

# A Logarléc

## Egyesített dokumentáció

9 – Echo

Konzulens:  
Dobos-Kovács Mihály

### Csapattagok

Mészáros Barna	PRYJJV	barna.meszaros02@gmail.com
Rékó Márton	KAMFNQ	rekomarci@gmail.com
Csiszár Alex Gergő	PL4XKM	csiszaralex@gmail.com
Csirmaz Gáspár	DVSBKJ	csirmaz@gmail.com

2024. május 21.

2. Követelmény, projekt, funkcionalitás	9
2.1 Bevezetés	9
2.1.1 Cél	9
2.1.2 Szakterület	9
2.1.3 Definíciók, rövidítések	9
2.1.4 Hivatkozások	9
2.1.5 Összefoglalás	9
2.2 Áttekintés	10
2.2.1 Általános áttekintés	10
2.2.2 Funkciók	11
2.2.3 Felhasználók	13
2.2.4 Korlátozások	13
2.2.5 Feltételezések, kapcsolatok	13
2.3 Követelmények	14
2.3.1 Funkcionális követelmények	14
2.3.2 Erőforrásokkal kapcsolatos követelmények	17
2.3.3 Átadással kapcsolatos követelmények	18
2.3.4 Egyéb nem funkcionális követelmények	18
2.4 Lényeges use-case-ek	19
2.4.1 Use-case leírások	19
2.4.2 Use-case diagram	20
2.5 Szótár	20
2.6 Projekt terv	22
<b>3. Analízis modell kidolgozása</b>	<b>23</b>
3.1 Objektum katalógus	23
3.1.1 Szoba	23
3.1.2 Hallgató	23
3.1.3 Tranzisztor	23
3.1.4 TVSZ denevérbőr	23
3.1.5 Szent söröspohár	23
3.1.6 Káposztás camembert	23
3.1.7 Oktató	24
3.1.8 Maszk	24
3.1.9 Nedves táblatörölő rongy	24
3.1.10 Logarléc	24
3.2 Statiszkus struktúra diagramok	25
3.3 Osztályok leírása	25
3.3.1 Labyrinth	25
3.3.2 Room	26
3.3.3 IItemTarget	26

3.3.4 IItemHolder	27
3.3.5 Player	27
3.3.6 Teacher	28
3.3.7 Student	28
3.3.8 Item	28
3.3.9 IVisitor	29
3.3.10 TVSZ	29
3.3.11 Beer	30
3.3.12 Transistor	31
3.3.13 Mask	31
3.3.14 Sponge	32
3.3.15 Cheese	33
3.3.16 Logarléc	33
3.4 Szekvencia diagramok	35
3.5 State-chartok	40
<b>4. Analízis modell kidolgozása</b>	<b>41</b>
4.1 Objektum katalógus	41
4.1.1 Szoba	41
4.1.2 Hallgató	41
4.1.3 Tranzisztor	41
4.1.4 TVSZ denevérbőr	41
4.1.5 Szent söröspohár	41
4.1.6 Káposztás camembert	42
4.1.7 Oktató	42
4.1.8 Maszk	42
4.1.9 Nedves táblatörlő rongy	42
4.1.10 Logarléc	42
4.2 Statikus struktúra diagramok	43
4.3 Osztályok leírása	44
4.3.1 Labyrinth	44
4.3.2 Room	44
4.3.3 IRoomManager	45
4.3.4 Player	45
4.3.5 Teacher	46
4.3.6 Student	46
4.3.7 IItem	47
4.3.8 TVSZ	48
4.3.9 Beer	48
4.3.10 Transistor	49
4.3.11 Mask	49
4.3.12 Sponge	50

4.3.13 Cheese	50
4.3.14 Logarléc	51
4.4. Szekvencia diagramok	52
Room-Player interakciók	52
Automatikus Player-Item interakció	54
Item használat	56
Tranzisztor	57
Controller-Room interakciók	58
End of Game	60
4.5. State-chartok	60
Room	60
Player	61
Itemek	62
<b>5. Szkeleton tervezése</b>	<b>64</b>
5.0 Változtatások	64
5.1 A szkeleton modell valóságos use-case-ai	64
5.1.0 Elemi események	64
5.1.1 Use-case diagram	67
5.1.2 Use-case leírások	68
5.2 A szkeleton kezelői felületének terve, dialógusok	78
5.3 Szekvencia diagramok a belső működésre	79
5.3.1 Student picks Item (standard)	79
5.3.2 Student picks Sponge	79
5.3.3 Student picks Logarléc	79
5.3.4 Student picks Paired transistor	79
5.3.5 Teacher picks item (standard)	80
5.3.6 Student tries to enter cursed room	80
5.3.7 Student enters empty room	80
5.3.8 Student tries to enter full room	81
5.3.9 Student enters poisoned room	81
5.3.10 Student enters room with other student(s)	81
5.3.11 Student enters room with teacher(s)	82
5.3.12 Student tries to leave cursed room	82
5.3.13 Student leaves poisoned room	82
5.3.14 Student's room is poisoned	82
5.3.15 Student's room is no longer poisoned	83
5.3.16 Teacher tries to enter cursed room	83
5.3.17 Teacher enters empty room	83
5.3.18 Teacher tries to enter full room	83
5.3.19 Teacher enters poisoned room	84
5.3.20 Teacher enters room with student(s)	84

5.3.21 Teacher enters room with other teacher(s)	84
5.3.22 Teacher tries to leave cursed room	85
5.3.23 Teacher leaves poisoned room	85
5.3.24 Teacher's room is poisoned	85
5.3.25 Teacher's room is no longer poisoned	85
5.3.26 Student drops beer	86
5.3.27 Student drops cheese	86
5.3.28 Student drops mask	87
5.3.29 Student drops sponge	87
5.3.30 Student drops TVSZ	87
5.3.31 Teacher drops beer	88
5.3.32 Teacher drops cheese	88
5.3.33 Teacher drops mask	88
5.3.34 Teacher drops sponge	89
5.3.35 Teacher drops TVSZ	89
5.3.36 Student pairs transistors	89
5.3.37 Student activates transistor	90
5.3.38 Student places activated transistor but pair is not placed	90
5.3.39 Student teleport after dropping it	90
5.3.40 Student places inactive or unpaired transistor	91
5.3.41 Room cannot merge student	91
5.3.42 Room cannot merge transistor	91
5.3.43 Room merge	92
5.3.44 Room cannot split student	92
5.3.45 Room cannot split transistor	92
5.3.46 Room split	92
5.3.47 Student saved by Beer	93
5.3.48 Student saved by Sponge	93
5.3.49 Student saved by Mask	93
5.3.50 Student saved by TVSZ	93
5.3.51 Cheese timer	94
5.3.52 Beer timer	94
5.3.53 Sponge timer	95
5.3.54 Mask pause	95
5.3.55 Teacher Drops Logarléc	95
5.3.56 Teacher Places Unpaired Transistor	96
5.4 Kommunikációs diagramok	97
5.4.1 Student drops items	97
5.4.2 Teacher Drops items	98
5.4.3 Student use	99
5.4.4 Pick Item	100

5.4.5 Room functionality	100
5.4.6 Room Interactions	101
5.4.7 Item functionality	102
<b>6. Szkeleton beadás</b>	<b>104</b>
6.1 Fordítási és futtatási útmutató	104
6.1.1 Fájllista	104
6.1.2 Fordítás	105
6.1.3 Futtatás	105
<b>7. Prototípus koncepciója</b>	<b>106</b>
7.0 Változás hatása a modellre	106
7.1.1 Módosult osztálydiagram	107
7.1.2 Új vagy megváltozó metódusok	108
7.1.3 Szekvencia-diagramok	109
7.2 Prototípus interface-definíciója	111
7.2.1 Az interfész általános leírása	111
7.2.2 Bemeneti nyelv	111
7.3 Összes részletes use-case	114
7.4 Tesztelési terv	117
7.5 Tesztelést támogató segéd- és fordítóprogramok specifikálása	121
<b>8. Részletes tervezet</b>	<b>122</b>
8.1 Osztályok és metódusok tervezet.	122
8.1.1 Room	123
8.1.2 Player	126
8.1.3 Student	127
8.1.4 Teacher	128
8.1.5 Cleaner	128
8.1.6 TVSZ	128
8.1.7 Beer	129
8.1.8 EBeerState (Enum)	130
8.1.9 Transistor	130
8.1.10 Mask	130
8.1.11 Sponge	131
8.1.12 Cheese	132
8.1.13 Purifier	133
8.1.14 Logarlec	134
8.1.15 Labyrinth	135
8.1.16 TimedObject	136
8.1.17 Timer	137
8.1.18 Controller	137
8.2 A tesztek részletes tervezet, leírásuk a teszt nyelvén	140
8.2.1 * teszt	140

8.2.2 Transistor teleportál	141
8.2.3 Transistor (aktiválatlan) lehelyezés	142
8.2.4 Transistor (nem párosított) lehelyezés	142
8.2.5 Transistor aktiválás	143
8.2.6 Sajt gázosít	143
8.2.7 Légfrissítő tisztít, maszk megvéd	144
8.2.8 Sör megvéd, majd lejár	144
8.2.9 Logarléc nyer	145
8.2.10 Logarléc hamis	145
8.2.11 Maszk lejár	146
8.2.12 Maszk hamis	146
8.2.13 Szivacs működik	147
8.2.14 TVSZ megvéd	147
8.2.15 TVSZ lejár	148
8.2.16 TVSZ hamis	148
8.2.17 Mozgásteszt	149
8.2.18 Szoba split	149
8.2.19 Szoba merge	150
8.2.20 Átkozott szoba	150
8.2.21 Gázos kilépés	151
8.2.22 Felkelés	151
8.2.23 Felvétel/Eldobás	153
8.3 A tesztelést támogató programok tervezettsége	155
<b>10. Prototípus beadása</b>	<b>156</b>
10.1 Fordítási és futtatási útmutató	156
10.1.1 Fájllista	156
10.1.2 Fordítás	158
10.1.3 Futtatás	158
10.2 Tesztek jegyzőkönyvei	159
10.2.1 * -os teszt	159
10.2.2 Transistor teleportál teszt	159
10.2.3 Transistor (aktiválatlan) lehelyezés teszt	159
10.2.4 Transistor (nem párosított) lehelyezés teszt	159
10.2.5 Transistor aktiválás teszt	159
10.2.6 Sajt teszt	159
10.2.7 Légfrissítő tisztít, maszk megvéd teszt	159
10.2.8 Sör megvéd, majd lejár teszt	159
10.2.9 Logarléc teszt	159
10.2.10 Logarléc teszt	159
10.2.11 Maszk lejár teszt	160
10.2.12 Maszk hamis teszt	160

10.2.13 Szivacs működik teszt	160
10.2.14 TVSZ megvéd teszt	160
10.2.15 TVSZ lejár teszt	160
10.2.16 TVSZ hamis teszt	160
10.2.17 Mozgásteszt teszt	160
10.2.18 Szoba split teszt	160
10.2.19 Szoba merge teszt	160
10.2.20 Átkozott szoba teszt	160
10.2.21 Gázos kilépés teszt	160
10.2.22 Felkelés teszt	161
10.2.23 Felvétel/Eldobás teszt	161
<b>11. Grafikus felület specifikációja</b>	<b>162</b>
11.1 A grafikus interfész	162
11.2 A grafikus rendszer architektúrája	163
11.2.1 A felület működési elve	164
11.2.2 A felület osztály-struktúrája	164
11.3 A grafikus objektumok felsorolása	166
11.3.1 Commands	166
11.3.2 Controller	166
11.3.3 MainWindow	167
11.3.4 VTeacher	167
11.3.5 VCleaner	168
11.3.6 VStudent	168
11.3.7 VRoom	168
11.3.8 VMask	169
11.3.9 VTVSZ	169
11.3.10 VCheese	170
11.3.11 VLogarlec	170
11.3.12 VPurifier	170
11.3.13 VSponge	171
11.3.14 VTransistor	171
11.3.15 Mask	172
11.3.16 TVSZ	172
11.3.17 Cheese	172
11.3.18 Transistor	173
11.3.19 Logarléc	173
11.3.20 Purifier	173
11.3.21 Beer	174
11.3.22 Sponge	174
11.4 Kapcsolat az alkalmazói rendszerrel	175
<b>13. Grafikus változat beadása</b>	<b>178</b>

13.1 Fordítási és futtatási útmutató	178
13.1.1 Fájllista	178
13.1.2 Fordítás és telepítés	182
13.1.3 Futtatás	182
<b>14. Összefoglalás</b>	<b>183</b>
14.1 A projektre fordított összes munkaidő	183
14.2 • Projekt összegzés	183
14.2.1 Mit tanultak a projektből konkrétan és általában?	183
14.2.2 Mi volt a legnehezebb és a legkönnyebb?	183
14.2.3 Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?	183
14.2.4 Ha nem, akkor hol okozott ez nehézséget?	184
14.2.5 Milyen változtatási javaslatuk van?	184
14.2.6 Milyen feladatot ajánlanának a projektre?	184
14.2.7 Egyéb kritika és javaslat	184

## 2. Követelmény, projekt, funkcionalitás

### 2.1 Bevezetés

#### 2.1.1 Cél

Ennek a dokumentumnak a célja, hogy részletes leírást adjon a Logarléc nevű játékról és az elkészítéséhez szükséges projektről. A dokumentum pontosan elmagyarázza a játék működését, a működéssel kapcsolatos követelményeket és a projekt felépítését.

A dokumentum végén egy napló található, amely a csapattagok tevékenységeit tartalmazza.

#### 2.1.2 Szakterület

A szoftverünk egy videójáték, melyet kikapcsolodási célokra lehet majd használni. A szoftver szórakoztatáson kívül más szolgáltatást nem nyújt.

#### 2.1.3 Definíciók, rövidítések

Kifejezés	Magyarázat
Felhasználó	A szoftvert igénybe vevő személy.
Hordozhatóság	Meghatározza, hogy a szoftvert mennyire könnyű különböző rendszereken futtatni.
Use-case	Magyarul használati eset, a rendszer és egy másik szereplő közötti interakció, mely egy megadott célohoz vezet.
OOP	Objektum Orientált Programozás, irányelveket határoz meg, amelyek alapján a szoftvert elkészítjük.

#### 2.1.4 Hivatkozások

- <https://github.com/MBakter/echo> - GitHub repository
- <https://niif.cloud.bme.hu/> - BME Cloud

#### 2.1.5 Összefoglalás

- Áttekintés

- **Általános áttekintés:**  
Általános tájékoztatás a szoftver főbb részeiről és ezek kapcsolatáról. A felhasználói kapcsolat alapja.
- **Funkciók:**  
A szoftver működésének végleges, részletes leírása.
- **Felhasználók:**  
A szoftver felhasználónak jellemzői.
- **Korlátozások:**  
A szoftverre vonatkozó előírások.
- **Feltételezések, kapcsolatok:**  
A Hivatkozásokban felsorolt anyagok, weboldalak kapcsolódása a feladathoz.

- **Követelmények**
  - **Funkcionális követelmények:**  
Követelmények és leírásuk halmaza, ezeket a szoftvernek minden képp teljesítenie kell.
  - **Erőforrásokkal kapcsolatos követelmények:**  
A szoftver fejlesztésével és használatával kapcsolatos követelmények.
  - **Átadással kapcsolatos követelmények:**  
A szoftver átadásával, telepítésével, üzembe helyezésével kapcsolatos követelmények
  - **Egyéb nem funkcionális követelmények:**  
A biztonsággal, hordozhatósággal, megbízhatósággal, tesztelhetőséggel, a felhasználóval kapcsolatos követelmények.
- **Lényeges use-case-ek**
  - **Use-case leírások:**  
A Funkcionális követelmények részben megfogalmazott követelmények közül az alapvető és fontos követelményekhez tartozó használati esetek megadása táblázatos formában.
  - **Use-case diagram:**  
Diagram a megfogalmazott használati esetekhez.
- **Szótár:**  
A követelményekben felhasznált fogalmak és rövidítések magyarázata.
- **Projekt terv:**  
A projekt végrehajtásának részletei, többek között a csapattagok beosztása, a csoporthunka eszközei és a végrehajtás lépései.
- **Napló:**  
A csapattagok hozzájárulásainak naplázása.

## 2.2 Áttekintés

### 2.2.1 Általános áttekintés

A szoftver főbb részei és azok kapcsolatai:

- **Szobák**
  - Az OOP szemlélet origója, e körül "forog" minden. A memóriában irányított gráfikent, szomszédsági mátrixal van tárolva, de a felhasználó ebből minden csak egy korlátozott képet kap. Egy szobát lát és annak adott pillanatban elérhető szomszédait.
  - Nem ismeri a játékosokat, velük nem törödik
  - Tárgyat tartalmazhatnak
- **Tárgyak**
  - Szobák tartalmazhatják őket
  - Játékosok interaktálhatnak velük. Tehát felveheti, használhatja és eldobhatja
  - A különféle tárgyak egymástól elkülönítve saját tulajdonságokkal rendelkeznek
- **Játékosok**
  - Interfácen keresztül kapcsolódnak a szobákhoz és a tárgyakhoz
  - A tárgyakat használják
  - A szobáktól kérdezik épp hol van és hova tud menni
  - minden játékos tudja magáról, hogy melyik szobában van

- Hallgatók az oktatókkal is kommunikálnak, egymást kérdezik egy szobában vannak-e
- A felhasználó a hallgatókkal áll közvetlen kapcsolatban. Nekik mondja mely tárggyal interaktáljon illetve ő változtathatja a hallgató jelenlegi szobáját
- **Grafika**
  - Független a logikától, egyik módosítása nem, de bármelyik bővítése érinti a másikat

## 2.2.2 Funkciók

- **Szobák:**
  - Ajtókat tartalmaznak, amik másik szobába visznek. Egy ajtón való átjárás lehet két illetve egyirányú is.
  - A szobák befogadóképessége előre definiált
  - Típusai:
    - **Tele szoba:** Senki nem léphet be
    - **Gázos szoba:** Belépést követően adott ( a játékban konfigurálható ) időre eszméletvesztő lesz a játékos.
    - **Elátkozott szoba:** Ajtajai bizonyos, változó időközökre eltűnnek. Egy bent tartózkodó játékos nem tud kimenni illetve kívülről sem lehet belépni.
  - Folyton mozgásban: A szobák nem stacionáriusok, tehát különböző jelenségekre képesek bizonyos, változó és a játékban konfigurálható időközönként. Ezek a jelenségek lehetnek:
    - **Egyesülés:** Két szomszédos szoba egyesülésekor az új szoba a korábbi két szoba tulajdonságaival és szomszédaival rendelkezik, de a befogadóképessége a nagyobb szoba befogadóképességével lesz azonos. Két szoba csak akkor egyesülhet ha egyikben sem tartózkodik játékos. A két szobában lévő tárgyak mind átkerülnek az új szobába.
    - **Osztódás:** Az osztódó szoba két olyan szobára válik szét, amelyek egymás szomszédai lesznek, és megosztznak a korábbi szoba képességein és szomszédain (a korábbi szomszédonként csak az egyik, vagy csak a másik “új” szobának lesznek szomszédai).
  - Csak abban az esetben egyesülhet illetve osztódhat egy szoba ha nem tartózkodik benne játékos
- **Tárgyak:**
  - A szobákban elhelyezett tárgyakat fel lehet venni illetve el lehet dobni (*abba a szobába ahol épp tartózkodik a játékos*) ezzel újra felvehetővé téve.
  - Ha egy hallgató vesz fel egy tárgyat, felhasználhatja annak képességeit. Maximum 5 tárgyat birtokolhat
  - Ha egy oktató vesz fel egy tárgyat, számára használhatatlan lesz (*de meggátolja a hallgatókat a felvételében*). Maximum 10 tárgyat birtokolhat, de egy típusú tárgyból maximum 2 lehet nála
  - Tárgyak listája és tulajdonságaik:
    - **Logarléc:** Felvétellel és használata után a hallgatók megnyerik a játékot
    - **TVSZ denevérbőr:** Három életpontja van a tárgynak. Amennyiben egy oktató és a denevérbőrt birtokló hallgató egy szobába kerül, a denevérbőrnek egygel kevesebb, de a hallgató nem esik ki a játékból.

Ekkor, amíg abban a szobában tartózkodik a hallgató és az oktató aki miatt elhasználódott a tárgy, nem történik semmi. Korlátlan idejű lehetősége van elhagyni a szobát a hallgatónak, de amennyiben egy másik oktató is belép a szobába, a denevérbőr még egy életpontot veszít és így tovább. Ha elhagyta a szobát, de a következő szobák bármelyikében szintén belép egy oktató, a denevérbőr újra veszít 1 életpontot. Ha mindenkor életpontja lement, elveszti varázserejét. Ilyenkor még a játékosnál marad, de használhatatlan lesz.

- **Szent söröspohár:** Ha egy hallgató felveszi és használja a tárgyat, egy bizonyos (*a játékban konfigurálható*) ideig a hallgatók immunisak az oktatókra, tehát nem veszítik el a lelküket a közös szobában való tartózkodás miatt.
- **Nedves táblatörlő rongy:** Ennek a tárgynak (*a játékban konfigurálható időtartamú*) lejáratú ideje van. Miután egy hallgató felvette, elkezd száradni és adott idő után elveszti erejét, de a hallgatónál marad. Amíg a tárgy varázserővel bír, addig minden oktatót aki egy szobában tartózkodik egy hallgatóval, lebénít a játékban konfigurálható mennyiséggű ideig.
- **Káposztás camembert:** Ha használjuk ezt a tárgyat a szoba amiben vagyunk gázos szobává változik.
- **Tranzisztorok:** Ha két tranzisztor is felvesz egy hallgató, majd használja az egyiket, akkor összekapcsolja őket. Egy összekapcsolt állapotban lévő tranzisztor le tud rakni egy szobában. Ha a nála maradó, összekapcsolt tranzisztor párrát egy másik szobában használja (*bekapcsolja*) majd lerakja akkor abba a szobába teleportál ahova az első tranzisztor lerakta. A második-ként lerakott tranzisztor ezután kikapcsol. Tehát visszafelé nem lehet teleportálni vele. A lerakott tranzisztorok lent maradnak és bármeddig lehet használni őket. Ha a kikapcsolt tranzisztor valaki felveszi, bekapcsolja és lerakja akkor teleportálni tud vele, de csak egy irányban.
- **FFP2 Maszk:** Ha gázos szobába lép akkor magától aktiválódik. Egy adott (*játékban konfigurálható*) ideig immunis az eszméletvesztéssel szemben. Ha lejár ez az idő akkor elhasználódik a tárgy és eszméletvesztett lesz a játékos abban az esetben ha az elhasználódás pillanatában még a gázos szobában tartózkodik. Ha az elhasználódás előtt elhagyja a szobát akkor a tárgy lejáratú ideje megáll, majd úgy marad. Ha újabb gázos szobába lép, akkor az aktuális időtől kezdve indul újra az elhasználódás.

- **Játékosok:**

- Állapotai:
  - **Eszméletvesztett:** Nem tud mozogni és minden tárgyat eldob maga mellé
  - **Terhelt:** Nem tud több tárgyat felvenni. Akkor lesz terhelt egy játékos, ha a számára maximális mennyiséggű tárgyat birtokolja
  - **Elbocsátott:** Csak hallgatóra vonatkozik. Többé nem tud részt venni a játékban.
- Hallgatók:
  - A felhasználó irányítja, egy szobában tartózkodik és látja, hogy onnan hova mehet
  - Tud mozogni a szobák között

- Tárgyakat látja, fel tudja venni, használni és eldobni
  - **Célja:** A logarléc nevű tárgyat megtalálni, felvenni és használni
- Oktatók:
    - A szobák között mozog és üldözi a hallgatókat
    - Látja és fel tudja venni és eldobni a tárgyakat
    - **Célja:** Elvenni a lelkét az összes hallgatónak.
    - **Lélek elvétel:** Amennyiben egy hallgató egy szobában tartózkodik egy oktatóval, a hallgatónak el lesz véve a lelke így kiesik a játekból.

### 2.2.3 Felhasználók

- **Játékosok:** A szoftver célirányosan nekik készül, az internetről letölthetik és használhatják. Amennyiben hibával találkoznak vagy esetleg a játék bővítéséhez van javaslatuk, egy github issueban jelezhetik a hozzájárulók illetve a fejlesztőknek.
- **Hozzájárulók (Contributors):** Segíthetnek a szoftver fejlesztésében, hibák észleléseben illetve javításában, új ötletek kitalálásában, azok megvalósításában. A játék forráskódjának közvetlen módosításához engedélyük nincs, azonban külön ágakra dolgozhatnak.  
A játékosok hozzájárulók lehetnek a fejlesztők külön engedélyével amit a velük való kapcsolatfelvétel és konzultálás következtében kaphatnak meg.
- **Fejlesztők:** a felhasználók egy nagyon szűk csoportja akik átlátják az egész szoftver működését, aktívan részt vesznek annak fejlesztésében, birtokukban van minden, a szoftverhez kapcsolódó szellemi termék és döntési joggal bírnak a szoftver minden aspektusa felett.

