**3. Analízis modell kidolgozása**

15 – jgoldfisch

Konzulens:

Bodó Zsófia

Csapattagok

|  |  |  |
| --- | --- | --- |
| Mátyás Gergely | IL21NI | matyasg97@gmail.com |
| Horváth Ákos | DKILK6 | horvath.akos1997@gmail.com |
| Gurubi Barnabás | DXEXVR | gurubibarni@gmail.com |
| Tolnai Márk | ID61MK | tolesz11@windowslive.com |
| Bertalan Bálint | HNN9GA | blintber@gmail.com |

2018. február 25.

# Analízis modell kidolgozása

## Objektum katalógus

[Minden, a feladatban szereplő objektum rövid, egy-két bekezdés hosszú ismertetése. Meg kell jelenjen minden objektumhoz, hogy mi a felelőssége. Informális leírás, ezért nem szabad még foglalkozni az örökléssel, az interfészekkel, az absztrakt osztályokkal, a segédosztályokkal.]

### SimpleField

A pálya egyes mezőit képviselik a játékban. Lekérdezhető tőle a megfelelő irányban lévő szomszédos **Field**. Egy cellán egyidőben állhat egy **Box,** vagy egy **Player** állhat, valamint tartalmazhat referenciát egy speciális elemre amely lehet **Hole** vagy **Switch**.

### Box

A box osztály minden példánya egy-egy különböző dobozt tárol. Amennyiben egy dobozt a célmezőre tolnak ellehetetleníti önmaga mozgatását.

### Game

A játék alapvető logikáját és mechanizmusait tárolja. Ezen felül referenciát tartalmaz a **Mapre** és tárolja a **Worker-öket**.

### Hole

Egy lyukat jelenít meg. Amennyiben egy **Box**, vagy **Worker** áll rajta, megsemmisíti a rajta álló objektumot.

### Map

A pályát képviseli, mely több **Fieldből** áll. A menüben a pálya kiválasztását követően az osztály képes betölteni azt egy külső fájlból.

### Switch

Egy a pályán található kapcsolót valósít meg. Minden kapcsolóhoz tartozik egy referencia egy **Hole** osztályra. Amennyiben egy **Box** áll a **Switchen**, akkor aktiválódik, más esetben nem.

### WallField

Egy oszlopot, vagy falszeletet ábrázol a pályán. Megakadályozza, hogy dobozt toljanak rá, vagy hogy rálépjen egy játékos.

### Worker

Egy játékost képvisel, aki pálya területén belül szabadon mozoghat. Segítségével lehetséges a ládák tologatása és tárolja, hogy hány ládát tolt már a helyére.

[02. 22 – 19:00 -  
Balu – 1,5 óra, Márk – 0,5 óra]

[Az objektumkatalógus alapján kiindulva kell megalkotni az objektumorientált analízis modellt. A 3.2, 3.3, 3.4, 3.5 alfejezetek ugyanannak a modellnek a különböző nézetei, ezért egyidőben, egymással összefüggésben készülnek. Megtörténik az objektumkatalógusban tárgyalt objektumok felelősségének formalizálása osztályokká, attribútumokká, metódusokká . Csak publikus metódusok szerepelhetnek. Megjelennek az interfészek, az öröklés, az absztrakt osztályok. Segédosztályokra még nincs szükség.]

## Statikus struktúra diagramok

[Az objektum katalógusban szereplő objektumokat megvalósító osztályok kapcsolatait és publikus metódusait bemutató osztálydiagram(ok). Tipikus hibalehetőségek: csillag-topológia, szigetek.]

## Osztályok leírása

[A diagramon szereplő osztályok bemutatása. Az osztályok ABC sorrendben kövessék egymást. Interfészek esetén az Interfészek, Attribútumok pontok kimaradnak.]

### Box

#### Felelősség

Ennek az osztálynak az egyes példányai fogják reprezentálni a dobozokat a játékban. Őket kell majd a célmezőre eljuttatni.

