# 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. Nem összenyomhatók, a játéktérből pedig csak úgy kerülhetnek ki, ha Lyukba esnek.

### 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 lehetnek szomszédosok, amelyeket 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.

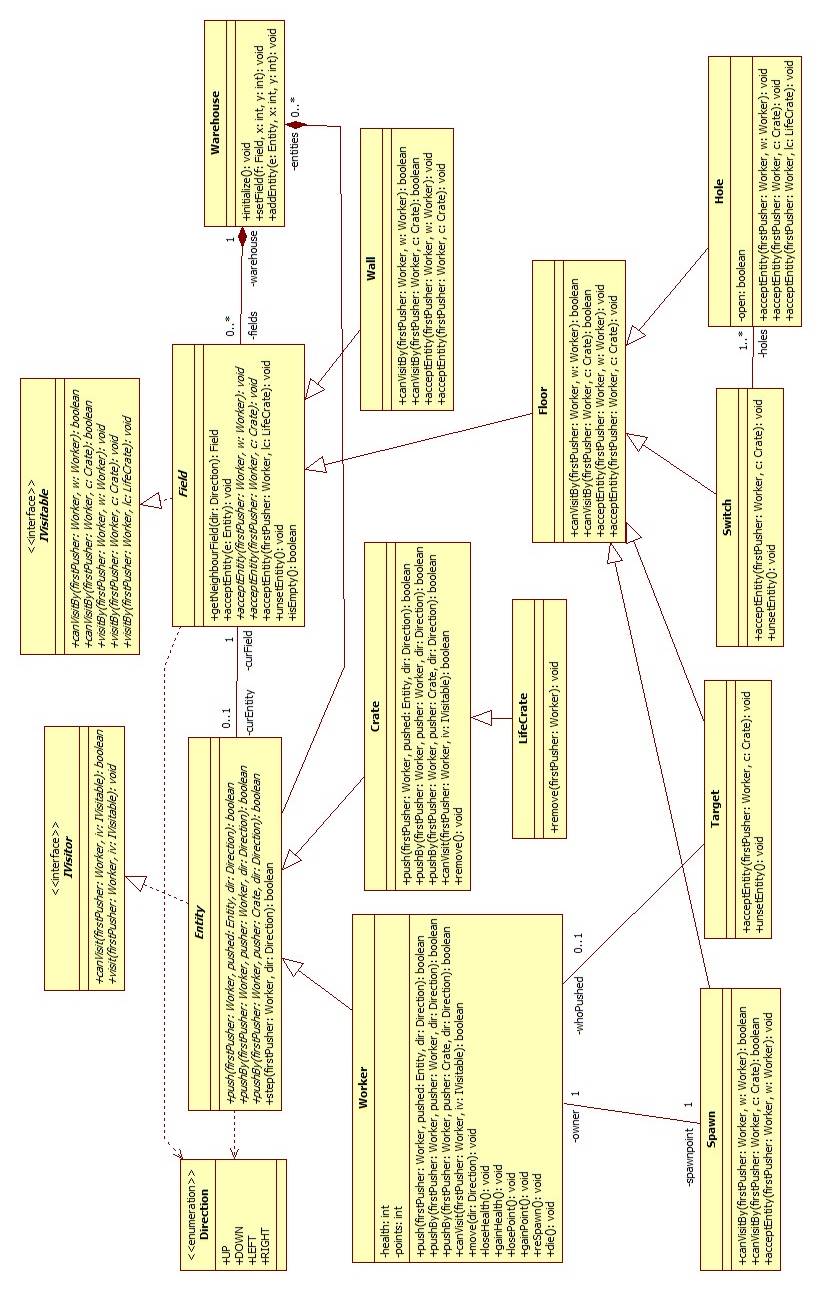
### Fal

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.

### 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 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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 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.