### 2.2.4 Korlátozások

- A kész termékkel kapcsolatos bármilyen észrevételt illetve megjegyzést egy github issueba kell jegyezni
- A szoftver terjesztése, a fejlesztők említésének vagy a github repository linkelése hiányában tilos
- Maximum 8 játékos játszhat egy időben a játékkal, de oktatók száma konfigurálható
- Csak 1 számítógépről lehet játszani, több gépes kommunikáció nem támogatott
- A felhasználók csak klaviatúrát és egeret használhatnak, az érintőképernyős irányítás nem támogatott

### 2.2.5 Feltételezések, kapcsolatok

- Github repository link: Itt található a nyílt forráskódú projekt. Itt lehet az észrevételeket jegyezni, megjegyzéseket tenni, a szoftver fejlesztését végig követni illetve csatlakozni.
- BME Cloud: Felhőszolgáltatás, ezen található a Windows 10 rendszer, melyen a szoftverünknek futnia kell majd.

## 2.3 Követelmények

### 2.3.1 Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
J001	Lejár az idő, játék vége.	Kivárnai a határidőt..	Magas	Feladatkiírás		
J002	Találkozás oktatóval, játékos kiesik	Oktatószobába lépés, oktató belép a szobába.	Magas	Feladatkiírás		Alapeset. Különleges tárgyak hatással lehetnek
J003	Találkozás oktatóval, ellenőrzi a különleges tárgyak meglétét, csökkenti azok hatását	Különleges védelmet adó tárgyak megfigyelése.	Magas	Feladatkiírás		
J004	Elgázosított szobában a maszk használódik, csökken a védőképessége	Elgázosított szobába lépni, elgázosítani a szobát, miközben maszk van.	Magas	Feladatkiírás		
J005	Gázos szoba eszméletvesztést okoz, egy időre megbénít.	Gázosított szobába belépés, sajt kibontása.	Magas	Feladatkiírás		Általános esetben. Tárgy hatással lehet.
J006	Több felhasználó játszhatja a játékot	Több játékos kezelésének megfigyelése.	Magas	Feladatkiírás		
SZ001	Ajtót kiválasztva át lehet rajta lépni az adott szobába. általános esetben	A lehetséges ajtót kiválasztva a másik szobába akarunk lépni.	Magas	Feladatkiírás		
SZ002	Használható ajtóknak meg kell jelennie.	Ha aktuális szobának van ajtaja, látható.	Magas	Feladatkiírás		
SZ003	Láthatóak a szobában található tárgyak.	Ha az aktuális szobának van felvehető tárgya, látható.	Magas	Feladatkiírás		
SZ004	Teli szobába nem lehet lépni.	Teli szoba kiválasztása.	Magas	Feladatkiírás		

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
SZ005	Láthatóak a szobában található tárgyak	Aktuális szobában állva, belépve látható a tárgy.	Magas	Feladatkiírás		
SZ006	Szobának van befogadóképessége	Befogadó-képesség nél több karakter beléptetési kísérlete.	Magas	Feladatkiírás		
SZ007	Szoba lehet elgázosított. Ha sajt ki van nyitva.	Gázosított szobában vagyunk, lépünk.	Magas	Feladatkiírás		
SZ008	Elgázosított szobában karakterek elejtik tárgyaikat.	Gázosított szobába lépés, sajt kinyitása.	Magas	Feladatkiírás		
SZ009	Szoba képes egyesülni. Rendelkezik az eredeti szomszédaival, tárgyaival.	Egyesülő szobában állva, észlelhető a bővülés.	Magas	Feladatkiírás		
SZ010	Szoba képes osztódni, szomszédosak és véletlenszerűen osztozik a szomszédain és tárgyaikon.	Osztódó szobában állva, módosulások megfigyelése.	Magas	Feladatkiírás		
SZ011	Egyesült szoba kapacitása a nagyobbikéval egyenlő.	Egyesülő szobában kapacitás megfigyelése.	Magas	Feladatkiírás		
SZ012	Ha vannak a szobában, egyesülés nem mehet végbe.	Szobaegyesítés, mikor a két egyesülő szoba valamelyikében van játékos.	Magas	Csapat		
SZ013	Szoba lehet elátkozott, ekkor lezárul, nincsenek ajtajai.	Elátkozott szobában állva, oda nyíló ajtók, szobában állva ajtójainak megfigyelése.	Magas	Feladatkiírás		
T000	Tárgyak lehetnek a szobákban.	Szobában állva megfigyelendő.	Magas	Feladatkiírás		

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
T001	Tárgy felvehető	Ha van szabad tárgykapacitás, a látható tárgy felvétele.	Magas	Feladatkiírás		
T002	Tárgykapacitás ötös méretű.	5-nél több tárgy felvételére kísérlet.	Magas	Feladatkiírás		
T003	Tárgy lerakható a szobába, szoba része lesz.	Felvett tárgy lerakása. Lerakás után a szoba megfigyelése.	Magas	Feladatkiírás		
T004	Logarléc felvehető, győzelem.	A tárgy felvétele.	Magas	Feladatkiírás		
T005	TVSZ denevérbőrre nyomtatott példánya védelmet ad.	Denevérbőrrel találkozás oktatóval.	Magas	Feladatkiírás		Általános esetben
T006	Denevérbőr elhasználódik oktatóval találkozás esetén.	Háromnál több oktató (alkalom ) meglátogatása denevérbőrrel.	Magas	Feladatkiírás		
T007	Söröspohár védelmet ad oktatók ellen.	Pohár tulajdonában találkozni oktatóval.	Magas	Feladatkiírás		Általános esetben
T008	Söröspohár védelme idővel elhasználódik	Söröspohár-időhatáron túl és előtte oktatóval találkozás.	Magas	Feladatkiírás		
T009	Elhasznált tárgy lerakható.	Elhasznált lerakása, szoba megfigyelése..	Magas	Csapat		
T010	Camember sajt felvételének kísérlete elgázosítást okoz.	Camember felvétele.	Magas	Feladatkiírás		
T011	Tranzisztor ha több van tulajdonban, párosítható.	Kettő vagy több tranzisztor alkalmazása.	Magas	Feladatkiírás		

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
T012	Párosított tranzisztor első része, lerakás után célállomássá válik.	Párosítás után, ebbe érkezünk a második aktiválásánál.	Magas	Feladatkiírás		
T013	Párosított tranzisztor második részének lerakása aktiválja azt, játékos az első rész szobájába kerül.	Párosítás után második tranzisztor elhelyezése, cél megfigyelése.	Magas	Feladatkiírás		
T014	Elhelyezett tranzisztorok használat után újra felvehetőek.	Használt tranzisztor felvétele.	Magas	Csapat		

### 2.3.2 Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
EF001	Fejlesztéshez: Szövegszerkesztésre alkalmas program, számítógép.	Szöveges állomány megnyitása.	Magas	Csapat	
EF002	Fejlesztéshez: Alaprogramok futtatáshoz, teszteléshez.	Programrész futtatása.	Magas	Csapat	Futtatókörnyezet.
EF003	Fejlesztéshez: forrásfájlok megosztását támogató plarform használata.	Közösen készített feladat megosztása.	Magas	Csapat	
EF004	Fejlesztéshez tagok között közvetlen megbeszélések.	Jegyzőkönyvek.	Magas	Csapat	
EF005	Grafikus megjelenítésre alkalmas számítógépes környezet.	Megjelenő képek átnézése.	Közepes	Csapat	
EF006	Hozzáférés a biztosított szerver konfigurációhoz, futtatási környezethez.	Csatlakozási kísérlet a szerverre.	Közepes	Feladatkiírás	<a href="https://niif.cloud.bme.hu/">https://niif.cloud.bme.hu/</a>
EF007	Használat: Program teljesítményigénye nem lehet magasabb a megadott számítógépnél.	Megfelelő felhasználói élmény elvárása testelés alatt.	Magas	Feladatkiírás	
EF008	Használathoz a megfelelő futást és előkészítést, generálást lehetővé kell tenni. A program alapját képező rendszernek elérhetőnek kell lennie.	Program futtatása, előzetes felderítés.	Magas	Feladatkiírás	

### 2.3.3 Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
AT001	Megadott számítógéppel azonos vagy jobb futtatási eszköz. Megfelelő alkatrészek és programok.	Elkészült és előzetes tesztek kísérleti futtatása. Adatok egyeztetése.	Magas	Feladatkiírás	Tesztkönyezet: Hozzáférés 2 magos processzorhoz, 4 GB RAM, 50 GB háttértár. Windows 10 operációs rendszer
AT002	Futtatókörnyezet, megfelelő alaprogramok megléte.	Telepített futtatókörnyezet vizsgálata, teszt program futtatása.	Magas	Feladatkiírás	megfelelő verzió
AT003	Kapcsolat, hozzáférés az adathordozóhoz, a program másolása, átadása céljából.	Külső adat megléte, tesztelése.	Közepes	Csapat	

### 2.3.4 Egyéb nem funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
OT001	A program nem rontja el a futtató eszközt, más funkcióit.	Futtatás utáni vizsgálat	Közepes	Csapat	
OT002	Más tárolt adatot nem módosít, töröl a számítógépen.	Adatok vizsgálata.	Közepes	Csapat	
OT003	A játékprogramnak nem szabad félbeszakadnia a játék vége előtt.	Játék futtatása.	Közepes	Csapat	
OT004	Játékot futtatása közben, funkciók tesztelhetőek.	Program futtatása	Alacsony	Csapat	
OT005	A felhasználónak értelmeznie kell tudni a leírás alapján a játék menetét.	Felhasználó választása.	Közepes	Csapat	
OT006	Felhasználó irányítást tud bevinni a rendszerbe	Felhasználó kiválasztása.	Magas	Csapat	

## 2.4 Lényeges use-case-ek

### 2.4.1 Use-case leírások

<b>Use-case neve</b>	Győzelem a logarléccel
<b>Rövid leírás</b>	A hallgatók a logarléc felvételét követően győznek
<b>Aktorok</b>	<ul style="list-style-type: none"> <li>• Hallgató</li> <li>• Szoba</li> <li>• Tárgy</li> </ul>
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A hallgató belép egy szobába, ahol a logarléc van</li> <li>2. A hallgató felveszi a logarlécet</li> </ol>

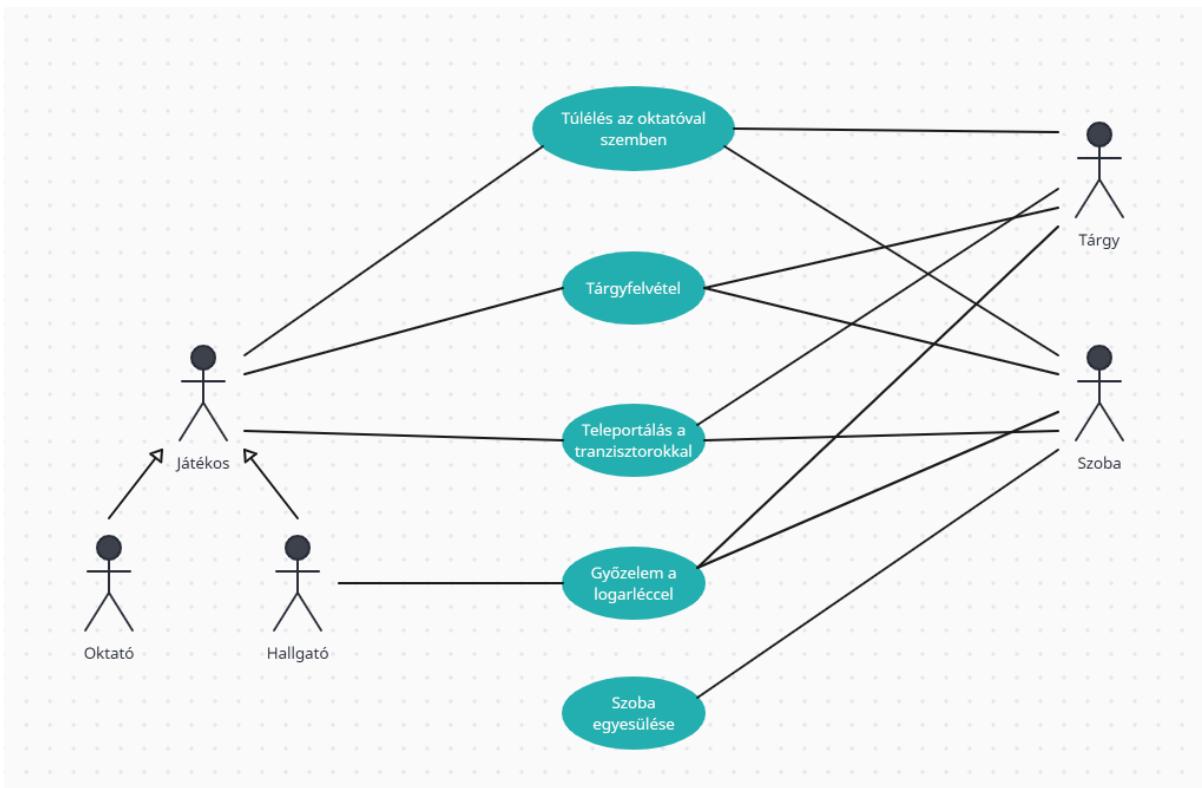
<b>Use-case neve</b>	Túlélés az oktatóval szemben
<b>Rövid leírás</b>	A hallgató találkozik egy oktatóval és túléli
<b>Aktorok</b>	<ul style="list-style-type: none"> <li>• Játékos</li> <li>• Szoba</li> <li>• Tárgy</li> </ul>
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A hallgató belép egy szobába, ahol oktató is van</li> <li>2. A hallgató denevérbőrt vagy söröspoharat használ a védekezéshez</li> </ol>

<b>Use-case neve</b>	Teleportálás a tranzisztorokkal
<b>Rövid leírás</b>	A játékos egyik szobából a másikba teleportál tranzisztorok segítségével
<b>Aktorok</b>	<ul style="list-style-type: none"> <li>• Játékos</li> <li>• Szobák</li> <li>• Tárgyak</li> </ul>
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A játékos összegyűjt két tranzisztorot</li> <li>2. A játékos párosítja a tranzisztorokat</li> <li>3. A játékos lerakja az egyik tranzisztorot</li> <li>4. A játékos lerakja a másik tranzisztorot</li> <li>5. A játékos teleportálódik az elsőnek lerakott tranzisztor szobájába</li> </ol>

<b>Use-case neve</b>	Tárgyfelvétel
<b>Rövid leírás</b>	A játékos felvesz egy szobában lévő tárgyat
<b>Aktorok</b>	<ul style="list-style-type: none"> <li>• Játékos</li> <li>• Szobák</li> <li>• Tárgyak</li> </ul>
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A játékos belép egy szobába</li> <li>2. Megpróbálja felvenni az egyik tárgyat</li> <li>3. Kevesebb mint 5 tárgy van nála</li> <li>4. A tárgy a játékoshoz/oktatóhoz kerül</li> </ol>

<b>Use-case neve</b>	Szoba egyesülése
<b>Rövid leírás</b>	Két szomszédos szoba egyesülése
<b>Aktorok</b>	<ul style="list-style-type: none"> <li>• Szoba</li> </ul>
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. Szomszédos szobák egyesülnek</li> <li>2. Mind a kettőben lévő tárgyak az új szobába kerülnek</li> </ol>

## 2.4.2 Use-case diagram



## 2.5 Szótár

Kifejezés	Magyarázat
Absztrakció	A projekt logikájának fő részei és azok kapcsolata.
GB	A RAM mennyiségeknek mértékegysége.
Github	Verziókezelést elősegítő és megosztó platform.
Github repository	Egy konkrét szoftvert és minden hozzá tartozó dolgot tartalmaz.
Hallgató	A projekt kontextusában a felhasználó által irányított karakter.
Háttértár	A számítógép itt tárolja azokat az adatokat, amikre hosszú távon szükségünk lesz.
Objektum	A valóság mintájára tulajdonságokkal és viselkedésekkel felruházott egység.
Objektum Orientált Programozás	Az objektumok fogalmán alapuló programozási paradigma. Az objektumok egységebe foglalják az adatokat és a hozzájuk tartozó műveleteket.
Oktató	A projekt kontextusában a számítógép által irányított karakter, aki a felhasználó (hallgató) ellensége lesz a játékban.
Operációs Rendszer	A számítógép alkatrészei és a felhasználó

Kifejezés	Magyarázat
	közti kapcsolatot megvalósító szoftver. A mi szoftverünk követelménye, hogy a számítógép a Windows 10 operációs rendszert használja.
Processzor	A számítógép azon egysége, mely az utasításainkat értelmezi és végrehajtásukat vezérli.
Processzormag	Általában a processzoroknak több "magja" van, ezáltal egy időben több számítást tudnak végezni, így hatékonyabbak.
Program/Szoftver	Utasítások sorozata, melyeket a számítógép végrehajt.
Programozási nyelv	A programozási nyelv megadja, hogy milyen formában fogunk utasításokat írni a számítógép számára.
RAM	A számítógép memóriája, a programjaink itt tárolják az ideiglenes adataikat.
Szomszédsági mátrix	Egy mátrix amely sorai és oszlopai a gráf csúcsaival vannak azonosítva. A táblázat egyes cellái pedig a csúcsok közötti élt tárolják.
UML diagram	A szoftver absztrakciójának vizualizálására és tervezésére szolgáló nyelv.
Use Case diagram	Használati eset diagram. Az UML Modellezés egy fajtája.
Windows 10	Microsoft által fejlesztett operációs rendszer.

## 2.6 Projekt terv

- Az ellentmondásmentes fejlesztés érdekében találkozókat tartunk: minden szombaton délután 1-2 óra illetve igény szerint hétfőnként
- A naplózást egy közös, mindenki számára elérhető excel fájlba vezetjük
- A tervezéshez szükséges dokumentumokhoz a Microsoft Google Docs nevű szoftverét használjuk és a különböző részeinek megvalósítását szétosztjuk egymás között
- A forráskód menedzselése, verziókezelése és megosztása a GitHub nevű weboldalon történik
- A dokumentálás nyomtatott formában is elérhető lesz, ezt Rékó Márton intézi
- A szoftver főbb részei és adminisztratív feladatok illetve az azokért felelős személyek listája:
  - Oktatók, grafikus felület, projekt vezető - Mészáros Barna
  - Tárgyak, programozás és verziókezelés- Csiszár Alex
  - Szobák, tesztelés - Rékó Márton
  - Hallgatók, követelményeknek való megfelelés és funkciók megléte - Csirmaz Gáspár
  - Programozó segítő és tanácsadó, modellezés és dokumentáció helyessége - Szabó Nándor
- UML diagramokat, a diagramok típusa illetve az előbb felsorolt részek alapján osztjuk szét, de közös megbeszélés és egyeztetés során végezzük
- A programozásban végrehajtandó részeket GitHub issuekban jegyezzük fel
- A játék logikája és a felhasználói felület két külön egymástól függetlenül kezelhető és módosítható rész
- Projekt megvalósításának lépései:
  - UML diagramok, absztrakciók és részek közötti kapcsolat definiálása
  - Program szkeleton elkészítése
  - Játék logikájának leprogramozása
  - Felhasználói felület megalkotása és összekapcsolása a játék logikával

### 3. Analízis modell kidolgozása

#### 3.1 Objektum katalógus

##### 3.1.1 Szoba

Ez a objektum amiben a játékosok állnak és amik között utaznak. Tárgyat tartalmaz, amiket a karakterek felvehetnek és ajtók nyílnak a szomszédos szobákba. Felelős, hogy számoltartsa a benne lévő karaktereket, és tárgyakat. Ezen kívül szomszédait is tárolnia kell, hogy nyílhasson ajtó beléjük.

Esetenként, amikor az állapota megváltozik, elgázosodik, elátkozottá válik, kezelni kell tudja ezeknek hatásait, esetenként a benne lévő szereplőket értesíteni ezen eseményekről.

Eseménynek számíthat, ha más is megérkezik a szobába, erről is értesíti a benne lévőket. Van egy maximális létszáma is minden szobának, aminél több karakter nem lehet az adott helyiségen.

##### 3.1.2 Hallgató

A játékosok által irányított karakter, lehet vele a szobák között mozogni, a tárgykapacitásáig tárgyat felvenni, azokat használni és lerakni. A tulajdonolt tárgyait számon kell hogy tudja tartani.

Hallgató több állapotban is lehet. Normál állapotában nincs akadályozva, a fent felsorolt akciókat minden tudja csinálni. Halott állapotában számára a játék véget ért, nem képes interakciókra. Kábított állapotában egy időre el van tiltva minden interakciótól, de az adott idő után újra irányíthatóvá, normális állapotúvá válik. Mindezen állapotváltoztatást az aktuális szobájának állapotára adott reakcióból viszi végbe.

##### 3.1.3 Tranzisztor

Egy tárgy, melyet használatkor először párosítunk egy másik tranzisztorral. A párosításhoz kettő tranzisztor kell, hogy legyen nálunk. Párosítás után használat parancsra a használt tranzisztor a szobába kerül (eldobjuk a jelenlegi szobába). Ha a nálunk maradt tranzisztor ezután használjuk, akkor az eldobott tranzisztor szobájába kerülünk. Ez után a tranzisztorokat újra párosítani kell majd, ha megint használni akarjuk őket.

##### 3.1.4 TVSZ denevérbőr

Egy tárgy, amely automatikusan képes megvédeni a hallgatót az oktató támadásától. A TVSZ eltárolja, hogy hányszor tudja még megvédeni a hallgatót. Ha a TVSZ nem tudja megvédeni a hallgatót, akkor a hallgató meghal.

##### 3.1.5 Szent söröspohár

A Szent söröspoharat először használni (aktiválni) kell, ezután a TVSZ-hez hasonlóan automatikus védelmet nyújt az oktatók ellen. A söröspohár eltárolja, hogy még mennyi ideig tudja megvédeni a hallgatót. Ha a söröspohárnak nem sikerül megvédeni a hallgatót, a hallgató meghal. A söröspohár a szobában tartózkodó összes hallgatót képes egyszerre megvédeni.

##### 3.1.6 Káposztás camembert

Egyszer használható tárgy. Használatra a felhasználó szobája gázossá válik. Azt, hogy már használt-e a Káposztás camembert, önmaga tárolja.

### **3.1.7 Oktató**

Hasonlít a hallgatókhoz, viszont nem tud tárgyakat használni:

Tud a szobák között mozogni, a tárgykapacitásáig tárgyakat felvenni, azokat lerakni. A tulajdonolt tárgyait számon kell hogy tudja tartani.

Oktató több állapotban is lehet. Normál állapotában nincs akadályozva, a fent felsorolt akciókat minden tudja csinálni. Halott állapotában számára a játék véget ért, nem képes interakciókra. Kábított állapotában egy időre el van tiltva minden interakciótól, de az adott idő után újra irányíthatóvá, normális állapotúvá válik. Mindezen állapotváltoztatást az aktuális szobájának állapotára adott reakcióból viszi végbe.

### **3.1.8 Maszk**

Olyan tárgy, amely tulajdonosát automatikusan megvédi a gázos szobák hatásától, ha még nem használódott el. A Maszk eltárolja, hogy mennyi ideig képes még megvédeni egy játékost. Ha lejárt az ideje, és a játékos belép egy gázos szobába, akkor eszméletvesztett állapotba kerül a játékos. A Maszk ideje csak gázos szobán belül tartózkodás esetén fogy.

