# Analízis modell kidolgozása

## Objektum katalógus

### Box

A Box osztály példányai egy-egy dobozt valósítanak meg. Rendelkeznek pillanatnyi pozícióval. Adott játékos tudja mozgatni őket, ha bizonyos feltételek teljesülnek. A cél, hogy ezeket az objektumokat egy cél helyre juttassuk.

### Direction

Egy enumerációt valósít meg, egy mező (Field) lehetséges szomszédainak irányát mutatja:

fel (Up), le(Down), balra (Left) vagy jobbra (Right).

### Field

A Field objektumok a játéktér mezőit alakítják, ezek összességéből áll össze egy pálya. Ilyen mezők lehetnek falak (Wall), szakadékok (Hole), kapcsolók (Switch), cél helyek (FinishPoint).

### FinishPoint

Magán a pályán elhelyezkedő célhelyek. Ide kell eljutnia egy-egy ládának (Box típusú objektumnak). Ha egy munkás idemozgatott egy ládát, a láda eltűnik, a munkást irányító játékos pedig pontot kap.

### Game

A játékot vezérli. Elindítja, illetve véget vet egy játéknak. Itt lehet kiválasztani a pályát, amin majd játszani lehet, illetve azt beállítani, hogy hányan fognak játszani.

### Hole

A szakadékokat (csapdákat) valósítja meg. Ha egy doboz kerül a szakadékra, az eltűnik. Ha egy játékos lép rá, a játékos kiesik a játékból (a munkás meghal). Van olyan speciális változata ennek a csapdának, amely csak akkor működik csapdaként, ha egy kapcsoló aktiválva van. Egyébként rendes pályaelemként funkcionál.

### Storage

Egy **N x M** –es tömböt tárol mezőkből (Field objektumokból). Ezek a mezők a pálya egyes celláit jelentik. Egy mezőn maximum egy doboz, munkás, oszlop (fal), szakadék, kapcsoló vagy cél hely állhat, melyeket a Storage számon is tart (tartalmaz).

### Switch

Ebből az osztályból lesz megvalósítva a kapcsoló. Egy speciális pályaelem, amelyre ládát rátolva egy bizonyos mező szakadékká (Hole) alakul át, bármely mozdítható dolog (Worker vagy Thing) megszűnik ha rajta van vagy ha rálép.

### Thing

A Worker és a Box alaposztálya. Minden Field objektumon maximum egy Thing objektum lehet rajta.

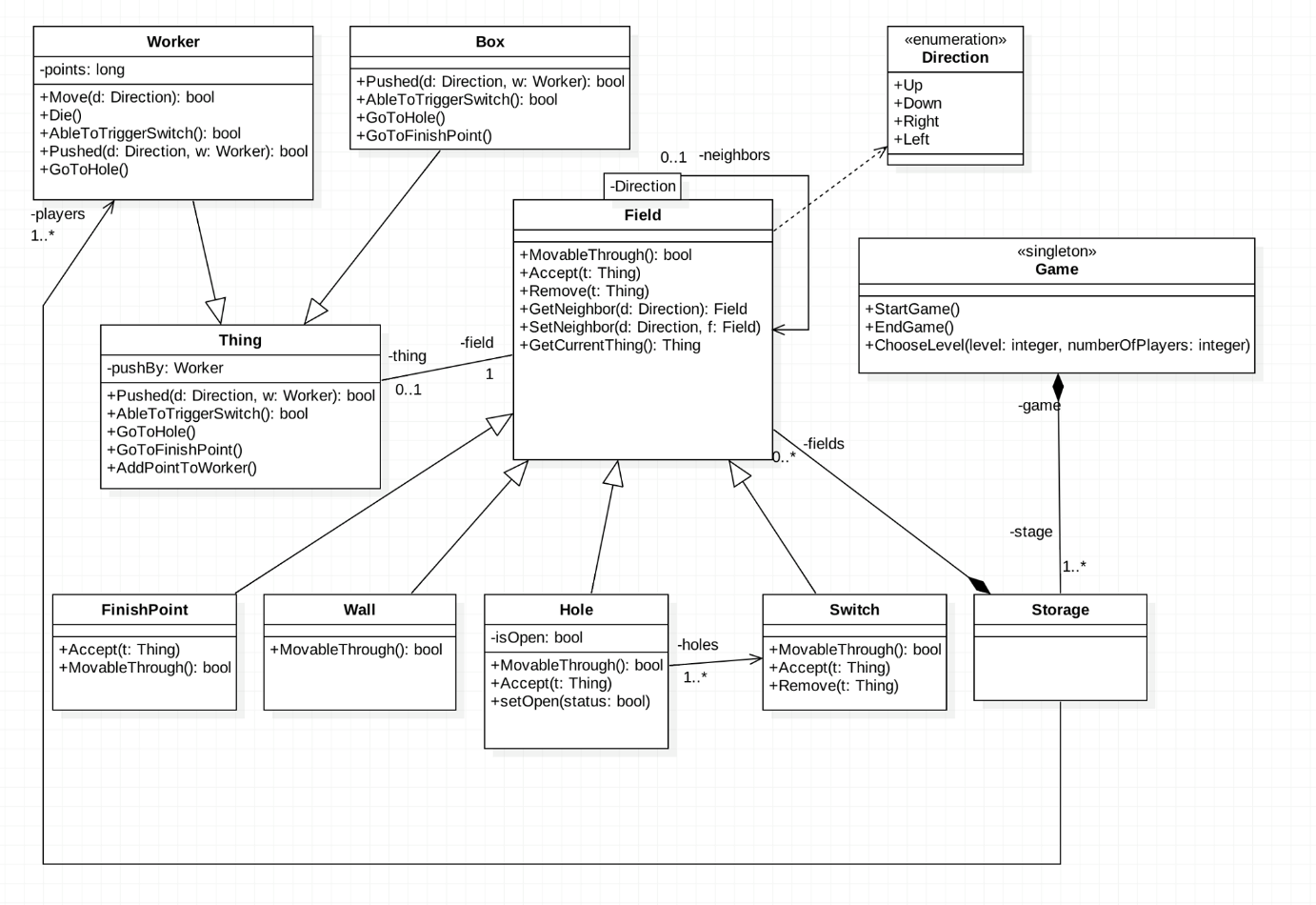
### Wall

A pálya egyik eleme, amely megakadályozza a játékos (Worker) vagy láda (Box) továbbmenetelét.

### Worker

A Worker osztály példányai egy-egy játékost reprezentálják, amelyek mozgatni tudják a ládákat. Minden egyes példány az aktuális pozíciót tárolja. Képesek jobbra, balra, fel, illetve lefelé közlekedni. Továbbá, a ládákat is e irányok felé tudják eltolni.

## Statikus struktúra diagramok



## Osztályok leírása

### Box

#### Felelősség

Egy pályán megtalálható, tologatható dobozok osztálya. Kapcsolót aktiváló szerepük lehet, cél helyre jutásukkor pedig eltűnnek. Nagy súlyuktól eredően képesek beszorult munkások összenyomására.

#### Ősosztályok

Thing  Box

#### Attribútumok

-

#### Metódusok

* **bool Pushed(Direction d, Worker w)**: egy munkás (**w**) valamilyen irányba(**d**) kezd tolni egy dobozt. Végig „zongorázódik” ez az esemény minden egyes doboz objektumra/mezőre a doboz után. Ha tudjuk tolni a doboz-sort, igazat kapunk vissza (és elkezdődik a folyamat), ellenkező esetben hamis, és nem történik semmi.
* **bool AbleToTriggerSwitch()**: Mindig igaz értéket ad vissza (avagy aktiválni tudja a kapcsolókat).
* **void GoToHole()**: a doboz egy szakadékhoz érkezett, megsemmisül
* **void GoToFinishPoint()**: a doboz egy cél helyhez érkezett, a doboz „megsemmisül” (eltűnik), a dobozt toló játékos kap egy pontot

### Direction

#### Felelősség

Enumeráció, egy mező lehetséges szomszédaihoz tartalmaz irányokat: fel (Up), le (Down), balra (Left) vagy jobbra (Right).

