Szoftver projekt laboratórium

Részletes tervek

Csapat

61 - Zeusz

Konzulens

Dr. Simon Balázs

Csapattagok

Balla Gergely	[NEPTUN]	[email]
Joób Zalán Miklós	[NEPTUN]	[email]
Nagy Alexandra	[NEPTUN]	[email]
Tóth Boldizsár	[NEPTUN]	[email]
Zelei Mátyás	[NEPTUN]	[email]

2024. április 15.

8. fejezet

Részletes tervek

8.0. Javítások

8.0.1. Prototípus interface definíciója

8.0.1.1. Bemeneti nyelv

Az interfész parancsokat vár a bemenetre. Egy parancs egy egyszavas parancsnévből és egyszavas argumentumokból áll. A parancs a parancsnévvel kezdődik, utána szóközzel elválasztva következnek az argumentumok. A parancsok:

- Add <holder> <object>
  - Leírás: Hozzáadja az első paraméterül megadott szoba vagy személy megfelelő tárolójához a második paraméterül megadott objektumot.
  - Első paraméter: Az első paraméter Person leszármazott, vagy szoba.
    - Janitor
    - Room
    - Student
    - Teacher
  - Második paraméter: A második paraméter lehet az Effect, Person vagy Item osztályok leszármazottjának példánya.
    - AirFreshener
    - Beer
    - BeerEffect
    - Camembert
    - CleanEffect
    - CodeOfStudies
    - Door
    - FakeCodeOfStudies
    - FakeMask
    - FakeSlideRule
    - GasEffect
    - Janitor (Person leszármazott esetén nem)

    * JanitorEffect
    * Mask
    * MaskEffect
    * RagEffect
    * Slide Rule
    * Student (Person leszármazott esetén nem)
    * Teacher (Person leszármazott esetén nem)
    * Transistor
    * WetRag

- Create <class> <name> [<constructor params>]
  - Leírás: Létrehoz egy objektum példányt a prototípusban megadott osztályból, megadott névvel.
  - Első paraméter: Az első paraméter határozza meg, hogy milyen osztályú objektumot hozunk létre. A paraméternek egyeznie kell valamelyik létrehozható osztály nevével:
    * AirFreshener
    * Beer
    * BeerEffect
    * Camembert
    * CleanEffect
    * CodeOfStudies
    * Door: a konstruktor paraméterekhez oda kell írni a két szoba nevét, amelyeket az ajtó összeköt: `Create Door d1 r1 r2`
    * FakeCodeOfStudies
    * FakeMask
    * FakeSlideRule
    * GasEffect
    * Janitor
    * JanitorEffect
    * Mask
    * MaskEffect
    * RagEffect
    * Room: opcionális paraméter a kapacitás, alapértelmezetten négy.
    * Student
    * SlideRule
    * Teacher
    * Transistor
    * WetRag
  - Második paraméter: A második paraméter az objektum neve a prototípusban. Ilyen néven kell hivatkozni rá további parancsok kiadásakor.

- Drop <person> <item>

  - Leírás: Az első paraméterként megadott Person leszármazott objektum eldobja a második paraméterként megadott Item leszármazott objektumot, ha az nála van.

- Link <transistor1> <transistor2>
  - Leírás: Összelinkeli a két megadott Transistor objektumot.
- Merge <room1> <room2>
  - Leírás: Beleolvasztja az első paraméterül megadott Room objektumba a második paraméterül megadott Room objektumot.
- Move <person> <door>
  - Leírás: Az első paraméterként megadott Person leszármazott objektum használja a második paraméterként megadott Door objektumot.
- Oneway <door> <value>
  - Leírás: Megadott Door objektum egyirányúra állítása
  - Opciók: value egy boolean, lehet true vagy false, hogy egyirányú-e az ajtó.
- Pickup <person> <item>
  - Leírás: Egy ember felvesz egy tárgyat.
  - Első paraméter: Az első paraméter a Person osztály valamely leszármazottjának, tehát az alábbi osztályok példányának neve.
    * Janitor
    * Student
    * Teacher
  - Második paraméter: A második paraméter lehet az Effect vagy Item osztályok leszármazottjának példánya.
    * AirFreshener
    * Beer
    * BeerEffect
    * Camembert
    * CleanEffect
    * CodeOfStudies
    * FakeCodeOfStudies
    * FakeMask
    * FakeSlideRule
    * GasEffect
    * JanitorEffect
    * Mask
    * MaskEffect
    * RagEffect

    * SlideRule
    * Transistor
    * WetRag

- Seed <seed>
  - Leírás: Véletlen szám generálásához használt seed beállítása. Ezzel determinisztikussá tehető minden véletlen esemény a tesztelés elősegítése érdekében.
- Split <room>
  - Leírás: Szétválasztja a paraméterül megadott Room objektumbot.
- Status <object>
  - Leírás: Kiiratja a megadott objektum aktuális állapotát.
- Use <person> <item>
  - Leírás: Az első paraméterként megadott Person leszármazott objektum használja a második paraméterként megadott Item leszármazott objektumot, ha az nála van.
- Update <deltaTime> [<object>]
  - Leírás: Meghívja a paraméterül megadott objektum frissítő metódusát.
  - Első paraméter: az előző Update óta eltelt idő.
  - Második paraméter: Egy Updatable interfészt megvalósító objektum:
    * BeerEffect
    * CleanEffect
    * GasEffect
    * Janitor
    * JanitorEffect
    * MaskEffect
    * RagEffect
    * Room
    * Student
    * Teacher
  - Opciók: Az object nem kötelező paraméter, ha nincs megadva, akkor minden objektumot frissít (ahogyan ez játék közben is történne).

Példák a bemeneti nyelvre:

```
Create Room r1
Create Room r2
Create Door d r1 r2
Status d

Create Student s
Create Mask m
Add s m
Update
```

8.0.1.2. Kimeneti nyelv

Az alábbi parancsok végrehajtása során a futtatás sikeressége vagy eredmény a kimeneten megjelenik.

- Merge
  - Van elég hely az új szobában: Nincs kimenet
  - Nincs elég hely az új szobában: Minden olyan emberhez, akinek nincs helye, egy sor fog tartozni a kimenetben. Ha át tudott menni egy szomszédos szobába, az alábbi sor jelenik meg:

    ```
    <person> moved to <room>.
    ```

    Amennyiben már nincs szomszédos szoba, melybe át lehet menni, az ember meghal:

    ```
    <person> died.
    ```

- Move
  - Van elég hely az ajtó túloldalán, és az ajtó járható erre:

    ```
    <person> moved to <room>.
    ```

  - Nem járható az ajtó ebben az irányban:

    ```
    <person> couldn't use the door.
    ```

  - Járható az ajtó, de nincs hely a túloldalon:

    ```
    The other room is full.
    ```

- Pickup
  - Sikerült felvenni: Nincs kimenet.
  - Nem sikerült felvenni, tele volt az inventory:

    ```
    Inventory is full.
    ```

- Update
  - Paraméter nélküli hívás esetén a kimenet megegyezik azzal, mintha az összes szobára létrehozási sorrendben meghívnánk az Update-et.
  - Room típusú paraméter esetén először az effektusokon megy végig, amennyiben az effektus ideje lejárt, azt az alábbi módon kiírja:

    ```
    <effect> ran out of time.
    ```

Ezen kívül a CleanEffect és a JanitorEffect a szobára is alkalmazható, ekkor az alábbi üzenet jelenik meg:

```
<cleanEffect> cleaned the room.
```

Ha ezalatt a mérgező gázt is eltávolították, utána ez fog megjelenni:

```
<cleanEffect> removed <gasEffect>.
```

Ezután az emberek következnek, rájuk kerülnek a szoba effektusai:

```
<person> got kicked out.
<person> got knocked out.
<teacher> became peaceful.
<student> was saved by beer.
```

