1. **Analízis modell kidolgozása**
   1. **Objektum katalógus**
      1. **Crate**

Doboz, amit a munkás tol. Ha kapcsolón van, akkor aktiválja azt. Ha a helyére kerül, akkor eltűnik. Eltolhat más dobozokat és munkást.

* + 1. **EndTile**

Rakódóterület. Az a típusú Tile ahova egy ládát eljuttatva a játékos pontot kap.

* + 1. **Hole**

A lyuk egy olyan pálya elem aminek 2 állapota van: nyitott és zárt, amennyiben nyitott és játékos vagy doboz kerül rá akkor azt eltünteti. Lyukak állapotát kapcsolók váltják, de vannak olyan eset mikor a lyuk nincs kapcsolóhoz kötve ezáltal állandóan nyitott állapotban van.

* + 1. **Map**

A játék egy pályáját valósítja meg. Tartalmazza a játék összes mezőjét, valamint felel a játékosok pontszámainak és a hátralévő idő nyilván tartásáért.

* + 1. **Switch**

Kapcsoló, amely irányítja a hozzá tartozó lyukat. Ha van rajt doboz, akkor aktív, ha nincs, akkor inaktív.

* + 1. **Thing**

Absztrakt osztály, amely az általános a pályán lévő nem statikus elemet reprezentálja.

* + 1. **Tile**

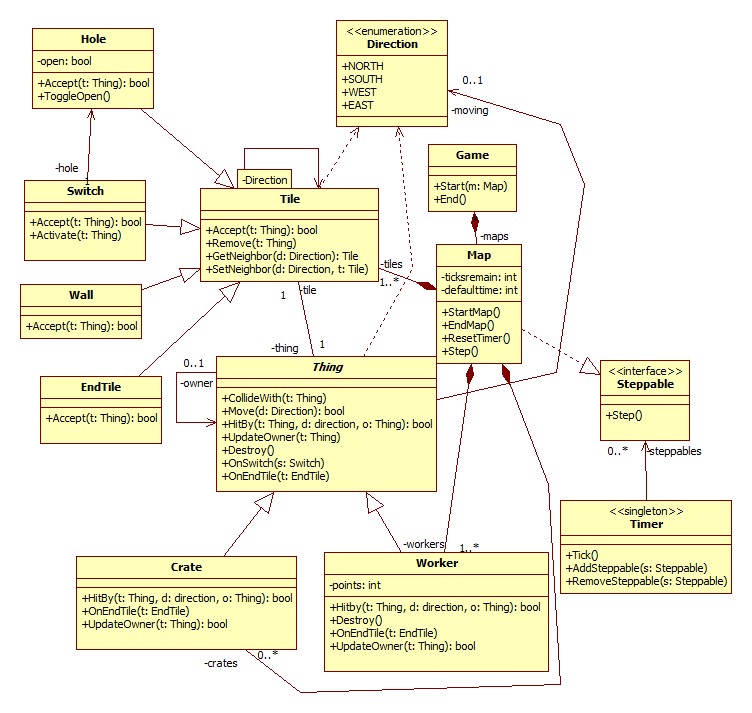
Ez az objektum reprezentálja a pálya egy mezőjét, amin lehet láda, munkás.

* + 1. **Wall**

Egy olyan pálya elem ami elérthetetlenné teszi azt a mezőt amin elhelyezkedik, kötelezően megjelenik a pálya szélét képző mezőkön, de előfordulhat belső mezőkön is.

* + 1. **Worker**

A munkás a játékos által irányított ‘karakter’, képes lépni, tolni ládákat, beleesni egy lyukba (ekkor meghal).

* 1. **Statikus struktúra diagramok**
  2. **Osztályok leírása**
     1. **Crate**
        + **Felelősség**

A pályán lévő ládákat reprezentáló osztály.

* + - * **Ősosztályok**

Thing

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**

Nincs.

* + - * **Metódusok**
* **bool HitBy(t: Thing,d: Direction,o: Thing)**: közli egy másik Thinggel hogy nekiment egy másik Thing, és, hogy mozduljon el ha tud.
* **void OnEndTile(t: EndTile)**: Pontot ad a helyretoló munkásnak és eltűnik a pályáról.
* **bool UpdateOwner(newvalue:Thing)**: Beállítja az owner attribútum értékét a newrvalue-ra.
  + 1. **Direction**
       - **Felelősség**

Enumeráció, amelyben a játékban lévő irányok vannak tárolva.