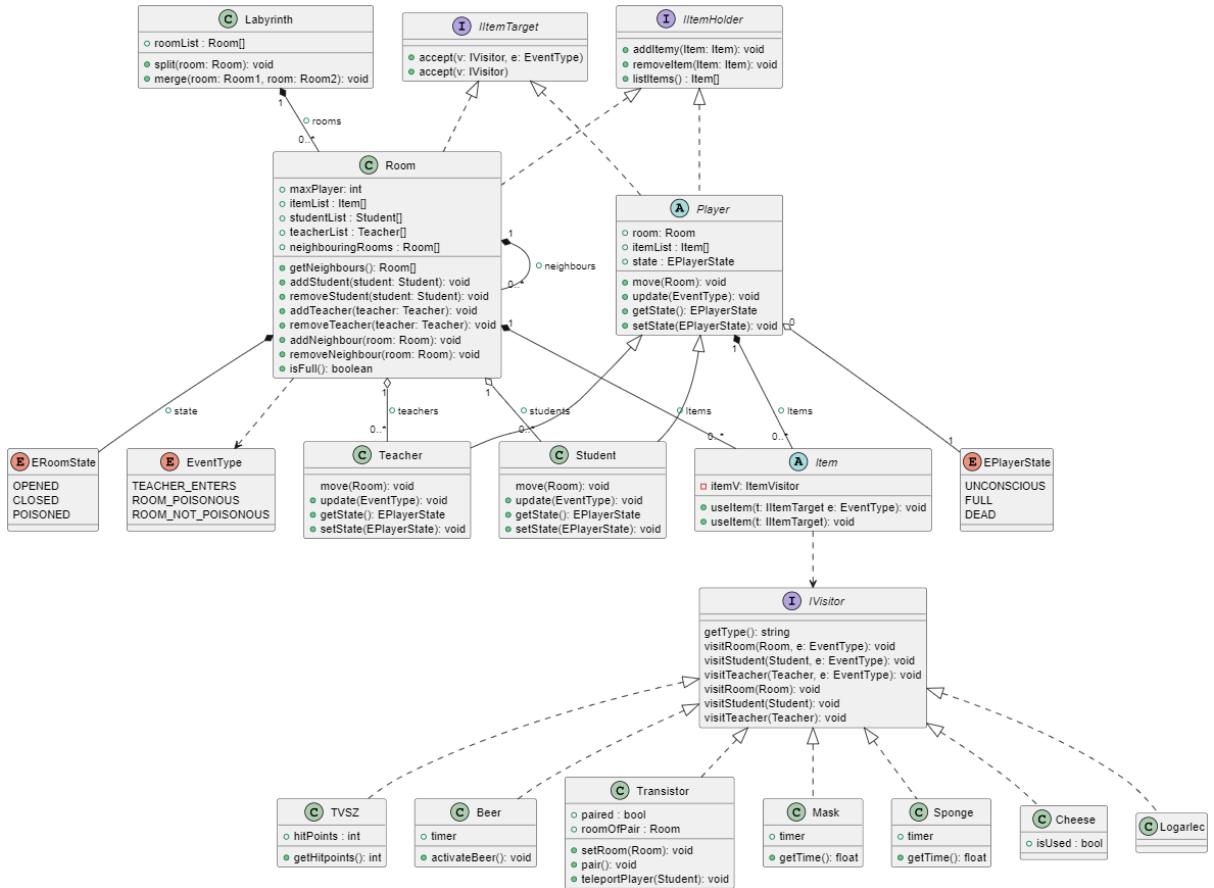
### **3.1.9 Nedves táblatörölő rongy**

Tárgy, felvétel után elkezd egyből elhasználódni. Amíg nem használódott el teljesen, addig képes automatikusan lebénítani minden olyan oktatót, aki a rongyat hordozó hallgatóval van egy szobában. Azt, hogy mennyi ideje van még hátra, a rongy saját maga tárolja el.

### **3.1.10 Logarléc**

Különleges tárgy, miután elhasználja egy hallgató, a játék győzelemmel véget ér.

### **3.2 Statikus struktúra diagramok**



### 3.3 Osztályok leírása

### 3.3.1 Labyrinth

- **Felelősség**
    - Nyilvántartja az összes szobát, képes a szobákat osztani és egyesíteni.
  - **Attribútumok**
    - **roomList**: A játékban aktív összes szoba listája
  - **Metódusok**
    - **void split(Room r)**: Az adott szobát megpróbálja elosztani két szobára. Az osztódást követően az egyik szoba megtartja összes szomszédját, a másiknak pedig egyetlen egy szomszédja lesz (az a szoba amiből osztódott). Az első szobában lévő tárgyak és tulajdonságok valamelyen algoritmus alapján szétosztódnak a két szoba között. Ha játékos tartózkodik a szobában, nem lesz split.
    - **void merge(Room r1, Room r2)**: Két adott szobát egyesít. Az új szoba a korábbi két szoba tulajdonságaival, tárgyaival és szomszédaival rendelkezik. Az új szoba befogadóképessége azonos lesz a nagyobbik szoba befogadóképességével. Ha játékos tartózkodik bármelyik szobában a kettő közül, nem lesz merge.

### 3.3.2 Room

- **Felelősség**

Eltárolja a játékosokat, szomszédait és az itemeket, majd értesíti a játékosokat állapotváltozáskor

- **Interfészek**

IItemHolder, IItemTarget

- **Attribútumok**

- **maxPlayer**
- **studentList**
- **teacherList**
- **neighbouringRooms**
- **itemList**

- **Metódusok**

- **Room[] getNeighbours()** : visszaadja a szomszédait egy listában
- **Student[] getStudents()** : visszaadja a benne lévő hallgatókat egy listában
- **Teacher[] getTeachers()** : visszaadja a benne lévő oktatókat egy listában
- **void addStudent(Student)** : Egy paraméterként kapott hallgatót belerak a listájába. Akkor kell használni ha belép a szobába egy hallgató
- **void removeStudent(Student)** : Egy paraméterként kapott hallgatót kivesz a listából. Akkor kell használni ha egy hallgató elhagyja a szobát
- **void addTeacher(Teacher)** : Egy paraméterként kapott oktatót belerak a listájába. Akkor kell használni ha belép a szobába egy oktató
- **void removeTeacher(Teacher)** : Egy paraméterként kapott oktatót kivesz a listából. Akkor kell használni ha egy oktató elhagyja a szobát
- **void addNeighbour(Room)** : Egy paraméterként kapott szobát belerak a szomszédsági listájába. Akkor kell használni ha egy split-el szoba
- **void removeNeighbour(Room)** : Egy paraméterként kapott szobát kivesz a szomszédsági listából. Akkor kell használni ha egy merge-el szoba
- **void addItem(item)** : Egy paraméterként kapott itemet belerak a listájába. Akkor kell használni ha valaki eldob egy tárgyat egy szobában
- **void removeItem(Item)** : Egy paraméterként kapott itemet kivesz a listából. Akkor kell használni ha valaki felvesz egy tárgyat
- **boolean isFull()** : Ha a maxPlayer értéke elér

### 3.3.3 IItemTarget

- **Felelősség**

Interface, az összes olyan osztály implementálja, melyen tárgyat lehet használni. Visitor pattern megvalósításához kell.

- **Metódusok**

- **void accept(IVisitor v, EventType e)**: Visitor pattern alapján ennek a segítségével használhatjuk a megfelelő célponton (szoba/hallgató/oktató). Egy EventType-ot is átadunk, ezzel megmondhatjuk, hogy milyen fajta event hatására lett az item (automatikusan) elhasználva.

- **void accept(IVisitor v)**: Itt egy játékos önként használ el egy itemet.

### 3.3.4 IItemHolder

- **Felelősség**  
Interface, az összes olyan osztály implementálja, mely tárgyakat fog tárolni.
- **Metódusok**
  - **void addItem(Item i)**: Az adott itemet megpróbáljuk hozzáadni a tárgyaink listájához.  
Ha nincs több kapacitás, akkor semmi se fog történni.
  - **void removeItem(Item i)**: Az adott itemet megpróbáljuk kitörölni az itemeink listájáról. Ha nincs ilyen item, nem történik semmi.
  - **Item[] listItems()**: Kilistázza az ezt megvalósító classok itemlistáinak itemeit

### 3.3.5 Player

- **Felelősség**  
Absztrakt osztály, az összes játékos leszármazik tőle (jelen esetben oktató/hallgató).  
Általános tulajdonságokat határoz meg, amikkel minden játékosnak rendelkeznie kell.
- **Interfészek**  
IItemHolder (tud tárgyakat tárolni), IItemTarget (lehet itemeket használni rajta)
- **Attribútumok**
  - **room** - A jelenlegi szobát tárolja
  - **itemList** - Az itemek listáját tárolja
  - **state** - A játékos állapotát tárolja
- **Metódusok**
  - **void move(Room r)**: A játékos megpróbál lépni az r szobába. Ha nem tud, nem történik semmi.
  - **void update(EventType e)**: A játékos reagál a kapott eventre.
  - **EPlayerState getState()**: Lekérhetjük a játékos jelenlegi állapotát.
  - **void setState(EPlayerState s)**: Beállíthatjuk a játékos állapotát.

### 3.3.6 Teacher

- **Felelősség**

Osztály, ami megvalósítja az Oktató működését.

- **Ősosztályok**

Player

- **Interfészek**

IItemHolder, IItemTarget

- **Metódusok**

- **void move(Room r)**: A játékos megpróbál lépni az r szobába. Ha nem tud, nem történik semmi.
- **void update(EventType e)**: A játékos reagál a kapott eventre.
- **EPlayerState getState()**: Lekérhetjük a játékos jelenlegi állapotát.
- **void setState(EPlayerState s)**: Beállíthatjuk a játékos állapotát.

### 3.3.7 Student

- **Felelősség**

A Hallgató működését megvalósító osztály. Ezzel kommunikál a felhasznál, meghívhatja a move függvényét, ezzel mozgatva, vagy a tárgyait lekérdezheti és használhat egyet

- **Ősosztályok**

Player

- **Interfészek**

IItemHolder, IItemTarget

- **Metódusok**

- **void move(Room r)**: A játékos megpróbál lépni az r szobába. Ha nem tud, nem történik semmi.
- **void update(EventType e)**: A játékos reagál a kapott eventre.
- **EPlayerState getState()**: Lekérhetjük a játékos jelenlegi állapotát.
- **void setState(EPlayerState s)**: Beállíthatjuk a játékos állapotát.

### 3.3.8 Item

- **Felelősség**

Absztrakt osztály, különböző tárgyak használatát segíti elő. Ha egy Player használni akar egy tárgyat, akkor az Item továbbadja a kérelmet a konkrét tárgynak.

- **Attribútumok**

**IVisitor:** Az IVisitor interface segítségével tudjuk elérni a helyes funkcionálitást a tárgyakhoz.

- **Metódusok**

- **void useItem(IItemTarget t, EventType e):** Tárgy használata, egy esemény hatására. A használt tárgy megkapja az eseményt, és azt, hogy min lesz használva a tárgy.
- **void useItem(IItemTarget t):** Event nélküli tárgyhasználat egy célponton.

### 3.3.9 IVisitor

- **Felelősség**

Visitor pattern alapján segít megteremteni a tárgyhasználatot. minden tárgunknak ezt implementálnia kell.

- **Metódusok**

- **string getType():** A tárgy nevével tér vissza, ez akkor lesz hasznos, ha ki kell iratnunk, hogy milyen tárgyak vannak egy játékosnál.
- **void visitRoom(Room r, EventType e):** A tárgy használata egy szobán, e event hatására.
- **void visitStudent(Student s, EventType e):** A tárgy használata egy hallgatón, e event hatására.
- **void visitTeacher(Teacher t, EventType e):** A tárgy használata egy oktatón, e event hatására.
- **void visitRoom(Room r):** A tárgy használata r szobára.
- **void visitStudent(Student s):** A tárgy használata s hallgatóra.
- **void visitTeacher(Teacher t):** A tárgy használata t oktatóra

### 3.3.10 TVSZ

- **Felelősség**

A TVSZ denevérbőr tárgy hatásáért felelős.

- **Interfészek**

- IVisitor

- **Attribútumok**

- hitpoints : int
- **Metódusok**
  - **string getType():** A tárgy nevével tér vissza.
  - **void visitRoom(Room r, EventType e):** Nincs ilyen hatása, üres.
  - **void visitStudent(Student s, EventType e):** Ha egy oktató belép a szobába, akkor a TVSZ automatikusan megvédi a hallgatót, akinél a TVSZ van. Ha a TVSZ hitpoints-ja 0, akkor a hallgató meghal.
  - **void visitTeacher(Teacher t, EventType e):** Nincs ilyen hatása, üres.
  - **void visitRoom(Room r):** Nincs ilyen hatása, üres.
  - **void visitStudent(Student s):** Nincs ilyen hatása, üres.
  - **void visitTeacher(Teacher t):** Nincs ilyen hatása, üres.
  - **int getHitPoints():** Visszatér a tárgy hitpoints-ával.

### 3.3.11 Beer

- **Felelősség**

A Szent söröspohár tárgy hatásáért felelős.
- **Interfészek**
  - IVisitor
- **Attribútumok**
  - timer
- **Metódusok**
  - **string getType():** A tárgy nevével tér vissza.
  - **void visitRoom(Room r, EventType e):** Nincs ilyen hatása, üres.
  - **void visitStudent(Student s, EventType e):** Oktatóval találkozik a hallgató, a szobában lévő összes hallgatót megvédi a sör (nem halnak meg). Ha lejárt a timer, meghalnak a hallgatók.
  - **void visitTeacher(Teacher t, EventType e):** Nincs ilyen hatása, üres.
  - **void visitRoom(Room r):** Nincs ilyen hatása, üres.
  - **void visitStudent(Student s):** A hallgató használja a sört, elindul a timer.

- **void visitTeacher(Teacher t):** Nincs ilyen hatása, üres.
- **void activateBeer():** Elindítjuk a sör számlálóját, innentől tud megvédeni minket.

### 3.3.12 Transistor

- **Felelősség**

A Tranzisztor tárgy hatásáért felelős.

- **Interfészek**

- IVisitor

- **Attribútumok**

- paired : bool
- roomOfPair : Room

- **Metódusok**

- **string getType():** A tárgy nevével tér vissza.
- **void visitRoom(Room r, EventType e):** Nincs ilyen hatása, üres.
- **void visitStudent(Student s, EventType e):** Nincs ilyen hatása, üres.
- **void visitTeacher(Teacher t, EventType e):** Nincs ilyen hatása, üres.
- **void visitRoom(Room r):** Nincs ilyen hatása, üres.
- **void visitStudent(Student s):** A hallgató használja a tranzisztort, a tranzisztor belső állapota szerint fognak történni dolgok: vagy párosul két tranzisztor, vagy leteszünk egyet, vagy teleportálunk.
- **void visitTeacher(Teacher t):** Nincs ilyen hatása, üres.
- **void setRoom(Room r):** Elmentjük a tranzisztorpár szobáját
- **void pair():** Párosít két tranzisztort.
- **void teleportPlayer():** A tranzisztorpárhoz teleportáljuk a felhasználót.

### 3.3.13 Mask

- **Felelősség**

A FFP2 Maszk tárgy hatásáért felelős.

- **Interfészek**
  - IVisitor
- **Attribútumok**
  - timer
- **Metódusok**
  - **string getType():** A tárgy nevével tér vissza.
  - **void visitRoom(Room r, EventType e):** Nincs ilyen hatása, üres.
  - **void visitStudent(Student s, EventType e):** Ha egy gázos szobába lép a hallgató, és még van ideje a maszknak, akkor a hallgató meg lesz védve, különben a hallgató elájul.
  - **void visitTeacher(Teacher t, EventType e):** Nincs ilyen hatása, üres.
  - **void visitRoom(Room r):** Nincs ilyen hatása, üres.
  - **void visitStudent(Student s):** Nincs ilyen hatása, üres.
  - **void visitTeacher(Teacher t):** Nincs ilyen hatása, üres.
  - **float getTime():** Visszatér a tárgy maradék idejével.

### 3.3.14      **Sponge**

- **Felelősség**

A Nedves táblatörlő rongy tárgy hatásáért felelős.
- **Interfész**

IVisitor
- **Attribútumok**
  - timer
- **Metódusok**
  - **string getType():** A tárgy nevével tér vissza.
  - **void visitRoom(Room r, EventType e):** Nincs ilyen hatása, üres.
  - **void visitStudent(Student s, EventType e):** Nincs ilyen hatása, üres.

- **void visitTeacher(Teacher t, EventType e):** Ha egy oktatóval találkozik valaki, és elhasználódik a szivacs, akkor az oktató, és az oktatóval egy szobában tartózkodó oktatók lebénulnak.
- **void visitRoom(Room r):** Nincs ilyen hatása, üres.
- **void visitStudent(Student s):** Nincs ilyen hatása, üres.
- **void visitTeacher(Teacher t):** Nincs ilyen hatása, üres.
- **float getTime():** Visszatér a tárgy maradék idejével.

### 3.3.15 Cheese

- **Felelősség**

A Káposztás camembert tárgy hatásáért felelős.

- **Interfészek**

IVisitor

- **Attribútumok**

- isUsed : bool

- **Metódusok**

- **string getType():** A tárgy nevével tér vissza.
- **void visitRoom(Room r, EventType e):** Nincs ilyen hatása, üres.
- **void visitStudent(Student s, EventType e):** Nincs ilyen hatása, üres.
- **void visitTeacher(Teacher t, EventType e):** Nincs ilyen hatása, üres.
- **void visitRoom(Room r):** A szoba gázos lesz.
- **void visitStudent(Student s):** Nincs ilyen hatása, üres.
- **void visitTeacher(Teacher t):** Nincs ilyen hatása, üres.

### 3.3.16 Logarléc

- **Felelősség**

A logarléc tárgy hatásáért felelős.

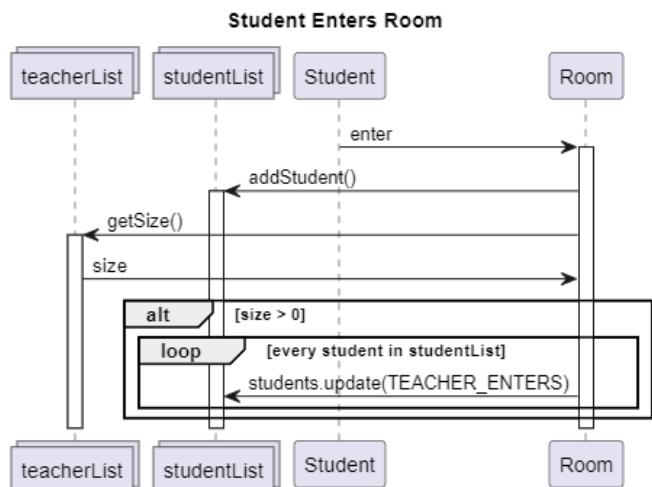
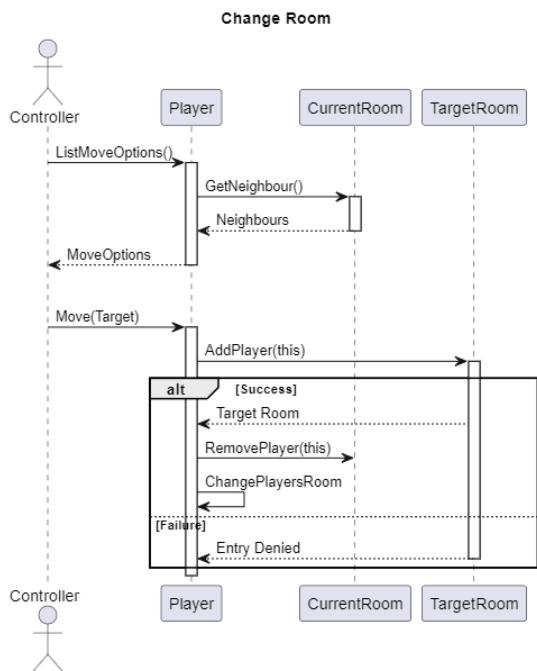
- **Interfészek**

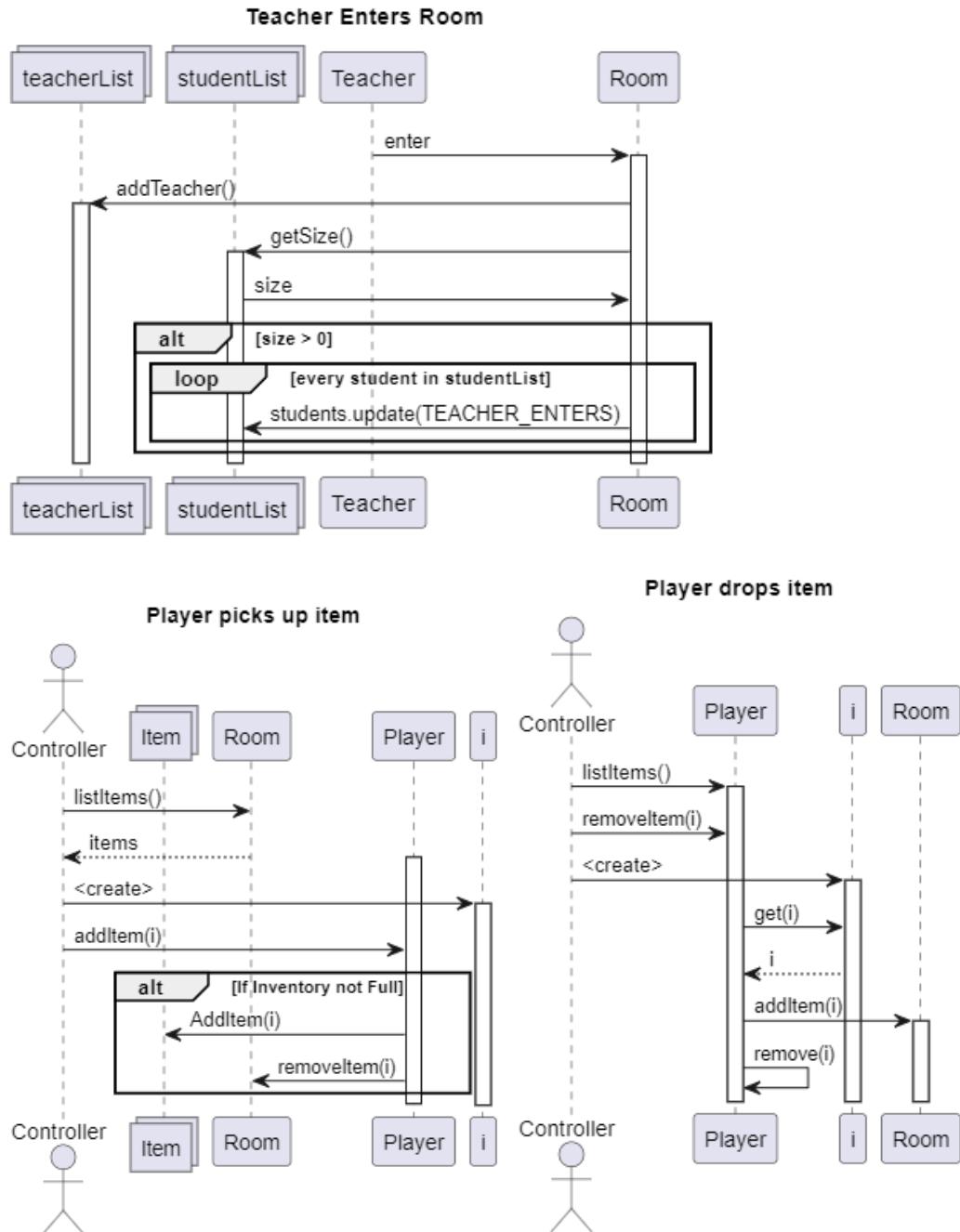
## IVisitor

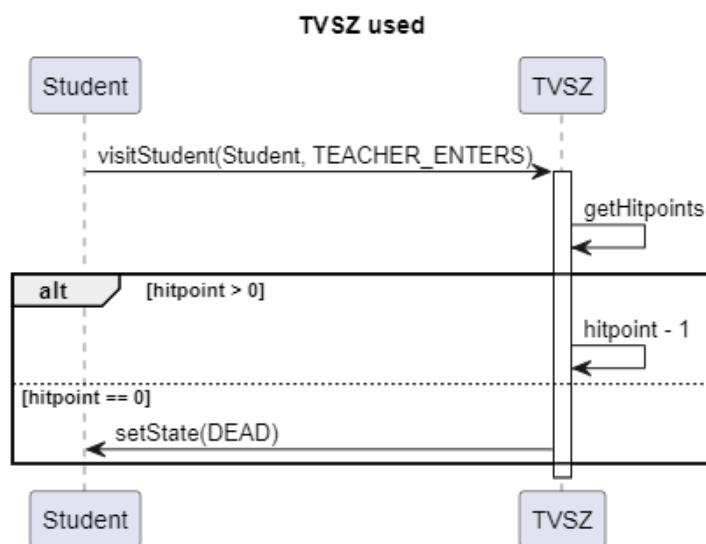
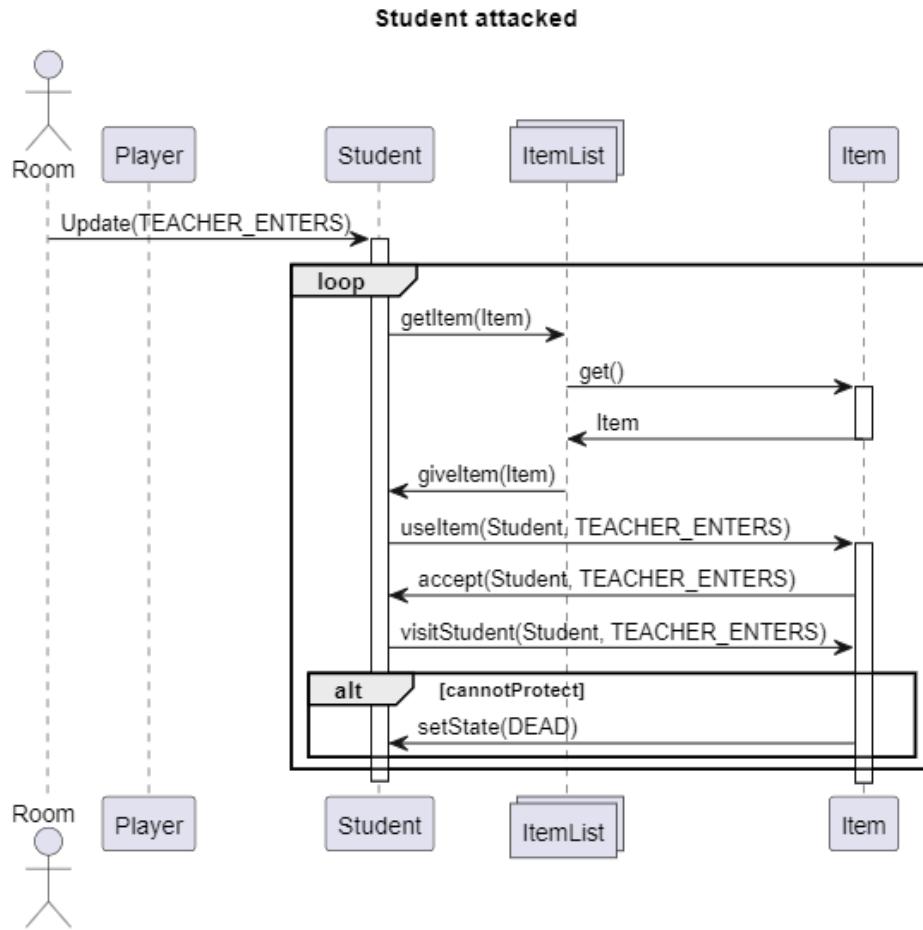
• **Metódusok**

