**8. Részletes tervek**

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. április 9.

# Részletes tervek

A programban nem szerepelnek bonyolult algoritmusok, hiszen a program egy egyszerű játék, így sokszor a legegyszerűbb megoldások is kielégítőek (hiszen a hatékonyság nem kulcskérdés, sokkal inkább a hibamentesség).

A programot alapvetően a **CommandHandler** osztály vezérli, hiszen ez kezeli a bemeneti nyelvet, amivel a felhasználó irányítja a programot.

## Osztályok és metódusok tervei.

### Box

#### Felelősség

Ennek az osztálynak az egyes példányai fogják reprezentálni a dobozokat a játékban. Őket kell majd a célmezőre eljuttatni. Csak bizonyos számút tud eltolni a munkás belőlük az ereje és a módosítók alapján, továbbá lyukba esve megsemmisülnek.

#### Ősosztályok

Movable

#### Attribútumok

* **-Field field:** az a mező, ahol éppen tartózkodik a láda.
* **-String name:** azonosításhoz/megjelenítéshez használt név

#### Metódusok

* **+ void die():** Ez a függvény felel az entitás elpusztításáért, illetve csökkenti a szabad dobozok számát.
* **+ 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 pushByBox(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 pushByWorker(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, így a hívást mindig továbbadja a megfelelő irányban álló **Movable**-nek (meghívja rajta önmagát).
* + **void finalizeStep():** Ez az a függvény, ami az egyes lépések véglegesítésért felel, a léptetés érvényessége vizsgálatát követően.
* **+ void setString(String s):** név beállítása
* **+ String toString():** név lekérdezése

### BoxRecorder

#### Felelősség

Nyilvántartás vezetése a mozdíthatatlan mezőkről, amiken beragadt doboz van. Segítségével lehetséges a beragadt (pl. sarokban lévő) dobozok detektálása. (Mivel csak egy példány lesz belőle ezért Singleton) Kezdetben csak a falmezők szerepelnek benne, továbbá azok a mezők melyeken doboz van és nem mozdítható. Ha minden doboz beragad, akkor véget ér a játék.

#### Attribútumok

* **- ArrayList<Field> fixFields:** referenciák azokra a mezőkre, amikre nem lehet rálépni.

#### Metódusok

* **+ List<Field> getFixFields():** Referencia a nem szabad mezőkre.
* **+ void setFixFields(List<Field> fixFields):** setter függvény a nem szabad mezők beállítására.
* **+ void checkRecordWith(Field f):** Ezzel a függvénnyel lehetséges annak az ellenőrzése, hogy a paraméterben kapott mező benne-e van már a nyilvántartásban. Ha nincs akkor csökkenti a szabad dobozok számát a **Game**-ben.
* **+ void endFieldOccupied(Field f):** Ennek a függvénynek a segítségével jelezhető, ha egy célmezőre (**EndField**) beérkezett egy láda (**Box**). A paraméter átadásával jelezzük, hogy melyik mezőről van szó, majd ez a mező bekerül a nyilvántartásba, hiszen onnan a doboz nem mozdítható el.
* **+ void update(Field f):** Ezt a függvényt mindig egy doboz (**Box**) mozgatása után hívjuk meg, miután elhelyeződött a mezőn (*Field….arrived()*). A paraméterben átadott mező az a mező, ahová a doboz újonnan került. A nyilvántartás alapján pedig el tudja dönteni, hogy a dobozunk beszorult-e (a szomszédai alapján). Amennyiben igen, akkor azt is felveszi a nyilvántartásba.

Egy doboz akkor van beszorulva, ha két nem átellenes oldalszomszédja benne vana nyilvántartásban, vagy legalább 3 szomszédja benne van.

* **+ void init(ArrayList<Field> walls):** A **Map**-től megkapja a falakat, erre azért van szükség mert kezdetben ezeket tárolja el a fixFields-be.

### CommandHandler

#### Felelősség

A bemeneti nyelvet kezelő osztály, a kiadott parancsokat értelmezi és dolgozza fel. Meghívja a parancs végrehajtásához szükséges egyéb metódusokat.

#### Attribútumok

-

#### Metódusok

* **+ void readInput():** A standard bemenetről beolvas egy adott parancsot, majd szétbontja argumentumokra, majd ezt átadja az ezeket kezelő függvénynek.
* **- void process(String[] args)**: Beolvassa az adott parancsot és meghívja az azt végrehajtó függvényt.
* **- void load\_map(String fileName):** A *load-map* parancsot végrehajtó függvény, létrehoz egy pályát, majd meghívja a pálya betöltéséért felelős függvényt. A kész pályát odaadja a *Game*-nek.
* **-** **void control(int workerID, Direction d):** A *control* parancsot végrehajtó függvény, az adott azonosítójú munkásnak meghívja a *control* függvényét az adott irányba. Ezzel egy teljes függvénylefutási sort elindítva (ezt ki is írja az adott naplókba). Az adott munkást a *Game*-től kapja.
* - **void drop\_special(int workerID, String type):** A *drop-special* parancs végrehajtása, az adott azonosítójú munkásnak meghívja a megfelelő (*type*-tól függő) függvényét *place…().* Az adott munkást a *Game*-től kapja.
* **- void ls\_workers():** Az *ls-workers* parancs végrehajtása, kilistázza az összes munkást a pályán a kimeneti nyelvnek megfelelően a beállított naplóba. A munkásokat a *Game*-től éri el.
* - **void ls\_boxes():**  Az *ls-boxes* parancs végrehajtása, kilistázza az összes dobozt a pályán a kimeneti nyelvnek megfelelően a beállított naplóba. A dobozokat a pályától éri el, amit a *Game*-től kap.
* - **void** **ls\_fields():** Az *ls-fields* parancs végrehajtása, kilistázza az összes mezőt a pályán a kimeneti nyelvnek megfelelően a beállított naplóba. A mezőket a pályától éri el, amit a *Game*-től kap.
* - **void log(String fileName):** A *log* parancs végrehajtása, beállítja a naplózás helyét az adott fájlba, illetve ha nincs fájlnév, akkor a standard kimenetre.
* - **void show\_log():** A *show-log* parancs végrehajtása, kilistázza a naplózások helyét, a kimeneti nyelvnek megfelelően.
* - **void** **log\_off(String arg):** A *log-off* parancs végrehajtása, leállítja az argumentumnak megfelelő naplózást, akár az összeset egyszerre.
* -  **void drop\_map():** A *drop-map* parancs végrehajtása, az aktuális pályát eldobja, vagyis törli.
* - **void show\_map():** A *show-map* parancs végrehajtása, kiírja az aktuális pályafájl nevét.
* - **void run\_test(String fileName):** A *run-test* parancs végrehajtása, az adott fájlból beolvassa az előre megírt parancsokat az adott fájlból, ezután végrehajtja őket sorban, a *process* függvény segítségével.
* - **void exit():** Az *exit* parancs végrehajtása, kilép a programból. Fontos, hogy a *run-test* parancson belül az *exit* parancs csak a teszteset futtatását zárja le, nem lép ki a programból.

