# Analízis modell kidolgozása

## Objektum katalógus

### 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.

### 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.

### Szívecskés láda

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

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

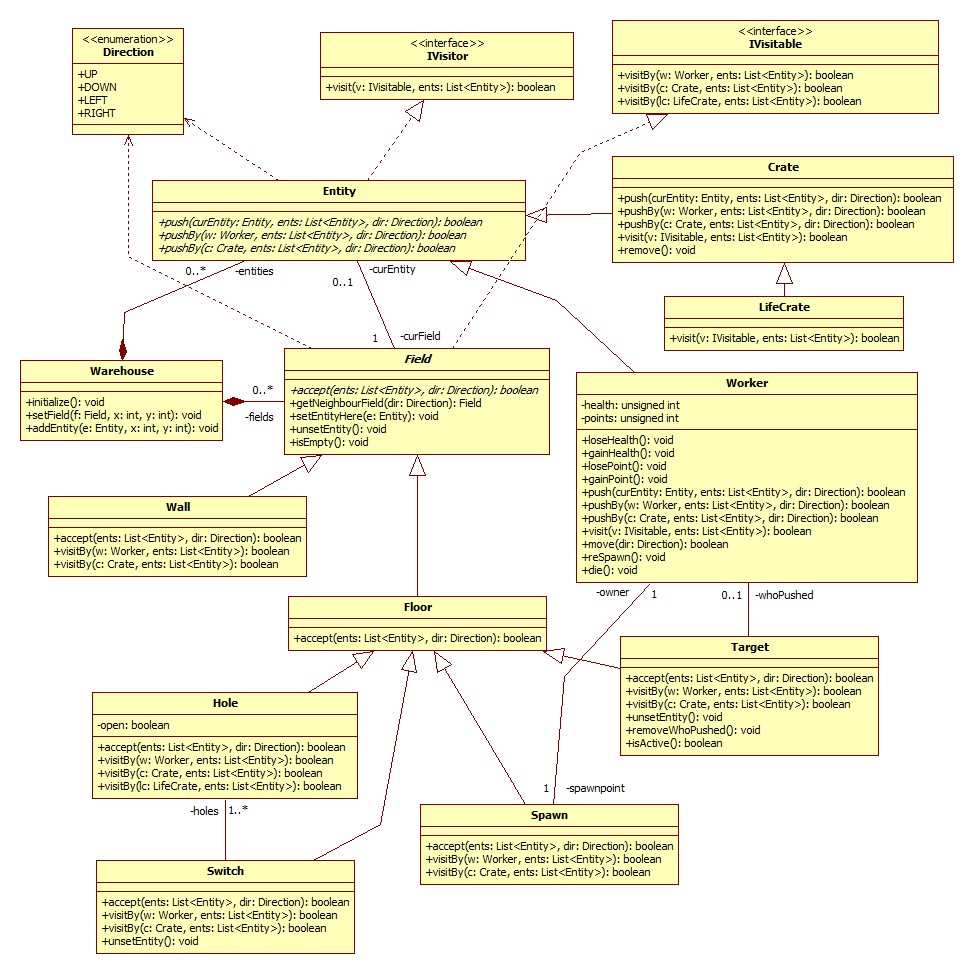
### 3.1.10 Fal

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.