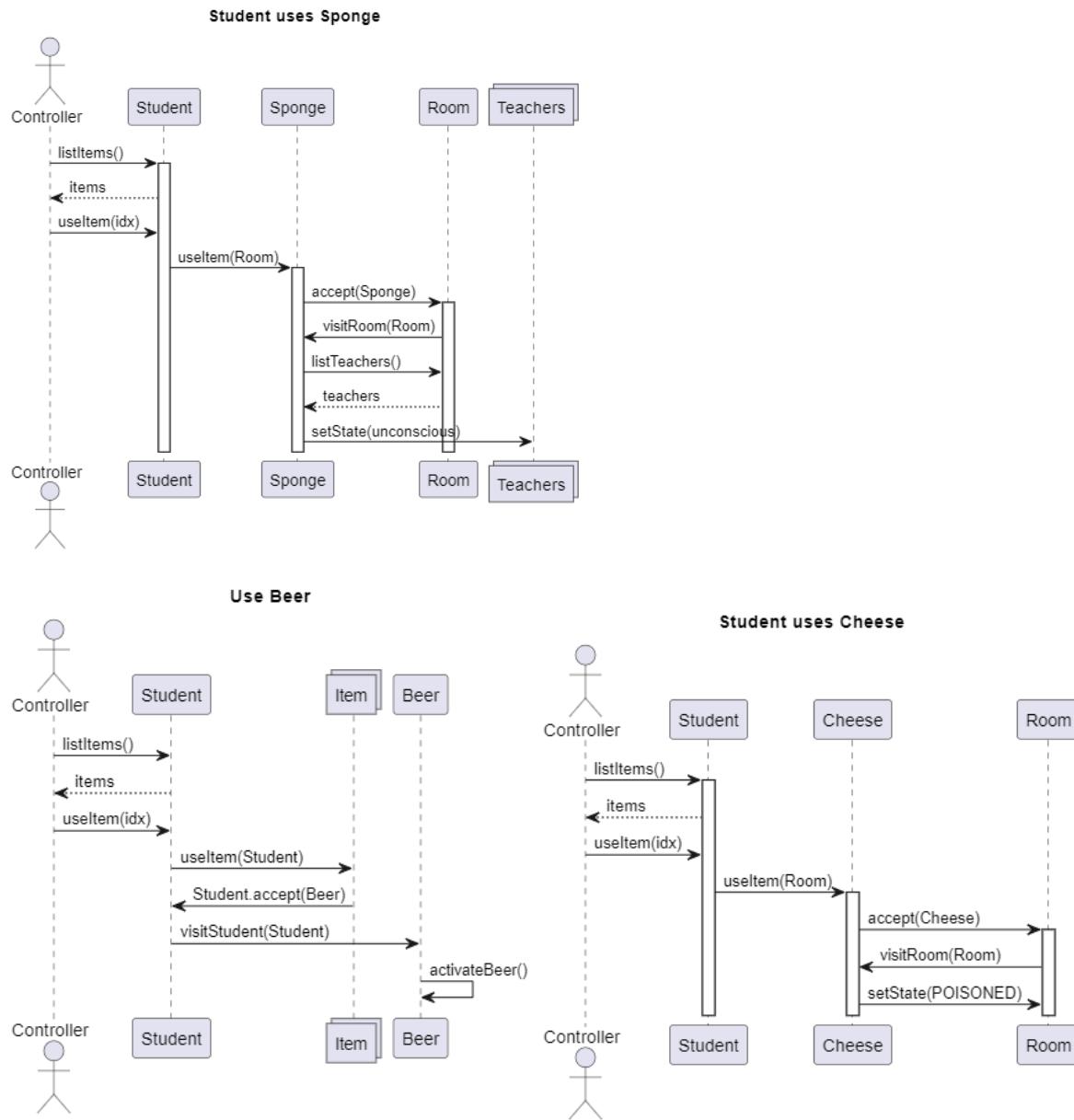
- **string getType():** A tárgy nevével tér vissza.
- **void visitRoom(Room r, EventType e):** Nincs ilyen hatása, üres.
- **void visitStudent(Student s, EventType e):** Nincs ilyen hatása, üres.
- **void visitTeacher(Teacher t, EventType e):** Nincs ilyen hatása, üres.
- **void visitRoom(Room r):** Nincs ilyen hatása, üres..
- **void visitStudent(Student s):** A tárgy használata s hallgatóra, a játéknak vége, győztek a hallgatók.
- **void visitTeacher(Teacher t):** Nincs ilyen hatása, üres.

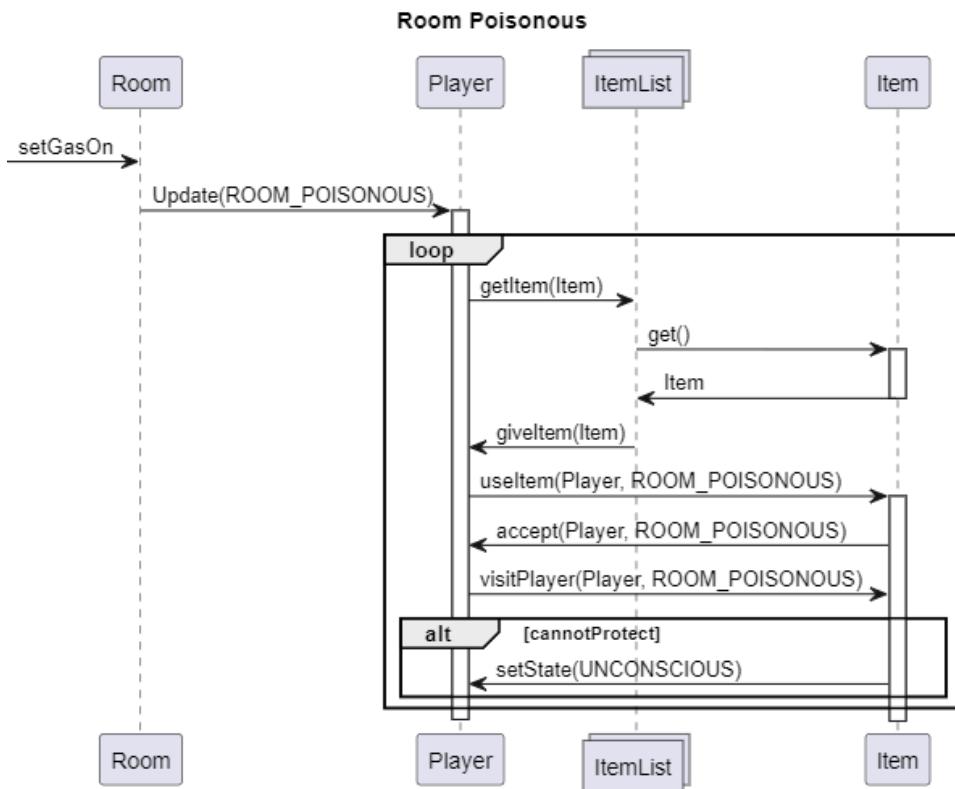
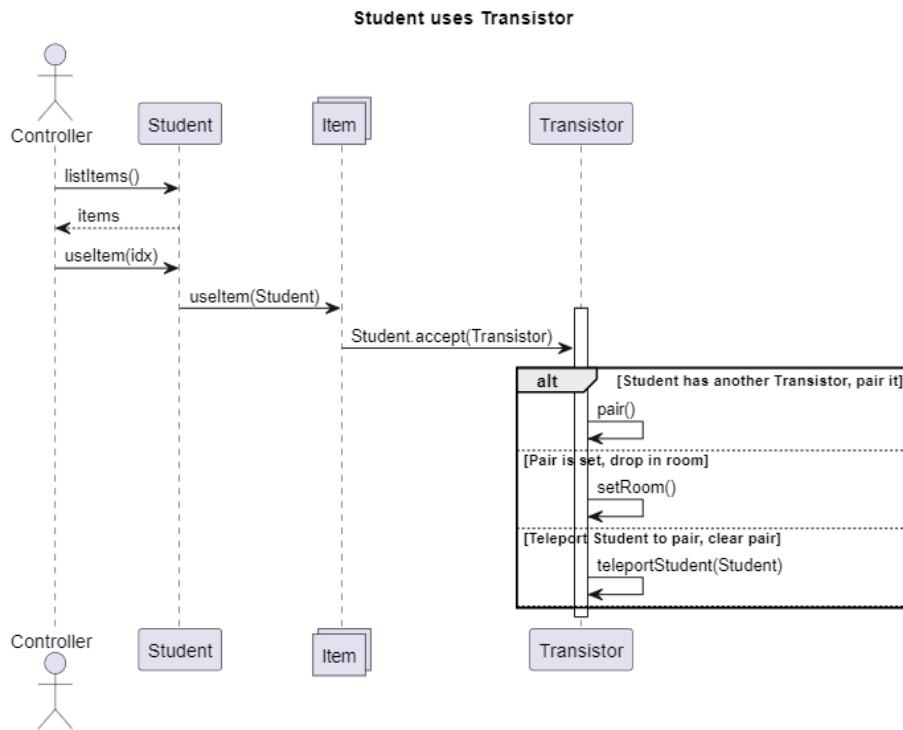
### 3.4 Szekvencia diagramok

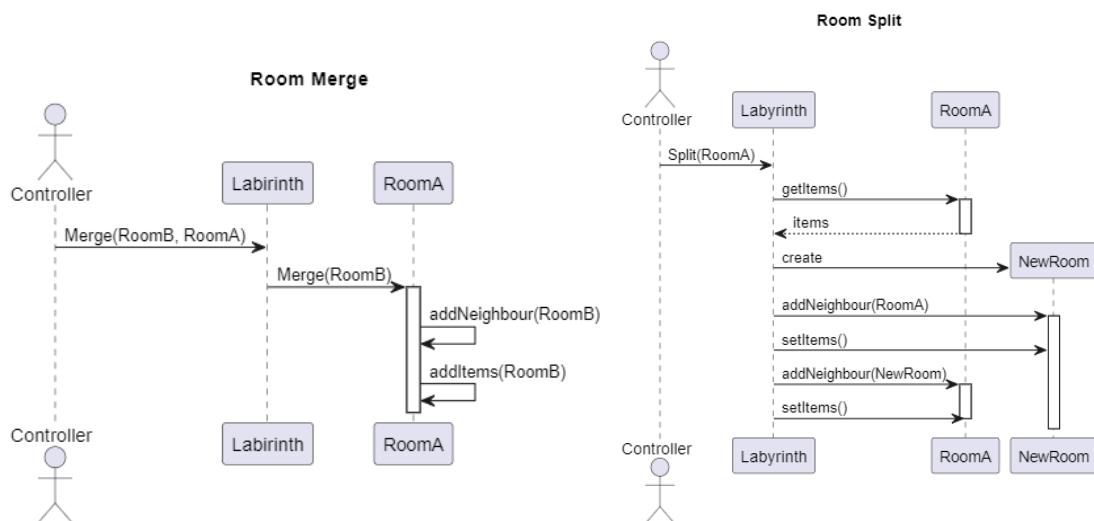
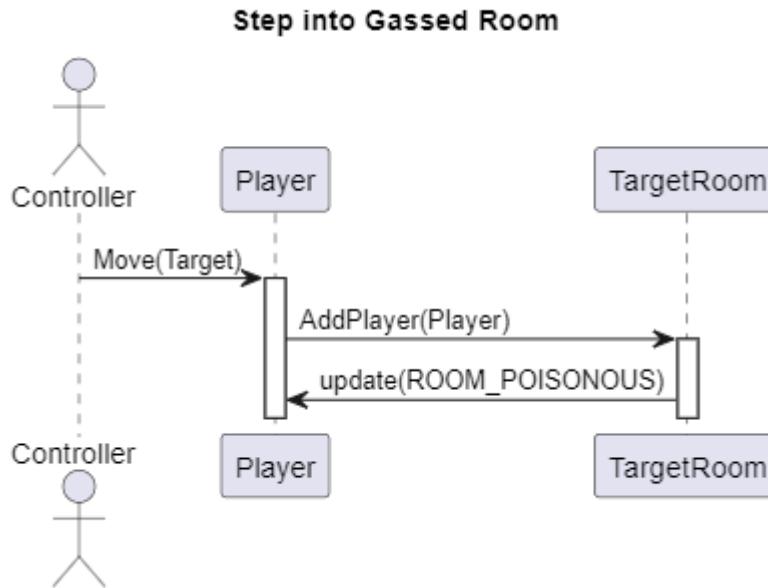






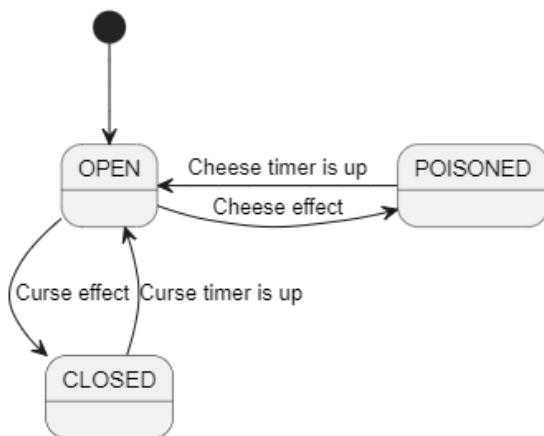






### 3.5 State-chartok

Room State



## 4. Analízis modell kidolgozása

### 4.1 Objektum katalógus

#### 4.1.1 Szoba

Ez a objektum amiben a játékosok állnak és amik között utaznak. Tárgyat tartalmaz, amiket a játékosok felvehetnek és ajtóból nyílnak a szomszédos szobákba. Felelős, hogy számon tartja a benne lévő karaktereket, és tárgyakat. Ezen kívül szomszédait is tárolnia kell, hogy nyílhasson ajtó beljük.

Esetenként, amikor az állapota megváltozik, elgázosodik, elátkozottá válik, kezelni kell tudja ezeknek hatásait, esetenként a benne lévő szereplőket értesíteni ezen eseményekről.

Eseménynek számíthat, ha más is megérkezik a szobába, erről is értesíti a benne lévőket. Van egy maximális létszáma is minden szobának, aminél több karakter nem lehet az adott helyiségen.

Képes lekezelni az osztódást és szétválaszt, meghatározni végrehajtható-e majd végrehajtani

#### 4.1.2 Hallgató

A játékosok által irányított karakter, lehet vele a szobák között mozogni, a tárgykapacitásig tárgyat felvenni, azokat használni és lerakni. A tulajdon tárgyait számon kell hosszúra tartani.

Hallgató több állapotban is lehet. Normál állapotában nincs akadályozva, a fent felsorolt akciókat mindenkorban tudja elvégezni. Halott állapotában számára a játék véget ért, nem képes interakciókra. Kábított állapotában egy időre el van tiltva minden interakciótól, de az adott idő után újra irányíthatóvá, normális állapotúvá válik. Mindezen állapotváltoztatást az aktuális szobájának állapotára adott reakcióból viszi végbe. Képes a tárgyainak üzenetet küldeni, hogy mentsék meg

#### 4.1.3 Tranzisztor

Egy tárgy, melyet használatkor először párosítunk egy másik tranzisztorral. A párosításhoz kettő tranzisztor kell, hogy legyen nálunk. Párosítás után használat parancsra a használt tranzisztor a szobába kerül (eldobjuk a jelenlegi szobába). Ha a nálunk maradt tranzisztor ezután használjuk, akkor az eldobott tranzisztor szobájába kerülünk. Ez után a tranzisztorokat újra párosítani kell majd, ha megint használni akarjuk őket.

Képes a szomszédját párosítani magával, beállítani a saját szobáját és teleportálni a játékost

#### 4.1.4 TVSZ denevérbőr

Egy tárgy, amely automatikusan képes megvédeni a hallgatót az oktató támadásától. A TVSZ eltárolja, hogy hányszor tudja még megvédeni a hallgatót. Ha a TVSZ nem tudja megvédeni a hallgatót, akkor a hallgató meghal.

#### 4.1.5 Szent söröspohár

A Szent söröspoharat először használni (aktiválni) kell, ezután a TVSZ-hez hasonlóan automatikus védelmet nyújt az oktatók ellen. A söröspohár eltárolja, hogy még mennyi ideig tudja megvédeni a hallgatót. Ha a söröspohárnak nem sikerül megvédeni a hallgatót, a hallgató meghal.

#### **4.1.6 Káposztás camembert**

Egyszer használható tárgy. Használatra a felhasználó szobája gázossá válik. Azt, hogy már használt-e a Káposztás camembert, önmaga tárolja. Illetve szól a szobának ha lejárt a gázossága

#### **4.1.7 Oktató**

Hasonlít a hallgatókhoz, viszont nem tud tárgyakat használni:

Tud a szobák között mozogni, a tárgykapacitásáig tárgyakat felvenni, azokat lerakni. A tulajdonolt tárgyait számon kell hogy tudja tartani.

Oktató több állapotban is lehet. Normál állapotában nincs akadályozva, a fent felsorolt akciókat minden tudja csinálni. Kábított állapotában egy időre el van tiltva minden interakciótól, de az adott idő után újra irányíthatóvá, normális állapotúvá válik. Mindezen állapotváltoztatást az aktuális szobájának állapotára adott reakcióból viszi végbe. Halott állapotban jelenleg nem lehet (bár képes lenni).

#### **4.1.8 Maszk**

Olyan tárgy, amely tulajdonosát automatikusan megvédi a gázos szobák hatásától, ha még nem használódott el. A Maszk eltárolja, hogy mennyi ideig képes még megvédeni egy játékost. Ha lejárt az ideje, és a játékos belép egy gázos szobába, akkor eszméletvesztett állapotba kerül a játékos. A Maszk ideje csak gázos szobán belül tartózkodás esetén fogyni.

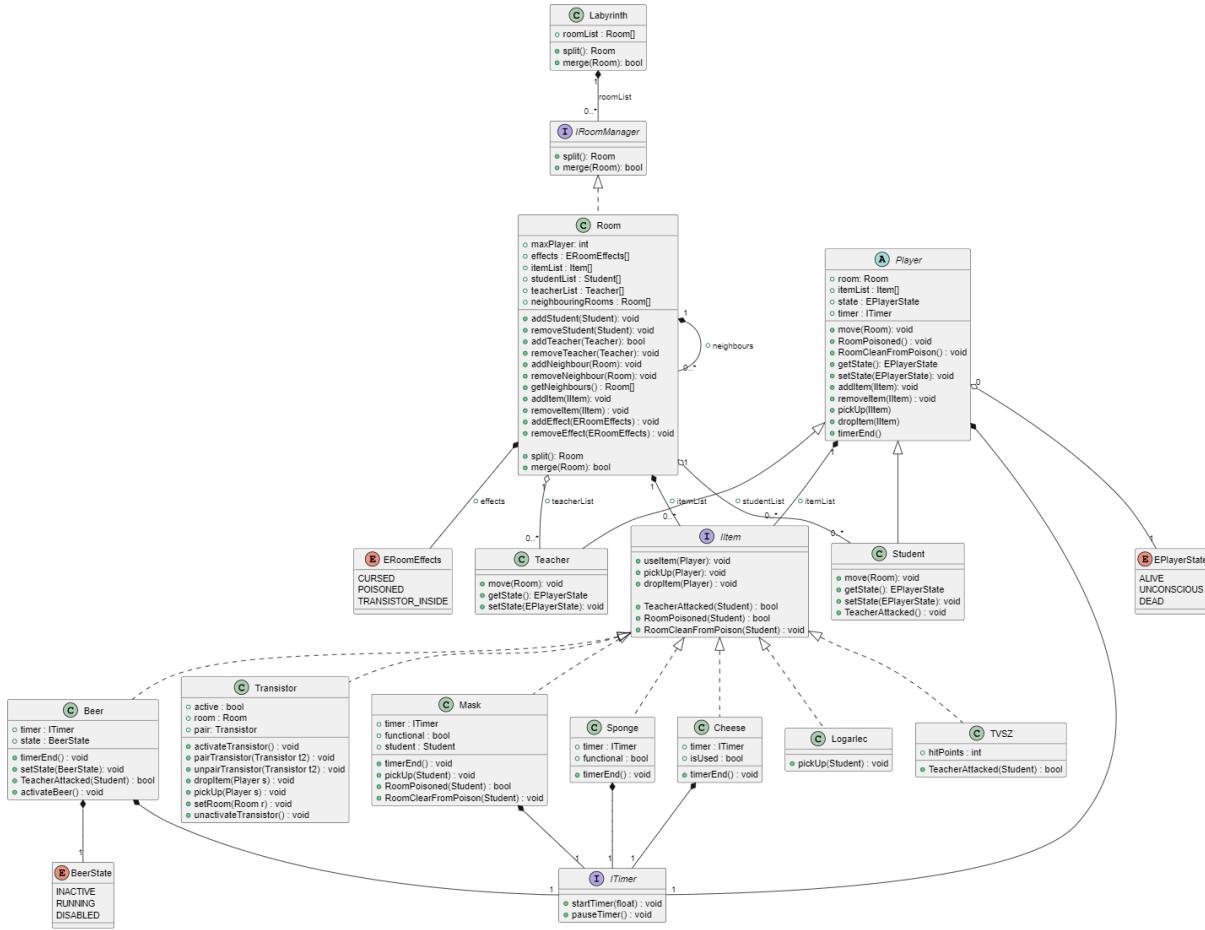
#### **4.1.9 Nedves táblatörlő rongy**

Tárgyfelvétel után elkezd egyből elhasználódni. Amíg nem használódott el teljesen, addig képes automatikusan lebénítani minden olyan oktatót, aki a rongyat hordozó hallgatóval van egy szobában. Azt, hogy mennyi ideje van még hátra, a rongy saját maga tárolja el.

#### **4.1.10 Logarléc**

Különleges tárgy, miután felveszi egy hallgató, a játék győzelemmel véget ér. Képes felvételkor üzenni a kontrollernek, hogy vége a játéknak

## 4.2 Statikus struktúra diagramok



## 4.3 Osztályok leírása

### 4.3.1 Labyrinth

- **Felelősség**

Nyilvántartja az összes szobát, képes a szobákat osztani és egyesíteni.

- **Attribútumok**

- **roomList:** A játékban aktív összes szoba listája

- **Metódusok**

- **void split(Room r):** Meghívja a labirintus adott szobájának split metódusát.
- **void merge(Room r1, Room r2):** Meghívja a labirintus adott szobájának merge metódusát

### 4.3.2 Room

- **Felelősség**

Eltárolja a játékosokat, kezeli a szobák összevonását és szétosztását, tárolja szomszédait és az itemeket, majd értesíti a játékosokat állapotváltozáskor. Számon tartja a rajta lévő hatásokat és menedzseli őket.

- **Interfészek**

IRoomManager

- **Attribútumok**

- **maxPlayer**
- **effects**
- **studentList**
- **teacherList**
- **neighbouringRooms**
- **itemList**

- **Metódusok**

- **void addStudent(Student) :** Egy paraméterként kapott hallgatót belerak a listájába. Akkor kell használni ha belép a szobába egy hallgató
- **void removeStudent(Student) :** Egy paraméterként kapott hallgatót kivesz a listából. Akkor kell használni ha egy hallgató elhagyja a szobát
- **void addTeacher(Teacher) :** Egy paraméterként kapott oktatót belerak a listájába. Akkor kell használni ha belép a szobába egy oktató
- **void removeTeacher(Teacher) :** Egy paraméterként kapott oktatót kivesz a listából. Akkor kell használni ha egy oktató elhagyja a szobát
- **void addNeighbour(Room) :** Egy paraméterként kapott szobát belerak a szomszédsági listájába. Akkor kell használni ha egy split-el szoba
- **void removeNeighbour(Room) :** Egy paraméterként kapott szobát kivesz a szomszédsági listából. Akkor kell használni ha egy merge-el szoba
- **Room[] getNeighbours() :** Visszaadja a szomszédait egy listában. Egy másik szoba hívja ha mergelni akar

- **void addItem(item)** : Egy paraméterként kapott itemet belerak a listájába. Akkor kell használni ha valaki eldob egy tárgyat egy szobában
- **void removeItem(Item)** : Egy paraméterként kapott itemet kivesz a listából. Akkor kell használni ha valaki felvesz egy tárgyat
- **void addEffect(ERoomEffects)** Hozzáadási művelet, hogy számontarthassa, hogy milyen hatások érik aktuálisan az adott szobát, hozzáad az effects listához egy ERoomEffects-et. Ha a listában még nem szereplő hatás kerül bele akkor üzenetet küldhet
- **void removeEffect(ERoomEffects)** Az effects “hatástömb” elemeiből kiveszi a megszűnt hatás egy elemét. Ha nem maradt több az adott hatásból akkor szükség esetén üzenetetküld

### 4.3.3 IRoomManager

- **Felelősség**

Interface a szobaműveletek szervezéséhez. A szoba osztály implementálja

- **Metódusok**

- **boolean split()**: Az adott szobát megpróbálja elosztani két szobára. Az osztódást követően az egyik szoba megtartja összes szomszédját, a másiknak pedig egyetlen egy szomszédja lesz (az a szoba amiből osztódott). Az első szobában lévő tárgyak és tulajdonságok valamilyen algoritmus alapján szétesztődnak a két szoba között. Ha játékos tartózkodik a szobában, nem lesz split.
- **boolean merge(Room r)**: Két adott szobát egyesít. Az új szoba a korábbi két szoba tulajdonságaival, tárgyaival és szomszédaival rendelkezik. Az új szoba befogadóképessége azonos lesz a nagyobbik szoba befogadóképességével. Ha játékos tartózkodik bármelyik szobában a kettő közül, nem lesz merge.