## Szekvencia diagramok

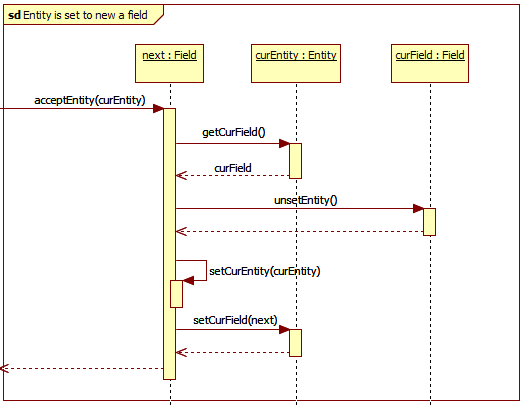
*Megjegyzés: A „dobozokban” lévő aláhúzásokat nem lehetséges eltüntetni a WhiteStarUML-ben, ezért úgy tekintjük, mintha azok ott sem lennének, hiszen aláhúzva Object-eket reprezentálnának. Továbbá mivel az UML eszköz nem támogatja a szabványos create jelölést, így az metódushívásként van modellezve.*

### A játékos lépteti a dolgozót

## 

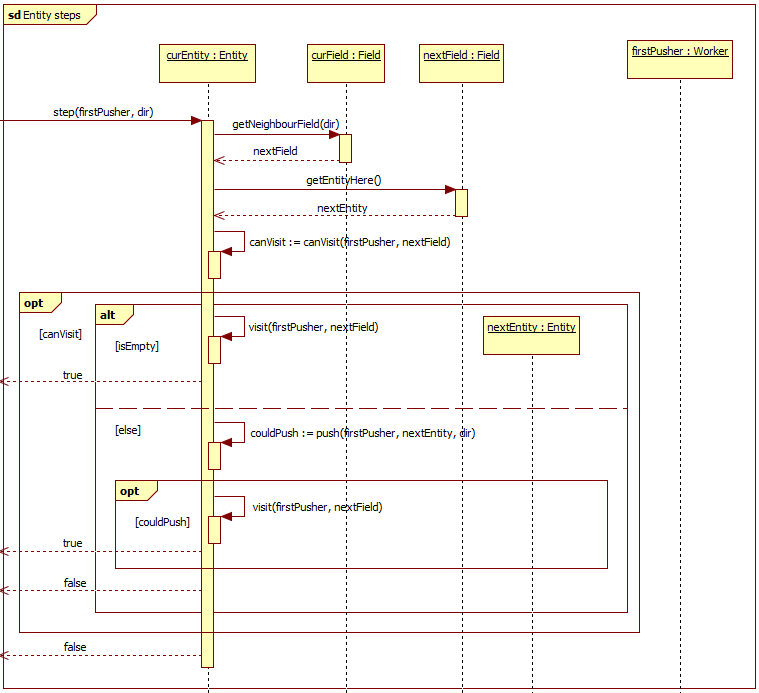
*Ilyenkor megpróbálunk ellépni az adott irányban.*

### Entitás új mezőre kerül



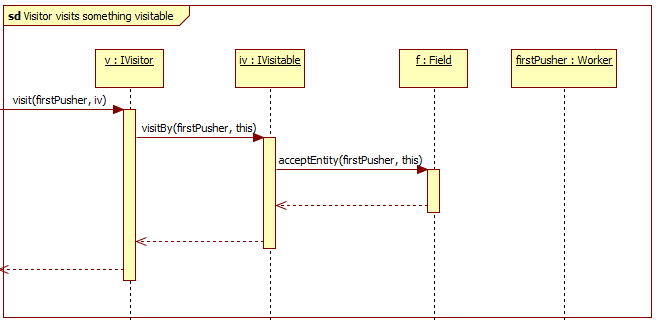
*A belépéskor adminisztráljuk mind az entitás mezőre vonatkozó, mind a mező entitásra vonatkozó referenciájának megváltozását.*

### Entitás lép



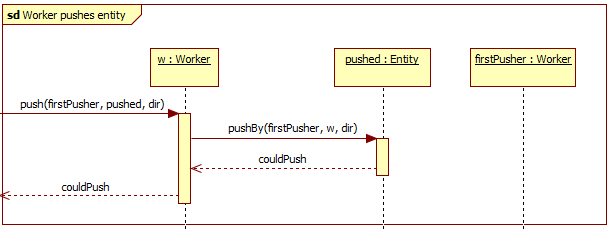
*Léptetéskor két dolgot kell ellenőriznünk. Elsőnek, hogy a célmező egyáltalán képes-e fogadni minket (pl.: Spawn csak a saját játékosát fogadja be), és ha ez fennáll, akkor azt, hogy jelenleg szabad-e, ugyanis, ha másik entitás már ott tartózkodik, akkor meg kell próbálnunk eltolni. Ha sikerült a tolás,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

