8. Részletes tervek

8.1. Osztályok és metódusok tervei.

1. Osztály: Entity

- **Felelősség**

Az Entity egy alap interfész a játék objektumainak számára. Az idő múlását modellezi időegységenként.

- **Ősosztályok**

Nincsenek.

- **Interfészek**

Nincsenek (ez egy legfelső szintű interfész).

- **Attribútumok**

Nincsenek (az interfészeknek nincsenek attribútumai).

- **Metódusok**
  - **+void tick():** Egy időegység leteltét jelzi az adot objektum számára.

2. Osztály: Person

- **Felelősség**

A Person egy absztrakt osztály, amely a játékban található karaktereket valósítja meg.

- **Ősosztályok**

Entity (közvetve, mivel ez egy interfész)

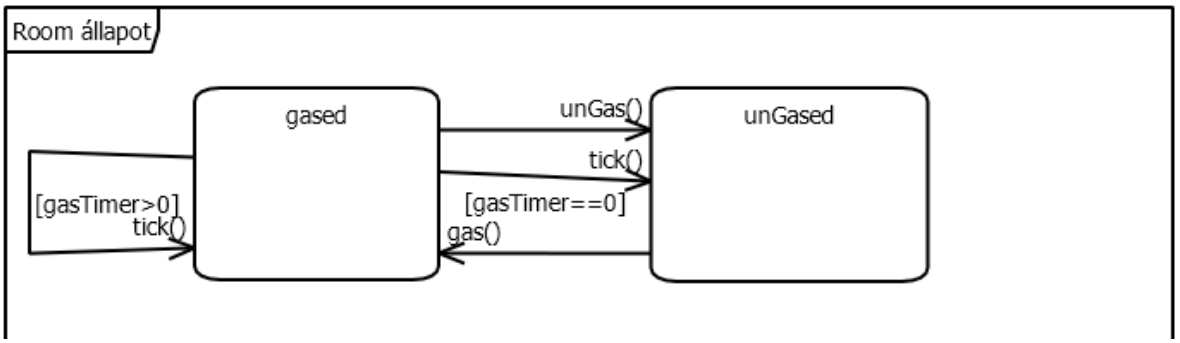
- **Interfészek**

Megvalósítja az Entity interfészt

- **Attribútumok**
  - **#String name:** A személy neve.
  - **#boolean stunned:** Jelzi, hogy a személy éppen kábult-e.
  - **#Room room:** A szoba ahol a személy jelenleg tartózkodik.
  - **#List<Item> items:** A személynél levő tárgyak listája.
  - **+stunTimer:** Visszaszámol, hogy meddig van az adott személy lebénulva. 0-nál ébred fel.
  - **#transistorToPair:** párosításra váró tranzisztor
- **Metódusok**
  - **+void setRoom(Room):** Beállítja a kapott szobát a játékos szobájának.
  - **+void clothStun():** Ezzel a hívással jelezzük a személynek, ha a személyre egy rongy próbál hatni.
  - **+void drop(Item)**: Eldobja a tárgyat a szobában, ahol van. A tárgy a szobába kerül.
  - **+void dropAll():** A személy eldobja az összes általa birtokolt tárgyat abban a szobában ahol épp tartózkodik. A tárgyak a szobába kerülnek.
  - **+void dropRandomItem():** A személy eldob egy véletlenszerűen választott tárgyat. A tárgy a szobába kerül.
  - **+void gasProtection(Item protectionProvider,int priority):** Ezzel a hívással jelez tulajdonosának a védelmet nyújtó tárgy a gasThreat() hatására ha védelmet kínál gáz ellen, ezáltal tulajdonképpen védelmi ajánlatot tesz.
  - **+void gasStun():** Ezzel a hívással jelezzük ha a személyre gáz próbál hatni.
  - **+void move(Door):** A személy átlép a megadott ajtón ha ezt megteheti.
  - **+abstarct void pickup(Item)**: Felveszi a tárgyat a szobából, ahol van.
  - **+void addItem(Item):** A kapott tárgy bekerül az adott személy tárgylistájába.
  - **+void removeItem(Item):**A kapott tárgy kikerül az adott személy tárgylistájából.

- **+void slideRuleNotifcation(Item slideRule)**: Ezt a metódust a SlideRule hívja ha a Person felvette őt. Ez implementációtól függően kezeli a SlideRule megszerzés eseményét.
- **+abstract void teacherAttack():** Ezzel a hívással jelezzük, ha a személyt egy oktató megtámadja.
- **+void teacherProtection(Item protectionProvider, int priority):** Ezzel a hívással jelez tulajdonosának a védelmet nyújtó tárgy a teacherThreat hatására.
- **void teleport(Room roomTo):** A személy átkerül a paraméterben kapott szobába.
- **+void tick():** Egy időegység elmúlását jelzi a személynek. Ha stunnolva van, akkor a stunTimer eggyel csökken.
- **+Room getRoom():**Visszaadja a szobát ahol a személy jelenleg tartózkodik.
- **+String toString():** Visszaadja a személy nevét.

3. Osztály: Room



- **Felelősség**

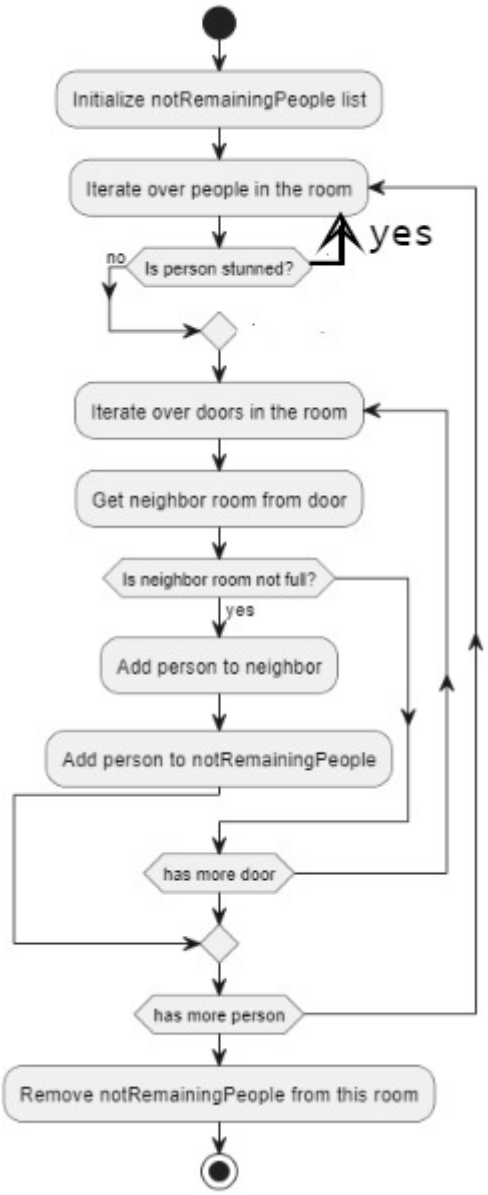
Egy szobát reprezentál, amely tartalmazhat személyeket és tárgyakat, és alapvető környezeti egységként működik az alkalmazásban.

- **Ősosztályok**
- **Interfészek**

Megvalósítja az Entity interfészt.