### Field

#### Felelősség

Egy pálya mezőit, és az egyes mezőtípusok funkcionalitását valósítja meg.

#### Attribútumok

* **Field neighbors[Direction]:** tárolja a szomszédos mezőket a lehetséges irányokba.
* **Thing[0..1] thing:** az éppen a mezőn álló dolgot raktározza.

#### Metódusok

* **bool MovableThrough()**: igaz/hamis értéket ad vissza attól függően, hogy rá lehet-e lépni a mezőre.
* **void Accept(Thing t)**: új dolog érkezik a mezőre, és ütközteti azt a már mezőn lévő dolgokkal.
* **void Remove(Thing t)**: eltávolítja az adott dolgot a mezőről.
* **Field GetNeighbor(Direction d)**: visszaadja az adott irányban található szomszédos mezőt.
* **void SetNeighbor(Direction d, Field f)**: beállítja az adott irányban szomszédos mezőt.
* **Thing GetCurrentThing()**: visszaadja az adott mezőn levő dolgot.

### FinishPoint

#### Felelősség

A pályán elhelyezkedő cél helyek funkciójáért felelős osztály.

#### Ősosztályok

Field  FinishPoint

#### Attribútumok

-

#### Metódusok

* **void Accept(Thing t)**: új dolog érkezik a cél helyre. Ha egy doboz az, a megfelelő lánc-reakció zajlik le. Ha munkás, nem történik semmi.
* **bool MovableThrough()**: igaz értéket ad vissza.

### Game

#### Felelősség

Vezérli a játék irányítását.

#### Attribútumok

* **Storage[1..\*] stage:** a játékhoz 1 – sok pálya tartozhat. Ezeket tárolja.

#### Metódusok

* **void StartGame()**: elindítja a játékot.
* **void EndGame()**: véget vet a játéknak.
* **void ChooseLevel(int level, int numberOfPlayer)**: kiválaszt egy pályát, amin a játék megkezdődhet. A pályát egy egész szám azonosítja, és megadandó, hogy hányan fogják játszani ezt a pályát.

### Hole

#### Felelősség

Szakadékot reprezentál a játéktérben. Elnyeli a ládákat és a játékosokat (megsemmisíti azokat). Léteznek olyan szakadékok, melyeknek két állapota van (nyitott, zárt).

#### Ősosztályok

Field  Hole

#### Attribútumok

* **bool isOpen:** a két-állapotos szakadékok jellemzője. Igaz/hamis értékkel határozható meg, hogy a szakadék szakadékként funkcionál, vagy pedig egyszerű mezőként. Olyan szakadék, melynek csak szakadék tulajdonsága van, ezen attribútuma mindig igaz értékű.

#### Metódusok

* **bool MovableThrough()**: igaz értéket ad vissza.
* **void Accept(Thing t)**: új dolog érkezik a szakadékhoz. Ha a szakadék nyitott állapotban van, a dolog megsemmisül (munkás/doboz).
* **void SetOpen(bool status)**: adott igaz/hamis állapotba állítjuk az **isOpen** attribútum értékét.

### Storage

#### Felelősség

Az egyes pálya mezőit tárolja a rajta tárolható objektumokkal.

#### Ősosztályok: -

#### Interfészek: -

#### Attribútumok

* **Field[0..\*] field:** a pálya mezőinek a tárolása.
* **Worker[1..\*] players:** a pályán játszó játékosok.

#### Metódusok

### Switch

#### Felelősség

A pályán fellelhető kapcsolók osztálya, amely a lyukak ki-, és bezárását végzi attól függően, hogy van-e az adott kapcsolón láda, avagy nincs.

#### Ősosztályok

Field → Switch

#### Interfészek

-

#### Attribútumok

* **Hole holes[1..\*]:** egy kapcsoló egy vagy több lyukat/szakadékot ki tudni nyitni és zárni.

#### Metódusok

* **bool MovableThrough()**: igaz értékkel tér vissza.
* **void Accept(Thing t)**: új mozdítható dolog érkezik a mezőre, amennyiben ez az új dolog egy láda, akkor az összes lyukat/szakadékot kinyitja.
* **void Remove(Thing t)**: eltávolítja az adott dolgot a mezőről, a lyukakat bezárja.

### Thing

#### Felelősség

A pályán mozdítható példányok(Box, Worker) ősosztálya. Meghatározza, hogy melyik mezőn van rajta, továbbá az, hogy melyik munkás tolja éppen az adott dolgot.

#### Ősosztályok

-

#### Interfészek

-

#### Attribútumok

* **pushBy Worker:** a munkást tárolja, ami az adott dolgot eltolja.
* **Field field:** egy thing mindig egy fielden van rajta.

#### Metódusok

* **bool Pushed(d: Direction, w: Worker):** az adott irányba eltolódhat-e a dolog (láda), a munkás által
* **bool AbleToTriggerSwitch():** amennyiben a dolog egy láda, akkor képes a kapcsolók átállítására, egyébként (munkás/játékos) nem
* **GoToHole():** az adott dolog reakciója, amennyiben egy lyuk pályaelemre lép
* **GoToFinishPoint():** a cél helyre lépett egy dolog, a dolog típusától történik ez az esemény
* **AddPointToWorker():** pontot ad a játékosnak

### Wall

#### Felelősség

A mozdítható dolgok blokkolásáért felelős osztály.

#### Ősosztályok

Field → Wall

#### Interfészek

-

#### Attribútumok

**-**

#### Metódusok

* **bool MovableThrough():** hamis értékkel tér vissza, ugyanis a fal nem engedi át a mozgatható dolgokat.

### Worker

#### Felelősség

Ezeket a példányokat irányítja a játékos.