### 4.3.4 Player

- **Felelősség**

Absztrakt osztály, az összes játékos leszármazik tőle (jelen esetben oktató/hallgató). Általános tulajdonságokat határoz meg, amikkel minden játékosnak rendelkeznie kell.

- **Attribútumok**

- **room** - A jelenlegi szobát tárolja
- **itemList** - Az itemek listáját tárolja
- **state** - A játékos állapotát tárolja
- **timer** - Az osztály saját időmérője

- **Metódusok**

- **void move(Room r)**: A játékos megpróbál lépni az r szobába. Ha nem tud, nem történik semmi.
- **EPlayerState getState()**: Lekérhetjük a játékos jelenlegi állapotát.
- **void setState(EPlayerState s)**: Beállíthatjuk a játékos állapotát.
- **void RoomPoisoned()**: A mérgezés hatását, ájulás kezeli a játékoson úgy, hogy az itemeitől kérdezi meg, hogy meg tudják-e védeni majd az adott válasz alapján eldönti, hogy mérgezett lesz-e vagy sem.
- **void RoomCleanFromPoison()**: A mérgező hatások végét kezeli a játékoson és az itemeit is értesíti

- **void addItem(Item):** Új tárgy játékoshoz rendelését kezeli.
- **void remove(Item):** Egy tárgy a játékostól való elvételét kezeli.
- **void pickItem(Item):** Egy tárgy felvételét kezeli a játékoson. Az itemnek szól (aki majd addItemmel eltárolja magát).
- **void dropItem(Item)** Egy tárgy elejtését kezeli az adott játékos esetében. Az itemnek szól (aki majd removeItemmel kitörli magát).
- **void timerEnd():** Az időzítés végét kezeli, amit a kontroller vezérel.

#### 4.3.5 Teacher

- **Felelősség**

Osztály, ami megvalósítja az Oktató működését.

- **Ősosztályok**

Player

- **Metódusok**

- **void move(Room r):** A játékos megpróbál lépni az r szobába. Ha nem tud, nem történik semmi.
- **EPlayerState getState():** Lekérhetjük a játékos jelenlegi állapotát.
- **void setState(EPlayerState s):** Beállíthatjuk a játékos állapotát.

#### 4.3.6 Student

- **Felelősség**

A Hallgató működését megvalósító osztály. Ezzel kommunikál a felhasznál, meghívhatja a move függvényét, ezzel mozgatva, vagy a tárgyait lekérdezheti és használhat egyet

- **Ősosztályok**

Player

- **Metódusok**

- **void move(Room r):** A játékos megpróbál lépni az r szobába. Ha nem tud, nem történik semmi.
- **EPlayerState getState():** Lekérhetjük a játékos jelenlegi állapotát.
- **void setState(EPlayerState s):** Beállíthatjuk a játékos állapotát.
- **void TeacherAttacked():** A támadás negatív hatását kezeli a játékoson. A tárgyainak küld üzenetet majd a válasz alapján eldönti, hogy halott lesz-e vagy sem.

### 4.3.7 IItem

- **Felelősség**

Absztrakt osztály, különböző tárgyak használatát segíti elő. Ha egy Player használni akar egy tárgyat, akkor az Item továbbadja a kérelmet a konkrét tárgynak. Visitor pattern alapján segít megteremteni a tárgyhasználatot. minden tárgyunknak ezt implementálnia kell.

- **Metódusok**

- **void useItem(Player p):** Tárgy használata, egy esemény hatására játékos esetén.
- **void useItem(IItemTarget t):** Event nélküli tárgyhasználat egy célponton.
- **void pickUp(Player p):** A tárgy felvétele p playerre. Belerakja a tárgylistájába az adott tárgyat
- **void dropItem(Player p):** A tárgy eldobása p playerre. Kiveszi a tárgylistából az adott tárgyat
- **bool TeacherAttacked(Student s):** Akkor implementálja egy tárgy ha képes megvédeni az oktatót. Eldönti, hogy meg tudja-e védeni majd értesíti a játékosot.
- **bool RoomPoisoned(Student s):** A mérgezés hatását, ájulás kezeli a játékoson.
- **void RoomCleanFromPoison(Student s):** A mérgező hatások végét kezeli az itemein.

### 4.3.8 TVSZ

- **Felelősség**

A TVSZ denevérbőr tárgy hatásáért felelős.

- **Interfészek**

IItem

- **Attribútumok**

- **hitpoints**: int

- **Metódusok**

- **bool TeacherAttacked(Student s)**: Eldönti, hogy meg tudja-e védeni majd értesíti a játékost

### 4.3.9 Beer

- **Felelősség**

A Szent söröspohár tárgy hatásáért felelős.

- **Interfészek**

IItem

- **Attribútumok**

- **timer** : ITimer
- **state** : BeerState

- **Metódusok**

- **void useItem(Student)**: Elindítjuk a sör számlálóját, innentől tud megvédeni minket.
- **void setState(BeerState)** : Be tudja állítani az állapotát
- **bool TeacherAttacked(Student s)**: Eldönti, hogy meg tudja-e védeni majd értesíti a játékost. inaktív, elindítja a timert
- **void timerEnd()**: disabled-re állítja magát disabled-re

### 4.3.10 Transistor

- **Felelősség**

A Tranzisztor tárgy hatásáért és akciójáért felelős. A kontroller hívásainak kezelését valósítja meg.

- **Interfészek**

IItem

- **Attribútumok**

- **active:** bool
- **room :** Room
- **pair:** Transistor

- **Metódusok**

- **void activateTransistor():** A kontroller hívására a tranzisztor aktiválni próbálja.
- **void pairTransistor(Transistor t2):** A függvény hatására a két tranzisztor beállítja egymásnak a pár tranzisztort.
- **void unpairTransistor(Transistor t2):** A kontroller hívja, abban az esetben a játékos fel akarja oldani a két tranzisztor közötti kapcsolatot.
- **void dropItem(Player s):** A kontroller hívására a tranzisztor a játékos szobájába próbálja helyezni. Ha a feltételek teljesülnek, a tranzisztor teleport hatása is végbemegy.
- **void pickUp(Player s):** A funkció hatására a elhelyezett tranzisztor a játékos felveheti, kontroller hívja.
- **void setRoom(Room r):** A kontroller hívására a tranzisztor pozíciója, szobája beállításra kerül.
- **void unactivateTransistor():** A funkció az aktív állapotot kísérli meg kikapcsolni.

### 4.3.11 Mask

- **Felelősség**

A FFP2 Maszk tárgy hatásáért felelős.

- **Interfészek**

IItem

- **Attribútumok**

- **timer:** ITimer
- **student:** Student
- **functional:** bool

- **Metódusok**

- **void pickUp(Student)**: Beállítja, hogy a student attribútumát, a kapott objektumra
- **bool RoomPoisoned(Student)**: Megnézi, hogy képes-e megvédeni a hallgatót, ha igen akkor elindítja a timert és visszaüzen
- **void RoomClearFromPoison(Student)**: Megállítja a timert (pauseTimer())
- **void timerEnd**: “nem funkcionál”-ra állítja a maszkot

### 4.3.12      **Sponge**

- **Felelősség**

A Nedves táblatörlő rongy tárgy hatásáért felelős.

- **Interfészek**

IItem

- **Attribútumok**

- **timer : ITimer**
- **functional**: bool

- **Metódusok**

- **void timerEnd()**: Beállítja magát nem funkcionálónak

### 4.3.13      **Cheese**

- **Felelősség**

A Káposztás camembert tárgy hatásáért felelős.

- **Interfész**

IItem

- **Attribútumok**

- **isUsed** : bool
- **timer**: ITimer

- **Metódusok**

- **timerEnd()**: Beállítja magát elhasználtnak

### 4.3.14 Logarléc

- **Felelősség**

A logarléc tárgy hatásáért felelős.

- **Interfészek**

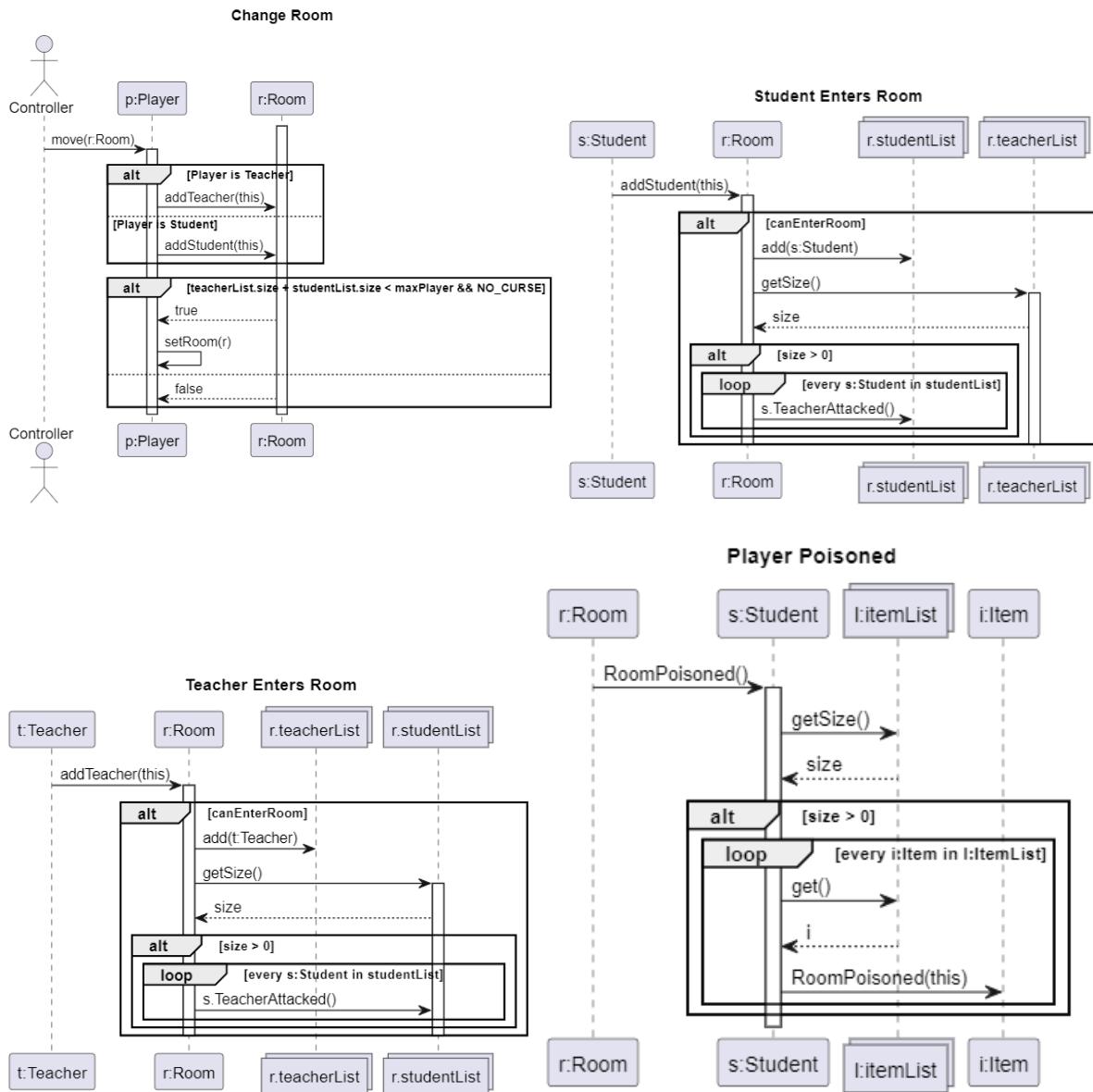
IItem

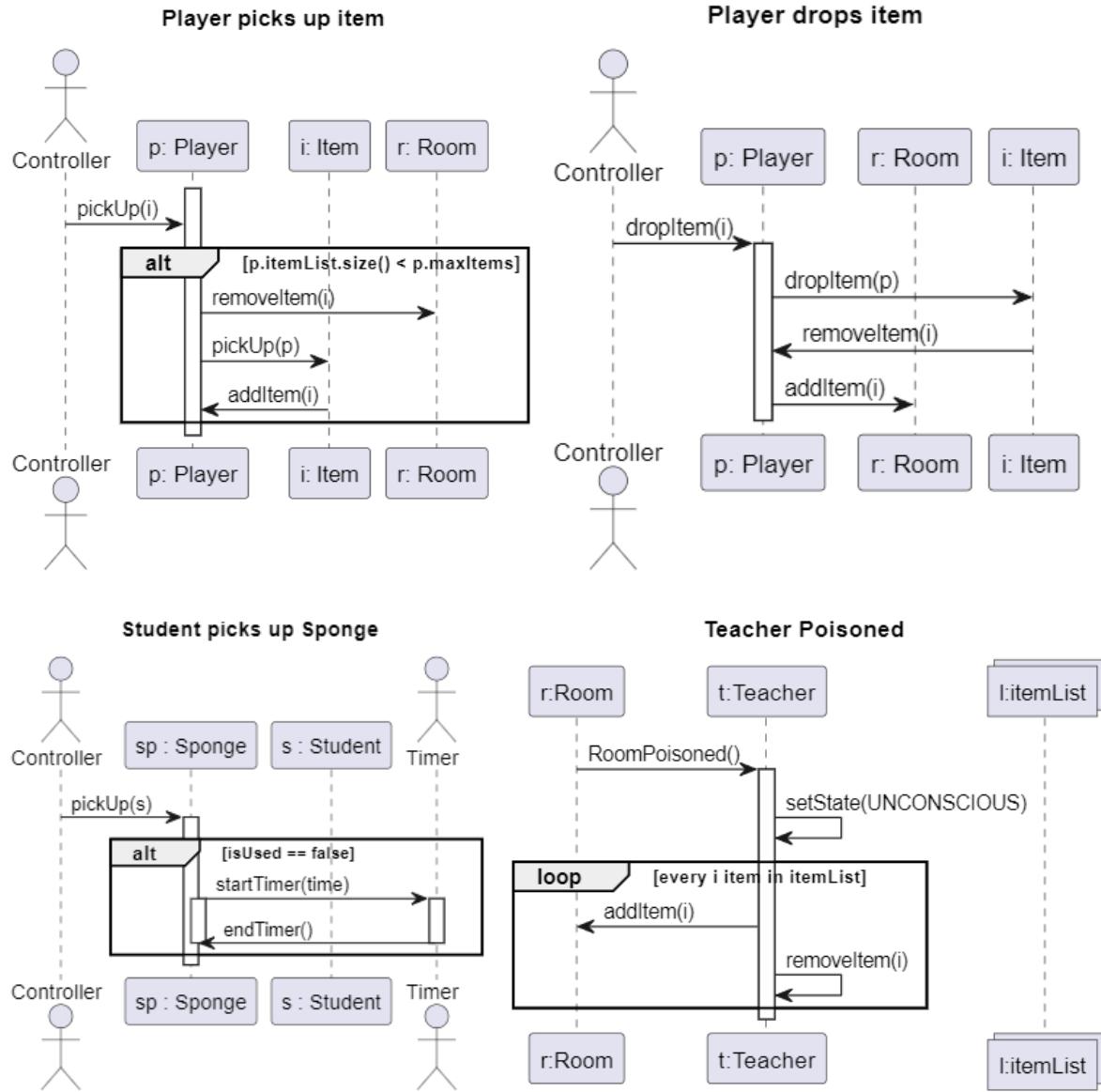
- **Metódusok**

- **pickUp(Student):** Üzen a kontrollernek, hogy a játék győzelemmel zárul

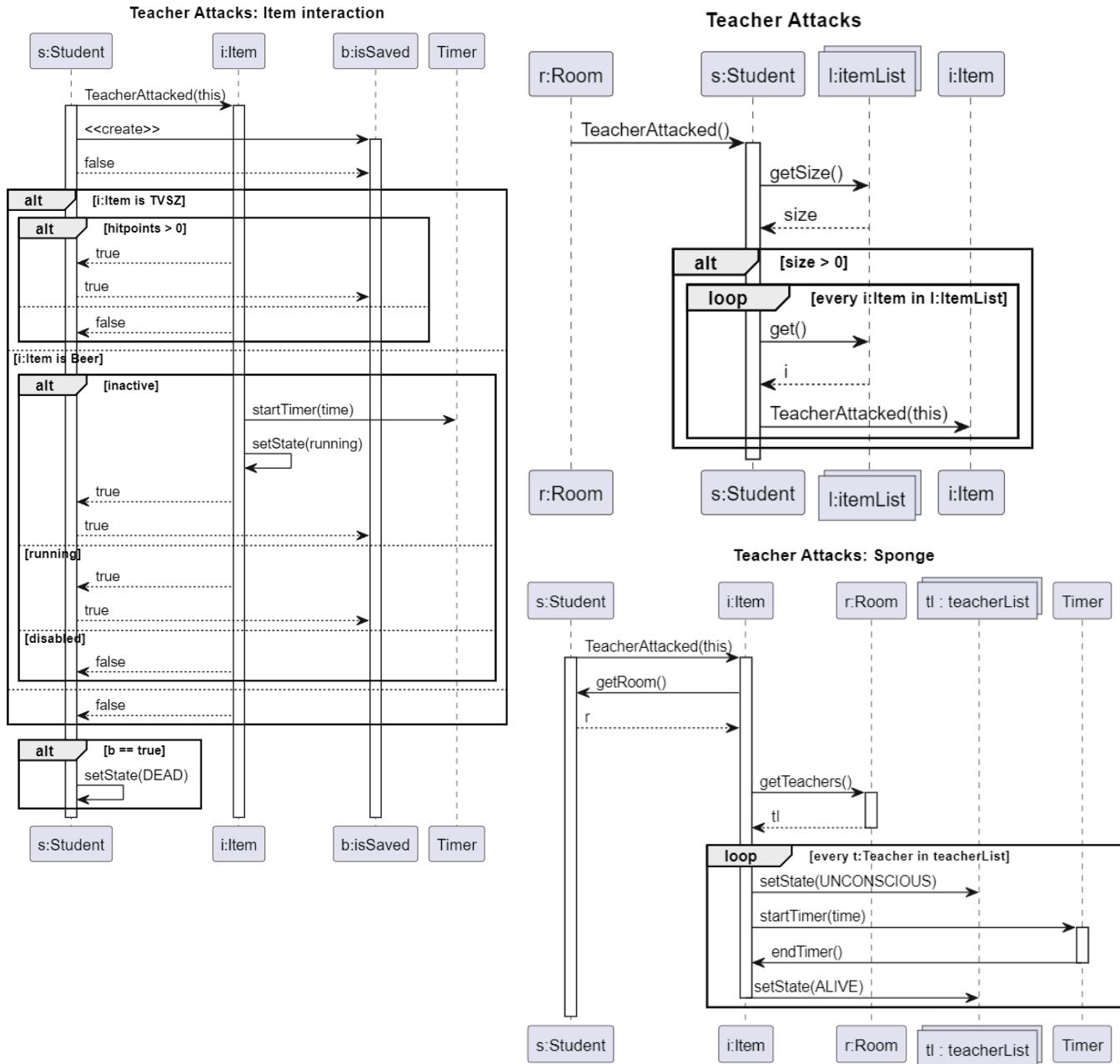
## 4.4. Szekvencia diagramok

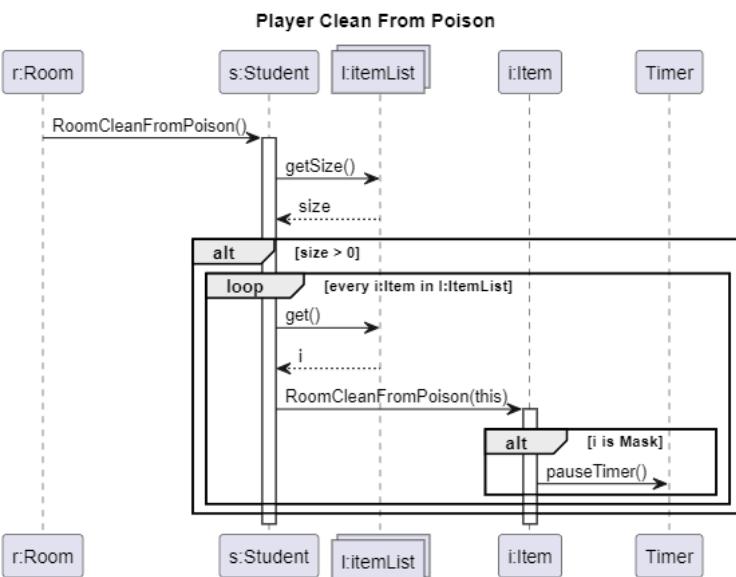
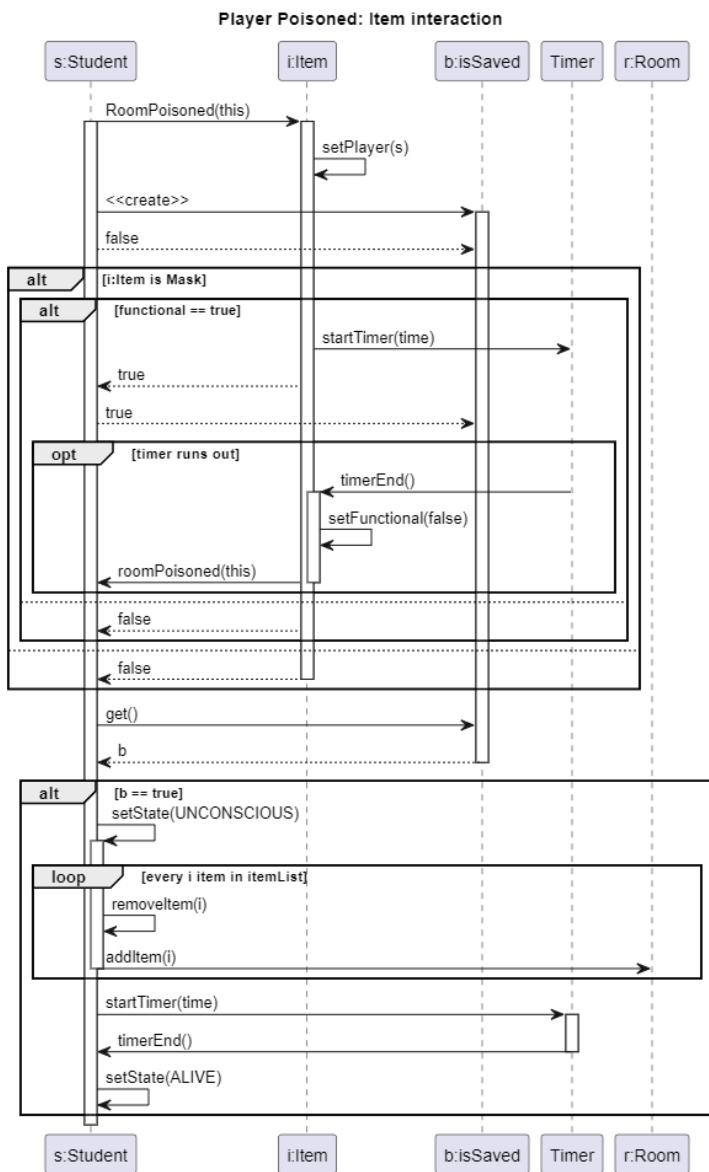
### Room-Player interakciók