### Visitáló objektum visitálhatót látogat



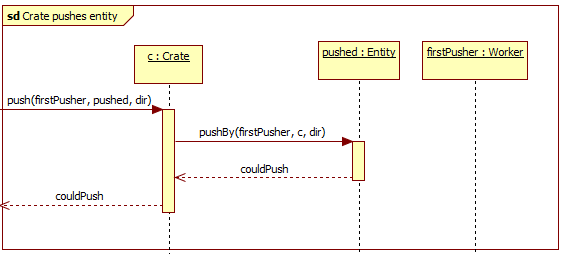
*A visitálás megtörténik minden lépés „valós” lépés előtt, vagyis amikor már biztos, hogy az adott entitás a mezőre lép, annak érdekében, hogy kiderüljön, milyen típusú entitás érkezik a mezőre.*

### Dolgozó entitást tol



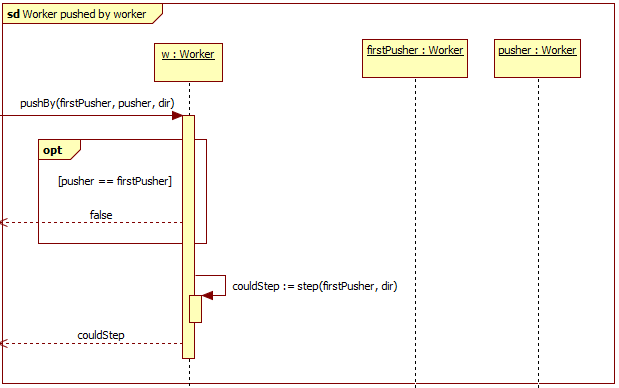
*Mini “Visitor” -> Dinamikus típussal való hívás miatt szükséges függvény.*

### Láda entitást tol



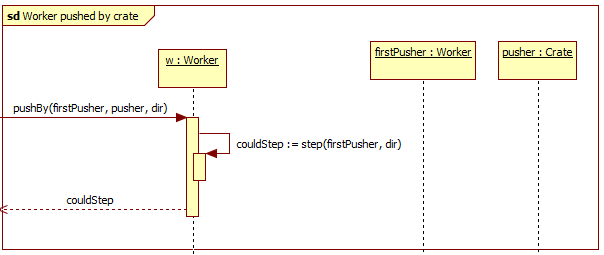
*Mini “Visitor” -> Dinamikus típussal való hívás miatt szükséges függvény.*

### Dolgozót dolgozó tol



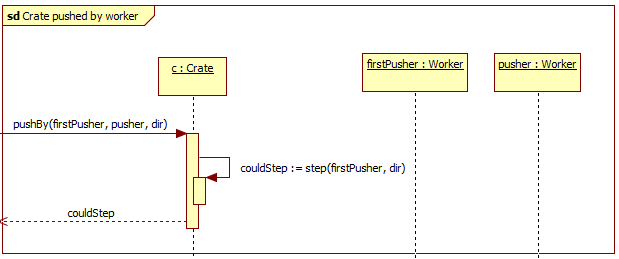
*Dolgozó dolgozót csak akkor tolhat, ha láncba van, ezért ellenőrizzük, hogy aki itt tol az nem az első-e. Ha az első akkor, visszatérünk hamissal, ezzel megtagadva a tolást, ha nem akkor megpróbálunk ellépni a megfelelő irányba és ennek sikerességével térünk vissza.*

### Dolgozót láda tol



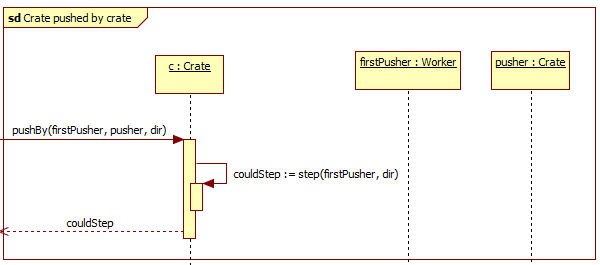
*Mivel láda mindig képes dolgozót tolni, ezért megpróbálunk ellépni a megfelelő irányba és ennek sikerességével térünk vissza.*