#### Ősosztályok

Thing → Worker

#### Interfészek

-

#### Attribútumok

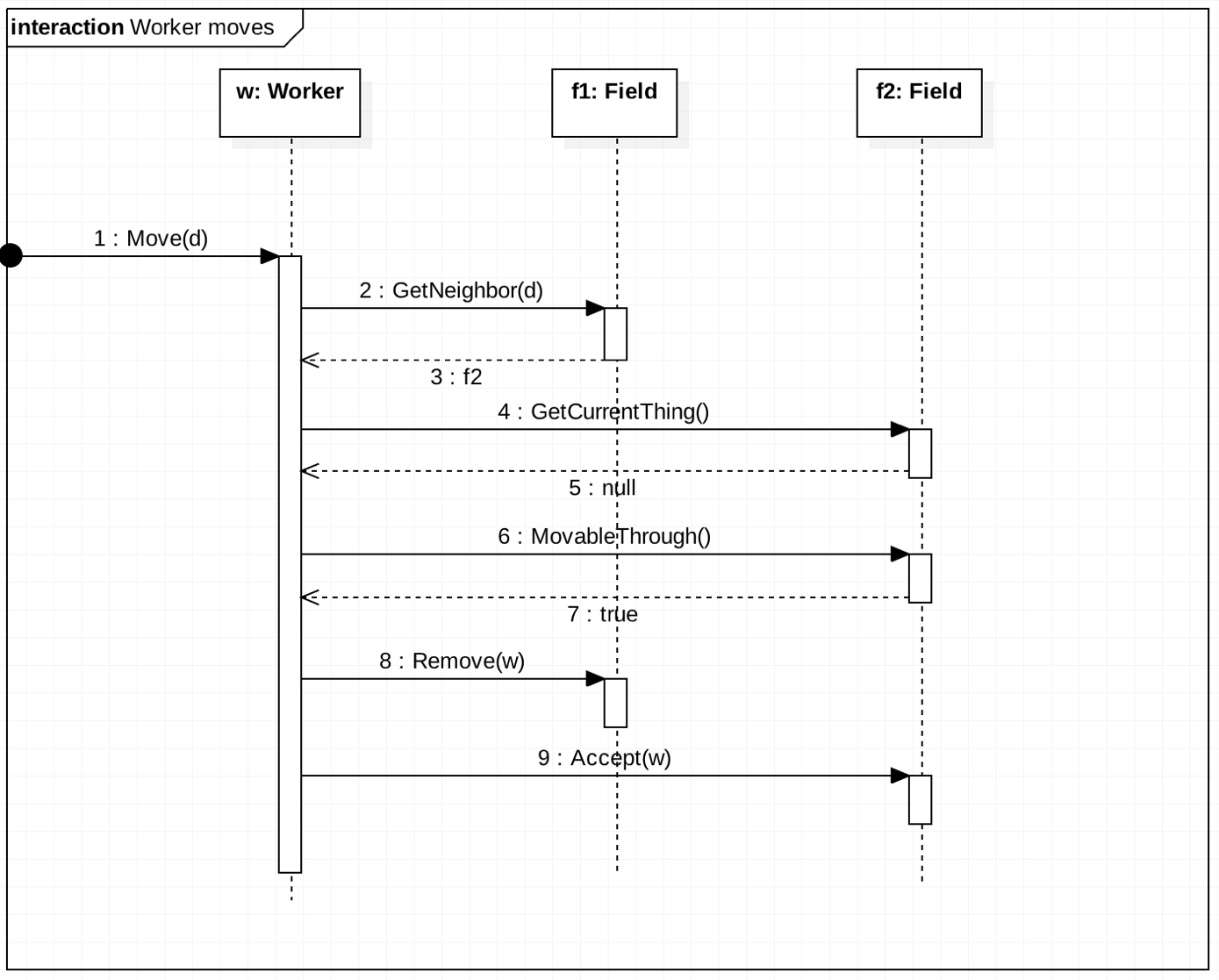
* **long Points:**  minden Worker példánynak létezik ez az attribútuma, amely az eddigi cél helyekre (FinishPoint) betolt ládákat tartja számon.

#### Metódusok

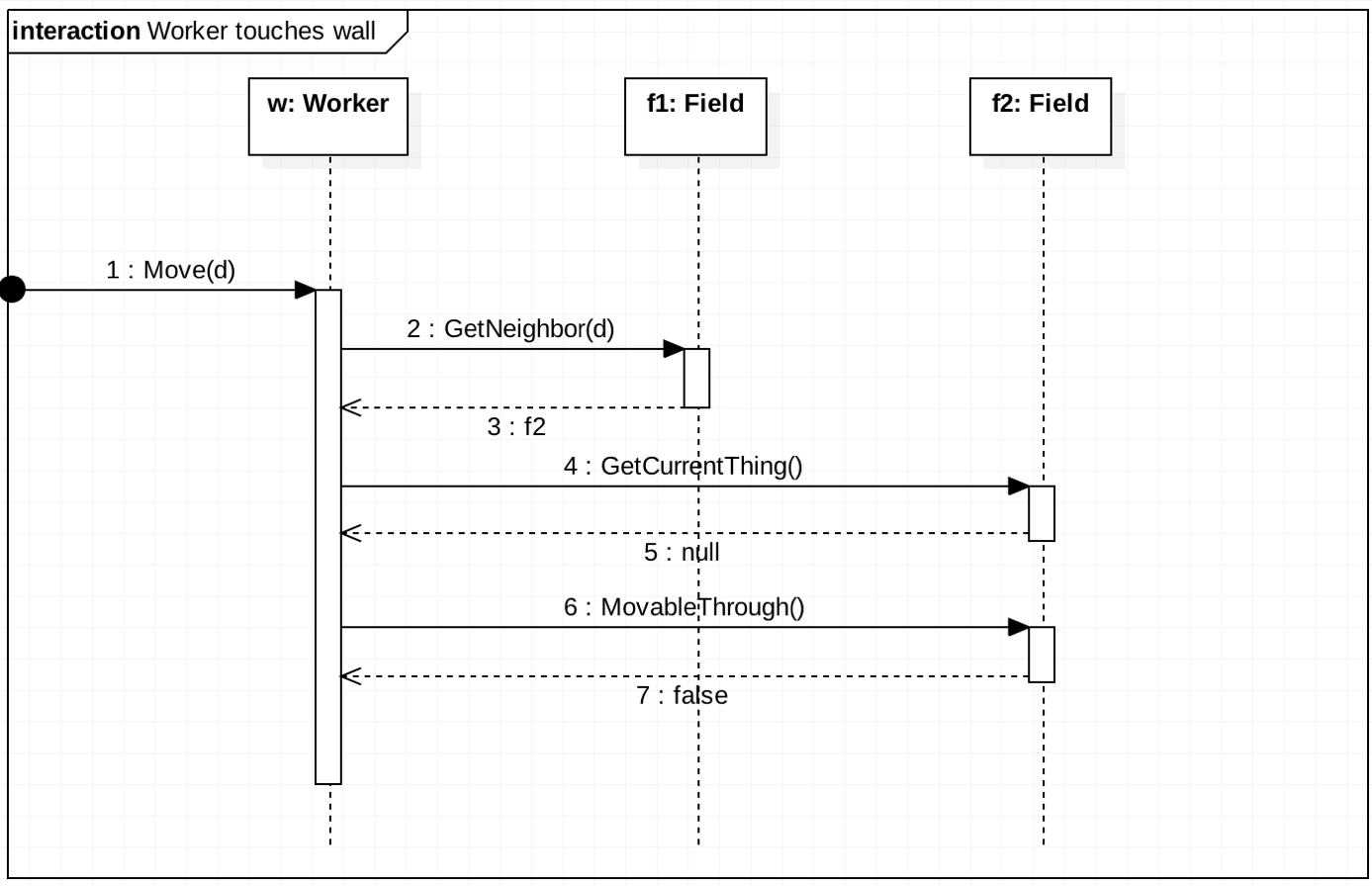
* **bool Move(d: Direction):** visszaadja a paraméterben megadott irányba a lépési képességét.
* **Die():** a játékos meghal, elveszíti a játékot (rálépett egy aktív lyukra vagy összepréselődött egy láda és egy fal között).
* **bool AbleToTriggerSwitch():** hamis válasszal tér vissza, mivel az objektumunk nem láda.
* **bool Pushed(d: Direction, w:Worker):** az adott irányba eltolódhat-e egy másik munkás láda eltolása által
* **GoToHole():** a játékos a szakadékra lépett, számára véget ért a játék és a játékos megszűnik létezni