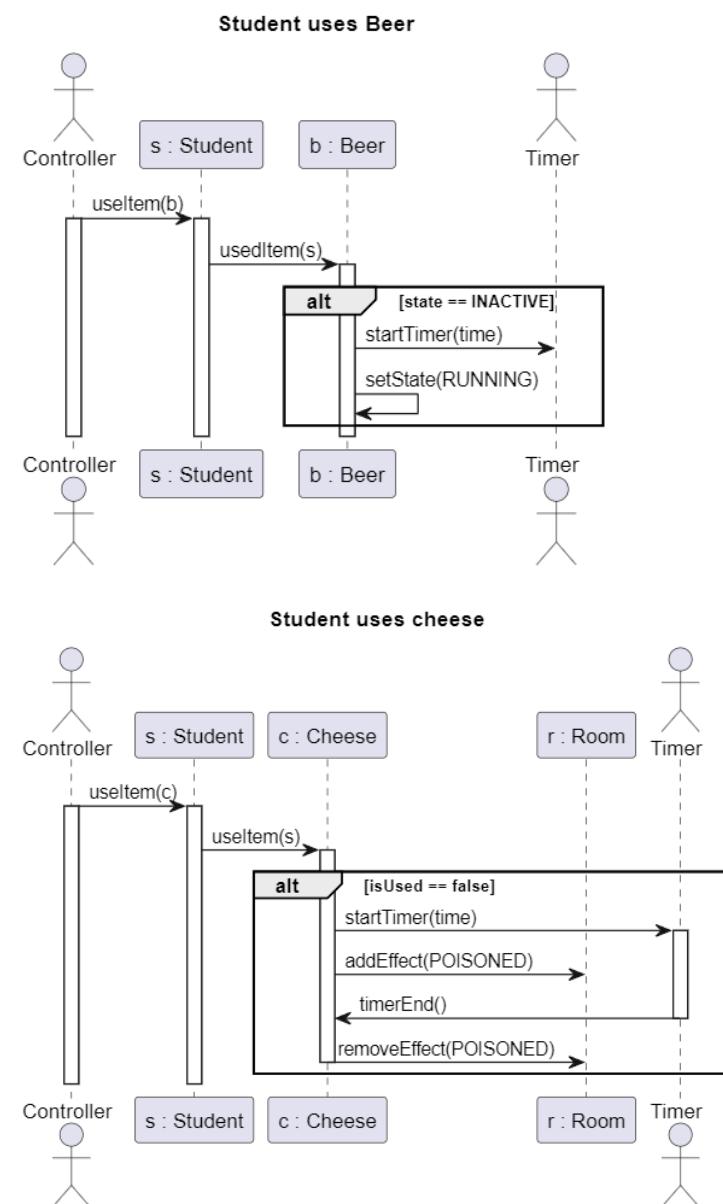


## Automatikus Player-Item interakció

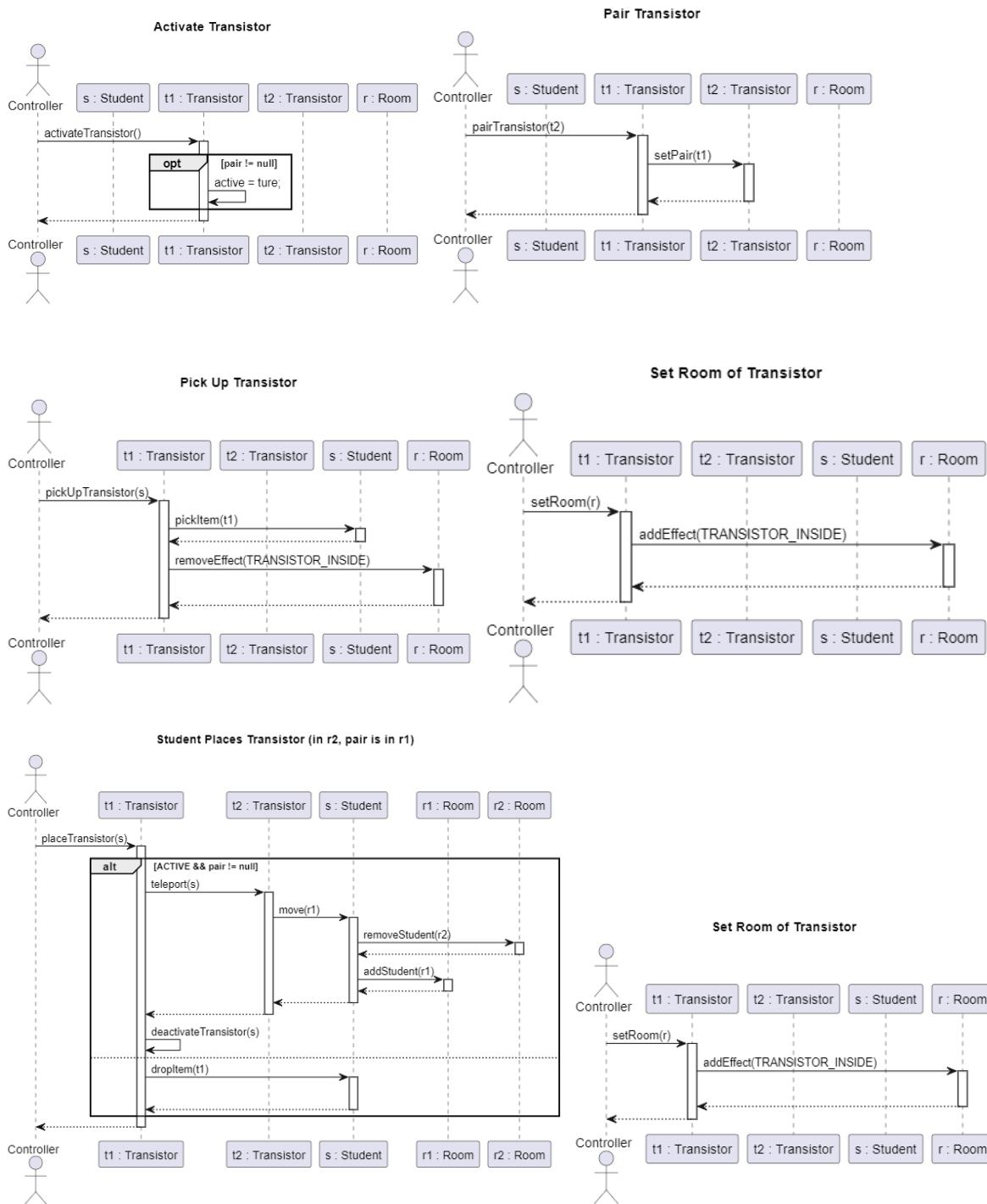


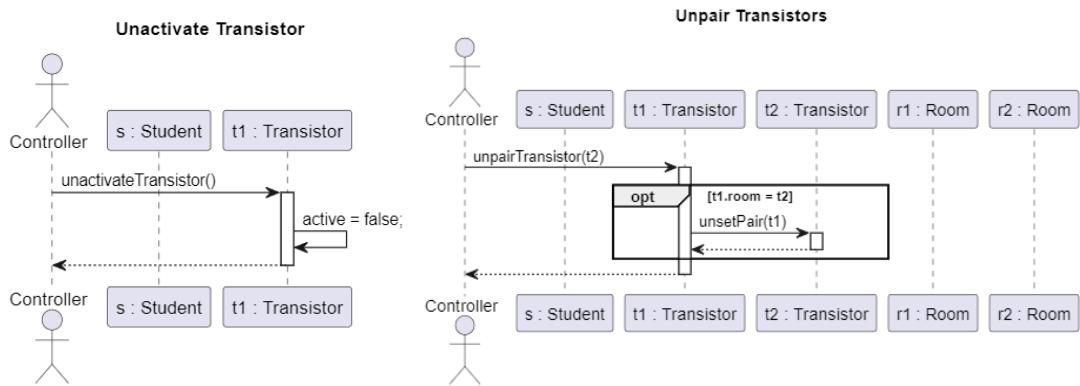


## Item használat

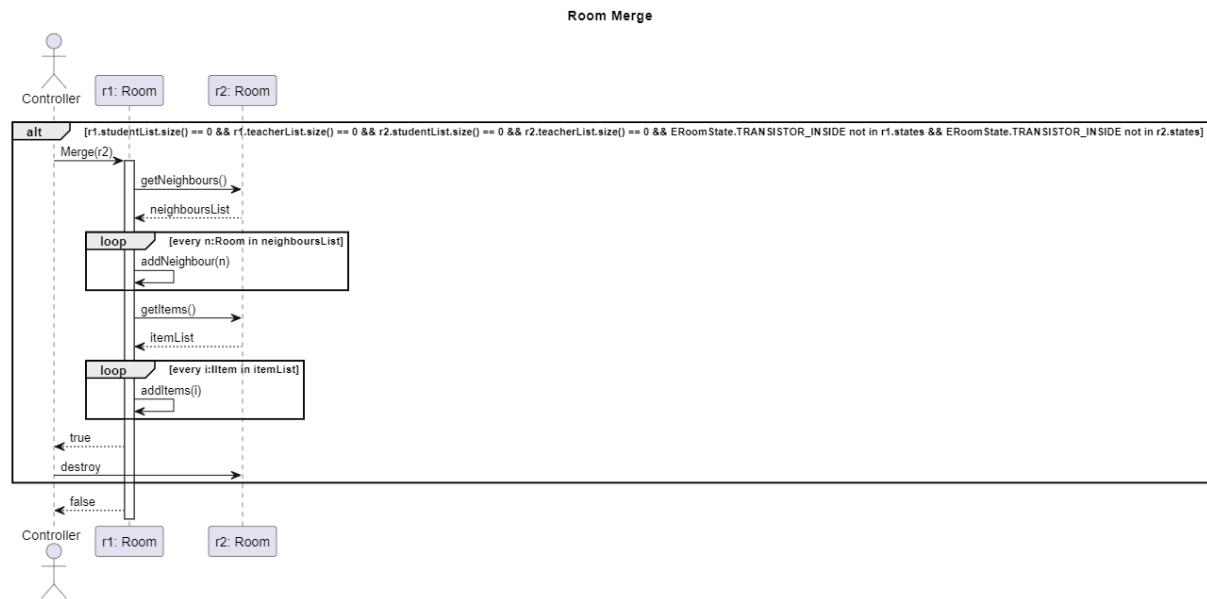


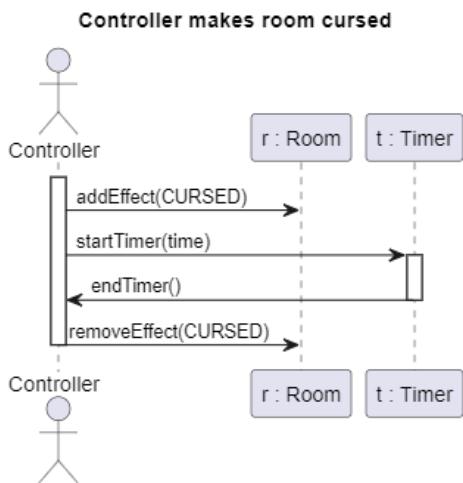
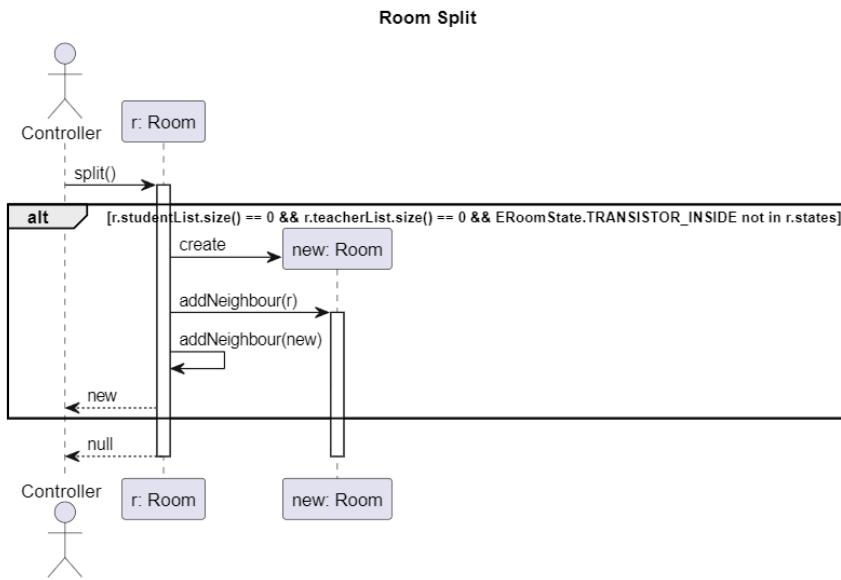
## Tranzisztor



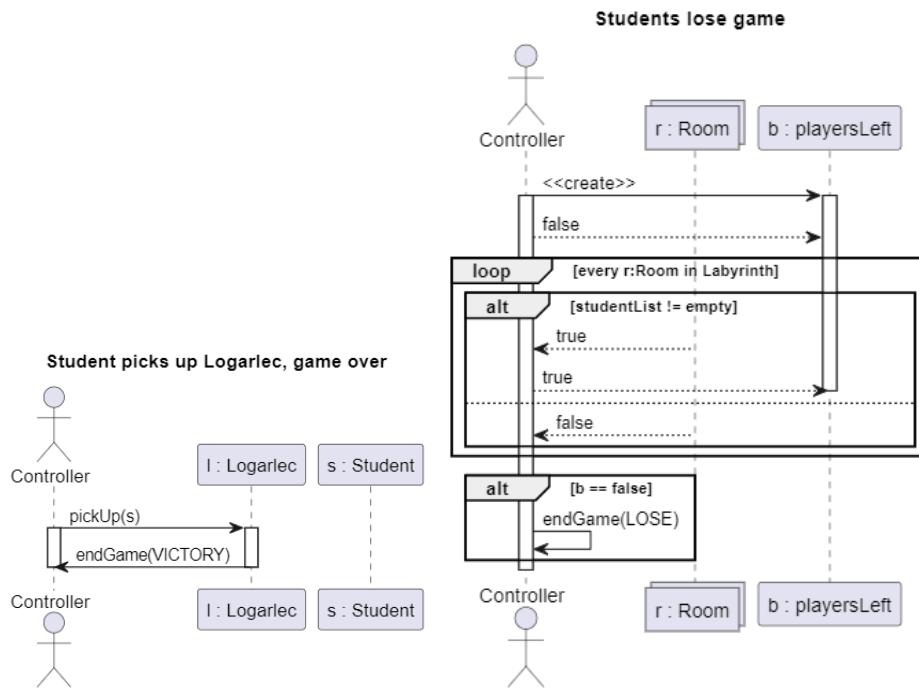


## Controller-Room interakciók



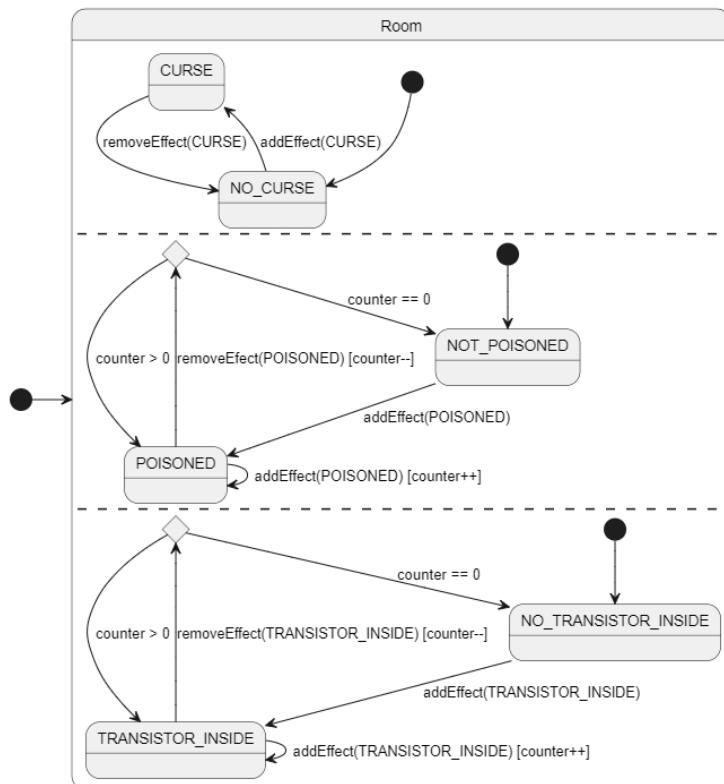


## End of Game

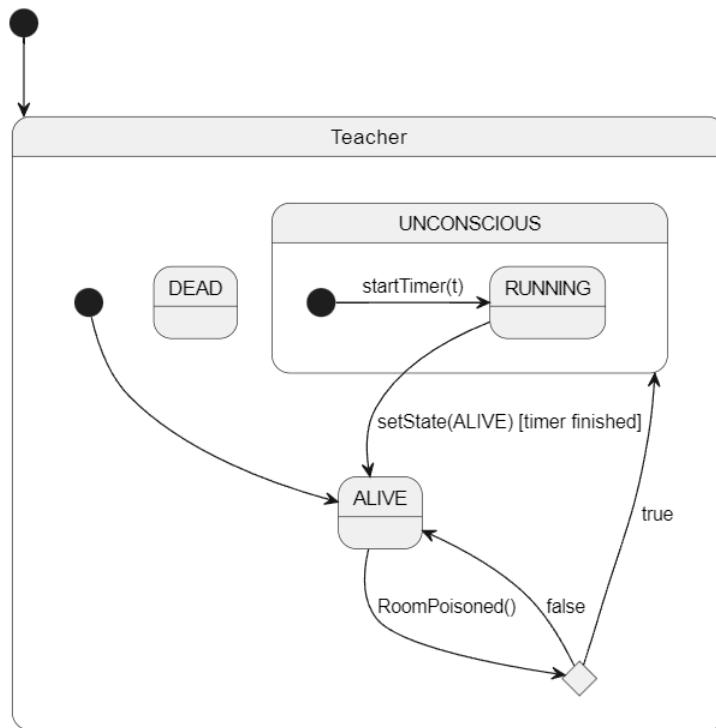
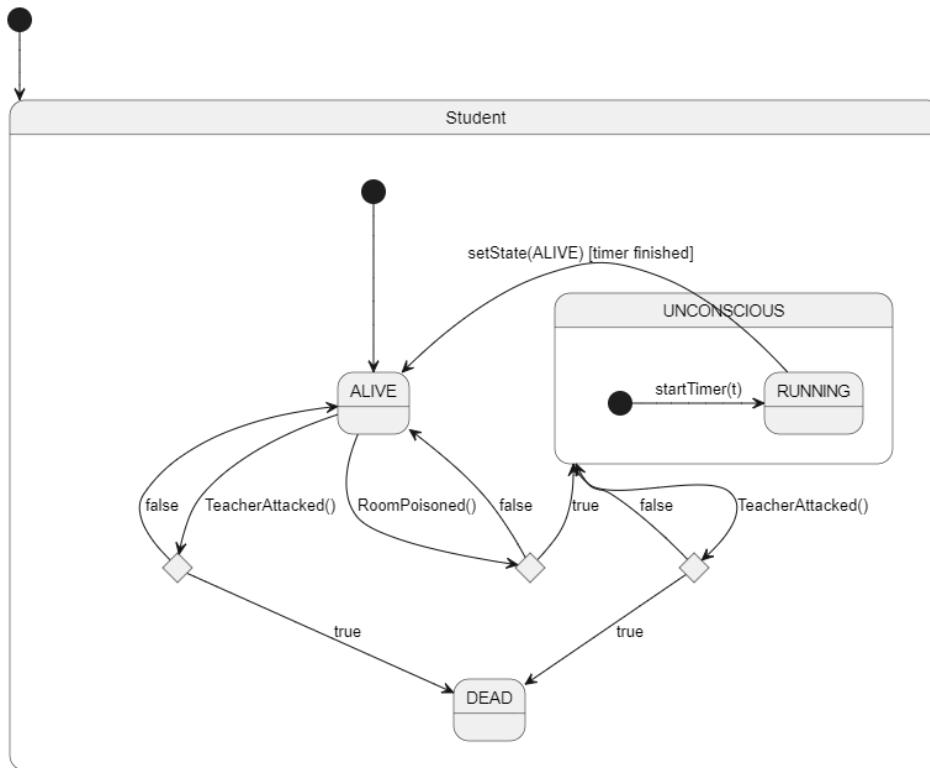