## Statikus struktúra diagramok



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

## Osztályok leírása

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

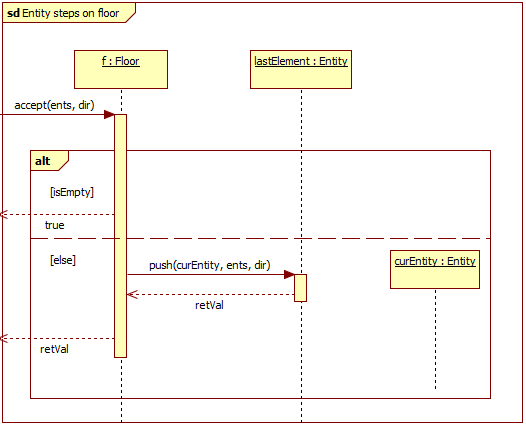
## Szekvencia diagramok

*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.*

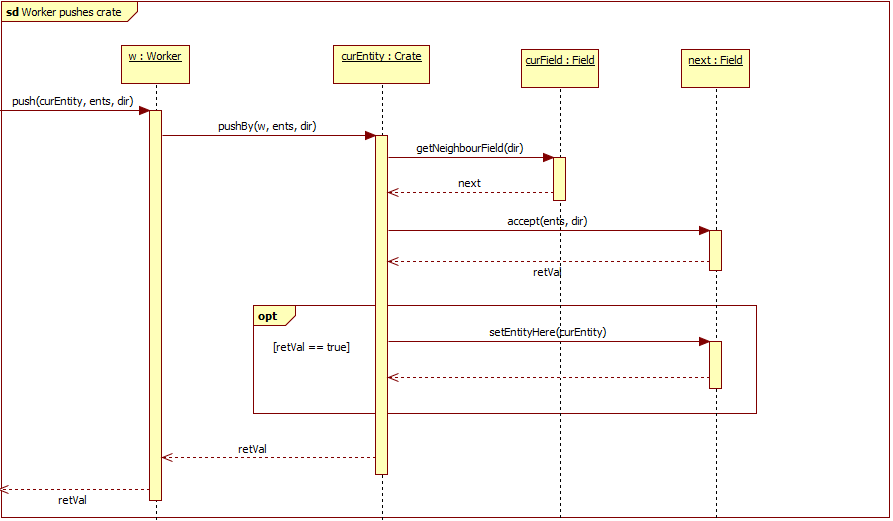
### A játékos lépteti a dolgozót

## 

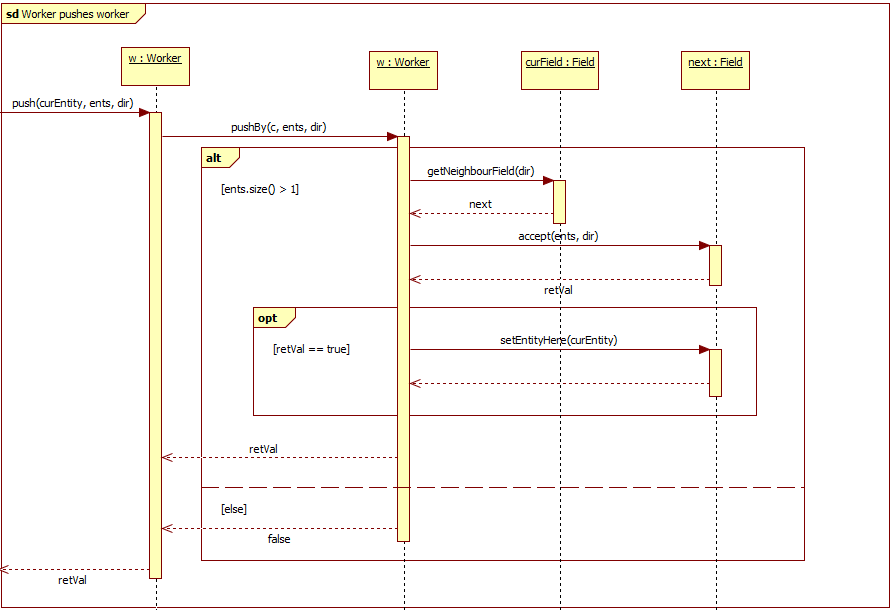
### Egy entitás egy padlóra lép



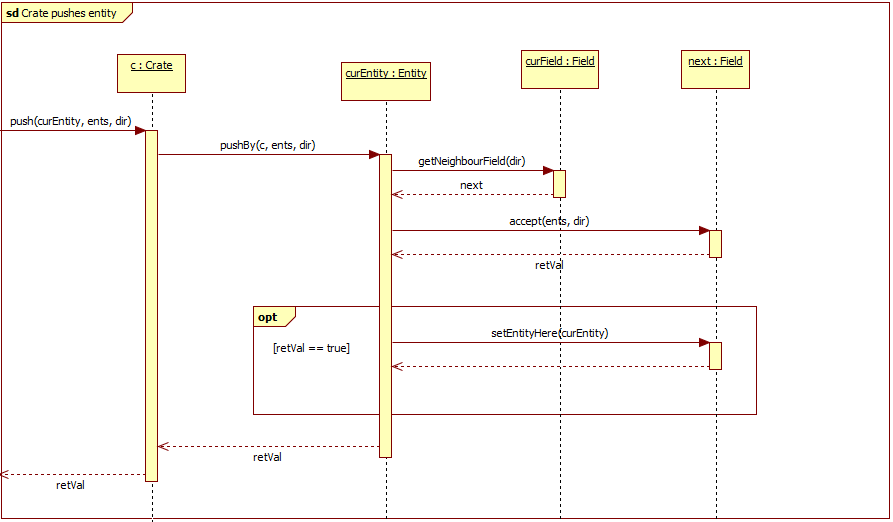