### EndField

#### Felelősség

Célmező, amire a dobozokat (**Box**)juttatni kell. Amennyiben rálép egy doboz, akkor az mozdíthatatlanná válik ott, ilyenkor a pontozási folyamat is elindul. Mivel mező így a **Field**-ből származik.

#### Ősosztályok

Field

#### Attribútumok

-

#### Metódusok

* **+ Movable boxEnters(Box b, Direction d):** Fogadja a mezőre érkező **Box**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött, null-ra állítja). Végezetül pedig mozdíthatatlanná teszi a mezőre érkezett **Box-**t. Továbbá elindítja a pontszerzési folyamatot a megfelelő munkás számára.
* **+ Movable workerEnters(Worker w, Direction d):** Fogadja a mezőre érkező **Worker**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött, null-ra állítja).
* **+ void boxArrived(Box x):** Elvégzi a kapott objektum mezőre való elhelyezését. És elindítja a pontosztást.

### Field

#### Felelősség

Absztrakt ősosztály, amiből a mező objektumok származnak. Tárolja a szomszédait, az irányokkal együtt. Továbbá tárolja a rajta lévő **Movable** objektumot. Egyszerre egy doboz vagy munkás fér el rajta.

#### Attribútumok

* **- Field neighbor[Direction]:** tárolja a szomszédos mezőket a lehetséges irányokban.
* **- Movable movable:** referencia egy **Movable** leszármazottra.
* **- Slime slime:** referencia a hozzá tartozó **Slime-**hoz.

#### Metódusok

* **+ Field getNeighbor(Direction d):** Visszaadja az átadott irányban lévő szomszédos mezőt.
* **+ void setNeighbor(Direction d, Field f):** A mező szomszédjának beállítása.
* **+ Movable boxEnters(Box b, Direction d):** Absztrakt függvény, ami a leszármazottakban kerül implementálásra.
* **+ Movable workerEnters(Worker w, Direction d):** Absztrakt függvény, ami a leszármazottakban kerül implementálásra.
* **+ Movable getMovable():** Visszaadja a mezőn lévő **Movable** leszármazott referenciáját.
* **+ void setMovable(Movable m**): A mezőn lévő Movable referencia beállítása.
* **+ void remove(Movable m):** A mezőhöz tartozó (tárolt) **Movable** referenciát eltávolítja, abban az esetben, ha önmaga referenciáját találja.
* **+ void workerArrived(Worker w):** Elvégzi a kapott objektum mezőre való elhelyezését.
* **+ void boxArrived(Box x):** Elvégzi a kapott objektum mezőre való elhelyezését.
* **+ void updateRecorder():** A **BoxRecorder** frissítéséért felelős függvény.
* **+ void setSlime(Slime s):** Beállítja a mezőn lévő **Slime** referenciáját a kapottra.
* **+ int interact(int f):** A kapott erőt a mezőn lévő anyag függvényében megváltoztatja.

### Game

#### Felelősség

A játék alapvető mechanizmusaiért felel és tárolja a nyeréshez, vagy vesztéshez szükséges adatokat. Továbbá tárolja a még életben lévő **Worker**-öket, és a **Map**-et. (Mivel csak egy példány lesz belőle ezért Singleton)

#### Attribútumok

* **- final Game instance:** A Singleton tervezési minta megvalósítását támogató változó.
* **- int freeBoxCounter:** a még mozgatható dobozok számát tárolja.
* **- int workerCounter:** a még játékban lévő munkások számát tárolja.
* **- Map map:** referencia a pályát tartalmazó osztályra.
* **- ArrayList<Worker> workers:** a pályán lévő, még élő munkások.
* **- BoxRecorder boxRecorder:** Referencia a szabad dobozokat vizsgáló objektumra.

#### Metódusok

* **+ void chooseMap(String s):** Betölti a paraméterként kapott pályát.
* **+ void decreaseBoxes(Box b):** Abban az esetben hívódik meg, amennyiben egy doboz megsemmisül. Ha ez a doboz nem volt beragadva, akkor csökkenti a még mozgatható dobozok számát egyel.
* **+ void decreaseFreeBoxes():** Csökkenti a még mozgatható dobozok számát 1-el, amennyiben egy doboz beragad.
* **+ void endGame():** Akkor hívódik meg, ha a mozgatható dobozok száma nullára csökken. Ekkor a legtöbb pontot elért játékos nyer (maximumkeresés a még élő játékosok pont attribútumán).
* **+ void init(ArrayList<Worker> workers):** Eltárolja a munkásokat, amelyet a **Map**-től kap.
* **+ void gameOver():** Akkor hívódik meg, ha minden játékos meghal. Ebben az esetben senki sem kerül ki nyertesként a játékból.
* **+ void workerDied(Worker w):** Amennyiben meghal a **Worker,** csökkenti a játékban lévő munkások számát, és kiveszi az adott munkás referenciáját.
* **+ List<Worker> getWorkers():** getter függvény a munkások listájára
* **+ void setWorkers(List<Worker> workers):** setter függvény a munkások listájára
* **+ BoxRecorder getBoxRecorder():** getter függvény a boxRecorderre
* **+ void setBoxRecorder(BoxRecorder boxRecorder):** setter függvény a boxRecorderre
* **+ Map getMap():** getter függvény a tárolt mapra
* **+ void setMap(Map map):** setter függvény a tárolt mapra

### HoleField

#### Felelősség

Megsemmisíti a rajta álló **Movable** osztályt. Tartozhat hozzá kapcsoló, ennek állapotát figyelembe veszi működése közben. Mivel mezőként viselkedik, így a **Field**-ből származik.

#### Ősosztályok

Field

#### Attribútumok

* **- SwitchField switch:** referencia a hozzá tartozó kapcsolóhoz.

