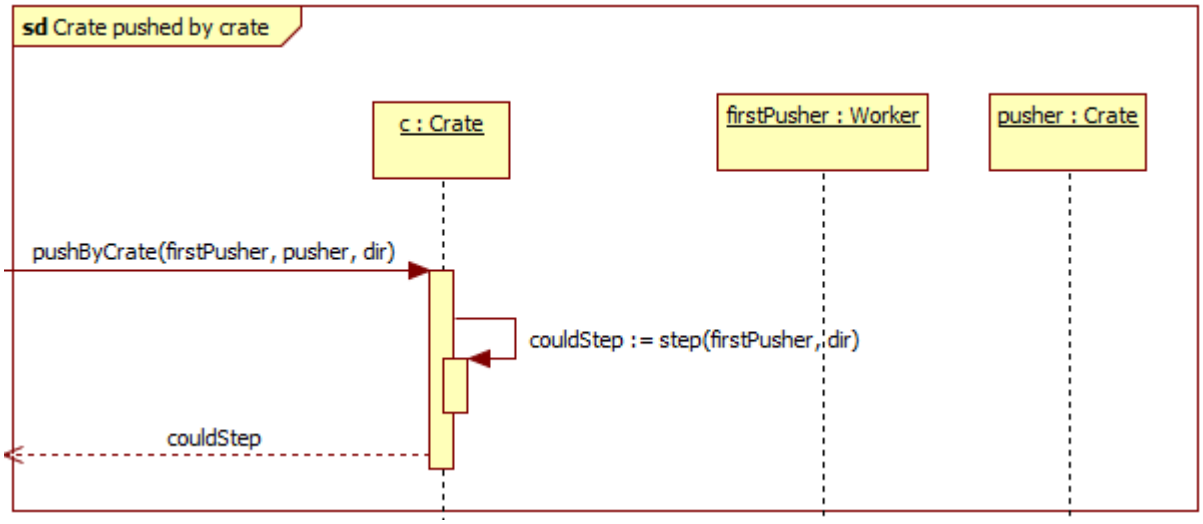
Mivel láda mindig képes dolgozót tolni, ezért megpróbálunk ellépni a megfelelő irányba és ennek sikerességével térünk vissza.

5.3.8 Ládát dolgozó tol



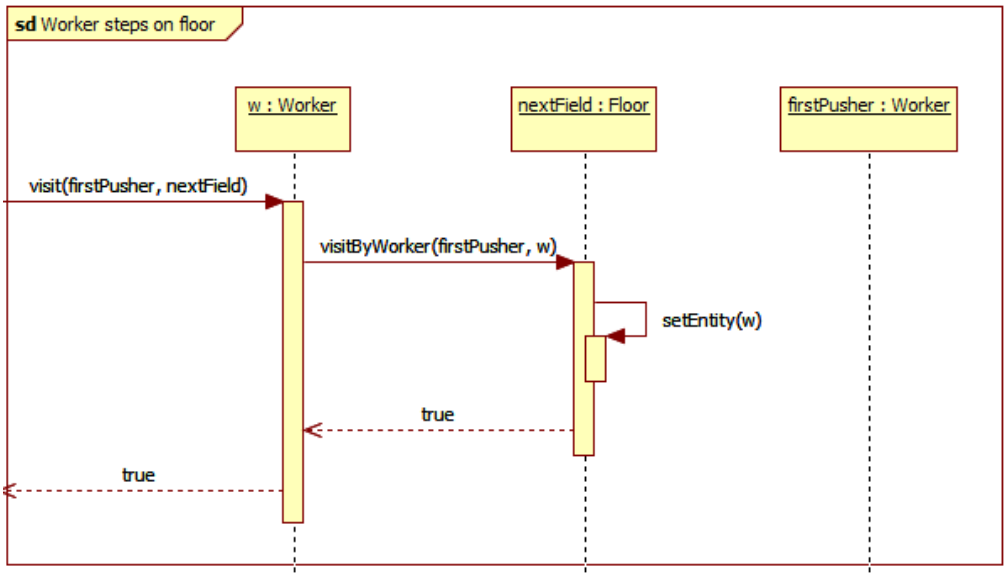
Mivel dolgozó mindig képes ládát tolni, ezért megpróbálunk ellépni a megfelelő irányba és ennek sikerességével térünk vissza.

5.3.9 Ládát láda tol



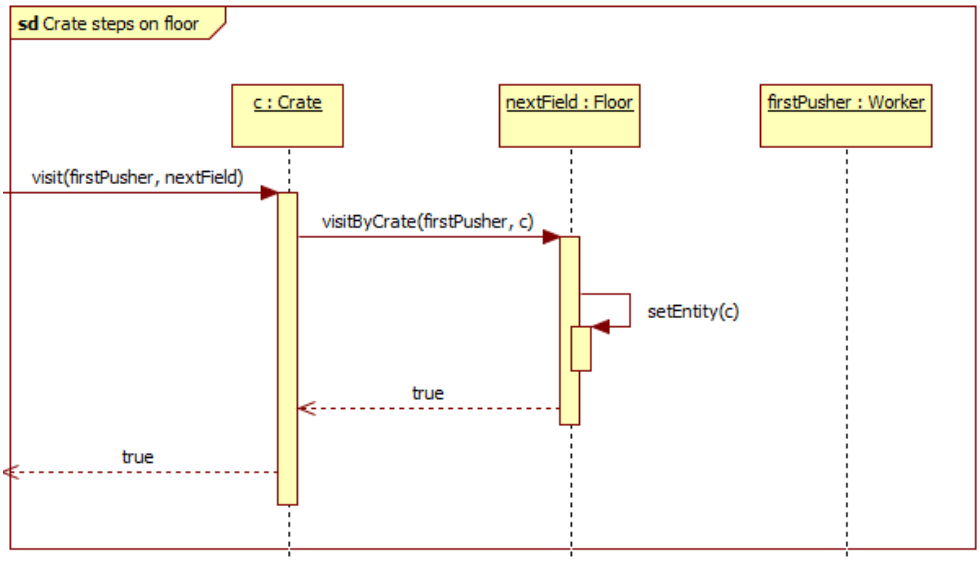
Mivel láda mindig képes ládát tolni, ezért megpróbálunk ellépni a megfelelő irányba és ennek sikerességével térünk vissza.

5.3.10 Dolgozó padlóra lép



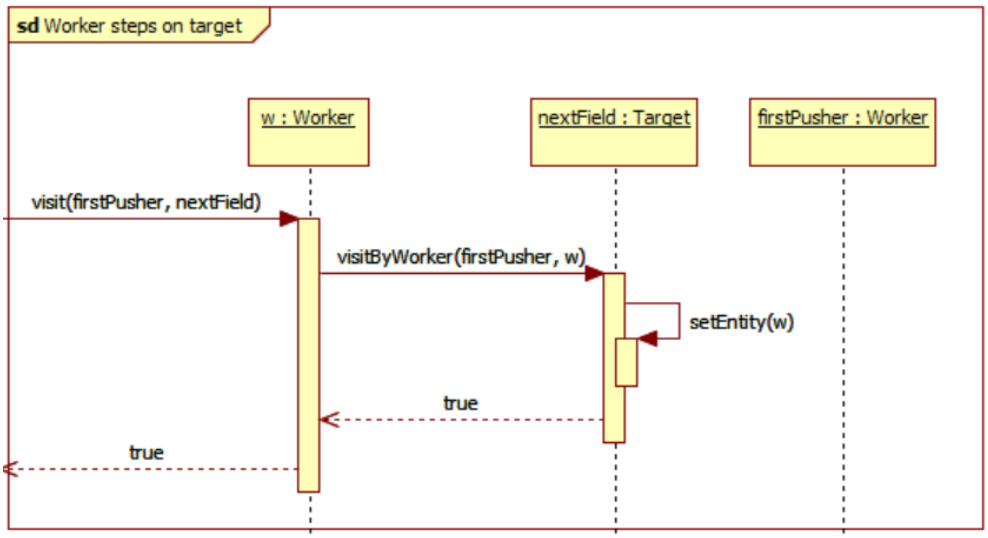
Egy dolgozó mindig ráléphet egy padlóra.

5.3.11 Láda padlóra lép



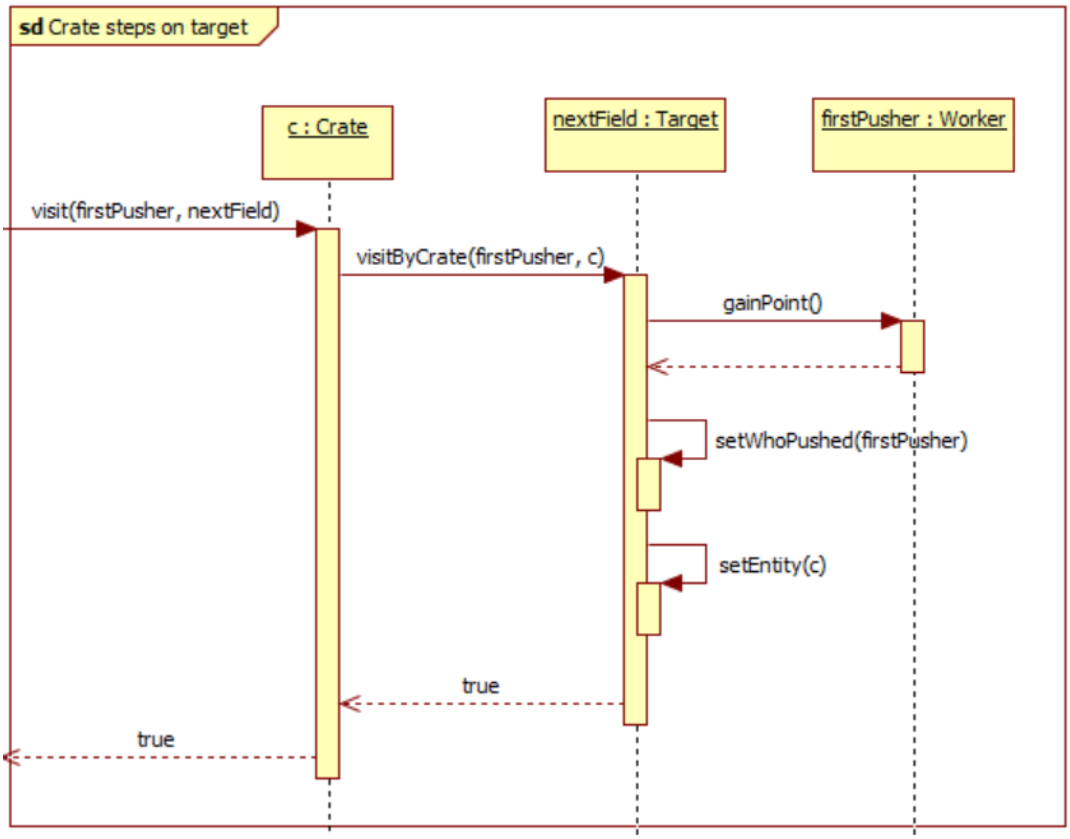
Egy láda mindig ráléphet egy padlóra.

5.3.12 Dolgozó előírt helyre lép



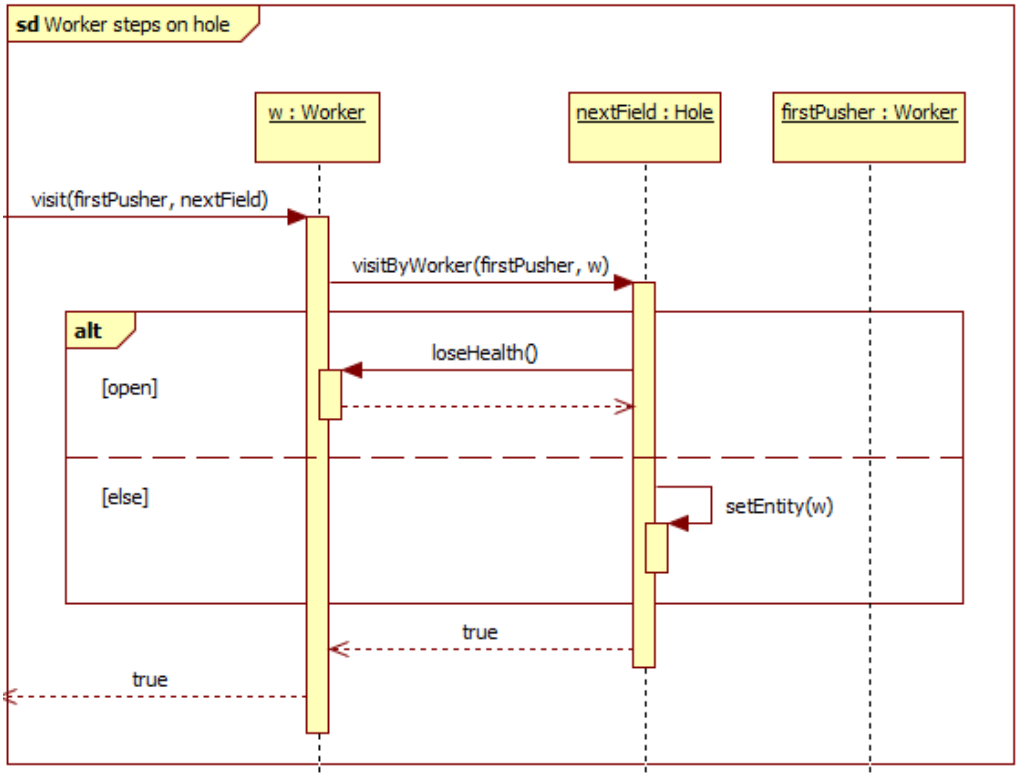
Egy dolgozó mindig ráléphet egy előírt helyre.

5.3.13 Láda előírt helyre lép



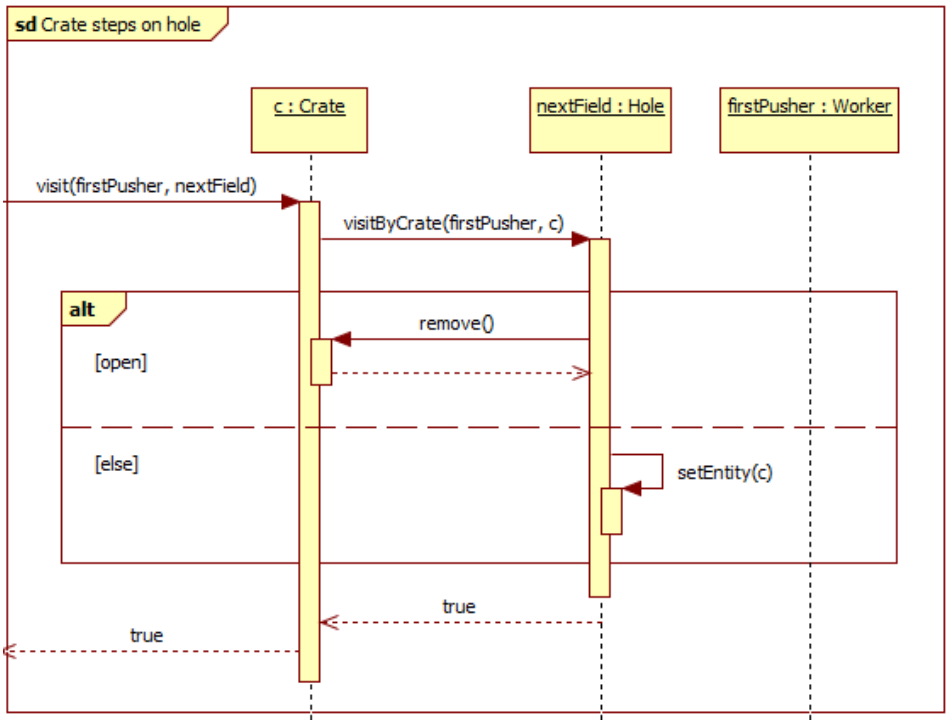
Mikor egy láda kilép innen, a szokásos adminisztráció során elvesszük a kiosztott pontot. Tehát beléptetéskor szokásos adminisztráción kívül, két dolgot kell tennünk, kiosztani a pontot annak aki miatt idekerült a láda, valamint referenciáját eltárolni, hogy később, ha kell, eltudjuk venni a pontot.

5.3.14 Dolgozó lyukra lép



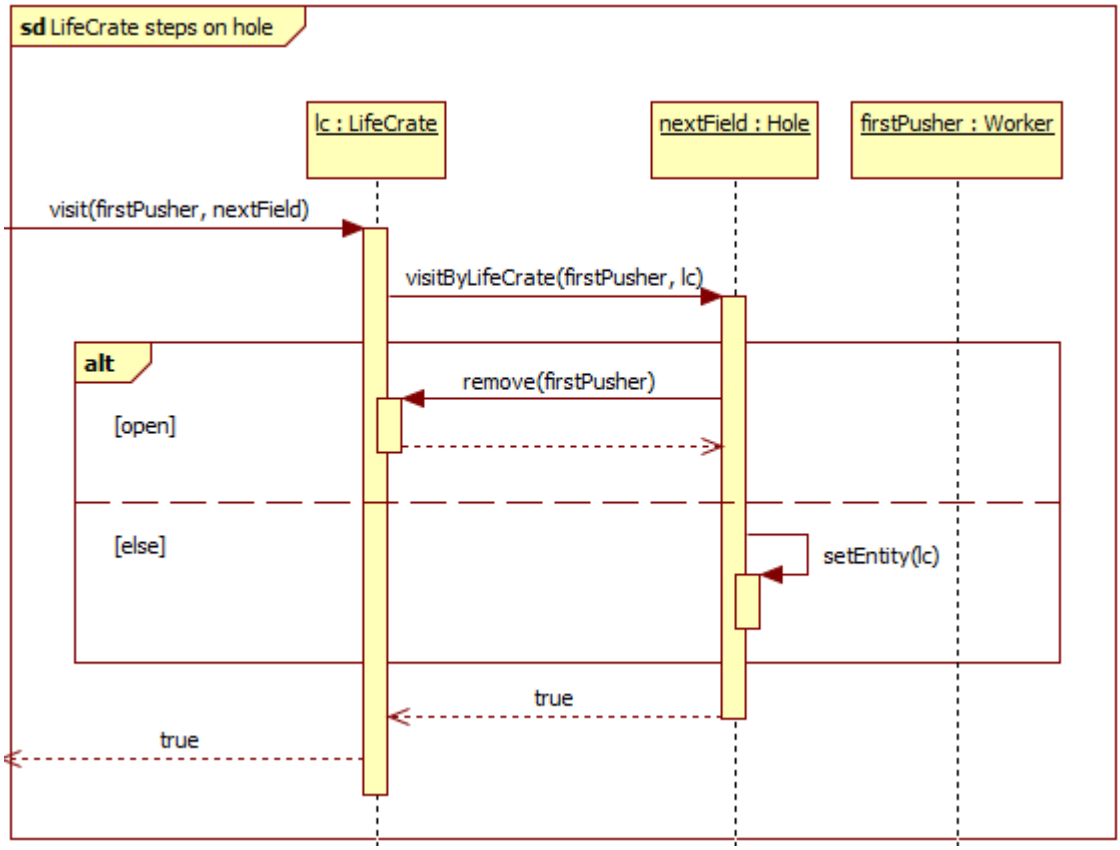
Ekkor ha nyitva van a lyuk akkor a lépés hatására életet kell veszítenie a dolgozónak, ha zárva akkor a szokásos műveleteket kell elvégezni.

5.3.15 Láda lyukra lép



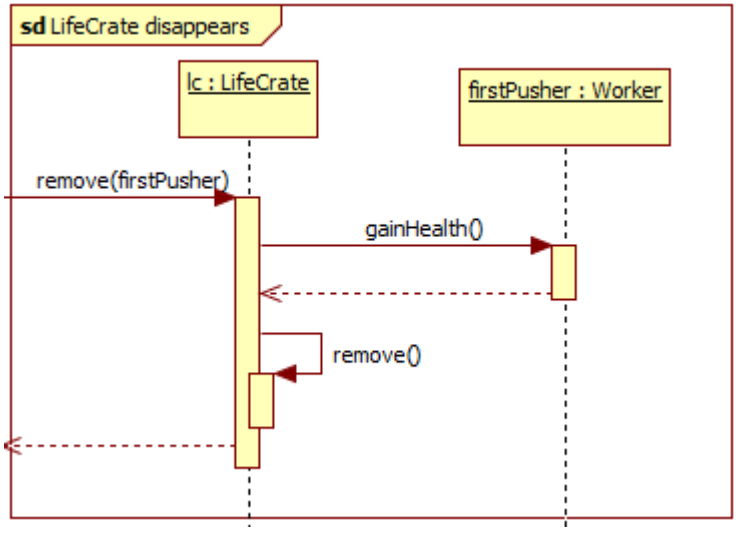
Ekkor ha nyitva van a lyuk akkor a lépés hatására megsemmisül a láda, ha zárva akkor a szokásos műveleteket kell elvégezni.

5.3.16 Szívecskés láda lyukra lép



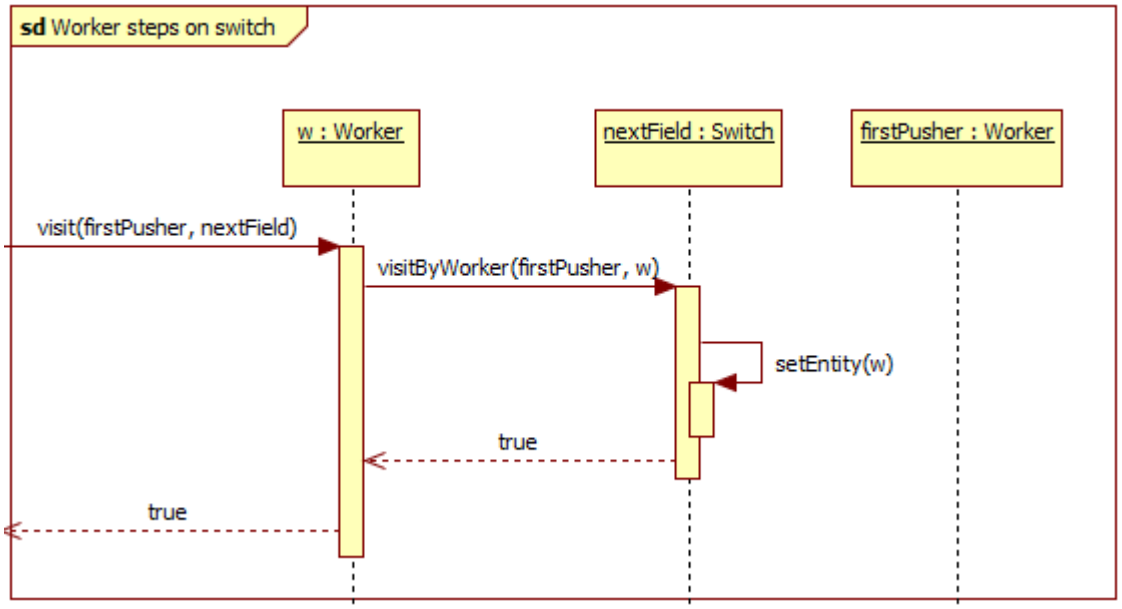
Ekkor ha nyitva van a lyuk akkor a lépés hatására megsemmisül a láda és aki ezt okozta kap egy életet (remove függvényen belül), ha zárva akkor a szokásos műveleteket kell elvégezni.

5.3.17 Szívecskés láda eltűnik



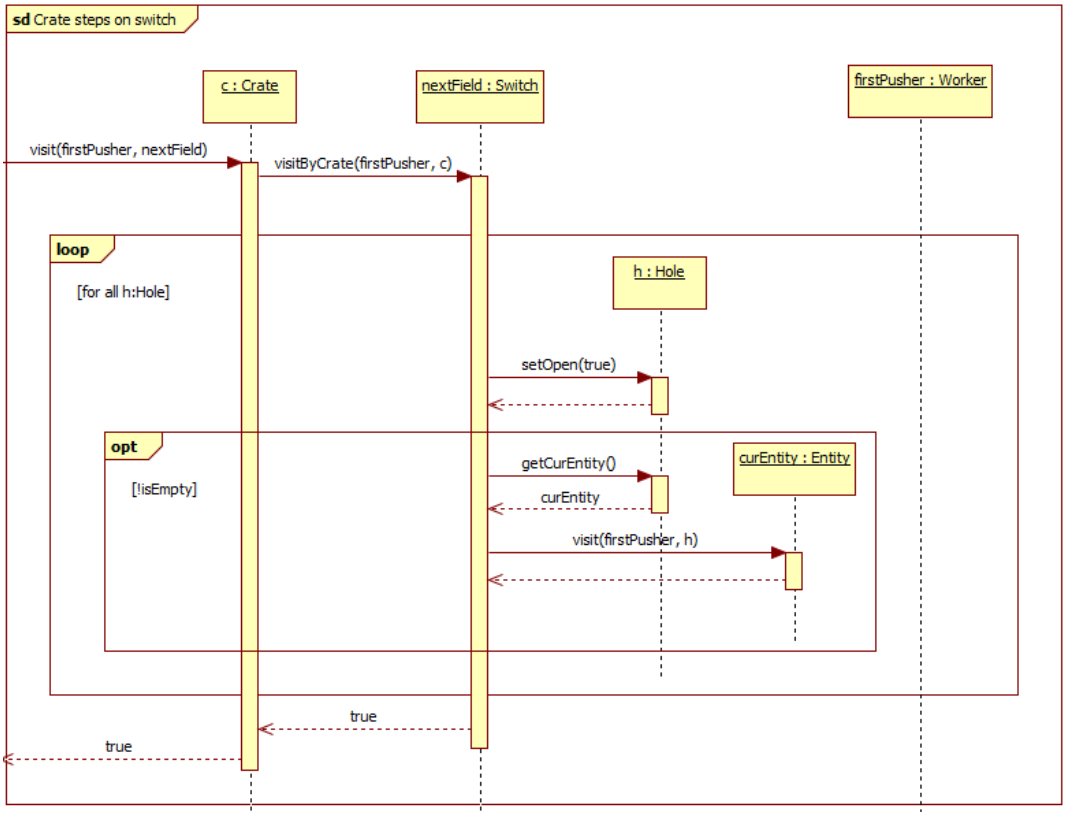
Szívecskés láda esetén a megsemmisüléskor az azt okozó kap egy életet, majd a szokásos módon eltűnik a láda.

5.3.18 Dolgozó kapcsolóra lép



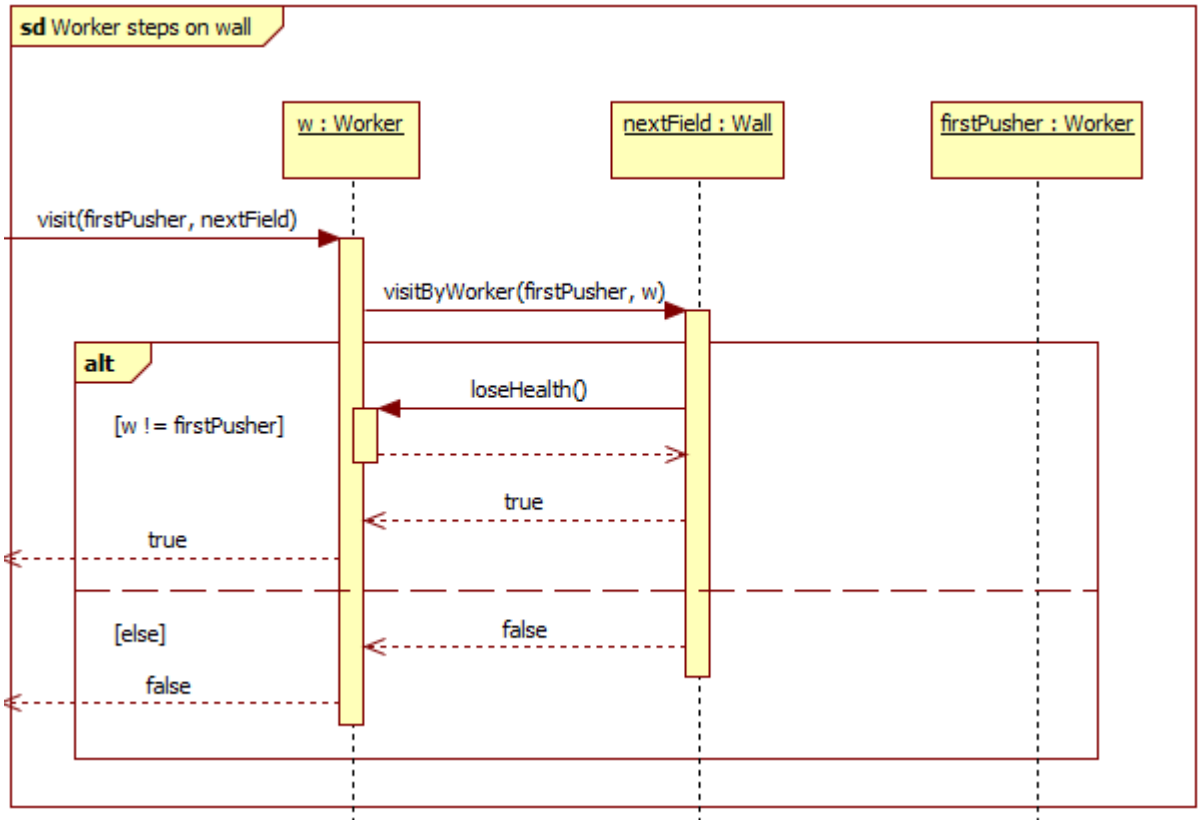
A dolgozó nem képes kapcsolni a kapcsolót, ezért csak a szokásos lépések történnek.

5.3.19 Láda kapcsolóra lép



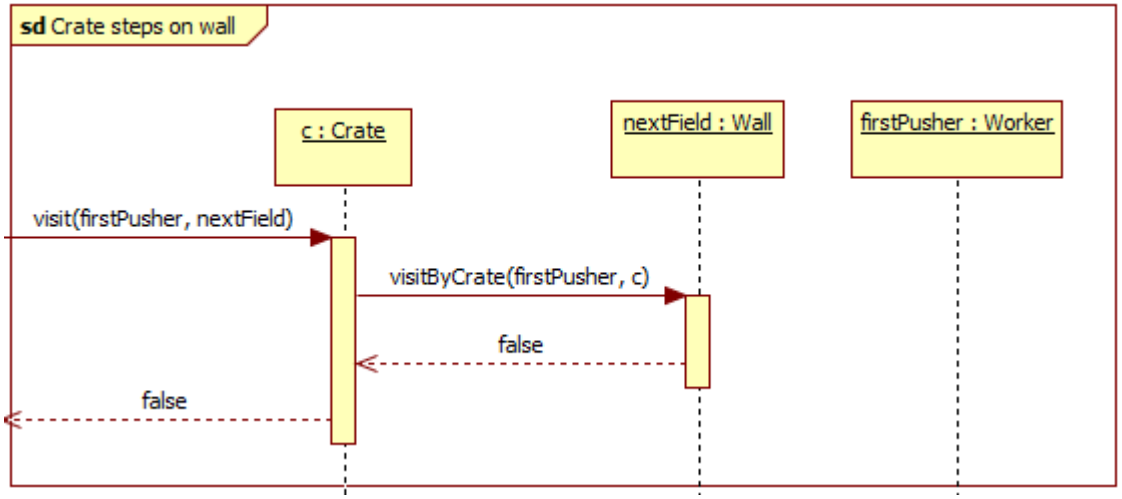
Ha láda egy kapcsolóra lép, akkor a szokásos adminisztráción kívül, ki kell nyitni az összes hozzá tartozó lyukat, valamint az ott álló entityket le kell ejteni. Mivel a leejtést az a dolgozó okozza, aki a kapcsolóra tolta a ládát, így a leejtés megfelel annak, mintha épp beletolta volna a lyukba az ott álló entitásokat.