### Egy dolgozó ládát tol



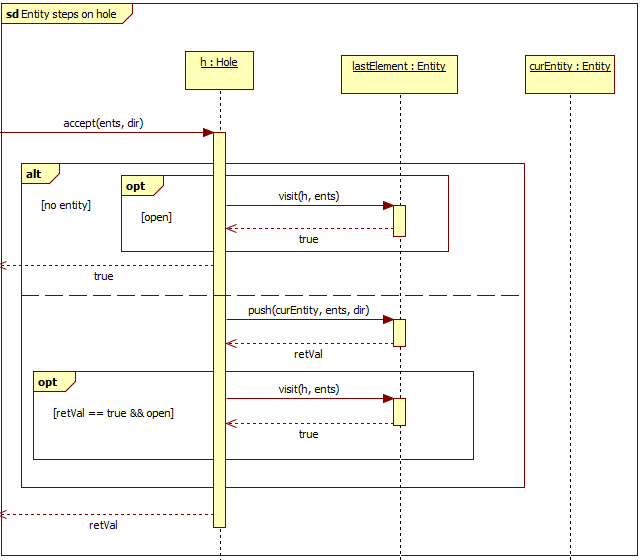
### Egy dolgozó dolgozót tol



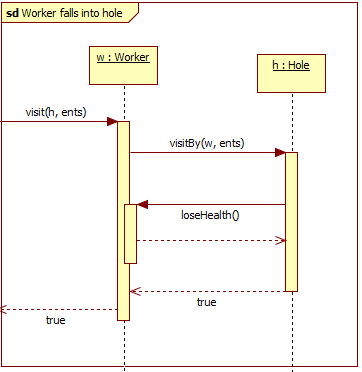
### Egy láda entitást tol



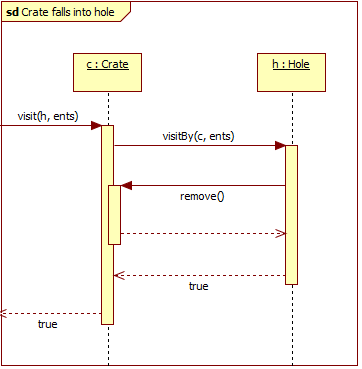
### Entitás lyukra lép



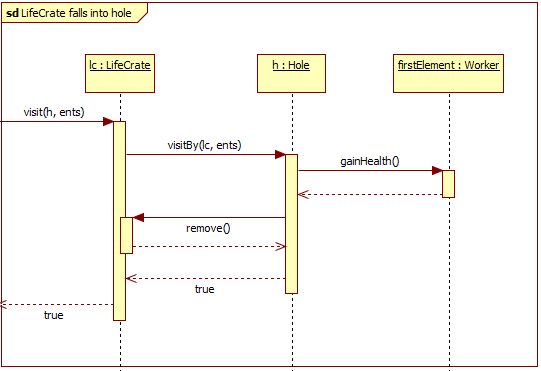
### Dolgozó lyukba esik



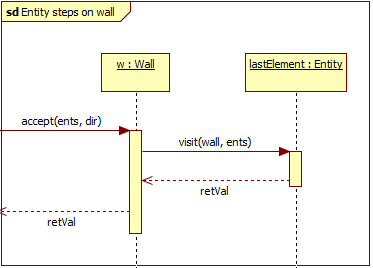
### Láda lyukba esik



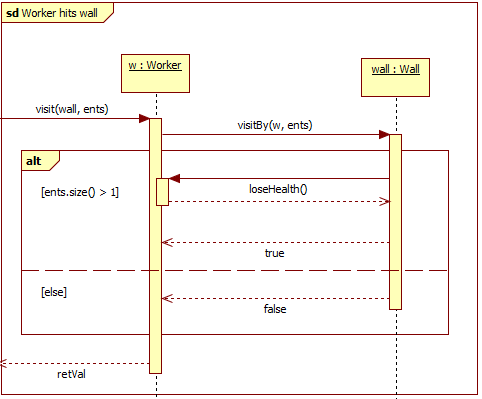