## 4.5. State-chartok

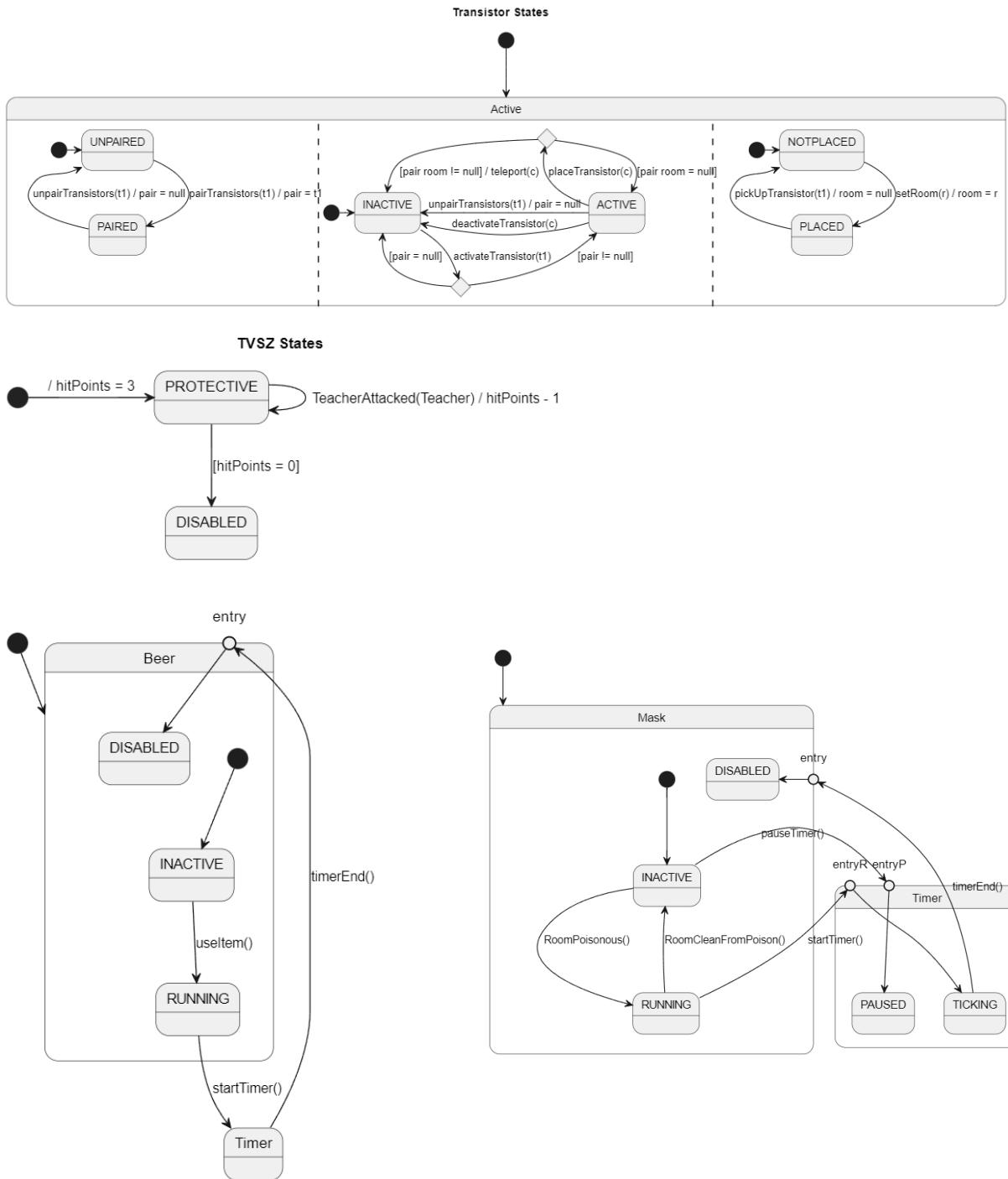
### Room

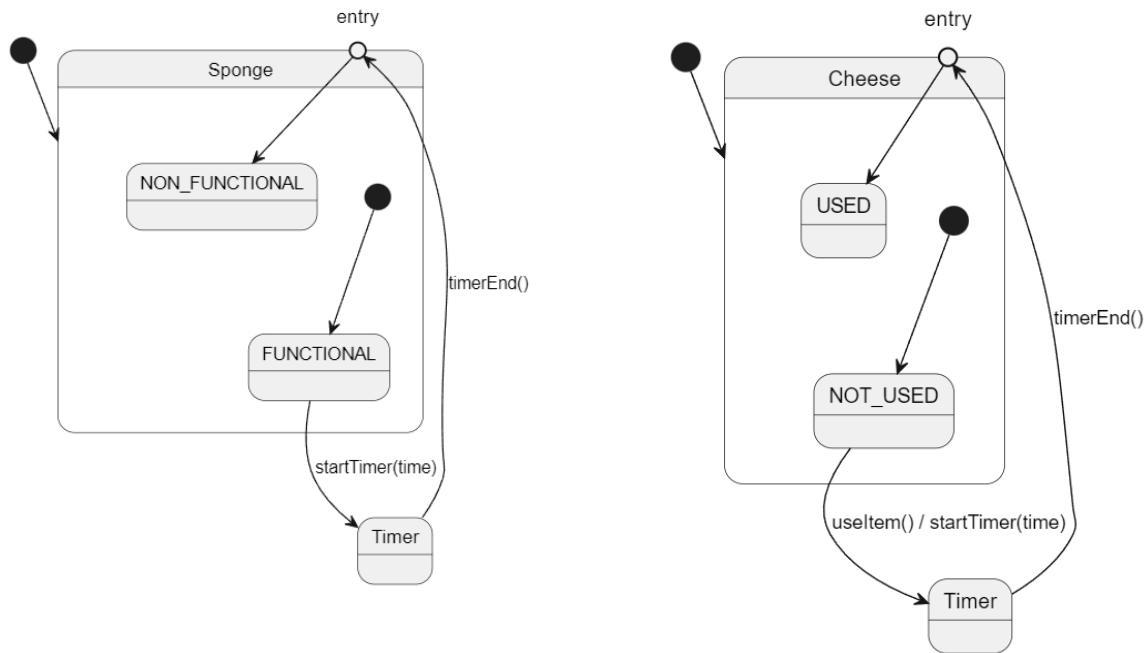


## Player



## Itemek

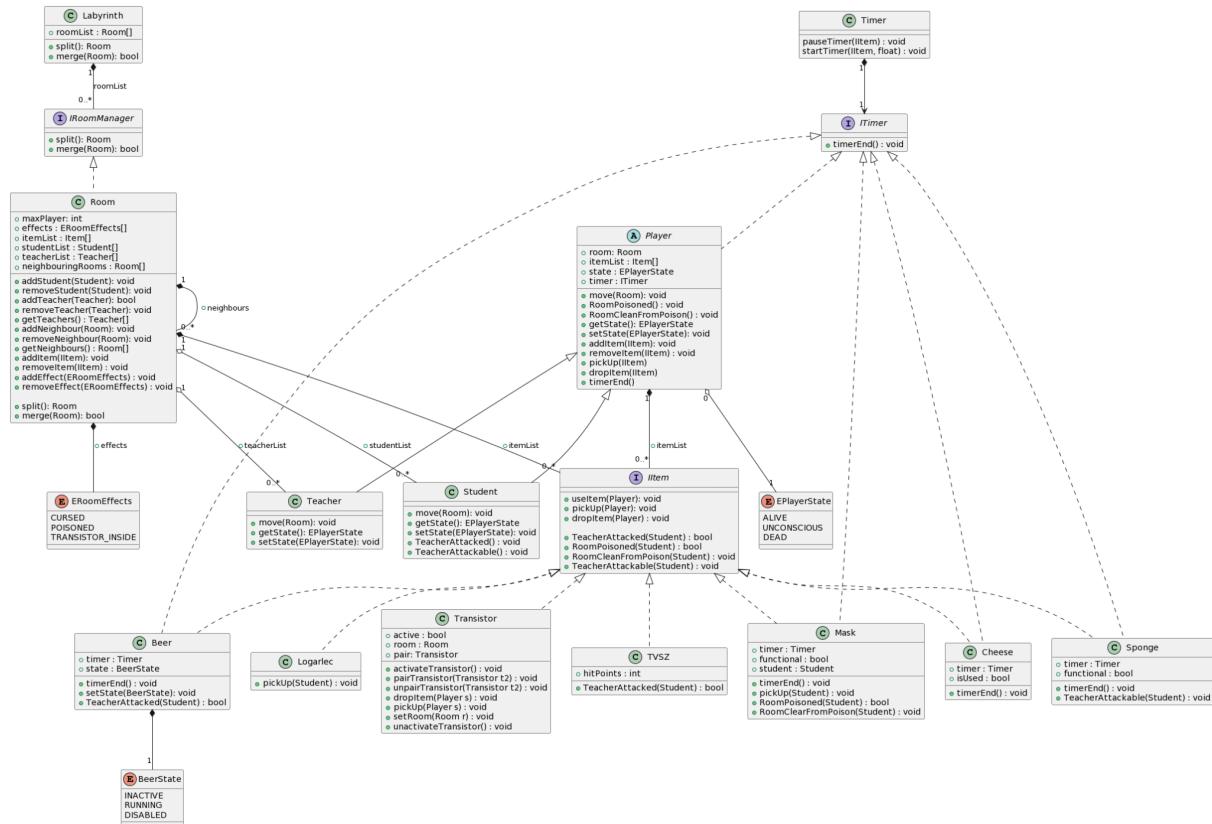




## 5. Szkeleton tervezése

### 5.0 Változtatások

Az Osztálydiagramon változtattunk, a konzultáción beszéltek szerint és egyéb észrevételek miatt



### 5.1 A szkeleton modell valóságos use-case-ai

#### 5.1.0 Elemi események

Tárgy Felvételi
1. Hallgató Logarlécet vesz fel
2. Hallgató maszkot vesz fel
3. Hallgató sajtot vesz fel
4. Hallgató szivacsot vesz fel
5. Hallgató tranzisztort vesz fel
6. Hallgató TVSZ-t vesz fel
7. Oktató Logarlécet vesz fel
8. Oktató maszkot vesz fel
9. Oktató sajtot vesz fel
10. Oktató szivacsot vesz fel

<b>Tárgy Felvétele</b>
11. Oktató tranzisztort vesz fel
12. Oktató TVSZ-t vesz fel
13. Hallgató párosított tranzisztort vesz fel
14. Oktató párosított tranzisztort vesz fel

<b>Tárgy Eldobása</b>
15. Hallgató maszkot dob el
16. Hallgató sört dob el
17. Hallgató sajtot dob el
18. Hallgató szivacsot dob el
19. Hallgató tranzisztort dob el
20. Hallgató TVSZ-t dob el
21. Oktató Logarécet dob el
22. Oktató maszkot dob el
23. Oktató sört dob el
24. Oktató sajtot dob el
25. Oktató szivacsot dob el
26. Oktató tranzisztort dob el
27. Oktató TVSZ-t dob el

<b>Tárgy Használata</b>
28. Hallgató két tranzisztort párosít
29. Hallgató sajtot használ
30. Hallgató sört használ
31. Hallgató tranzisztor által teleportál
32. Hallgató tranzisztort aktivál
33. Sajt használata a szobát gázossá állítja, adott időre

<b>Szoba Interakciók</b>
34. Hallgató átkozott szobába próbál lépni
35. Hallgató átkozott szobából próbál kilépni
36. Hallgató gázos szobába lép
37. Hallgató szobája gázos lesz
38. Hallgató kilép a gázos szobából
39. Hallgató szobájáról lekerül a gáz

<b>Szoba Interakciók</b>
40. Hallgató szobába lép, amiben hallgató van
41. Hallgató szobába lép, amiben oktató van
42. Hallgató teli szobába próbál lépni
43. Hallgató üres szobába lép
44. Hallgató áll egy szobában, majd belép egy oktató
45. Oktató átkozott szobába próbál lépni
46. Oktató átkozott szobából próbál kilépni
47. Oktató gázos szobába lép
48. Oktató szobája gázos lesz
49. Oktató szobájáról lekerül a gáz
50. Oktató szobába lép, amiben hallgató van
51. Oktató szobába lép, amiben oktató van
52. Oktató teli szobába próbál lépni
53. Oktató üres szobába lép

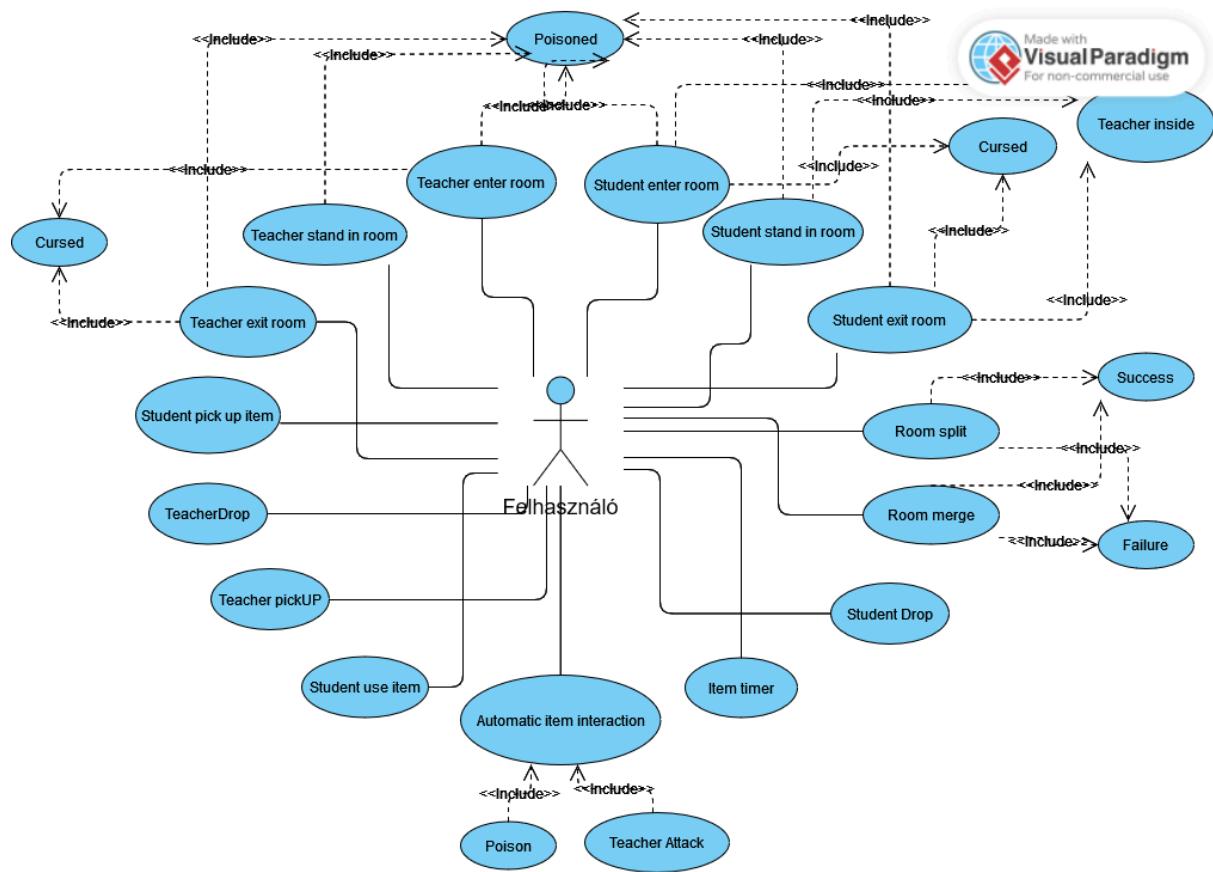
  

<b>Szoba Működése</b>
54. A szoba átkozott állapota a megadott idő után eltűnik
55. Szoba merge-l
56. Szoba nem tud merge-tni tranzisztor miatt
57. Szoba nem tud merge-tni, mert tartózkodnak benne
58. Szoba nem tud split-tni tranzisztor miatt
59. Szoba nem tud split-tni, mert tartózkodnak benne
60. Szoba split-el

<b>Tárgyak Működése</b>
61. Hallgatókat megmenti a sör
62. Hallgatókat megmenti a szivacs
63. Hallgatót megmenti a maszk
64. Hallgatót megmenti a TVSZ
65. Sajt időzítő lejárta után a szobáról lejön egy gázos hatás
66. Sör időzítő lejárta után a sör már nem védi meg a hallgatót
67. Szivacs időzítő lejárta után a szivacs már nem védi meg a hallgatót
68. Maszk időzítője, szoba megtisztulása után megáll

### 5.1.1 Use-case diagram



## 5.1.2 Use-case leírások

<b>Use-case neve</b>	Hallgató Logarléket vesz fel
<b>Kapcsolódó</b>	5.3.3
<b>Rövid leírás</b>	Hallgató felveszi a Logarléket
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hallgató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és a hallgató tárgyait. A hallgató felveszi a szobában lévő Logarléket. A játéknak győzelemmel vége, ezt konzolra kiírjuk.

<b>Use-case neve</b>	Hallgató maszkot vesz fel
<b>Kapcsolódó</b>	5.3.1
<b>Rövid leírás</b>	Hallgató felvesz egy maszkot
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hallgató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és a hallgató tárgyait. A hallgató felveszi a szobában lévő maszkot. A maszk bekerül a hallgató tárgyai közé és kikerül a szoba tárgyai közül. Ismét kiírjuk a szoba és a hallgató tárgyait.

<b>Use-case neve</b>	Hallgató szivacsot vesz fel
<b>Kapcsolódó</b>	5.3.2
<b>Rövid leírás</b>	Hallgató felvesz egy szivacsot
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hallgató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és a hallgató tárgyait. A hallgató felveszi a szobában lévő szivacsot. A szivacs bekerül a hallgató tárgyai közé és kikerül a szoba tárgyai közül. Ismét kiírjuk a szoba és a hallgató tárgyait. Elindul a szivacs időzítője. Kiírjuk az időzítő függvény hívását.

<b>Use-case neve</b>	Hallgató sajtot vesz fel
<b>Kapcsolódó</b>	5.3.1
<b>Rövid leírás</b>	Hallgató felvesz egy sajtot
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hallgató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és a hallgató tárgyait. A hallgató felveszi a szobában lévő sajtot. A sajt bekerül a hallgató tárgyai közé és kikerül a szoba tárgyai közül. Ismét kiírjuk a szoba és a hallgató tárgyait.

<b>Use-case neve</b>	Hallgató TVSZ-t vesz fel
<b>Kapcsolódó</b>	5.3.1
<b>Rövid leírás</b>	Hallgató felvesz egy TVSZ-t
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hallgató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és a hallgató tárgyait. A hallgató felveszi a szobában lévő TVSZ-t. A TVSZ bekerül a hallgató tárgyai közé és kikerül a szoba tárgyai közül. Ismét kiírjuk a szoba és a hallgató tárgyait.

<b>Use-case neve</b>	Hallgató tranzisztort vesz fel
<b>Kapcsolódó</b>	5.3.1
<b>Rövid leírás</b>	Hallgató felvesz egy tranzisztort
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hallgató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és a hallgató tárgyait. A hallgató felveszi a szobában lévő párosítatlan tranzisztort. A tranzisztor bekerül a hallgató tárgyai közé és kikerül a szoba tárgyai közül. Ismét kiírjuk a szoba és a hallgató tárgyait.

<b>Use-case neve</b>	Oktató Logarlécet vesz fel
<b>Kapcsolódó</b>	5.3.5
<b>Rövid leírás</b>	Oktató felveszi a Logarlécet
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az oktató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és az oktató tárgyait. Az oktató felveszi a szobában lévő Logarlécet. A Logarléc bekerül az oktató tárgyai közé és kikerül a szoba tárgyai közül. Ismét kiírjuk a szoba és az oktató tárgyait.

<b>Use-case neve</b>	Oktató maszkot vesz fel
<b>Kapcsolódó</b>	5.3.5
<b>Rövid leírás</b>	Oktató felvesz egy maszkot
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az oktató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és az oktató tárgyait. Az oktató felveszi a szobában lévő maszkot. A maszk bekerül az oktató tárgyai közé és kikerül a szoba tárgyai közül. Ismét kiírjuk a szoba és az oktató tárgyait.

<b>Use-case neve</b>	Oktató szivacsot vesz fel
<b>Kapcsolódó</b>	5.3.5
<b>Rövid leírás</b>	Oktató felvesz egy szivacsot
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az oktató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és az oktató tárgyait. Az oktató felveszi a szobában lévő szivacsot. A szivacs bekerül az oktató tárgyai közé és kikerül a szoba tárgyai közül. Ismét kiírjuk a szoba és az oktató tárgyait.

<b>Use-case neve</b>	Oktató TVSZ-t vesz fel
<b>Kapcsolódó</b>	5.3.5
<b>Rövid leírás</b>	Oktató felvesz egy TVSZ-t
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az oktató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és az oktató tárgyait. Az oktató felveszi a szobában lévő TVSZ-t. A TVSZ bekerül az oktató tárgyai közé és kikerül a szoba tárgyai közül. Ismét kiírjuk a szoba és az oktató tárgyait.

<b>Use-case neve</b>	Oktató sajtot vesz fel
<b>Kapcsolódó</b>	5.3.5
<b>Rövid leírás</b>	Oktató felvesz egy sajtot
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az oktató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és az oktató tárgyait. Az oktató felveszi a szobában lévő sajtot. A sajt bekerül az oktató tárgyai közé és kikerül a szoba tárgyai közül. Ismét kiírjuk a szoba és az oktató tárgyait.

<b>Use-case neve</b>	Oktató tranzisztort vesz fel
<b>Kapcsolódó</b>	5.3.5
<b>Rövid leírás</b>	Oktató felvesz egy tranzisztort
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az oktató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és az oktató tárgyait. Az oktató felveszi a szobában lévő párosítatlan tranzisztort. A tranzisztor bekerül az oktató tárgyai közé és kikerül a szoba tárgyai közül. Ismét kiírjuk a szoba és az oktató tárgyait.

<b>Use-case neve</b>	Hallgató párosított tranzisztort vesz fel
<b>Kapcsolódó</b>	5.3.4

<b>Use-case neve</b>	Hallgató párosított tranzisztort vesz fel
<b>Rövid leírás</b>	Hallgató felvesz egy párosított tranzisztort
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hallgató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és a hallgató tárgyait. A hallgató felveszi a szobában lévő párosított tranzisztort. A tranzisztor bekerül a hallgató tárgyai közé és kikerül a szoba tárgyai közül. A tranzisztor és párja között megszakad a kapcsolat. Ismét kiírjuk a szoba és a hallgató tárgyait, és a tranzisztorok állapotát is.

<b>Use-case neve</b>	Oktató párosított tranzisztort vesz fel
<b>Kapcsolódó</b>	5.3.4
<b>Rövid leírás</b>	Oktató felvesz egy párosított tranzisztort
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az oktató egy szobában tartózkodik és van szabad tárhelye, kiírjuk a szoba és a hallgató tárgyait. Az oktató felveszi a szobában lévő párosított tranzisztort. A tranzisztor bekerül az oktató tárgyai közé és kikerül a szoba tárgyai közül. A tranzisztor és párja között megszakad a kapcsolat. Ismét kiírjuk a szoba és az oktató tárgyait, és a tranzisztorok állapotát is.

<b>Use-case neve</b>	Student drops mask
<b>Kapcsolódó</b>	5.3.28
<b>Rövid leírás</b>	Hallgató maszkot dob el
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hallgató egy szobában tartózkodik és van maszk a tárgyai között, kiírjuk a szoba és a hallgató tárgyait. A hallgató eldobja a maszkját a szobába. A maszk kikerül a hallgató tárgyai közül és bekerül a szoba tárgyai közé. Ismét kiírjuk a szoba és a hallgató tárgyait.

<b>Use-case neve</b>	Student drops beer
<b>Kapcsolódó</b>	5.3.26
<b>Rövid leírás</b>	Hallgató sört dob el
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hallgató egy szobában tartózkodik és van sör a tárgyai között, kiírjuk a szoba és a hallgató tárgyait. A hallgató eldobja a söret a szobába. A sör kikerül a hallgató tárgyai közül és bekerül a szoba tárgyai közé. Ismét kiírjuk a szoba és a hallgató tárgyait.

<b>Use-case neve</b>	Student drops cheese
<b>Kapcsolódó</b>	5.3.27
<b>Rövid leírás</b>	Hallgató sajtot dob el
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hallgató egy szobában tartózkodik és van sajt a tárgyai között, kiírjuk a szoba és a hallgató tárgyait. A hallgató eldobja a sajtját a szobába. A sajt kikerül a hallgató tárgyai közül és bekerül a szoba tárgyai közé. Ismét kiírjuk a szoba és a hallgató tárgyait.

<b>Use-case neve</b>	Student drops sponge
<b>Kapcsolódó</b>	5.3.29
<b>Rövid leírás</b>	Hallgató szivacsot dob el
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hallgató egy szobában tartózkodik és van szivacs a tárgyai között, kiírjuk a szoba és a hallgató tárgyait. A hallgató eldobja a szivacsát a szobába. A szivacs kikerül a hallgató tárgyai közül és bekerül a szoba tárgyai közé. Ismét kiírjuk a szoba és a hallgató tárgyait.

<b>Use-case neve</b>	Student drops TVSZ
<b>Kapcsolódó</b>	5.3.30

<b>Use-case neve</b>	Student drops TVSZ
<b>Rövid leírás</b>	Hallgató TVSZ-t dob el
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hallgató egy szobában tartózkodik és van TVSZ a tárgyai közt, kiírjuk a szoba és a hallgató tárgyait. A hallgató eldobja a TVSZ-t a szobába. A TVSZ kikerül a hallgató tárgyai közül és bekerül a szoba tárgyai közé. Ismét kiírjuk a szoba és a hallgató tárgyait.

<b>Use-case neve</b>	Student places inactive or unpaired transistor
<b>Kapcsolódó</b>	5.3.40
<b>Rövid leírás</b>	Hallgató tranzisztort dob el
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hallgató egy szobában tartózkodik és van párosítatlan tranzisztor a tárgyai közt, kiírjuk a szoba és a hallgató tárgyait. A hallgató eldobja a tranzisztort a szobába. A tranzisztor kikerül a hallgató tárgyai közül és bekerül a szoba tárgyai közé. Ismét kiírjuk a szoba és a hallgató tárgyait.

<b>Use-case neve</b>	Teacher Drops Logarléc
<b>Kapcsolódó</b>	5.3.55
<b>Rövid leírás</b>	Oktató Logarlécet dob el
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az oktató egy szobában tartózkodik és van Logarléc a tárgyai közt, kiírjuk a szoba és az oktató tárgyait. Az oktató eldobja a Logarlécet a szobába. A Logarléc kikerül az oktató tárgyai közül és bekerül a szoba tárgyai közé. Ismét kiírjuk a szoba és az oktató tárgyait.

<b>Use-case neve</b>	Teacher drops mask
<b>Kapcsolódó</b>	5.3.33
<b>Rövid leírás</b>	Oktató maszkot dob el
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az oktató egy szobában tartózkodik és van maszk a tárgyai közt, kiírjuk a szoba és az oktató tárgyait. Az oktató eldobja a maszkot a szobába. A maszk kikerül az oktató tárgyai közül és bekerül a szoba tárgyai közé. Ismét kiírjuk a szoba és az oktató tárgyait.

<b>Use-case neve</b>	Teacher drops cheese
<b>Kapcsolódó</b>	5.3.32
<b>Rövid leírás</b>	Oktató sajtot dob el
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az oktató egy szobában tartózkodik és van sajt a tárgyai közt, kiírjuk a szoba és az oktató tárgyait. Az oktató eldobja a sajtot a szobába. A sajt kikerül az oktató tárgyai közül és bekerül a szoba tárgyai közé. Ismét kiírjuk a szoba és az oktató tárgyait.

<b>Use-case neve</b>	Teacher drops sponge
<b>Kapcsolódó</b>	5.3.34
<b>Rövid leírás</b>	Oktató szivacsot dob el
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az oktató egy szobában tartózkodik és van szivacs a tárgyai közt, kiírjuk a szoba és az oktató tárgyait. Az oktató eldobja a szivacsot a szobába. A szivacs kikerül az oktató tárgyai közül és bekerül a szoba tárgyai közé. Ismét kiírjuk a szoba és az oktató tárgyait.

<b>Use-case neve</b>	Teacher drops TVSZ
<b>Kapcsolódó</b>	5.3.35
<b>Rövid leírás</b>	Oktató TVSZ-t dob el

<b>Use-case neve</b>	Teacher drops TVSZ
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az oktató egy szobában tartózkodik és van TVSZ a tárgyai közt, kiírjuk a szoba és az oktató tárgyait. Az oktató eldobja a TVSZ-t a szobába. A TVSZ kikerül az oktató tárgyai közül és bekerül a szoba tárgyai közé. Ismét kiírjuk a szoba és az oktató tárgyait.

<b>Use-case neve</b>	Teacher Places Unpaired Transistor
<b>Kapcsolódó</b>	5.3.56
<b>Rövid leírás</b>	Oktató tranzisztorot dob el
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az oktató egy szobában tartózkodik és van párosítatlan tranzisztor a tárgyai közt, kiírjuk a szoba és az oktató tárgyait. Az oktató eldobja a tranzisztorat a szobába. A tranzisztor kikerül az oktató tárgyai közül és bekerül a szoba tárgyai közé. Ismét kiírjuk a szoba és az oktató tárgyait.