Majd az effekt rákerülése után az emberek saját maguk is frissülnek:

```
<student> was protected by mask.
<teacher> attacked everyone.
<student> was eliminated.
<student> was revived.
```

- Status
  - A parancs hatására a program kiírja a kért objektum adatatait. Egy sorba kiírja az osztályt és nevet. Ezután a belső állapotának változóit írja ki külön sorokba. Ha egy belső változó egy kollekció akkor ennek elemeit írja ki szóközzel elválasztva. (Ha az elemek objektumok, akkor a prototípusban szereplő nevüket írja ki.)
    A kimenet különböző osztályok esetén:
    * AirFreshener:
      ```
      AirFreshener <name>
      Person: <person>
      Room: <room>
      ```
    * Beer:
      ```
      Beer <name>
      Person: <person>
      Room: <room>
      ```
    * BeerEffect:
      ```
      BeerEffect <name>
      Holder: <holder>
      Time remaining: <timeRemaining>
      ```
    * Camembert:

```
Camembert <name>
Person: <person>
Room: <room>
```

* CleanEffect:

```
CleanEffect <name>
Holder: <holder>
```

* CodeOfStudies:

```
CodeOfStudies <name>
Person: <person>
Room: <room>
Uses: <uses>
```

* Door:

```
Door <name>
Hidden: <hidden>
One-way: <oneWay>
Room 1: <room1>
Room 2: <room2>
```

* FakeCodeOfStudies:

```
FakeCodeOfStudies <name>
Person: <person>
Room: <room>
```

* FakeMask:

```
FakeMask <name>
Person: <person>
Room: <room>
```

* FakeSlideRule:

```
FakeSlideRule <name>
Person: <person>
Room: <room>
```

* GasEffect:

```
GasEffect <name>
Holder: <holder>
Time remaining: <timeRemaining>
```

* Janitor:

```
Janitor <name>
Effects: <effects>
Inventory: <items in inventory>
```

```
Knock-out time: <knockOutTime>
```

* JanitorEffect:

```
JanitorEffect <name>
Holder: <holder>
```

* Mask:

```
Mask <name>
Person: <person>
Room: <room>
Uses: <uses>
```

* MaskEffect:

```
MaskEffect <name>
Holder: <holder>
Time remaining: <timeRemaining>
```

* RagEffect:

```
RagEffect <name>
Holder: <holder>
Time remaining: <timeRemaining>
```

* Room:

```
Room <name>
Capacity: <capacity>
Doors: <doors>
Effects: <effects>
Items: <items>
People: <people>
```

* SlideRule:

```
SlideRule <name>
Person: <person>
Room: <room>
```

* Student:

```
Student: <name>
Effects: <effects>
Eliminated: <eliminated>
Inventory: <items in inventory>
Knock-out time: <knockOutTime>
Room: <room>
```

* Teacher:

```
Teacher: <name>
```

```
Effects: <effects>
Inventory: <items in inventory>
Knock-out time: <knockOutTime>
Room: <room>
```

* Transistor:

```
Transistor <name>
Pair: <other>
Person: <person>
Room: <room>
Target room: <target>
```

* WetRag:

```
WetRag <name>
Person: <person>
Room: <room>
```

– Egyéb:

Amikor egy hallgató felvette a logarlécet, az alábbi szöveg jelenik meg:

```
The students have won.
```

8.0.2. A tesztterv

Ebbe a dokumentumba csak azokat a teszttterveket raktuk bele, melyeken változtatnunk kellett.

Teszt-eset neve	ApplyMaskEffectToStudent
Rövid leírás	A teszt létrehoz egy Diákot és egy Maszkot. A Maszkot a diákkal felveteti. Ellenőrzi, hogy a diák rendelkezik-e a megfelelő paraméterű MaskEffect-el.
Teszt célja	A Maszk által hordozott Effect Diákra való terjedésének tesztelése.

Megjegyzés: A fenti két teszt elvégezhető lenne az összes tárgyra és szereplőre is, de mivel az ősosztályaik közösek és a FakeItem implementációknál metódusok kerülnek törlésre, ezért ezekre külön teszteket nem terveztünk.

Teszt-eset neve	UpdateBeerEffect
Rövid leírás	A teszt létrehoz egy Diákot és egy Beer effectet. Az effecet a diákra rakja. Meghívja a Diák update függvényét. Ellenőrzi, hogy frissült-e az effect.
Teszt célja	Az effect frissítésének ellenőrzése.

Teszt-eset neve	UpdateRagEffect
Rövid leírás	A teszt létrehoz egy Diákot és egy RagEffectet. Az effectet a diákra rakja. Meghívja a Diák update függvényét. Ellenőrzi, hogy frissült-e az effect.
Teszt célja	Az effect frissítésének ellenőrzése.

Teszt-eset neve	UpdateMaskEffect
Rövid leírás	A teszt létrehoz egy Diákot és egy Maszkot. A Maszokt a diákkal felveteti. Meghívja a Diák update függvényét. Ellenőrzi, hogy frissült-e az effect.
Teszt célja	Az effect frissítésének ellenőrzése.

Teszt-eset neve	UpdateGasEffect
Rövid leírás	A teszt létrehoz egy Szobát, egy Diákot és egy Camember-et. A Camembert-et felveteti és kibontatja a Diákkal. Meghívja a Szoba update függvényét. Ellenőrzi, hogy frissült-e az effect.
Teszt célja	Az effect frissítésének ellenőrzése.

Teszt-eset neve	PickUpItem
Rövid leírás	A teszt létrehoz egy szobát, egy hallgatót és egy Sört A hallgatót és a Sört a szobába helyezi. A hallgatóval felveteti a Sört. Ellenőrzi, hogy a Sör a hallgatóhoz került-e.
Teszt célja	A felvétel funkció ellenőrzése.

Megjegyzés: Elegendő egy tetszőleges típusú Player-el és egy tetszőleges típusú tárgyal tesztelni, mivel a teszetesethez releváns metódusok az absztrakt osztályokban kerültek definiálásra.

Teszt-eset neve	TeleportUsingTransistors
Rövid leírás	A teszt létrehoz két szobát(szobaA, szobaB), két tranzisztort(tranzisztorA, tranzisztorB) és egy hallgatót. A hallgatót elhelyezi szobaA-ban. A tranzisztorokat összelinkeli. TranzisztorA-t odaadja a hallagtónak, tranzisztorB-t pedig elhelyezi szobaB-ben. A hallgatót teleportáltatja TranzisztorA segítségével. Ellenőrzi, hogy a diák átkerült-e szobaB-be.
Teszt célja	A teleportálás funkció működésének bizonyítása.

Teszt-eset neve	BeerProtectStudent
Rövid leírás	A teszt létrehoz egy szobát, egy Hallgatót és egy Oktatót egy Sört és egy maszkot. A Sört és a maszkot felveteti a Hallgatóval. A Hallgatót és Oktatót a Szobába rakja. Ellenőrzi, hogy az oktató nem támadja meg a Hallgatót.
Teszt célja	A Sör védelemnyújtásának ellenőrzése, miközben a hallgató elejti a nála lévő másik tárgyat.

Teszt-eset neve	KnockOutWithOutMask
Rövid leírás	A teszt létrehoz egy Szobát, egy Diákot és egy Camember-et. A Diákot a szobába helyezi. A diákkal felveteti és kibontatja a sajtot Ellenőrzzi, hogy a Diák elájult-e.
Teszt célja	A Cammabret működésének ellenőrzése.

Megjegyzés: Ezt a tesztesetet a Teacher és Janitor objektumra is elvégezhetnénk, de a tesztben releváns metódusok a Student és Teacher esetében azonosak, mivel mind a hárman a Person osztály leszármazottai.