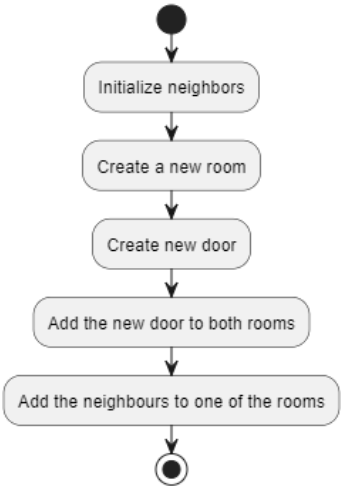
- **Attribútumok**
  - **~List<Person> people:**A szobában tartózkodó személyek listája.
  - **~List<Item> items:** A szobában lévő tárgyak listája.
  - **~List<Door> doors:** A szobában található ajtók listája.
  - **~int id:** A szoba azonosítója.
  - **~int capacity:** Hány ember fér az adott szobába.
  - **~int cleanliness:** A szoba tisztasági szintje.
  - **~boolean gassed:** El vane gázosítva a szoba.
  - **~boolean cursed:** El vane átkozva a szoba.
  - **~boolean isFull:** Tele vane a szoba.
- **Metódusok**
  - **+Room(int id, int capacity, boolean cursed):** A szoba konstruktora.
  - **+void addDoor(Door door):** Hozzáadja a paraméterként kapott ajtót a szoba ajtóinak listájához.
  - **+void addPerson(Person person):** Hozzáadja a paraméterként kapott személyt a szoba személyeinek listájához, abban az esetben, ha van a szobában hely. Ekkor a szoba tisztaságát eggyel csökkenti. Továbbá kezeli a szoba isFull attribútumát.
  - **+void addItem(Item item):** Hozzáadja a paraméterként kapott tárgyat a szoba tárgyainak listájához.

- **+void removePerson(Person person):**Eltávolítja a paraméterként kapott személyt a szoba személyeinek listájából.
- **+void removeItem(Item item):** Eltávolítja a paraméterként kapott tárgyat a szoba tárgyainak listájából.
- **+void removeDoor(Door door):** Eltávolítja a paraméterként kapott tárgyat a szoba tárgyainak listájából.
- **+list<Person> getPeople():** Visszaadja szobában levő személyek listáját.
- **+list<Door> getDoors():**Visszaadja szobában levő szobák listáját.
- **+list<Item> getItems():**Visszaadja szobában levő tárgyak listáját.
- **+void destroy():** Megsemmisíti a szobát.
- **+void gas():** Elgázosítja a szobát.
- **+void unGas():** Megszünteti a szoba gázosságát.
- **+void evacuate():**Kiüríti az embereket, minden mozogni képes Person átkerül egy szomszédos szobába.

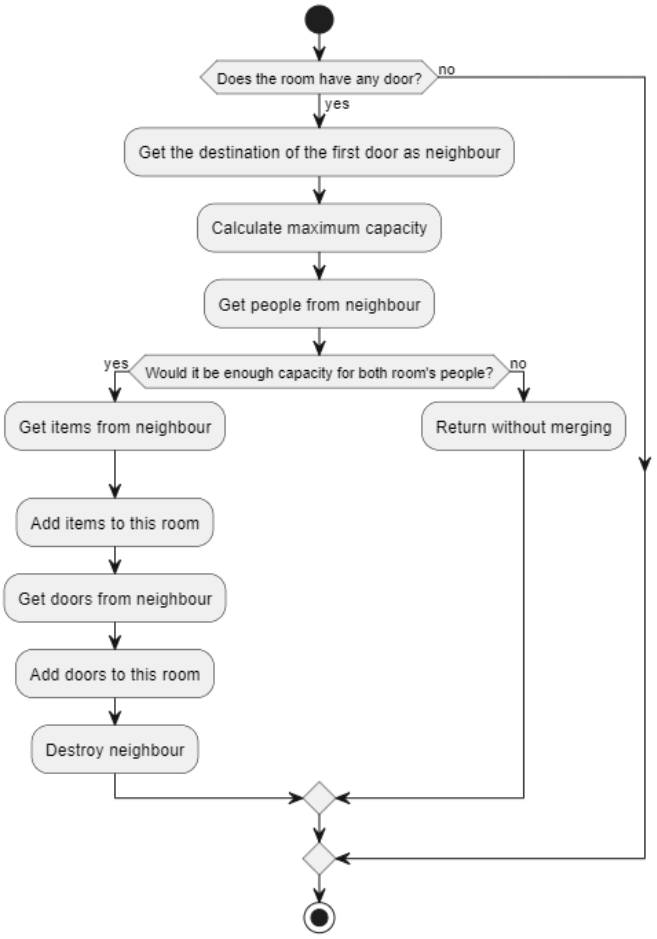


- **+void clean():** A szoba tisztaságát(cleanliness) visszaállítja 10-re.

- **+void divide():** A szoba kettéosztódik. Az osztódás után a két újonnan keletkezett szoba ugyanakkora kapacitással rendelkezik, mint az eredeti szoba. Az osztódás után létrehozott két "új" szoba között egy kétirányú ajtó jön létre.

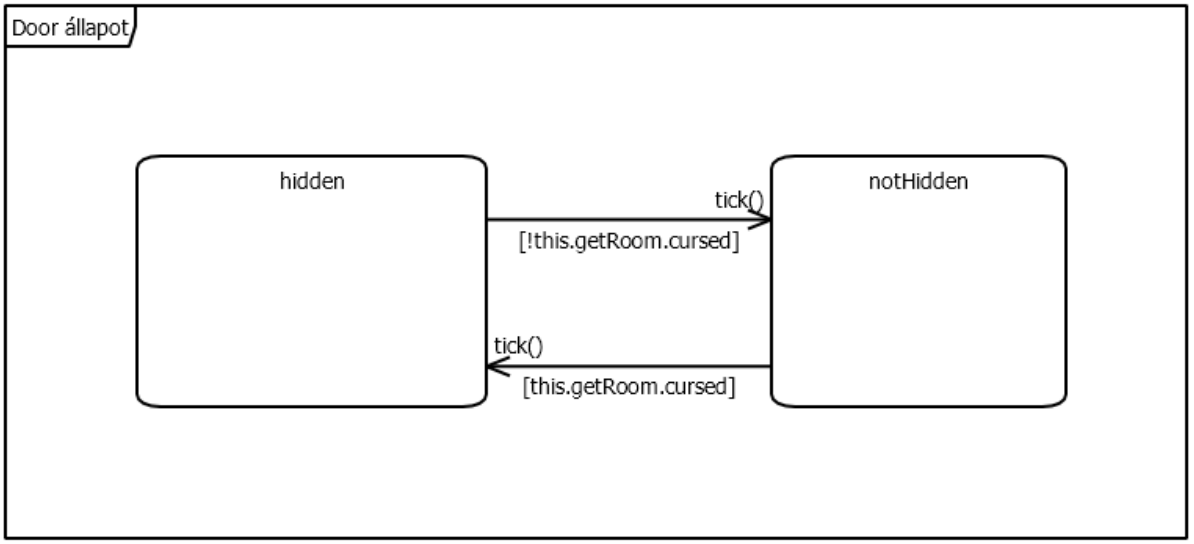


- **+void merge():**A szoba egy véletlenszerű szomszédos szobával egyesül. A függvény llenőrzi, hogy van-e elegendő hely az összes ember elhelyezésére, ha nincs, akkor egyesülés nélkül tér vissza.



- **+boolean isFull():** A szobában tartózkodó személyek és a szoba kapacitása alapján eldönti, hogy a szoba tele vane.
- **+void tick():** Ha a szoba gázos, minden időegységre meghívja a benne levő személyek gasStun() függvényét.