5.3.20 Dolgozó falra lép



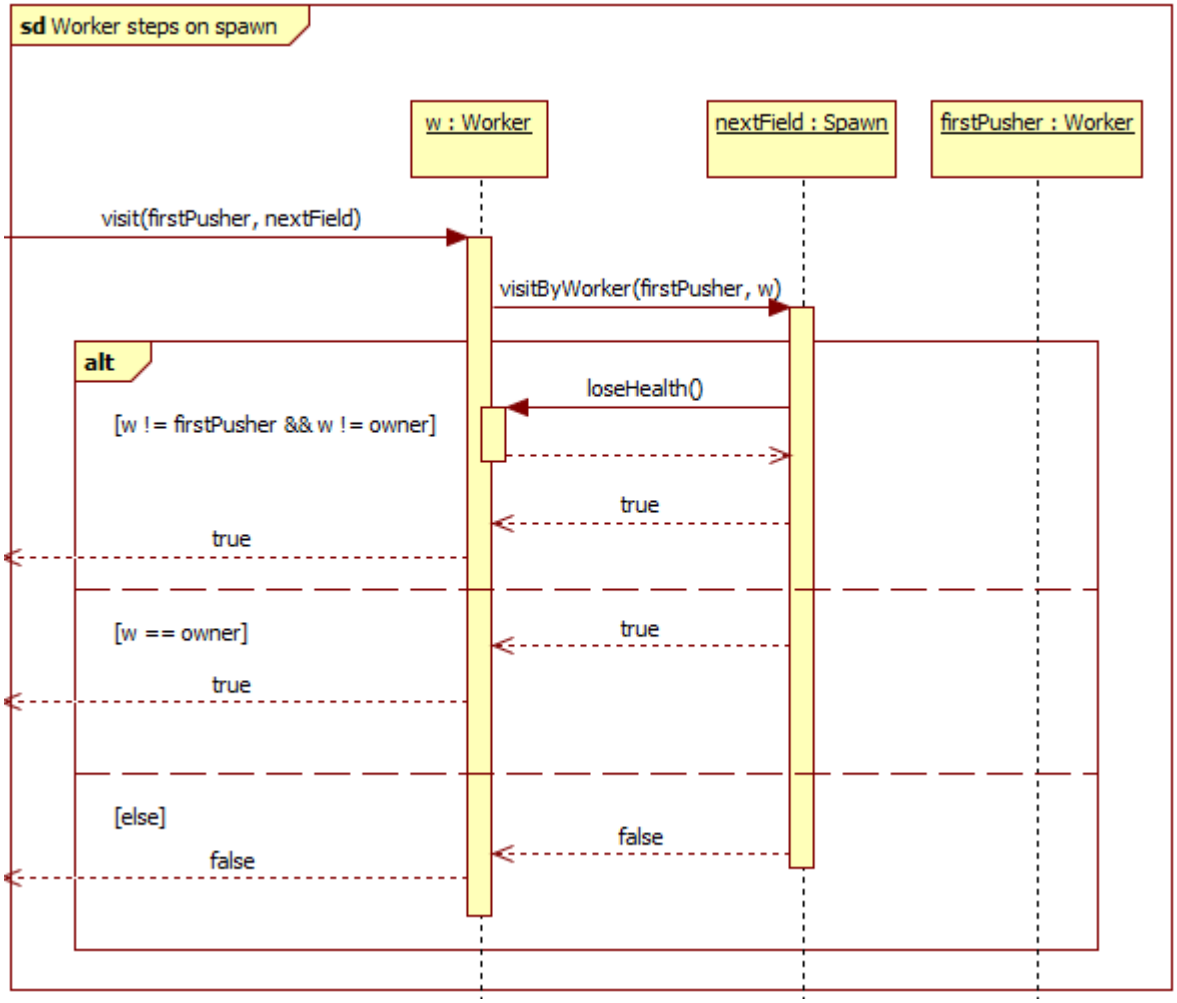
Egy dolgozó csak akkor léphet falra, ha rátolják, ekkor életet veszít.

5.3.21 Láda falra lép



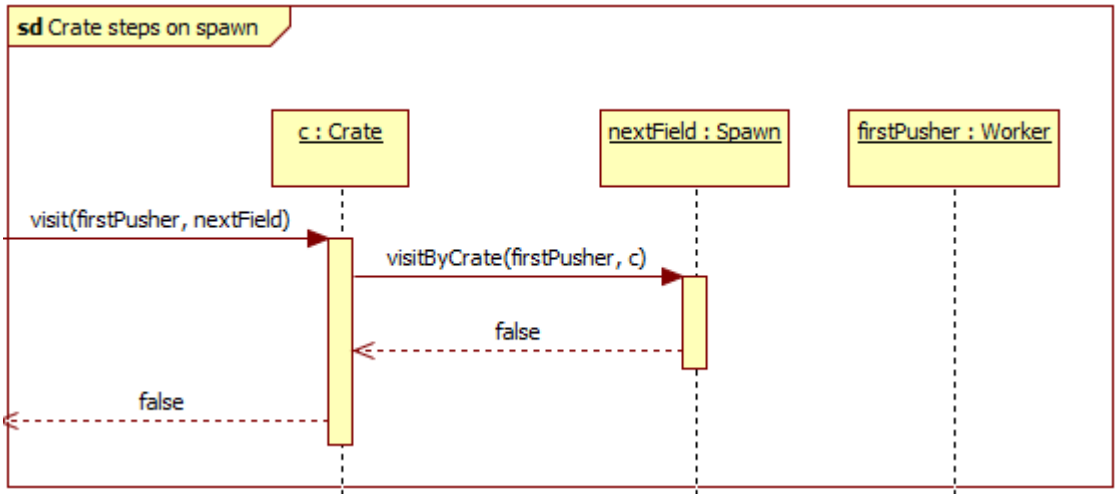
Egy láda soha nem léphet rá egy falra.

5.3.22 Dolgozó kiindulási helyre lép



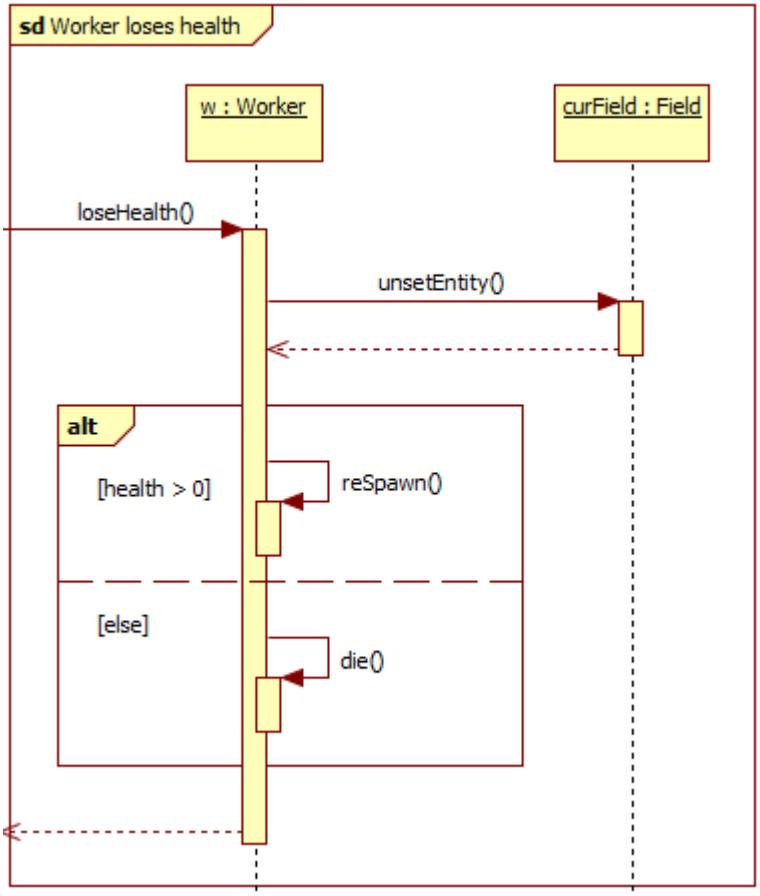
A spawnra csak a tulajdonosa léphet rá, minden más dolgozó életet veszít a probálkozásért.

5.3.23 Láda kiindulási helyre lép



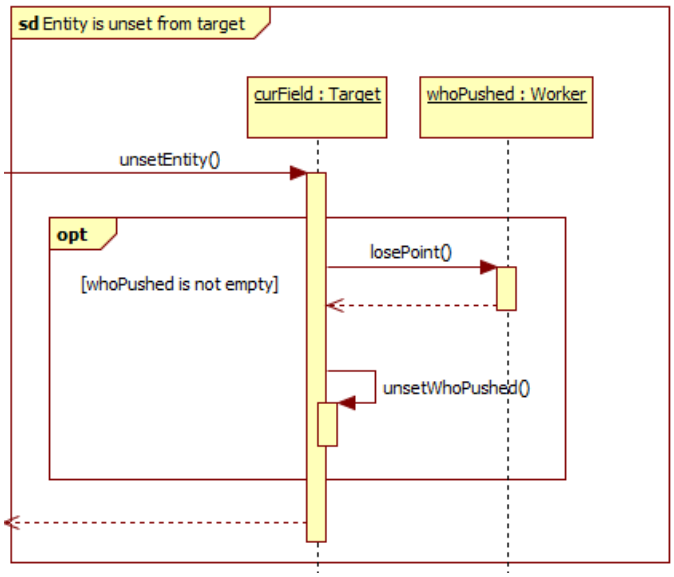
Egy láda soha nem léphet rá egy kiindulási helyre.

5.3.24 Dolgozó életet veszít



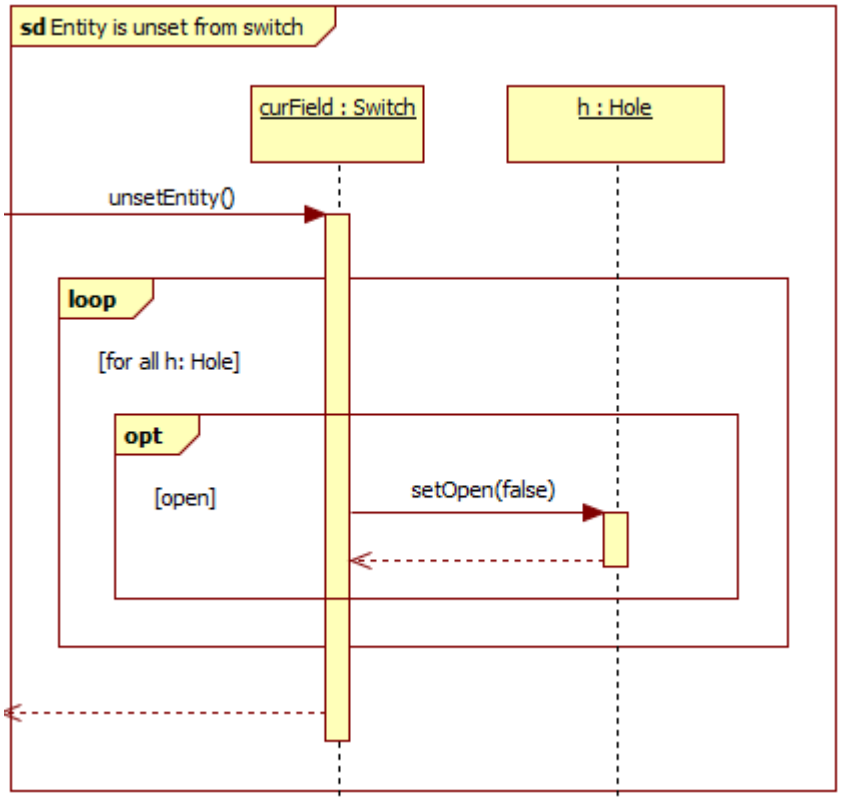
Ha a dolgozó életet veszít akkor le kell venni a mezőről, ahol állt, csökkenteni az életeinek számát, majd az élet mennyiségének megfelelően vagy újrateremteni, vagy megölni.

5.3.25 Entitás lekerül előírt helyről



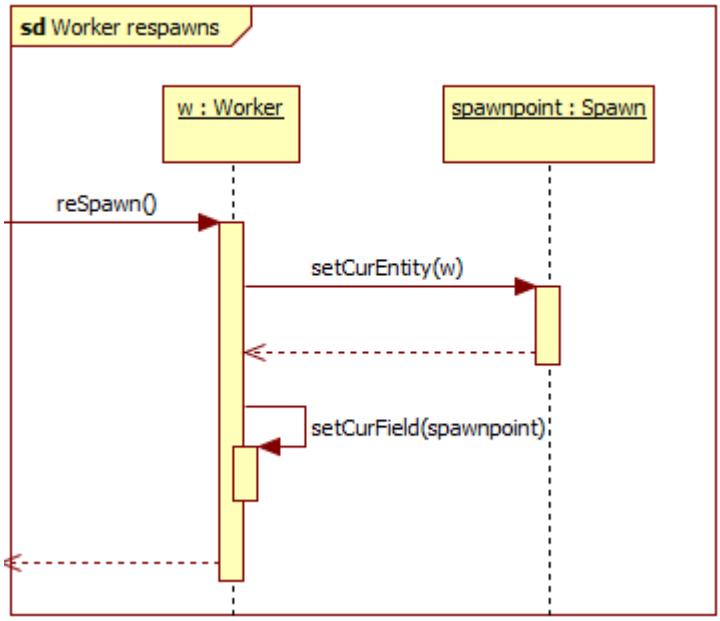
A whoPushed referencia csak akkor van beállítva, ha valaki kapott pontot, mert ládát tolt ide, ezért ebben az esetben, el kell venni tőle ezt a pontot és kinullázni a referenciát.

5.3.26 Entitás lekerül kapcsolóról



Ha üressé válik a kapcsoló mező, akkor minden hozzá tartozó lyukat be kell zárnunk.

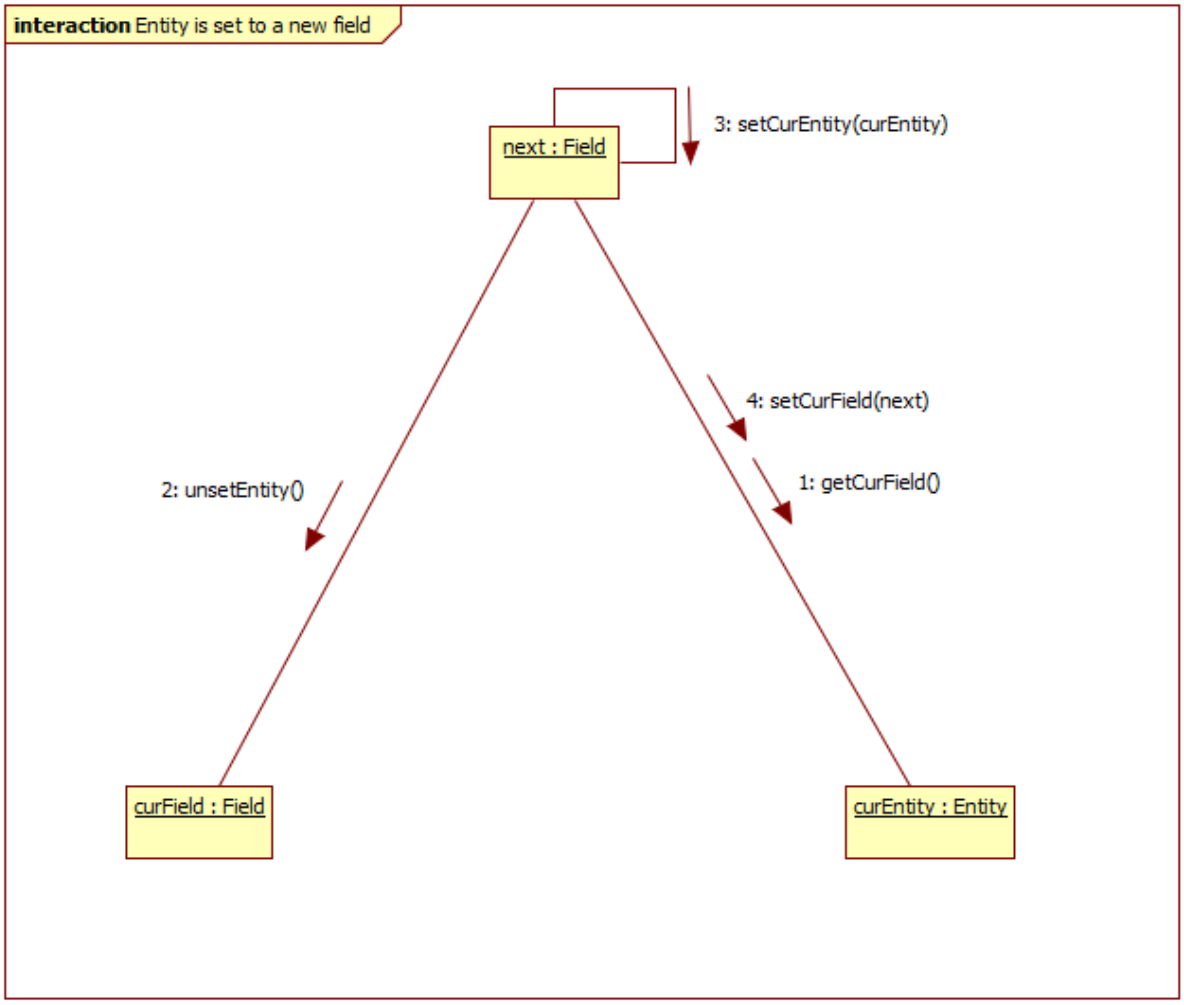
5.3.27 Dolgozó respawnol



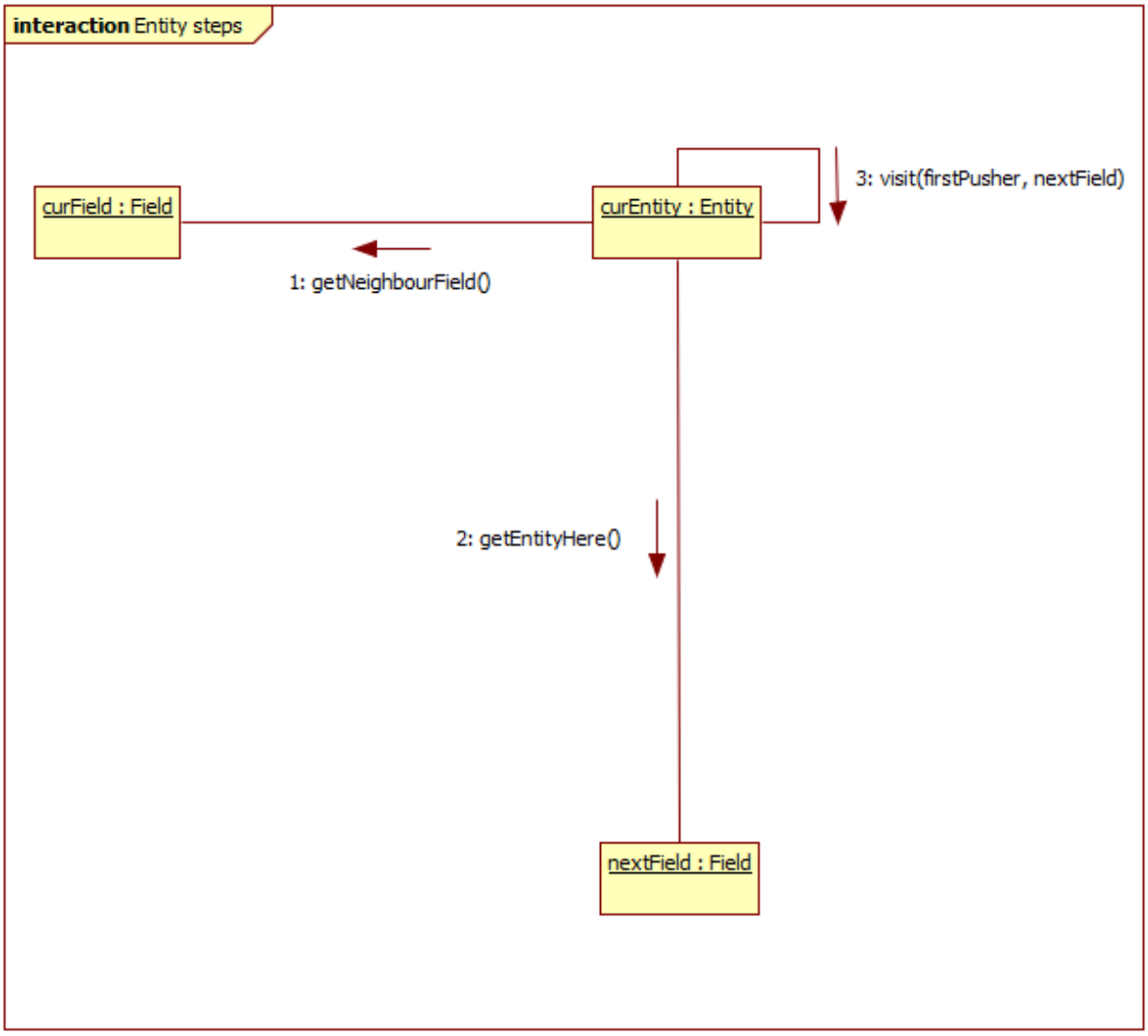
Újrateremtődés esetén a dolgozót elhelyezzük a spanpointjára, ezt a dolgozó mező és a mező entitás referenciájának állításával tehetjük meg.