### Ládát dolgozó tol



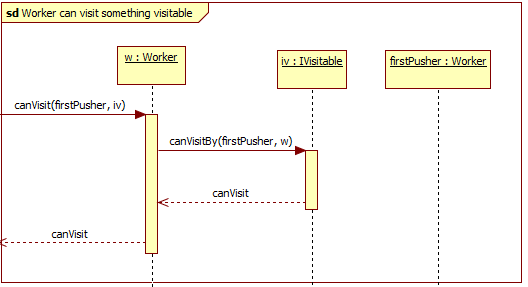
*Mivel dolgozó mindig képes ládát tolni, ezért megpróbálunk ellépni a megfelelő irányba és ennek sikerességével térünk vissza.*

### Ládát láda tol



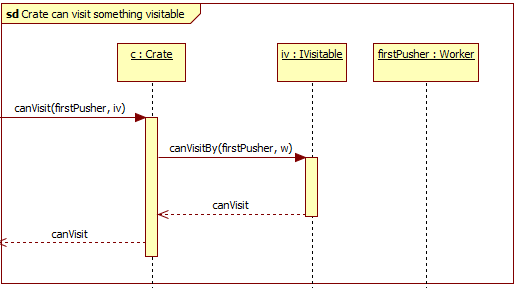
*Mivel láda mindig képes ládát tolni, ezért megpróbálunk ellépni a megfelelő irányba és ennek sikerességével térünk vissza.*

### Dolgozó léphet-e valahova



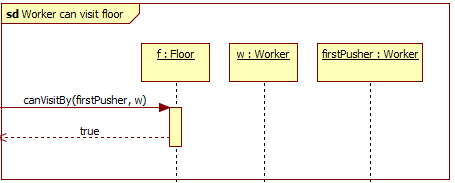
*“Visitor” -> Dinamikus típussal való hívás megvalósítása.*

### Láda léphet-e valahova



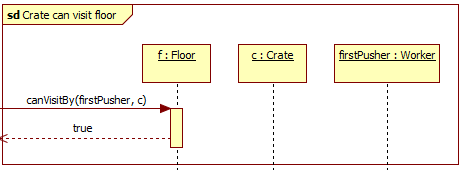
*“Visitor” -> Dinamikus típussal való hívás megvalósítása.*

### Dolgozó léphet-e padlóra



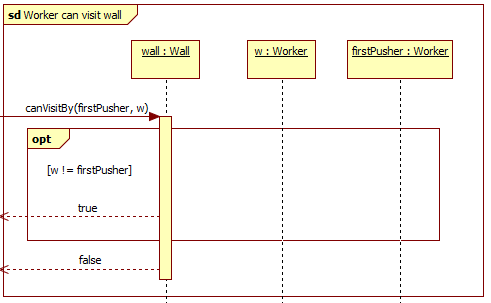
*Dolgozónak mindig megvan a lehetősége, hogy padlóra lépjen.*

### Láda léphet-e padlóra



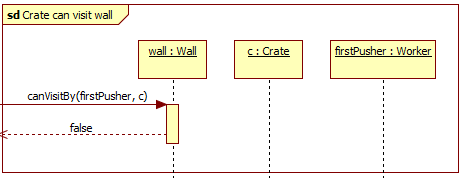
*Ládának mindig megvan a lehetősége, hogy padlóra lépjen.*

### Dolgozó léphet-e falra



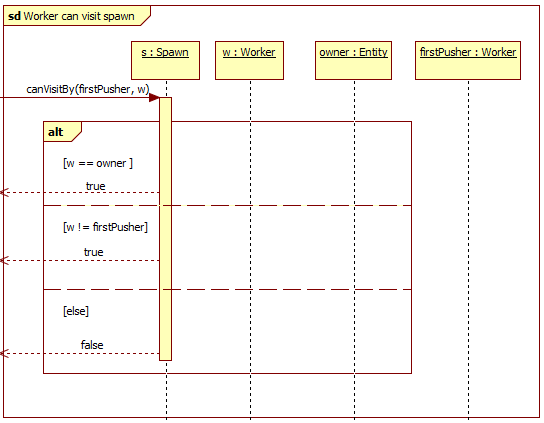
*Csak akkor léphet falra (azaz nyomódhat szét a falon a) dolgozó, ha láncban van, így ezt ellenőrizzük azzal, hogy nem a dolgozó a firstPusher, és ennek eredményét adjuk vissza.*