4. Osztály: Door



- **Felelősség**

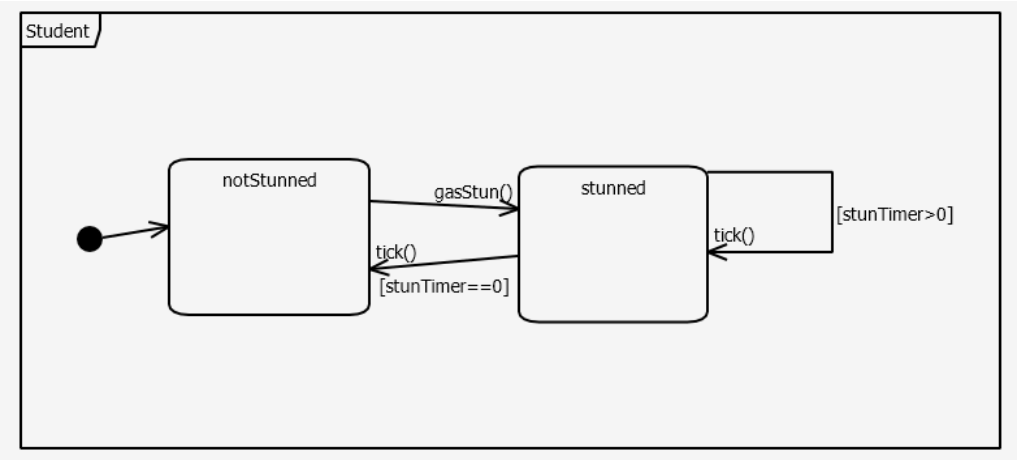
A Door osztály az ajtókat írja le, amelyek két szoba közötti átjárhatóságot biztosítanak, csak egy irányba átjárhatóak.

- **Ősosztályok**
- **Interfészek**

Megvalósítja az Entity interfészt.

- **Attribútumok**
  - **~Room source:** Szomszédos szoba, innen nyílik.
  - **~Room destination:** Szomszédos szoba, ide vezet.
  - **~boolean hidden:** Látható-e az adott ajtó.
- **Metódusok**
  - **+Door():** Ajtó paraméter nélküli konstruktora.
  - **+Door(Room src, Room dst):** Ajtó konstruktora cél- és indulószobával.
  - **+void destroy():** Megsemmisíti a szobát.
  - **+ Room getSrc():** Visszaadja az induló szobát.
  - **+void setSrc(Room room):** Beállítja az induló szobát.
  - **+Room getDest():** Visszaadja a célszobát.
  - **+void setDest(Room room):** Beállítja a célszobát.
  - **+void tick():** Beállítja az ajtajai láthatóságát.

5. Osztály: Student



- **Felelősség**

A Student osztály egy specifikus típusú Person, a hallgatókat megvalósító osztály.

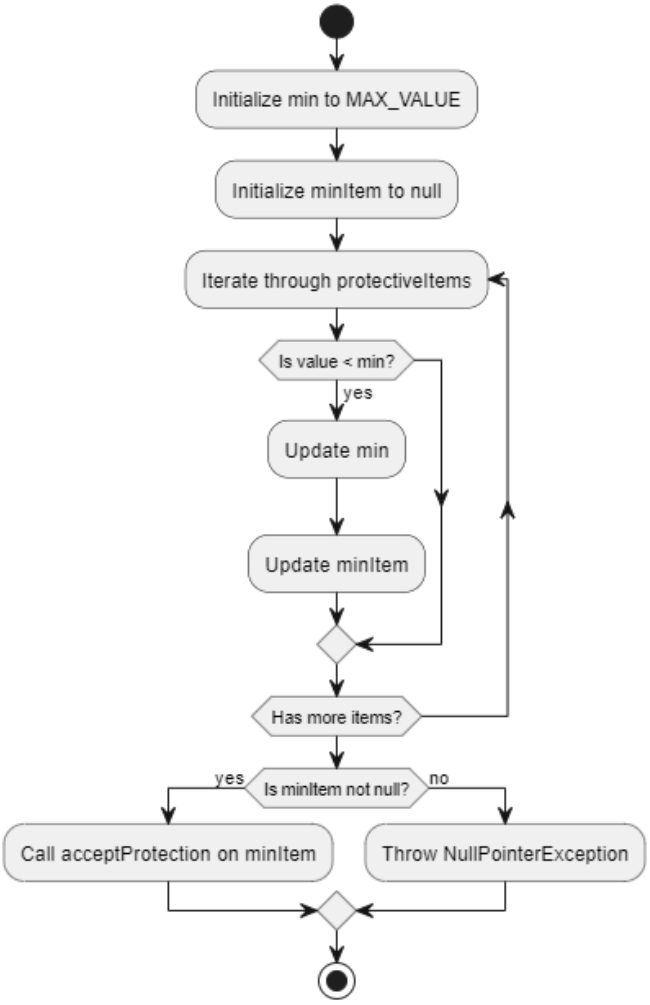
- **Ősosztályok**

Person → Student

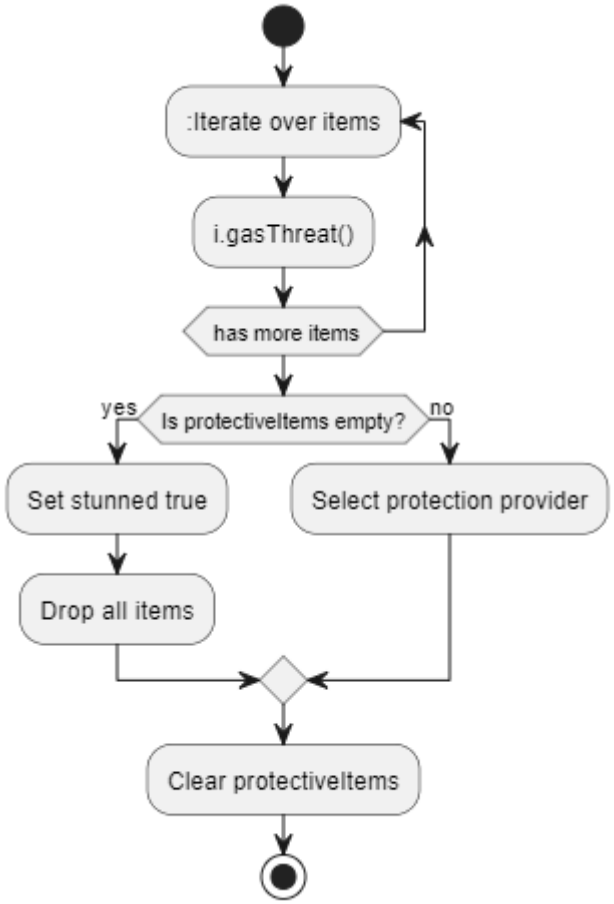
- **Interfészek**

Megvalósítja az Entity interfészt.

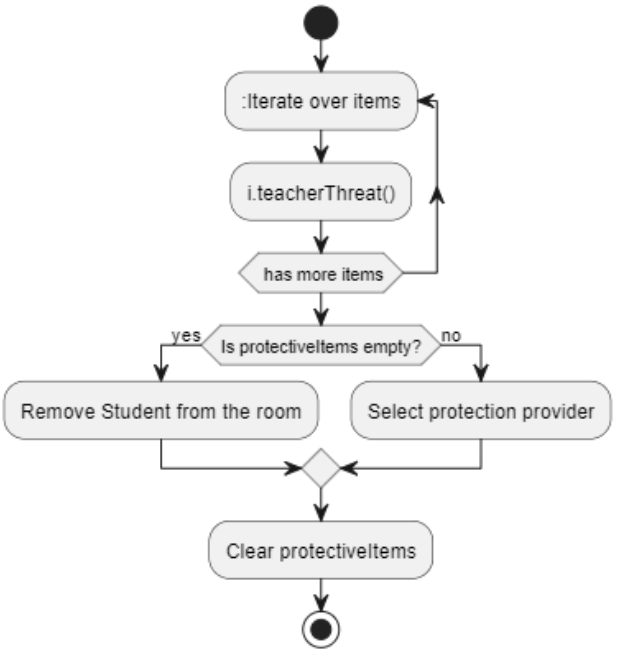
- **Attribútumok**
  - **-HashMap<Item, Integer> protectiveItems:** A védelmi tárgyakat tároló adatszerkezet, azaz azok a tárgyak tárolódnak benne, amelyek védelmet ajánlottak a Studentnek egy bizonyos fenyegetettség ellen.
- **Metódusok**
  - **+Student():** A Student konstruktora.
  - **+void pickup(Item item):**Hozzáadja a paraméterként kapott tárgyat a tárgyainak a listájához, ha még nincsen tele az inventoryja.
  - **-void selectProtectionProvider():** Kiválasztja a legkisebb prioritásút a protectionProviderek listájából.