#### Metódusok

* **+ void setSwitchField(SwitchField switchField):** setter a hozzá tartozó kapcsolóhoz
* **+ Movable boxEnters(Box b, Direction d):** Fogadja a mezőre érkező **Box**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy van e rajta **Movable**,és visszaadja annak a referenciáját (ha nincs akkor null).
* **+ Movable workerEnters(Worker w, Direction d):** Fogadja a mezőre érkező **Worker**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy van e rajta **Movable**,és visszaadja annak a referenciáját (ha nincs akkor null).
* **+ void changeToActive():** A kapcsoló megváltoztatása esetén hívjuk meg, hogy ha áll valami abban a pillanatban a lyukon, akkor az megsemmisül. Ezen felül megszünteti a lyukon lévő **Slime** referenciáját.
* **+ void execute(Movable m):** Függvény arra, ha meg akarjuk semmisíteni a mezőn álló **Movable** objektumot. Meghívja a kellő metódusokat az objektum megsemmisítéséhez.
* **+ void workerArrived(Worker w):** Munkás lyukra való érkezését kezeli. Ellenőrzi, hogy tartozik-e hozzá kapcsoló, továbbá meghívja a reagáláshoz tartozó további metódusokat.

### + void boxArrived(Box b): A működése megegyezik az előző függvényével.

### Honey

#### Felelősség

A mezőn elhelyezhető módosító, melynek a célja, hogy a surlódást növelje, emiatt egy munkás kevesebb dobozt tud eltolni. Munkás helyezheti el adott mezőre. Ha már található a mezőn **Slime**, akkor felülírja azt.

#### Interfészek

Slime

#### Attribútumok

-

#### Metódusok

* **+ int interact(int f)**: A kapott erőt módosítja, a viselkedésének megfelelően, mivel nehezebbé teszi az eltolást így többel csökkenti az erőt mint alapesetben csökkenne.

### Map

###### Felelősség

Szerepe a pálya betöltése. Ismeri az összes mezőt. Egy példány létezik belőle, singleton osztály.

###### Attribútumok

* **-ArrayList<Field>fields:** referenciák az egyes mezőkre.

###### Metódusok

* - **void setNeighbours():** Beállítja a loadMap által betöltött Field-ek szomszédait.
* +**ArrayList<Field> getFields():** Visszaadja a fields listát.
* +**void setFields(ArrayList<Field> f):** Beállítja a fields listát.
* +**void loadMap(int n):** A pálya betöltését intézi egy külső fájl segítségével, mely tartalmazza a pálya szerkezetét.

### MethodWriter

#### Felelősség

A programban lefutó játéklogikát megvalósító függvények megfelelő megjelenítéséért felelős osztály, és az ehhez tartozó feladatokat irányítja, valamint az adatokat tárolja. A feladata általános így minden metódusa és változója statikus.

#### Attribútumok

* - **int metodDepth**: Az adott függvényhívás mélysége, erre szükség van a megfelelő kiírtáskor a tabuláláshoz.
* - **int boxNum**: Az aktuális esetben szereplő dobozok száma, fontos a megfelelő nevezéktanhoz.

#### Metódusok

* - **String boxNamePostFix()**: A dobozok számozását intéző függvény, ez szükséges a megfelelő nevezéktanhoz.
* **+ String nameGenerator(String n)**: A nevezéktannak megfelelő nevet állít elő felhasználva a kapott előzetes, félkész nevet (paraméter). Ez doboz esetén b, munkás esetén pedig az adott munkás száma.
* **+ void printOutMethod(String methodName, Stirng params):** Egy metódust ír ki, az előre meghatározott (lsd. szkeleton terv) formában. Felhasználja az eltárolt adatokat a kiíráshoz (pl. függvényhívás mélysége), valamint a paraméterben kapott adatokat.
* **+ void printOutRet(String retVal):** Egy visszatérést ír ki, az előre meghatározott (lsd. szkeleton terv) formában. Felhasználja az eltárolt adatokat a kiíráshoz (pl. függvényhívás mélysége), továbbá a paraméterben megkapott adatot.
* **+ void printOutQuestion(String question):** Kérdést ír ki, a megfelelő előírt formában. (lsd. szkeleton terv). Felhasználja az eltárolt adatokat a kiíráshoz (pl. függvényhívás mélysége), valamint a kapott adatot.
* **+ void resetCounters():** A kiíráshoz használt számlálók alaphelyzetbe való állítása.

### Movable

###### Felelősség

Egy absztrakt ősosztály, amiből a mozgatható objektumok származnak. Tárolja azt, hogy melyik mezőn helyezkedik el.

###### Attribútumok

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

###### Metódusok

* **Field getField():** Visszaadja a field attribútum értékét.
* **void setField(Field f):** Beállítja a Movable objektum field attribútumának értékét.
* **+void die():** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* +**void finalizeStep():** Ez az a függvény, ami az egyes lépések véglegesítésért felel.
* **+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 pushByBox(Box b, Direction d, int f):** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* +**void pushByWorker(Worker w, Direction d, int f):** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* +**void scorePoint(Direction d):** Absztrakt függvény, leszármazottakban kerül megvalósításra.

### Oil

#### Felelősség

A mezőn elhelyzehető módosító, melynek a célja, hogy a surlódást csökkentse, emiatt egy munkás több dobozt tud eltolni. Munkás helyezheti el adott mezőre. Ha már található a mezőn **Slime**, akkor felülírja azt.

#### Interfészek

Slime

#### Attribútumok

-

#### Metódusok

* **+ int interact(int f)**: A kapott erőt módosítja, a viselkedésének megfelelően, mivel könnyebbé teszi az eltolást így kevesebbel csökkenti az erőt mint alapesetben csökkenne.

### SimpleField

###### Felelősség

Közönséges mező, amire szabadon rá lehet lépni. Ezeken közlekednek a munkások és a ládák.

###### Ősosztályok

Field

###### Metódusok

* + **String** **toString():** Kiírja a Field típúsát (“simple”).
* + **Movable boxEnters(Box b, Direction d):** Fogadja a mezőre érkező **Box**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy van e rajta **Movable**,és visszaadja annak a referenciáját (ha nincs akkor null).
* + **Movable workerEnters(Worker w, Direction d):** Fogadja a mezőre érkező **Worker**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy van e rajta **Movable**,és visszaadja annak a referenciáját (ha nincs akkor null).
* + **void boxArrived(Box b):** A *finalizeStep(),* azaz a lépés véglegesítésnél van szerepe ennek a függvénynek. Átadja az adott mezőre lépő **Box** referenciát, és azt elmenti a mező. Továbbá frissíti a **BoxRecorder**-t.
* + **void workerArrived(Worker w)**: Működése az előző függvénnyel megegyező, azonban nem frissíti a **BoxRecorder**-t.

### Slime

#### Felelősség

A mezőn elhelyzehető módosítók interfésze, melynek a célja, hogy a surlódást megváltoztassa. Munkás helyezheti el adott mezőre. Ha már található a mezőn **Slime**, akkor felülírja azt.