## Szekvencia diagramok

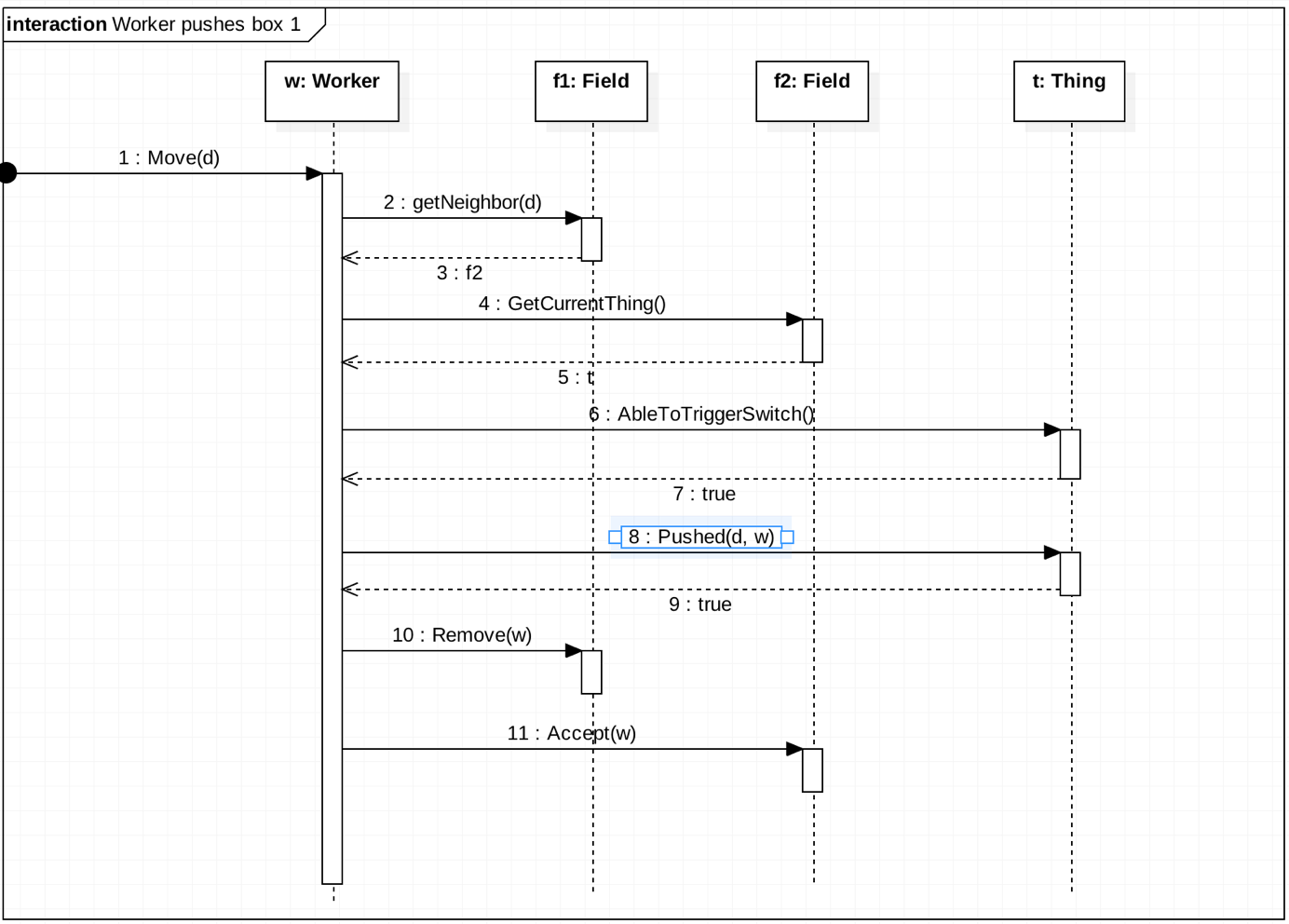
### Worker moves



### Worker touches wall

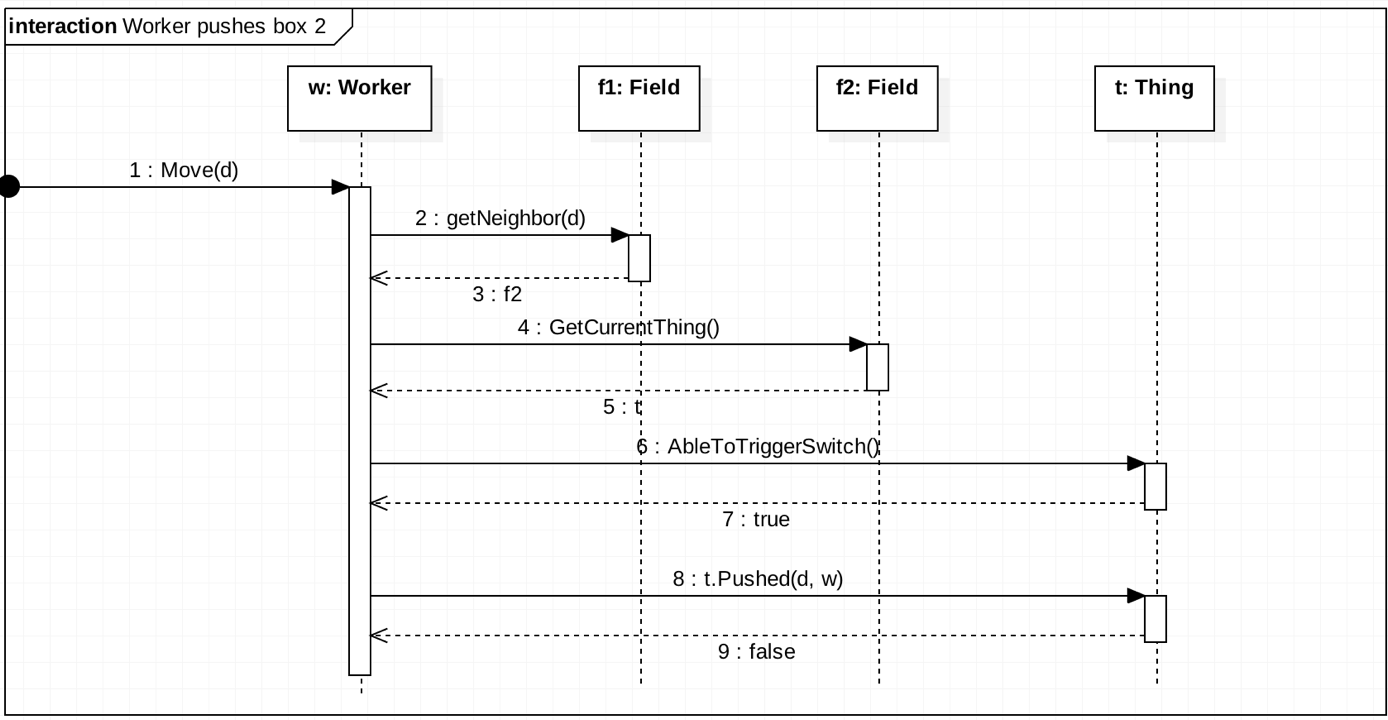


### Box pushes box 1

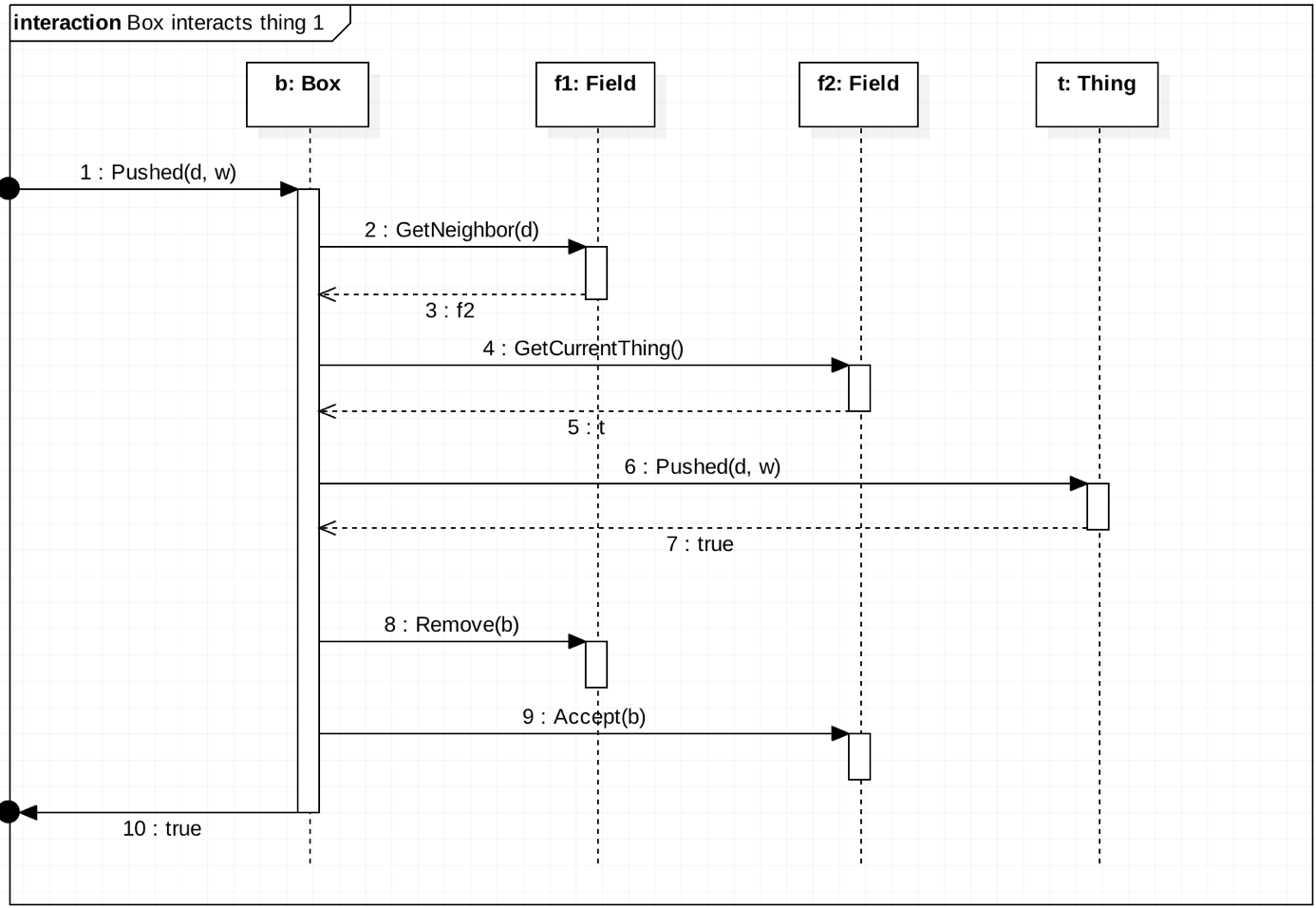


Két játékos/munkás nem tudja eltolni egymást közvetlenül, ezért az AbleToTriggerSwitch meghívódik, hogy meg lehessen különböztetni a dolgokat, hogy vajon azok ládák vagy munkások.