* + - * **Ősosztályok**

Nincs.

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**
* **+NORTH:** Észak
* +**SOUTH**: Dél
* **+WEST:** Nyugat
* +**EAST**: Kelet
  + - * **Metódusok**

Nincs.

* + 1. **EndTile**
       - **Felelősség**

Azt a Tilet reprezentálja, amelyre rá kell tolni a dobozokat.

* + - * **Ősosztályok**

Tile

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**

Nincs.

* + - * **Metódusok**
* **bool Accept(t:Thing):** Ha doboz kerül rá, akkor eltűnik az adott doboz és a munkás, aki rátolta pontot kap. Ha nem doboz kerül rá, akkor sima tileként viselkedik.
  + 1. **Game**
       - **Felelősség**

A játék egészét reprezentáló osztály.

* + - * **Ősosztályok**

Nincs.

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**
* **-maps: Map[\*]:** A játékot alkotó pályák összessége.
  + - * **Metódusok**
* **void Start():** Elindítja a játékot.
* **void End():** Leállítja a játékot.
* **Map[] GetMaps():** Visszaadja a maps attribútum értékét.
* **void SetMaps(newvalue:Map[]):** Beállítja a maps attribútum értékét a newrvalue-ra.
  + 1. **Hole**
       - **Felelősség**

A pályán lévő lyukakat reprezentáló osztály.

* + - * **Ősosztályok**

Tile

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**
* **-open:bool:** Nyitott-e a lyuk. (Ha nem nyitott, akkor sima tileként viselkedik.)
  + - * **Metódusok**
* **bool Accept(t: Thing):** Ha a lyuk nem nyitott, akkor sima tileként viselkedik, ha nyitott, akkor elnyeli a rálépő Thinget.
* **bool GetOpen():** Visszaadja az open attribútum értékét.
* **void ToggleOpen():** Negálja az open attribútum értékét és biztosítja a láda vagy munkás leesését ha éppen kinyitották és rajta állnak.
  + 1. **Map**
       - **Felelősség**

Egy pályát reprezentáló osztály.

* + - * **Ősosztályok**

Nincs.

* + - * **Interfészek**

Steppable

* + - * **Attribútumok**
* **-tiles: Tile [1..\*]:** A pályát alkotó tileok.
* **-workers: Worker [1..\*]:** A pályán lévő munkások.
* **-ticksremain: int**: Visszamaradó idő, tickekben.
* **-defaulttime: int**: Alapértelmezett visszamaradó idő, tickekben.
  + - * **Metódusok**
* **void StartMap():** Inicializálja a pályát.
* **void EndMap():** Miután eltűnt minden Crate, ez a függvény hívódik meg. Elindítja a következő pályát (ha van).
* **Tile[] GetTiles()**: Visszaadja a tiles attribútum értékét.
* **void SetTiles(newvalue:Tile[]):** Beállítja a tiles attribútum értékét a newrvalue-ra.
* **Worker[] GetWorkers():** Visszaadja a workers attribútum értékét.
* **void SetWorkers(newvalue:Worker[]):** Beállítja a workers attribútum értékét a newrvalue-ra.
* **void SetTicksRemain (i: int):** Beállítja a visszamaradó időt.
* **int GetTicksRemain():** Lekérdezi, mennyi idő van hátra.
* **void ResetTimer()**: Visszaállítja a visszamaradó időt az alapértlmezettre.
* **int getDefaultTime()**: Lekérdezi az alapértelmezett visszamaradó időt.
* **void setDefaultTime(t: int)**: Beállítja az alapértelmezett visszamaradó időt.
  + 1. **Steppable**
       - **Felelősség**

Interfész, ami a lépni képes dolgokat reprezentálja a játékban.

* + - * **Ősosztályok**

Nincs

* + - * **Metódusok**
* **void Step():** E függvény meghívásakor lép.
  + 1. **Switch**
       - **Felelősség**

A pályán lévő Hole-okat kapcsoló kapcsolókat reprezentáló osztály.

* + - * **Ősosztályok**

Tile

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**
* **-hole: Hole:** Az a Hole amelyet az adott Switch kapcsol.
  + - * **Metódusok**
* **bool Accept(t:Thing):** Befogadja a rálépő Thinget, mint a rendes Tile.
* **Hole GetHole():** Visszaadja a Hole attribútum értékét.
* **void SetHole(newvalue:Hole):** Beállítja a Hole attribútum értékét a newrvalue-ra.
* **void Activate(t: Thing):** Megnyomja a kapcsolót ha Crate érkezik rá.
  + 1. ***Thing***
       - **Felelősség**

Absztrakt osztály, amely az általános a pályán lévő nem statikus elemet reprezentálja.