Teszt-eset neve	KnockOutWithMask
Rövid leírás	A teszt létrehoz egy Szobát, egy Diákot, Maszkot és egy Camember-et. A Maszkot a Diákra helyezi. A Camembert-et felveteti és kibontatja a Diákkal. A Diákot a szobába helyezi. Ellenőrzzi, hogy a Diák nem elájult
Teszt célja	A Maszk működésének ellenőrzése.

Megjegyzés: Ezt a tesztesetet a Teacher és Janitor objektumra is elvégezhetnénk, de a tesztben releváns metódusok a Student és Teacher esetében azonosak, mivel mind a hárman a Person osztály leszármazottai.

8.1. Osztályok és metódusok tervei

8.1.1. AirFreshener

- Felelősség

A tárgy használatakor a szoba levegőjét megtisztítja.

- Interfészek

-

- Ősosztály

Item

- Attribútumok

-

- Metódusok
  - ◇ *+use() : void* - Létrehoz egy CleanEffectet a szobára, majd önmegsemimisít.

8.1.2. Beer

- Felelősség

Megvédi a hallgatót az oktatóktól.

- Interfészek

-

- Ősosztály

Item

- Attribútumok

-

- Metódusok
  ◇ *+use() : void* - Egy BeerEffectet rak a személyre, akinél van, majd önmegsemimisít. Használakor a hallgató elejt egy nála lévő tárgyat.

```
create beerEffect
add beerEffect to holder
remove self from the holder
make the holder drop a random item
```

.

8.1.3. BeerEffect

- Felelősség

A Beer hatásának megvalósítása: védelem az oktatókkal szemben.

- Interfészek

Updatable

- Ősosztály

Effect

- Attribútumok

-

- Metódusok
  ◇ *+applyToStudent(target : Student) : void* - Megvédi a játékost az oktatóktól, közben a játékos elejt egy nála lévő tárgyat.

8.1.4. Camembert

- Felelősség

A tárgy használatakor a szoba levegőjét mérgzővé teszi.

- Interfészek

-

- Ősosztály

Item

- Attribútumok

-

- Metódusok
  ◇ *+use() : void* - Egy GasEffectet rak a szobára, ahol van, majd önmegsemimisít.

8.1.5. CleanEffect

- Felelősség

A mérgező szobák levegőjét megtisztítja.

- Interfészek

Updatable

- Ősosztály

Effect

- Attribútumok

-

- Metódusok
  - ◇ *+applyToRoom(target : Room) : void* - A szoba levegőjét megtisztíttatja.

8.1.6. CodeOfStudies

- Felelősség

A hallgató használhatja, hogy megvédje magát egy oktatóval szemben.

- Interfészek

-

- Ősosztály

Item

- Attribútumok
  - ◇ *-uses : int* - A tárgy használatának száma.
- Metódusok
  - ◇ *+useAgainst(target : Teacher) : void* - Megvédi a birtokosát az oktatótól. Ha a *uses* 0-ra csökken megsemmisül.

8.1.7. Door

- Felelősség

Ajtó. Két Room között megy. Lehet egyirányú vagy kétirányú. Rajta keresztül lépnek át a személyek a szobákba.

- Interfészek

-

- Ősosztály

-

- Attribútumok
  - ◇ *-from : Room* - Első szoba, ahova az ajtó nyílik.
  - ◇ *-to : Room* - Második szoba, ahova az ajtó nyílik.

◇ *-oneWay : boolean* - Ha igaz, egyirányú az ajtó. Ilyen esetben csak $from \longrightarrow to$ irányban lehet átlépni rajta. Alapértelmezetten false.
◇ *-hidden : boolean* - Ha igaz, rejtve van az ajtó. Ilyen esetben nem lehet rajta átlépni.

■ Metódusok

◇ *+Door(from : Room, to : Room)* - Konstruktor, ami megadja a szobákat. A szobákhoz hozzáadja az ajtót.
◇ *+use(person : Person, useFrom : Room) : void* - Ajtó használata: a személy átlép a másik szobába, ha teheti. *person*: a személy, aki használja az ajtót. *useFrom*: a szoba ahonnan jön a személy. Rejtett ajtón nem lehet átlépni. A *useFrom* paraméternek egyeznie kell az ajtóhoz tartozó egyik szobával. Ha az ajtó egyirányú, akkor csak abban az esetben lehet átlépni rajta, ha a *useFrom* paraméter megegyezik az ajtó *from* attribútumával. Ha az ajtón át tud lépni, akkor megpróbálja beléptetni a *person*-t a másik szobába. Ha az átlépés sikeres, a person eltávolítódik a kiinduló szobából.

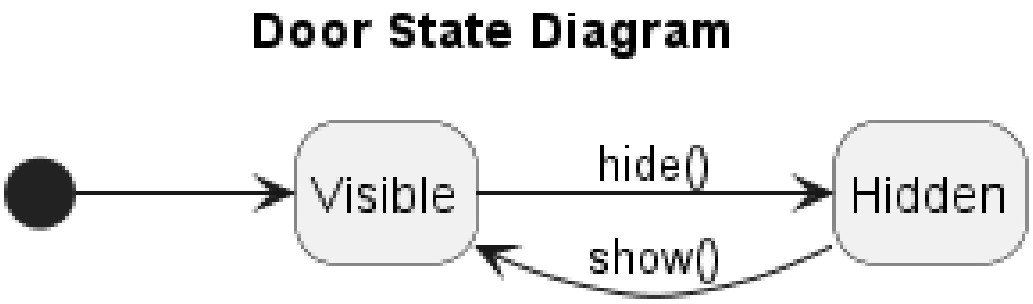
```
if hidden:
    return
if oneWay and (the direction is not correct)
    return
if entering the other room is successful:
    leave the previous room
```

◇ *+move(moveFrom : Room, moveTo : Room) : void* - Ajtó mozgatása: Ha a *moveFrom* paraméter egyezik az ajtó egyik szobájával, akkor ehelyett a szoba helyett a *moveTo* lesz az a szobája. Ha az áthelyezés miatt az ajtó mindkét szobája ugyan az lenne, akkor az ajtó törlődik.

```
if moveTo = from or moveTo = to:
    remove door from moveTo
else:
    remove door from moveFrom
    if moveFrom = from:
        from = moveTo
    else if moveFrom = to
        to = moveFrom
    add this door to moveTo
```

◇ *+hide() : void* - A *hidden* attribútum hamisra állítása.
◇ *+show() : void* - A *hidden* attribútum igazra állítása.
◇ *+setOneWay(oneWay : boolean) : void* - Setter.

■ Állapotgép



8.1. ábra. Ajtó állapota

8.1.8. Effect

- Felelősség

A játék objektumainak hatását megvalósító effectek ősosztálya.

- Interfészek

Updatable

- Ősosztály

-

- Attribútumok
  - ◇ - *timeRemaining : double* - Mennyi ideig érvényes még a hatás.
  - ◇ - *holder : GameObject* - Megadja a hatás birtokosát.
- Metódusok
  - ◇ *+applyToStudent(target : Student) : void* - A hatás alkalmazása a hallgatóra.
  - ◇ *+applyToTeacher(target : Teacher) : void* - A hatás alkalmazása az oktatóra.
  - ◇ *+applyToRoom(room : Room) : void* - A hatás alkalmazása a szobára.
  - ◇ *+interactCleanEffect(cleanEffect : CleanEffect) : void* - A tisztasg hatás.
  - ◇ *+setHolder(holder : GameObject) : void* - Beállítja a hatás birtokosát.

8.1.9. FakeCodeOfStudies

- Felelősség

A TVSZ hamis megfelelője, a felvételekor nem történik semmi, nem használható semmire.

- Interfészek

-

- Ősosztály

Item → CodeOfStudies

- Attribútumok

-

- Metódusok
  - ◇ *+useAgainst(target : Teacher) : void* - Nem történik semmi.

8.1.10. FakeMask

- Felelősség

A Mask hamis megfelelője, a felvételekor nem történik semmi, nem használható semmire.