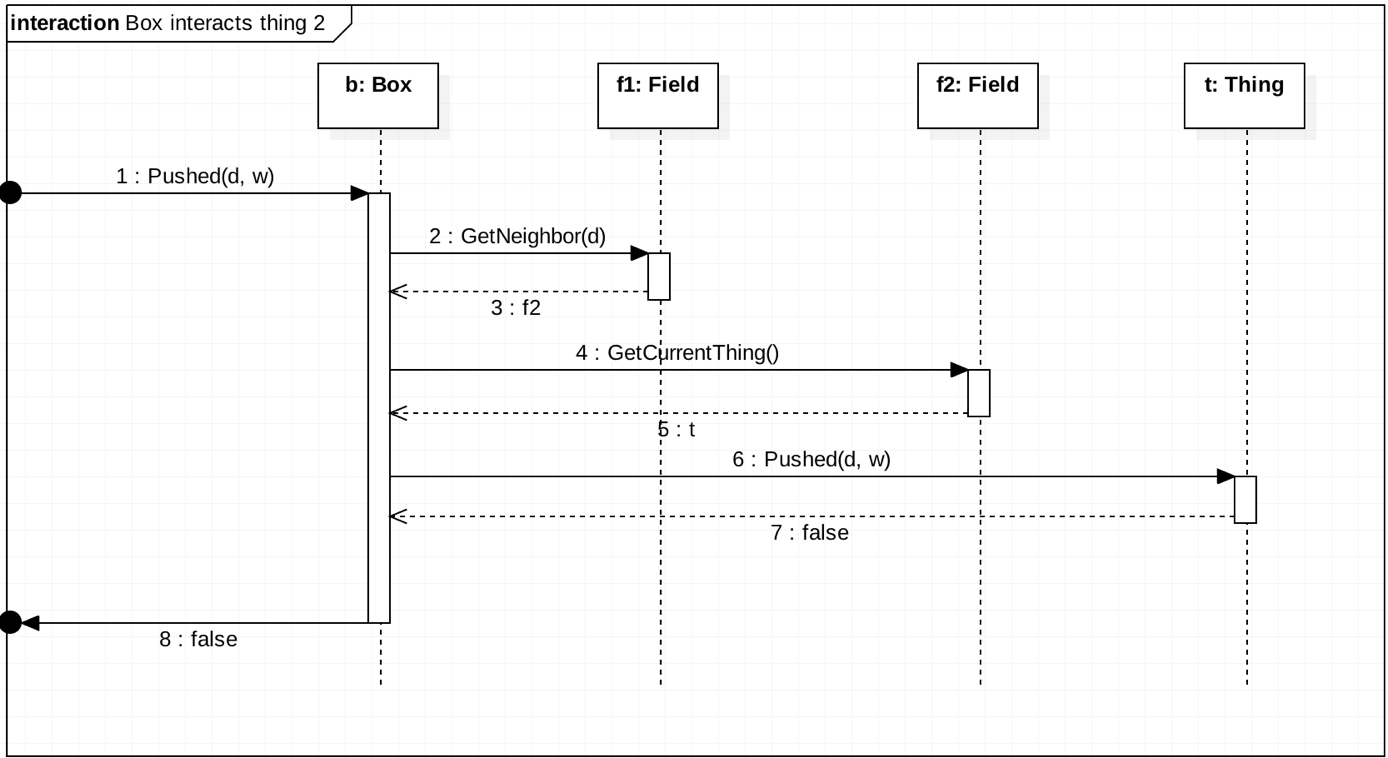
### Box pushes box 2



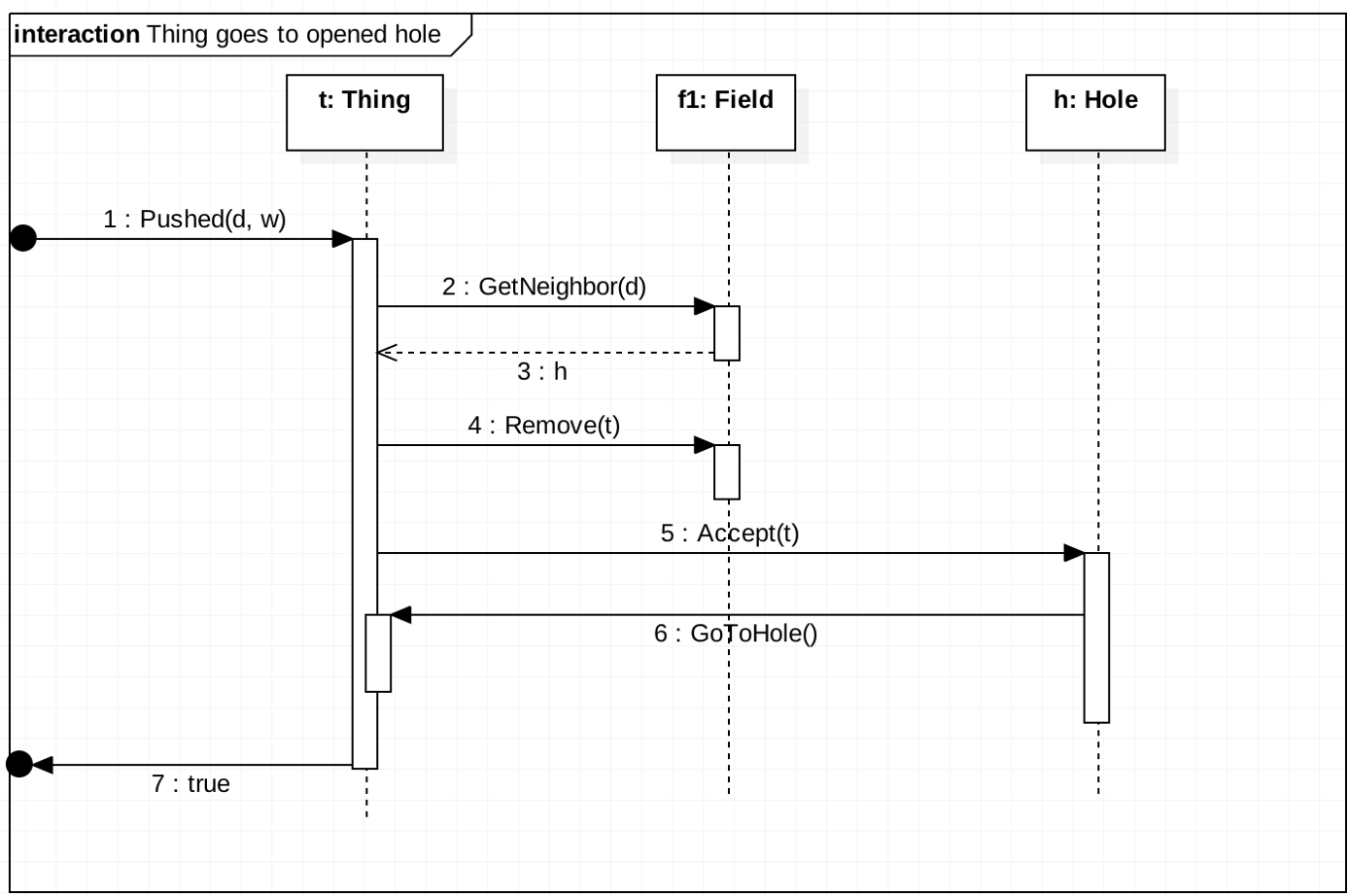
### Box interacts thing 1



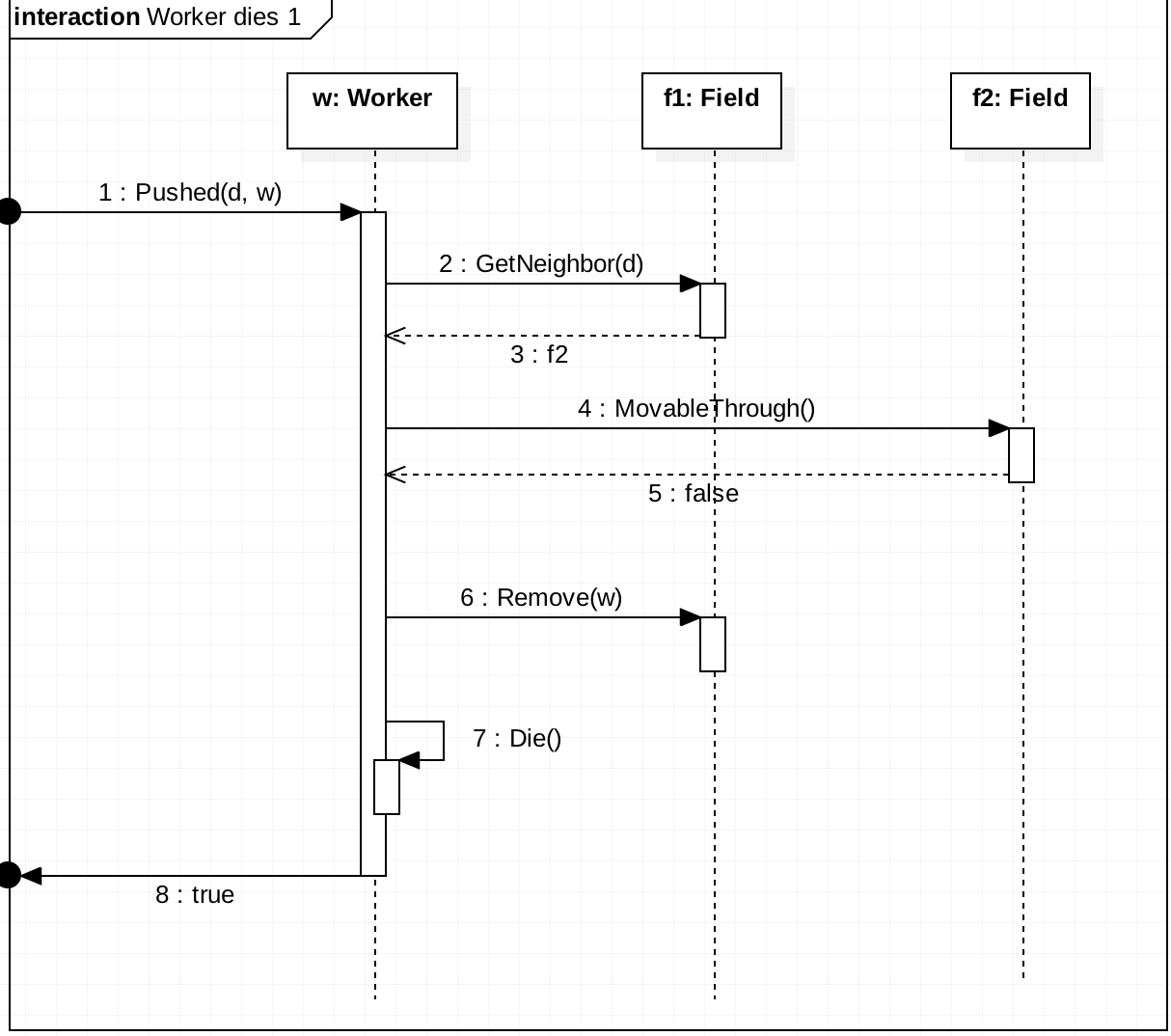