#### Ősosztályok

Movable

#### Attribútumok

* **bool movable:** ezzel jelezzük, hogy mozgatható-e a láda.
* **Field field:** az a mező, ahol éppen tartózkodik a láda.

#### Metódusok

* **void pushBack(Direction d):** Ennek a függvénynek a segítségével a tolás irányába visszaléptethetjük a dobozt. (Erre szükség van pl. amikor a falba bele akarnánk tolni a dobozt.)
* **void pushBy(Box b, Direction d):** Függvény arra az esetre, amennyiben a ládánkat (Box) egy másik doboz tolja meg, intézi, hogy a láda a megfelelő helyre kerüljön.
* **void pushBy(Worker w, Direction d):** A láda (Box) egy játékos által való megtolásákor hívjuk meg, intézi hogy a láda a megfelelő helyre kerüljön.
* **void scorePoint(Direction d):** Amennyiben a célmezőre tolódott a láda, ennek a függvénynek a segítségével végezhetjük el a megfelelő játékos pontjainak növelését.

### Field

#### Felelősség

Absztrakt ősosztály, amiből a mező objektumok származnak.

#### Attribútumok

* **Movable movable:** referencia egy Movable leszármazottra.
* **Field neighbor[Direction]:** Tárolja a szomszédos mezőket a lehetséges irányokban.

#### Metódusok

* **Field getNeighbor(Direction d):** Visszaadja az átadott irányban lévő szomszédos mezőt.
* **Movable getMovable():** Visszaadja a mezőn lévő Movable leszármazott referenciáját.
* **Movable enter(Worker w, Direction d):** Absztrakt függvény, ami a leszármazottakban kerül implementálásra.
* **Movable enter(Box b, Direction d):** Absztrakt függvény, ami a leszármazottakban kerül implementálásra.
* **void remove():** A mezőhöz tartozó (tárolt) Movable referenciát eltávolítja.

### Game

#### Felelősség

[Mi az osztály felelőssége. Kb 1 bekezdés.]

#### Ősosztályok

[Mely osztályokból származik (öröklési hierarchia)]

Legősebb osztály → Ősosztály2 → Ősosztály3...

#### Interfészek

[Mely interfészeket valósítja meg.]

#### Attribútumok

[Milyen attribútumai vannak]

* **attribútum1**
* **attribútum2**

#### Metódusok

[Milyen publikus metódusokkal rendelkezik. Metódusonként egy-három mondat arról, hogy a metódus mit csinál. Hiba, ha egy osztálynak nincs egyetlen metódusa sem.]

* **int foo(Osztály3 o1, Osztály4 o2)**: metódus leírása
* **int bar(Osztály5 o1)**: metódus leírása

### Hole

#### Felelősség

Megsemmisíti a rajta álló **Movable** osztályt.

#### Ősosztályok

Special

#### Attribútumok

* **SimpleField field:** Referencia a mezőre, melyen lyuk áll
* **Switch switch:** Referencia a hozzá tartozó kapcsolóhoz

#### Metódusok

* **void interact(Movable m):** Egy a mezőn lévő lyukkal való interakció (rálépés) esetén hívjuk meg. Ellenőrzi, hogy tartozik-e hozzá kapcsoló, továbbá meghívja a reagáláshoz tartozó további metódusokat.
* **void execute(Movable m):** Függvény arra, ha meg akarjuk semmisíteni a mezőn álló osztályt. Meghívja a kellő metódusokat a objektum megsemmisítéséhez.
* **void changeToActive():** A kapcsoló megváltoztatása esetén hívjuk meg, hogyha áll valami abban a pillanatban a lyukon, akkor azt semmisítse meg.

### Map

#### Felelősség

[Mi az osztály felelőssége. Kb 1 bekezdés.]