* + - * **Ősosztályok**

Nincs.

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**
* **-tile:Tile:** Ezen a tileon helyezkedik el a Thing.
* **-owner:Thing:** Ez a Thing mozgatja éppen.
* **-moving: Direction**: Ebbe az irányba mozog a Thing.
  + - * **Metódusok**
* **bool CollideWith(t:Thing)**: Ütközik t-vel és visszatér igaz/hamis értékkel, ami azt jelenti, hogy sikerült-e ütközni.
* **void Move(d: Direction):** ‘d’ irányba mozgatja a Thing-et.
* **bool HitBy(t: Thing,d: Direction,o: Thing)**: közli egy másik Thinggel hogy nekiment egy másik Thing, és, hogy mozduljon el ha tud.
* **void OnEndTile(t: EndTile)**: Közli az adott Thinggel, hogy EndTile-on van.
* void **OnSwitch(s: Switch)**:Közli az adott Thinggel hogy Switch-en van.
* **Thing GetOwner()**: Visszaadja az owner attribútumot.
* **bool UpdateOwner(newvalue:Thing)**: Beállítja az owner attribútum értékét a newrvalue-ra.
* **Direction GetMoving()**: Visszaadja a moving attribútum értékét.
* **void SetMoving(newvaluevalue:Direction)**: Beállítja a moving attribútum értékét a newvalue-ra.
* **Tile GetTile()**: Visszaadja a Tile attribútum értékét.
* **void SetTile(newvaluevalue:Tile)**: Beállítja a Tile attribútum értékét a newvalua-ra.
* **void Destroy():** Leveszi magát a pályáról és meghívja a saját destruktorát.
  + 1. **Timer**
       - **Felelősség**

Idózítő osztály, singleton.

* + - * **Ősosztályok**

Nincs.

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**

**-steppables: Steppable [0..\*]:** Timer által léptetendő léptethető objektumok.

* + - * **Metódusok**
* **void Tick():** Egyet “kattan” az óra, léptet minden lépni képes objektumot.
* **void AddSteppable(s: Steppable)**: Hozzáad egy Steppable objektumot a léptetendő objektumok listájára.
* **void RemoveSteppable(s: Steppable)**: Eltávolítja ‘s’-t a léptetendő objektumok listájáról, ha a listában van.
  + 1. **Tile**
       - **Felelősség**

Az általános pályaelemet reprezentálja, amin lehet Thing.

* + - * **Ősosztályok**

Nincs.

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**
* **-nexttiles[Direction]:Tile**: A tileból induló tileok.
* **-thing:Thing**: A tileon lévő thing.
  + - * **Metódusok**
* **bool Accept(m: Thing)** Befogadja a bejövő Thing objektumot. Ha nem üres, akkor ütközteti a rajtalévővel.
* **void Remove(m: Thing)**: eltávolítja az éppen rajta lévő Thinget.
* **Tile GetNeighbor(d: Direction)**: d irányban lévő szomszédos Tile-lal tér vissza, vagy NULL-lal, ha nincs ebben az irányban tile.
* **void SetNeighbor(d: Direction, t: Tile)**: Beállítja t-re d irányban a szomszédját.
  + 1. **Wall**
       - **Felelősség**

A pálya szélén és a pályán lévő falakat reprezentáló osztály.

* + - * **Ősosztályok**

Tile

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**

Nincs.

* + - * **Metódusok**
* **bool Accept(t: Thing):** Mindig false-sal tér vissza, mivel falra menni nem lehet.
  + 1. **Worker**
       - **Felelősség**

A játékos által irányított munkást reprezentáló osztály.