### Box interacts thing 2



### Thing goes to opened hole

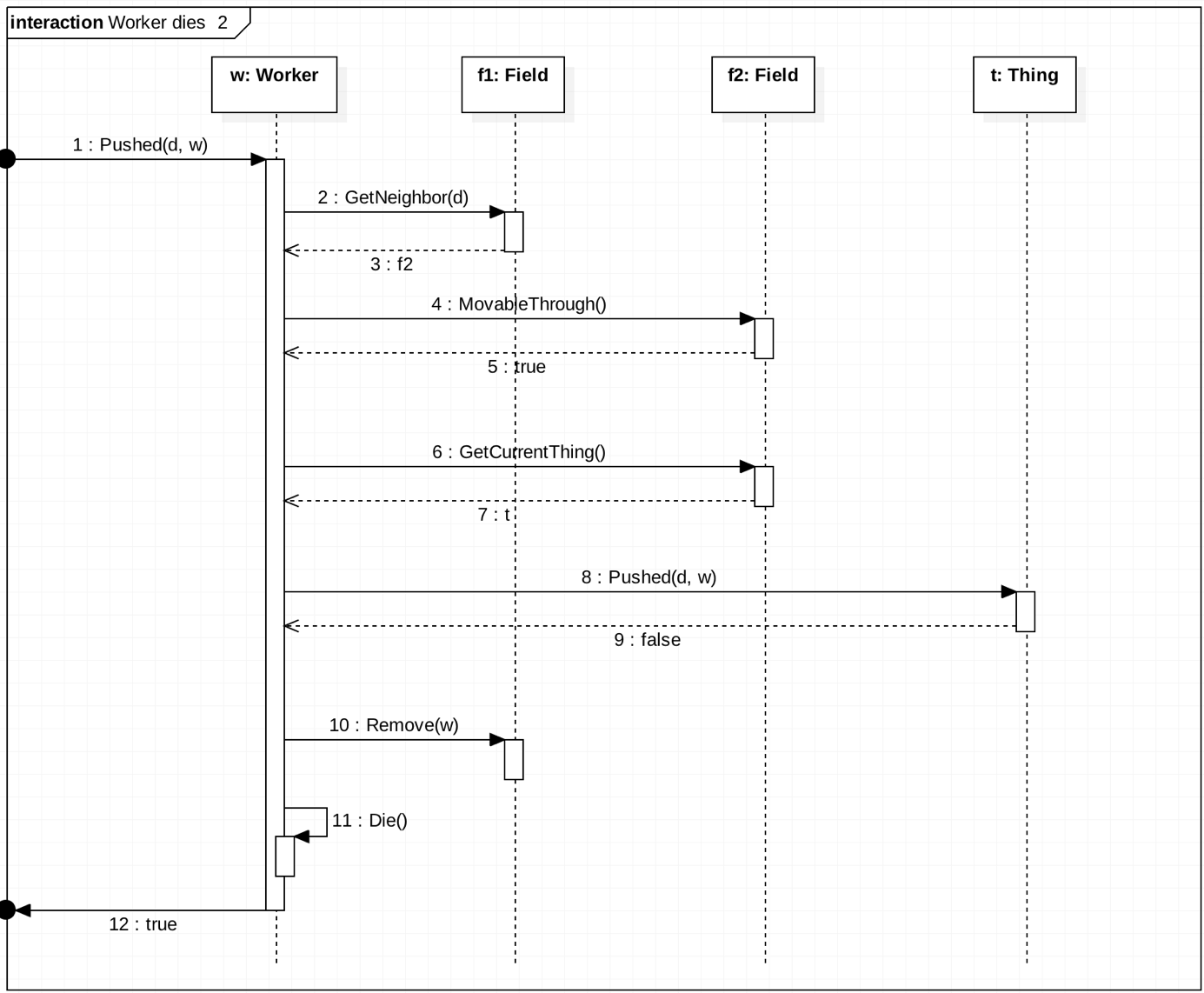


### Worker dies 1



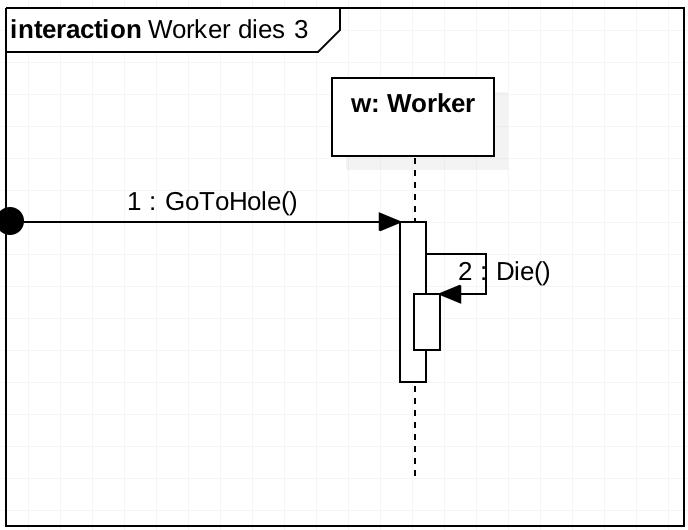