5.4 Kommunikációs diagramok

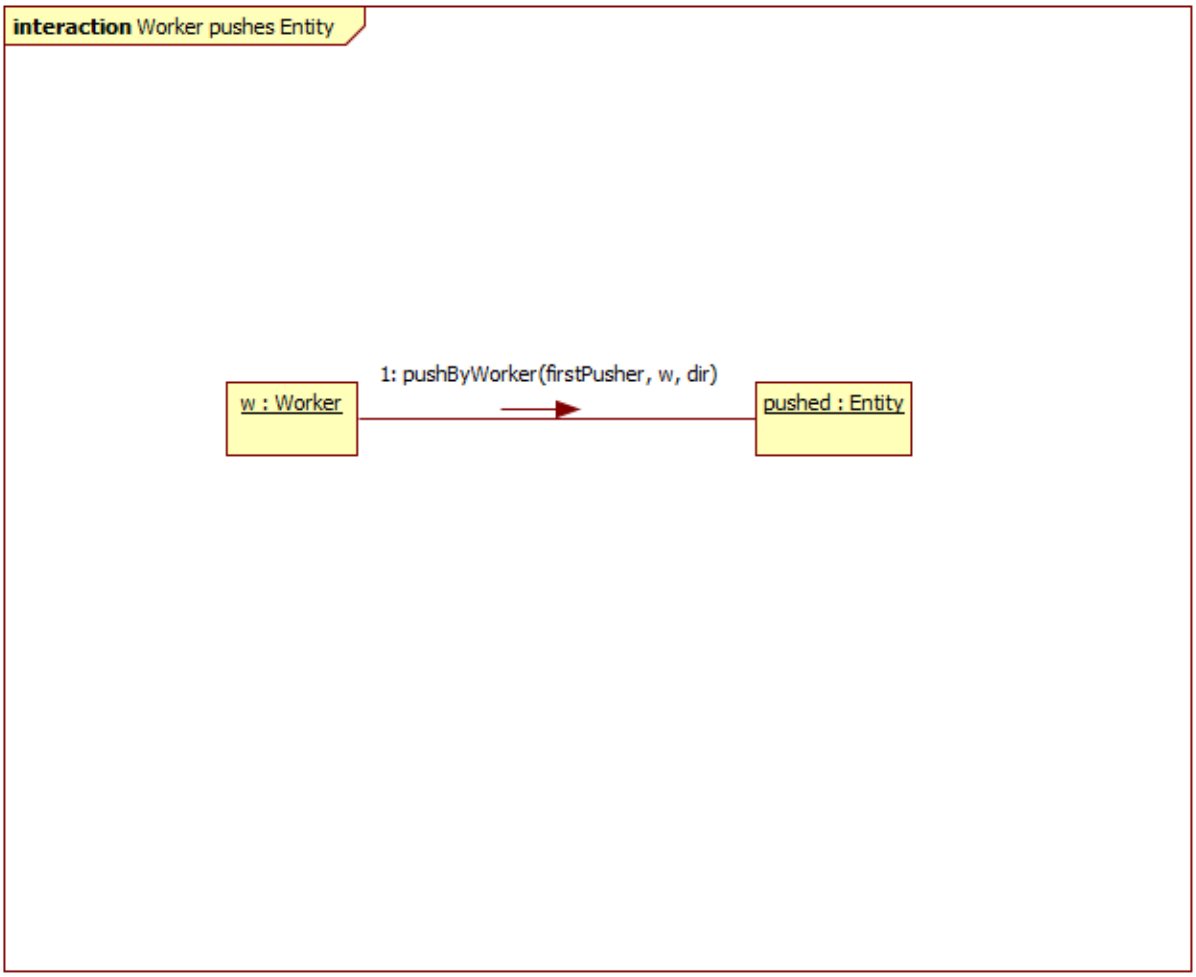
5.4.1 Entitás új mezőre lép



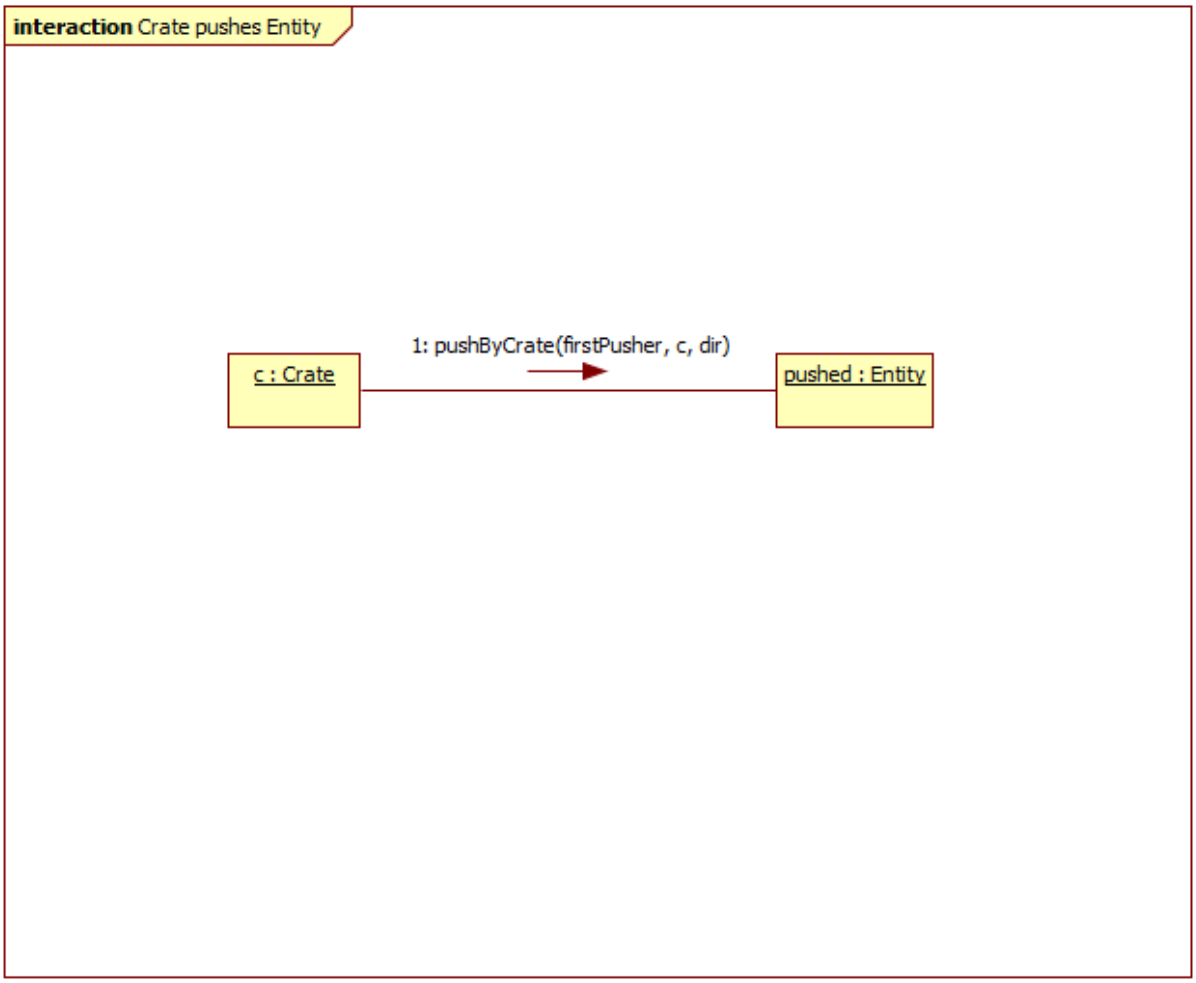
5.4.2 Entitás lép



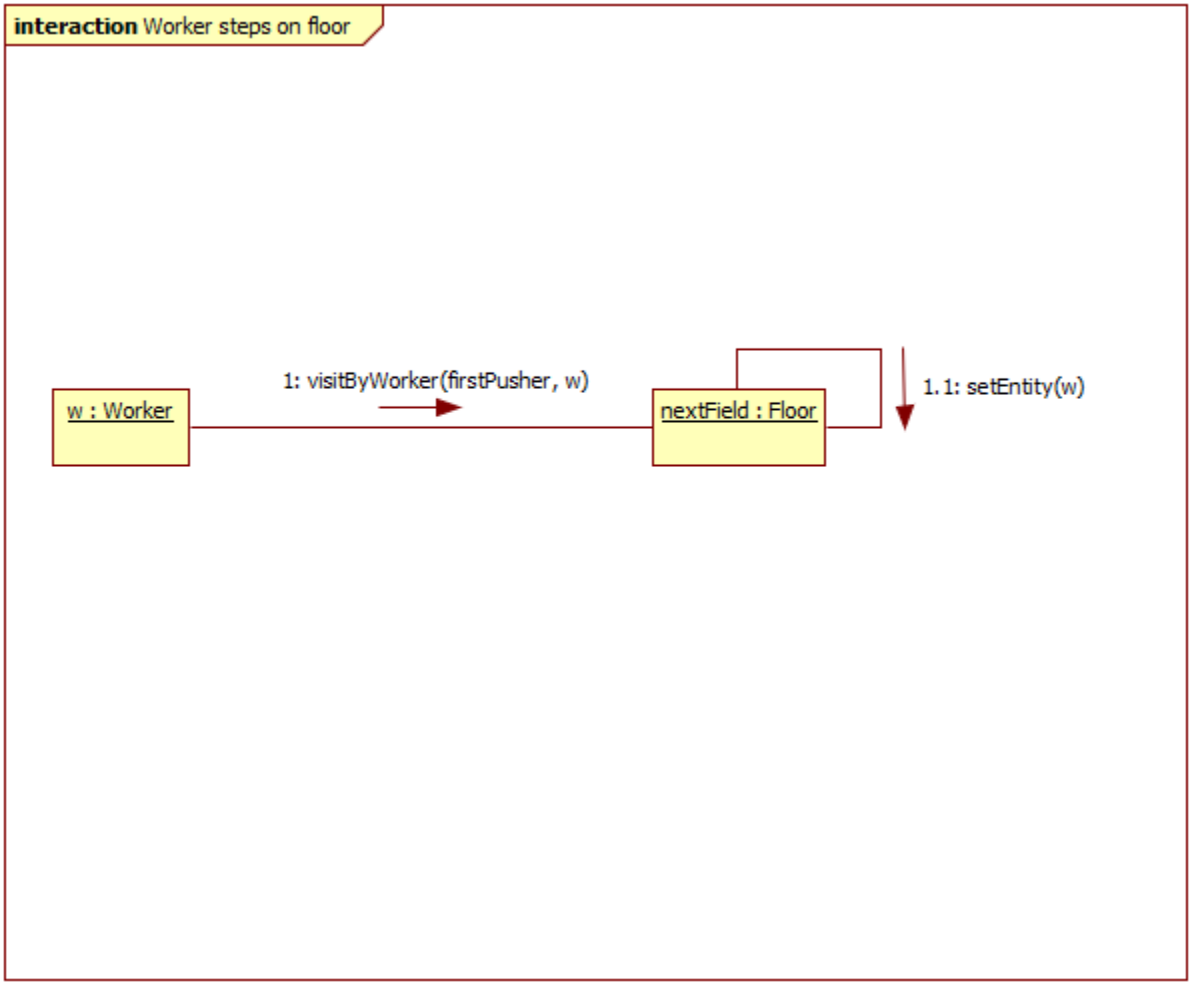
5.4.3 Dolgozó entitást tol



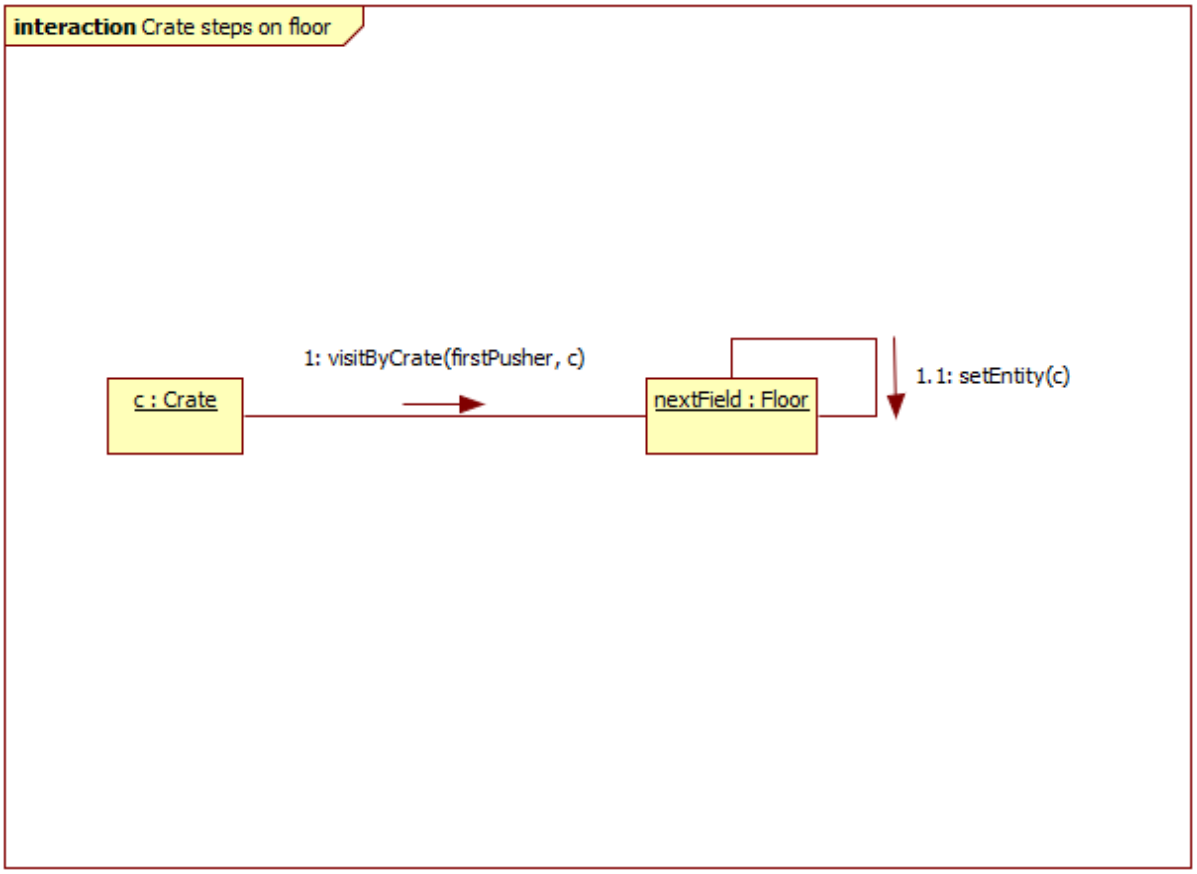
5.4.4 Láda entitást tol



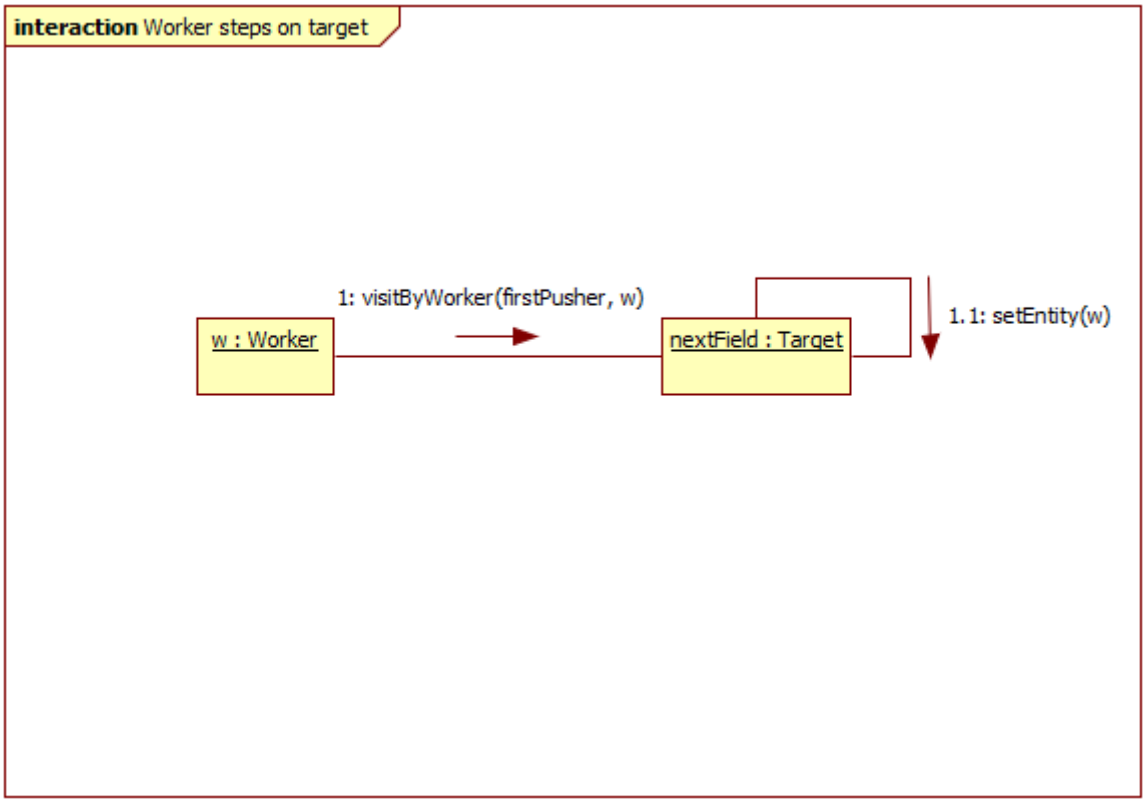
5.4.5 Dolgozó padlóra lép



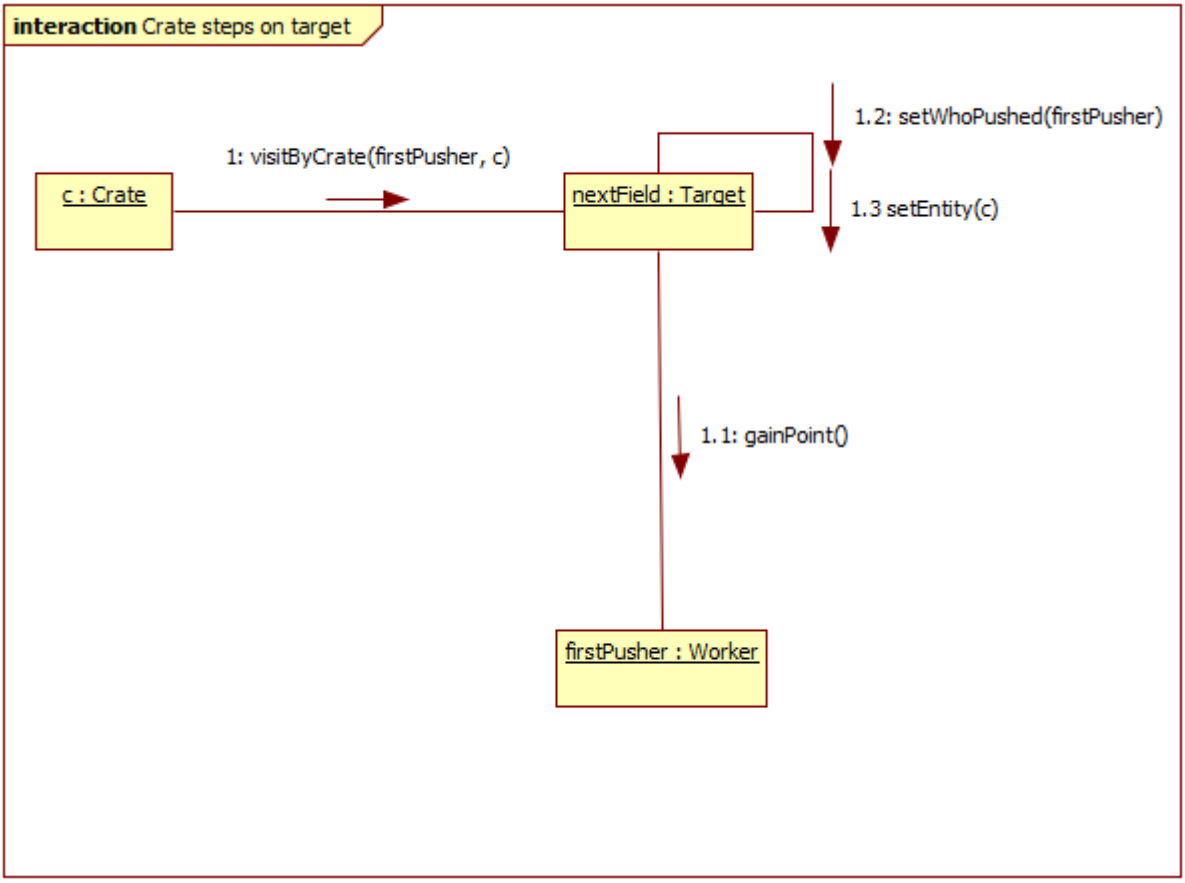
5.4.6 Láda padlóra lép



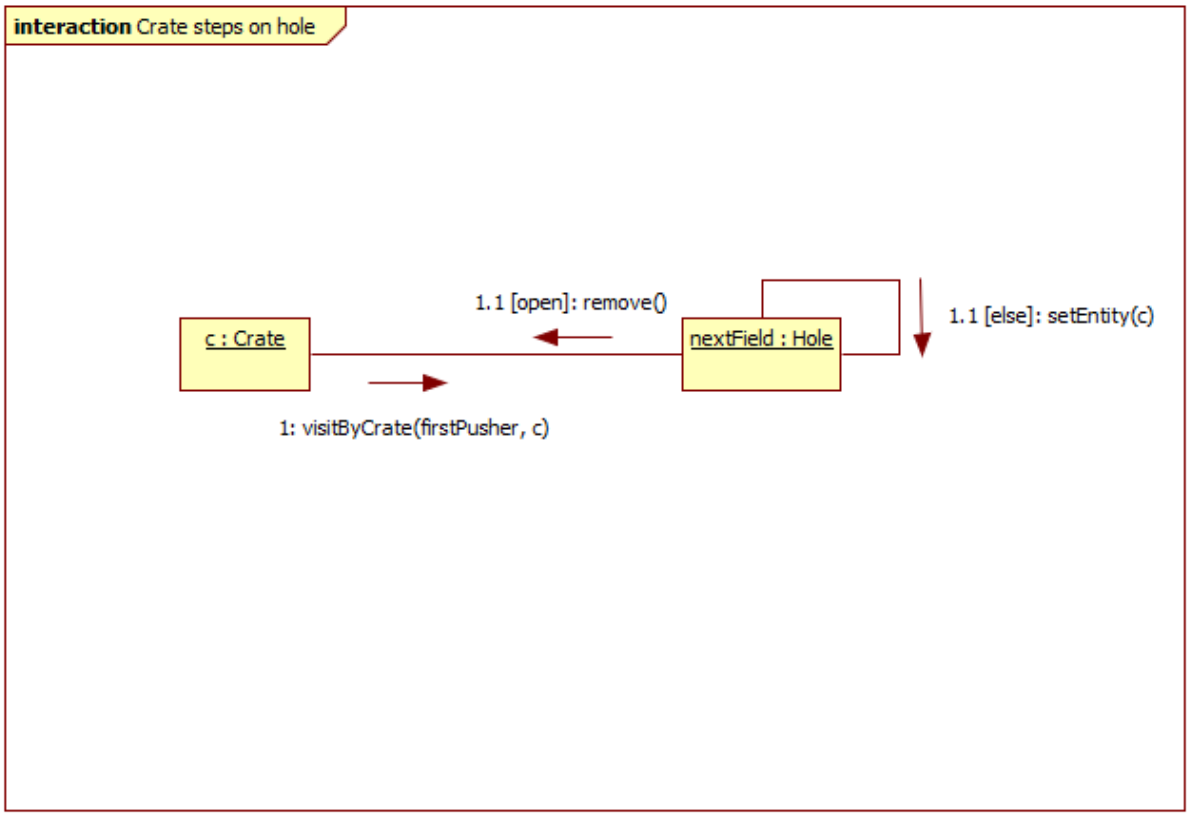
5.4.7 Dolgozó előírt helyre lép



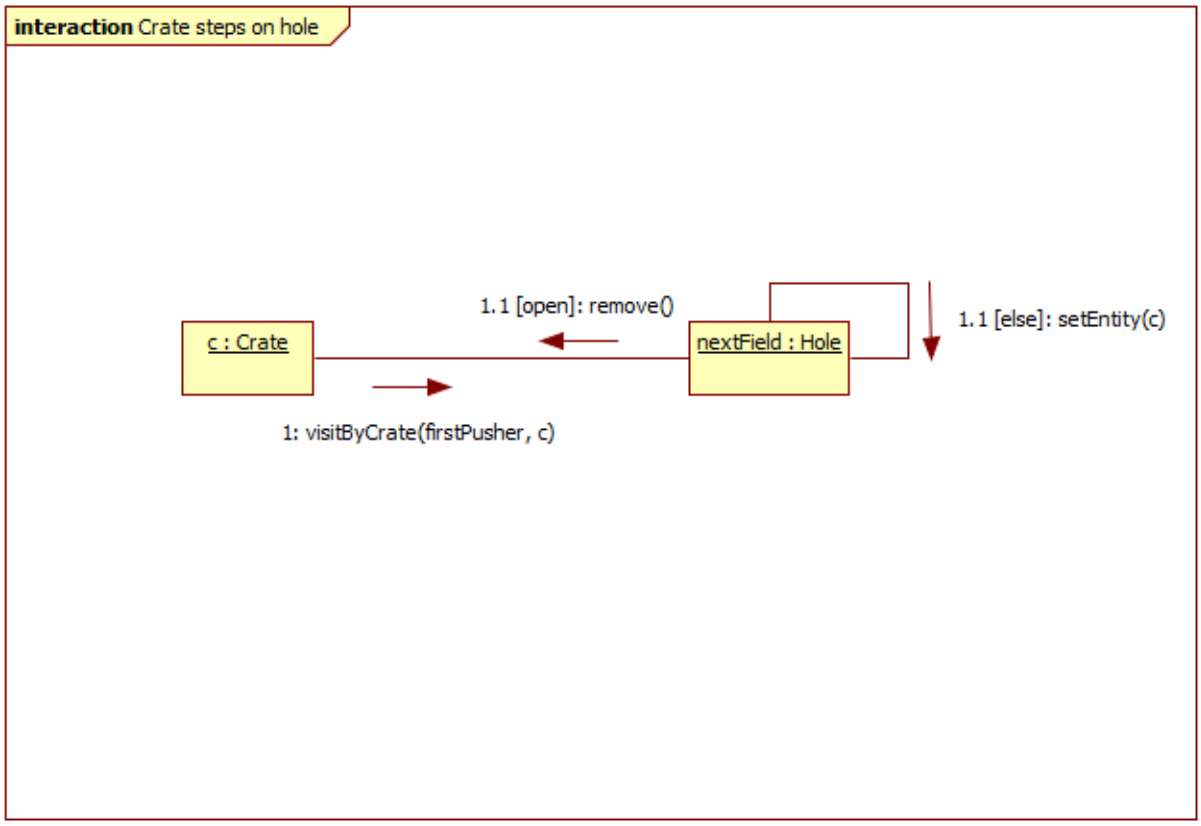
5.4.8 Láda előírt helyre lép



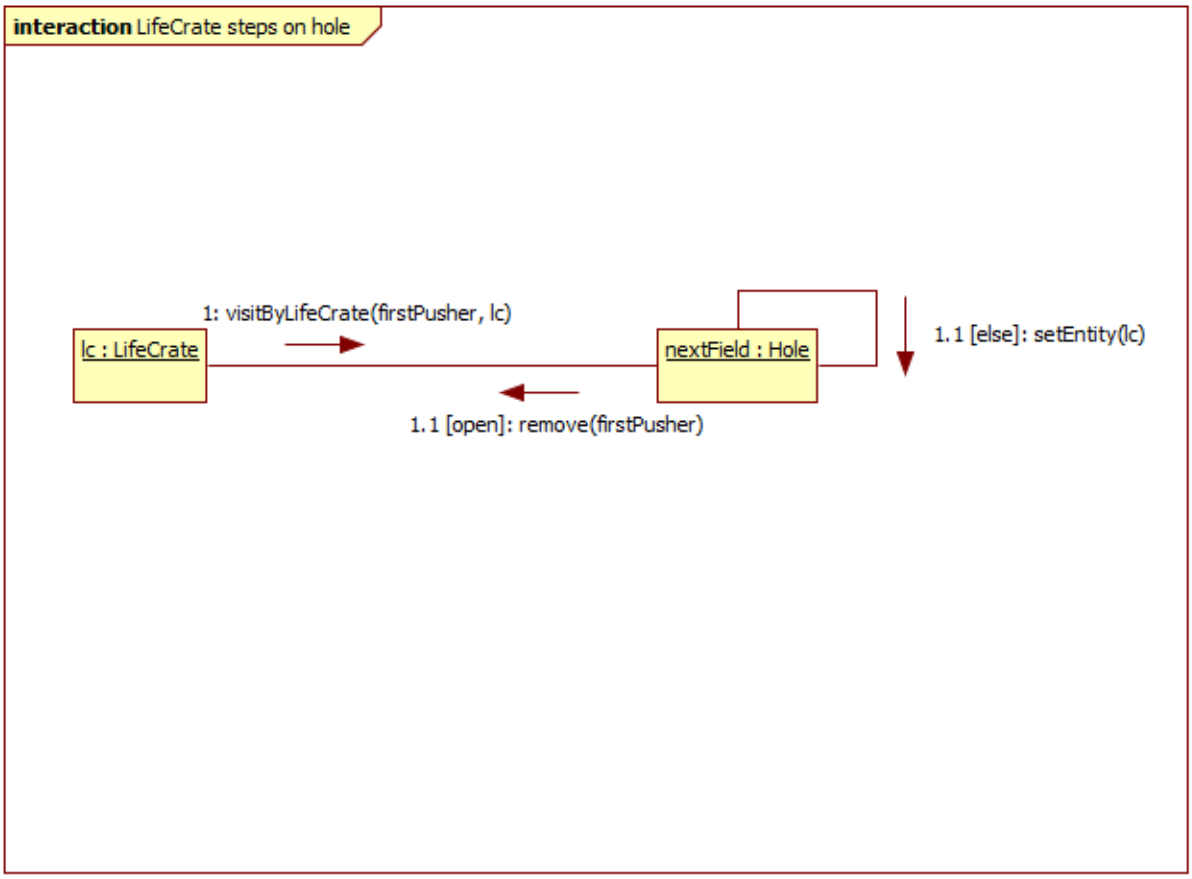
5.4.9 Dolgozó lyukra lép



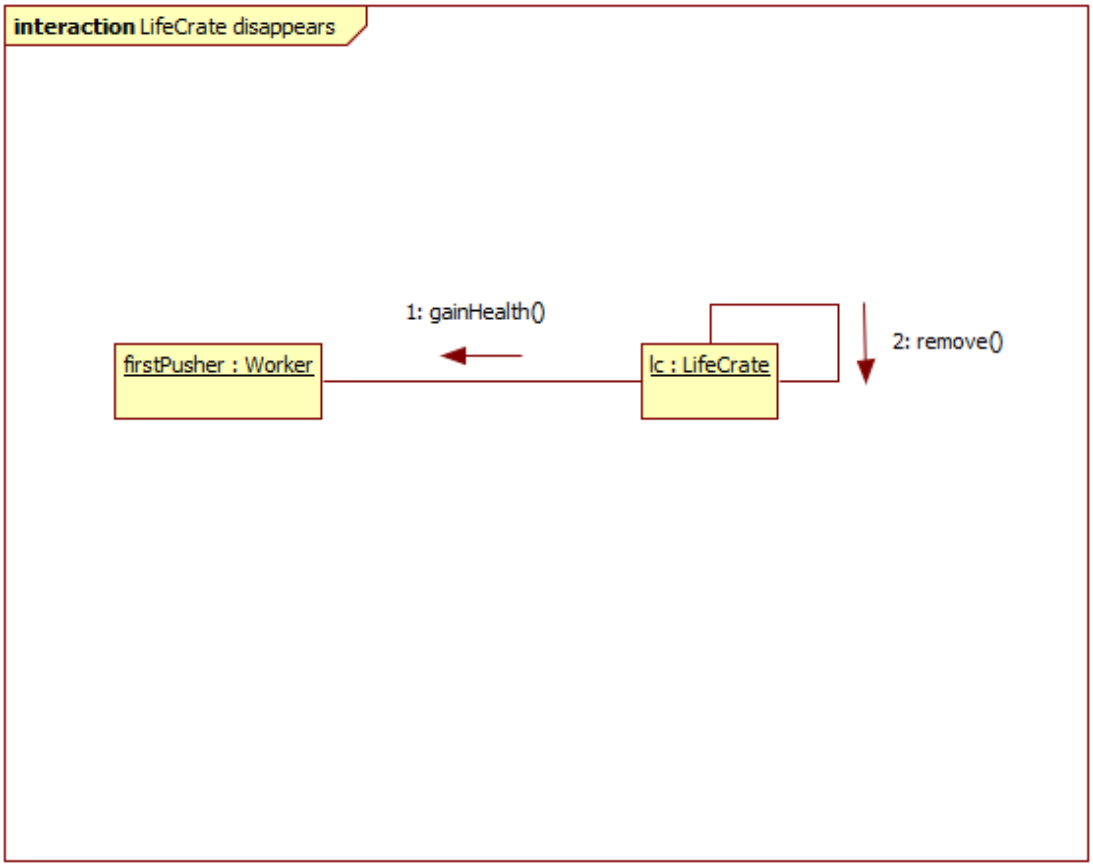
5.4.10 Láda lyukra lép



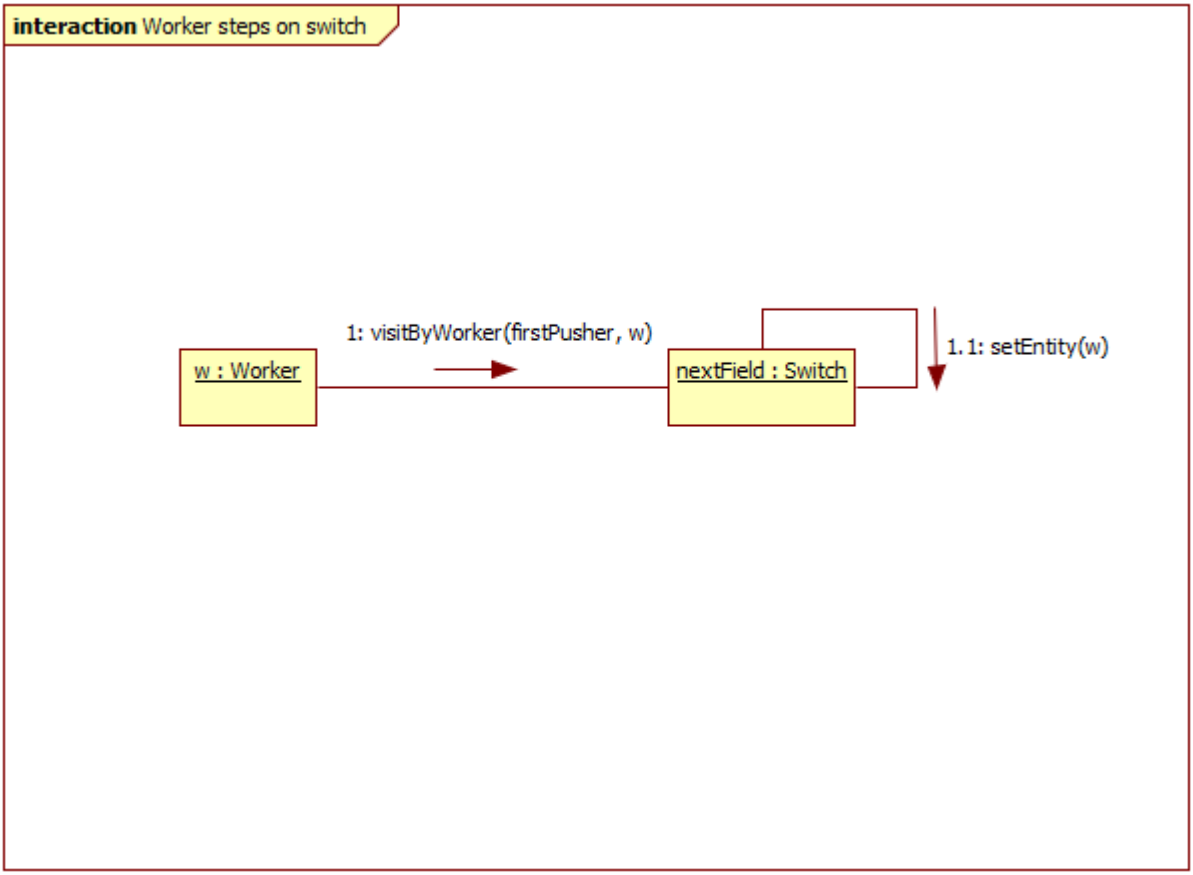
5.4.11 Szívecskés láda lyukra lép



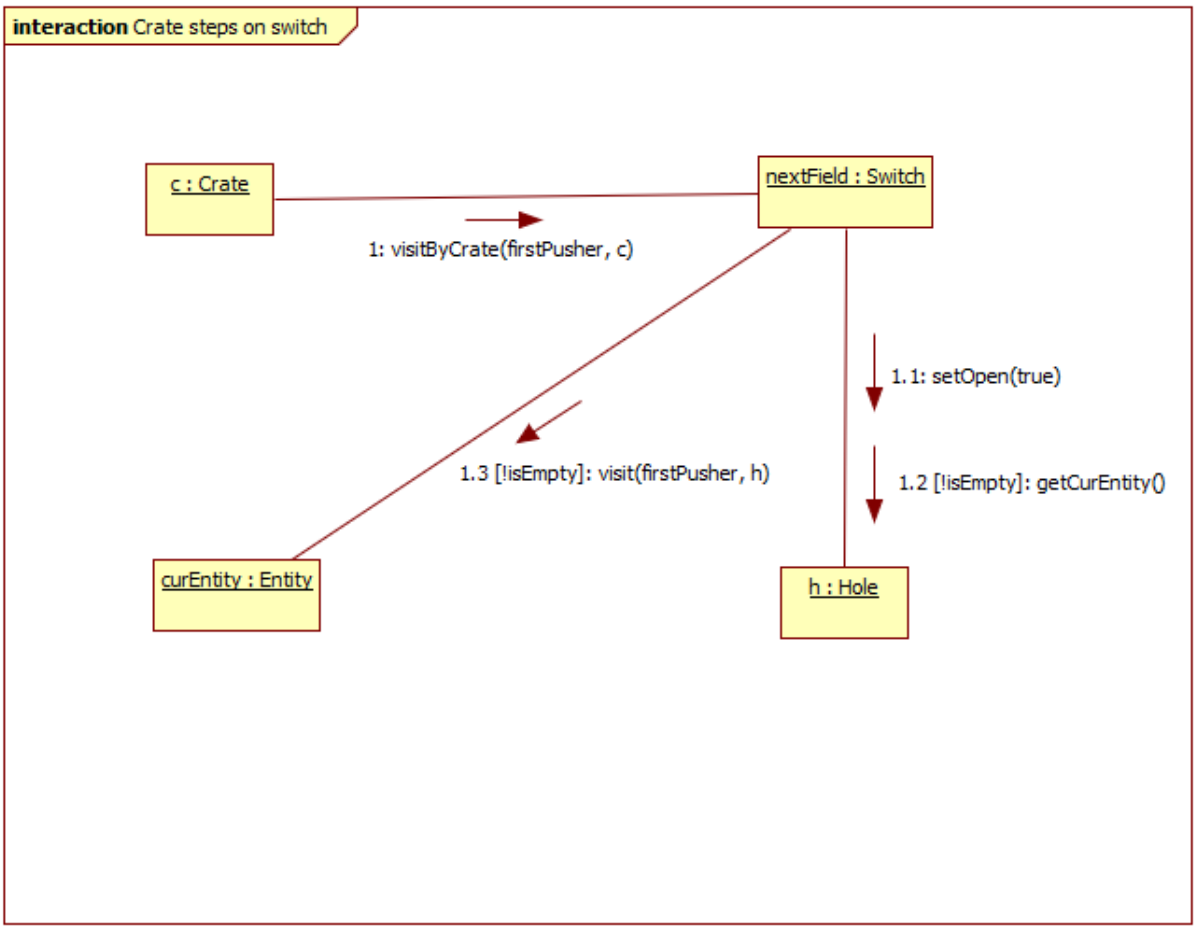
5.4.12 Szívecskés láda eltűnik



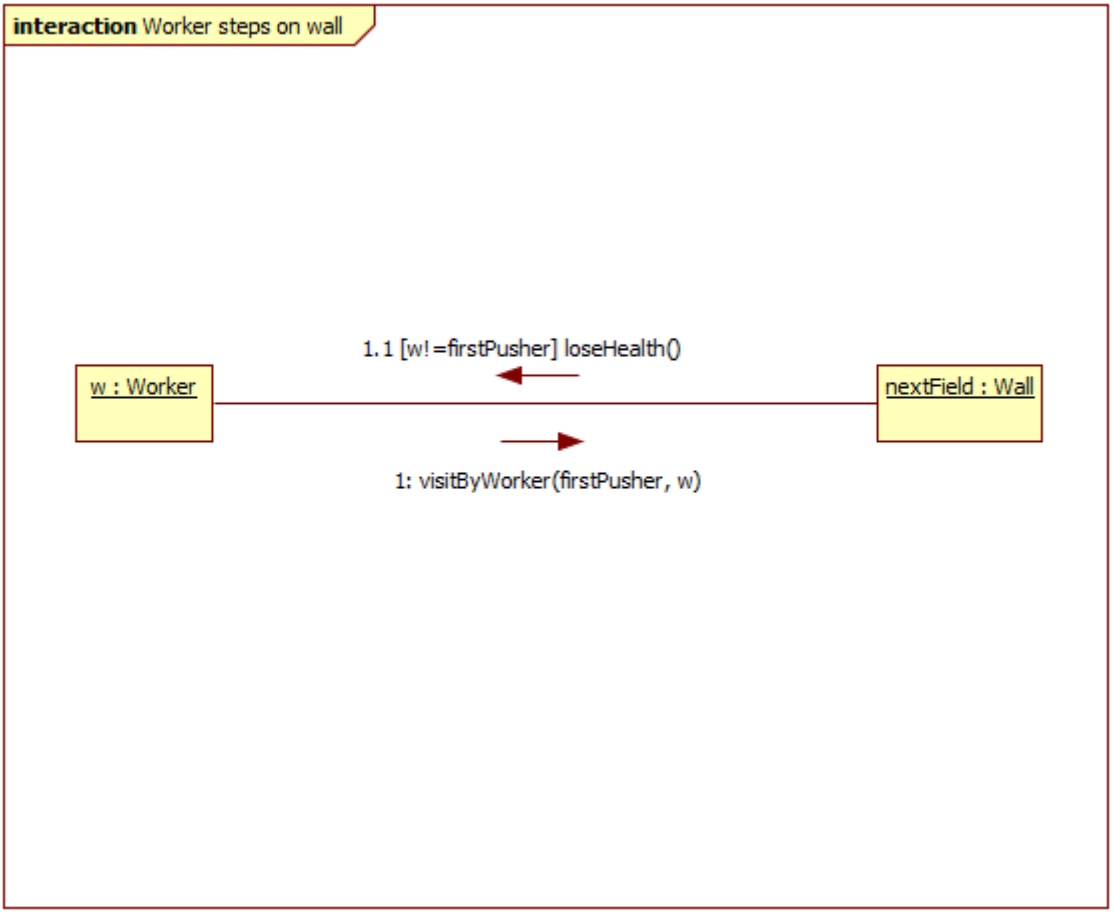
5.4.13 Dolgozó kapcsolóra lép



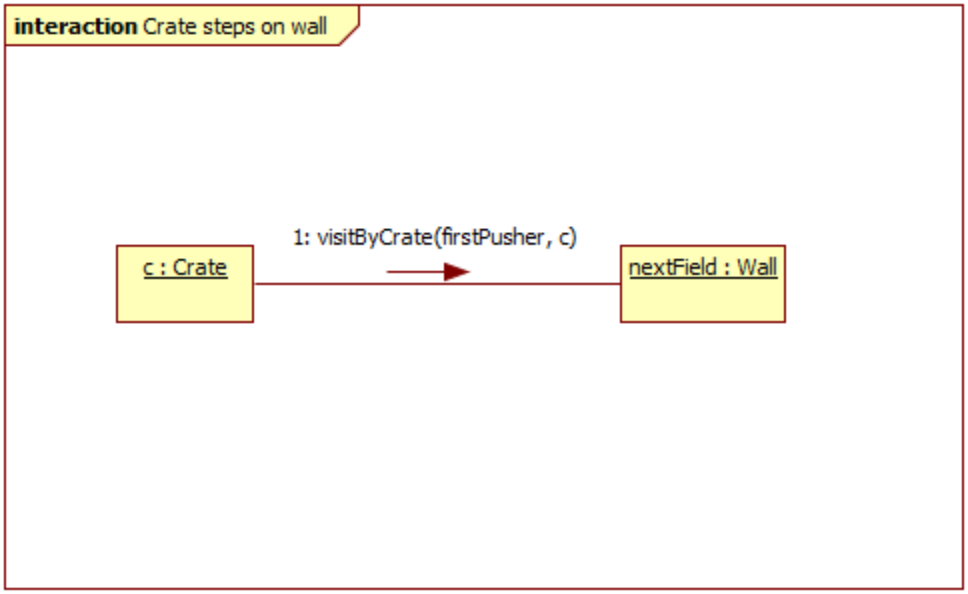
5.4.14 Láda kapcsolóra lép



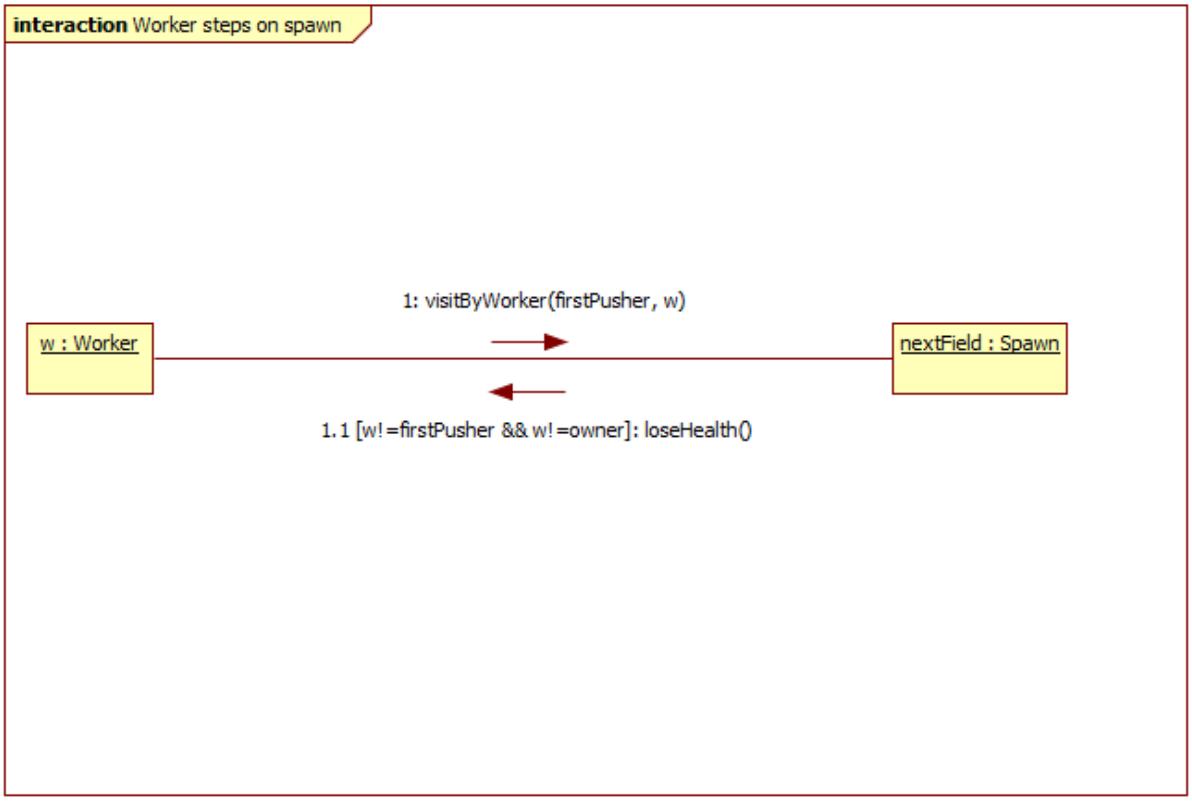
5.4.15 Dolgozó falra lép



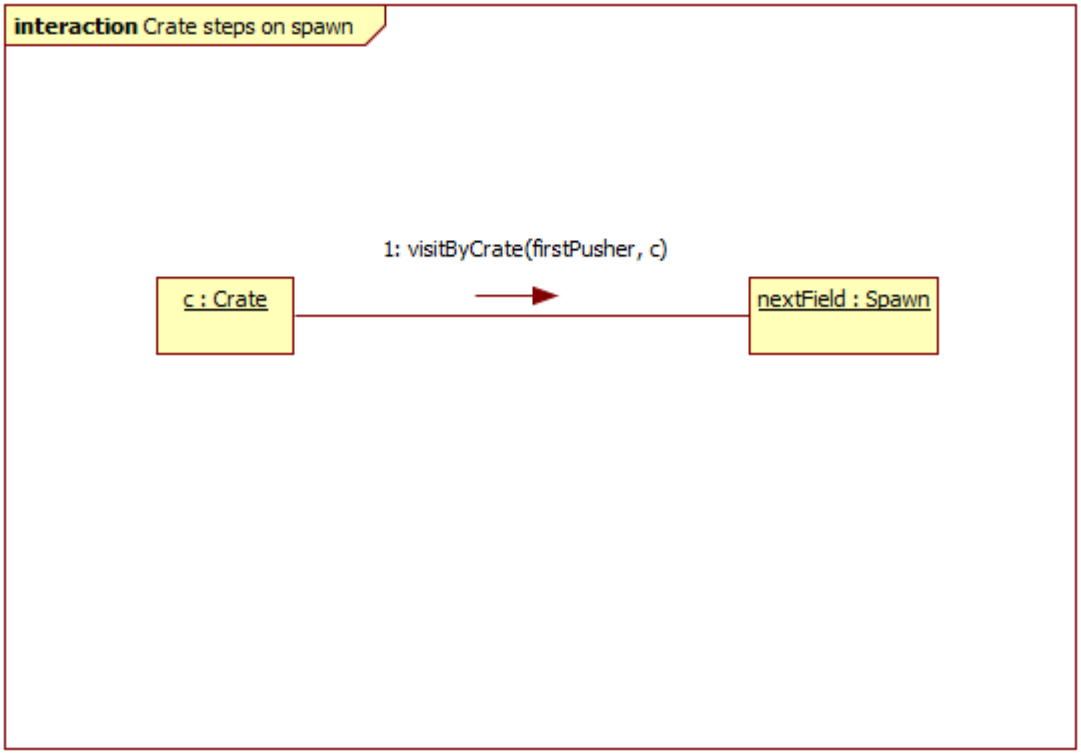
5.4.16 Láda falra lép



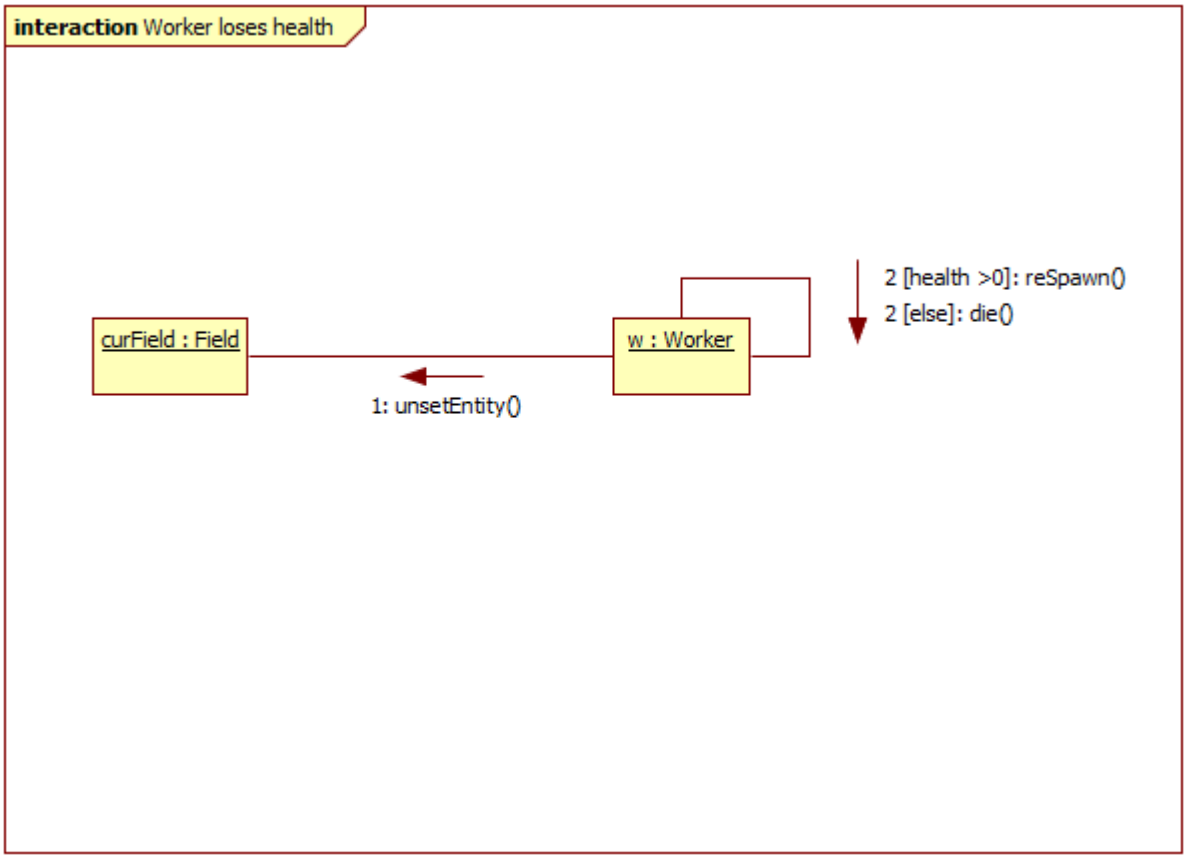
5.4.17 Dolgozó kiindulási helyre lép



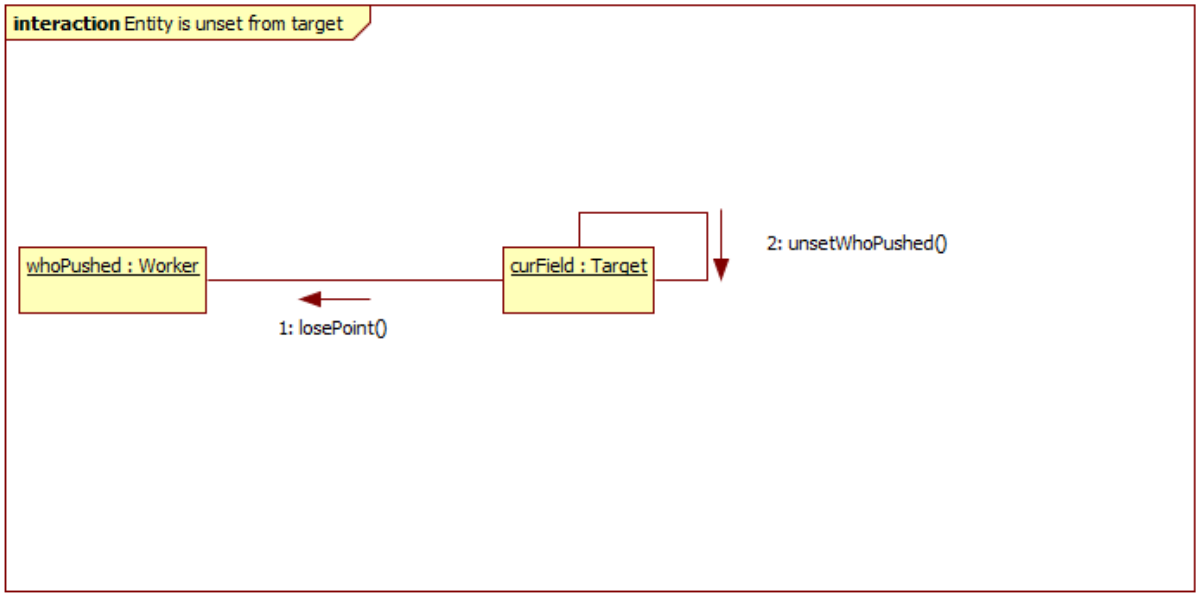
5.4.18 Láda kiindulási helyre lép



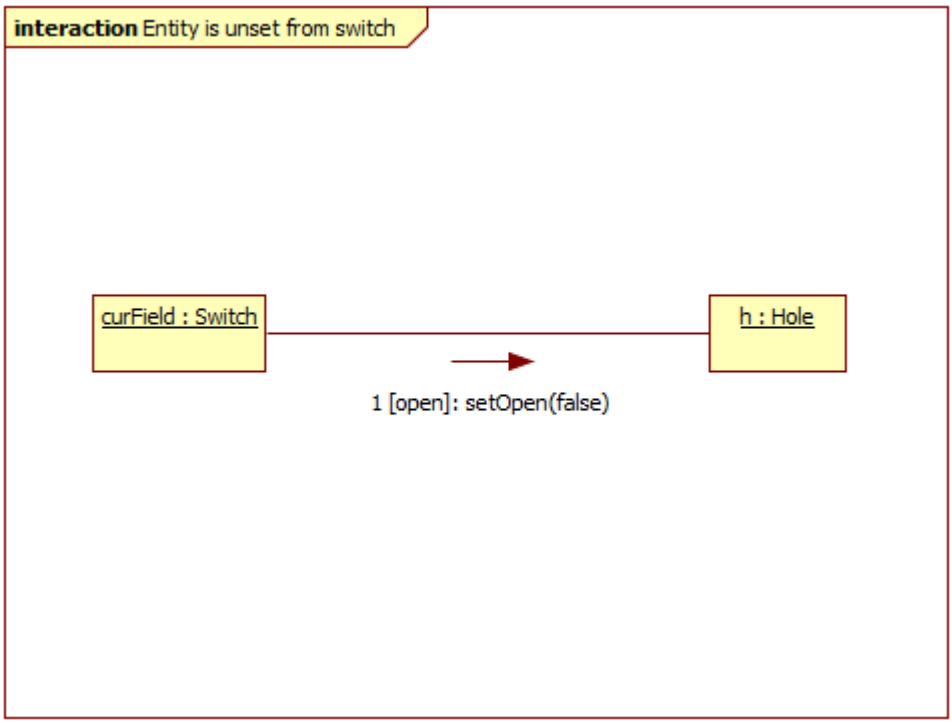
5.4.19 Dolgozó életet veszít



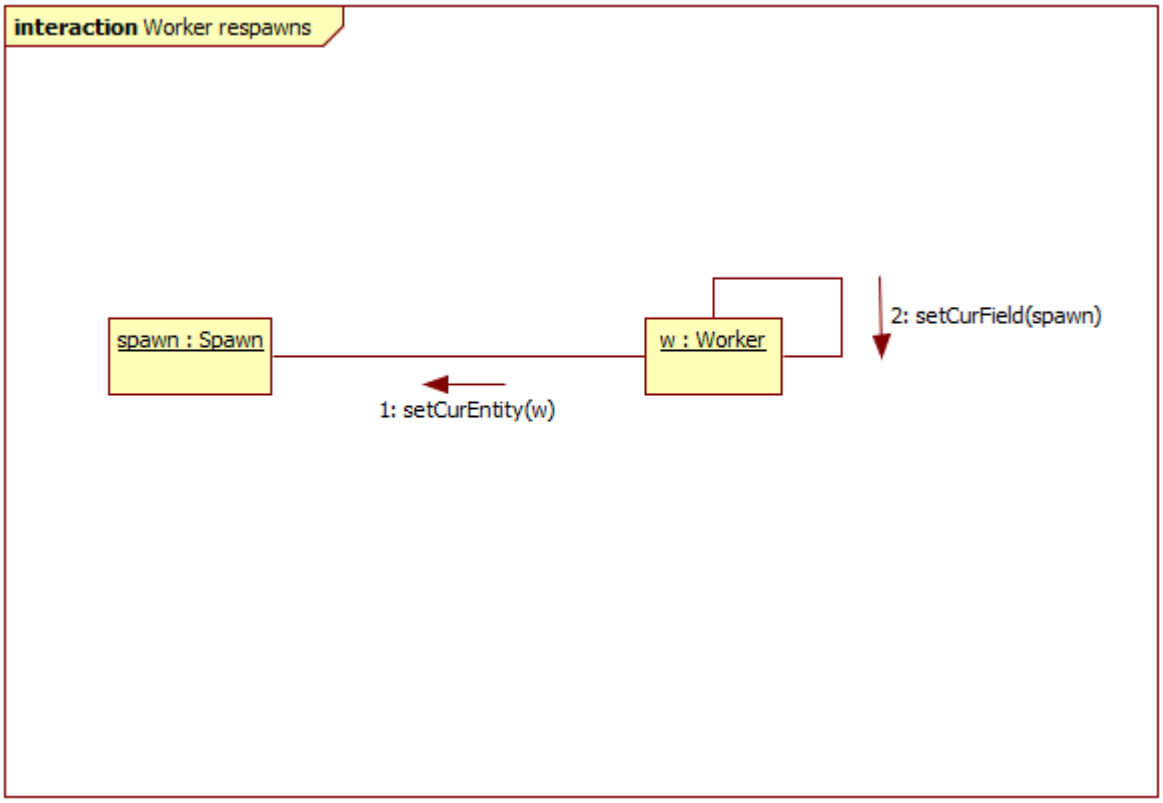
5.4.20 Entitás lekerül előírt helyről



5.4.21 Entitás lekerül kapcsolóról



5.4.22 Dolgozó respawnol



5.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.03.05. 12:00	2 óra	LAKATOS LENKEFI JANI CSANÁDY	Értekezlet, melynek döntései: CSANÁDY és JANI elkészíti a use-case diagramot és a use-casek leírását. LENKEFI elkészíti a kezelői felületet és dialógusok tervét. LAKATOS elkészíti a szekvencia diagramokat. SZAKÁLLAS elkészíti a kommunikációs diagramokat.
2018.03.05. 18:15	1,5 óra	CSANÁDY	Use-case diagram (5.1.1) koncepciójának megtervezése.
2018.03.08. 21:00	2,5 óra	SZAKÁLLAS	Use-case diagram (5.1.1) elkészítése.
2018.03.09. 17:00	1 óra	JANI	Use-case leírások (5.1.2) elkészítése.
2018.03.08. 20:30	3 óra	SZAKÁLLAS	Kommunikációs diagramok (5.4) elkészítése.
2018.03.09. 18:00	2 óra	LAKATOS	Szekvencia diagramok (5.3) elkészítése.
2018.03.10. 22:45	2 óra	CSANÁDY	Use-case leírások pontosítása (5.1.2).
2018.03.10. 23:30	1,5 óra	SZAKÁLLAS	Use-case leírások pontosítása (5.1.2).
2018.03.11. 9:45	2 óra	JANI	Dokumentum általános felülvizsgálata, konzisztencia ellenőrzése, apróbb módosítások.
2018.03.11. 18:30	2 óra	LENKEFI	A kezelői felület és dialógusok tervének elkészítése (5.2).
2018.03.11. 21:00	2 óra	LENKEFI	Dokumentum általános felülvizsgálata, konzisztencia ellenőrzése.
2018.03.11. 23:00	1 óra	LAKATOS	Dokumentum véglegesítése.

6. Szkeleton beadás

6.1 Fordítási és futtatási útmutató

6.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom

AutomaticQuestionHandler.java	1.96 KB	2018.03.16. 19:51	AutomaticQuestionHandler segédosztály.
Crate.java	1.87 KB	2018.03.17. 19:28	Crate osztály.
Direction.java	82 Byte	2018.03.14. 08:10	Direction enumeráció.
Entity.java	4.06 KB	2018.03.17. 19:28	Entity osztály.
Field.java	4.25 KB	2018.03.17. 19:28	Field osztály.
Floor.java	1.13 KB	2018.03.17. 19:28	Floor osztály.
Hole.java	3.01 KB	2018.03.17. 19:28	Hole osztály.
IQuestionHandler.java	1.26 KB	2018.03.16. 19:40	IQuestionHandler segédinterfész.
IVisitable.java	1.49 KB	2018.03.17. 19:28	IVisitable interfész.
IVisitor.java	710 Byte	2018.03.17. 19:28	IVisitor interfész.
LifeCrate.java	1.13 KB	2018.03.17. 19:28	LifeCrate osztály.
Main.java	5.70 KB	2018.03.16 20:24	Main osztály. Program belépési pont.
PrettyPrinter.java	5.93 KB	2018.03.16. 20:15	PrettyPrinter segédosztály.
RecorderQuestionHandler.java	1.31 KB	2018.03.16. 19:57	RecorderQuestionHandler segédosztály.
Spawn.java	2.40 KB	2018.03.17. 19:28	Spawn osztály.
Switch.java	1.53 KB	2018.03.17. 19:28	Switch osztály.
Target.java	1.28 KB	2018.03.17. 19:28	Target osztály.
Wall.java	1.27 KB	2018.03.17. 19:28	Wall osztály.
Warehouse.java	1.21 KB	2018.03.16. 11:26	Warehouse osztály.
Worker.java	4.72 KB	2018.03.17. 19:28	Worker osztály.

6.1.2 Fordítás

javac -d bin skeleton/.java skeleton/meta/*.java skeleton/model/*.java*

6.1.3 Futtatás