  - **+void initActivate(Item item):**Mehívja a kapott tárgy activate() függvényét.
  - **+void clothStun():** Semmi nem történik, a Student-re nem hat a rongy.
  - **+void gasStun():** Eseménykezelő a gáz okozta bénításhoz. Végigmegy a Studentnél levő tárgyakon és szól nekik a gáz fenyegetettségről(gasThreat). Ha kapott védelmi ajánlatot, kiválasztja a legalacsonyabb prioritásút (min ellenőrzés). Ha nincs védelem, akkor a karakter megszenvedi a gáz támadást.



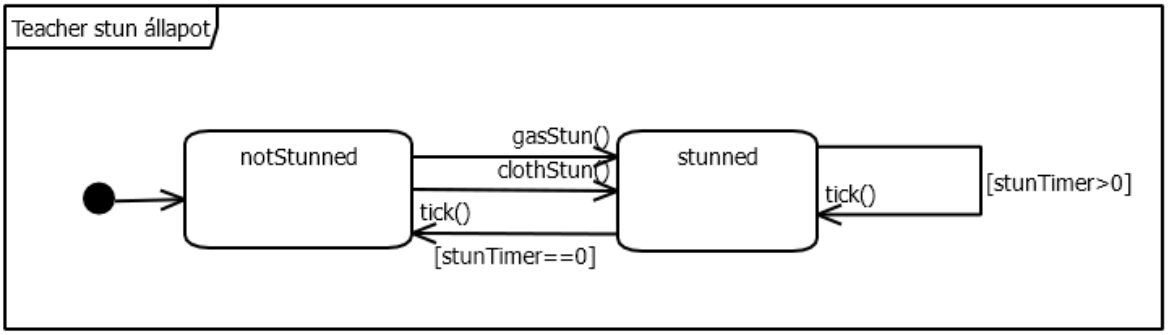
- **+void teacherAttack():**Végigmegy a Studentnél levő tárgyakon és szól nekik a tanár fenyegetettségről(teacherThreat). Ha kapott védelmi ajánlatot, kiválasztja a legalacsonyabb prioritásút (min ellenőrzés). Ha nincs védelem, akkor a karakter megszenvedi a tanár támadást.



- **+void gasProtection():** Egy tárgy védelmet ajánl a gázfenyegetettség ellen. Bekerül a Studen protectiveItems listájába.

- **+void teacherProtection():**Egy tárgy védelmet ajánl a tanár fenyegetettség ellen. Bekerül a Studen protectiveItems listájába.
- **+void slideRuleNotification():** A játék vége, a játékos nyer.

6. Osztály: Teacher



- **Felelősség**

A Teacher osztály egy specifikus típusú Person, egy NPC, ami képes szobákat váltani, tárgyakat felvenni és megsemmisíteni, valamint megtámadja a vele egy szobába levő hallgatókat.

- **Ősosztályok**

Person → Teacher

- **Interfészek**

Megvalósítja az Entity interfészt.

- **Attribútumok**
- **Metódusok**
  - **+void clothStun():** A Teacher stunned állapotát változtatja igazra.
  - **+void gasStun():**A Teacher stunned állapotát változtatja igazra.
  - **+void initAttack():** Meghívja minden szobában levő személy teacherAttack függvényét.
  - **+void pickup(Item item):** Felveszi és megsemmisíti a kapott tárgyat.
  - **+void teacherAttack():** Nem történik semmi, egy oktató csak hallgatót képes megölni.
  - **+void slideRuleNotification():** Az oktató elkapja az általa felvett slideRule-t.

7. Osztály: Cleaner

- **Felelősség**

Ha egy Cleaner olyan szobába lép amiben személyek(Person) vannak, akiket “kitessékel”, azaz szomszédos szobákba fog kerülni minden mozogni képes (nem bénult / ájult / van hova mozognia) személy rajta kívül. Ha gázos szobába lép, kiszellőztet, megszüntetve a szoba gázosságát. A szobák a takarítást követően adott számú látogató után ragacsossá válnak: a bennük lévő és bennük letett tárgyakat nem lehet felvenni.

- **Ősosztályok**

Person → Cleaner

- **Interfészek**

Megvalósítja az Entity interfészt.

- **Attribútumok**
- **Metódusok**
  - **+void initClean():** Meghívja a szobának amiben tartózkodik, a clean(), az evacuate() és az unGas() függvényét.

- **+void gasStun():** Nem történik semmi,(a takarító ki tud szellőztetni gáz esetén).
- **+void clothStun():** Nem történik semmi, (a rongy csak az oktatókra hat).
- **+void teacherAttack():** Nem történik semmi, (az oktató csak a hallgatókat képes megölni).

8. Osztály: Item

- **Felelősség**

A tárgyak absztrakt ősosztálya. Nyilvántartja a tárgy nevét, élettartamát, aktiválja a tárgyat és eltárolja, hogy aktív-e. Elpusztítja magát, ha szükséges. Ha kérnek tőle protection-t és van neki megfelelő, akkor szól a Studentnek.

- **Ősosztályok**

- **Interfészek**
  - Entity
- **Attribútumok**
  - **#Person owner:** Kinél van az adott tárgy. Ha szobában van, null.
  - **#Room room:** Melyik szobában van az adott tárgy. Ha személynél van, null.
  - **#int life:** Hátralévő élettartam, ennyiszer használható még a tárgy.
  - **#bool active:** Jelzi, hogy aktív-e az item.
- **Metódusok**
  - **+void acceptProtection()**: A Person ezzel a hívással jelez ha elfogadja a tárgy védelmi ajánlatát. Aktiválja a tárgyat.
  - **+void activate()**: Az active attribútumot igazra állítja
  - +**void setRoom(Room room):** Beállítja a room attribútumot.
  - +**void destroy():** A tárgy megsemmisítése. Attól függően, hogy szobában, vagy Person-nél van, eltávolítja magát onnan. Life = 0;
  - **+void gasThreat():** A Person ezzel a hívással jelez az összes általa birtokolt Item felé ha gázos szobában tartózkodik. Alapvetően üres implementáció.
  - **+void setOwner(Person owner):** Beállítja a tárgy tulajdonosát és a szobát nullra állítja.
  - **+void teacherThreat():** A Person ezzel a hívással jelez az összes általa birtokolt Item felé ha egy oktató megtámadta őt. Alapvetően üres implementáció.
  - **+void tick():** Ha aktív, csökkenti az élettartamát. Ha ez 0-ra csökken, megsemmisíti a tárgyat.

9. Osztály: Beer

- **Felelősség**

Ha a hallgató aktiválja, akkor védelmet nyújt neki az oktatók ellen. Számontartja, hogy aktiválása után meddig van érvényben.

- **Ősosztályok**

Item

- **Interfészek**

- **Attribútumok**

- **Metódusok**
  - **+void activate():** [override] active = true, owneren dropRandomItem-et hív

- **+void teacherThreat():** [override] Az ownernek teacherProtectiont nyújt 0 prioritással.

10. Osztály: Cloth

- **Felelősség**

Ha aktív, megpróbál mindenkit megbénítani, aki a szobában tartózkodik, de csak az oktatókra lesz hatással. Használat után megsemmisül.

- **Ősosztályok**

Item

- **Interfészek**
- **Attribútumok**
- **Metódusok**
  - **+void activate():** [override] active = true, a szobában tartózkodó összes személyre meghívja a clothStun metódust, végül megsemmisíti magát.