Egy falhoz tolódik

### Worker dies 2



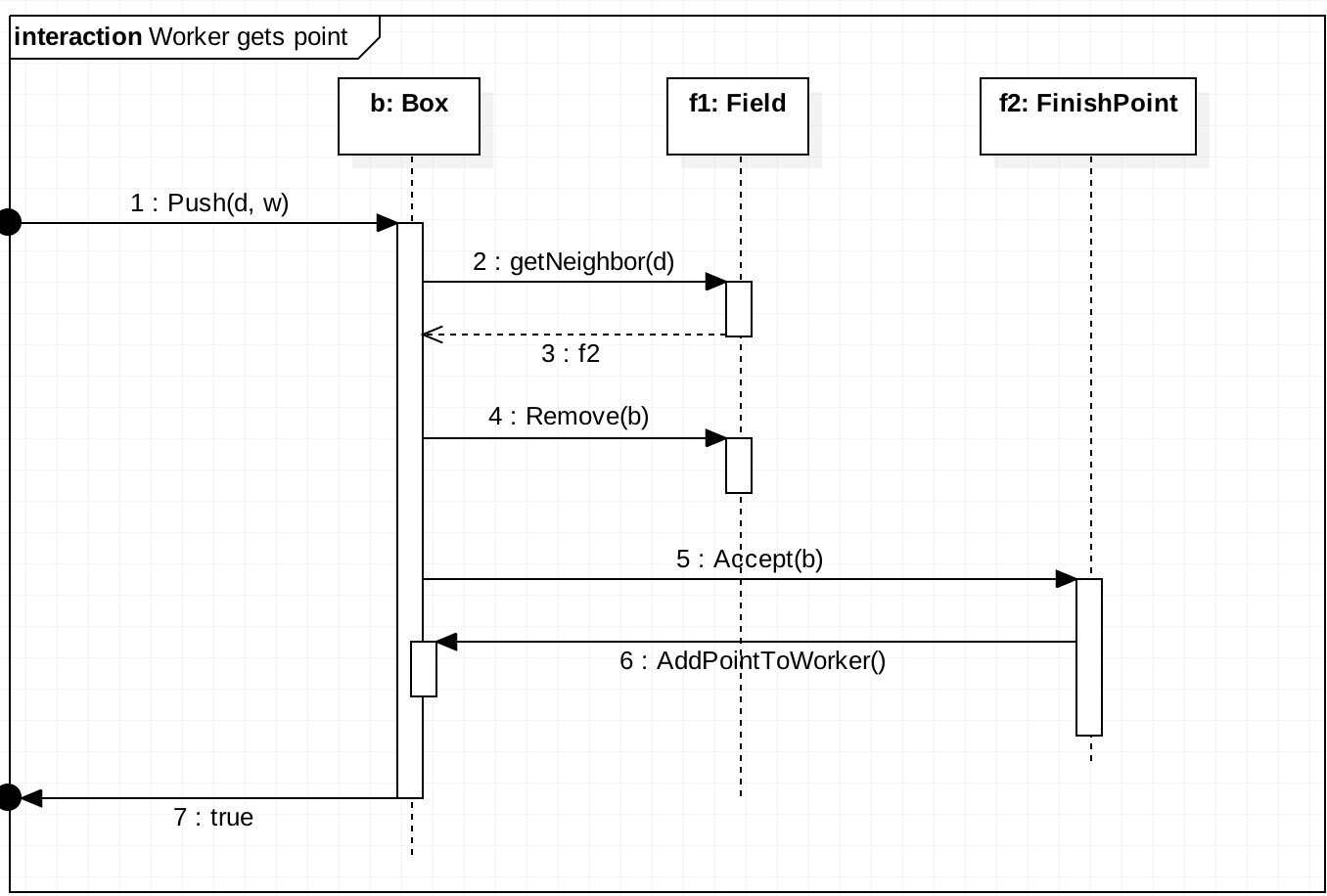
Két láda között van és azok közül az egyik rátolódik