bin mappában: *java skeleton.Main*

6.2 Értékelés

Tag neve	Munka százalékban
Csanády Máté Bende	15.47 %
Jani Balázs Gábor	19.62 %
Lakatos Dániel	23.02 %
Lenkefi Péter	19.62 %
Szakállas Gergely	22.26 %

6.3 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.03.15. 16:00	1 óra	LENKEFI	A szkeleton alapvető szerkezetének elkészítése, project és osztályok előkészítése.
2018.03.16. 10:00	3 óra	CSANÁDY	Entity, Worker, Crate és LifeCrate osztályok és metódusaik megvalósítása.
2018.03.16. 14:00	3 óra	LAKATOS	Floor, Target és Wall osztályok és metódusaik megvalósítása.
2018.03.17. 08:00	3 óra	SZAKÁLLAS	Spawn, Hole és Switch osztályok és metódusaik megvalósítása.
2018.03.17. 14:00	2 óra	JANI	IVisitor, IVisitable, Warehouse és Field osztályok és metódusaik megvalósítása.
2018.03.17. 14:00	2 óra	LENKEFI	Main és a szkeletonhoz szükséges segédosztályok és metódusaik megvalósítása.
2018.03.18. 10:00	1 óra	LENKEFI	Az elkészült osztályok és a konzolos felület összehangolása.
2018.03.18. 16:00	1 óra	LENKEFI	A program tesztelése.

7. Prototípus koncepciója

7.0 Változás hatása a modellre

A megrendelő által igényelt változások a következők voltak:

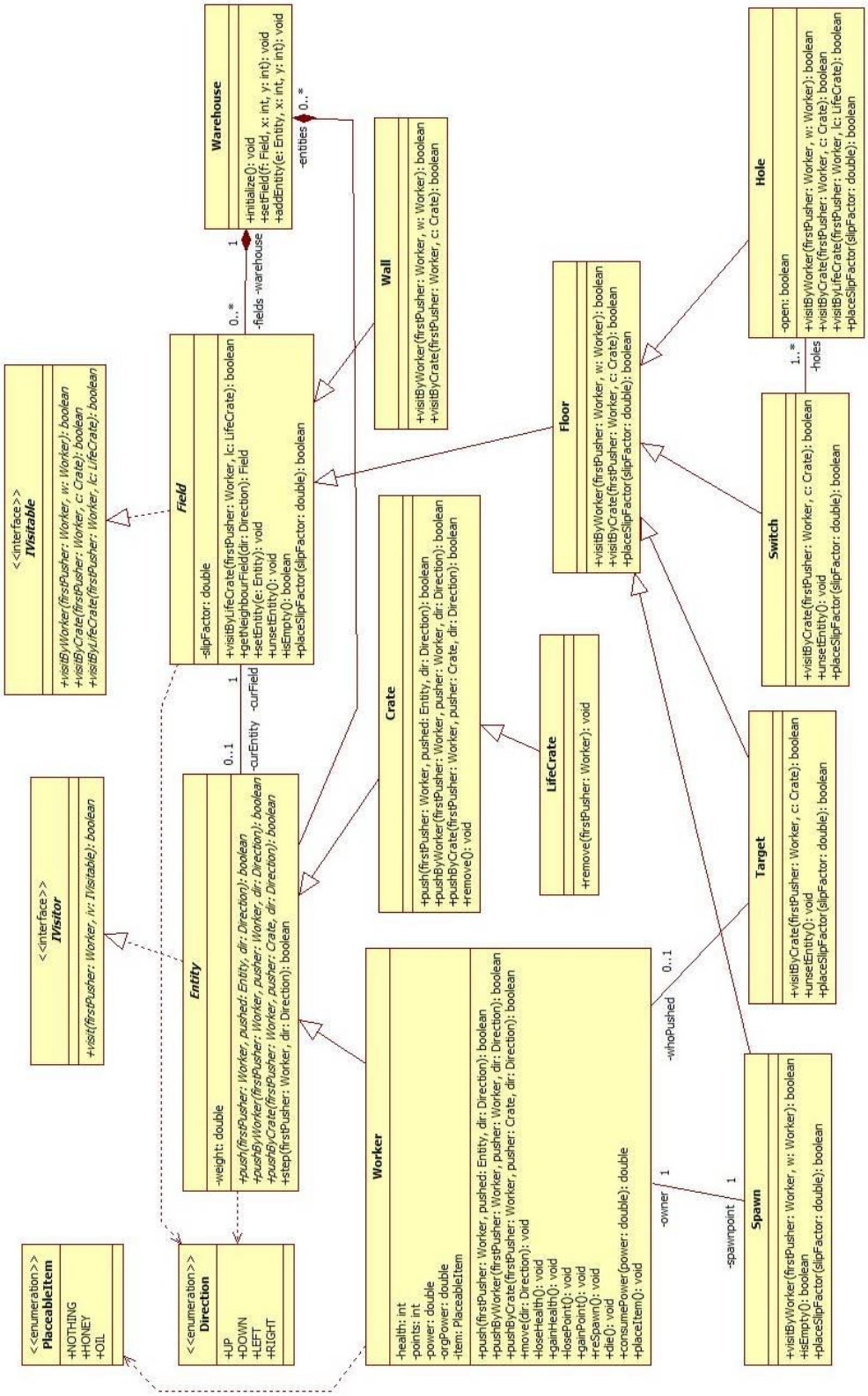
- *A munkások bár több ládát is eltolhatnak egyszerre, minden munkás rá jellemző erővel tol. Ha a ládák együttes tapadási súrlódási ereje ennél nagyobb, akkor a tolás nem sikerül.*
- *A padlóra különböző kenőanyagokat tehetnek a munkások: olajat, amitől csúszósabb lesz (csökken a tapadása) és mézet, amitől ragacsos (nő a tapadása).*

Megoldásunkban minden entitás rendelkezik egy rá jellemző tömeggel, ill. minden mező rendelkezik egy rá jellemző tapadási faktorral. A tapadási faktor alapállapotban 1.0. Ha a tapadási faktor ennél nagyobb, az mézet reprezentál a mezőn, ha ennél kisebb, az olajat.

Toláskor a munkás kiindulási erejével tol, ami mezőnként csökken a mezőn lévő entitás tömegének és a mező tapadási faktorának szorzatával. Ha a lánc utolsó entitásának megtolása utána a munkás csökkentett ereje nem negatív, az azt jelenti, hogy sikerült a tolás, ellenkező esetben nem sikerült.

Az osztálydiagramon nem volt szükséges felvenni új osztályt, egyetlen enumerációval, valamint metódusokkal és attribútumokkal bővítettük csak.

7.0.1 Módosult osztálydiagram



7.0.2 Új vagy megváltozó metódusok

7.0.2.1 Field

- **boolean placeSlipFactor(double slipFactor):** Egy entitás próbálja megváltoztatni egy mező tapadását. 1.0-nál nagyobb érték esetén a mező tapad (méz lesz rajta), 1.0-nál kisebb érték esetén a mező csúszik (olaj lesz rajta). Az anyag a mezőről folyamatosan párolog, így a tapadási tényező az idő múlásával 1.0-hoz tart. Ha a tapadási tényező 1.0, a mezőről eltűnik az anyag és a rajta lévő entitás tapadási súrlódási erejét nem befolyásolja többé. Nem minden mezőnek változtatható meg a tapadási tényezője, így a művelet igazzal tér vissza, ha sikerült a változtatás, egyébként pedig hamissal.

7.0.2.2 Floor

- **boolean placeSlipFactor(double slipFactor):** Egy entitás próbálja megváltoztatni egy padló tapadását. Ez lehetséges, így a művelet igazzal tér vissza.

7.0.2.3 Spawn

- **boolean placeSlipFactor(double slipFactor):** Egy entitás próbálja megváltoztatni egy kiindulási hely tapadását. A művelet hamissal tér vissza, mert ez nem lehetséges.

7.0.2.4 Target

- **boolean placeSlipFactor(double slipFactor):** Egy entitás próbálja megváltoztatni egy előírt hely tapadását. A művelet hamissal tér vissza, mert ez nem lehetséges.

7.0.2.5 Switch

- **boolean placeSlipFactor(double slipFactor):** Egy entitás próbálja megváltoztatni egy kapcsoló tapadását. A művelet hamissal tér vissza, mert ez nem lehetséges.

7.0.2.6 Hole

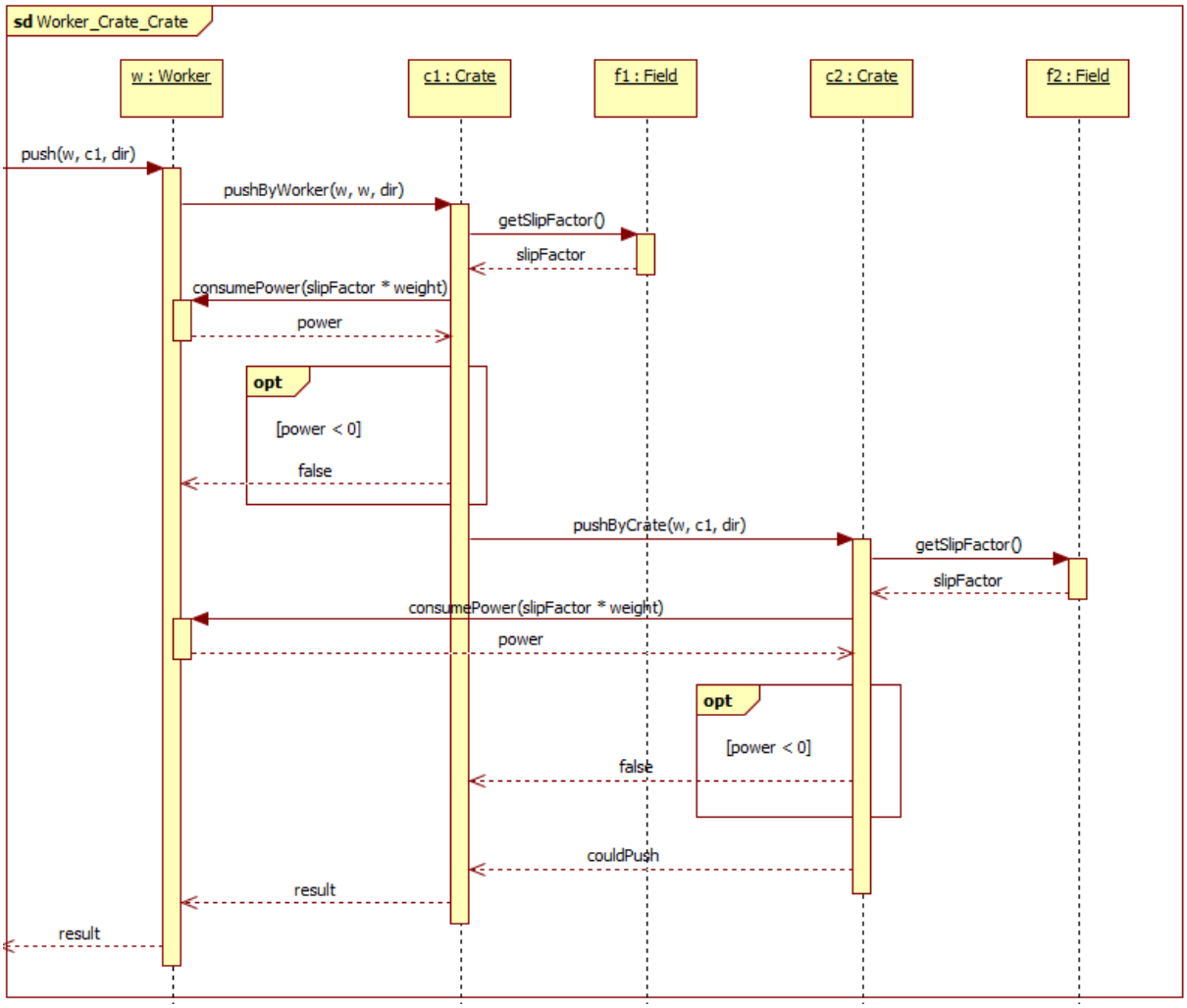
- **boolean placeSlipFactor(double slipFactor):** Egy entitás próbálja megváltoztatni egy lyuk tapadását. A művelet hamissal tér vissza, mert ez nem lehetséges.

7.0.2.7 Worker

- **double consumePower(double power):** A munkás ereje az átvett értékkel csökken, mivel éppen próbál valamit eltolni. A művelet a munkás csökkentett erejét adja vissza, melynek negatív volta jelzi, hogy a munkásnak nincs elég ereje a toláshoz.
- **void placeItem():** A munkás megpróbálja letenni a nála lévő dolgot arra a mezőre, amin jelenleg tartózkodik.