### Szívecskés láda lyukba esik



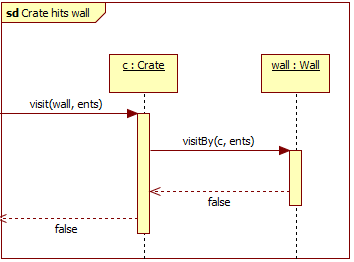
### Entitás falnak megy



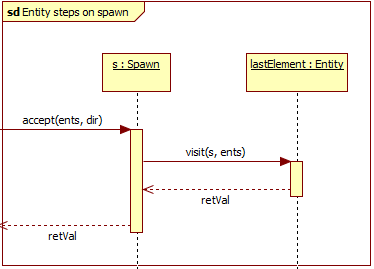
### Dolgozó falnak megy



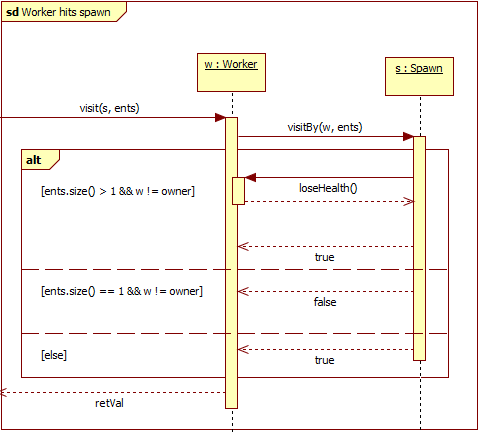
### Láda falnak megy



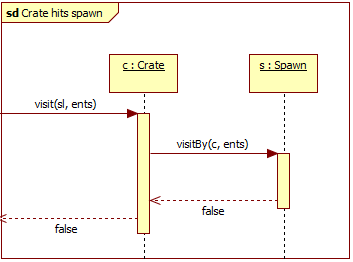
### Entitás kiindulási mezőnek ütközik



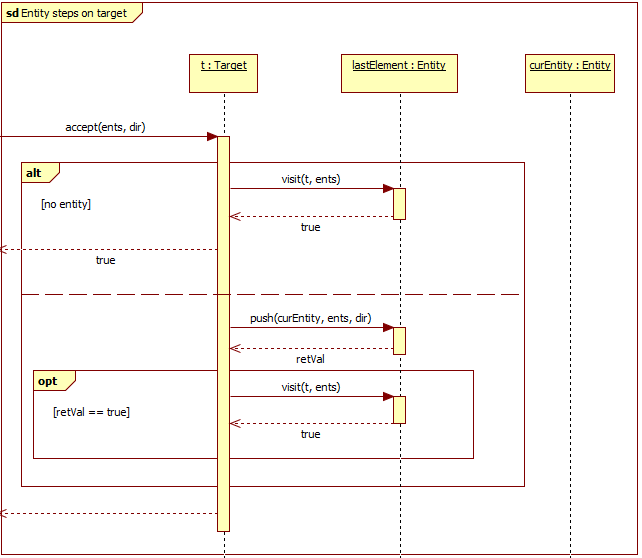
### Dolgozó kiindulási mezőnek ütközik



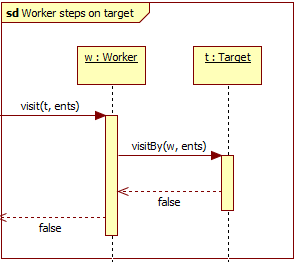