#### Interfészek

-

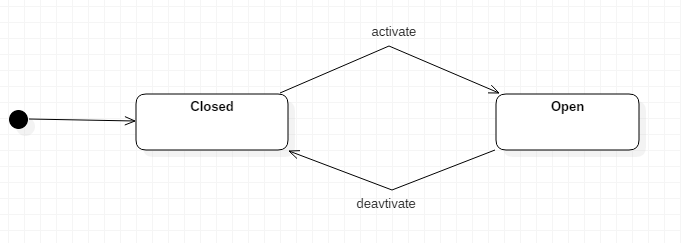
#### Metódusok

* **+ int interact(int f)**: A kapott erőt módosítja, a viselkedés az implementáló osztályokban kerül megfelelő megvalósításra.

### SwitchField

###### Felelősség

Egy kapcsolót és hozzá tartozó állapotot reprezentáló mező objektum. Ismeri a hozzá tartozó **HoleField**-t. Doboz rálépésekor kapcsol, ezt a következő állapotdiagramm írja le.



###### Ősosztályok

Field

###### Attribútumok

* - **HoleField hole:** referencia a kapcsolóhoz tartozó lyukhoz.
* - **boolean active:** változó, hogy a kapcsoló jelen pillanatban aktiválva van-e.

###### Metódusok

* + **HoleField** **getHole():** Visszaadja a hole attribútum értékét.
* + **boolean** **getActive():** Visszaadja a kapcsoló állapotát.
* + **void** **setHole(HoleField h):** Beállítja a hole értékét.
* + **String** **toString():** Kiírja a Field típúsát (“switch”).
* + **Movable boxEnters(Box b, Direction d):** Fogadja a mezőre érkező **Box**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy van e rajta **Movable**,és visszaadja annak a referenciáját (ha nincs akkor null).
* + **Movable workerEnters(Worker w, Direction d):** Fogadja a mezőre érkező **Worker**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy van e rajta **Movable**,és visszaadja annak a referenciáját (ha nincs akkor null).
* + **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 és ennek függvényében meghívja a szükséges metódusokat.
* + **void boxArrived(Box b):** Doboz kapcsolóra való érkezését kezeli. Szól a hozzá tartozó lyuknak, továbbá meghívja a reagáláshoz tartozó további metódusokat.
* + **void remove():** Az ős függvényének bővítése, azzal, hogy **false** értékre állítja az **active** változót.

### WallField

###### Felelősség

Olyan mező objektum, mely egy falat reprezentál. Nem lehet rajta átmenni, és ismeri a szomszédait. A pálya szélei ilyen mezők, mindenképp fix mezőnek számít így a **BoxRecorder** nyilvántartásában mindenképp szerepel.

###### Ősosztályok

Field

###### Metódusok

* + **String** **toString():** Kiírja a Field típúsát (“wall”).
* + **Movable boxEnters(Box b, Direction d):** Megakadályozza, hogy egy **Box** rálépjen a mezőre. Ezt azzal éri el, hogy amikor rálép, akkor a paraméterben kapott objektum referenciáját adja vissza.
* + **Movable workerEnters(Worker w, Direction d):** Megakadályozza, hogy egy **Worker** rálépjen a mezőre. Ezt azzal éri el, hogy amikor rálép, akkor a paraméterben kapott objektum referenciáját adja vissza.

### Worker

###### Felelősség

A játékosok által irányított munkást reprezentáló osztály. Ő tudja a ládákat (**Box**) eltolni, illetve a pontokat is ő kapja majd. Ismeri a mezőt, ahol áll és tárolja a pontjait. Meg tud halni, ez többféleképpen is bekövetkezhet, ilyenkor eltűnik a pályáról.

###### Ősosztályok

Movable

###### Attribútumok

* **- int points:** a játékos pontjai, amiket az egyes ládák célmezőre juttatásáért kap.
* - **int force:** A játékos ereje, mely minden meghatározza, hogy el tudja-e tolni a dobozok sorát.
* - **ArrayList<Oil> oils:** Referenciák a munkás által lerakható olaj **Slime**-okra.
* - **ArrayList<Honey> honeys:** Referenciák a munkás által lerakható méz **Slime**-okra.
* - **int id:** A munkás azonosítója.

###### Metódusok

* + **void setId(int i):** Beállítja a munkás azonosítóját.
* + **int getPoints():** Visszaadja a játékos pontját.
* + **String toString():** Kiírja a munkás azonosítóját, valamint, hogy ő munkás(“wid”).
* + **void control(Direction d):** A munkás (**Worker**) 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 placeOil():** Függvény arra az esetre, ha a játékos egy olajmezőt helyez a pályára.
* **+ void placeHoney():** Függvény arra, ha a játékos egy mézet helyez a pályára.
* **+ void die():** Ez a függvény felel az entitás elpusztításáért, illetve csökkenti a munkások számát.
* + **void finalizeStep():** Ez az a függvény, ami az egyes lépések véglegesítésért felel.
* + **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 increasePoints():** A munkás pontjainak növelése.
* + **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. Ekkor ha a munkás visszalépendő mezőjén tartózkodik valami, akkor a munkás meghal.  
  Ennek a megvalósításáért részben felelős ez a metódus is.
* **+ void pushByBox(Box b, Direction d, int f):** Ez a függvény arra az esetre szolgál, hogyha a munkást egy dobozzal (**Box**) próbálnának meg eltolni. Továbbá továbbadja a kapott erőt a következőnek a sorban.
* **+ void pushByWorker(Worker w, Direction d, int f):** Ez a függvény arra az esetre szolgál, ha a munkást egy másik munkás próbálná meg eltolni. Továbbá továbbadja a kapott erőt a következőnek a sorban.
* + **void scorePoint(Direction d):** A pontok növeléséért felelős, méghozzá az *increasePoints()* függvény segítségével.

## A tesztek részletes tervei, leírásuk a teszt nyelvén

A tesztelések során használt pályák a következők:

test\_basic\_movement.txt:

[w w w w w w w w]

[w w w 1 e w w w]

[w s h x b e x w]

[w h s x h w w w]

[w w w w w w w w]

test\_basic\_pushes.txt:

[w w w w w w w w w]

[w x x 1 b x x x w]

[w x x x b b e x w]

[w s b x b h w x w]

[w x x x b h x x w]

[w w w w w w w w w]

basic\_map.txt:

[W W W W W W W]

[W X X X X 2 W]

[W X B B B X W]

[W 1 X X X X W]

[W W W W W W W]

second\_basic\_map.txt:

[W W W W W W W W]

[W X X X X X X W]

[W X B 2 B X X W]

[W X 1 X X X X W]

[W X X X X X X W]

[W W W W W W W W]

not\_so\_basic\_map.txt:

[W W W W W W W W]

[W X X X X X X W]

[W X 1 B B H X W]

[W 2 B S X X X W]

[W W W W W W W W]

### Move worker to SimpleField

* **Leírás**

A munkás egyet lép lefelé egy üres mezőre.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Az alapvető mozgatás funkciók tesztelése.