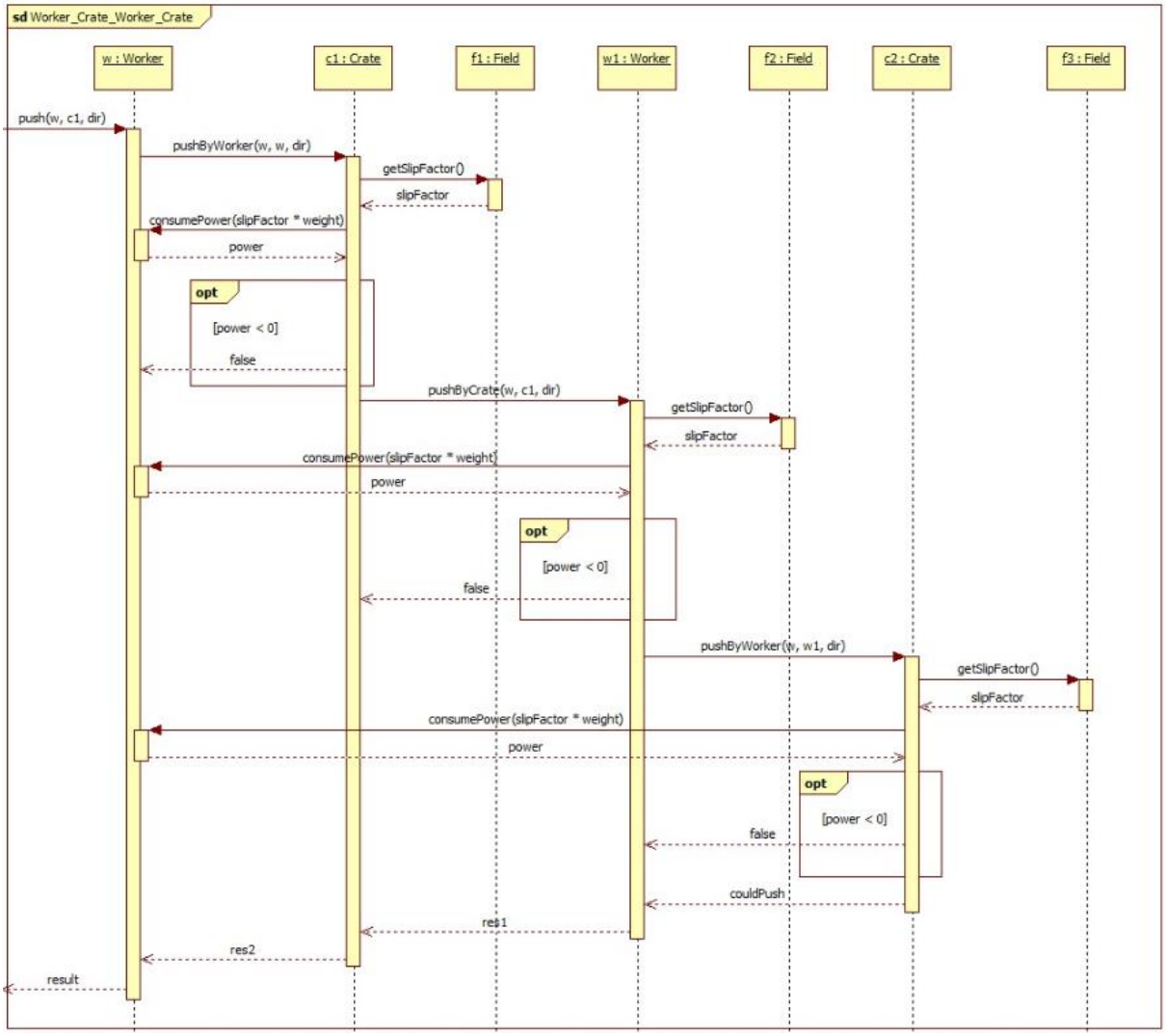
7.0.3 Szekvencia-diagramok

7.0.3.1 Munkás-láda-láda



A munkás adott erőjével megpróbálja eltolni a ládát. Ekkor a láda a saját tömegéből és a mező csúszási faktorából kiszámolt erővel csökkenti a munkás tolóerejét. Ha az így csökkent erő nemnegatív, akkor a munkásnak van ereje eltolni a ládát. A második láda eltolása hasonlóan történik, azzal a különbséggel, hogy itt az első láda által csökkentett erő szerepel, mint tolási elő, így ez csökken tovább. Ha a második csökkentés után a munkás ereje negatív lenne, az azt jelentené, hogy két láda eltolására már nincs elég ereje, így természetesen az egész láncot sem tudná eltolni.

7.0.3.2 Munkás-láda-munkás-láda



Az előzővel analóg módon történik a munkás eltolása. A munkásnak is van tömege, így az ő eltolásához is erő kell. (Természetesen munkást továbbra is csak láncban tolhatunk, a közvetlen tolás erőtől függetlenül nem lehetséges).

7.1 Prototípus interface-definíciója

7.1.1 Az interfész általános leírása

Az interfész csak a szabványos bemenetről fogad parancsokat, és a szabványos kimenetre írja ki az esetleges kimenetet. Ezáltal terminálból is használható, valamint az elkészítendő tesztelő segédprogram segítségével átirányítható a ki- és bemenet fájlokra, így van mód automatikus tesztelésre, előre elkészített teszteseteket felhasználva. A tesztesetek a prototípusnak adandó parancsok sorozatából, valamint az adott sorozatra adandó helyes kimenet található. A tesztek sikeresek, ha a valós, és a leírt elvárt kimenet megegyezik.

7.1.2 Bemeneti nyelv

STEP <player index> <direction>

Leírás: A ‘player index’-edik játékosnak ad ki mozgási utasítást.
Opciók: *player index*: A játékos sorszáma [1 - 4].
direction: A lépés iránya L, R, U D mint Left, Right, Up, Down.

PLACE <player index>

Leírás: A ‘player index’-edik játékost utasítja a nála levő tárgy (méz/olaj) lerakására.
Opciók: *player index*: A játékos sorszáma [1 - 4].

WAIT <time>

Leírás: A további parancsok futtatása előtt vár egy bizonyos időt. Az aszinkron interakciók teszteléséhez szükséges.
Opciók: *time*: A várakozandó idő milliszekundumban.

SAVE <name>

Leírás: Elmenti a futás aktuális állapotát egy bizonyos néven.
Opciók: *name*: A mentés neve, ahogy később hivatkozni tudunk rá.

LOAD <name>

Leírás: Betölt egy korábban mentett futási állapotot.
Opciók: *name*: A hivatkozott korábbi állapot neve.

LEVEL <name>

Leírás: Betölt egy adott nevű pályát.
Opciók: *name*: A betöltendő pálya neve.

LISTLEVELS

Leírás: Kilistázza az elérhető pályákat.
Opciók: -

LISTSAVES

Leírás: Kilistázza az elérhető mentett futási állapotokat.
Opciók: -

SWITCHINFO <switch_xy>

Leírás: Megadja egy kapcsoló állapotát.
Opciók: *switch_xy:* A kapcsoló x és y koordinátája, vesszővel elválasztva.

HOLEINFO <switch_xy>

Leírás: Megadja egy kapcsolóhoz tartozó lyukak állapotát.
Opciók: *switch_xy:* A kapcsoló x és y koordinátája, vesszővel elválasztva.

PLAYERINFO <player index> <type>
Leírás: A 'player index'-edik játékosról szolgáltat valamilyen státusz-információt.
Opciók: *player index*: A játékos sorszáma [1 - 4].
type: HP (a munkás életereje), PTS (a munkás pontjainak száma), ITEM (a munkásnál lévő letehető dolog).

CNT <option>
Leírás: A munkásokról és a ládákról szolgáltat számszerű adatokat.
Opciók: *option:* WA (az életben lévő munkások száma), WD (a halott munkások száma), C (a ládák száma), CR (az előírt helyen lévő ládák száma), CW (az előírt helyen nem lévő ládák száma), CS (a nem mozgatható ládák száma).

A pályák a "Tiled" nevű program JSON kimenetű file formátumában vannak tárolva méghozzá 2 rétegen, egy mező és egy entitás rétegen.

7.1.3 Kimeneti nyelv

A kimeneti nyelv minden parancsa elé rakható egy NOT, amivel jelezzük hogy a bizonyos dologtól elvárjuk hogy az NE legyen (viszont ekkor már ugyanarra a parancsra valaminek az igazát nem tudjuk már vizsgálni).

STEP:
- STEP_OK <player index>
- STEP_FAIL <player index>

PLACE:
- PLACE_OK <player index>
- PLACE_FAIL <player index>

SAVE/LOAD/LEVEL:
- (SAVE/LOAD/LEVEL)_SUCCESS <name>
- (SAVE/LOAD/LEVEL)_FAIL <name>

LISTLEVELS/LISTSAVES:
- (LEVELS/SAVES){ <összes file név vesszővel elválasztva idézőjelek közt>}

SWITCHINFO:
- ACTIVE <switch_xy>
- NOT_ACTIVE <switch_xy>

HOLEINFO:

- OPEN <switch_xy>
- CLOSED <switch_xy>

PLAYERINFO:

- PLAYER_(HP/PTS/ITEM) <player index>, <a kérdezett dolog>
- kérdezett dolog HP, PTS-re szám, ITEM-re ‘H’ ha méz, ‘O’ ha olaj, ‘N’ ha semmi

CNT:

- (WA/WD/C/CR/CW/CS) <number>

7.2 Összes részletes use-case

Use-case neve	**STEP**
Rövid leírás	Lépteti a megadott játékos dolgozóját a megadott irányba.
Aktorok	Tesztelő
Forgatókönyv	A megadott dolgozó a megadott irányba lép, és lehetőség szerint akár egy komplexebb lépési vagy tolási szekvenciát is végrehajthat annak érdekében, hogy a megadott irányban lévő mezőre kerüljön. A lépés vagy sikerül, vagy nem.

Use-case neve	**PLACE**
Rövid leírás	A játékos dolgozója mézet vagy olajat rak a padlóra.
Aktorok	Tesztelő
Forgatókönyv	A kiválasztott dolgozó a nála lévő tárgyat elhelyezi a padlón, amin tartózkodik. Ennek a hatása akkor érvényesül, amikor a dolgozó lelép a mezőről, akár mert magától lép vagy mert eltolják onnan.

Use-case neve	**WAIT**
Rövid leírás	Adott ideig várakoztatja a programot.
Aktorok	Tesztelő
Forgatókönyv	A parancs beütését követően a sorba következő parancsok csak azután futnak le, hogy a visszaszámláló letelik. A korábban bevitt parancsok hatása továbbra is érvényes és szabadon változtathatják a futás kimenetét.

Use-case neve	**SAVE**
Rövid leírás	A futás állapotát elmenti egy megadott fájlnéven.
Aktorok	Tesztelő
Forgatókönyv	A megadott fájlnéven lementésre kerül a futás aktuális állapota, ahonnan az később visszatölthető.

Use-case neve	**LOAD**
Rövid leírás	Beölt egy korábban elmentett futási állapotot.
Aktorok	Tesztelő
Forgatókönyv	A parancs betölti a megadott néven található mentett futást és lehetővé teszi annak folytatását, ha az létezik, ellenkező esetben hibaüzenetet dob.

Use-case neve	**LEVEL**
Rövid leírás	Betölt egy pályát a megadott néven.
Aktorok	Tesztelő
Forgatókönyv	A parancs betölti a megadott néven található pályát, ha az létezik, ellenkező esetben hibaüzenetet dob.

Use-case neve	**LISTLEVELS**
Rövid leírás	Kilistázza az elérhető pályákat.
Aktorok	Tesztelő
Forgatókönyv	Kilistázza a konzolra az elérhető pályákat, ha vannak.

Use-case neve	**LISTSAVES**
Rövid leírás	Kilistázza az elérhető korábban elmentett futási állapotokat.
Aktorok	Tesztelő
Forgatókönyv	Kilistázza a konzolra az elérhető korábban elmentett futási állapotokat, ha vannak.

Use-case neve	**SWITCHINFO**
Rövid leírás	Megadja egy kapcsoló állapotát.
Aktorok	Tesztelő
Forgatókönyv	Megadja egy kapcsoló állapotát, azaz, hogy egy kapcsolón tartózkodik-e jelenleg láda vagy sem.

Use-case neve	**HOLEINFO**
Rövid leírás	Megadja egy kapcsolóhoz tartozó lyukak állapotát.
Aktorok	Tesztelő
Forgatókönyv	Megadja egy kapcsolóhoz tartozó lyukak állapotát, azaz, hogy egy kapcsolóhoz tartozó lyukak nyitva vagy csukva vannak-e.

Use-case neve	**PLAYERINFO**
Rövid leírás	Megadja egy játékos állapotát.
Aktorok	Tesztelő
Forgatókönyv	Lekérdezhető egy játékos állapota, azaz az életereje, a pontjainak a száma, valamint az éppen nála lévő lerakható dolog típusa is.

Use-case neve	**CNT**
Rövid leírás	Megadja kiválasztott tulajdonságú entitások számát.
Aktorok	Tesztelő
Forgatókönyv	Lekérdezhető a halott és élő munkások száma, a ládák összes száma, az előírt helyen és a nem előírt helyen lévő, valamint a tovább már nem tolható (beragadt) ládák száma.

7.3 Tesztelési terv

Teszt-eset neve	**Move worker on empty field**
Rövid leírás	Egy munkás léptetésének tesztelése.
Teszt célja	Egy üres padlóra lépteti a munkást és ellenőrzi, hogy sikerült-e a lépés.

Teszt-eset neve	**Worker pushes crate to empty field**
Rövid leírás	Lépteti a munkást, ami így eltol egy ládát és ellenőrzi, hogy sikerült-e a tolás.
Teszt célja	Egy láda eltolásának tesztelése.

Teszt-eset neve	**Worker pushes other worker directly**
Rövid leírás	Lépteti a munkást egy olyan mezőre ahol egy másik munkás áll, és ellenőrzi hogy nem sikerült a lépés.
Teszt célja	Munkás közvetlen tolásának tesztelése

Teszt-eset neve	**Worker pushes other worker in chain**
Rövid leírás	Lépteti a munkást, hogy egy ládán keresztül eltoljon egy másik munkást, és ellenőrzi a tolás sikerességét.
Teszt célja	Munkás láncban való tolásának tesztelése

Teszt-eset neve	**Worker smashed by wall**
Rövid leírás	Lépteti egy munkást, hogy egy ládán keresztül falnak toljon egy másik munkást, és ellenőrzi a tolás sikerességét, valamint a meghalt munkás életét.
Teszt célja	Munkás falhoz való passzírozásának tesztelése

Teszt-eset neve	**Worker smashed by spawn**
Rövid leírás	Lépteti egy munkást, hogy egy ládán keresztül egy nem saját spawnra toljon egy másik munkást, és ellenőrzi a tolás sikerességét, valamint a meghalt munkás életét.
Teszt célja	Munkás spawnhoz való passzírozásának tesztelése

Teszt-eset neve	**Worker falls into hole**
Rövid leírás	Léptet egy munkást, hogy egy aktív lyukba lépjen, és ellenőrzi a meghalt munkás életét.
Teszt célja	Munkás lyukba esésének tesztelése

Teszt-eset neve	**Worker pushes lifecrate into hole**
Rövid leírás	Léptet egy munkást, hogy egy szívecskés ládát eltoljon egy lyukba, és ellenőrzi a tolás sikerességét, illetve a munkás életszámának növekvését.
Teszt célja	Szívecskés láda működésének tesztelése.

Teszt-eset neve	**Target activation test**
Rövid leírás	Léptet egy munkást, hogy az egy ládát toljon egy üres cél mezőre, majd ellenőrzi a tolás sikerességét és, hogy a munkás kapott-e pontot.
Teszt célja	Láda cél területre való mozgatásával szerezhető pontok tesztje.

Teszt-eset neve	**Target deactivation test**
Rövid leírás	Léptet egy munkást, hogy az egy ládát letoljon egy cél mezőről, majd ellenőrzi a tolás sikerességét és, hogy a ládát a targetra toló munkás vesztett-e pontot.
Teszt célja	Láda cél területre való mozgatásával szerezhető pontok tesztje.

Teszt-eset neve	**Switch activation test**
Rövid leírás	Léptet egy munkást, hogy az egy ládát toljon egy üres kapcsoló mezőre, majd ellenőrzi a tolás sikerességét és a kapcsolt mezők állapotát.
Teszt célja	Kapcsoló ládával való aktiválásának tesztje

Teszt-eset neve	**Switch deactivation test**
Rövid leírás	Léptet egy munkást, hogy az egy ládát eltoljon egy kapcsoló mezőről, majd ellenőrzi a tolás sikerességét és a kapcsolt mezők állapotát.
Teszt célja	Kapcsoló deaktiválásának tesztje

Teszt-eset neve	**Field modificator test - oil**
Rövid leírás	Egy olaj nélkül éppen nem eltolható sorba, elhelyez egy munkás olajat, és utána eltolja azt. Ellenőrizzük az eltolás sikerességét.
Teszt célja	Olaj síkosító hatásának tesztelése.

Teszt-eset neve	**Field modificator test - honey**
Rövid leírás	Egy méz nélkül éppen eltolható sorba, elhelyez egy munkás mézet, és utána megpróbálja eltolni azt. Ellenőrizzük az eltolás sikertelenségét.
Teszt célja	Méz tolás hátráltató hatásának tesztelése

Teszt-eset neve	**Locking mechanism of simultaneously pushed conflicting chains**
Rövid leírás	Időben nagyon közel egymás után elindított, egymás útjában álló tolásokat szimulál, majd ellenőrzi, hogy az első parancs helyesen lezárta-e a lánc tagjait amíg végre nem hajtódott a tolás.
Teszt célja	Egyszerre történő, egymást kizáró láncok viselkedésének teszte.

Teszt-eset neve	**Game ends because there is a crate on all target**

Rövid leírás	Léptet egy munkást, ami betol egy ládát, az utolsó szabad célterületre, majd ellenőrizzük, hogy észlelte-e a játék végét, és pontszám alapján a jó játékost választotta győztesnek.
Teszt célja	A célterületek feltöltése miatti játék vég észlelésének tesztje

Teszt-eset neve	**Game ends because there are no moveable crates**
Rövid leírás	Léptet egy munkást, ami betolja az utolsó mozgatható ládát, az egy olyan helyre, ahol az beragad. Majd ellenőrizzük, hogy észlelte-e a játék végét, és pontszám alapján a jó játékost választotta győztesnek.
Teszt célja	Beragadás miatti játék vég észlelésének tesztje

Teszt-eset neve	**Game ends because all the workers died except one**
Rövid leírás	Egy munkást úgy mozgat, hogy meghaljon (vagy lyukba lép, vagy összepaszíroz egy másikat), majd ellenőrziük, hogy észlelte-e azt, hogy a játéknak vége kell lennie és az életben maradt játékos nyert.
Teszt célja	Játékos elimináció miatti játék vég észlelésének tesztelése
Teszt-eset neve	**Player gains an item**
Rövid leírás	A szituációban megteremti a tárgy szerzésnek a feltételeit. Majd ellenőrizzük hogy ténylegesen meg kapja-e a tárgyat a munkás.
Teszt célja	Tárgy (pl.: olaj, méz) szerzésének tesztelése

Teszt-eset neve	**Worker puts down item**
Rövid leírás	A worker elhelyez egy tárgyat arra a mezőre amin épp áll. Ellenőrizzük, hogy ténylegesen lekerült-e a tárgy, valamint, hogy már nincs a munkásnál.
Teszt célja	Tárgy elhelyezésének tesztje

7.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása

A bemeneti nyelv által specifikált tesztesetek szöveges formátumban, külön file-okban vannak tárolva. A szoftver induláskor betölti a TEST_ALL nevű szöveges file-t, mely soronként 2 filenevet/elérési utat tartalmaz. A sorban első a bemeneti nyelvet tartalamazó szöveg, egy konkrét teszt eset. A második pedig az ehhez elvárt kimenet.

7.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.03.22. 20:00	2 óra	LAKATOS	A változtatások megvalósítása és dokumentálása. (7.0)
2018.03.23. 16:00	2 óra	CSANÁDY	A prototípus alapgondolatának, bemeneti és kimeneti nyelvének megfogalmazása. (7.1)

2018.03.23. 16:00	2 óra	LENKEFI	A prototípus alapgondolatának, bemeneti és kimeneti nyelvének megfogalmazása. (7.1)
2018.03.24. 23:00	2 óra	SZAKÁLLAS	Use-casek megfogalmazása. (7.2)
2018.03.25. 10:00	1 óra	SZAKÁLLAS	Tesztesetek megfogalmazása. (7.3)
2018.03.25. 15:00	2 óra	JANI	Tesztesetek megfogalmazása. (7.3)
2018.03.25. 23:00	1 óra	LAKATOS	Tesztelést támogató segéd- és fordítóprogramok specifikálása, a dokumentum véglegesítése. (7.4)

8. Részletes tervek

8.1 Osztályok és metódusok tervei

8.1.1 Crate

- **Felelősség**

Egy ládát szimbolizál. Nyilvántartja a láda pozícióját.

- **Ősosztályok**

Entity

- **Interfészek**

-

- **Attribútumok**
  - **-**

- **Metódusok**
  - **+ push(Worker, Entity, Direction): boolean:** A láda eltol egy entitást a megadott irányban lévő szomszédos mezőre. Igazzal tér vissza, ha sikerült eltolni, egyébként hamissal.
  - **+ pushByWorker(Worker, Worker, Direction): boolean:** Egy dolgozó tolja a ládát a megadott irányban lévő szomszédos mezőre, így a láda oda lép. Igazzal tér vissza, ha sikerült lépni, egyébként hamissal.
  - **+ pushByCrate(Worker, Crate, Direction): boolean:** Egy láda tolja a ládát a megadott irányban lévő szomszédos mezőre, így a láda oda lép. Igazzal tér vissza, ha sikerült lépni, egyébként hamissal.
  - **+ remove(): void:** A láda kikerül a raktárból.
  - **+ visit(Worker, IVisitable): boolean:** A láda egy vizitálható objektumra (mezőre) lép. Igazzal tér vissza amennyiben sikerült a lépés, egyébként hamissal.

8.1.2 Direction

- **Felelősség**

Az egyes lépési irányokat tartalmazza, mint fel (UP), le (DOWN), balra (LEFT) és jobbra (RIGHT).

8.1.3 Entity

- **Felelősség**

Absztrakt osztály, ami az entitásokat, avagy mozgatható dolgokat reprezentálja.

- **Ősosztályok**

-

- **Interfészek**

IVisitor

- **Attribútumok**
  - **- curField: Field:** A mező, amelyen jelenleg az entitás tartózkodik.
  - **- weight: Double:** Az entitás tömege.

- **Metódusok**
  - ***+ push(Worker, Entity, Direction): boolean:*** Megpróbál eltolni egy megadott irányban lévő mezőn tartózkodó entitást, szintén a megadott irányban. Átveszi a dolgozót, aki a láncot tolja. Igazzal tér vissza, amennyiben sikerült eltolnia az entitást, egyébként hamissal.
  - ***+ pushByWorker(Worker, Worker, Direction): boolean:*** Az entitást megpróbálja eltolni egy dolgozó a megadott irányban. Átveszi a dolgozót, aki a láncot tolja és az entitást tolni próbáló dolgozót. Igazzal tér vissza, amennyiben a tolás sikeres volt, egyébként hamissal.
  - ***+ pushByCrate(Worker, Crate, Direction): boolean:*** Az entitást megpróbálja eltolni egy láda a megadott irányban. Átveszi a dolgozót, aki a láncot tolja és az entitást tolni próbáló ládát. Igazzal tér vissza, amennyiben a tolás sikeres volt, egyébként hamissal.
  - **+ step(Worker, Direction): boolean:** Az entitás lép a megadott irányban lévő szomszédos mezőre. Ha léphet a következő mezőre, de ott tartózkodik egy entitás, akkor megpróbálja őt eltolni. Igazzal tér vissza, ha sikerült lépni, egyébként hamissal.
  - **+ setCurField(Field): void:** Beállítja az entitás mezőjét.
  - **+ getCurField(): Field:** Lekérdezi az entitás mezőjét.

8.1.4 Field

- **Felelősség**

Absztrakt osztály, ami a játéktér (raktár) egy mezőjét szimbolizálja.

- **Ősosztályok**

-

- **Interfészek**

IVisitable

- **Attribútumok**
  - **- curEntity: Entity:** Az entitás, ami jelenleg ezen a mezőn tartózkodik.
  - **- slipFactor: Double:** A mező tapadásának mértéke.
  - **- warehouse: Warehouse:** A raktár, amelyhez a mező tartozik.

- **Metódusok**
  - **+ visitByLifeCrate(Worker, LifeCrate): boolean:** Egy szívecskés láda a mezőre lép. Ugyanaz történik, mint amikor egy láda lép a mezőre.
  - **+ getNeighbourField(Direction): Field:** Az mező megadja az adott irányban lévő szomszédját.
  - **+ setEntity(Entity): void:** A mezőre helyez egy entitást.
  - **+ unsetEntity(): void:** A mezőn lévő entitás lekerül a mezőről, ami után a mező üres lesz.
  - **+ isEmpty(): boolean:** Megadja, hogy van-e épp entitás a mezőn. Ha van, hamissal tér vissza, egyébként igazzal.
  - **+ placeSlipFactor(double): boolean:** Egy entitás próbálja megváltoztatni egy mező tapadását. A művelet igazzal tér vissza, ha sikerült a változtatás, egyébként pedig hamissal.
  - **+ setCurEntity(Entity): void:** Beállítja a mezőn lévő entitást.
  - **+ getCurEntity(): Entity:** Lekérdezi a mezőn lévő entitást.

8.1.5 Floor

- **Felelősség**

A játéktér egy padlóját képviseli. Egy padlón tartózkodhat entitás, így rá is lehet lépni.

- **Ősosztályok**

Field

- **Interfészek**

-

- **Attribútumok**
  - **-**

- **Metódusok**
  - **+ visitByWorker(Worker, Worker): boolean:** Egy dolgozó a padlóra lép. Igazzal tér vissza, mivel egy munkás mindig ráléphet egy padlóra.
  - **+ visitByCrate(Worker, Crate): boolean:** Egy láda a padlóra lép. Igazzal tér vissza, mivel egy láda mindig ráléphet egy padlóra.
  - **+ placeSlipFactor(double): boolean:** Egy entitás próbálja megváltoztatni egy padló tapadását. Ez lehetséges, így a művelet igazzal tér vissza.

8.1.6 Hole

- **Felelősség**

A játéktér egy lyukas padlóját képviseli, ami lehet nyitott vagy zárt állapotú. Zárt állapotban ugyanúgy viselkedik, mint egy egyszerű padló. Nyitott állapotban mind ha dolgozó, mind ha láda lép rá, az beleesik.

- **Ősosztályok**

Field ⇨ Floor

- **Interfészek**

-

- **Attribútumok**
  - **- open: boolean:** Igaz, amennyiben a lyuk nyitott állapotban van, egyébként hamis.

- **Metódusok**
  - **+ visitByWorker(Worker, Worker): boolean:** Egy dolgozó a lyukra lép. Igazzal tér vissza, mivel egy munkás mindig ráléphet egy lyukra. Ha nyitva van a lyuk, a munkás leesik.
  - **+ visitByCrate(Worker, Crate): boolean:** Egy láda a lyukra lép. Igazzal tér vissza, mivel egy láda mindig ráléphet egy lyukra. Ha nyitva van a lyuk, a láda leesik.
  - **+ visitByLifeCrate(Worker, LifeCrate): boolean:** Egy szívecskés láda a lyukra lép. Igazzal tér vissza, mivel egy szívecskés láda mindig ráléphet egy lyukra. Ha nyitva van a lyuk, a szívecskés láda leesik és életet ad annak a dolgozónak, aki a lyukra tolta.
  - **+ placeSlipFactor(double): boolean:** Egy entitás próbálja megváltoztatni egy lyuk tapadását. A művelet hamissal tér vissza, mert ez nem lehetséges.
  - **+ setOpen(boolean): void:** Beállítja a lyuk állapotát. Ha paraméterként igaz logikai értéket kap, a lyukat kinyitja, egyébként becsukja.

8.1.7 IVisitable

- **Felelősség**

A Visitor minta egyik fele, ami a lépés sikerességét adja vissza. Ez a fél a sikeresség eldöntéséért felel.

- **Ősosztályok**

-

- **Metódusok**
  - ***+ visitByWorker(Worker, Worker): boolean:*** Egy dolgozó lépésének sikerességét adja vissza.
  - ***+ visitByCrate(Worker, Crate): boolean:*** Egy láda lépésének sikerességét adja vissza.
  - ***+ visitByLifeCrate(Worker, LifeCrate): boolean:*** Egy szívecskés láda lépésének sikerességét adja vissza.

8.1.8 IVisitor

- **Felelősség**

A Visitor minta másik fele, ami a lépés sikerességét adja vissza. Ez a fél a dinamikus típussal való visszahívásért felel.

- **Ősosztályok**

-

- **Metódusok**
  - ***+ visit(Worker, IVisitable): boolean:*** Dinamikus típussal való visszahívást tartalmazza, a visszatérési érték az átlépési lehetőséget fejezi ki.

8.1.9 LifeCrate

- **Felelősség**

Egy szívecske szimbólummal rendelkező ládát szimbolizál. Ez a láda egy hagyományos ládaként viselkedik, de ha leesik egy lyukba, akkor az őt leejtő dolgozó kap egy életet.

- **Ősosztályok**

Entity ⇨ Crate

- **Interfészek**

-

- **Attribútumok**
  - **-**

- **Metódusok**
  - **+ remove(Worker): void:** A szívecskés láda kikerül a raktárból, miközben a paraméterként átvett dolgozónak életet ad.

8.1.10 PlaceableItem

- **Felelősség**

Egy munkásnál lévő letehető dolgot szimbolizálja, ami lehet méz (HONEY) vagy olaj (OIL) de az is lehet, hogy nincs nála semmi (NOTHING).

8.1.11 Spawn

- **Felelősség**

A játéktér egy olyan padlóját képviseli, ami egy dolgozó kiindulási mezője. Minden dolgozó saját kiindulási mezővel rendelkezik, innen kezdik a játékot, és innen is folytatják életvesztés után, ha van még életük. Erre a mezőre kizárólag a hozzá tartozó dolgozó léphet, más entitás nem. Ha más dolgozó próbálna meg közvetetten rálépni (tehát úgy, hogy egy láda tolja őt), akkor a dolgozó életet veszít.

- **Ősosztályok**

Field ⇨ Floor

- **Interfészek**

-

- **Attribútumok**
  - **- owner: Worker:** A dolgozó, amelynek ez a kiindulási mezője.

- **Metódusok**
  - **+ visitByWorker(Worker, Worker): boolean:** Egy munkás kiindulási mezőre lép. Ez csak akkor lehetséges, ha a dolgozónak ez a kiindulási mezője vagy ha a dolgozót láncban tolják. Utóbbi esetben, ha a dolgozónak nem ez a kiindulási mezője, akkor ő nekiütközik a kiindulási mezőnek és életet veszít.
  - **+ isEmpty(): boolean:** Minden esetben hamissal tér vissza, így biztosítható a kiindulási mezőre való lépés helyes működése.
  - **+ placeSlipFactor(double): boolean:** Egy entitás próbálja megváltoztatni egy kiindulási hely tapadását. A művelet hamissal tér vissza, mert ez nem lehetséges.

8.1.12 Switch

- **Felelősség**

A játéktér egy olyan padlóját képviseli, melyen egy kapcsoló található. Amennyiben erre a kapcsolóra egy láda lép, egy vagy több lyuk nyitott állapotba kerül. Ha a láda lekerül a mezőről, a kapcsoló által kinyitott lyukak bezáródnak. Ha dolgozó lép a kapcsolóra, nem kapcsol.

- **Ősosztályok**

Field ⇨ Floor

- **Interfészek**

-

- **Attribútumok**
  - **- holes: List<Hole>:** Azon lyukak, melyek ehhez a kapcsolóhoz vannak rendelve.

- **Metódusok**
  - **+ visitByCrate(Worker, Crate): boolean:** Egy láda a kapcsolóra lép. Igazzal tér vissza, mivel egy láda mindig ráléphet egy kapcsolóra. Ilyenkor a kapcsolóhoz rendelt lyukak nyitott állapotba kerülnek.
  - **+ unsetEntity(): void:** A mezőn lévő entitás lekerül a mezőről, ezek után a mező üres lesz. Ilyenkor a kapcsolóhoz rendelt lyukak zárt állapotba kerülnek.
  - **+ placeSlipFactor(double): boolean:** Egy entitás próbálja megváltoztatni egy kapcsoló tapadását. A művelet hamissal tér vissza, mert ez nem lehetséges.

8.1.13 Target

- **Felelősség**

A játéktér egy olyan padlóját képviseli, ami egy előírt hely a ládák számára. Ha egy dolgozó erre a mezőre tol egy ládát, akkor pontot kap érte, azonban ha később valaki eltolja innen a ládát, az érte pontot kapó dolgozó elveszti a kapott pontot.

- **Ősosztályok**

Field ⇨ Floor

- **Interfészek**

-

- **Attribútumok**
  - **- whoPushed: Worker:** Az a dolgozó, amelyik pontot kapott egy láda ide tolásával.

- **Metódusok**
  - **+ visitByCrate(Worker, Crate): boolean:** Egy láda az előírt helyre lép. Igazzal tér vissza, mivel egy láda mindig ráléphet egy előírt helyre. Ilyenkor a láncot toló munkás pontot kap.
  - **+ unsetEntity(): void:** A mezőn lévő entitás lekerül a mezőről, ezek után a mező üres lesz. Ilyenkor a dolgozó, aki eredetileg a helyére tolta a ládát, pontot veszít.
  - **+ placeSlipFactor(double): boolean:** Egy entitás próbálja megváltoztatni egy előírt hely tapadását. A művelet hamissal tér vissza, mert ez nem lehetséges.
  - **+ unsetWhoPushed(): void:** Beállítja, hogy jelenleg nincs dolgozó aki ládát tolt volna az előírt helyre.
  - **+ setWhoPushed(Worker): void:** Beállítja, hogy melyik dolgozó tolta a ládát az előírt helyre.