### Láda kiindulási mezőnek ütközik



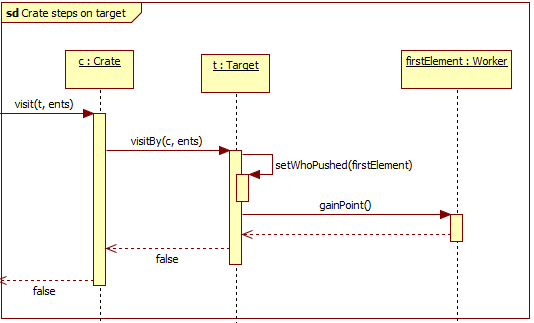
### Entitás előírt helyre lép



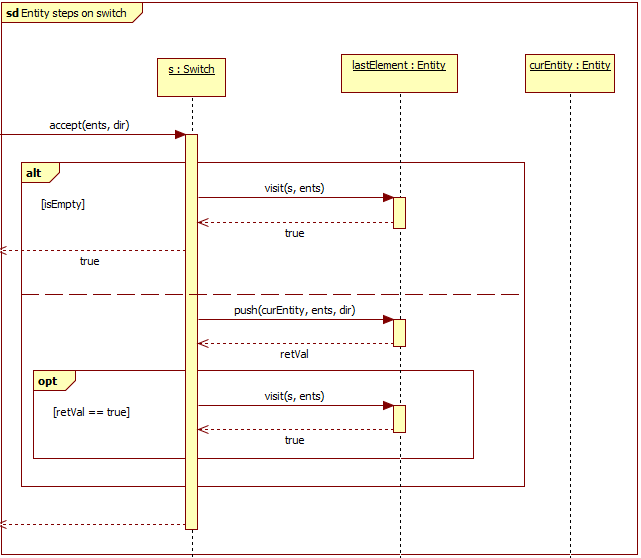
### Dolgozó előírt helyre lép



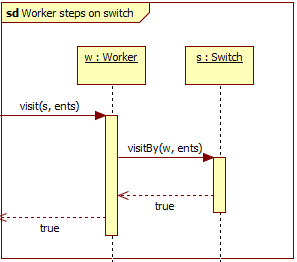
### Láda előírt helyre lép



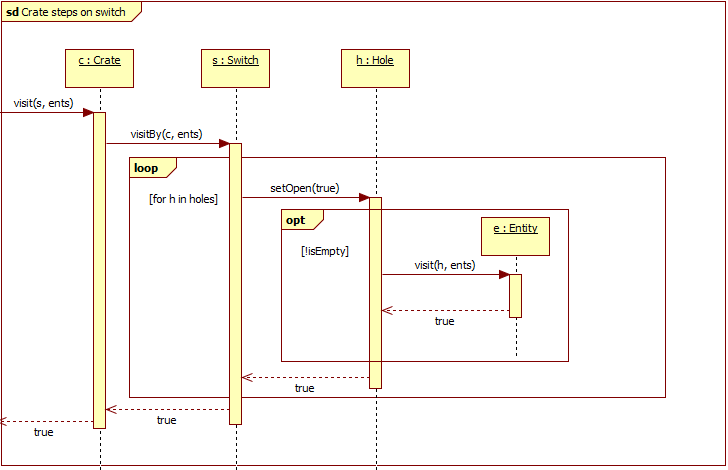
### Entitás kapcsolóra lép



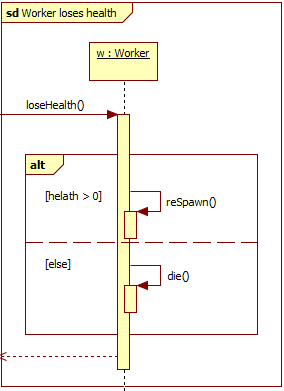
### Dolgozó kapcsolóra lép



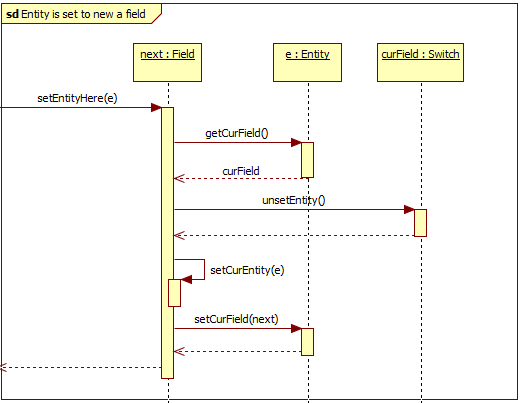
### Láda kapcsolóra lép



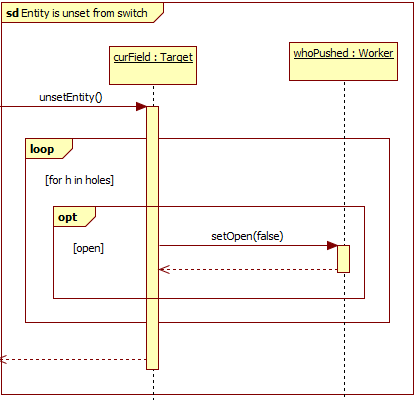