- Interfészek

-

- Ősosztály

Item → Mask

- Attribútumok

-

- Metódusok
  - ◇ *+usePassive() : void* - Nem történik semmi.

8.1.11. FakeSlideRule

- Felelősség

A Logarléc hamis megfelelője, a felvételekor nem történik semmi.

- Interfészek

-

- Ősosztály

Item → SlideRule

- Attribútumok

-

- Metódusok
  - ◇ *+setPerson(person : Person) : void* - Nem történik semmi.

8.1.12. GasEffect

- Felelősség

A mérgező szobák mérgező levegőjének megvalósítása: emberek elájulását okozza.

- Interfészek

Updatable

- Ősosztály

Effect

- Attribútumok

-

- Metódusok
  - ◇ *+applyToStudent(target : Student) : void* - Kiüti a hallgatót meghatározott időre.
  - ◇ *+applyToTeacher(target : Teacher) : void* - Kiüti az oktatót meghatározott időre.
  - ◇ *+interactCleanEffect(cleanEffect : CleanEffect) : void* - Eltávolítja az effectet a szobáról.

8.1.13. GameObject

- Felelősség

Ősosztály az olyan játékban szereplő objektumoknak, amelyek Effecteket és Itemeket tárolnak.

- Interfészek

Updatable

- Ősosztály

-

- Attribútumok
  - ◇ *#effects [0..*] : Effect* - Effektek

- Metódusok
  - ◇ *+addItem(item : Item) : void* - Tárgy hozzáadása az objektumhoz.
  - ◇ *+removeItem(item : Item) : void* - Tárgy eltávolítása az objektumról.
  - ◇ *+addEffect(effect : Effect) : void* - Hatás hozzáadása az objektumhoz.
  - ◇ *+removeEffect(effect : Effect) : void* - Hatás eltávolítása az objektumról.
  - ◇ *+applyEffect(effect : Effect) : void* - Hatás alkalmazása az objektumra.
  - ◇ *+interactTeacher(teacher : Teacher) : void* - Absztrakt. Az objektum interakciója oktatóval.

8.1.14. Inventory

- Felelősség

Felszerelés. Egy személynél van. Ez tárolja a személy tárgyait.

- Interfészek

-

- Ősosztály

-

- Attribútumok
  - ◇ *-items : Item[0..5]* - A személy tárgyai.
- Metódusok
  - ◇ *+add(item : Item) : boolean* - Az *item* hozzáadása az *items*-hez, ha az kevesebb, mint 5 tárgyat tartalmaz. Ha a művelet sikeres volt igazat ad vissza, különben hamisat.
  - ◇ *+remove(item : Item) : void* - Eltávolítja az *item*-et az *items*-ből, ha az benne volt.
  - ◇ *+setRoom(room : Room) : void* - Az *items* minden elemére meghívja a *setRoom*(*room*) metódust.
  - ◇ *+protectFrom(teacher : Teacher) : void* - Az *items* minden elemére meghívja a *useAgainst*(*teacher*) metódust.
  - ◇ *+dropRandomItem() : void* - Véletlenszerű elem eldobása az *items*-ből.

8.1.15. Item

- Felelősség

Ősosztály biztosítása a játék tárgyai számára.

- Interfészek

-

- Ősosztály

-

- Attribútumok
  - ◇ *#room : Room* - A szoba ahol a tárgy van.
  - ◇ *#person : Person* - Az ember akinél a tárgy van.
- Metódusok
  - ◇ *+use() : void* - A tárgyat használja valaki.
  - ◇ *+useAgainst(target : Teacher) : void* - A tárgyat egy oktató ellen használják.
  - ◇ *+usePassive() : bool* - A tárgy működését a környezet váltja ki.
  - ◇ *+useItem(item : Item) : void* - A tárgyy használata egy másik tárggyal.

◇ *+link(other : Transistor) : void* - Két tranzisztor összekapcsolása
◇ *+drop() : void* - A tárgy eldobódik.
◇ *+setRoom(room : Room) : void* - A tárgynak beállítódik a szoba, ahol van.
◇ *+setPerson(person : Person) : void* - A tárgynak beállítódik az ember, akinél van.

8.1.16. Janitor

■ Felelősség

A mérgező szobák hatását szüntetit meg, a szobákat tisztává teszi és kirakja a Student és Teacher objektumokat.

■ Interfészek

Updatable

■ Ősosztály

GameObject → Person

■ Attribútumok

-

■ Metódusok

◇ *+enterRoom(room : Room) : void* - Belép a szobába és létrehoz egy Janitor-Effectet a szobán.
◇ *+interactTeacher(teacher : Teacher) : void* - Nem csinál semmit
◇ *+protectFromTeacher(teacher : Teacher) : void* - Nem csinál semmit
◇ *+applyEffect(effect : Effect) : void* - Nem csinál semmit
◇ *+pickedUpSlideRule() : void* - Nem csinál semmit

8.1.17. JanitorEffect

■ Felelősség

A szobákat megtisztítja és kirakja a benne tartózkodókat.

■ Interfészek

Updatable

■ Ősosztály

Effect → CleanEffect

■ Attribútumok

-

■ Metódusok

◇ *+applyToStudent(target : Student) : void* - Kirakja a szobából a hallgatót
◇ *+applyToTeacher(target : Teacher) : void* - Kirakja a szobából az oktatót.
◇ *+applyToRoom(target : Room) : void* - Megtisztítja a szobát.

8.1.18. Mask

■ Felelősség

Megvédi a hallgatót az mérgező levegőjű szobkáktól.

■ Interfészek

-

■ Ősosztály

Item

■ Attribútumok

◇ *-uses : int* - Megadja hányszor használták a maszkot.
◇ *-effect : MaskEffect* - A tárgy által kifejtett hatás, ha éppen aktív.

■ Metódusok

◇ *+usePassive() : void* - Akkor hívódik meg, ha birtokosa gázzal teli szobában van és épp nincs rajta MaskEffect, ekkor új MaskEffect-et alkalmaz birtokosára. Ha a *uses* 0-ra csökken, megsemmisül.
◇ *+drop() : void* - Amikor valaki eldobja a maszkot, ha a hatása éppen aktív volt, az megszűnik.

8.1.19. MaskEffect

■ Felelősség

A Mask hatásának megvalósítása: védelem a mérgező levegővel szemben.

■ Interfészek

Updatable

■ Ősosztály

Effect

■ Attribútumok

-

■ Metódusok

◇ *+applyToStudent(target : Student) : void* - Megvédi a hallgatót a mérgező gáztól.

◇ *+applyToTeacher(target : Teacher) : void* - Megvédi az oktatót a mérgező gáztól.

8.1.20. Person

■ Felelősség

A játékban szereplő emberek közös őse.

■ Interfészek

Updatable

■ Ősosztály

GameObject

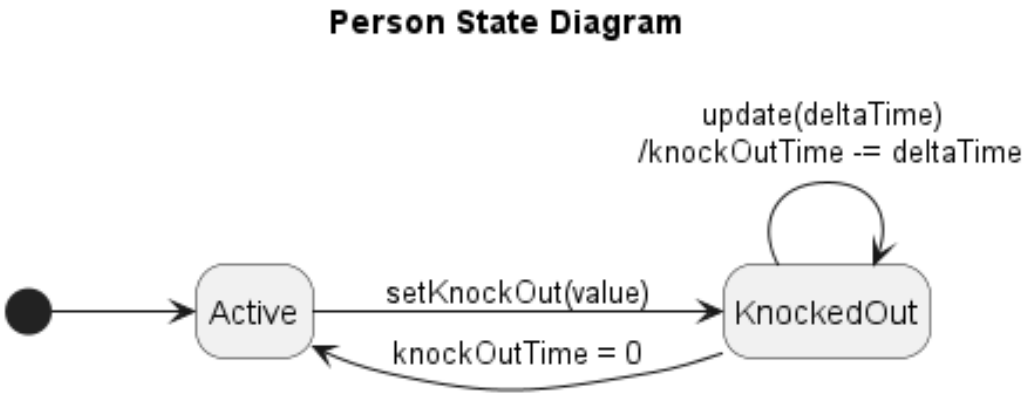
■ Attribútumok

◇ *-name : String* - A személy neve.
◇ *-currentRoom : Room* - A szoba, ahol aktuálisan van az ember.
◇ *-knockOutTime : double* - Mennyi idő van hátra az ájult állapotból. Ha 0, akkor nincs ájult állapot.
◇ *#inventory : Inventory* - A felszerelés. Ez tárolja az ember tárgyait.