8.1.14 Wall

- **Felelősség**

A játéktér egy olyan mezőjét képviseli, amire nem lehet lépni. Ha dolgozó próbálna meg közvetetten rálépni (tehát úgy, hogy egy láda tolja őt), akkor a dolgozó életet veszít.

- **Ősosztályok**

Field

- **Interfészek**

-

- **Attribútumok**
  - **-**

- **Metódusok**
  - **+ visitByWorker(Worker, Worker): boolean:** Egy dolgozó a falra lép. Hamissal tér vissza, mivel egy munkás sosem léphet rá egy falra.

- **+ visitByCrate(Worker, Crate): boolean:** Egy láda a falra lép. Hamissal tér vissza, mivel egy láda sosem léphet rá egy falra.

8.1.15 Warehouse

- **Felelősség**

A játékteret szimbolizálja, tartalmazza a benne lévő mezőket és entitásokat.

- **Ősosztályok**

Field

- **Interfészek**

-

- **Attribútumok**
  - **- fields: List<Field>:** A raktárban található mezők.
  - **- entities: List<Entity>:** A raktárban található entitások.

- **Metódusok**
  - **+ initialize(): void:** Inicializálja a játékteret, azzal felépíti a mezőket, és létrehozza az entitásokat.
  - **+ setField(Field, int, int): void:** Beállítja a pálya adott helyére a paraméterként kapott mezőt.
  - **+ addEntity(Entity, int, int): void:** Hozzáad egy entitást az adott pozíción levő mezőre.
  - **- workerDie(Worker): void:** Amikor egy dolgozó véglegesen meghalt, a raktár kiregisztrálja őt.
  - **- crateStuck(Crate): void:** Amikor egy láda beragad, azaz tovább már semmilyen irányban nem tolható el, a raktár regisztrálja neki ezt az állapotot.
  - **- update(): void:** A raktár frissíti minden benne található mező és entitás állapotát, valamint figyeli a játék végét is.

8.1.16 Worker

- **Felelősség**

Egy dolgozót szimbolizál. Nyilvántartja a dolgozó pozícióját, életét, valamint pontszámát.

- **Ősosztályok**

Entity

- **Interfészek**

-

- **Attribútumok**
  - **- health: int:** A dolgozó élete. Ha eléri a nullát, a dolgozó meghal és kikerül a játékból.
  - **- points: int:** A dolgozó pontszáma. Minden egyes előírt helyre tolt ládával egyel nő, ám amennyiben azok a ládák később elmozdulnak az előírt helyekről, a pontok elvesznek.
  - **- power: double:** A munkás jelenlegi ereje, mely egy tolási kísérlet során entitásonként csökken.
  - **- orgPower: double:** A munkás eredeti ereje.
  - **- item: PlaceableItem:** A dolgozónál lévő dolog.

- **- spawnpoint: Spawn:** A dolgozóhoz rendelt kiindulási mező.

- **Metódusok**
  - **+ push(Worker, Entity, Direction): boolean:** A dolgozó eltol egy entitást a megadott irányban lévő szomszédos mezőre. Igazzal tér vissza, ha sikerült eltolni, egyébként hamissal.
  - **+ pushByWorker(Worker, Worker, Direction): boolean:** Egy dolgozó tolja a dolgozót a megadott irányban lévő szomszédos mezőre, így a dolgozó oda lép. Hamissal tér vissza, mivel dolgozó nem tolhat közvetlenül másik dolgozót.
  - **+ pushByCrate(Worker, Crate, Direction): boolean:** Egy láda tolja a dolgozót a megadott irányban lévő szomszédos mezőre, így a dolgozó oda lép. Igazzal tér vissza, ha sikerült lépni, egyébként hamissal.
  - **+ move(Direction): void:** A játékos lépteti a dolgozót a megadott irányban.
  - **+ loseHealth(): void:** A dolgozó egy életet veszít.
  - **+ gainHealth(): void:** A dolgozó egy életet kap.
  - **+ losePoint(): void:** A dolgozó egy pontot veszít.
  - **+ gainPoint(): void:** A dolgozó egy pontot kap.
  - **+ reSpawn(): void:** A dolgozó átkerül jelenlegi mezőjéről a számára kijelölt kiinduló mezőre.
  - **+ die(): void:** A dolgozó meghal és kikerül a játékból.
  - **+ consumePower(double): double:** A dolgozó próbál eltolni egy entitást. A művelet a munkás entitáshoz viszonyított erejét adja vissza, melynek negatív volta jelzi, ha a munkásnak nincs elég ereje a toláshoz.
  - **+ placeItem(): void:** A dolgozó megpróbálja letenni a nála lévő dolgot arra a mezőre, amin jelenleg tartózkodik.
  - **+ visit(Worker, IVisitable): boolean:** A dolgozó egy vizitálható objektumra (mezőre) lép. Igazzal tér vissza amennyiben sikerült a lépés, egyébként hamissal.

8.2 A tesztek részletes tervei, leírásuk a teszt nyelvén

8.2.1 A tesztek rövid leírása

Minden tesztesetünk egy-egy alap lépést próbál tesztelni, abból az elvből kiindulva, hogy ha minden elem jó akkor az egész is jó lesz.

Felépítést tekintve minden teszt eset a következő struktúrát követi:

Betöltünk egy előre elkészített pályát, amin végrehajtunk egy esemény szekvenciát, ahol az esemény vagy játékos inputot vagy belső eseményt (pl.: eszköz adást) vagy adatlekérést jelöl.

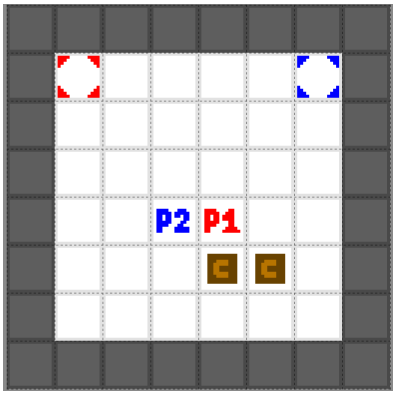
A modell alapján, megadjuk az elvárt viselkedést. Ehhez hasonlítjuk a tényleges eredményt, ha a kettő egyezik akkor sikeres a teszt.

Megjegyzés: LEVEL_SUCCESS jelentés a sikeres pályabetöltés.

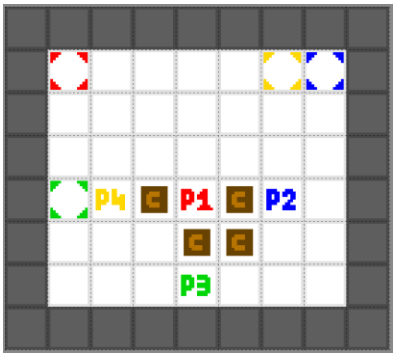
8.2.1.1 A tesztpályák felépítése

Az egyszerűség kedvéért felsoroljuk a tesztesetekben használt pályák grafikus képeit, ezáltal könnyedén el lehet igazodni az egyes tesztesetek lépéseiben.

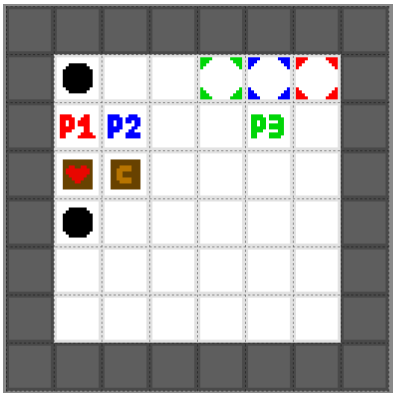
8.2.1.2 testlevel1



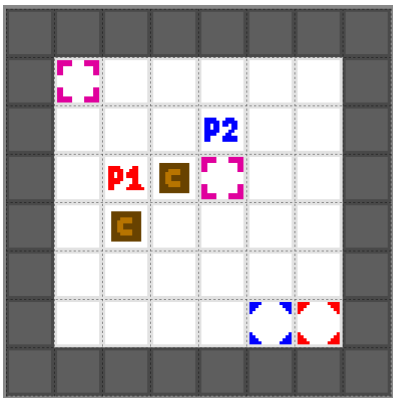
8.2.1.3 testlevel2



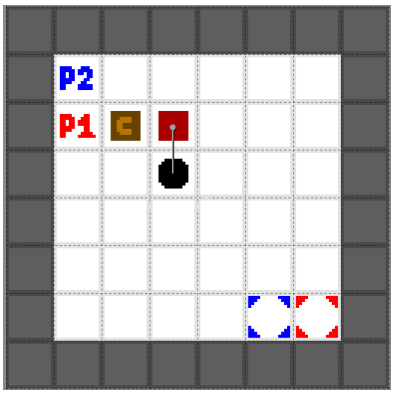
8.2.1.4 testlevel3



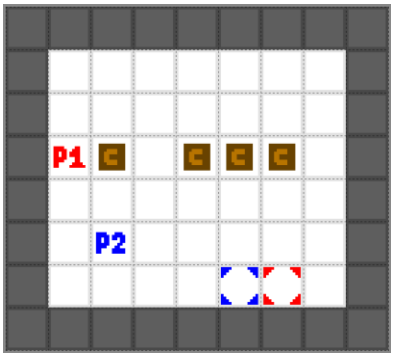
8.2.1.5 testlevel4



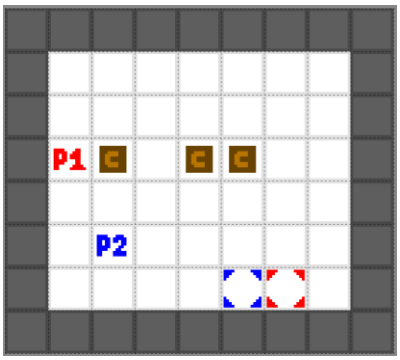
8.2.1.6 testlevel5



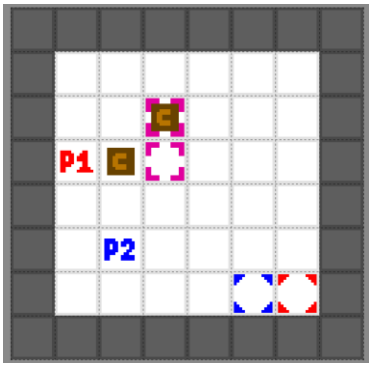
8.2.1.7 testlevel6



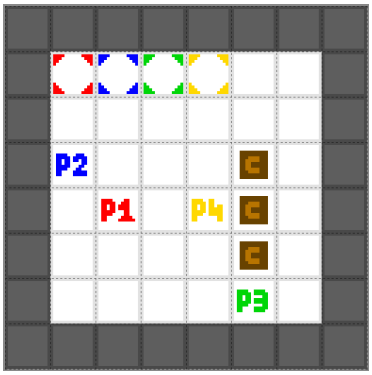
8.2.1.8 testlevel7



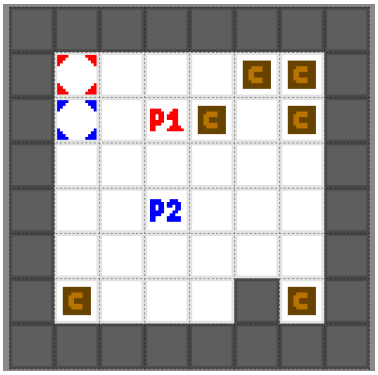
8.2.1.9 testlevel8



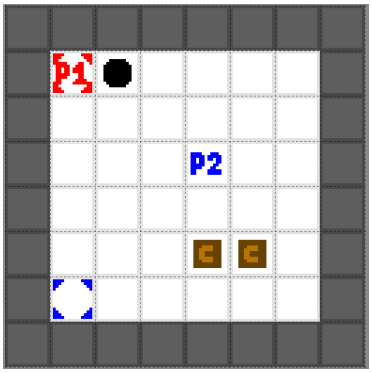
8.2.1.10 testlevel9



8.2.1.11 testlevel10



8.2.1.12 testlevel11



8.2.2 Move worker on empty field

- **Leírás**

Egy munkás léptetésének tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Az első lépés a teszthez szükséges pálya betöltése, melyen az objektumok előre beállított - a tesztnek kedvező - helyen tartózkodnak. Az 1-es azonosítóval rendelkező dolgozó mellett jobb oldalt egy üres padló van, melyre minden esetben rá kell tudnia lépni. A teszt ráléptetei a dolgozót erre a padlóra, majd ellenőrzi a lépés sikerességét.

- **Bemenet**

LEVEL "testlevel1.json"
STEP 1 R

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel1.json"
STEP_OK 1

8.2.3 Worker pushes crate to empty field

- **Leírás**

Egy láda eltolásának tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A pálya betöltését követően az 1-es számú dolgozóval megpróbálunk jobbra lépni. Ebben az irányban egy láda van, így a dolgozó eltolja azt. A kimenet mutatja a tolás sikerességét.

- **Bemenet**

LEVEL "testlevel1.json"

STEP 1 D

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel1.json"

STEP_OK 1

8.2.4 Worker pushes other worker directly

- **Leírás**

Egy dolgozó dolgozót közvetlen tolásának tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A pálya betöltését követően az 1-es számú dolgozóval megpróbálunk balra lépni. Mivel ezen a mezőn egy másik dolgozó tartózkodik, akit közvetlenül nem tud eltolni, ezért a lépés sikertelen.

- **Bemenet**

LEVEL "testlevel1.json"

STEP 1 L

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel1.json"

STEP_FAIL 1

8.2.5 Worker pushes other worker in chain

- **Leírás**

Egy dolgozó dolgozót láncban tolásának tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A pálya betöltését követően az 1-es számú dolgozóval megpróbálunk jobbra lépni. Ebben az irányban egy lánc van, a dolgozó melletti végén egy ládával, így egy dolgozó, amely a láncban van, közvetetten eltolhatóvá válik. A kimenet mutatja a tolás sikerességét.

- **Bemenet**

LEVEL "testlevel2.json"

STEP 1 R

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel2.json"

STEP_OK 1

8.2.6 Worker smashed by wall

- **Leírás**

Egy dolgozó fal általi összenyomódásának tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A pályát betöltjük, lekérjük a 3-as sorszámú játékos életeinek a számát, majd a dolgozóját az 1-es sorszámú dolgozóval egy falba toljuk. Az életei számának ismételt lekérdezésével megbizonyosodhatunk arról, hogy a játékos elveszített egy életet.

- **Bemenet**

LEVEL "testlevel2.json"

PLAYERINFO 3 HP

!STEP 1 D

PLAYERINFO 3 HP

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel2.json"

PLAYER_HP 3 3

STEP_OK 1

PLAYER_HP 3 2

8.2.7 Worker smashed by spawn

- **Leírás**

Egy dolgozó kiindulási hely általi összenyomódásának tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A pályát betöltjük, lekérjük a 4-es sorszámú játékos életeinek a számát, majd a dolgozóját az 1-es sorszámú dolgozóval egy idegen kiindulási mezőre toljuk. Az életei számának ismételt lekérdezésével megbizonyosodhatunk arról, hogy a játékos elveszített egy életet.

- **Bemenet**

LEVEL "testlevel2.json"

PLAYERINFO 4 HP

!STEP 1 L

PLAYERINFO 4 HP

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel2.json"

PLAYER_HP 4 3

STEP_OK 1

PLAYER_HP 4 2

8.2.8 Worker falls into hole

- **Leírás**

Egy dolgozó lyukba esésének tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Betöltjük az előkészített tesztpályát, lekérdezzük az 1-es sorszámú játékos életeinek a számát illetve a mellette található lyuk állapotát, majd a lyukra léptetjük a játékoshoz tartozó dolgozót. A léptetést követően ismét lekérdezzük az életek számát, amely eggyel csökkent, mivel a dolgozó beleesett a nyitott állapotú lyukba.

- **Bemenet**

LEVEL "testlevel3.json"

PLAYERINFO 1 HP

HOLEINFO 1,1

!STEP 1 U

PLAYERINFO 1 HP

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel3.json"

PLAYER_HP 1 3

OPEN 1,1

STEP_OK 1

PLAYER_HP 1 2

8.2.9 Worker pushes lifecrate into hole

- **Leírás**

Egy szívecskés ládát lyukba tolásának tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Első lépésben betöltjük a pályát. Ezt követően lekérdezzük az 1-es sorszámú játékos életeinek a számát, továbbá a dolgozójától két mezőre található lyuk állapotát. A lyuk és a játékos között egy szívecskés láda van, amit a dolgozó léptetésével a lyukba tolunk. A kimenet mutatja, hogy a lyuk nyitott állapotban volt, a tolást követően pedig a játékos egy életet kapott, tehát a láda lyukba tolása és annak leesése sikeres volt.

- **Bemenet**

LEVEL "testlevel3.json"

PLAYERINFO 1 HP

HOLEINFO 1,4

!STEP 1 D

PLAYERINFO 1 HP

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel3.json"

PLAYER_HP 1 3

OPEN 1,4

STEP_OK 1

PLAYER_HP 1 4

8.2.10 Target activation test

- **Leírás**

Egy láda előírt helyre tolásának ellenőrzése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Szokásos módon betöltjük a tesztpályát, majd lekérjük az előírt helyen lévő ládák számát és az 1-es sorszámú játékos pontjait. A játékos dolgozójával jobbra lépünk, amellyel a mellette lévő ládát egy előírt helyre toljuk. Az ismételt lekérdezéssel megbizonyosodhatunk a teszt sikerességéről, végezetül pedig fájlba mentjük a teszt állapotát, amely egy későbbi tesztesetben kerül felhasználásra.

- **Bemenet**

LEVEL "testlevel4.json"

PLAYERINFO 1 PTS

CNT CR

!STEP 1 R

PLAYERINFO 1 PTS

CNT CR

SAVE "TEST9_P1_SCORED.ser"

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel4.json"

PLAYER_PTS 1 0

CNT_CR 0

STEP_OK 1

PLAYER_PTS 1 1

CNT_CR 1

SAVE_SUCCESS "TEST9_P1_SCORED.ser"

8.2.11 Target deactivation test

- **Leírás**

Egy láda előírt helyről eltolásának ellenőrzése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Betöltjük a 8.2.9-es tesztesetben elmentett állapotot és lekérdezzük az előírt helyen lévő ládák számát illetve az 1-es sorszámú játékos pontjait. Ezt követően a 2-es sorszámú dolgozót lefele léptetjük, aminek keretében az 1-es sorszámú játékos dolgozója által odatolt láda lekerül az előírt helyről, így a játékos elveszíti a tolásért kapott pontot. Az újbóli lekérdezés mutatja a teszt sikerességét a kimeneten.

- **Bemenet**

LOAD "TEST9_P1_SCORED.ser"

PLAYERINFO 1 PTS

CNT CR

!STEP 2 D

PLAYERINFO 1 PTS

CNT CR

- **Elvárt kimenet**

LOAD_SUCCESS "TEST9_P1_SCORED.ser"

PLAYER_PTS 1 1

CNT_CR 1

STEP_OK 2

PLAYER_PTS 1 0

CNT_CR 0

8.2.12 Switch activation test

- **Leírás**

Egy kapcsoló bekapcsolásának tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Az 5-ös számú tesztpálya betöltését követően lekérdezzük a 3,3-as pozíciójú lyuk állapotát, majd az 1-es sorszámú dolgozót jobbra léptetjük, ahol egy láda található. A dolgozó a ládát a láda mellett található kapcsolóra tolja, ezzel aktiválva azt. A kapcsoló kinyitja a hozzárendelt lyukakat, beleértve a 3,3-ast, aminek kinyílásáról a státuszának ismételt lekérdezésével megbizonyosodhatunk. Végezetül pedig fájlba mentjük a teszt állapotát, amely egy későbbi tesztesetben kerül felhasználásra.

- **Bemenet**

LEVEL "testlevel5.json"

HOLEINFO 3,3

!STEP 1 R

HOLEINFO 3,3

SAVE "TEST11_HOLE_ACT.ser"

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel5.json"

CLOSED 3,3

STEP_OK 1

OPEN 3,3

SAVE_SUCCESS "TEST11_HOLE_ACT.ser"

8.2.13 Switch deactivation test

- **Leírás**

Egy kapcsoló kikapcsolásának tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Betöltjük a 8.2.11-es tesztesetben elmentett állapotot, majd lekérdezzük a 3,3-as pozíciójú lyuk állapotát. Az 1-es sorszámú dolgozót jobbra léptetjük, ahol egy kapcsoló, rajta pedig egy láda található, amely eltolható. Azáltal, hogy a ládát letoljuk a kapcsolóról és helyére a dolgozó kerül, a kapcsoló kikapcsol és bezárja a hozzárendelt lyukakat, beleértve a 3,3-ast, aminek bezáródásáról a státuszának ismételt lekérdezésével megbizonyosodhatunk.

- **Bemenet**

LOAD "TEST11_HOLE_ACT.ser"

HOLEINFO 3,3

!STEP 1 R

HOLEINFO 3,3

- **Elvárt kimenet**

LOAD_SUCCESS "TEST11_HOLE_ACT.ser"

OPEN 3,3

STEP_OK 1

CLOSED 3,3

8.2.14 Field modificator test - oil

- **Leírás**

Egy dolgozó olajat lehelyezésének tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A hatos számú tesztpálya, úgy van előkészítve, hogy az 1-es számú játékostól jobbra egy láda, üres padló, majd egy lánc található. A lánc és az említett láda súlya túl nagy alap esetben, hogy egy munkás eltolja, de ha a kiegészítő láda eltolását olaj segítségével megkönnyítjük, akkor már eltolható a teljes lánc.

Ennek megfelelően a pályát betöltjük majd jobbra léptetjük a dolgozó, ami nem sikerül, majd újra betöltjük a pályát elhelyezzük az olajat a láda és a lánc közti üres padlóra, ami után a kezdeti helyre visszalépve végrehajtjuk a tolást, ami most sikeres.

- **Bemenet**

LEVEL "testlevel6.json"

!STEP 1 R

!STEP 1 R

LEVEL "testlevel6.json"

!STEP 1 U

!STEP 1 R

!STEP 1 R

!STEP 1 D

GIVE 1 O

!PLACE 1

!STEP 1 U

!STEP 1 L

!STEP 1 L

!STEP 1 D

!STEP 1 R

!STEP 1 R

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel6.json"

STEP_OK 1

STEP_FAIL 1

LEVEL_SUCCESS "testlevel6.json"

STEP_OK 1

STEP_OK 1

STEP_OK 1

STEP_OK 1

PLACE_OK 1

STEP_OK 1

STEP_OK 1

STEP_OK 1

STEP_OK 1

STEP_OK 1

STEP_OK 1

8.2.15 Field modificator test - honey

- **Leírás**

Egy dolgozó méz lehelyezésének tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A hetes számú tesztpálya, úgy van előkészítve, hogy az 1-es számú játékostól jobbra egy láda, üres padló, majd egy lánc található. A lánc és az említett láda súlya pont akkora alap esetben, hogy egy munkás eltolja, de ha a kiegészítő láda eltolását méz segítségével megnehezítjük, akkor már nem eltolható a teljes lánc.

Ennek megfelelően a pályát betöltjük majd jobbra léptetjük a dolgozó, ami pont sikerül, majd újra betöltjük a pályát elhelyezzük a mézet a láda és a lánc közti üres padlóra, ami után a kezdeti helyre visszalépve végrehajtjuk a tolást, ami most sikertelen.

- **Bemenet**

LEVEL "testlevel7.json"

!STEP 1 R

!STEP 1 R

LEVEL "testlevel7.json"

!STEP 1 U

!STEP 1 R

!STEP 1 R

!STEP 1 D

GIVE 1 H

!PLACE 1

!STEP 1 U

!STEP 1 L

!STEP 1 L

!STEP 1 D

!STEP 1 R

!STEP 1 R

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel7.json"

STEP_OK 1

STEP_OK 1

LEVEL_SUCCESS "testlevel7.json"

STEP_OK 1

STEP_OK 1

STEP_OK 1

STEP_OK 1

PLACE_OK 1

STEP_OK 1

STEP_OK 1

STEP_OK 1

STEP_OK 1

STEP_OK 1

STEP_FAIL 1

8.2.16 Locking mechanism of simultaneously pushed conflicting chains

- **Leírás**

Egyszerre történő, egymást kizáró láncok viselkedésének tesztje.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A tesztpálya két független ütközési helyet tartalmaz. Az 1-es és 2-es dolgozók azonos helyre próbálnak lépni, illetve ettől függetlenül a 3-as és 4-es játékos pedig olyan láncot szeretnének tolni, melynek van közös tagja. A test során azonos időperiódusban? mind a 4 játékos a hozzá tartozó ütközési hely irányába lép, ekkor a 2-2 lépés közül 1-1 sikerül (az első) és 1-1 nem (a második).

- **Bemenet**

LEVEL "testlevel9.json"

STEP 1 U

STEP 2 R

STEP 3 U

!STEP 4 R

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel9.json"

STEP_OK 1

STEP_FAIL 2

STEP_OK 3

STEP_FAIL 4

8.2.17 Game ends because there is a crate on all target

- **Leírás**

A célterületek feltöltése miatti játék vég észlelésének tesztje.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A tesztpálya olyan, hogy 1 kivételével minden célterületén láda található, más játék végét jelentő feltétel sem teljesül. Betöljük a tesztpályát, és az 1-es munkással eltolunk egy ládát úgy, hogy az utolsó célterületre kerüljön, várjuk hogy jó okkal legyen vége.

- **Bemenet**

LEVEL "testlevel8.json"

!STEP 1 R

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel8.json"

STEP_OK 1

LEVEL_END TR

8.2.18 Game ends because there are no moveable crates

- **Leírás**

Beragadás miatti játék vég észlelésének tesztje.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A tesztpálya olyan, hogy egy kivételével az rajta található ládák nem mozgatható, más játék végét jelentő feltétel sem teljesül.

Betöljük a tesztpályát, és az 1-es munkással eltoljuk a ládát úgy, hogy beragadjon, várjuk hogy jó okkal legyen vége.

A ládák különféle edge-case konfigurációkban vannak, amelyek lefogják az összes esetet ami miatt egy láda többé nem mozgathatóvá válik.

- **Bemenet**

LEVEL "testlevel10.json"

!STEP 1 R

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel10.json"

STEP_OK 1

LEVEL_END CR

8.2.19 Game ends because all the workers died except one

- **Leírás**

Játékos elimináció miatti játék vég észlelésének tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A tesztpálya olyan, hogy 2 játékos található rajta, melyek közül az 1-esnek már csak két élete van, játék végét jelentő feltétel nem teljesül.

Betöljük a tesztpályát, és az 1-es munkással kétszer lyukba lépünk, várjuk hogy jó okkal legyen vége.

- **Bemenet**

LEVEL "testlevel11.json"

HOLEINFO 2,1

PLAYERINFO 1 HP

!STEP 1 R

PLAYERINFO 1 HP

!STEP 1 R

PLAYERINFO 1 HP

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel11.json"

PLAYER_HP 1 2

OPEN X,X

STEP_OK 1

PLAYER_HP 1 1

STEP_OK 1

PLAYER_HP 1 0

LEVEL_END P

8.2.20 Player gains an item

- **Leírás**

Tárgy (pl.: olaj, méz) szerzésének tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszteset során egy dolgozónak oda adjuk sorban az összes lehetséges itemet, majd ellenőrizzük, hogy megkapta-e.

- **Bemenet**

LEVEL "testlevel11.json"

GIVE 1 N

PLAYERINFO 1 ITEM

GIVE 1 O

PLAYERINFO 1 ITEM

GIVE 1 H

PLAYERINFO 1 ITEM

GIVE 1 N

PLAYERINFO 1 ITEM

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel11.json"

PLAYER_ITEM 1 N

PLAYER_ITEM 1 O

PLAYER_ITEM 1 H

PLAYER_ITEM 1 N

8.2.21 Worker puts down item

- **Leírás**

Tárgy elhelyezésének tesztje.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszteset során egy dolgozó minden lehetséges item mellett kísérletet tesz a tárgy lehelyezésére.

- **Bemenet**

LEVEL "testlevel11.json"

GIVE 1 O

!PLACE 1

GIVE 1 H

!PLACE 1

GIVE 1 N

!PLACE 1

- **Elvárt kimenet**

LEVEL_SUCCESS "testlevel11.json"

PLACE_OK 1

PLACE_OK 1

PLACE_FAIL 1

8.3 A tesztelést támogató programok tervei

8.3.1 Tesztesetek elkészítése

Az előírt szintaxist használva szövegesen, külön *.txt* fájlokba kell elkészíteni az egyes tesztesetekhez tartozó bemenetet és elvárt kimenetet. A teszteset nevét a fájok neve határozza meg, a program is erre hivatkozik a későbbiek során.

Egy tesztesethez tehát két fájl tartozik, az egyik a bemeneti parancsokat tartalmazza soronként és *tst_in<name>.txt* a neve, a másik pedig az elvárt kimenetet, és *tst_out<name>.txt* a neve. A *<name>* tag a teszteset nevét tartalmazza, és kizárólag a standard 128 karakteres ASCII betűiből állhat. A program később ezen a néven fog hivatkozni a tesztesetre. Amennyiben valamely követelmény nem teljesül (pl. bemeneti fájlhoz nincsen kimeneti) az adott teszteset automatikusan ignorálásra kerül.

Egy fájlon belül a bemeneti parancsok vagy kimeneti eredmények soronként szerepelnek. Az üres sor ignorálásra kerül, a pontosvesszővel pedig kommentezés lehetséges (pontosvessző után az adott tartalom a sor végéig ignorálásra kerül).

A tesztekhez tartozó pályá(ka)t a Tiled nevű programmal kell elkészíteni a mellékelt segédlet alapján.

8.3.2 A tesztek kiértékelése

A program karakteres menüben jeleníti meg az elkészített teszteseteket, a nevükkel hivatkozva rájuk. A program a menüben a tesztekhez immár egy azonosítót (egész szám) is társít, melynek megadásával lehet egy tesztesetet lefuttatni. A program a bemeneti parancsok lefutása után összehasonlítja az eredményt az elvárt kimenettel, majd szövegesen jelzi a teszteset sikerességét vagy bukását, utóbbi esetben megadja a bukást okozó hibás kimenetet, illetve a hozzá tartozó elvárt eredményt is. Lehetőség van egy speciális menüpontban az összes teszteset egymás után történő futtatására. Ezek után a konzolon szintén szöveges formában jelennek meg a tesztek eredményei, többek között az összes teszt és a sikeres tesztek aránya.

Előfordulhatnak olyan tesztesetek, melyek egymásra hivatkoznak, például egy teszteset egy másik által kimentet állapotot használ. Ilyen esetben a tesztesetek ezt figyelmen kívül hagyó sorrendben történő futtatása automatikusan hibához vezet. Továbbá szintén automatikus hibát okoz egy nem létező pálya vagy állapot betöltésének kísérlete, általánosabban a *LEVEL/LOAD* parancsokkal hívott műveletek hibája.

8.4 *Napló*

Kezdet	Időtartam	Résztvevők	Leírás
2018.04.07. 10:00	1,5 óra	SZAKÁLLAS	Osztályok terveinek (8.1) elkészítése
2018.04.07. 12:00	1 óra	LAKATOS	Osztályok terveinek elkészítése, kiegészítése (8.1)
2018.04.07. 19:00	1,5 óra	SZAKÁLLAS	Tesztesetek (8.2) leírásainak elkészítése
2018.04.07. 20:00	2 óra	LENKEFI	Tesztesetek (8.2) leírásainak elkészítése, tesztesetekhez tartozó pályák elkészítése
2018.04.08. 16:00	3 óra	JANI	Tesztesetek (8.2) leírásainak elkészítése, kiegészítése, felülvizsgálata
2018.04.08. 16:00	2 óra	CSANÁDY	Tesztesetek (8.2) leírásainak elkészítése, kiegészítése
2018.04.08. 20:00	0,5 óra	LAKATOS	A tesztelést támogató programok terveinek elkészítése (8.3)
2018.04.08. 22:00	1 óra	JANI	Tesztesetek kiegészítése (8.2)
2018.04.08. 22:00	1 óra	LENKEFI	Tesztesetek kiegészítése (8.2)
2018.04.08. 23:00	1 óra	CSANÁDY	Dokumentum átnézése, konzisztencia vizsgálata
2018.04.09. 00:30	0,5 óra	LAKATOS	Dokumentum véglegesítése

8. Prototípus beadása

8.1 Fordítási és futtatási útmutató

8.1.1 Fájllista

Fájl neve	Méret (kB)	Keletkezés ideje	Tartalom
Main.java	4	2018.04.22.	Main osztály, a program belépési pontja.
Crate.java	2	2018.04.22.	Crate osztály.
Direction.java	1	2018.04.22.	Direction típus.
Entity.java	5	2018.04.22.	Entity osztály.