### Láda léphet-e falra



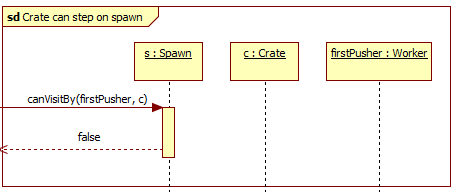
*Láda soha nem léphet falra.*

### Dolgozó léphet-e kiindulási helyre



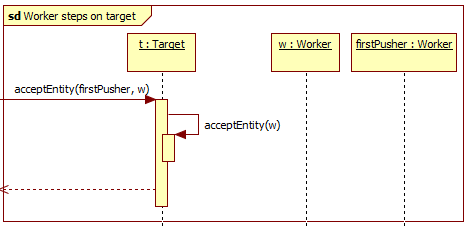
*Dolgozók közül csak annak van lehetősége a kiindulási helyre lépni, ami vagy annak a tulajdonosa, vagy láncban tolják (ekkor felnyomódik a mezőre, mint egy falra). Ezt a feltételt kell kiértékelnünk és visszatérnünk a válasszal.*

### Láda léphet-e kiindulási helyre



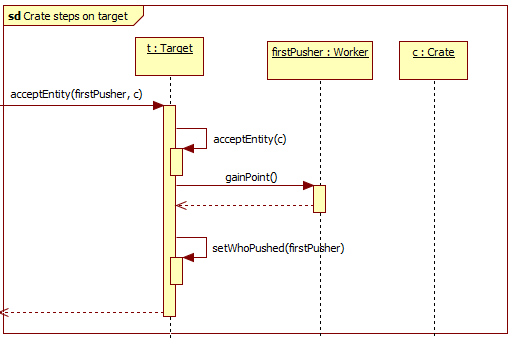
*Láda soha nem léphet Spawnra.*

### Dolgozó előírt helyre lép



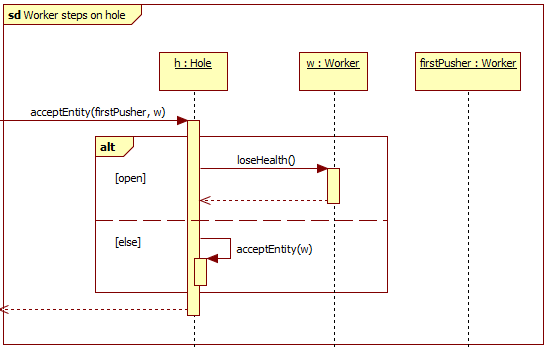
*Olyan, mintha a dolgozó egy padlóra lépne.*

### Láda előírt helyre lép és pontot ad



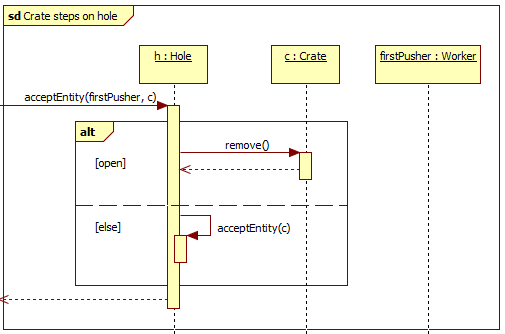
*Mikor egy láda kilép innen, a szokásos adminisztráció során elvesszük a kiosztott pontot. Tehát beléptetéskor szokásos adminisztráción kívül, két dolgot kell tennünk, kiosztani a pontot annak, aki miatt idekerült a láda, valamint referenciáját eltárolni, hogy később, ha kell, eltudjuk venni a pontot.*

### Dolgozó lyukra lép és életet veszít



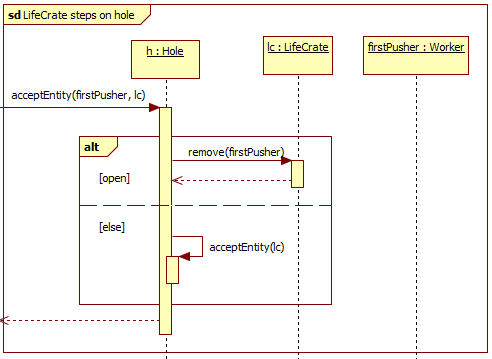
*Ekkor, ha nyitva van a lyuk akkor a lépés hatására életet kell veszítenie a dolgozónak, ha zárva akkor a szokásos műveleteket kell elvégezni.*