■ Metódusok

◇ *+protectFromTeacher(target : Teacher) : void* - Absztrakt. Személy reagál arra, hogy oktatóval találkozik.
◇ *+enterRoom(room : Room) : void* - Frissíti a *currentRoom*-ot és az *inventory*-t is értesíti a szoba váltásról.
◇ *+dropItem(item : Item) : void* - Tárgy eldobása: az *item*-et eltávolítja az *inventory*-ból, és a *currentRoom*-hoz adja.
◇ *+setKnockout(value : double) : void* - Setter.
◇ *+update(deltaTime : double) : void* - Az $effects$ összes elemére: először frissíti, majd alkalmazza magára.
◇ *+addItem(item : Item) : void* - Az *item* hozzáadása az *inventory*-hoz és az *item* person-ja ez az objektum lesz.
◇ *+removeItem(item : Item) : void* - Az *item* eltávolítása az *inventory*-ból.
◇ *+pickedUpSlideRule() : void* - Absztrakt. A személy felvette a logarlécet, és erre reagál.
◇ *+getOut() : void* - A személynek el kell hagynia a szobát.
◇ *+dropRandomItem() : void* - Véletlenszerű tárgy eldobása az *inventory*-ból.

■ Állapotgép



8.2. ábra. Ember állapota

8.1.21. RagEffect

- Felelősség

A WetRag hatásának megvalósítása: oktatók megbénítása.

- Interfészek

Updatable

- Ősosztály

Effect

- Attribútumok

-

- Metódusok
  - ◇ *applyToTeacher(target : Teacher) : void* - Megbénítja az oktatót.

8.1.22. Room

- Felelősség

Szoba. Személyek, tárgyak, effektek találhatók benne. Ezek a Room-on keresztül lépnek kapcsolatba, és a Room értesíti őket az idő múlásáról.

- Interfészek

Updatable

- Ősosztály

GameObject

- Attribútumok

◇ *-people [0..*] : Person* - Személyek
◇ *-items [0..*] : Item* - Tárgyak
◇ *-doors [0..*] : Door* - Ajtók
◇ *-capacity : int* - Kapacitás : ennél több személy nem lehet a szobában
◇ *-visitorsSinceClean : int* - A legutóbbi tisztítás óta mennyien látogatták a szobát. Ha valaki belép a szobába ez eggyel nő.

■ Metódusok

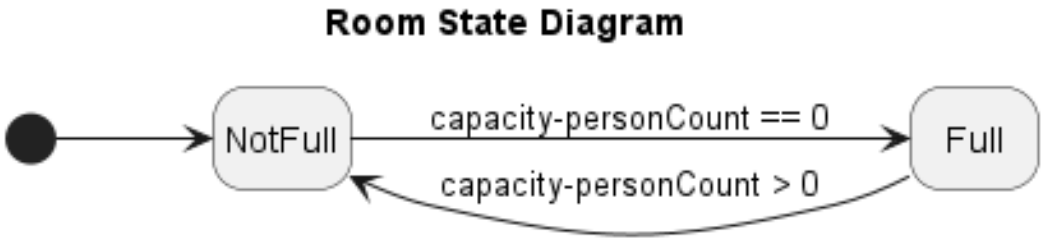
◇ *+Room()* - Konstruktor, ami alapértelmezett kapacitású, üres szobát hoz létre, aminek nincs ajtaja.
◇ *+Room(effects : Effect[0..*], capacity : int)* - Konstruktor, ami megadja a hatásokat és a kapacitást.
◇ *+enter(person : Person) : boolean* - Személy megpróbál belépni a szobába. Ha kevesebben vannak a szobában, mint a *capacity*, akkor a művelet sikeres: a *person* a szobába lép és a visszatérési érték igaz, különben nem történik semmi és a visszatérési érték hamis.
◇ *+leave(person : Person) : void* - A *person* elhagyja a szobát.
◇ *+merge(room : Room) : void* - A *room* beleolvasztása ebbe a szobába. Ha a két szobának nincs közös ajtaja, vagy a nagyobbik kapacitás kevesebb, mint a két szobában található személyek összesen, akkor nem történik semmi. Különben a *room* tartalmát ebbe a szobába helyezi át és kapacitását beállítja a kettő közül a nagyobbra.
◇ *-moveContents(room : Room) : void* - A szoba objektumait (emberek, tárgyak, hatások és azon ajtók, amelyek nem a két szoba köz mennek) a *room*-ba helyezi át.
◇ *+split() : Room* - Létrehoz egy új szobát. Az *items* és *people* tartalmának véletlenszerű részét átadja az új szobának. A *doors* véletlenszerű részét átmozgatja az új szobára. Az $effects$ minden tagjának másolatát átadja az új szobának. Létrehoz egy új ajtót ami ebből a szobából vezet az újba. Végül vissza adja az új szoba objektumot.

```
create newRoom
for item in items:
    get a random integer between 0 and 1
    if random int = 0:
        move the current item to the newRoom
for person in people:
    get a random integer between 0 and 1
    if randomInt = 0:
        move the current person to the newRoom
for door in doors:
    get random integer between 0 and 1
    if random int = 0:
        move the current door to the newRoom
for effect in effects:
    copy the current effect to the newRoom
create a door between room and newRoom
return newRoom
```

◇ *+interactTeacher(teacher : Teacher) : void* - A szobában tartózkodó összes embert értesíti az oktatóval való interakcióról.

◇ *+addDoor(door : Door) : void* - Ajtó hozzáadása.
◇ *+removeDoor(door : Door) : void* - Ajtó eltávolítása.
◇ *+interactCleanEffect(effect : CleanEffect) : void* - Az $effects$ minden elemét értesíti az interakcióról.
◇ *+isClean() : boolean* - Visszaadja, hogy ragadós-e a szoba. Attól függ, a *visitorsSinceClean* egy adott értéknél nagyobb-e.
◇ *+clean() : void* - A *visitorsSinceClean* értékét 0-ra állítja.
◇ *+update(deltaTime : double) : void* - Frissíti az $effects$ összes elemét. A *people* minden elemére először alkalmazza a szoba hatásait, majd frissíti őket.
◇ *+getOut(person : Person) : void* - A személy a doors minden elemét használja, amíg valamelyiken keresztül el nem hagyja a szobát. Ha egyiken kereszül se tud távozni, akkor marad a szobában. A használat sorrendje az ajtók a szobához adásának sorrendje.

■ Állapotgép



8.3. ábra. Szoba állapota

8.1.23. SlideRule

■ Felelősség

A felvételével véget ér a játék.

■ Interfészek

-

■ Ősosztály

Item

■ Attribútumok

-

■ Metódusok

◇ *+setPerson(person : Person) : void* - Értesíti a *person*-t, hogy felvette a logarlécet.

8.1.24. Student

- Felelősség

A játékosok reprezentálása a játékban.

- Interfészek

Updatable

- Ősosztály

GameObject → Person

- Attribútumok

◇ *-eliminated: boolean* - Azt adja meg, hogy a hallgató játékban van-e még.
◇ *-immuneToTeacher : Teacher[0..*]* - Oktatók akik ellen TVSZ segítségével védekezett a hallgató.