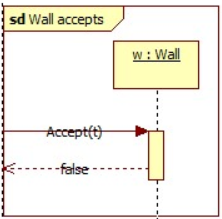
* + - * **Ősosztályok**

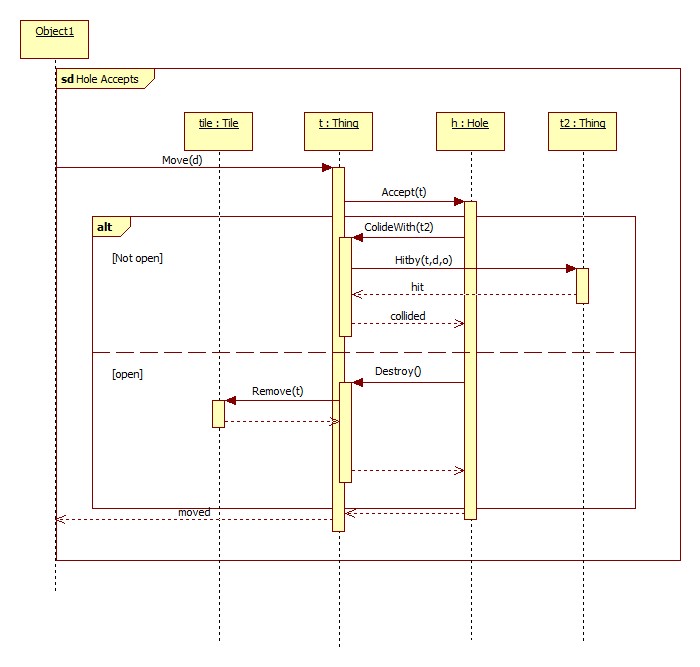
Thing

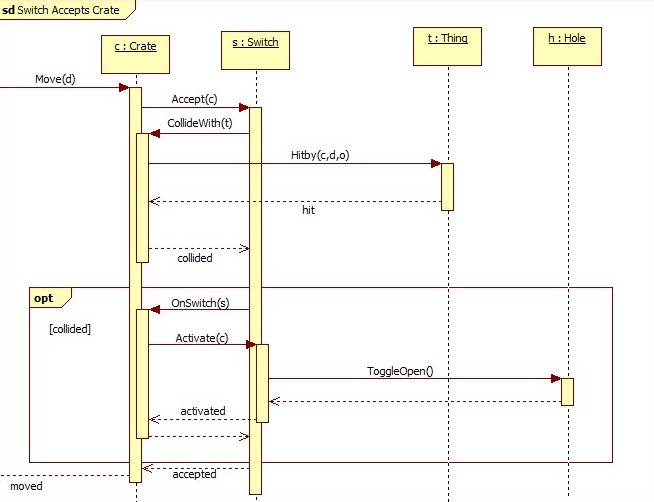
* + - * **Interfészek**

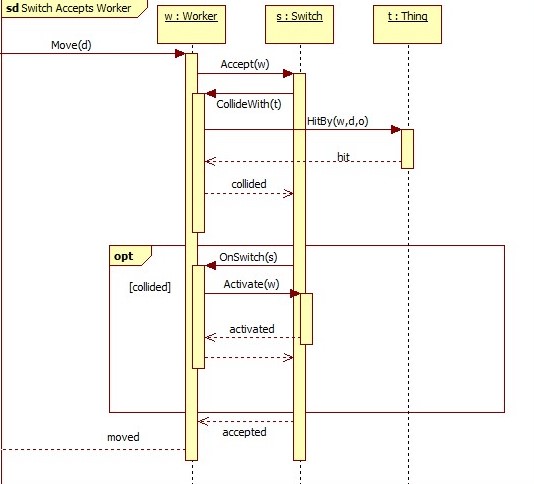
Nincs.

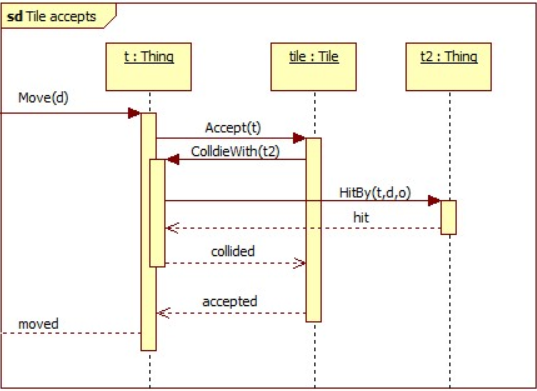
* + - * **Attribútumok**
* **-points: int:** A munkás pontjainak száma.
  + - * **Metódusok**
* **bool HitBy(t: Thing)**: metódus leírása
* **void OnEndTile(t: EndTile)**: Nem csinál semmit.
* **int GetPoints()**: Visszaadja a points attribútum értékét.
* **void SetPoints(newvalue:int)**: Beállítja a points attribútum értékét a newvalue-ra.
* **bool UpdateOwner(newvalue:Thing)**: Beállítja az owner attribútum értékét a newrvalue-ra.
* **void Destroy():** Leveszi magát a pályáról és meghívja a saját destruktorát.
  1. **Szekvencia diagramok**