### Láda lyukra lép és eltűnik



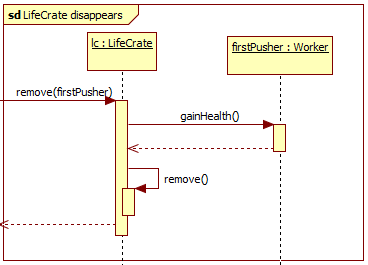
*Ekkor, ha nyitva van a lyuk akkor a lépés hatására megsemmisül a láda, ha zárva akkor a szokásos műveleteket kell elvégezni.*

### Szívecskés láda lyukra lép és eltűnik



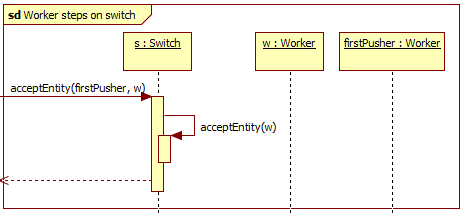
*Ekkor, ha nyitva van a lyuk akkor a lépés hatására megsemmisül a láda és aki ezt okozta kap egy életet (remove függvényen belül), ha zárva akkor a szokásos műveleteket kell elvégezni.*

### Szívecskés láda eltűnik és életet ad



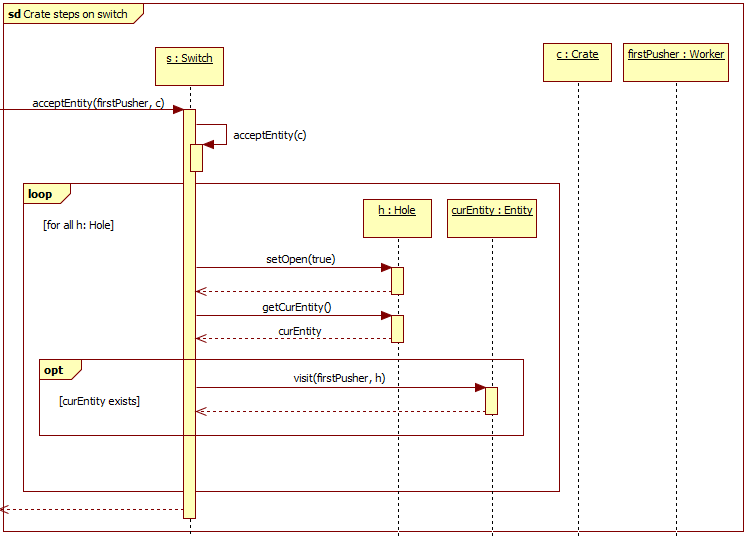
*Szívecskés láda esetén a megsemmisüléskor az azt okozó kap egy életet, majd a szokásos módon eltűnik a láda.*

### Dolgozó kapcsolóra lép



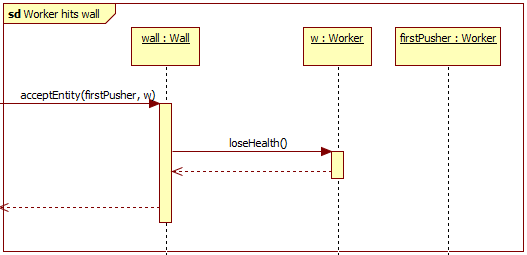
*A dolgozó nem képes kapcsolni a kapcsolót, ezért csak a szokásos lépések történnek.*

### Láda kapcsolóra lép és kapcsol



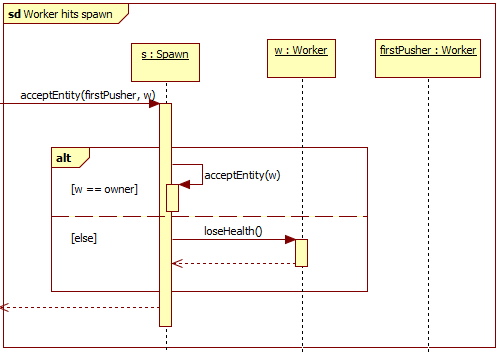
*Ha láda egy kapcsolóra lép, akkor a szokásos adminisztráción kívül, ki kell nyitni az összes hozzá tartozó lyukat, valamint az ott álló entityket le kell ejteni. Mivel a leejtést az a dolgozó okozza, aki a kapcsolóra tolta a ládát, így a leejtés megfelel annak, mintha épp beletolta volna a lyukba az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