- Metódusok

◇ *+setEliminated(value : bool) : void* - Beállítja, hogy a játékos játékban van-e.
◇ *+interactTeacher(teacher : Teacher) : void* - Interakcióba kerül egy oktatóval, megpróbálja megvédeni magát a rá ható hatások és nála lévő tárgyak segítségével.
◇ *+protectFromTeacher(target : Teacher) : void* - Sikeresen megvédte magát az oktatóval szemben TVSZ segítségével.
◇ *+applyEffect(effect : Effect) : void* - Értesíti a hatást, hogy hallgatóra kell alkalmazódnia.
◇ *+pickedUpSlideRule() : void* - A hallgatók nyertek, vége a játéknak.

8.1.25. Teacher

- Felelősség

Oktatók, a hallgatók lelkét igyekeznek elvenni.

- Interfészek

Updatable

- Ősosztály

GameObject → Person

- Attribútumok

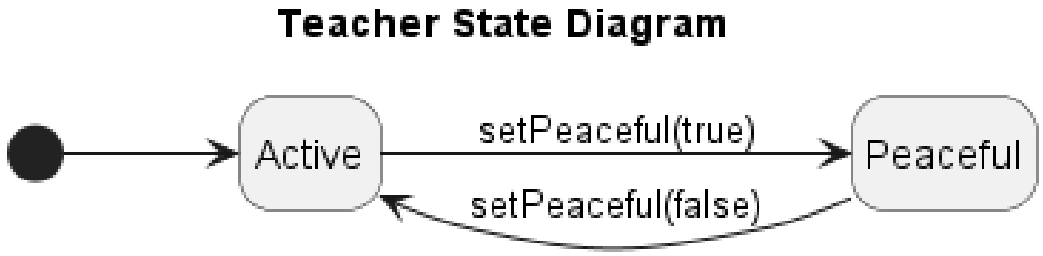
◇ *-peaceful : boolean* - Azt adja meg, hogy az oktató bénult állapotban van-e.

- Metódusok

◇ *+interactTeacher(teacher : Teacher) : void* - Nem csinál semmit
◇ *+protectFromTeacher(teacher : Teacher) : void* - Nem csinál semmit
◇ *+setPeaceful(value : boolean) : void* - Az oktatót bénult állapotba helyezi.

◇ *+applyEffect(effect : Effect) : void* - Értesíti az $effect$-et, hogy oktatóra kell alkalmazódnia.
◇ *+update(deltaTime : double) : void* - Az ősosztály szerint elvégzi a frissítést, majd értesíti a szobát, ahol éppen tartózkodik, hogy oktató van ott.
◇ *+pickedUpSlideRule() : void* - Nem csinál semmit

■ Állapotgép



8.4. ábra. Oktató állapota

8.1.26. Transistor

■ Felelősség

Szobák közötti teleportációra használható.

■ Interfészek

-

■ Ősosztály

Item

■ Attribútumok

◇ - *other : Transistor* - A tranzisztor párja, amivel össze lett kapsolva.
◇ - *target : Room* - A szoba, ahova a tranzosztorokkal el lehet jutni.

■ Metódusok

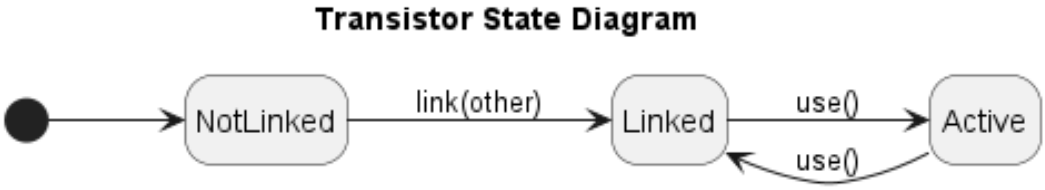
◇ *setTarget(target : Room) : void* - A tranzisztor-teleportáció célszobájának beállítása
◇ *setPair(pair : Transistor) : void* - A tranzisztor párjának beállítása.
◇ *+useItem(item : Item) : void* - Megpróbálja párosítani magát a másik tárggyal.
◇ *+link(other : Transistor) : void* - Párosítja magát a másik tranzisztorral.
◇ *+use() : void* - A tranzisztor birtokosa használja. Ha nincs párosítva, nem történik semmi. Ha párosítva van, de nem aktív, akkor lehelyeződik és aktiválja párját a cél szoba beállításával. Ha aktív, akkor eldobódik és birtokosát a cél szobába teleportálja.

```
if not linked:
    return
```

```
if linked and not active:
    set the current room as target for other
    drop this
if linked and active:
    drop this
    teleport holder to the target room
```

.

- Állapotgép



8.5. ábra. Tranzisztor állapota

8.1.27. Updatable

- Felelősség

Interfész, mely lehetőséget ad az őt megvalósító objektumok frissítésére.

- Interfészek

-

- Ősosztály

-

- Metódusok
  ◇ *+update(deltaTime : double) : void* - Objektum frissítése: Telt az idő, és erre reagál az objektum. deltaTime: eltelt idő a legutóbbi update óta.

8.1.28. WetRag

- Felelősség

Megvédi a hallgatót azzal, hogy a szobában lévő oktatókat megbénítja.

- Interfészek

-

- Ősosztály

Item

- Attribútumok
  - ◇ *-effect : RagEffect* - A tárgy által kifejtett hatás.
- Metódusok
  - ◇ *+setRoom(room :Room) : void* - Amikor a rongyot használó hallgató átmegy egy másik szobába a hatása vele együtt átvándorol.

8.2. A tesztek részletes tervei, leírásuk a teszt nyelvén

8.2.1. AddDoor

- Leírás

A teszt célja megbizonyosodni arról, hogy az ajtó elhelyezés két szoba közt megfelelően működik.

- Ellenőrzött funkcionalitás, várható hibahelyek

Az ajtó működését vizsgáljuk.

- Bemenet

```
Create Room room1
Create Room room2
Create Door door room1 room2
Status door
```

- Elvárt kimenet

```
Door door
Hidden: false
One-way: false
Room 1: room1
Room 2: room2
```

8.2.2. ApplyCleanEffectToRoom

- Leírás

A teszt célja a légfrissítő működésének ellenőrzése.

- Ellenőrzött funkcionalitás, várható hibahelyek

A GasEffect eltávolítása légfrissítő által.

- Bemenet

```
Create Student student
Create AirFreshener freshener
Create Room room
Create GasEffect effect
Add room freshener
Add room effect
Add room student
Pickup student freshener
Use student freshener
Update 1
Status room
```

- Elvárt kimenet

```
Room room
Capacity: 4
Doors:
Effects:
Items:
People: student
```

8.2.3. ApplyBeerEffectToStudent

- Leírás

A teszt célja a BeerEffect Diákra való terjedésének ellenőrzése.

- Ellenőrzött funkcionalitás, várható hibahelyek

A BeerEffect terjedését vizsgáljuk

- Bemenet

```
Create Student student
Create Beer beer
Pickup student beer
Use student Beer
Status student
```

- Elvárt kimenet

```
Student student
Effects: beerEffect
Eliminated: false
Inventory:
Knock-out time: -5
Room:
```

8.2.4. ApplyGasEffectToRoom

- Leírás

A teszt célja a Camembert által hordozott Effect szobára való terjedésének tesztelése.

- Ellenőrzött funkcionalitás, várható hibahelyek

Az GasEffect terjedését vizsgáljuk.

- Bemenet

```
Create Student student
Create Camembert cheese
Create Room room
Add room student
Add room cheese
Pickup student cheese
Use student cheese
Status room
```

- Elvárt kimenet

```
Room room
Capacity: 4
Doors:
Effects: gasEffect
Items:
People: student
```

8.2.5. ApplyMaskEffectToStudent

- Leírás

A teszt célja a MaskEffect Diákra való terjedésének ellenőrzése.