11. Osztály: Camambert

- **Felelősség**

Ha aktiválódik, akkor elgázosítja azt a szobát, amelyikben van. Jelzi a szobának, hogy el van gázosítva.

- **Ősosztályok**

Item

- **Interfészek**
- **Attribútumok**
- **Metódusok**
  - **+void activate():** [override] active = true, az ownertől kapott szobán meghívja a gas metódust, majd megsemmisíti magát.

12. Osztály: Mask

- **Felelősség**

Ha aktiválódik, akkor elgázosítja azt a szobát, amelyikben van. Jelzi a szobának, hogy el van gázosítva.

- **Ősosztályok**

Item

- **Interfészek**
- **Attribútumok**
  - **-bool fake:** hamis tárgy esetén igaz
- **Metódusok**
  - **+void gasThreat():** [override] Ha nem fake, akkor gasProtectiont nyújt az élettartamával megegyező prioritással.

13. Osztály: TVSZ

- **Felelősség**

Ha az őt birtokló hallgatót megtámadja egy oktató és nincs magasabb prioritású védelme (Beer), akkor aktiválódik, ezzel megvédve a hallgatót. Számontartja, hogy hány alkalommal képes még megvédeni a hallgatót, mielőtt elhasználódna.

- **Ősosztályok**

Item

- **Interfészek**

- **Attribútumok**
  - **-bool fake:** hamis tárgy esetén igaz
- **Metódusok**
  - **+void teacherThreat():** [override] Ha nem fake, akkor teacherProtectiont nyújt az élettartamával megegyező prioritással.

14. Osztály: SlideRule

- **Felelősség**

Szól a Person-nek aki felvette, hogy ez egy SlideRule. Az oktató vagy hallgató eszerint kezeli a tárgyat.

- **Ősosztályok**

Item

- **Interfészek**

- **Attribútumok**
  - **-bool fake:** hamis tárgy esetén igaz
- **Metódusok**
  - **+void setOwner(Person owner):** [override] Owner beállítása, majd ha owner nem null, és nem fake, akkor owneren meghívja a slideRuleNotification metódust.

15. Osztály: AirFreshener

- **Felelősség**

Gázos szobában lerakva semlegesíti a gázhatást.

- **Ősosztályok**

Item

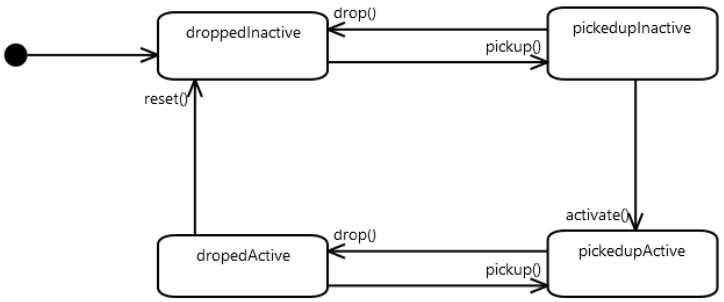
- **Interfészek**

- **Attribútumok**

- **Metódusok**
  - **+void activate():** [override] az ownertől lekért szobán meghívja az unGas metódust, majd megsemmisíti magát.

16. Osztály: Transistor

stm Transistor



- **Felelősség**

Két tranzisztort össze lehet kötni, mindegyik számon tartja párját. Ha létrejött egy aktív pár, akkor elmozdítja a Studentet a másik tranzisztor szobájába. Deaktiválja magát és a párját ezután. Ha egy Teacher veszi fel, akkor reseteli a másik tranzisztort.

- **Ősosztályok**

Item

- **Interfészek**

- **Attribútumok**
  - **-int id:** Ha felvesznek egy tranzisztort és aktiválják, kap egy id-t az aktiválásuk sorrendjében, és majd ezek az id-k szerint fognak összekapcsolódni.
  - **-Transistor pair:** Az összekapcsolt tranzisztor párja.
- **Metódusok**
  - **+tick():** [override] Nem csökkenti az élettartamot.
  - **+void setPair(Transistor t):** Beállítja a párt.
  - **+void activate():** [override] A tranzisztor bekapcsolása.

```
active = true
Ha még nincs párja:
        lekérdezi az owner transistorToPair tagját
        Ha null:
                Beállítja magát ennek
        Egyébként:
                Pair = transistorToPair
```

  - **+void setRoom(Room room):** [override] Szobához adás.

```
Ha van párja:
        Ha pair.room nem null:
                owner.teleport(pair.room))
                pair.deactivate()
                pair = null
                return
room = room
```

  - **+void reset()**: A párja hívja meg ezt a függvényt, amikor elpusztul (oktató megsemmisíti). Ekkor a tranzisztor deaktiválódik, amíg újra fel nem veszi egy hallgató és össze nem köti egy új tranzisztorral.
  - **+void deactivate():** Deaktiválja a tranzisztort. A párját nullra állítja.
  - **+void destroy():** [override] Megsemmisíti a tranzisztort, és reseteli a párját.

17. Osztály: Game

- **Felelősség**
- Működteti a játék programot.
- **Attribútumok**
  - **~List<Person> people:** A játékban szereplő entitások
  - **~List<Room> rooms:** A játékban levő szobák listája.
- **Metódusok**
  - **+public static void main():** Ebben a függvényben fut maga a program. Itt lesznek a tesztek hívásai és a játék futásának a megvalósításai.

8.2. A tesztek részletes tervei, leírásuk a teszt nyelvén

8.2.1. Tárgy felvétele hallgató által

- **Leírás**
  A hallgató felvesz egy sört, ami bekerül az inventoryjába, és eltűnik a szobából.
- **Ellenőrzött funkcionalitás várhatóhibahelyek**
  pickup(), inventory működése Hallgatónál
- **Bemenet**
  init
  Room room
  Student student room
  Beer beer room
  endinit
  pickup beer
- **Elvárt kimenet**
  status
  Room room
  Student student room
  Beer beer student
  endstatus

8.2.2. Tárgy felvétele oktató által

- **Leírás**
  Az oktató, ha felvesz egy tárgyat megsemmisíti.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  pickup() működikése Oktatónál
- **Bemenet**
  init
  Room room
  Teacher teacher room
  Beer beer room
  endinit
  act teacher pickup beer
- **Elvárt kimenet**
  status
  Room room
  Teacher teacher room
  endstatus

8.2.3. Oktató megöl egy hallgatót

- **Leírás**
  Az oktatók a velük egy szobában tartózkodó hallgatóknak “kiszívják a lelkét”, azaz megtámadják őket, ez védekezés hiányában a hallgató halálát jelenti, azaz az illető játékos számára ilyenkor a játék véget ért.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  teacherAttack() működése Hallgatónál
- **Bemenet**
  init
  Room room
  Student student room
  Teacher teacher room
  endinit
  tick
- **Elvárt kimenet**
  Players lost
  statusz
  Room room
  Teacher teacher room
  endstatus

8.2.4. Oktató megtámad oktatót

- **Leírás**
  Az oktatók a velük egy szobában tartózkodó oktatókat is megtámadják, de ez rájuk nincs hatással.
- **Ellenőrzött funkcionalitás, várható hibahelyek**

  teacherAttack() működése Oktatónál
- **Bemenet**
  init
  Room room
  Teacher teacher1 room
  Teacher teacher2 room
  endinit
  tick
- **Elvárt kimenet**
  status
  Room room
  Teacher teacher1 room
  Teacher teacher2 room
  endstatus

8.2.5. TVSZ használata

- **Leírás**
  Egy oktató megtámad egy Hallgatót, mivel egy szobában vannak, de mivel a Hallgató birtokában van egy TVSZ, az megvédi őt az Oktatóval szemben.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  passzív tárgyak és Oktató elleni védelmi rendszer működése