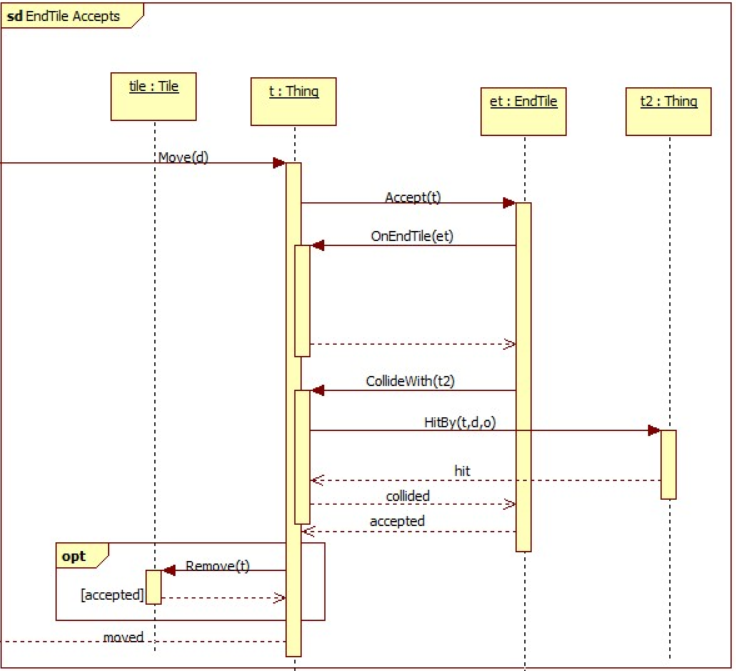
****

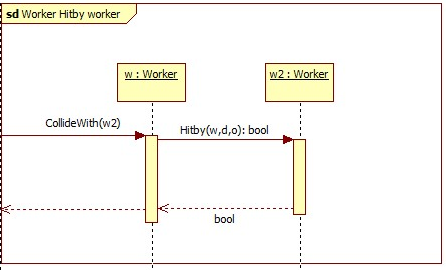
****

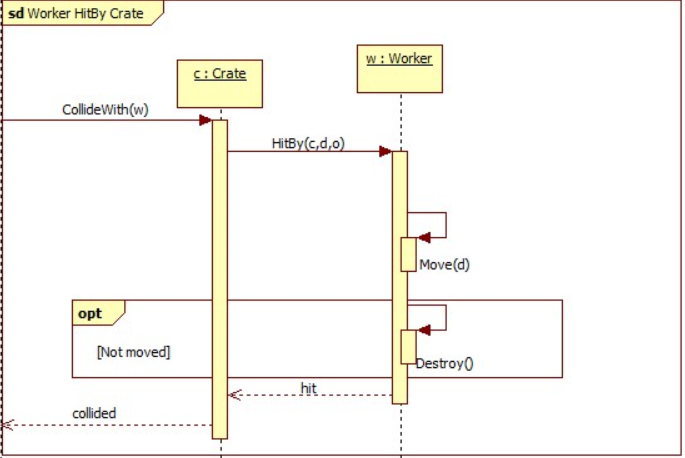
****

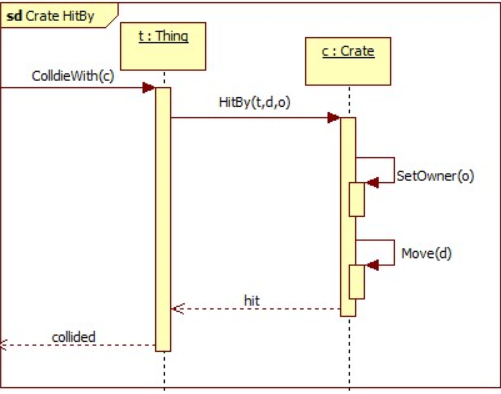
****

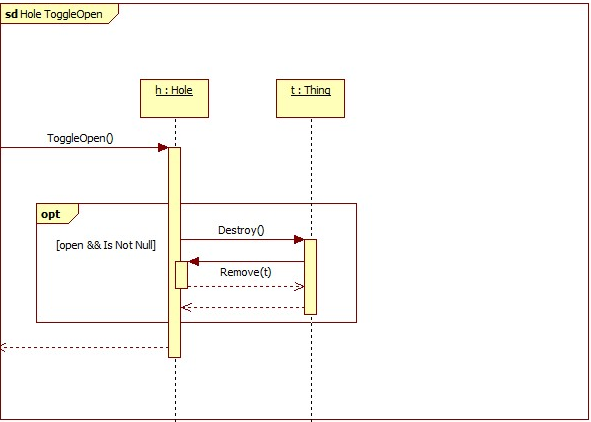
****

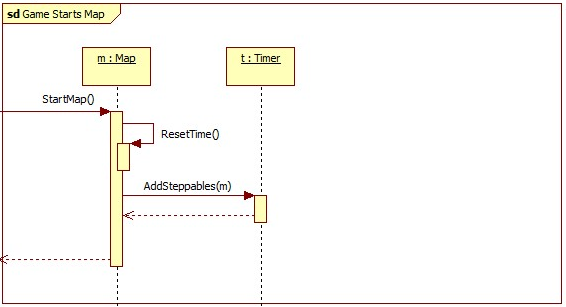
****

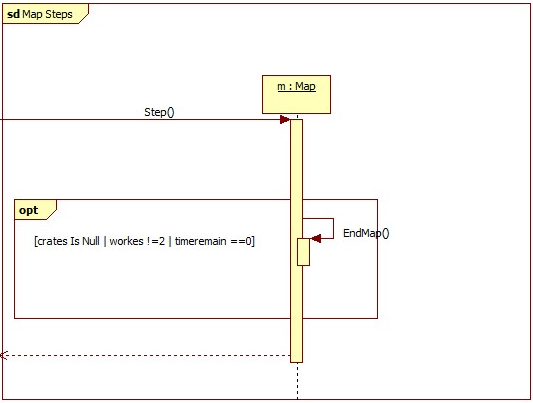
****

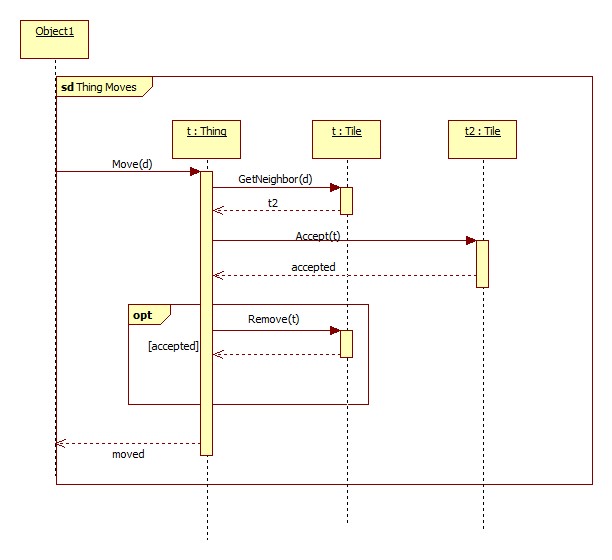
****

****

****

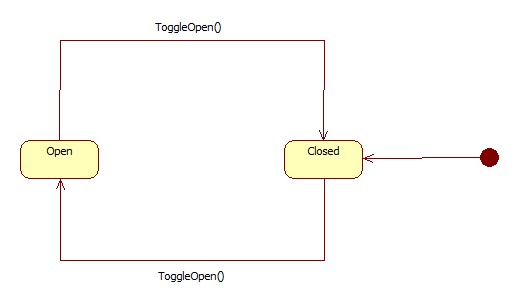
****

****

****

* 1. **State-chartok**

**Hole:**

****

**Napló**

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2017.03.01. 17:00 | 0,5 óra | Torma  Pünkösd  Tóth Vince  Tóth D  Veres | Gyűlés: Változások megbeszélése. Döntés: Szekvencia diagrammok esztétikai javítása, felmerült funkcionális javítások elkészítése. |
| 2017.03.4. 20:00 | 0.5 óra | Pünkösd | Tevékenység: Osztály diagram javítása. |
| 2017.03.2. 22:00 | 0.5 óra | Tóth Vince | Tevékenység: Szekvencia diagrammok logikai javítása. |
| 2017.03.22. 15:00 | 0.5 óra | Tóth D | Tevékenység: Szekvencia diagrammok esztétikai javítása. |
| 2017.03.23. 19:00 | 0.5 óra | Veres | Tevékenység: Dokumentum előkészítése, apró szekvencia diagram hibák javítása. |
| 2017.03.5. 00:00 | 0.5 óra | Torma | Tevékenység: Dokumentum véglegesítése, lektorálás, nyomtatás. |