### Dolgozó falnak ütközik és életet veszít



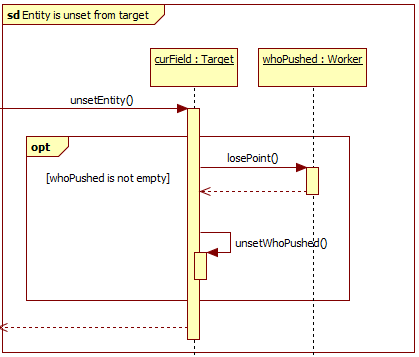
*A dolgozó, ha láncban tolják felnyomódik a falra és életet veszít.*

### Dolgozó kiindulási helyre lép, vagy nekiütközik és életet veszít



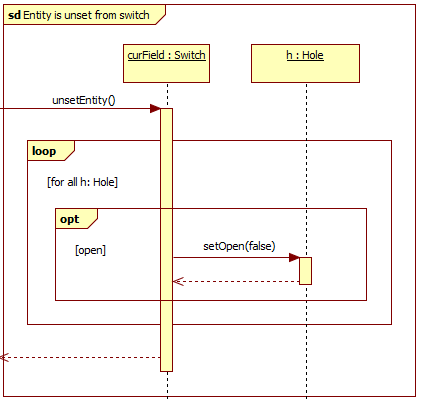
*A spawnra csak a tulajdonosa léphet rá, minden más dolgozó csak láncban tolva kerülhet ide, amiért életet veszít.*

### Entitás lekerül egy előírt helyről



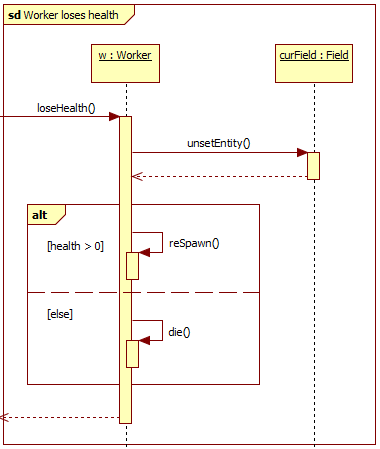
*A whoPushed referencia csak akkor van beállítva, ha valaki kapott pontot, mert ládát tolt ide, ezért ebben az esetben, el kell venni tőle ezt a pontot és kinullázni a referenciát.*

### Entitás lekerül egy kapcsolóról



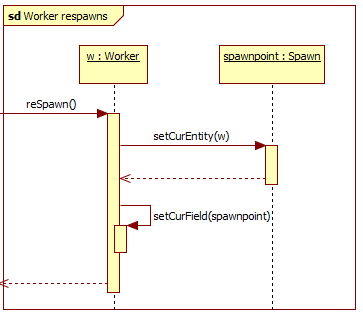
*Ha üressé válik a kapcsoló mező, akkor minden hozzá tartozó lyukat be kell zárnunk.*

### Dolgozó életet veszít



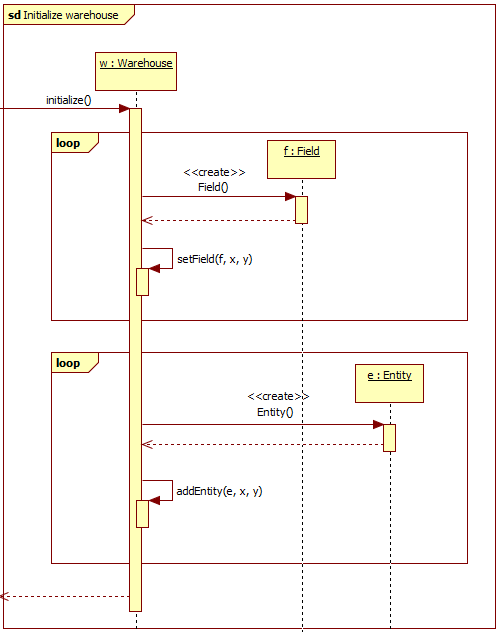
*Ha a dolgozó életet veszít akkor le kell venni a mezőről, ahol állt, csökkenteni az életeinek számát, majd az élet mennyiségének megfelelően vagy újrateremteni, vagy megölni.*

### Dolgozó respawn-ol



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

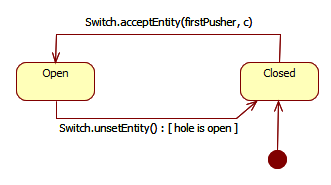
### Raktárépület inicializálása



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

## State-chartok

### 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.*

## 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 |