- **Bemenet**
  init
  Room room
  Student student room
  TVSZ tvsz student lifetime: 3
  Teacher teacher room
  endinit
  tick
- **Elvárt kimenet**
  status
  Room room
  Student student room
  TVSZ tvsz student lifetime: 2
  Teacher teacher room
  endstatus

8.2.6. Gázos szoba hatása oktatóra

- **Leírás**
  Egy Oktatót megtámad egy gázos szoba. Az Oktató lebénul.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  gasStun() működése Oktatónál
- **Bemenet**
  init
  Room room gassed: true
  Teacher teacher room
  endinit
  tick
- **Elvárt kimenet**
  status
  Room room gassed: true
  Teacher teacher room stunned: true
  endstatus

8.2.7. Rongy használata

- **Leírás**
  Egy Hallgató aktiválja a nála levő rongyot, ami megbénítja a vele egy szobában levő Oktatókat.
- **Ellenőrzött funkcionalitás, várható hibahelyek**

  activate() működése
  clothStun() működése Oktatónál
- **Bemenet**
  init
  Room room
  Student student room
  Cloth cloth student
  TVSZ tvsz student
  Teacher teacher room
  endinit
  act student activate cloth

- **Elvárt kimenet**
  status
  Room room
  Student student room
  TVSZ tvsz student
  Teacher teacher room stunned: true
  endstatus

8.2.8. Camembert használata

- **Leírás**
  Egy Hallgató egy Oktatóval van egy szobában és felnyitja a Camembert-t, mindketten lebénulnak.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  activate() működése,
  gasStun() működése Hallgatónál és Oktatónál
  drop() működése
- **Bemenet**
  init
  Room room gassed: false
  Student student room
  Camembert camembert student
  TVSZ tvsz student
  Teacher teacher room
  endinit
  act student activate camembert
  tick
- **Elvárt kimenet**
  status
  Room room gassed: true
  Student student room stunned: true
  Teacher teacher room stunned: true
  TVSZ tvsz room
  endstatus

8.2.9. Mask használata

- **Leírás**
  Egy Hallgató elgázosított szobába lép, de a nála lévő maszk megvédi a lebénulástól.
- **Ellenőrzött funkcionalitás, várható hibahelyek**

tick működése,
gasStun() működése Hallgatónál
Mask protectionje

- **Bemenet**
  init
  Room room
  Student student room
  Mask m lifetime: 3
  endinit

- **Elvárt kimenet**
  status
  Room room gassed: true
  Student student room stunned: false
  Mask m student lifetime: 2
  endstatus

8.2.10. Beer és TVSZ együttes használata

- **Leírás**
  Egy oktató megtámad egy Hallgatót, mivel egy szobában vannak. A Hallgató birtokában van egy söröspohár és egy TVSZ is. Ha a söröspohár nincs aktiválva akkor a TVSZ, ha aktiválva van akkor pedig a söröspohár fog védelmi nyújtani mivel utóbbinak mindig nagyobb a prioritása.
- **Ellenőrzött funkcionalitás, várható hibahelyek**

teacherAttack() működése
teacherThreat és teacherProtection működése
acceptProtection működése

- **Bemenet**
  init
  Room room
  Student student room
  Teacher teacher room
  TVSZ tvsz student
  Beer beer student isActive: true
  endinit
  tick
- **Elvárt kimenet**
  status
  Room room
  Beer beer student lifetime: 1
  TVSZ tvsz lifetime: 3
  Student student room
  Teacher teacher room
  endstatus

8.2.11. Szoba osztódás

- **Leírás**
  A szobák körönként véletlenszerűen “dönthetnek” úgy, hogy osztódnak. Az osztódó szoba két olyan szobára válik szét, amelyek egymás szomszédai lesznek, és megosztoznak a korábbi szoba tulajdonságain és szomszédain. Az osztódás után létrejövő két szoba között kétirányú ajtó jön létre. A két “új” szoba befogadóképessége az eredeti szobának a befogadóképességével fog megegyezni.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

divide() működése
szoba létrehozásának működése

- **Bemenet**
  init
  Room room1 12
  endinit
  divide room1 room2
- **Elvárt kimenet**
  status
  Room room1 12
  Room room2 12
  door1: room1 <> room2
  endstatus

8.2.12. Szoba összeolvadás

- **Leírás**
  A szobák körönként véletlenszerűen “dönthetnek” úgy, hogy egyesülnek egy szomszédukkal. Két szoba egyesülésével létrejövő új szoba a korábbi két szoba tulajdonságaival, szomszédaival, tárgyaival és entitásaival rendelkezik, de a befogadóképessége a nagyobb szoba befogadóképességével lesz azonos. Két szoba csak akkor egyesülhet, ha a két szobában lévő entitások összege nem haladja meg a nagyobb szoba befogadóképességét.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

merge() működése

- **Bemenet**
  init
  Room room1 12
  Room room2 10
  Room room3 10
  door1: room2 <> room3
  endinit
  merge room1 room2
- **Elvárt kimenet**
  status
  Room room1 12
  Room room3 12
  door1: room1 <> room3
  endstatus

8.2.13. Logarléc hallgatóval

- **Leírás**
  Egy hallgató felvesz egy logarlécet egy szobában. Ezzel meg is nyeri a játékot.
- **Ellenőrzött funkcionalitás, várható hibahelyek**

slideRuleNotification() működése Studentnél

- **Bemenet**
  init
  Room room
  Student student room
  SlideRule sr room
  endinit
  act student pickup sr
- **Elvárt kimenet**
  Players won
  status
  Room room
  Student student room
  SlideRule sr student
  endstatus

8.2.14. Logarléc oktatóval

- **Leírás**
  Egy oktató felvesz egy logarlécet a szobában. Rögtön el is dobja ennek hatására.
- **Ellenőrzött funkcionalitás, várható hibahelyek**

slideRuleNotification() működése Teachernél

- **Bemenet**
  init
  Room room
  Teacher teacher room
  SlideRule sr room
  endinit
  act student pickup sr
- **Elvárt kimenet**
  status
  Room room
  Teacher teacher room
  SlideRule sr room
  endstatus

8.2.15. Tranzisztor használata

- **Leírás**
  Egy hallgató egy szobában van és van nála két tranzisztor. Ezeket aktiválja majd lerakja az egyiket. Ezután átmegy egy másik szobába, ahol lerakja a másikat és ezzel elteleportál abba szobába ahonnan jött. Deaktiválódnak a tranzisztorok.
- **Ellenőrzött funkcionalitás, várható hibahelyek**

Transistor teleport függvénye
Transistorok deaktiválódása
Student mozgása
Transistorok párosodása

- **Bemenet**
  init
  Room room1
  Room room2
  door1: room1 > room2

  Student student room1
  Transistor t1 student isActive: false
  Transistor t2 student isActive: false
  endinit
  act student activate t1
  act student activate t2
  act student drop t1
  act student move door1
  act student drop t2
- **Elvárt kimenet**
  status
  Room room1
  Room room2
  door1: room1 > room2
  Student student room1
  Transistor t1 room1 isActive: false pair: t2
  Transistor t2 room2 isActive: false pair: t1
  endstatus

8.2.16. Tranzisztor felvétele oktató által

- **Leírás**
  Egy oktató felvesz egy lerakott tranzisztort, ami össze van kapcsolva egy másikkal. Megsemmisíti azt, amelyiket felvett és a párja pedig resetelődik.
- **Ellenőrzött funkcionalitás, várható hibahelyek**

Transistor resetelődése
Transistor megsemmisítése

- **Bemenet**
  init
  Room room1
  Room room2
  Teacher teacher room1
  Transistor t1 room1 isActive: false pair: t2
  Transistor t2 room2 isActive: true
  endinit
  act teacher pickup t1
- **Elvárt kimenet**
  status
  Room room1
  Room room2
  Teacher teacher room1
  Transistor t2 room2 isActive: false
  endstatus