* **Bemenet**

load-map test\_basic\_movement.txt

control 1 -d

ls-workers

exit

* **Elvárt kimenet**

ID:1 X:3 Y:4 F:5

### Move worker to HoleField

* **Leírás**

A munkást elléptetjük egy olyan mezőig, melyen egy olyan lyuk található, melyhez nem tartozik kapcsoló. A munkás erre a mezőre lépve meghal.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Annak tesztelése, hogy a munkás egy lyukra lépve meghal.

* **Bemenet**

load-map test\_basic\_movement.txt

control 1 -d

control 1 -d

contorl 1 -r

ls-workers

exit

* **Elvárt kimenet**

### Move worker to closed HoleField

* **Leírás**

A munkást egy olyan mezőre léptetjük, amely zárva van, mert a hozzá tartozó kapcsolón nem áll semmi.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A bezárt lyuk mező sima mezőként viselkedése, ha egy munkás lép rá.

* **Bemenet**

load-map test\_basic\_movement.txt

control 1 -d

control 1 -l

ls-workers

exit

* **Elvárt kimenet**

ID:1 X:3 Y:3 F:5

### Move worker to empty EndField

* **Leírás**

A munkás elléptetjük jobbra egy üres cél mezőre.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Az üres cél mező egyszerű mezőként való viselkedése, amennyiben munkás lép rá.

* **Bemenet**

load-map test\_basic\_movement.txt

control 1 -r

ls-workers

exit

* **Elvárt kimenet**

ID:1 X:2 Y:5 F:5

### Move worker to non-empty EndField

* **Leírás**

A munkással egy dobozt tolunk a célmezőre és utána megpróbálunk rálépni. A doboz ezt nem fogja engedni, mert az már nem mozgatható.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Annak tesztelése, hogy a célmezőn lévő doboz eltolhatatlan-e.

* **Bemenet**

load-map test\_basic\_movement.txt

control 1 -d

control 1 -r

control 1 -r

ls-boxes

ls-workers

exit

* **Elvárt kimenet**

ID:b1 X:3 Y:6 true

ID:1 X:3 Y:5 F:3

### Move worker to SwitchField

* **Leírás**

A munkást egy olyan kapcsolóra léptetjük és megnézzük, hogy a hozzá tartozó lyuk aktiválódik-e.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A kapcsoló sima mezőként viselkedése, amennyiben egy munkás lép rá.

* **Bemenet**

load-map test\_basic\_movement.txt

control 1 -d

control 1 -d

control 1 -l

ls-fields

exit

* **Elvárt kimenet**

X:1 Y:1…8 wall

X:2 Y:1…3 wall

X:2 Y:4 simple x x

X:2 Y:5 endz x x

X:2 Y:6…8 wall

X:3 Y:1 wall

X:3 Y:2 switch 0 x x

X:3 Y:3 hole 0 x x

X:3 Y:4 simple x x

X:3 Y:5 simple x b1

X:3 Y:6 endz x x

X:3 Y:7 simple x x

X:3 Y:8 wall

X:4 Y:1 wall

**X:4 Y:2 hole 0 x x**

**X:4 Y:3 switch 0 x 1**

X:4 Y:4 simple x x

X:4 Y:5 hole 1 x x

X:4 Y:6…8 wall

X:5 Y:1…8 wall

### Move worker to WallField

* **Leírás**

A munkás jobbra lép egy fal mezőbe, azonban a fal mező ezt megakadályozza.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Annak tesztelése, hogy a falmező megakadályozza a munkás mozgását.

* **Bemenet**

load-map test\_basic\_movement.txt

control 1 -l

ls-workers

exit

* **Elvárt kimenet**

ID:1 X:2 Y:4 F:5

### Push Box with Worker on SimpleField

* **Leírás**

A munkást jobbra léptetjük, hogy eltoljon egy dobozt egy sima mezőre.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Alapvető tolási funkció ellenőrzése.

* **Bemenet**

load-map test\_basic\_pushes.txt

control 1 -r

ls-boxes

exit

* **Elvárt kimenet**

**ID:b1 X:2 Y:6 false**

ID:b2 X:3 Y:5 false

ID:b3 X:3 Y:6 false

ID:b4 X:4 Y:3 false

ID:b5 X:4 Y:5 false

ID:b6 X:5 Y:5 false

### Push Box with Worker on HoleField

* **Leírás**

A munkást egy olyan mezőre léptetjük előszőr, amitől jobbra egy doboz található mögötte pedig egy lyuk. Ezt követően megtoljuk a dobozt a munkással.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A lyuk viselkedése ha doboz kerül rá, azaz a doboz elpusztul-e.

* **Bemenet**

load-map test\_basic\_pushes.txt

control 1 -d

control 1 -d

control 1 -d

control 1 -r

ls-boxes

exit

* **Elvárt kimenet**

ID:b1 X:2 Y:5 false

ID:b2 X:3 Y:5 false

ID:b3 X:3 Y:6 false

ID:b4 X:4 Y:3 false

ID:b5 X:4 Y:5 false

### Push Box with Worker on non-empty EndField

* **Leírás**

A munkást először a megfelelő helyre léptetjük, majd először rátol egy dobozt a célmezőre. Ezt követően megpróbál még egyet rátolni, ez azonban már nem sikerül.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Teszteli, hogy a célmezőre tolt doboz valóban eltolhatatlan-e másik dobozzal szemben is.

* **Bemenet**

load-map test\_basic\_pushed.txt

control 1 -d

control 1 -r

control 1 -r

ls-boxes

exit

* **Elvárt kimenet**

ID:b1 X:2 Y:6 false

**ID:b2 X:3 Y:6 false**

**ID:b3 X:3 Y:7 true**

ID:b4 X:4 Y:3 false

ID:b5 X:4 Y:5 false

ID:b6 X:5 Y:5 false

### Push Box with Worker on SwitchField

* **Leírás**

A megfelelő helyre lépve először egy dobozt tolunk a zárt lyukra, majd azt követően egy dobozt tolunk a hozzá tartozó lyukra.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Teszteli, hogy a kapcsoló aktiválódik-e egy doboz rátolásával, valamint, hogy a lyukon lévő tárgy megsemmisül-e.