- Ellenőrzött funkcionalitás, várható hibahelyek

A MaskEffect terjedését vizsgáljuk

- Bemenet

```
Create Room room
Create GasEffect effect
Create Student student
Create Mask mask
Add room student
Add room effect
Add room mask
Pickup student mask
Update 1
Status student
Status mask
```

- Elvárt kimenet

```
student got knocked out.
student was protected by mask.

Student student
Effects: maskeffect
Eliminated: false
Inventory: mask
Knock-out time: -5
Room: room

Mask mask
Person: student
Room: room
Uses: 4
```

8.2.6. ApplyRagEffectToRoom

- Leírás

A teszt célja a Táblatörlő által hordozott Effect szobára való terjedésének tesztelése.

- Ellenőrzött funkcionalitás, várható hibahelyek

Az tárgy Effect terjedése.

- Bemenet

```
Create WetRag rag
Create Room room 4
Add rag room
Status room
```

- Elvárt kimenet

```
Room room
Capacity: 4
Effects: wetRagEffect
Items:
People: student
```

8.2.7. AttackStudent

- Leírás

Az oktató támadásának tesztelése

- Ellenőrzött funkcionalitás, várható hibahelyek

A tanár támadása

- Bemenet

```
Create Room room
Create Teacher teacher
Create Student student
Add room teacher
Add room student
Update 1
Status student
```

- Elvárt kimenet

```
teacher attacked everyone.
student was eliminated.

Student: student
Effects:
Eliminated: true
Inventory:
Knock-out time: -5
Room: room
```

8.2.8. BeerProtectStudent

- Leírás

A Söröspohár védelme a tanárral szemben, közben a hallgató elejt egy tárgyat.

- Ellenőrzött funkcionalitás, várható hibahelyek

A tanár támadása

- Bemenet

```
Create Room room
Create Teacher teacher
Create Student student
Create Mask mask
Create Beer beer
Add room teacher
Add room beer
Add room mask
Add room student
Pickup student beer
Pickup student mask
Update 1
Status student
```

- Elvárt kimenet

```
teacher attacked everyone.
student was eliminated.
student was revived.

Student: student
Effects:
Eliminated: false
Inventory:
Knock-out time: -5
Room: room
```

8.2.9. CleanCheese

- Leírás

A teszt célja tesztelni a Camembert által hordozott effect kitisztítását a szobából egy Takarító belépését követően.

- Ellenőrzött funkcionalitás, várható hibahelyek

A Takarító tesztelése, a sajt tesztelése, az effect terjedés tesztelése

- Bemenet

```
Create Room roomA
Create Room roomB
Create Door door roomA roomB
Create Janitor janitor
Create Student student
Create Camembert cheese
Add room cheese
Pickup student cheese
Add roomA student
Add roomB janitor
Use student cheese
Move janitor door
Status roomA
Status roomB
```

- Elvárt kimenet

```
Room roomA
Capacity: 4
Effects:
Items:
People: janitor

Room roomB
Capacity: 4
Effects:
Items:
People: student
```

8.2.10. DenyStudentMovementDueToCapacity

- Leírás

A teszt célja megbizonyosodni arról, hogy hogy az ajtók rossz irányú mozgást nem engednek át.

- Ellenőrzött funkcionalitás, várható hibahelyek

Az ajtók működése

- Bemenet

```
Create Room roomA
Create Room roomB 0
Create Student student
Create door roomA roomB
Add roomA student
Move person door
```

- Elvárt kimenet

```
The other room is full.
```

8.2.11. DenyStudentMovementDueToDirection

- Leírás

A teszt célja megbizonyosodni arról, hogy hogy az ajtók rossz irányú mozgást nem engednek át.

- Ellenőrzött funkcionalitás, várható hibahelyek

Az ajtók működése

- Bemenet

```
Create Room roomA
Create Room roomB
Create Student student
Create door roomA roomB
Oneway door true
Add roomB student
Move student door
Status roomA
```

- Elvárt kimenet

```
student couldn't use the door.
Room roomA
Capacity: 4
Effects:
Items:
People:
```

8.2.12. DropItem

- Leírás

A teszt célja az eldobás funkció ellenőrzése

- Ellenőrzött funkcionalitás, várható hibahelyek

A tárgyeldobás funkció ellenőrzése.

- Bemenet

```
Create Student student
Create Room room
Create Beer beer
Add room student
Add room beer
Pickup student beer
Drop student beer
Status student
Status room
```

- Elvárt kimenet

```
Student student
Effects:
Eliminated: false
Inventory:
Knock-out time: -5
Room: room

Room room
Capacity: 4
Effects:
Items: beer
People: student
```

8.2.13. InheritGasEffect

- Leírás

A teszt célja megbizonyosodni arról, hogy a szoba által számontartott Effectet átveszik-e a benne lévő szereplők

- Ellenőrzött funkcionalitás, várható hibahelyek

Az Effcet terjedése.

- Bemenet

```
Create Room room
Create Student student
Create Camembert cheese
Add room student
Add student cheese
Use student cheese
Update 1
Status student
```

- Elvárt kimenet

```
student got knocked out.

Student student
Effects:
Eliminated: false
Inventory:
Knock-out time: 5
Room: room
```

8.2.14. KnockOutWithMask

- Leírás

A teszt célja a Camembert működésének ellenőrzése.

- Ellenőrzött funkcionalitás, várható hibahelyek

A Camembert működésének ellenőrzése.

- Bemenet

```
Create Room room
Create Student student
Create Mask mask
Create GasEffect effect
Add room effect
Add room mask
Add room student
Pickup student mask
Update 1 room
```

```
Status student
```

- Elvárt kimenet

```
student got knocked out.
student was protected by mask.

Student: student
Effects:
Eliminated: false
Inventory:
Knock-out time: -5
Room: room
```

8.2.15. KnockOutWithoutMask

- Leírás

A teszt célja a Camembert működésének ellenőrzése.

- Ellenőrzött funkcionalitás, várható hibahelyek

A Camembret működésének ellenőrzése.

- Bemenet

```
Create Room room
Create Student student
Create GasEffect effect
Add room effect
Add room student
Update 1 room
Status student
```

- Elvárt kimenet

```
student got knocked out.

Student: student
Effects:
Eliminated: false
Inventory:
Knock-out time: 5
Room: room
```

8.2.16. MergeRooms

- Leírás

A teszt célja a szobaösszeolvadás ellenőrzése

- Ellenőrzött funkcionalitás, várható hibahelyek

A szobaösszeolvadás ellenőrzése

- Bemenet

```
Create Room roomA 2
Create Room roomB 4
Create Student student
Add roomB student
Merge roomA roomB
Status roomA
```

- Elvárt kimenet

```
Room roomA
Capacity: 4
Effects:
Items:
People: student
```

8.2.17. MoveStudentFromRoomToRoom

- Leírás

A teszt célja megbizonyosodni arról, hogy a szereplők képesek használni az szobák közti ajtókat.

- Ellenőrzött funkcionalitás, várható hibahelyek

Az ajtók működése

- Bemenet

```
Create Room roomA
Create Room roomB
Create Student student
Create door roomA roomB
```

```
Add roomA student
Move student door
Status roomB
Status roomA
```

- Elvárt kimenet

```
Room roomB
Capacity: 4
Effects:
Items:
People: student
Room roomA
Capacity: 4
Effects:
Items:
People:
```

8.2.18. PickUpItemWithInventorySpace

- Leírás

A teszt célja a tárgyfelvétel funkció ellenőrzése.

- Ellenőrzött funkcionalitás, várható hibahelyek

A tárgyfelvétel funkció ellenőrzése.