#### Ősosztályok

[Mely osztályokból származik (öröklési hierarchia)]

Legősebb osztály → Ősosztály2 → Ősosztály3...

#### Interfészek

[Mely interfészeket valósítja meg.]

#### Attribútumok

[Milyen attribútumai vannak]

* **attribútum1**
* **attribútum2**

#### Metódusok

[Milyen publikus metódusokkal rendelkezik. Metódusonként egy-három mondat arról, hogy a metódus mit csinál. Hiba, ha egy osztálynak nincs egyetlen metódusa sem.]

* **int foo(Osztály3 o1, Osztály4 o2)**: metódus leírása
* **int bar(Osztály5 o1)**: metódus leírása

### Movable

#### Felelősség

Egy absztrakt ősosztály, amiből a mozgatható objektumok származnak.

#### Attribútumok

* **Field field:** a mező, amin tartózkodik

#### Metódusok

* **void die():** Ez a függvény fog felelni a leszármazottak eltávolításáért.
* **void finalizeStep():** Ez az a függvény, ami az egyes lépések véglegesítésért felel. Meghívja annak a „SimpleField”-nek az „arrive(m: Movable)” függvényét, ahová a Movable leszármazott majd megérkezne, ami pedig, ha tárolja a „Special” osztály egy példányát (jellemzően kapcsolót (Switch), vagy lyukat (Hole)), akkor meghívja annak az „interact(Movable m)” függvényét, ami pedig megteszi a kellő intézkedést (bekapcsolja a kapcsolót vagy megsemmisíti a Movable leszármazottat).
* **void place(Field f):** Ez a függvény átállítja a Movable mező referenciáját a kapott mezőre.
* **void pushBack(Direction d):** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* **void pushBy(Box b, Direction d):** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* **void pushBy(Worker w, Direction d):** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* **void scorePoint(Direction d):** Absztrakt függvény, a leszármazottakban kerül megvalósításra.

### SimpleField

#### Felelősség

[Mi az osztály felelőssége. Kb 1 bekezdés.]

#### Ősosztályok

[Mely osztályokból származik (öröklési hierarchia)]

Legősebb osztály → Ősosztály2 → Ősosztály3...

#### Interfészek

[Mely interfészeket valósítja meg.]

#### Attribútumok

[Milyen attribútumai vannak]

* **attribútum1**
* **attribútum2**

#### Metódusok

[Milyen publikus metódusokkal rendelkezik. Metódusonként egy-három mondat arról, hogy a metódus mit csinál. Hiba, ha egy osztálynak nincs egyetlen metódusa sem.]

* **int foo(Osztály3 o1, Osztály4 o2)**: metódus leírása
* **int bar(Osztály5 o1)**: metódus leírása

### Special

#### Felelősség

Absztrakt osztály. Egy speciális mezőt ábrázol.

#### Attribútumok

* **SimpleField field:** referencia a mezőre, amin a speciális mező áll

#### Metódusok

* **void interact(Movable m):** Absztrakt függvény, leszármazottakban kerül megvalósításra.

### Switch

#### Felelősség

Egy kapcsolót és hozzá tartozó állapotot tárolja.

#### Ősosztályok

Special

#### Attribútumok

* **SimpleField field:** referencia a mezőre, melyen a kapcsoló található
* **Hole hole:** referencia a kapcsolóhoz tartozó lyukhoz
* **bool active:** változó, hogy a kapcsoló jelen pillanatban aktiválva van-e

#### Metódusok

* **void activate():** Aktiválja a kapcsolót
* **void deactivate():** Deaktiválja a kapcsolót
* **void holeInteracted(Movable m):** Akkor hívódik meg, ha a kapcsolóhoz tartozó mezőre valaki rálép. Eldönti, hogy a kapcsoló éppen aktív-e én ennek függvényében meghívja a szükséges metódusokat.
* **void interact(Box b):** Lekezeli, ha doboz érkezik a mezőre. Aktiválja a kapcsolót és szól a lyuknak, hogy megváltozott az állapota.