### Worker dies 3

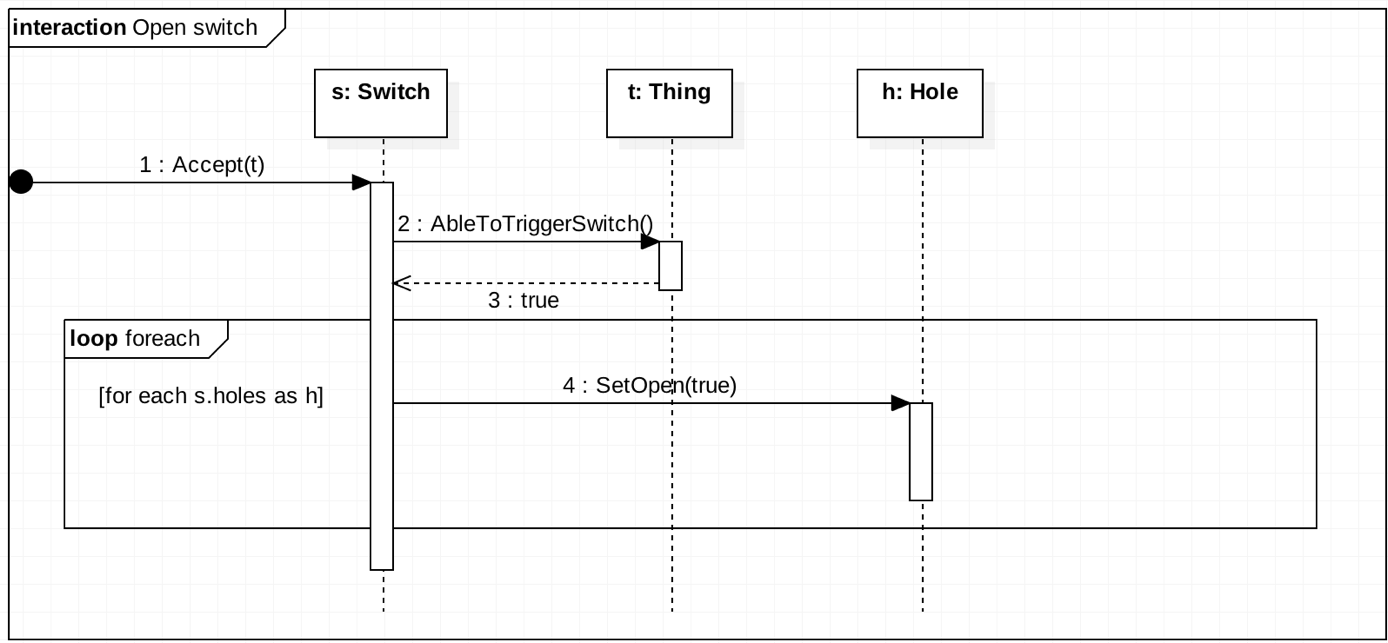


Egy lyukra lép

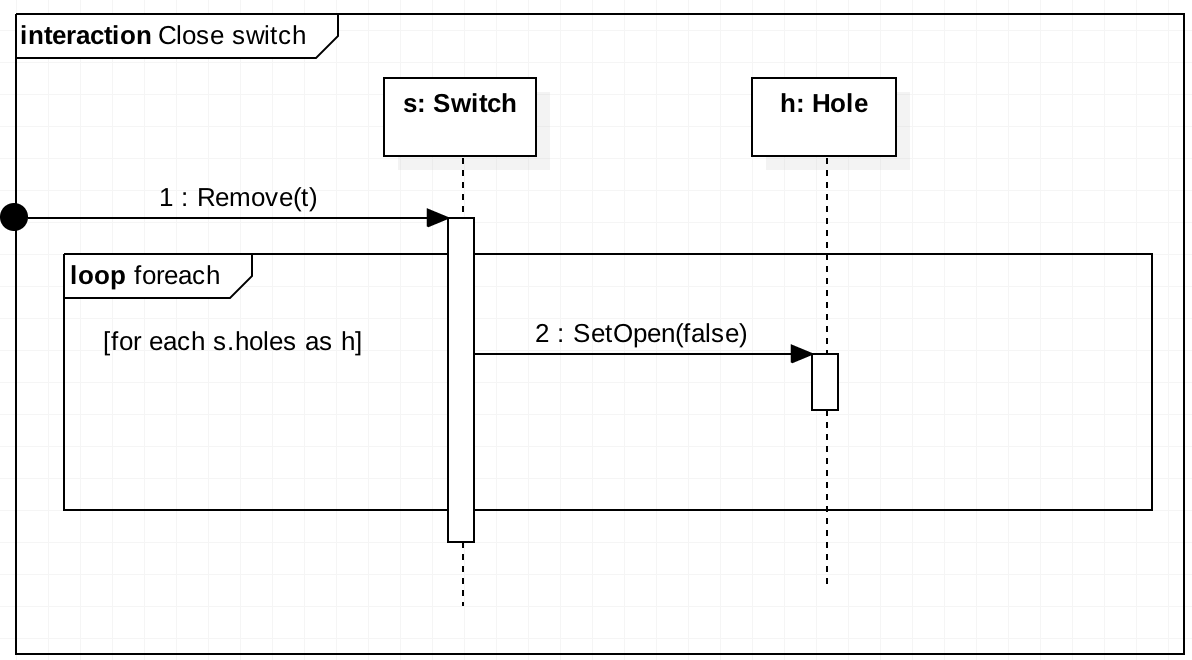
### Worker gets point



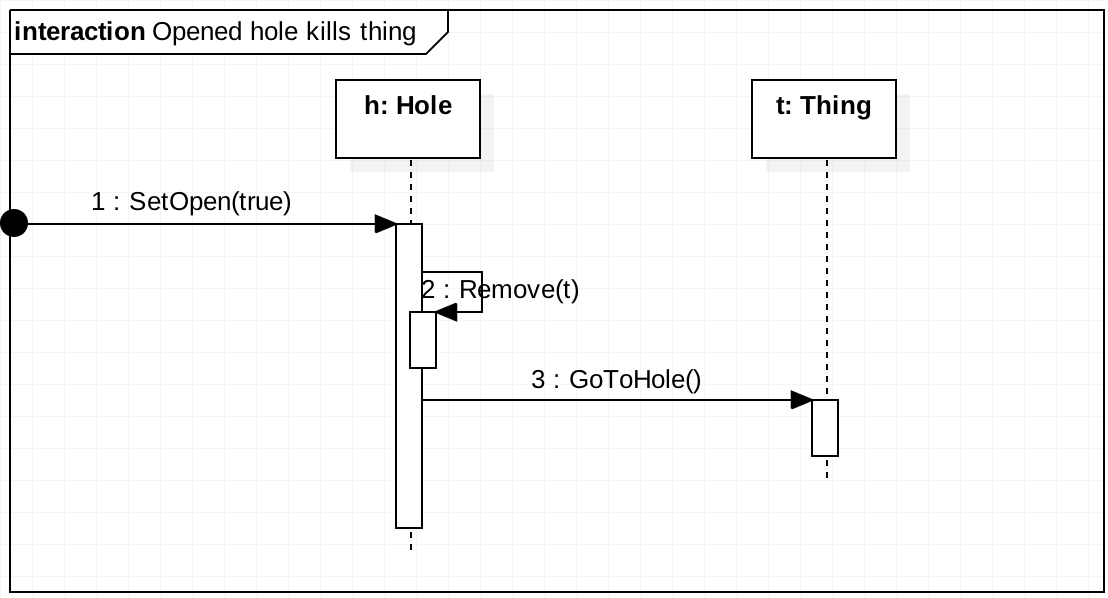
### Open switch



### Close switch

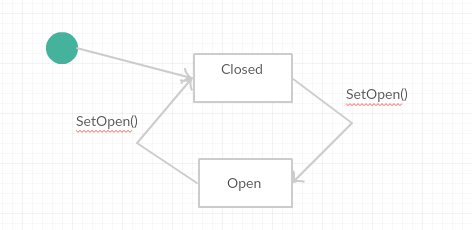


### Opened hole kills thing



## State-chartok

A lyukak állapota, amennyiben rá-, illetve letolódik róla egy láda.



## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2018. 02. 23. 14:30 | 2 óra | Vu | Structurális diagramok készítése |
| 2018. 02. 23. 10:30 | 2 óra | Le, Vu | Szekvencia diagramok készítése |
| 2018. 02. 24. 10:30 | 3,5 óra | Kállai | Objektum modellen kisebb változtatások, Osztály leírása |
| 2018. 02. 24. 19:00 | 2,5 óra | Hoang | Osztályleírás, objektum katalógus folytatása |