- Bemenet

```
Create Student student
Create Room room
Create Beer beer
Add room beer
Add room student
Pickup student beer
Status student
```

- Elvárt kimenet

```
Student student
Effects:
Eliminated: false
Inventory: beer
```

```
Knock-out time: -5
Room: room
```

8.2.19. PickUpItemWithoutInventorySpace

- Leírás

A teszt célja a tárgyfelvétel funkció ellenőrzése.

- Ellenőrzött funkcionalitás, várható hibahelyek

A tárgyfelvétel funkció ellenőrzése.

- Bemenet

```
Create Student student
Create Room room
Create Beer beer1
Create Beer beer2
Create Beer beer3
Create Beer beer4
Create Beer beer5
Create Beer beer6
Add room student
Add room beer1
Add room beer2
Add room beer3
Add room beer4
Add room beer5
Pickup student beer1
Pickup student beer2
Pickup student beer3
Pickup student beer4
Pickup student beer5
Add room beer6
Pickup student beer6
```

- Elvárt kimenet

```
Inventory is full.
```

8.2.20. PlaceStudent

- Leírás

A teszt célja megbizonyosodni arról, hogy az elvárt belső állapotválltozások mennek végre, amikor egy Person típusú entitást egy szobába rakunk.

- Ellenőrzött funkcionalitás, várható hibahelyek

Az elhelezést vizsgáljuk.

- Bemenet

```
Create Room room
Create Student student
Add room student
Status room
```

- Elvárt kimenet

```
Room room
Capacity: 4
Effects:
Items:
People: student
```

8.2.21. RoomUpdateEffect

- Leírás

A teszt célja az update esemény terjedésének ellenörzése Effectekre.

- Ellenőrzött funkcionalitás, várható hibahelyek

Az effect frissülésének terjedésének ellenőrzése.

- Bemenet

```
Create Room room
Create BeerEffect effect
Add room effect
Update 1 room
Status effect
```

- Elvárt kimenet

```
BeerEffect effect
Holder: student
Time remaining: 4
```

8.2.22. SplitRooms

- Leírás

A teszt célja a szobaszétválás elleőrzése

- Ellenőrzött funkcionalitás, várható hibahelyek

A szobaszétválás elleőrzése

- Bemenet

```
Create Room room
Split room
Status room_S1
```

- Elvárt kimenet

```
Room room_s1
Capacity: 4
Effects:
Items:
People:
```

8.2.23. TeleportUsingTransistors

- Leírás

A teszt célja a teleportálás funkció ellenőrzése

- Ellenőrzött funkcionalitás, várható hibahelyek

A teleportálás funkció ellenőrzése.

- Bemenet

```
Create Room roomA
Create Room roomB
Create Door door roomA roomB
Create Transistor transistorA
```

```
Create Transistor transistorB
Create Student student
Add roomB student
Add roomB transistorA
Add roomB transistorB
Pickup student transistorA
Pickup student transistorB
Link transistorA transistorB
Use student transistorA
Move student door
Use student transistorB
Status roomB
```

- Elvárt kimenet

```
Room roomB
Capacity: 4
Effects:
Items:
People: student
```

8.2.24. UpdateBeerEffect

- Leírás

A teszt célja az effect frissülésének ellenőrzése.

- Ellenőrzött funkcionalitás, várható hibahelyek

Az effect frissülésének ellenőrzése.

- Bemenet

```
Create Student student
Create BeerEffect effect
Add student effect
Update 1 student
Status effect
```

- Elvárt kimenet

```
BeerEffect effect
Holder: student
Time remaining: 4
```

8.2.25. UpdateGasEffect

- Leírás

A teszt célja az effect frissülésének ellenőrzése.

- Ellenőrzött funkcionalitás, várható hibahelyek

Az effect frissülésének ellenőrzése.

- Bemenet

```
Create Room room
Create GasEffect effect
Add room effect
Update 1 room
Status effect
```

- Elvárt kimenet

```
GasEffect effect
Holder: room
Time remaining: 4
```

8.2.26. UpdateMaskEffect

- Leírás

A teszt célja az effect frissülésének ellenőrzése.

- Ellenőrzött funkcionalitás, várható hibahelyek

Az effect frissülésének ellenőrzése.

- Bemenet

```
Create Student student
Create MaskEffect effect
Add student effect
Update 1 student
Status effect
```

- Elvárt kimenet

```
MaskEffect effect
Holder: student
Time remaining: 14
```

8.2.27. UpdateRagEffect

- Leírás

A teszt célja az effect frissülésének ellenőrzése.

- Ellenőrzött funkcionalitás, várható hibahelyek

Az effect frissülésének ellenőrzése.

- Bemenet

```
Create Student student
Create RagEffect effect
Add student effect
Update 1 student
Status effect
```

- Elvárt kimenet

```
RagEffect effect
Holder: student
Time remaining: 4
```

8.3. A tesztelést támogató programok tervei

A tesztadatok struktúrája:

```
tests/
├── test1/
│   ├── info.txt (optional)
│   ├── in.txt
│   └── out.txt
│
└── test2/
    ├── info.txt (optional)
    ├── in.txt
    └── out.txt
```

Az egyes tesztesetek mappákban helyezkednek el. Minden tesztesethez tartozik egy `in.txt`, ez tartalmazza a teszt megvalósításához szükséges bemenetet, illetve egy `out.txt`, ami az elvárt kimenetet tartalmazza. Az `info.txt`-t nem kötelező megadni, ez tartalmazhatja a teszt programban megjelenő nevét (amennyiben ez eltér a mappa nevétől) illetve a teszt rövid leírását.

A teszt futtató program viselkedése parancssori argumentumok segítségével adható meg:

- `testrunner run <test-case>` - Egy bizonyos tesztesetet futtat és megadja, hogy sikeres volt-e vagy sem. `test-case` paraméternek az adott teszteset mappáját kell megadni.
- `testrunner runall <tests>` - Az összes tesztet lefuttatja, mindegyik tesztesetről megadja, hogy sikeres volt-e vagy sem, illetve a végén egy statisztikát ad, hogy hány teszt volt sikeres/sikertelen. `tests` paraméternek a teszteseteket tartalmazó mappát kell megadni.
- `testrunner interactive <tests>` - Interaktív mód, megjeleníti az elérhető teszteseteket, ezután a felhasználó megadhatja, hogy melyik tesztesetet szeretné futtatni. `tests` paraméternek a teszteseteket tartalmazó mappát kell megadni.
- `testrunner command` - Parancs mód, manuálisan lehet parancsokat begépelni, illetve a program megadja az adott parancshoz tartozó kimenetet.

8.4. Napló

Kezdet	Időtartam	Résztvevők	Leírás
ápr. 10. 10:00	1 óra	Balla Joób Nagy Tóth Zelei	Labor
ápr. 10. 11:00	4 óra	Balla Joób Nagy Tóth Zelei	Közös feladatmegoldás
ápr. 10. 12:00	3 óra	Zelei	Prototípus nyelv javítása.
ápr. 12. 12:00	2 óra	Balla	Osztályleírások.
ápr. 13. 8:00	2 óra	Nagy	Osztályleírások készítése.
ápr. 13. 13:00	2 óra	Zelei	Prototípus nyelv bővítése.
ápr. 13. 16:00	4.5 óra	Joób	Tesztek implementációinak leírása, teszttervek javítása
ápr. 14. 8:00	2 óra	Nagy	Osztályleírások készítése, review.
ápr. 14. 12:00	0.5 óra	Tóth	Tesztelést támogató program tervei.
ápr. 14. 21:00	3 óra	Tóth	Osztály leírások ellenőrzése, javítása.
ápr. 15. 2:00	1 óra	Tóth	Tesztesetek átnézése.
ápr. 15. 8:00	1.5 óra	Nagy	Tesztek javítása.
ápr. 15. 9:00	1.5 óra	Zelei	Tesztek javítása.
ápr. 15. 10:00	1 óra	Tóth	Dokumentum átnézése, javítások.