8.2.17. Légfrissítő használata

- **Leírás**
  Egy hallgató gázos szobába lép, a Légfrissítő aktiválásával megszűnik a szoba gázossága. A légfrissítő egy használat után megsemmisül.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  Airfreshener aktiválása
  Szobának a gázosságának megszűntetése
- **Bemenet**
  init
  Room room gassed: true
  Student student room
  Airfreshener a student
  endinit
  act student activate a
- **Elvárt kimenet**
  status
  Room room gassed: false
  Student student room
  endstatus

8.2.18. Takarító kitessékel embereket és kiszellőztet

- **Leírás**
  Egy takarító olyan szobába lép amiben személyek vannak, akiket “kitessékel”, azaz szomszédos szobákba fog kerülni minden mozogni képes (nem bénult / ájult / van hova mozognia) személy rajta kívül.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  Airfreshener aktiválása
  Szobának a gázosságának megszűntetése

- **Bemenet**
  init
    Room room1 gassed: true
    Room room2
    door1: room1 > room2
    Student student room1
    Cleaner cleaner room1
  endinit
- **Elvárt kimenet**
  status
    Room room1 gassed: false
    Room room2
    door1: room1 > room2
    Student student room2
    Cleaner cleaner room1
  endstatus

8.2.19. Hallgató, oktató, takarító egy gázos szobában

- **Leírás**
  Egy elgázosított szobába belép egy Hallgató és elájul, elejti a nála lévő Camembert-t. Ezután belép egy Takarító, kiszellőztet, kitakarít, majd belép egy Oktató is és kiszívja az ájult Hallgató lelkét.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  gasStun() és drop() működése Hallgatónál,
  clean() működése
  move() működése
  elájult Hallgató esetén a teacherAttack() működése
- **Bemenet**
  init
    Room room1 gassed: true
    Room room2
    Room room3
    Room room4
    Student student room2
    Cleaner cleaner room3
    Teacher teacher room4
    Camembert camembert student
    door12: room2 <> room1
    door13: room3 <> room1
    door14: room4 <> room1
  endinit
  act student move door12
  act cleaner move door13
  act teacher move door14
- **Elvárt kimenet**
  status
    Room room1 gassed: false
    Room room2
    Room room3

Room room4
Cleaner cleaner room1
Teacher teacher room1
Camembert camembert room1
door12: room2 <> room1
door13: room3 <> room1
door14: room4 <> room1
endstatus

8.3. A tesztelést támogató programok tervei

8.3.1 A prototípus interface javítása, pontosítása:

A 7. dokumentum 7.1-es fejezete (Prototípus interface-definíciója) számos hiányosságot, pontatlanságot tartalmaz, melyek a tesztek megadásának szempontjából is lényegesek. Ezeknek a javításait, illetve pontosításait tartalmazza ez a fejezet.

- **A tick parancs bevezetése:** A korábbi tervből kimaradt az időegység végét jelző parancs, azonban e nélkül nem lehet olyan dolgokat tesztelni, mint bizonyos Item-ek (Pl: Beer) élettartama vagy pedig a Stun ideje. Erre szolgál a tick parancs:

  tick

  Leírás: Ugrik egy időegységet. A parancs hatására az időegység múlásához kötődő események megtörténnek (pl: Item lejárata, Stun vége). **Ez a parancs triggereli az NPC-k alap akcióit is, a Teacher esetében a hallgató lelkének kiszívását, a Cleaner esetében pedig a szoba kitakarítását, kiszellőztetését! A gázos szoba is ennek hatására fejti ki a mérgező hatását.**

  Opciók: -

- **Status-endstatus blokk:** A teszt elvárt kimenetét tartalmazó fájlban a végső játékállapot leírását a **status** kulcsszó előzi meg és az **endstatus** kulcsszó zárja!
- **Játék végének tesztelhetősége:** A játék lezárását is lehet teszttel ellenőrizni, ehhez az elvárt kimeneti fájlban a **status** előtti sorba kell írni a játék végeredményét (Players won/ Players lost).
- **Hamis Item-ek lehetősége:** Minden Item-leszármazott osztálynak beállítható a fake tulajdonsága. Ez alapértelmezetten hamis, ilyenkor az Item eredeti.
- **Játékállapot megadásának egyszerűsítése:** A korábbi tervvel ellentétben a játékállapot leírás során nem szükséges részekre tagolni és az elvárt kimenet során sem szükséges figyelni a betűrendet.
- **Szobák deklarációja:** A szobák deklarációjánál, mivel nincs szükség a rooms-end blokkra, szükség van arra, hogy a szoba neve előtt ott legyen, hogy Room!
  Példa: Room r1 gassed:true capacity: 5
- **Az ajtók elnevezésének lehetősége:** Ez a hiányosság a játékállapot-leíráshoz kapcsolódik. Az előző tervben az ajtók deklarációjának bemutatása során nincs megadva az ajtók neve, csak az szerepel, hogy “door”. Azonban az ajtók nevére szükség van, mivel a move parancs ezt kapja paraméterként. Ezért az ajtó deklarációk

megadásának módja a következő:
<*ajtó neve*>: room1 <> room2: ha kétirányú,
<*ajtó neve*>: room1 < room2: ha csak room2-ből room1 felé járható,
<*ajtó neve*>: room1 > room2: ha csak room1-ből room2 felé járható.

- **A tárgyak tulajdonságainak megadása:** A tárgyak deklarációjának leírásakor volt szó a tárgyak tulajdonságainak megadásáról. A tárgyak leírása a következő módon történik:
  <tárgy típusa> <tárgy neve> <tárgy szobája/tulajdonosa> <tárgy tulajdonságai>
  A tárgyak tulajdonságait a tulajdonság nevével és értékével adhatjuk meg, a nevet és az értéket egymástól kettősponttal elválasztva:
  <*tulajdonság neve*>: <*tulajdonság értéke*>
  A tulajdonságok megadásának sorrendje nem számít! A tulajdonság értéke lehet egy igaz-hamis érték (true vagy false) (pl: isActive), pozitív egész szám(pl: lifetime), illetve objektum hivatkozás név szerint (pl: a Transistor pair tulajdonsága). A bemenet során nem kötelező egy tárgynak minden lehetséges tulajdonságát megadni, a nem megadott tulajdonságok értéke ha a tulajdonság igaz-hamis típusú akkor hamis lesz, ha egész szám akkor a tárgy típusától függő alapértelmezett érték, ha hivatkozás akkor pedig null!
- **Szoba tulajdonságainak megadása:** A korábbi tervben leírtakkal ellentétben a szintaxis egyszerűbbé, átláthatóbbá tétele céljából a szobák tulajdonságainak megadását is a tárgyaknál bemutatott módon kell megtenni! Egy szobának három tulajdonsága van: capacity (egész szám), cursed(igaz-hamis), gassed(igaz-hamis).
  A capacity alapértelmezett értéke 10.
- **Tulajdonságok megadása a tesztek elvárt kimenetében:** Ha egy teszt elvárt kimenetében valamely objektum valamely tulajdonságának értékét nem kötjük ki az azt jelenti, hogy az adott tulajdonságra nem tesztelünk, értéke a teszt sikeressége szempontjából mindegy.
- **Tesztelés támogatás egyszerűsítése:** A bemeneti interface teszt módja esetén nincs szükség a **startgame** és az **endgame** parancsokra!
- **Tesztelés és véletlenszerűség:** Tesztelés során a futás teljesen determinisztikus az ellenőrizhetőség érdekében. ez azt jelenti, hogy csak a deklarált objektumok jönnek létre és csak a leírt akciók hajtódnak végre.