* **Bemenet**

load-map test\_basic\_pushes.txt

control 1 -d

control 1 -d

control 1 -r

control 1 -l

control 1 -l

ls-boxes

exit

* **Elvárt kimenet**

ID:b1 X:2 Y:6 false

ID:b2 X:3 Y:5 false

ID:b3 X:3 Y:6 false

**ID:b4 X:4 Y:2 false**

ID:b6 X:5 Y:5 false

### Push Box with Worker on WallField

* **Leírás**

A munkást először a megfelelő helyre léptetjük, majd rátol egy dobozt a falra. Ezt a fal megakadályozza.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Ellenőrzi, hogy a fal nem engedi, hogy a dobozt oda toljuk.

* **Bemenet**

load-map test\_basic\_pushes

control 1 -d

control 1 -d

control 1 -l

control 1 -l

ls-boxes

exit

* **Elvárt kimenet**

ID:b1 X:2 Y:6 false

ID:b2 X:3 Y:6 false

ID:b3 X:3 Y:7 false

**ID:b4 X:4 Y:2 false**

ID:b5 X:4 Y:5 false

ID:b6 X:5 Y:5 false

### Kill Worker by pushing into WallField

* **Leírás**

A munkással eltolunk egy dobozt, aki egy másik munkást a falhoz nyom.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A munkás meghal, ha a falnak toljuk egy dobozzal.

* **Bemenet**

1. load-map basic\_map.txt
2. control 1 -r
3. control 1 -u
4. control 1 -l
5. control 1 -u
6. control 1 -r
7. control 1 -r
8. control 1 -r
9. ls-workers
10. ls-boxes
11. ls-fields
12. exit

* **Elvárt kimenet**

ID:1 X:2 Y:4 F:5

ID:b0 X:2 Y:6 true

ID:b1 X:3 Y:4 false

ID:b2 X:3 Y:5 false

X:1 Y:1..7 wall

X:2 Y:1 wall

X:2 Y:2..4 simple x x

X:2 Y:5 simple x 1

X:2 Y:6 simple x b0

X:2 Y:7 wall

X:3 Y:1 wall

X:3 Y:2..3 simple x x

X:3 Y:4 simple x b1

X:3 Y:5 simple x b2

X:3 Y:6 simple x x

X:3 Y:7 wall

X:4 Y:1 wall

X:4 Y:2..6 simple x x

X:4 Y:7 wall

X:5 Y:1..7 wall

### Worker pushed by worker

* **Leírás**

A munkással el szeretnénk tolni egy szomszédos mezőn lévő munkást.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A munkások a lépés után is a helyükön kell, hogy maradjanak.

* **Bemenet**

1. load-map basic\_map.txt
2. control 1 -r
3. control 2 -d
4. control 1 -r
5. control 2 -d
6. control 2 -l
7. control 2 -l
8. ls-workers
9. ls-fields
10. exit

* **Elvárt kimenet**

ID:1 X:4 Y:4 F:5

ID:2 X:4 Y:5 F:5

ID:b0 X:3 Y:3 false

ID:b1 X:3 Y:4 false

ID:b2 X:3 Y:5 false

X:1 Y:1..7 wall

X:2 Y:1 wall

X:2 Y:2..6 simple x x

X:2 Y:7 wall

X:3 Y:1 wall

X:3 Y:2 simple x x

X:3 Y:3 simple x b0

X:3 Y:4 simple x b1

X:3 Y:5 simple x b2

X:3 Y:6 simple x x

X:3 Y:7 wall

X:4 Y:1 wall

X:4 Y:2..3 simple x x

X:4 Y:4 simple x 1

X:4 Y:5 simple x 2

X:4 Y:6 simple x x

X:4 Y:7 wall

X:5 Y:1..7 wall

### Push several Boxes with Worker

* **Leírás**

A munkással egymás után lévő több doboz eltolása.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A munkás ereje 5, amivel mindössze 2 doboz eltolása lehetséges (egy dobozhoz 2 erő kell, ha nincs alatta semmi). Így itt nem szabad, hogy eltolja a dobozokat.

* **Bemenet**

1. load-map basic\_map.txt
2. control 1 -u
3. control 1 -r
4. ls-workers
5. ls-boxes
6. ls-fields
7. exit

* **Elvárt kimenet**

ID:1 X:3 Y:2 F:5

ID:2 X:2 Y:6 F:5

ID:b0 X:3 Y:3 false

ID:b1 X:3 Y:4 false

ID:b2 X:3 Y:5 false

X:1 Y:1..7 wall

X:2 Y:1 wall

X:2 Y:2..5 simple x x

X:2 Y:6 simple x 2

X:2 Y:7 wall

X:3 Y:1 wall

X:3 Y:2 simple x 1

X:3 Y:3 simple x b0

X:3 Y:4 simple x b1

X:3 Y:5 simple x b2

X:3 Y:6 simple x x

X:3 Y:7 wall

X:4 Y:1 wall

X:4 Y:2..6 simple x x

X:4 Y:7 wall

X:5 Y:1..7 wall

### Push several Boxes with Worker (with Oil)

* **Leírás**

A munkással egymás után lévő több doboz eltolása, amelyek között van olajos mező.

(A tesztben először lerakunk kettő olajat, aztán pedig az olajos mezővel azonos sorba rendezzük a dobozokat, majd végül pedig eltoljuk őket.)

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A munkás ereje 5. Amennyiben egy láda alatt nincs semmi, akkor 2 erő szükséges az eltolásához. Esetünkben 3 ládát fogunk eltolni, 2 alatt lesz olaj (így azok eltolásához csak 1 erő kell), azaz így már sikeres lesz az eltolás.

Ebben a tesztben sorra kerül az „Oil”, azaz olaj lerakás tesztelése is, hiszen kezdetben 2 mezőre is lerakunk olajat.