### Dolgozó életet veszít



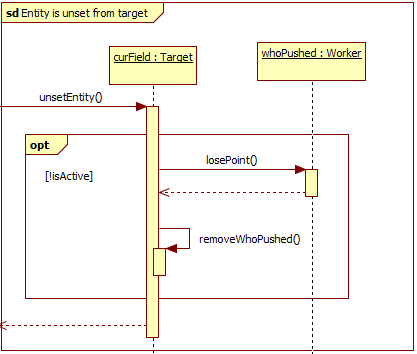
### Entitás új mezőre kerül



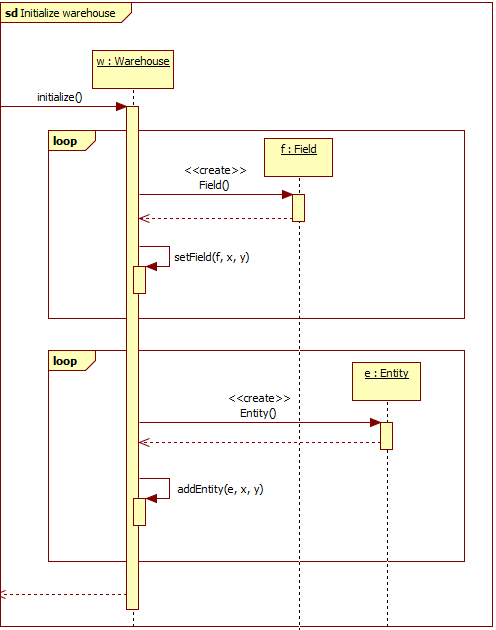
### Entitás új mezőre kerül egy kapcsolóról



### Entitás új mezőre kerül egy előírt helyről

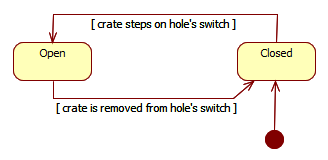


### Raktárépület inicializálása



## State-chartok

### Lyuk állapota változik



## 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 |