Field.java	4	2018.04.22.	Field osztály.
Floor.java	2	2018.04.22.	Floor osztály.
Hole.java	3	2018.04.22.	Hole osztály.
IVisitable.java	2	2018.04.22.	IVisitable interfész.
IVisitor.java	1	2018.04.22.	IVisitor interfész.
LevelFormatException.java	1	2018.04.22.	LevelFormatException osztály.
LifeCrate.java	1	2018.04.22.	LifeCrate osztály.
PlaceableItem.java	1	2018.04.22.	PlaceableItem típus.
Process.java	1	2018.04.22.	Process interfész.
Spawn.java	2	2018.04.22.	Spawn osztály.
StepHoleProcess.java	1	2018.04.22.	StepHoleProcess segédosztály.
StepProcess.java	1	2018.04.22.	StepProcess segédosztály.
StepWallProcess.java	1	2018.04.22.	StepWallProcess segédosztály.
Switch.java	2	2018.04.22.	Switch osztály.
Target.java	2	2018.04.22.	Target osztály.
Wall.java	2	2018.04.22.	Wall osztály.
Warehouse.java	12	2018.04.22.	Warehouse osztály.
Worker.java	6	2018.04.22.	Worker osztály.
Command.java	1	2018.04.22.	Command interfész.
InputLanguageException.java	1	2018.04.22.	InputLanguageException osztály.
Test.java	3	2018.04.22.	Test segédosztály.
TestEnvironment.java	1	2018.04.22.	TestEnvironment segédosztály.
TestExecutionException.java	1	2018.04.22.	TestExecutionException osztály.
TestReader.java	4	2018.04.22.	TestReader segédosztály.
CNTCommand.java	2	2018.04.22.	CNTCommand segédosztály.
GiveCommand.java	1	2018.04.22.	GiveCommand segédosztály.
HoleInfoCommand.java	1	2018.04.22.	HoleInfoCommand segédosztály.
LevelCommand.java	1	2018.04.22.	LevelCommand segédosztály.

LevelEndCommand.java	1	2018.04.22.	LevelEndCommand segédosztály.
LoadCommand.java	2	2018.04.22.	LoadCommand segédosztály.
PlaceCommand.java	1	2018.04.22.	PlaceCommand segédosztály.
PlayerInfoCommand.java	2	2018.04.22.	PlayerInfoCommand segédosztály.
SaveCommand.java	2	2018.04.22.	SaveCommand segédosztály.
StepCommand.java	2	2018.04.22.	StepCommand segédosztály.

8.1.2 Fordítás

8.1.3 Futtatás

8.2 Tesztek jegyzőkönyvei

8.2.1 Move worker on empty field

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.2 Worker pushes crate to empty field

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.3 Worker pushes other worker directly

Tesztelő neve	Csanády Máté

Teszt időpontja	2018.04.22

8.2.4 Worker pushes other worker in chain

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.5 Worker smashed by wall

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.6 Worker smashed by spawn

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.7 Worker falls into hole

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.8 Worker pushes lifecrate into hole

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.9 Target activation test

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.10 Target deactivation test

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.11 Switch activation test

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.12 Switch deactivation test

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.13 Field modificator test - oil

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.14 Field modificator test - honey

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.15 Locking mechanism of simultaneously pushed conflicting chains

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.16 Game ends because there is a crate on all targets

Tesztelő neve	Lakatos Dániel
Teszt időpontja	2018.04.23

Tesztelő neve	Jani Balázs
Teszt időpontja	2018.04.22
Teszt eredménye	Nem szerepel a kimeneten a játék végét jelző parancs.
Lehetséges hibaok	Nincsen lekezelve, ha egy láda egyből az előírt helyen kezd. Nincsen lekezelve az az eset, hogy nincs egyetlen előírt hely sem.
Változtatások	Az előírt helyen kezdő ládák beleszámolása a „helyére tolt” ládákba, továbbá döntési feltétel módosítása, hogy kezeljük azt, amikor nincs egyetlen előírt hely sem.

8.2.17 Game ends because there are no moveable crates

Tesztelő neve	Lakatos Dániel

Teszt időpontja	2018.04.23

Tesztelő neve	Jani Balázs
Teszt időpontja	2018.04.22
Teszt eredménye	Nem szerepel a kimeneten a játék végét jelző parancs.
Lehetséges hibaok	A ládák beragadásának vizsgálata később történik, mint a játék vége vizsgálat.
Változtatások	Láda-beragadás vizsgálat minden lépés után.

8.2.18 Game ends because all the workers died except one

Tesztelő neve	Lakatos Dániel
Teszt időpontja	2018.04.23

Tesztelő neve	Jani Balázs
Teszt időpontja	2018.04.22
Teszt eredménye	Nem csökken a dolgozó élete, ha lyukba esik, így meg sem hal.
Lehetséges hibaok	A dolgozó rosszul van áthelyezve a kiindulási mezejére életvesztés után.
Változtatások	Visitor felvétele a lyukra lépéshez, ami egy másik process-t indít ami nem állítja a dolgozó helyzetét, az megmarad a lyuk felelősségének, pontosabban az életvesztés során hívott respawn-nak.

8.2.19 Player gains an item

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.2.20 Worker puts down item

Tesztelő neve	Csanády Máté
Teszt időpontja	2018.04.22

8.3 Értékelés

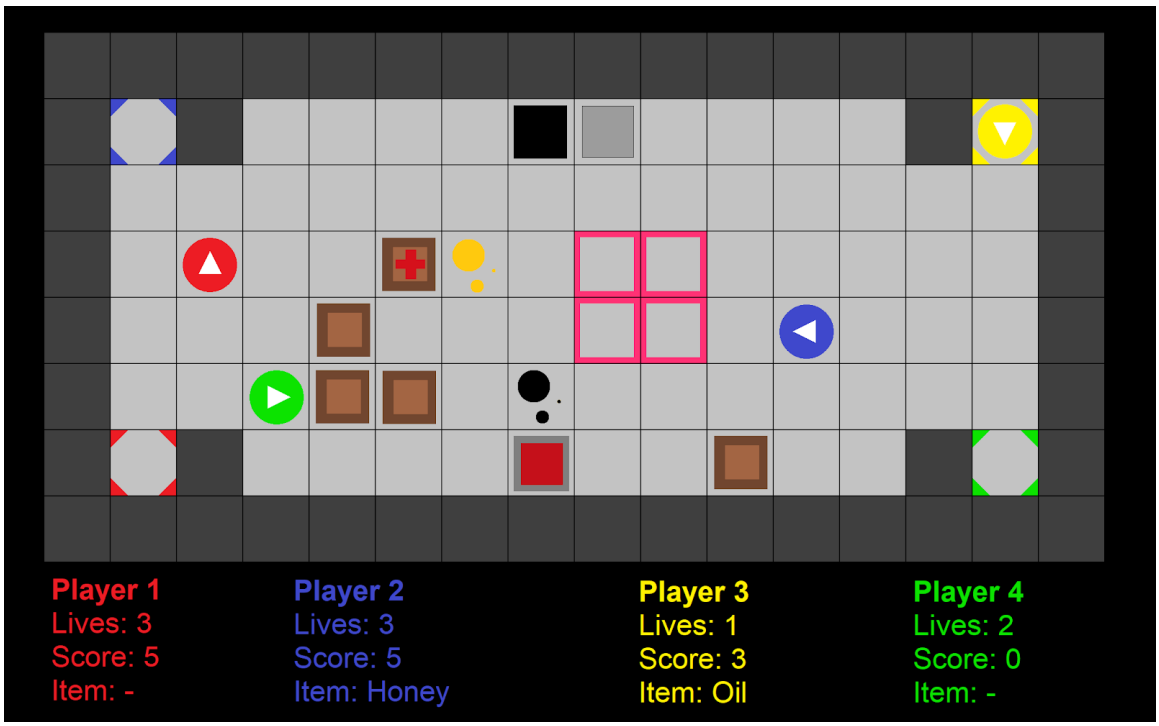
Tag neve	Munka százalékban
Csanády Máté	17.22%
Jani Balázs	20.05%
Lenkefi Péter	21.08%
Lakatos Dániel	21.34%
Szakállas Gergely	20.30%

8.4 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.04.21 16:00	7 óra	JANI	Tesztparancsok implementálása, tesztelés.
2018.04.22 14:00	6 óra	LAKATOS	A segédprogramok összehangolása, osztályok átalakítása a legújabb dizájn szerint, tesztelés és a hibák javítása.
2018.04.21 19:00	4 óra	SZAKÁLLAS	Tesztparancsok implementálása.
2018.04.22 10:00	8 óra	CSANÁDY	A menü és a tesztösszehasonlító elkészítése, tesztelés.
2018.04.21 16:00	10 óra	LENKEFI	A segédprogramok elkészítése és összekapcsolása a prototípussal.

9. Grafikus felület specifikációja

9.1 A grafikus interfész



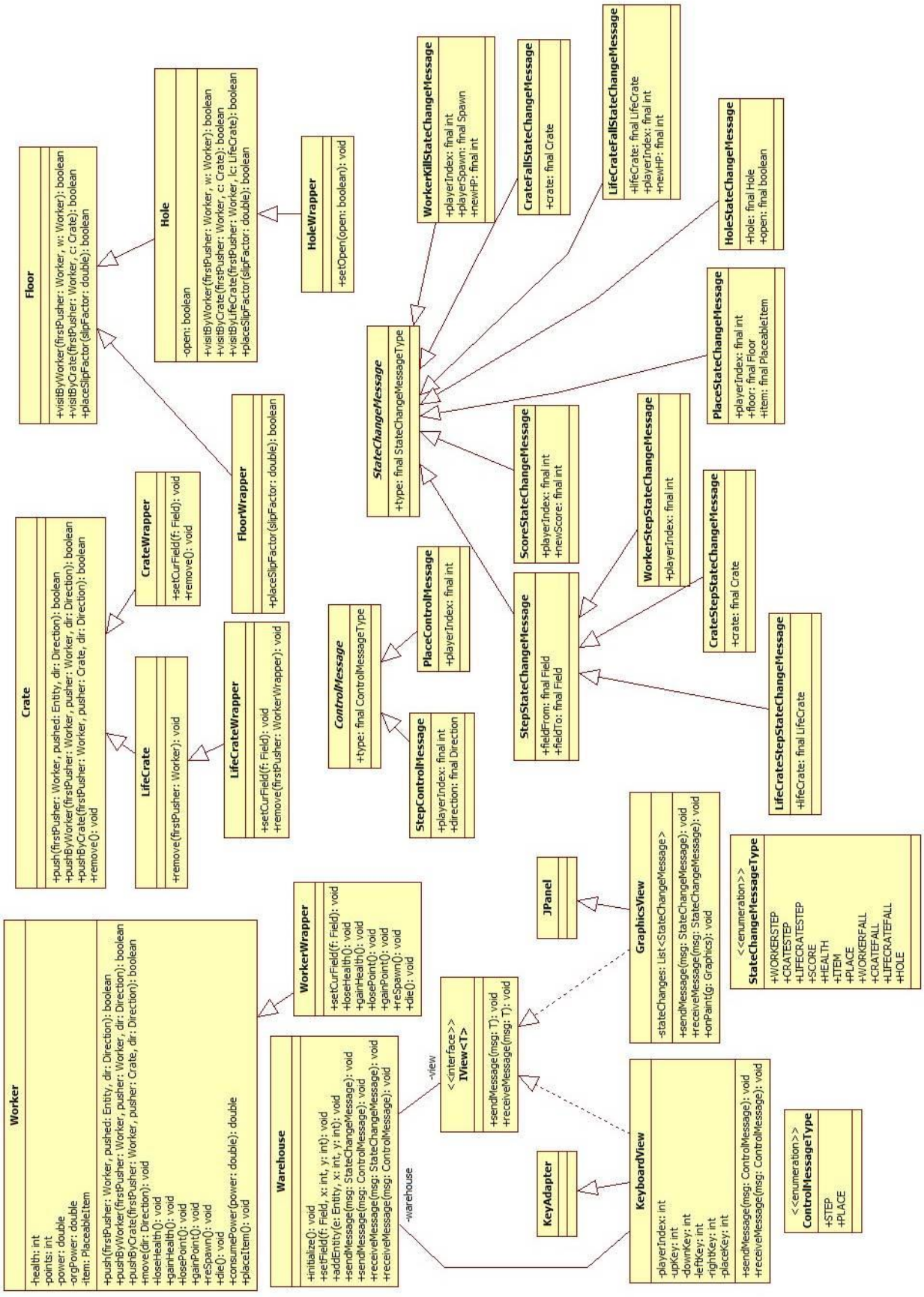
	Kék dolgozó, a nyíl irányába néz		Kijelölt hely
	Üres padló		Padló, melyre mézet raktak
	Fal		Padló, melyre olajat raktak
	A kék dolgozó kiindulási helye		Lyuk kapcsolója
	Láda		Zárt lyuk
	Szívecskés láda		Nyitott lyuk

9.2 A grafikus rendszer architektúrája

9.2.1 A felület működési elve

Egy push-alapú értesítési rendszer kialakítását tűztük ki célul: Minden külső elemet érintő művelet esemény értesítést küld a megfelelő komponensnek.
A **bemeneti nézetek** (billentyűzet gombjai) a megfelelő felhasználói bemenetekre felszólítják a modellt bizonyos feladatok/akciók végrehajtására (például: első játékos lépjen balra). A nézet elabsztrahálja a bemenet módját. Nem billentyűnyomás üzeneteket küld, hanem már - a modell számára értelmezhető - parancsokat, így egyszerűen kicserélhető egy billentyűzet alapú irányítás egy speciális játékvezérlőre, hang alapú irányításra, stb.
A **kimeneti nézetek** hasonlóképp tesznek, a változott állapotokról csupán nyers adatot biztosít a modell, annak értelmezése már a nézet feladata, jelen esetben ez egy grafikus megjelenítés, de lehetne akár konzol kimenet, vagy packetek küldése egy szerver-kliens alapú játéknál.
A megvalósításhoz burkolóosztályokat hozunk létre. Minden kommunikálni képes osztályt beburkolunk egy osztályba, mely felülírja azon metódusait, melyek üzenetküldést váltanak ki. Ennek hátránya, hogy az összes példányosítás ezen burkolók cseréjére szorul. Minden modell-béli objektum a Warehouse-on keresztül küldi az üzeneteit, illetve minden view-beli üzenet a Warehouse-nak küldi azt, így egyetlen objektumra szűkítjük a model-view és view-model interfészt.

9.2.2 A felület osztály-struktúrája


Floor
+visitByWorker(firstPusher: Worker, w: Worker): boolean
+visitByCrate(firstPusher: Worker, c: Crate): boolean
+placeSlipFactor(slipFactor: double): boolean
Hole
-open: boolean
+visitByWorker(firstPusher: Worker, w: Worker): boolean
+visitByCrate(firstPusher: Worker, c: Crate): boolean
+visitByLifeCrate(firstPusher: Worker, lc: LifeCrate): boolean
+placeSlipFactor(slipFactor: double): boolean
HoleWrapper
+setOpen(open: boolean): void
WorkerKillStateChangeMessage
+playerIndex: final int
+playerSpawn: final Spawn
+newHP: final int
CrateFallStateChangeMessage
+crate: final Crate
LifeCrateFallStateChangeMessage
+lifeCrate: final LifeCrate
+playerIndex: final int
+newHP: final int
HoleStateChangeMessage
+hole: final Hole
+open: final boolean
PlaceStateChangeMessage
+playerIndex: final int
+floor: final Floor
+item: final PlaceableItem
StateChangeMessage
+type: final StateChangeMessageType
ScoreStateChangeMessage
+playerIndex: final int
+newScore: final int
WorkerStepStateChangeMessage
+playerIndex: final int
CrateStepStateChangeMessage
+crate: final Crate
LifeCrateStepStateChangeMessage
+lifeCrate: final LifeCrate
Crate
+push(firstPusher: Worker, pushed: Entity, dir: Direction): boolean
+pushByWorker(firstPusher: Worker, pusher: Worker, dir: Direction): boolean
+pushByCrate(firstPusher: Worker, pusher: Crate, dir: Direction): boolean
+remove(): void
CrateWrapper
+setCurField(f: Field): void
+remove(): void
FloorWrapper
+placeSlipFactor(slipFactor: double): boolean
LifeCrate
+remove(firstPusher: Worker): void
LifeCrateWrapper
+setCurField(f: Field): void
+remove(firstPusher: WorkerWrapper): void
ControlMessage
+type: final ControlMessageType
PlaceControlMessage
+playerIndex: final int
StepControlMessage
+playerIndex: final int
+direction: final Direction
StepStateChangeMessage
+fieldFrom: final Field
+fieldTo: final Field
Worker
-health: int
-points: int
-power: double
-orgPower: double
-item: PlaceableItem
+push(firstPusher: Worker, pushed: Entity, dir: Direction): boolean
+pushByWorker(firstPusher: Worker, pusher: Worker, dir: Direction): boolean
+pushByCrate(firstPusher: Worker, pusher: Crate, dir: Direction): boolean
+move(dir: Direction): void
+loseHealth(): void
+gainHealth(): void
+losePoint(): void
+gainPoint(): void
+reSpawn(): void
+die(): void
+consumePower(power: double): double
+placeItem(): void
WorkerWrapper
+setCurField(f: Field): void
+loseHealth(): void
+gainHealth(): void
+losePoint(): void
+gainPoint(): void
+reSpawn(): void
+die(): void
Warehouse
+initialize(): void
+setField(f: Field, x: int, y: int): void
+addEntity(e: Entity, x: int, y: int): void
+sendMessage(msg: StateChangeMessage): void
+sendMessage(msg: ControlMessage): void
+receiveMessage(msg: StateChangeMessage): void
+receiveMessage(msg: ControlMessage): void
-warehouse
-view
<<interface>>
IView<T>
+sendMessage(msg: T): void
+receiveMessage(msg: T): void
JPanel
GraphicsView
-stateChanges: List<StateChangeMessage>
+sendMessage(msg: StateChangeMessage): void
+receiveMessage(msg: StateChangeMessage): void
+onPaint(g: Graphics): void
<<enumeration>>
StateChangeMessageType
+WORKERSTEP
+CRATESTEP
+LIFECRATESTEP
+SCORE
+HEALTH
+ITEM
+PLACE
+WORKERFALL
+CRATEFALL
+LIFECRATEFALL
+HOLE
KeyAdapter
KeyboardView
-playerIndex: int
-upKey: int
-downKey: int
-leftKey: int
-rightKey: int
-placeKey: int
+sendMessage(msg: ControlMessage): void
+receiveMessage(msg: ControlMessage): void
<<enumeration>>
ControlMessageType
+STEP
+PLACE

9.3 A grafikus objektumok felsorolása

9.3.1 WorkerWrapper

- **Felelősség**

Értesítést küld a Warehouse osztálynak a dolgozó olyan cselekvéseiről, melyek hatással vannak a grafikus felületre, azaz változtatnak annak állapotán. Felülírja a Worker osztály azon metódusait, melyek grafikai változtatást eredményeznek: delegálja a hívást az ősnek, majd ezután üzenetet küld a változtatás szükségességéről.

- **Ősosztályok**

Worker -> Entity

- **Interfészek**

-

- **Attribútumok**
  - **-**

- **Metódusok**
  - + **setCurField(f: Field): void**: A dolgozó egy új mezőre lépett, át kell állítani a figuráját oda.
  - + **gainPoint(): void**: A dolgozó pontot szerzett, frissíteni kell a számlálót.
  - + **losePoint(): void**: A dolgozó pontot vesztett, frissíteni kell a számlálót.
  - + **gainHealth(): void**: A dolgozó életet szerzett, frissíteni kell a számlálót.
  - + **loseHealth(): void**: A dolgozó életet szerzett, frissíteni kell a számlálót.
  - + **reSpawn(): void:** A dolgozó újraéled, így át kell állítani a figuráját a spawn-jára.
  - + **die(): void**: A dolgozó meghalt, ki kell venni a játékból.

9.3.2 CrateWrapper

- **Felelősség**

Értesítést küld a Warehouse osztálynak a láda olyan cselekvéseiről, melyek hatással vannak a grafikus felületre, azaz változtatnak annak állapotán. Felülírja a Crate osztály azon metódusait, melyek grafikai változtatást eredményeznek: delegálja a hívást az ősnek, majd ezután üzenetet küld a változtatás szükségességéről.

- **Ősosztályok**

Crate -> Entity

- **Interfészek**

-

- **Attribútumok**
  - **-**

- **Metódusok**
  - **+setCurField(f: Field):void:** A láda egy új mezőre lépett, át kell állítani a figuráját.
  - **+ remove(): void:** A láda leesett, ki kell venni a játékból.

9.3.3 LifeCrateWrapper

- **Felelősség**

Értesítést küld a Warehouse osztálynak a szívecskés láda olyan cselekvéseiről, melyek hatással vannak a grafikus felületre, azaz változtatnak annak állapotán. Felülírja a LifeCreate osztály azon metódusait, melyek grafikai változtatást eredményeznek: delegálja a hívást az ősnek, majd ezután üzenetet küld a változtatás szükségességéről.

- **Ősosztályok**

LifeCrate -> Crate -> Entity

- **Interfészek**

-

- **Attribútumok**
  - **-**

- **Metódusok**
  - **+setCurField(f: Field):void:** A láda egy új mezőre lépett, át kell állítani a figuráját.
  - **+ remove(firstPusher: Worker): void:** A láda leesett, ki kell venni a játékból.

9.3.4 FloorWrapper

- **Felelősség**

Értesítést küld a Warehouse osztálynak a padló olyan cselekvéseiről, melyek hatással vannak a grafikus felületre, azaz változtatnak annak állapotán. Felülírja a Floor osztály azon metódusait, melyek grafikai változtatást eredményeznek: delegálja a hívást az ősnek, majd ezután üzenetet küld a változtatás szükségességéről.

- **Ősosztályok**

Floor -> Field

- **Interfészek**

-

- **Attribútumok**
  - **-**

- **Metódusok**
  - **+placeSlipFactor(double slipFactor):boolean**: A padlóra valamilyen kenőanyag került, frissíteni kell a kinézetét.

9.3.5 HoleWrapper

- **Felelősség**

Értesítést küld a Warehouse osztálynak a lyuk olyan cselekvéseiről, melyek hatással vannak a grafikus felületre, azaz változtatnak annak állapotán. Felülírja a Hole osztály azon metódusait, melyek grafikai változtatást eredményeznek: delegálja a hívást az ősnek, majd ezután üzenetet küld a változtatás szükségességéről.

- **Ősosztályok**

Hoel -> Floor -> Field

- **Interfészek**

-

- **Attribútumok**
  - -

- **Metódusok**
  - + **setOpen(boolean open):void**: A lyuk állapota megváltozik, át kell rajzolni a kinézetét.

9.3.6 IView<T>

- **Felelősség**

Implementálja minden nem modellhez tartozó, de a modellel interakcióba lépő osztály.

- **Ősosztályok**

-

- **Metódusok**
  - **+ sendMessage (msg: T): void:** Üzenetet küld a modellnek.
  - **+ receiveMessage (msg: T): void:** Üzenetet fogad a modelltől.

9.3.7 KeyboardView

- **Felelősség**

Egy grafikus elemhez kapcsolva figyeli a billentyűzet lenyomott gombjait. Ha a lenyomott gomb a nézethez tartozó játékos egyik irányításához tartozó gombja, üzenettel szólítja fel a modellt - a

Warehouse-on keresztül - a megfelelő akció végrehajtására. Például 4 játékos esetén 4 darab keyboard view van.

- **Ősosztályok**

KeyAdapter

- **Interfészek**

IView<ControlMessage>

- **Attribútumok**
  - - **playerIndex: int**: A felügyelt játékos indexe.
  - - **upKey: int**: A játékos felfele lépés gombjának kódja.
  - - **downKey: int**: A játékos lefele lépés gombjának kódja.
  - - **leftKey: int**: A játékos balra lépés gombjának kódja.
  - - **rightKey: int**: A játékos jobbra lépés gombjának kódja.
  - - **placeKey: int**: A játékos tárgy-lerakás gombjának kódja.
  - - **warehouse: Warehouse**: A játékteret képező raktár, mely felelős az üzenetek továbbításáért.

- **Metódusok**
  - + **sendMessage (msg: ControlMessage): void**: Ha a lenyomott billentyű irányítja valamire a felügyelt játékost, a megfelelő ControlMessage-re fordítja és elküldi a modellnek.
  - + **receiveMessage (msg: ControlMessage): void**: Nem szükséges.

9.3.8 GraphicsView

- **Felelősség**

Az állapotváltozás üzenetekből állít elő egy grafikus ábrát a hozzá tartozó ablakba.

- **Ősosztályok**

JPanel

- **Interfészek**

IView<StateChangeMessage>

- **Attribútumok**
  - - **stateChanges: ArrayList<StateChangeMessage>**: Tárolja az üzeneteket a kirajzolás pillanatáig. (onPaint() kiüríti)

- **Metódusok**
  - + **sendMessage (msg: StateChangeMessage): void**: Nem szükséges.
  - + **receiveMessage (msg: StateChangeMessage): void**: Eltárolja az üzenetet a *stateChanges* listába.

- + **onPaint (Graphics g): void**: Kirajzolja az állapotváltozások alapján az új megjelenítendő képet, üríti a *stateChanges* listát.

9.3.9 ControlMessageType <<Enumeration>>

- **Felelősség**

Mivel az üzenetek POD-ok, a típuslekérdezés a legegyszerűbb és legmegfelelőbb módja a konkrét üzenettípus meghatározására, ez az enumeráció a ControlMessage-k pontos típusait írja le: *STEP*, *PLACE*.

9.3.10 ControlMessage

- **Felelősség**

Az irányítás üzenetek őse, a munkásokat utasítja.

- **Ősosztályok**

-

- **Interfészek**

-

- **Attribútumok**
  - + **type: final MessageType**: Az kontroll üzenet konkrét típusa.
- **Metódusok**
  - -

9.3.11 StepControlMessage

- **Felelősség**

Lépésre utasítja az egyik munkást.

- **Ősosztályok**

ControlMessage

- **Interfészek**

-

- **Attribútumok**
  - + **playerIndex: final int**: A lépésre utasított munkás indexe.
  - + **direction: final Direction**: Az irány, melybe a munkást lépésre utasítja.

- **Metódusok**
  - -

9.3.12 PlaceControlMessage

- **Felelősség**

A munkásnál levő tárgy letevésére utasítja őt.

- **Ősosztályok**

ControlMessage

- **Interfészek**

-

- **Attribútumok**
  - + **playerIndex: final int**: A letevésre utasított munkás indexe.
- **Metódusok**
  - -

9.3.13 StateChangeMessageType <<Enumeration>>

- **Felelősség**

Mivel az üzenetek POD-ok, a típuslekérdezés a legegyszerűbb és legmegfelelőbb módja a konkrét üzenettípus meghatározására, ez az enumeráció a StateChangeMessage-k pontos típusait írja le: *WORKERSTEP, CRATESTEP, LIFECRATESTEP, SCORE, HEALTH, ITEM, PLACE, WORKERFALL, CRATEFALL, LIFECRATEFALL, HOLE.*

9.3.14 StateChangeMessage

- **Felelősség**

Az irányítás üzenetek őse.

- **Ősosztályok**

-

- **Interfészek**

-

- **Attribútumok**
  - + **type: final StateChangeMessageType**: Az állapotváltozás üzenet konkrét típusa

- **Metódusok**
  - -

9.3.15 StepStateChangeMessage

- **Felelősség**

Közös ős ami entitások kirajzolásáért felelős.

- **Ősosztályok**

StateChangeMessage

- **Interfészek**

-

- **Attribútumok**
  - + **playerIndex: final int**: A lépő dolgozó indexe.
  - + **fieldFrom: final Field**: A mező, melyikről a dolgozó lelépett.
  - + **fieldTo: final Field**: A mező, melyikre a dolgozó lép.

- **Metódusok**
  - -

9.3.16 WorkerStepStateChangeMessage

- **Felelősség**

Egy dolgozót rajzol ki a felületen egy megadott pozícióba, ezzel animálva egy lépést.

- **Ősosztályok**

StepStateChangeMessage -> StateChangeMessage

- **Interfészek**

-

- **Attribútumok**
  - + **playerIndex: final int**: A lépő dolgozó indexe.

- **Metódusok**
  - -

9.3.17 CrateStepStateChangeMessage

- **Felelősség**

Egy ládát rajzol ki a felületen egy megadott pozícióba, ezzel animálva egy lépést.

- **Ősosztályok**

StepStateChangeMessage -> StateChangeMessage

- **Interfészek**

-

- **Attribútumok**
  - + **crate: final Crate**: A tolt láda.
- **Metódusok**
  - -

9.3.18 LifeCrateStepStateChangeMessage

- **Felelősség**

Egy szívecskés ládát rajzol ki a felületen egy megadott pozícióba, ezzel animálva egy lépést.

- **Ősosztályok**

StepStateChangeMessage -> StateChangeMessage

- **Interfészek**

-

- **Attribútumok**
  - + **lifeCrate: final LifeCrate**: A tolt szívecskés láda.
- **Metódusok**
  - -

9.3.19 CrateFallStateChangeMessage

- **Felelősség**

Egy láda lyukba esését animálja.

- **Ősosztályok**

StateChangeMessage

- **Interfészek**

-

- **Attribútumok**
  - + **crate: final Crate**: A leesett láda.
- **Metódusok**
  - -

9.3.20 LifeCrateFallStateChangeMessage

- **Felelősség**

Egy szívecskés láda lyukba esését animálja.

- **Ősosztályok**

StateChangeMessage

- **Interfészek**

-

- **Attribútumok**
  - + lifeCrate: final LifeCrate: A leesett szívecskés láda.
- **Metódusok**
  - -

9.3.21 ScoreStateChangeMessage

- **Felelősség**

Egy dolgozót pontszámát változtatja a kijelző felületén.

- **Ősosztályok**

StateChangeMessage

- **Interfészek**

-

- **Attribútumok**
  - + **playerIndex: final int**: A pontot kapó vagy veszítő játékos indexe.
  - + **newScore: final int:** A játékos új pontszáma.

- **Metódusok**
  - -

9.3.22 HealthStateChangeMessage

- **Felelősség**

Egy dolgozót életeinek számát változtatja a kijelző felületén.

- **Ősosztályok**

StateChangeMessage

- **Interfészek**

-

- **Attribútumok**
  - + **playerIndex: final int**: A pontot kapó vagy veszítő játékos indexe.
  - + **newScore: final int:** A játékos új pontszáma.

- **Metódusok**
  - -

9.3.23 ItemStateChangeMessage

- **Felelősség**

Egy dolgozónál lévő tárgyat változtatja a kijelző felületén.

- **Ősosztályok**

StateChangeMessage

- **Interfészek**

-

- **Attribútumok**
  - + **playerIndex: final int**: Az tárgyat kapó vagy lerakó játékos indexe.
  - + **item: final PlaceableItem:** A játékos dolgozójánál lévő tárgy.

- **Metódusok**
  - -

9.3.24 PlaceStateChangeMessage

- **Felelősség**

Egy tárgy lerakását animálja.

- **Ősosztályok**

StateChangeMessage

- **Interfészek**

-

- **Attribútumok**
  - + **floor: final Floor**: A padló, amelyre lerakjuk a tárgyat.
  - + **item: final PlaceableItem:** A padlóra lerakandó tárgy.

- **Metódusok**
  - -

9.3.25 HoleStateChangeMessage

- **Felelősség**

Egy lyuk állapotváltozását reprezentálja.

- **Ősosztályok**

StateChangeMessage

- **Interfészek**

-

- **Attribútumok**
  - + **hole: final Hole**: A pontot kapó vagy veszítő játékos indexe.
  - + **open: final boolean:** A lyuk új állapota.

- **Metódusok**
  - -

9.3.26 WorkerKillStateChangeMessage

- **Felelősség**

A játékos meghalását reprezentálja (lyukba esi .

- **Ősosztályok**

StateChangeMessage

- **Interfészek**

-

- **Attribútumok**
  - + **playerIndex: final int**: A leesett dolgozóhoz tartozó játékos indexe.
  - + **playerSpawn: final Spawn:** A leesett dolgozóhoz tartozó kiindulási hely.
  - + **newHP: final int:** A leesett dolgozóhoz új csökkentett élete.

- **Metódusok**
  - -

9.3.27 Warehouse

- **Felelősség**

Az osztály abban változik, hogy ő felelős az üzenetek továbbításáért a csomagoló osztályok és a View-k között.