* **Bemenet**

1. load-map basic\_map.txt
2. control 1 -r
3. drop-special 1 -o
4. control 1 -r
5. drop-special 1 -o
6. control 1 -l
7. control 1 -l
8. control 1 -u
9. control 1 -u
10. control 1 -r
11. control 1 -d
12. control 1 -u
13. control 1 -r
14. control 1 -d
15. control 1 -u
16. control 1 -r
17. control 1 -d
18. control 1 -r
19. control 1 -d
20. control 1 -l
21. ls-workers
22. ls-boxes
23. ls-fields
24. exit

* **Elvárt kimenet**

ID:1 X:4 Y:5 F:5

ID:2 X:2 Y:6 F:5

ID:b0 X:4 Y:2 true

ID:b1 X:4 Y:3 false

ID:b2 X:4 Y:4 false

X:1 Y:1..7 wall

X:2 Y:1 wall

X:2 Y:2..5 simple x x

X:2 Y:6 simple x 2

X:2 Y:7 wall

X:3 Y:1 wall

X:3 Y:2..6 simple x x

X:3 Y:7 wall

X:4 Y:1 wall

X:4 Y:2 simple x b0

X:4 Y:3 simple o b1

X:4 Y:4 simple o b2

X:4 Y:5 simple x 1

X:4 Y:6 simple x x

X:4 Y:7 wall

X:5 Y:1..7 wall

### Push several Boxes with Worker (with Honey)

* **Leírás**

A munkással egymás után lévő több doboz eltolása, amelyek között van mézes mező.

(A tesztben először lerakunk egy mézet, aztán pedig a mézes mezővel azonos sorba rendezzük a dobozokat, majd végül pedig eltoljuk őket.)

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A munkás ereje 5. Amennyiben egy láda alatt nincs semmi, akkor 2 erő szükséges az eltolásához. Esetünkben 2 ládát fogunk eltolni, 1 alatt lesz méz (a mézes mezőn lévő doboz eltolásához 3 erő kell), azaz így is siker lesz a tolás eredménye.

Ebben a tesztben sorra kerül a „Honey”, azaz méz lerakás tesztelése is, hiszen kezdetben 1 mezőre lerakunk mézet.

* **Bemenet**

1. load-map basic\_map.txt
2. control 1 -r
3. drop-special 1 -h
4. control 1 -l
5. control 1 -u
6. control 1 -u
7. control 1 -r
8. control 1 -d
9. control 1 -u
10. control 1 -r
11. control 1 -d
12. control 1 -l
13. control 1 -l
14. control 1 -d
15. control 1 -r
16. ls-workers
17. ls-boxes
18. ls-fields
19. exit

* **Elvárt kimenet**

ID:1 X:4 Y:3 F:5

ID:2 X:2 Y:6 F:5

ID:b0 X:4 Y:4 false

ID:b1 X:4 Y:5 false

ID:b2 X:3 Y:5 false

X:1 Y:1..7 wall

X:2 Y:1 wall

X:2 Y:2..5 simple x x

X:2 Y:6 simple x 2

X:2 Y:7 wall

X:3 Y:1 wall

X:3 Y:2..4 simple x x

X:3 Y:5 simple x b2

X:3 Y:6 simple x x

X:3 Y:7 wall

X:4 Y:1 wall

X:4 Y:2 simple x x

X:4 Y:3 simple h 1

X:4 Y:4 simple x b0

X:4 Y:5 simple x b1

X:4 Y:6 simple x x

X:4 Y:7 wall

X:5 Y:1..7 wall

### Push Boxes and Worker (with Oil)

* **Leírás**

A munkással el szeretnénk tolni egy doboz, munkás, doboz sort, miközben a dobozok alatt olaj van. (A tesztben először lerakunk kettő olajat, aztán pedig az olajos mezővel azonos sorba rendezzük a dobozokat, majd végül pedig eltoljuk őket.)

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Mivel a dobozok alatt olaj van, ezért egy-egy doboz eltolása csak 1 erőegységbe kerül (munkás ereje 5 erőegység), így a dobozok között lévő munkás is el fog tolódni a dobozokkal együtt.

* **Bemenet**

1. load-map map19.txt
2. drop-special 1 -o
3. control 1 -r
4. control 1 -r
5. drop-special 1 -o
6. control 2 -d
7. control 1 -r
8. control 1 -u
9. control 1 -u
10. control 1 -l
11. control 1 -d
12. control 1 -u
13. control 1 -l
14. control 1 -l
15. control 1 -d
16. control 1 -l
17. control 1 -d
18. control 1 -r
19. ls-workers
20. ls-boxes
21. ls-fields
22. exit

* **Elvárt kimenet**

ID:1 X:4 Y:3 F:5

ID:2 X:4 Y:5 F:5

ID:b0 X:4 Y:4 false

ID:b1 X:4 Y:6 false

X:1 Y:1..8 wall

X:2 Y:1 wall

X:2 Y:2..7 simple x x

X:2 Y:8 wall

X:3 Y:1 wall

X:3 Y:2..7 simple x x

X:3 Y:8 wall

X:4 Y:1 wall

X:4 Y:2 simple x x

X:4 Y:3 simple x 1

X:4 Y:4 simple o b0

X:4 Y:5 simple x 2

X:4 Y:6 simple o b1

X:4 Y:7 simple x x

X:4 Y:8 wall

X:5 Y:1 wall

X:5 Y:2..7 simple x x

X:5 Y:8 wall

X:6 Y:1..8 wall

### Push Boxes and Worker (with Honey)

* **Leírás**

A munkással el szeretnénk tolni egy doboz, munkás, doboz sort, miközben a dobozok alatt olaj van. (A tesztben először lerakunk kettő mézet, aztán pedig az mézes mezővel azonos sorba rendezzük a dobozokat, majd végül pedig eltoljuk őket.)

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Mivel a dobozok alatt méz van, ezért egy-egy doboz eltolása 3 erőegységbe kerül (munkás ereje 5 erőegység), így a második, a munkás túloldalán lévő dobozt már nem tudjuk majd eltolni, viszont 1 dobozt még igen, ezzel a két doboz lévő munkás halálát okozva.

* **Bemenet**

1. load-map second\_basic\_map.txt
2. drop-special 1 -h
3. control 1 -r
4. control 1 -r
5. drop-special 1 -h
6. control 2 -d
7. control 1 -r
8. control 1 -u
9. control 1 -u
10. control 1 -l
11. control 1 -d
12. control 1 -u
13. control 1 -l
14. control 1 -l
15. control 1 -d
16. control 1 -l
17. control 1 -d
18. control 1 -r
19. ls-workers
20. ls-boxes
21. ls-fields
22. exit

* **Elvárt kimenet**

ID:1 X:4 Y:3 F:5

ID:b0 X:4 Y:4 false

ID:b1 X:4 Y:5 false

X:1 Y:1..8 wall

X:2 Y:1 wall

X:2 Y:2..7 simple x x

X:2 Y:8 wall

X:3 Y:1 wall

X:3 Y:2..7 simple x x

X:3 Y:8 wall

X:4 Y:1 wall

X:4 Y:2 simple x x

X:4 Y:3 simple x 1

X:4 Y:4 simple o b0

X:4 Y:5 simple o b1

X:4 Y:6 simple x x

X:4 Y:7 simple x x

X:4 Y:8 wall

X:5 Y:1 wall

X:5 Y:2..7 simple x x

X:5 Y:8 wall

X:6 Y:1..8 wall

### Scheduling test 1.

* **Leírás**

Egy dobozt egyszerre két munkás is elszeretné tolni különböző irányba (pontosan egymással ellentétes irányba, azaz egymással szembe).