8.3.2 A tesztek lefutását támogató program terve

8.3.2.1 Általános leírás

A tesztek lefutását egy, a projekt részét képező, azonban külön osztályban és main függvényben megvalósított tesztprogram segíti. A tesztprogram semmi mást nem csinál, mint a számára a futtatás előtt átadott teszteket kiértékeli: ez abból áll, hogy fix sorrendben futtat minden egyes teszt-bemenetet, majd a teszt kimenetét (lásd: prototípu interface teszt mód) összehasonlítja a teszt-bemenettel párban átadott elvárt kimenettel. A tesztprogram futása során az összes, annak átadott teszteset lefut pontosan egyszer, a tesztek megadásának sorrendjében. A tesztprogram tekinthető a parancssori interfész további absztrakciójának a

tesztelés támogatása céljából, kiegészítve az elvárt és a tényleges kimenet összehasonlításának képességével.

8.3.2.2 Tesztek megadásának menete

Egy teszt bemenetét és elvárt kimenetét egy-egy szövegfájlban (txt) kell megadni. A bemenetet tartalmazó fájl neve testN.txt, az elvárt kimenetet tartalmazó fájl neve pedig outputN.txt, mely fájlnevekben az N a teszteset sorszámát jelenti 0-tól kezdve növekvő sorrendben. A sorszámozásra figyelni kell, mivel a tesztprogram a test0.txt és az output0.txt fájlokat fogja keresni először: ha ezeket nem találja, akkor nem fog továbbmenni a nagyobb sorszámú tesztekre. Az összes tesztfájlt egyetlen mappában szükséges megadni.

A teszt-bemenet egy, a prototípus-interface bemeneti nyelven (lásd 7. dokumentum 7.1.2 fejezet) megírt, a teszt üzemmód tulajdonságai szerinti program, mely a kezdő játékállapot leírásával kezdődik egy init-endinit blokkban (lásd 7. dokumentum 7.1.2 fejezetben a játékállapot-definíciós nyelv).

A teszt elvárt kimenete két részből áll, ebből az első opcionális. Először az elvárt hibaüzeneteket kell felsorolni, vagy a játék lezárt ha vannak, aztán pedig a játék elvárt végállapotát (status-endstatus blokkban). A játék lezárását is lehet teszttel ellenőrizni, ehhez az elvárt kimeneti fájlban a **status** előtti sorba kell írni a játék végeredményét (Players won/ Players lost).

8.3.2.3 A tesztprogram interfész-leírása

A tesztprogram az indításakor egyből felhasználói bemenetet vár: a teszteket tartalmazó mappa elérési útját. A felhasználó addig próbálkozhat, amíg nem sikerül érvényes elérési utat megadnia. Ha az elérési út érvényes volt, akkor a program megpróbálkozik a tesztek futtatásával és ha nem találja az első tesztet (test0.txt és output0.txt), akkor a program kilépésre vár. Ha a program kilépésre vár az azt jelenti, hogy a felhasználó két lehetőség közül választhat: új elérési út megadása vagy a program bezárása. A program a tesztek futása előtt azt is megkérdezi, hogy szeretnénk-e debug módban futtatni a teszteket: ha igen, akkor a teszt futása során megjelennek a Skeleton programból már ismert hívási láncok is. Ha volt legalább egy teszt a mappában (azaz a fájlok ottvoltak), akkor a lefutott tesztek státuszjelentései megjelennek egymás alatt a következő módon:

#TESZT 0: <teszt0 státusza>
<Ha teszt0 sikertelen, akkor az elvárttól való eltérések felsorolása. Ha teszt0 hibás, akkor itt az szerepel, hogy az első észlelt hiba melyik tesztfájl hányadik sorában szerepel.>

#TESZT1: <teszt1 státusza>
<ha teszt0 sikertelen, akkor az elvárttól való eltérések felsorolása. Ha teszt1 hibás, akkor itt az szerepel, hogy az első észlelt hiba melyik tesztfájl hányadik sorában szerepel>
.
.
…..

#TESZTN: <tesztN státusza>
<Ha tesztN sikertelen, akkor az elvárttól való eltérések felsorolása. Ha tesztN hibás, akkor itt az szerepel, hogy az első észlelt hiba melyik tesztfájl hányadik sorában szerepel.>

Amikor ez a kiírás megtörtént, akkor a program kilépésre vár. Egy teszt státusza négyféle lehet: sikeres(SUCCESS), hibás(SYNTAX ERROR), sikertelen(FAILED), permanens hiba(SYSTEM ERROR). Sikeres akkor, ha a végső játékállapot megegyezik az elvárttal. Sikertelen, ha a teszt maga sikeresen lefutott, de a végső játékállapot és az elvárt játékállapot között eltérések vannak. Hibás, ha valamelyik tesztfájlban a program által értelmezhetetlen szöveg található. Ha a teszt futása hibás, akkor a program megjeleníti azt, hogy az adott teszt első hibája melyik fájl hányadik sorában található. Permanens hiba mint státusz akkor fordul elő, ha 401 Unexpected error állt elő. Ez mindenképp programozói hibára utal. Ilyenkor a stack trace megjelenik és az adott teszt futása megszakad.

Ha a teszt sikertelen, akkor az eltérések megjelennek egymás alatt, egy sorban egy eltérés. Kétfajta eltérés lehetséges: egzisztenciális (EXISTENTIAL) vagy differenciális (DIFFERENTIAL).

Az egzisztenciális eltérés azt jelenti, hogy vagy egy olyan objektum létezik a valós kimeneti állapotban melynek az elvárt állapot szerint nem kellene léteznie, vagy azt, hogy egy olyan objektum nem létezik a valós kimeneti állapotban, melynek az elvárt állapot szerint léteznie kellene. Erre egy példa:

EXISTENTIAL: Cam1 (Camembert) EXISTS (vagy DOESN’T EXIST) CONTRARY TO THE EXPECTED STATE

A defferenciális eltérés (DIFFERENTIAL) valamely objektum valamely tulajdonságának az elvárttól való eltérését jelenti. Erre is egy példa:

DIFFERENTIAL: Cam1 (Camembert) HAS isActive: false CONTRARY TO THE EXPECTED isActive: true

Vagyis a Cam1 nevű Camembert típusú objektum isActive tulajdonsága az elvárt hamis helyett igaz értéket vett fel.

Ha a játék végeredményét illetően volt eltérés, akkor DIFFERENTIAL eltérés íródik ki olyan formában, mintha az egyébként nem létező Game osztály isFinsished enum típusú attribútumában lenne eltérés.

8.3.2.4 Az elvárt és a tényleges kimenet összehasonlításának mechanizmusa

Az elvárt és a tényleges kimeneti állapotok összehasonlítása nem egyszerű sztring-összehasonlítás, hanem annál kifinomultabb módszer szerint, katalógusok alapján történik. Egy katalógus jelen esetben kulcs-érték párokat tárol. Egy teszteset futása során több katalógus is létrejön, egy maguknak a létrejött objektumoknak, ebben a kulcsok az objektumok nevei, az értékek pedig maguk az objektumok. A futás során az összes objektum-hivatkozás ezen keresztül történik. Egy másik katalógus is létrejön, ez az objektumok neveihez, mint kulcsokhoz tárolja az objektumokhoz tartozó tulajdonság-katalógusokat, melyekben a kulcsok a tulajdonságok nevei. Az elvárt kimeneti állapothoz is létrejönnek ezen katalógusok. A két különböző állapothoz tartozó azonos szerepű katalógusok tartalmának összehasonlítása módot ad az egzisztenciális és differenciális eltérések feltárására.

Napló

Kezdet	Időtartam	Résztvevők	Leírás
2024.04.10. 21:00	0.5 óra	Cardinael Görömbey Riba	Feladatok kiosztása
2024.04.13.	6 óra	Görömbey	Osztályok részletes leírása.
2024.04.14.	4 óra	Szakos	A 8.3 fejezet megírása
2024.04.15	4 óra	Cardinael	Tesztek elkészítése, ellenőrzése