- **Ősosztályok**

-

- **Interfészek**

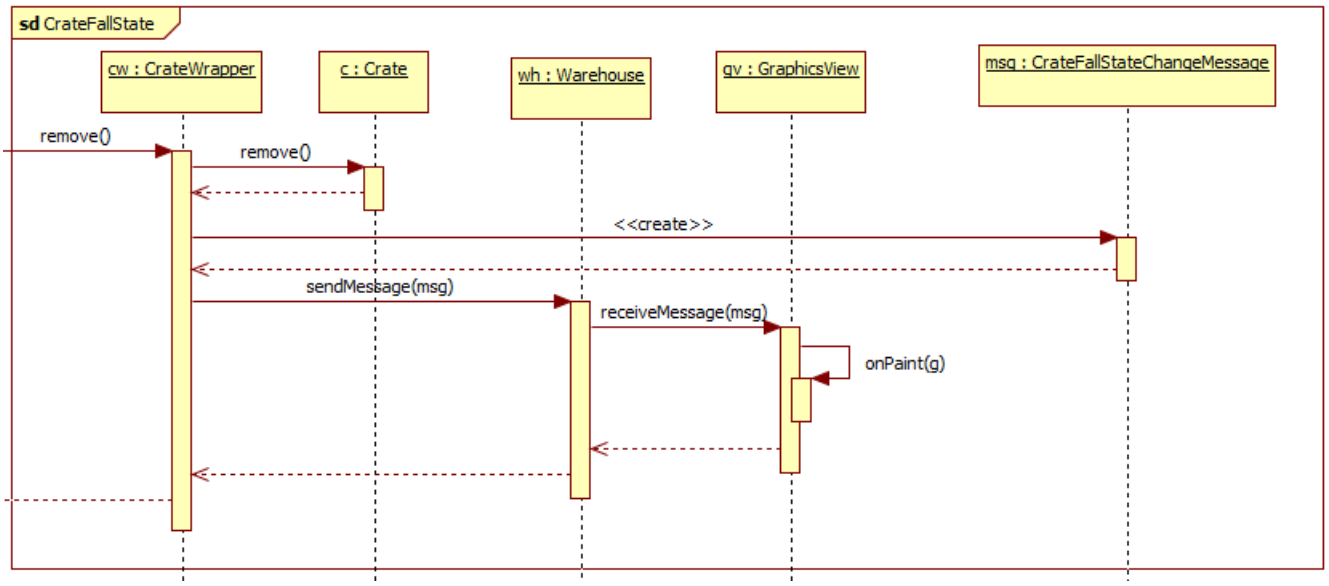
-

- **Attribútumok**
  - **- view: IView<T>**

- **Metódusok**
  - + **sendMessage (msg: StateChangeMessage): void**: A raktár üzenetet továbbít.
  - **+ sendMessage (msg: ControlMessage): void**: A raktár üzenetet továbbít.
  - **+ receiveMessage (msg: StateChangeMessage): void**: A raktár üzenetet fogad.
  - **+ receiveMessage (msg: ControlMessage): void**: A raktár üzenetet fogad.

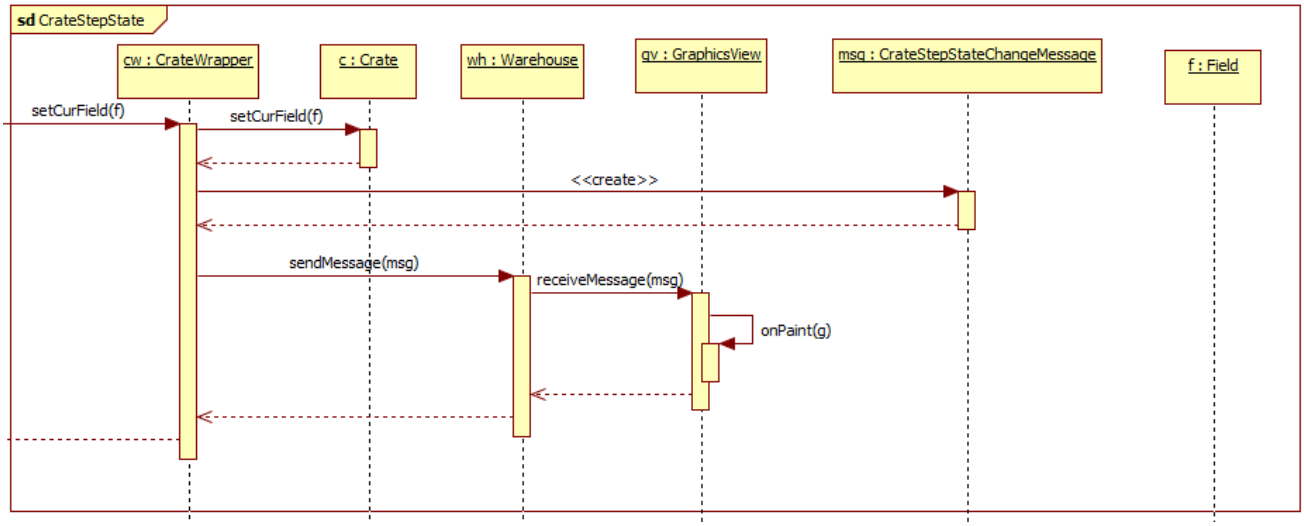
9.4 Kapcsolat az alkalmazói rendszerrel

9.4.1 CrateFallState



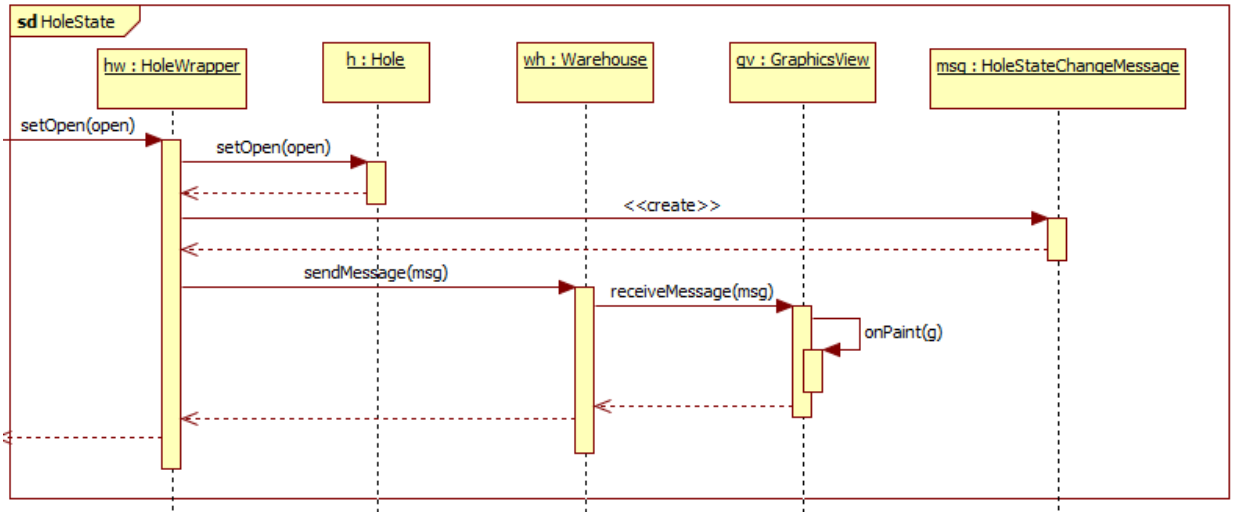
Repaint függvényt hívunk, és annak hatására fog meghívódni az onPaint, nem közvetlen mi hívjuk.

9.4.2 CrateStepState



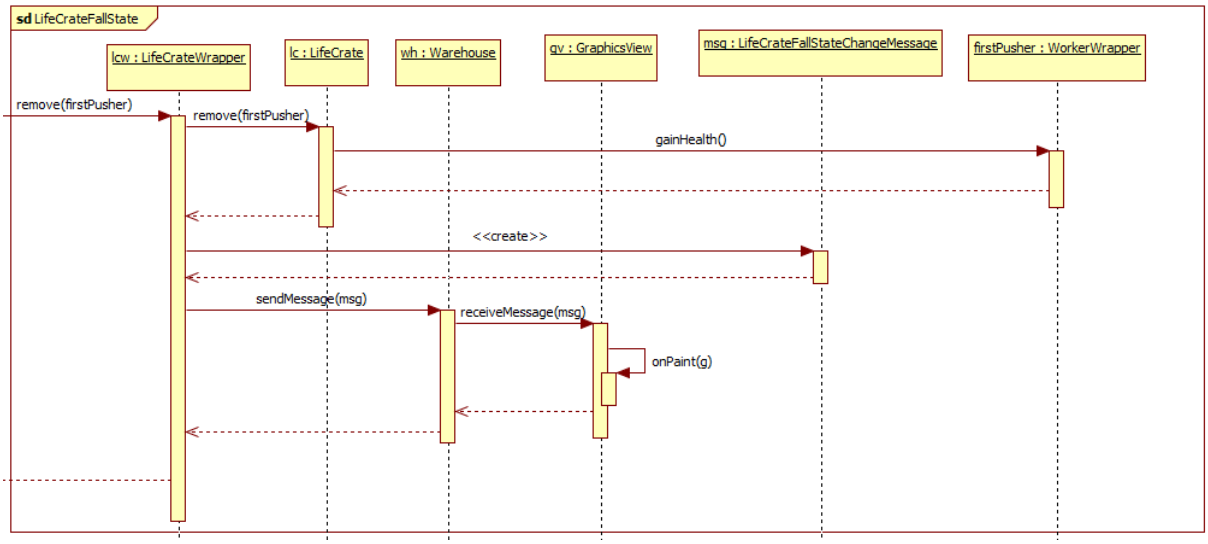
Repaint függvényt hívunk, és annak hatására fog meghívódni az onPaint, nem közvetlen mi hívjuk.

9.4.3 HoleState



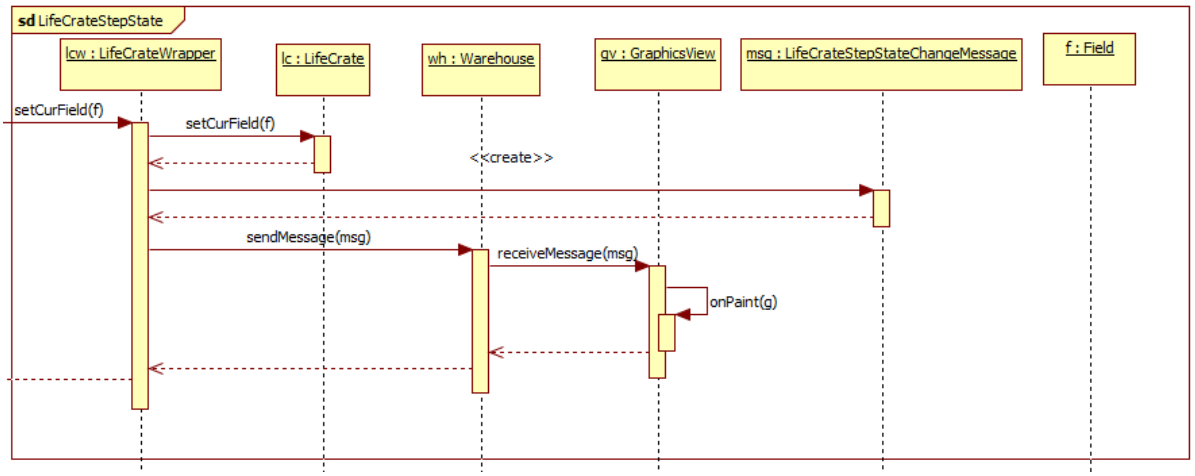
Repaint függvényt hívunk, és annak hatására fog meghívódni az onPaint, nem közvetlen mi hívjuk.

9.4.4 LifeCrateFallState



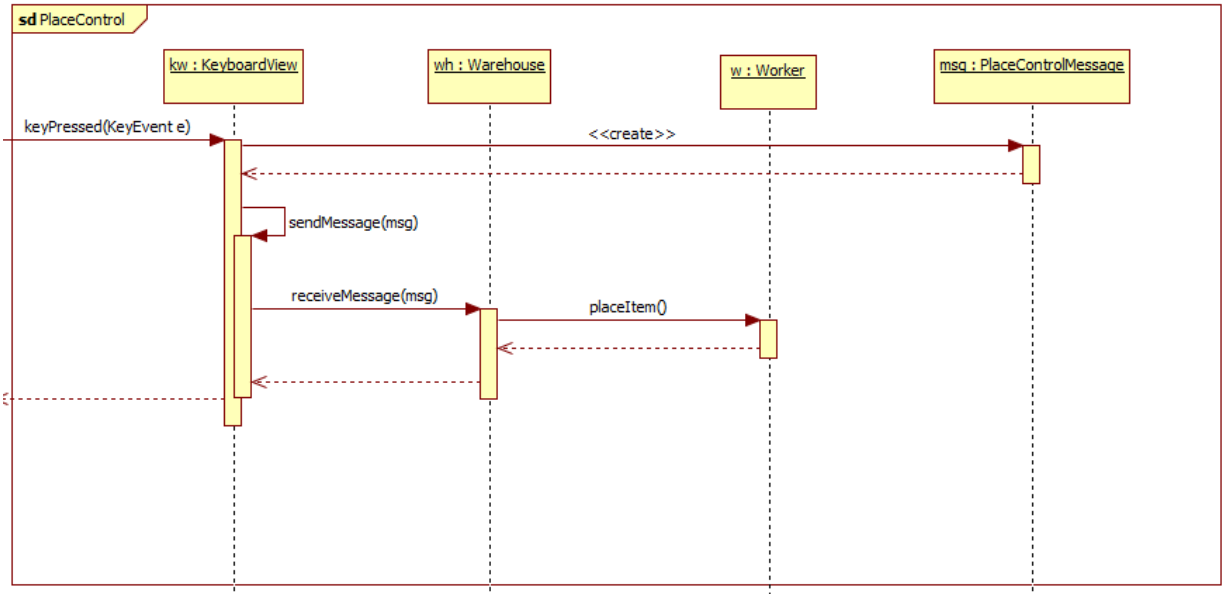
Repaint függvényt hívunk, és annak hatására fog meghívódni az onPaint, nem közvetlen mi hívjuk.

9.4.5 LifeCrateStepState

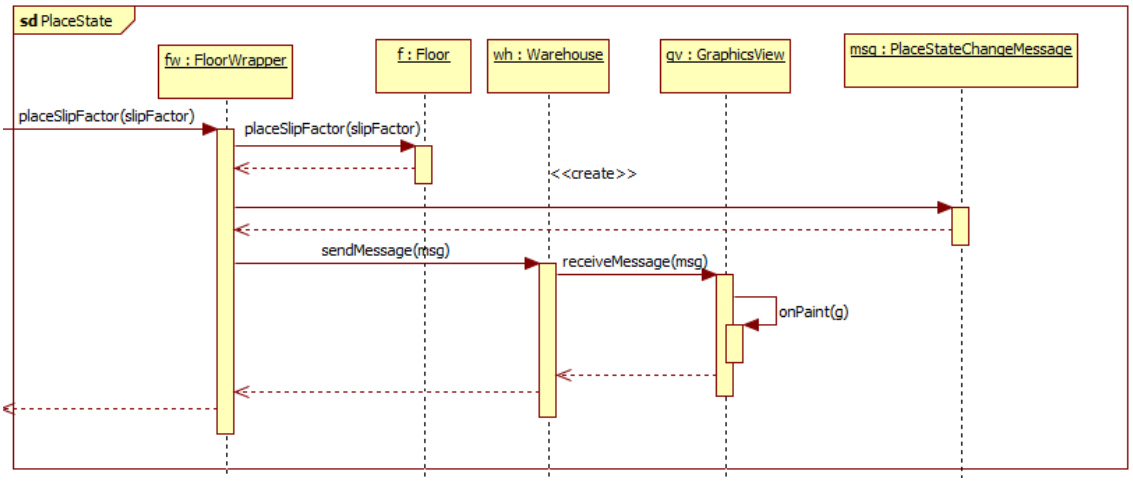


Repaint függvényt hívunk, és annak hatására fog meghívódni az onPaint, nem közvetlen mi hívjuk.

9.4.6 PlaceControl

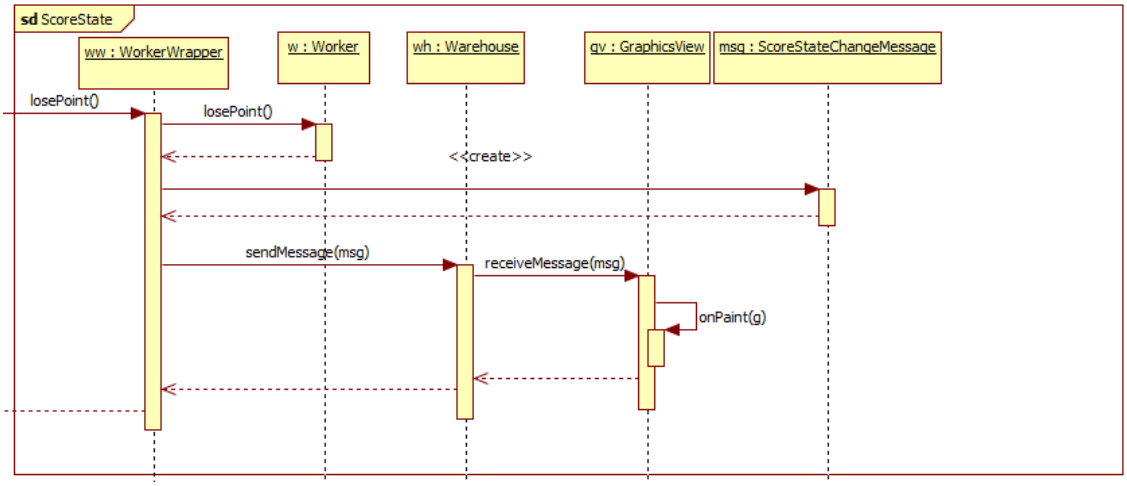


9.4.7 PlaceState



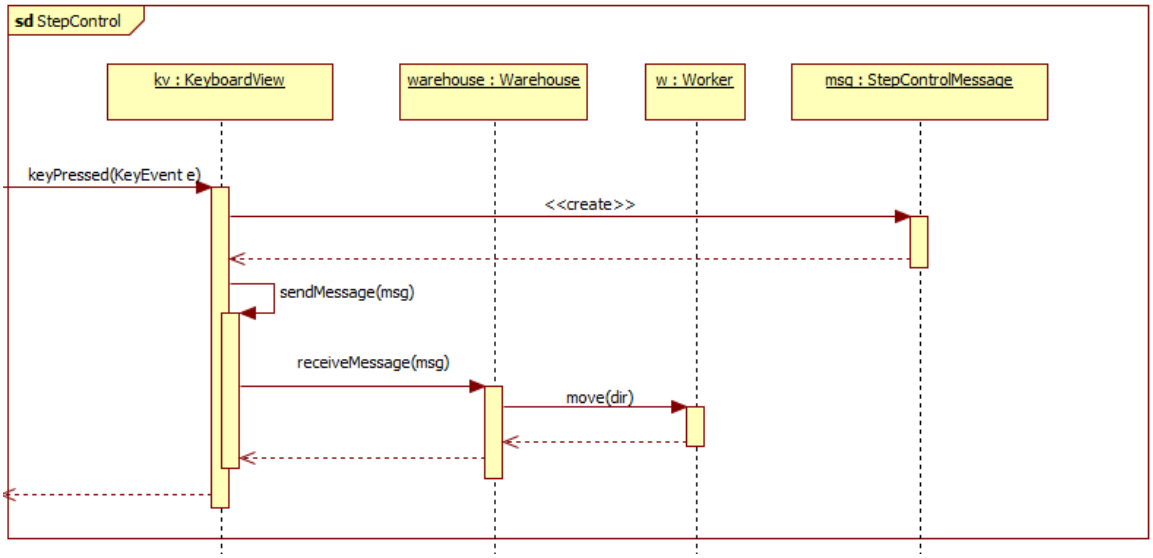
Repaint függvényt hívunk, és annak hatására fog meghívódni az onPaint, nem közvetlen mi hívjuk.

9.4.8 ScoreState

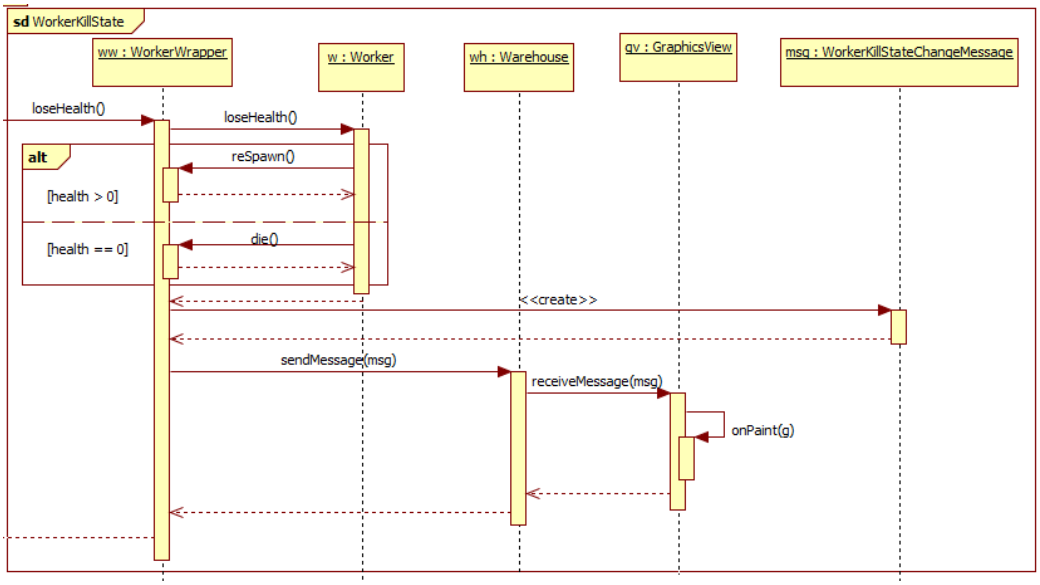


Repaint függvényt hívunk, és annak hatására fog meghívódni az onPaint, nem közvetlen mi hívjuk.

9.4.9 StepControl

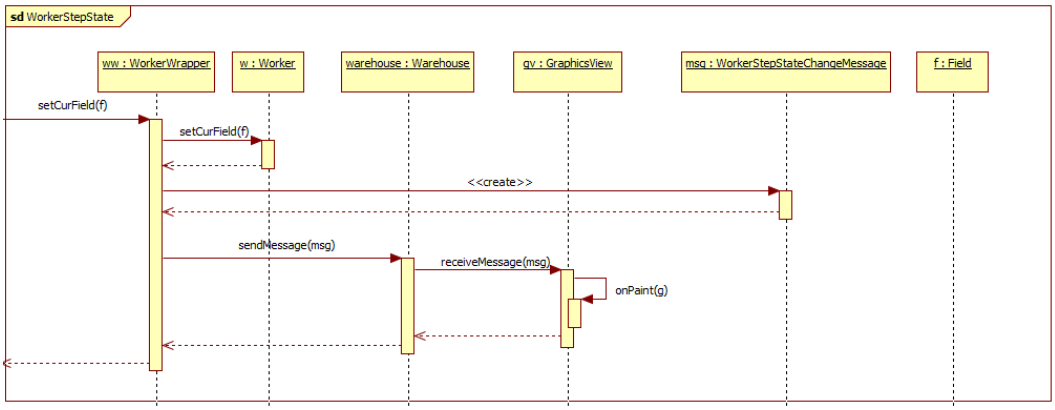


9.4.10 WorkerKillState



Repaint függvényt hívunk, és annak hatására fog meghívódni az onPaint, nem közvetlen mi hívjuk.

9.4.11 WorkerStepState



Repaint függvényt hívunk, és annak hatására fog meghívódni az onPaint, nem közvetlen mi hívjuk.

9.5 Napló

Dátum	Időtartam	Résztvevők	Leírás
2018.01.04-05	3 óra	JANI	Osztályok leírása.
2018.01.04-05	3 óra	SZAKÁLLAS	Szekvenciadiagramok.
2018.01.04-05	3 óra	CSANÁDY	Osztályok leírása.
2018.01.04-05	3 óra	LENKEFI	Grafikus interfész és a felület működési elve.
2018.01.04-05	3 óra	LAKATOS	Osztálydiagram, osztályok leírása, adminisztráció.

10. Grafikus változat beadása

10.1 Fordítási és futtatási útmutató

10.1.1 Fájllista

Fájl létrehozásának dátuma, mérete (byte) és neve*

```
05/13/2018  22:37         2,051 Main.java
05/13/2018  22:37         2,112 Crate.java
05/13/2018  22:37           677 CrateWrapper.java
05/13/2018  22:37           556 Direction.java
05/13/2018  22:37         5,242 Entity.java
05/13/2018  22:37         4,600 Field.java
05/13/2018  22:37           938 Floor.java
05/13/2018  22:37           914 FloorWrapper.java
05/13/2018  22:37         2,369 Hole.java
05/13/2018  22:37           934 HoleWrapper.java
05/13/2018  17:36         1,533 IVisitable.java
05/13/2018  17:36           710 IVisitor.java
05/13/2018  22:41           128 LevelFormatException.java
05/13/2018  22:41         1,113 LifeCrate.java
05/13/2018  22:37           669 LifeCrateWrapper.java
05/13/2018  22:41           636 PlaceableItem.java
05/13/2018  22:45           439 Process.java
05/13/2018  22:45         1,545 Spawn.java
05/13/2018  22:46           964 SpawnWrapper.java
05/13/2018  23:05         1,911 StepHoleProcess.java
05/13/2018  23:02         2,587 StepHoleProcessWrapper.java
05/13/2018  23:06         1,574 StepProcess.java
05/13/2018  23:05         1,920 StepProcessWrapper.java
05/13/2018  23:08         1,845 StepWallProcess.java
05/13/2018  23:26         1,585 StepWallProcessWrapper.java
05/13/2018  23:27         1,683 Switch.java
05/13/2018  23:27           698 SwitchWrapper.java
05/13/2018  23:27         1,331 Target.java
```

05/13/2018 23:28	697 TargetWrapper.java
05/13/2018 23:28	1,303 Wall.java
05/13/2018 23:28	686 WallWrapper.java
05/13/2018 23:48	15,316 Warehouse.java
05/13/2018 23:35	5,360 Worker.java
05/13/2018 22:37	1,805 WorkerWrapper.java
05/13/2018 23:42	346 IView.java
05/13/2018 23:42	1,934 KeyboardView.java
05/13/2018 23:44	1,119 CrateShape.java
05/13/2018 23:44	1,051 FloorShape.java
05/13/2018 23:44	743 GoalShape.java
05/13/2018 23:49	11,102 GraphicsView.java
05/13/2018 23:45	678 HoleShape.java
05/13/2018 23:45	870 LifeCrateShape.java
05/13/2018 23:45	2,083 PlayerShape.java
05/13/2018 23:45	970 PlayerSpawnShape.java
05/13/2018 23:46	345 Shape.java
05/13/2018 23:47	579 SwitchShape.java
05/13/2018 23:47	407 WallShape.java
05/13/2018 22:37	195 ControlMessage.java
05/13/2018 22:37	85 ControlMessageType.java
05/13/2018 22:37	367 CrateFallStateChangeMessage.java
05/13/2018 22:37	415 CrateStepStateChangeMessage.java
05/13/2018 22:37	396 FallStateChangeMessage.java
05/13/2018 22:37	403 GameOverStateChangeMessage.java
05/13/2018 22:37	334 HealthStateChangeMessage.java
05/13/2018 22:37	349 HoleStateChangeMessage.java
05/13/2018 22:37	384 ItemStateChangeMessage.java

05/13/2018 22:37	407 LifeCrateFallStateChangeMessage.java
05/13/2018 22:37	455 LifeCrateStepStateChangeMessage.java
05/13/2018 22:37	255 PlaceControlMessage.java
05/13/2018 22:37	399 PlaceStateChangeMessage.java
05/13/2018 22:37	345 ScoreStateChangeMessage.java
05/13/2018 22:37	211 StateChangeMessage.java
05/13/2018 22:37	227 StateChangeMessageType.java
05/13/2018 22:37	376 StepControlMessage.java
05/13/2018 22:37	476 StepStateChangeMessage.java
05/13/2018 22:37	451 TileRegisterStateChangeMessage.java
05/13/2018 22:37	156 TileType.java
05/13/2018 22:37	482 WorkerFallStateChangeMessage.java
05/13/2018 22:37	523 WorkerSquashStateChangeMessage.java
05/13/2018 22:37	527 WorkerStepStateChangeMessage.java

*A fájl neve leírja annak tartalmát, így a tartalom mezőt kihagytuk.

10.1.2 Fordítás és telepítés

javac -d bin -classpath deps/javax.json-1.0.4.jar src/skeleton/*.java src/skeleton/model/*.java src/skeleton/view/*.java src/skeleton/view/gfx/*.java src/skeleton/view/message/*.java

10.1.3 Futtatás

cd bin
java -cp .;../deps/javax.json-1.0.4.jar skeleton.Main

10.2 Értékelés

Tag neve	Munka százalékban
CSANÁDY Máté	18.16
JANI Balázs	20.13
LENKEFI Péter	20.57
LAKATOS Dániel	20.79

SZAKÁLLAS Gergely	20.35

10.3 Napló

Kezdet	Időtartam (óra)	Résztvevők	Leírás
2018.05.12-13	3	LAKATOS	Üzenetek, wrapperek
2018.05.12-13	3	LENKEFI	GUI
2018.05.12-13	4	SZAKÁLLAS	Üzenetek, View
2018.05.12-13	5	CSANÁDY	Wrapperek
2018.05.12-13	4	JANI	Játéklogika, tesztelés

11. Összefoglalás

11.1 A projektre fordított összes munkaidő

Tag neve	Munkaidő (óra)
CSANÁDY Máté	41.5
JANI Balázs Gábor	46
LAKATOS Dániel	47.5
LENKEFI Péter	47
SZAKÁLLAS Gergely	46.5
Összesen	**228.5**

- ***A feltöltött programok forrássorainak száma***

Fázis	Kódsorok száma
Szkeleton	1667
Prototípus	2685
Grafikus változat	4063

Összesen	**8415**

11.2 • Projekt összegzés

11.2.1 Mit tanultak a projektből konkrétan és általában?

Nem lehet minden OOP elvet teljesíteni, mivel azok ellentmondanak egymásnak kisebb-nagyobb mértékben.

A logika és megjelenítés (MVVM / MVC / hívjuk ahogy akarjuk) elválasztása igen hasznos. Például egy konzolos tesztelésnél útban lenne a GUI, így egyszerűen leválasztható. A portolás is könnyebb, hiszen (simplified example) ha egy DirectX megjelenítőnk van, a linuxra portoláshoz elég egy OpenGL megjelenítőt írni.

Szóban sokkal egyszerűbb elmagyarazni egy, a projekten kívüli személynek a program működését, a teszteléshez szükséges infókat, mint egy dokumentációt írni hozzá.

11.2.2 Mi volt a legnehezebb és a legkönnyebb?

Viszonylag nehéz volt olyan analízis modellt kidolgozni, amely robusztus ugyanakkor nem sérti az objektum-orientáltság szabályainak, alapelveinek nagy részét. A feladat elején még nincs tisztában az ember a lehetséges felmerülő problémákkal akár a modellezés, akár az implementálás szintjén, így az akármennyire is kidolgozottnak látszik az alapmodell, majdnem biztosan finomításra szorul a projekt későbbi fázisaiban.

Az együttműködés és munkafolyamatok összehangolása eleinte nehézkesen indult. Nehéz volt átlátni más kódját és gondolatmenetét és eleinte nehéz volt tényleg csak “egy fekete doboz”-ként tekinteni rá.

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

A kredit vs. ráfordítás itt is probléma, de nem szeretnék belemenni, hiszen az kari szintű dolog (egy Grafika 3 kredit, egy MEV pedig 4… utóbbival tizedannyit foglalkoztam, mint ezzel a tárggyal vagy a Grafikával és simán 5-ös lettem).

Az időt jobban el kéne (kellett volna tudnunk) osztani. Előre dolgozni nem lehet, mert szerdán derül ki, hogy mi a rossz az előző leadásban, viszont a következővel már hétfőre végezni kell. Idén a hét elején voltak a ZH-k, amire nyilván hétvégén tanultunk.

11.2.4 Ha nem, akkor hol okozott ez nehézséget?

Mivel egy-két kivétellel minden héten kellet leadni anyagot, volt, amikor már sok volt a Projlab a házik, a mérések és a ZH-k mellett.

11.2.5 Milyen változtatási javaslatuk van?

Elektronikusan kelljen leadni az összes dokumentumot. Egyrészt 2018 van, másrészt óvjuk a fákat. Csak a mi csapatunk közel 180 A4-es oldalt nyomtatott ki, ez évfolyamszinten rengeteg. Nekünk nem jelentett semmilyen előnyt a papír alapú dokumentáció, inkább csak hátrány volt, hogy folyamatosan nyomtatgatni kellett. Persze lehet, hogy a konzulens életét könnyíti meg a papír alapú dokumentum, mert azt egyszerűbb ellenőrizni, ezt nem tudhatjuk.

Ne 97-es Word sablonokkal kelljen dolgozni. Nagyon idegesítő, amikor az ember lementené a fájlt egy újabb verzióban és csomó minden átformálódik és elromlik.

11.2.6 Milyen feladatot ajánlanának a projektre?

-

11.2.7 Egyéb kritika és javaslat

Úgy éreztük, nem feltétlenül konzisztens a számonkérés a Szoftvertechnológiában leadottakkal. Volt, hogy pl. a Szoftech mintaházban használt módszereket követtük, mely itt hibának számított és pontlevonást is kaptunk érte.

Az időt valahogy jobban el kéne osztani, hogy ne csak a saját, de a konzulensünk terhelése is csökkenjen. Érthető, hogy ennyi idő alatt nem lehet mindenki munkáját részletesen átnézni, de előfordult, hogy egy Analízis 1-ben észre nem vett (és ezért nem javított) hiba Alanízis 2-ben derült ki, ami mindkét fél részéről kellemetlen volt.