Az elvárt viselkedés a parancsok kiadásának sorrendjétől függ, hiszen nem tudunk egy időpillanatban egyszerre két parancsot is kiadni. (pl. billentyűzet gombjának lenyomását sem egyszerre, hanem csak egymás továbbítódnak a számítógép felé)

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A munkások a lépés után is a helyükön kell, hogy maradjanak, hiszen egyet jobbra majd egyet balra tolnak egymáson (kvázi egyszerre).

* **Bemenet**

1. load-map second\_basic\_map.txt
2. control 1 -l
3. control 1 -u
4. control 1 -r
5. control 2 -l
6. ls-workers
7. ls-boxes
8. ls-fields
9. exit

* **Elvárt kimenet**

ID:1 X:3 Y:2 F:5

ID:2 X:3 Y:4 F:5

ID:b0 X:3 Y:5 false

ID:b1 X:3 Y:3 false

X:1 Y:1..8 wall

X:2 Y:1 wall

X:2 Y:2..7 simple x x

X:2 Y:8 wall

X:3 Y:1 wall

X:3 Y:2 simple x 1

X:3 Y:3 simple x b0

X:3 Y:4 simple x 2

X:3 Y:5 simple x b1

X:3 Y:6..7 simple x x

X:3 Y:8 wall

X:4 Y:1 wall

X:4 Y:2..7 simple x x

X:4 Y:8 wall

X:5 Y:1 wall

X:5 Y:2..7 simple x x

X:5 Y:8 wall

X:6 Y:1..8 wall

### Scheduling test 2.

* **Leírás**

Egy sor dobozt tol az egyik munkás egy zárt lyukon (HoleField) keresztül, miközben a lyukat/kapcsolót aktiválja egy másik munkás által eltolt doboz.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A HoleField-re tolt ládának meg kell semmisülnie, amennyiben kinyitják alatta a lyukat.

* **Bemenet**

1. load-map not\_so\_basic\_map.txt
2. control 1 -r
3. control 2 -r
4. ls-workers
5. ls-boxes
6. ls-fields
7. exit

* **Elvárt kimenet**

ID:1 X:3 Y:4 F:5

ID:2 X:4 Y:3 F:5

ID:b0 X:3 Y:5 false

ID:b2 X:4 Y:4 false

X:1 Y:1..8 wall

X:2 Y:1 wall

X:2 Y:2..7 simple x x

X:2 Y:8 wall

X:3 Y:1 wall

X:3 Y:2 simple x x

X:3 Y:3 simple x x

X:3 Y:4 simple x 1

X:3 Y:5 simple x b0

X:3 Y:6 hole 1 x x

X:3 Y:7 simple x x

X:3 Y:8 wall

X:4 Y:1 wall

X:4 Y:2 simple x x

X:4 Y:3 simple x 2

X:4 Y:4 switch 1 x b2

X:4 Y:5..7 simple x x

X:4 Y:8 wall

X:5 Y:1..8 wall

## A tesztelést támogató programok tervei

Az automatikus tesztelést segíti egy tesztellenőrző program, amelynek egy teszteset által generált kimenetet (a teszteset futtatása során naplózott fájlba) és egy elvárt kimenetet kell megadni (tesztelő által leírt, ellenőrzött). A program összehasonlítja a két fájlt és a különbségeket jelzi. Az összehasonlítás szöveg szinten történik.

### Osztályok és metódusok tervei

#### TestValidater

#### Felelősség

Ez egy tesztelést támogató külön program, melynek feladata, hogy egy fájlban található **elvárt** teszteredményt összehasonlítson, egy szintén fájlban található **tényleges** teszteredménnyel. Az összehasonlítás karakter alapon történik, az eltérő részt jelzi.

#### Ősosztályok

-

#### Attribútumok

-

#### Metódusok

* **+ void validate(String expectedFile, String actualFile):** Fogja a két fájlban található szöveget, beolvassa őket, és az első paramétert mintául véve összehasonlítja. Amennyiben eltérést talál azt jelzi.
* **-** **String readFromFile(String fileName):** Adott szöveges fájlt beolvas és visszaadja a tartalmát.

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2018.03.30. 18:00 | 2 óra | Bertalan  Tolnai  Horváth  Mátyás  Gurubi | **Értekezlet**.  **Döntés**: Horváth elkészíti az osztályok és metódusok terveinek első harmadát, Gurubi a második harmadát, Mátyás a maradék harmadát, Tolnai és Bertalan pedig a teszteket készítik el.  Határidő vasárnap délelőtt (04.08). |
| 2018.04.01 18:00 | 2 óra | Bertalan | **Tevékenység**: Bertalan elkezdi a tesztek részletes terveinek leírását. |
| 2018.04.02 17:00 | 3 óra | Tolnai | **Tevékenység**: Tolnai elkezdi a tesztek bementetét leírni. |
| 2018.04.03 19:00 | 2 óra | Mátyás | **Tevékenység**: Osztályok részletes terveinek leírása. |
| 2018.04.03 10:00 | 2 óra | Horváth | **Tevékenység**: Osztályok metódusainak tisztázása. |
| 2018.04.05 11:00 | 2 óra | Gurubi | **Tevékenység**: A korábban változott osztályok és metódusaik leírása és átgondolása. |
| 2018.04.05 15:00 | 2 óra | Tolnai | **Tevékenység**: Tesztek leírásának folytatása. |
| 2018.04.07 19:00 | 3 óra | Bertalan | **Tevékenység**: Bertalan befejezi a tesztek leírását. |
| 2018.04.07 09:00 | 3 óra | Horváth | **Tevékenység**: Osztály leírások és metódusok befejezése. |
| 2018.04.08 13:00 | 2 óra | Mátyás | **Tevékenység**: Osztály leírások befejezése, egyeztetése. |
| 2018.04.08 15:00 | 2 óra | Gurubi | **Tevékenység**: Korábban változott osztályok és metódusaik leírásának befejezése. |
| 2018.04.08 19:00 | 2 óra | Mátyás  Gurubi | **Döntés és Tevékenység**: Dokumentum összeállítása, felmerülő hibák, problémák javítása. |