### WallField

#### Felelősség

Olyan mező objektum, mely egy falat reprezentál.

#### Attribútumok

* **Movable movable:** referencia egy Movable leszármazottra.
* **Field neighbor[Direction]:** Tárolja a szomszédos mezőket a lehetséges irányokban.

#### Metódusok

* **Field getNeighbor(Direction d):** Visszaadja az átadott irányban lévő szomszédos mezőt.
* **Movable getMovable():** Visszaadja a mezőn lévő Movable leszármazott referenciáját.
* **Movable enter(Worker w, Direction d):** Megakadályozza, hogy egy **Worker** rálépjen a mezőre.
* **Movable enter(Box b, Direction d):** Megakadályozza, hogy egy **Box** rálépjen a mezőre.
* **void remove():** A mezőhöz tartozó (tárolt) Movable referenciát eltávolítja.

### Worker

#### Felelősség

A játékosok által irányított munkást reprezentáló osztály, ő tudja a ládákat (Box) eltolni, illetve a pontokat is ő kapja majd.

#### Ősosztályok

Movable

#### Attribútumok

* **Field field:** referencia a mezőre, amin a munkás épp tartózkodik.
* **int points:** a játékos pontjai, amiket az egyes ládák célmezőre juttatásáért kaphat.

#### Metódusok

* **void control(Direction d):** A munkás irányításáért felelős függvény. A paraméterben kapott irányban fogja megpróbálni elmozdítani a munkást a szomszédos mezők figyelembe vételével (Mindez önakaratból történik, tehát a felhasználó beavatkozására).
* **void pushBy(Box b, Direction d):** Ez a függvény arra az esetre szolgál, hogyha a munkást egy dobozzal (Box) próbálnának meg eltolni.
* **void pushBy(Worker w, Direction d):** Ez a függvény arra az esetre szolgál, ha a munkást egy másik munkás próbálná meg eltolni.
* **void goBack(Direction d):** Szintén a munkás -> munkás ütközés / eltolás megfelelő lekezelésében van szerepe, mégpedig elindít egy visszafele való lökést, hiszen nem szabad hagyni, hogy az egyik munkás a másikat közvetlenül eltolja, tehát úgymond visszarendezi a dolgokat.
* **void pushBack(Direction d):** Ennek a függvénynek ott van szerepe, amikor egy munkás a dobozt a falba bele akarná tolni. Ekkor ezt nem engedi majd, hanem az egész sor (pl.: munkás->doboz->fal) egyet visszafele fog lépni   
  (munkás->doboz->üres mező->fal).  
  Ennek a megvalósításáért részben felelős ez a metódus is.
* **void scorePoint(Direction d):** A pontok növeléséért felelős függvény.

## Szekvencia diagramok

[Inicializálásra, use-case-ekre, belső működésre. Konzisztens kell legyen az előző alfejezettel. Minden metódus, ami ott szerepel, fel kell tűnjön valamelyik szekvenciában. Minden metódusnak, ami szekvenciában szerepel, szereplnie kell a valamelyik osztálydiagramon. Fontos, hogy az aktor use-case-eitől bármelyik diagramhoz el lehessen jutni a metódushívások követésével.]

## State-chartok

[Csak azokhoz az osztályokhoz, ahol van értelme. Egyetlen állapotból álló state-chartok ne szerepeljenek. A játék működését bemutató state-chart-ot készíteni tilos.]

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2010.03.21. 18:00 | 2,5 óra | Horváth  Németh  Tóth  Oláh | Értekezlet.  Döntés: Horváth elkészíti az osztálydiagramot, Oláh a use-case leírásokat. |
| 2010.03.23. 23:00 | 5 óra | Németh | Tevékenység: Németh implementálja a tesztelő programokat. |
| … | … | … | … |