<b>Use-case neve</b>	Student pairs transistors
<b>Kapcsolódó</b>	5.3.36
<b>Rövid leírás</b>	Hallgató két tranzisztorot párosít
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A két tranzisztor pair változónak beállítását a konzolon megjelenítjük

<b>Use-case neve</b>	Cheese timer
<b>Kapcsolódó</b>	5.3.51
<b>Rövid leírás</b>	Hallgató sajtot használ
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A szoba ahol tartózkodik gázossá válik és elindul a sajt időzítője. A szoba állapotváltozását és az időzítő elindulását és hátralevő idejét kiírjuk a konzolra

<b>Use-case neve</b>	Beer timer
<b>Kapcsolódó</b>	5.3.52
<b>Rövid leírás</b>	Hallgató söröt használ
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Ez elindítja a sör időzítőjét. Ennek elindulását és hátralevő idejét kiírjuk a konzolra

<b>Use-case neve</b>	Student teleport after dropping activated transistor
<b>Kapcsolódó</b>	5.3.39
<b>Rövid leírás</b>	Hallgató tranzisztor által teleportál
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A szobaváltást és a tranzisztorok deaktiválását feltüntetjük a konzolon

<b>Use-case neve</b>	Student activates transistor
<b>Kapcsolódó</b>	5.3.37
<b>Rövid leírás</b>	Hallgató tranzisztorot aktivál
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk, a tranzisztor aktiválás függvényét plusz a beállított változót

<b>Use-case neve</b>	Student tries to enter cursed room
<b>Kapcsolódó</b>	5.3.32
<b>Rövid leírás</b>	Hallgató átkozott szobába próbál lépni

<b>Use-case neve</b>	Student tries to enter cursed room
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szoba állapotát, a hallgató jelenlegi szobáját, majd a szobaváltás függvény eredményét

<b>Use-case neve</b>	Student tries to leave cursed room
<b>Kapcsolódó</b>	5.3.38
<b>Rövid leírás</b>	Hallgató átkozott szobából próbál kilépni
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szoba állapotát, a hallgató jelenlegi szobáját, majd a szobaváltás függvény eredményét

<b>Use-case neve</b>	Student enters poisoned room
<b>Kapcsolódó</b>	5.3.35
<b>Rövid leírás</b>	Hallgató gázos szobába lép
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Megjelenítjük a függvényhívást amit a szoba hív a hallgatóra és a szobaváltás eredményét

<b>Use-case neve</b>	Student enters room with other student(s)
<b>Kapcsolódó</b>	5.3.36
<b>Rövid leírás</b>	Hallgató szobába lép, amiben hallgató van
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szoba állapotát, a hallgató jelenlegi szobáját, majd a szobaváltás függvény eredményét

<b>Use-case neve</b>	Student enters room with teacher(s)
<b>Kapcsolódó</b>	5.3.37
<b>Rövid leírás</b>	Hallgató szobába lép, amiben oktató van
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szobaváltás eredményét és a függvényhívást amit a szoba hív a hallgatóra

<b>Use-case neve</b>	Student tries to enter full room
<b>Kapcsolódó</b>	5.3.34
<b>Rövid leírás</b>	Hallgató teli szobába próbál lépni
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szoba állapotát, a hallgató jelenlegi szobáját, majd a szobaváltás függvény eredményét

<b>Use-case neve</b>	Student enters empty room
<b>Kapcsolódó</b>	5.3.33
<b>Rövid leírás</b>	Hallgató üres szobába lép
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szoba állapotát, a hallgató jelenlegi szobáját, majd a szobaváltás függvény eredményét

<b>Use-case neve</b>	Teacher enters room with student(s)
<b>Kapcsolódó</b>	5.3.46
<b>Rövid leírás</b>	Hallgató áll egy szobában majd belép egy oktató. A szoba üzenetet küld a hallgatónak
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A szoba hallgatóra hívott függvényét kiírjuk a konzolra

<b>Use-case neve</b>	Teacher tries to enter cursed room
<b>Kapcsolódó</b>	5.3.42
<b>Rövid leírás</b>	Oktató átkozott szobába próbál lépni

<b>Use-case neve</b>	Teacher tries to enter cursed room
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szoba állapotát, az oktató jelenlegi szobáját, majd a szobaváltás függvény eredményét
<b>Use-case neve</b>	Teacher tries to leave cursed room
<b>Kapcsolódó</b>	5.3.48
<b>Rövid leírás</b>	Oktató átkozott szobából próbál kilépni
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szoba állapotát, az oktató jelenlegi szobáját, majd a szobaváltás függvény eredményét
<b>Use-case neve</b>	Teacher enters poisoned room
<b>Kapcsolódó</b>	5.3.45
<b>Rövid leírás</b>	Oktató gázos szobába lép
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Megjelenítjük a függvényhívást amit a szoba hív az oktatóra és a szobaváltás eredményét
<b>Use-case neve</b>	Teacher enters room with teacher(s)
<b>Kapcsolódó</b>	5.3.47
<b>Rövid leírás</b>	Oktató szobába lép, amiben oktató van
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szobaváltás eredményét a konzolra
<b>Use-case neve</b>	Teacher tries to enter full room
<b>Kapcsolódó</b>	5.3.44
<b>Rövid leírás</b>	Oktató teli szobába próbál lépni
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szoba állapotát, az oktató jelenlegi szobáját, majd a szobaváltás függvény eredményét
<b>Use-case neve</b>	Teacher enters empty room
<b>Kapcsolódó</b>	5.3.45
<b>Rövid leírás</b>	Oktató üres szobába lép
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szoba állapotát, az oktató jelenlegi szobáját, majd a szobaváltás függvény eredményét
<b>Use-case neve</b>	Student's room is poisoned
<b>Kapcsolódó</b>	5.3.40
<b>Rövid leírás</b>	Hallgató szobája gázos lesz
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szoba állapotát, majd gázossá tesszük a szobát. Ismét kiírjuk a szoba állapotát és megjelenítjük a függvényhívást, amit a szoba hív a hallgatóra.
<b>Use-case neve</b>	Student leaves poisoned room
<b>Kapcsolódó</b>	5.3.39
<b>Rövid leírás</b>	Hallgató kilép a gázos szobából
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szobaváltás eredményét a konzolra. Ezen látszani fog, hogy meghívódik a hallgatónak a RoomCleanFromPoison függvénye.
<b>Use-case neve</b>	Student's room is no longer poisoned
<b>Kapcsolódó</b>	5.3.41

<b>Use-case neve</b>	Student's room is no longer poisoned
<b>Rövid leírás</b>	Hallgató szobájáról lekerül a gáz
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szoba állapotát és a hallgató állapotát, majd a gáz effektus leszedése után újra kiírjuk az állapotukat. Kiírjuk ezen kívül a RoomCleanFromPoison függvény hívódását is.

<b>Use-case neve</b>	Teacher's room is poisoned
<b>Kapcsolódó</b>	5.3.50
<b>Rövid leírás</b>	Oktató szobája gázos lesz
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szoba állapotát, majd gázossá tesszük a szobát. Ismét kiírjuk a szoba állapotát és megjelenítjük a függvényhívást, amit a szoba hív az oktatóta.

<b>Use-case neve</b>	Teacher's room is no longer poisoned
<b>Kapcsolódó</b>	5.3.51
<b>Rövid leírás</b>	Oktató szobájáról lekerül a gáz
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiírjuk a szoba állapotát és az oktató állapotát, majd a gáz effektus leszedése után újra kiírjuk az állapotukat. Kiírjuk ezen kívül a RoomCleanFromPoison függvény hívódását is.

<b>Use-case neve</b>	A szoba átkozott állapota a megadott idő után eltűnik
<b>Kapcsolódó</b>	
<b>Rövid leírás</b>	Lejár az átkozottság időzítője, értesíti a szobában tartózkodókat.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A szoba átkozott állapotban van. Miután lejár az időzítő, a megadott visszatérési függvénye meghívódik a szobának. Itt megváltozik a szoba állapota, értesíti a benne lévő karaktereket az állapotváltozásról.

<b>Use-case neve</b>	Szoba merge után a szomszékok helyesen láthatók
<b>Kapcsolódó</b>	5.3.43 : Room merge
<b>Rövid leírás</b>	Szobák összeolvadnak, kettőből egy lesz.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az összeolvadás a két szoba kijáratai összevonja, a második szoba ajtajai bekerülnek az első szoba listájába is.

<b>Use-case neve</b>	Szoba merge után a tárgyak megmaradnak
<b>Kapcsolódó</b>	5.3.43 : Room merge
<b>Rövid leírás</b>	Összeolvadás után összegződnek a tárgyak
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az összeolvadás hatására a második szoba tárgyai is bekerülnek az elsőben.

<b>Use-case neve</b>	Szoba merge után az effektusok összeadódnak
<b>Kapcsolódó</b>	5.3.43 : Room merge
<b>Rövid leírás</b>	Az hatások bekerülnek az összeolvadt szobába.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A hatások áthelyeződnek az összeolvadás tárgy szobájába. Az új állapotot a szoba kezeli, értesíti a szoba résztvevőit.

<b>Use-case neve</b>	Szoba mergel
<b>Kapcsolódó</b>	5.3.43 : Room merge

<b>Use-case neve</b>	Szoba merge
<b>Rövid leírás</b>	Egybeolvadnak az adott szobák.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A szoba átveszi az összeolvadás tárgyát képező szoba szomszédait, tárgyait, hatásait, értesíti a benne lévőket a szobájuk változásáról és a hatásváltozásokról.

<b>Use-case neve</b>	Room cannot merge transistor
<b>Kapcsolódó</b>	5.3.42
<b>Rövid leírás</b>	Szoba nem tud merge-tni tranzisztor miatt
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az összeolvadás a szobaállapotokat ellenőrzi, ha tranzisztor talál, nem hajthatja vége az összeolvadást, sikertelenül tér vissza.

<b>Use-case neve</b>	Room cannot merge student
<b>Kapcsolódó</b>	5.3.41
<b>Rövid leírás</b>	Szoba nem tud merge-tni, mert tartózkodnak benne
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az összeolvadás, ellenőrzi a szobák jelenlévők listáját, ha talál jelenlévőt, nem mehet vége a folyamat, sikertelenül tér vissza.

<b>Use-case neve</b>	Room cannot split transistor
<b>Kapcsolódó</b>	5.3.45
<b>Rövid leírás</b>	Szoba nem tud split-tni tranzisztor miatt
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A split esemény a szoba hatásait ellenőrizve, ha tranzisztor van benne nem hajta vége, sikertelenül tér vissza.

<b>Use-case neve</b>	Room cannot split student
<b>Kapcsolódó</b>	5.3.44
<b>Rövid leírás</b>	Szoba nem tud split-tni, mert tartózkodnak benne.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A szétválás abban az esetben, ha a tartózkodói listában szerepel karakter, nem hajta vége a folyamatot, sikertelenül tér vissza.

<b>Use-case neve</b>	Room split
<b>Kapcsolódó</b>	5.3.46
<b>Rövid leírás</b>	Szoba split-el
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Szétválás hatására egy új szoba keletkezik, szomszédos az eredetivel, osztozik hatásokon, tárgyakon.

<b>Use-case neve</b>	Student saved by beer
<b>Kapcsolódó</b>	5.3.47
<b>Rövid leírás</b>	Sör aktiválódik a hallgatónál, ha találkozik oktatóval.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Szoba értesíti a hallgatót a tanár belépéséről, előrébb lévő sör használódik, időzítője elindul, védelmet nyújt, visszatér a játékoshoz, hogy nem hal meg.

<b>Use-case neve</b>	Student saved by sponge
<b>Kapcsolódó</b>	5.3.48

<b>Use-case neve</b>	Student saved by sponge
<b>Rövid leírás</b>	Lebénítja a szivacs az ellenséges tanárt.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A szoba értesíti a hallgatót. Sorrendben az első szivacs aktiválódik, időzítője elindul. Az oktató értesül a szivacs hatásáról, állapota bénult lesz, a szivacs időzítőnek lejárakor állapot visszaáll. Visszatér a hallgatóhoz lélek eltulajdonító hatása nélkül.

<b>Use-case neve</b>	Student saved by mask
<b>Kapcsolódó</b>	5.3.49
<b>Rövid leírás</b>	Gázos szoba aktiválja a maszkot, nem bénítja el a hallgatót.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A szoba állapotváltozásról értesíti a hallgatót, a sorrendben első masz aktiválódik, időzítője elindul. bénítás nem megy végbe.

<b>Use-case neve</b>	Student saved by TVSZ
<b>Kapcsolódó</b>	5.3.50
<b>Rövid leírás</b>	A tvsz védelmet ad a tanárral szemben
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A szoba értesíti a hallgatót, a sorrendben első tvsz aktiválódik, értéke csökken eggyel. A lélekelvétel nem megy végbe.

<b>Use-case neve</b>	Cheese timer
<b>Kapcsolódó</b>	5.3.51
<b>Rövid leírás</b>	Megtisztul a szoba a gáz hatásától egy adott idő után.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Sajt időzítőjének lejártakor a visszatérő funkció kivesz egy gázos hatást a hatásokból elhasználttá változik. Ha nincs több, állapotát megváltoztatja, értesíti a benne lévőket az állapotváltozásról.

<b>Use-case neve</b>	Beer timer
<b>Kapcsolódó</b>	5.3.52
<b>Rövid leírás</b>	Sör időzítő lejárta után a sör már nem védi meg a hallgatót
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Sör időzítő visszatérési függvénye elhasználttá teszi az eszközt.

<b>Use-case neve</b>	Sponge timer
<b>Kapcsolódó</b>	5.3.53
<b>Rövid leírás</b>	Szivacs időzítő lejárta után a szivacs már nem védi meg a hallgatót
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Szivacs időzítő visszatérési függvénye elhasználttá teszi az eszközt.

<b>Use-case neve</b>	Mask pause
<b>Kapcsolódó</b>	5.3.54
<b>Rövid leírás</b>	Ha a szobáról lekerül a méreg akkor a maszknak meg kell állnia
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Ha a szobáról lekerül a méreg akkor a maszknak meg kell állnia. Ilyenkor a konzolon feltüntetjük a függvényhívást

## 5.2 A szkeleton kezelői felületének terve, dialógusok

A konzolos alkalmazás a program különböző részeinek tesztelésére szolgál.

Ez egy menü ahol sorszámozva vannak a különböző tesztesetek. Ha a felhasználó beüt egy számot akkor lefuttatja az adott tesztet. Ez a menü két részre van osztva a könnyebb átláthatóság érdekében. Először kiválasztunk egy fő pontot majd az a megfelelő almenübe visz minket. Itt lesznek a konkrét tesztesetek.

Ezeknek különböző lefutásuk van. Van amelyik igényel felhasználói beavatkozást illetve olyan is ami nem. A program általában úgy bizonyítja a helyes lefutást, hogy a változók értékeit és annak módosulását illetve a meghívott legfontosabb függvények nevét és az érintett objektumot írja ki.

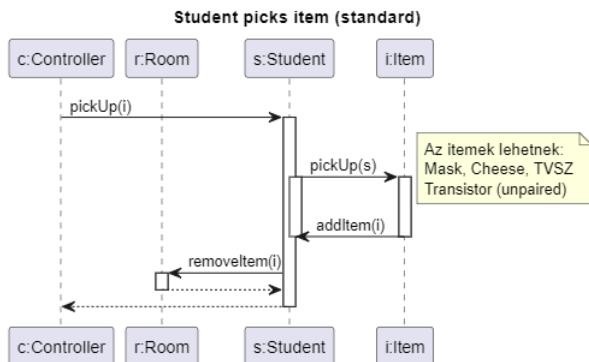
A tervezett formátum ehhez hasonló: {Obj1} - [foo()] -> {Obj2}. Hasonlóan ahogy a szekvencia diagramon is látható.

Amennyiben szükséges a felhasználói beavatkozás, ezeket a program egy kérdés formájában felteszi majd a megfelelő számot begépelve ki tudjuk választani a kérdéses tesztet. Az összes tesztesetet külön diagramon vettük fel, ez az opció választás arra szolgál, hogy ne legyen ilyen sok külön teszteset és ezzel együtt rengeteg almenü pont hanem egy tesztnél lehessen két külön helyzetet bemutatni.

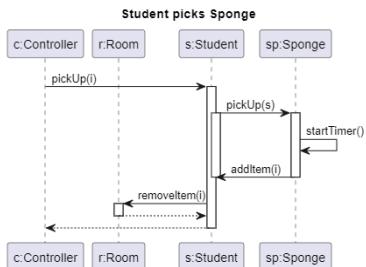
A teszteket egy kontroller nevű osztály fogja menedzselni. Ő az aki mindenkit elér és minden függvényt meg tud hívni az objektumokra

## 5.3 Szekvencia diagramok a belső működésre

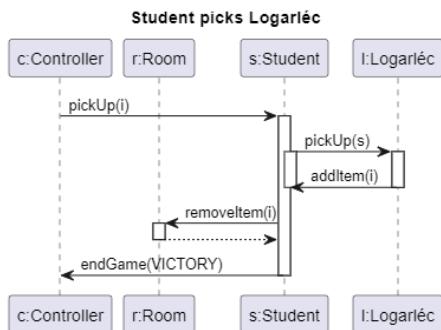
### 5.3.1 Student picks Item (standard)



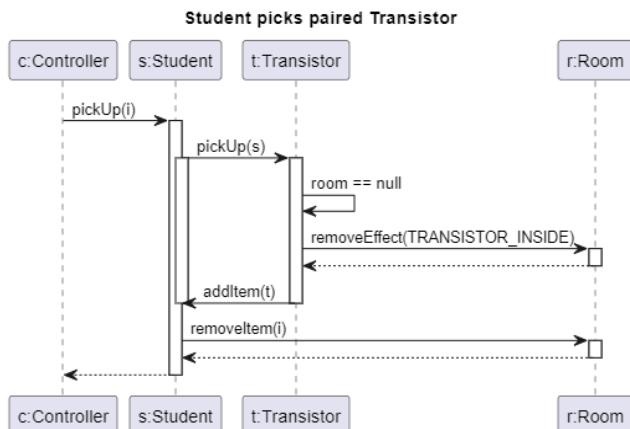
### 5.3.2 Student picks Sponge



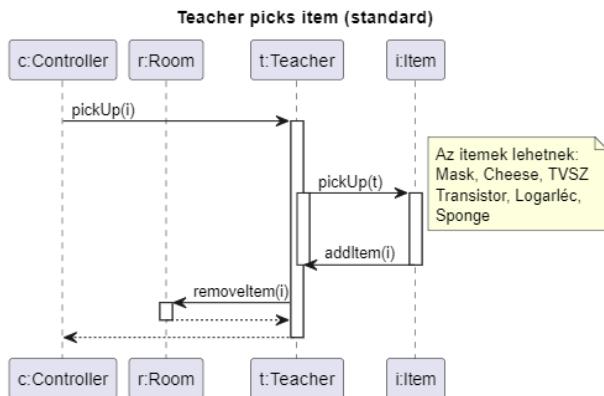
### 5.3.3 Student picks Logarléc



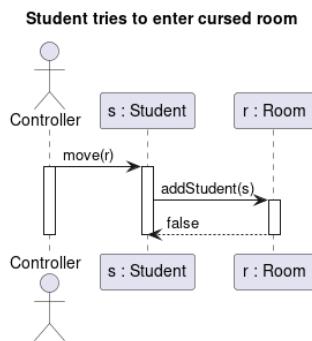
### 5.3.4 Student picks Paired transistor



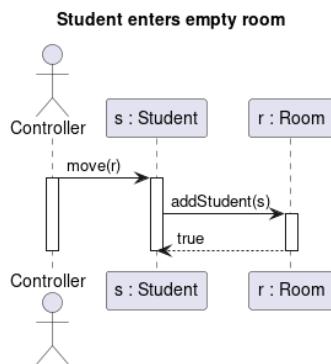
### 5.3.5 Teacher picks item (standard)



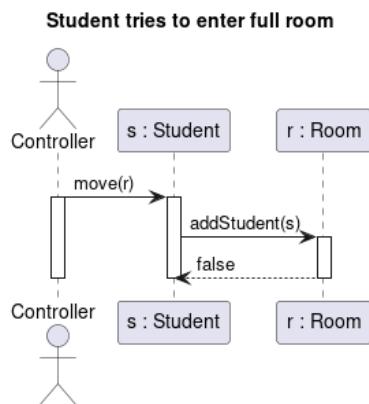
### 5.3.6 Student tries to enter cursed room



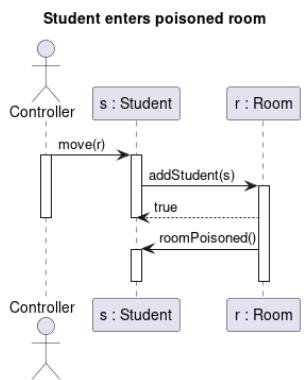
### 5.3.7 Student enters empty room



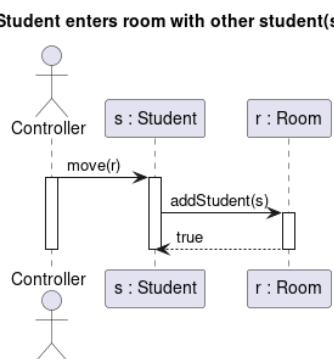
### 5.3.8 Student tries to enter full room



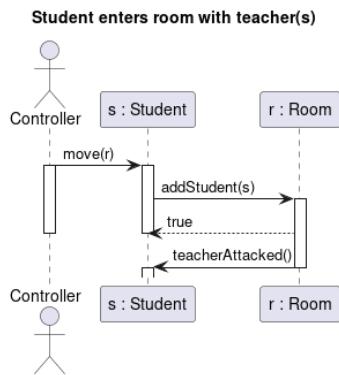
### 5.3.9 Student enters poisoned room



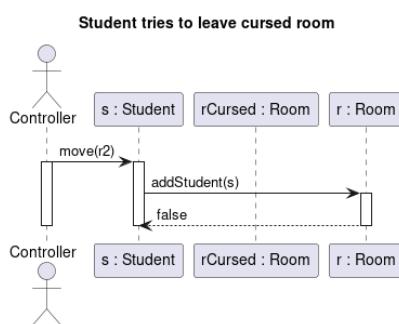
### 5.3.10 Student enters room with other student(s)



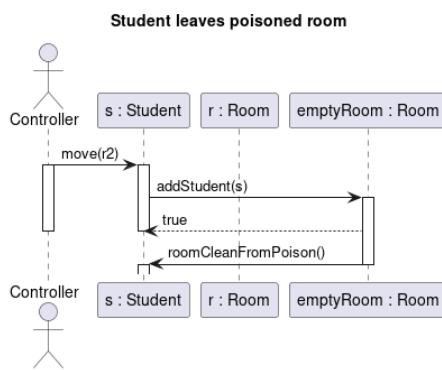
### 5.3.11 Student enters room with teacher(s)



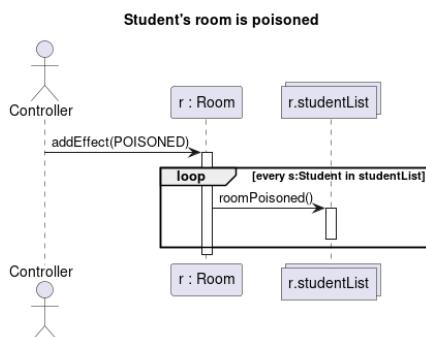
### 5.3.12 Student tries to leave cursed room



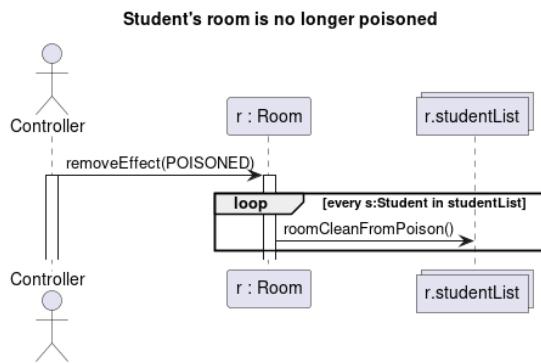
### 5.3.13 Student leaves poisoned room



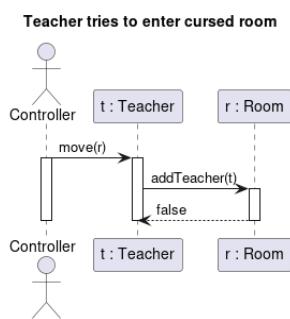
### 5.3.14 Student's room is poisoned



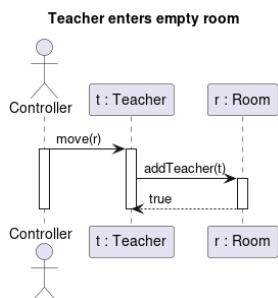
### 5.3.15 Student's room is no longer poisoned



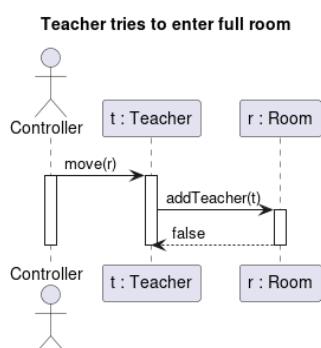
### 5.3.16 Teacher tries to enter cursed room



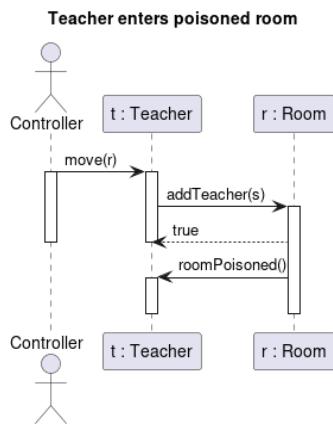
### 5.3.17 Teacher enters empty room



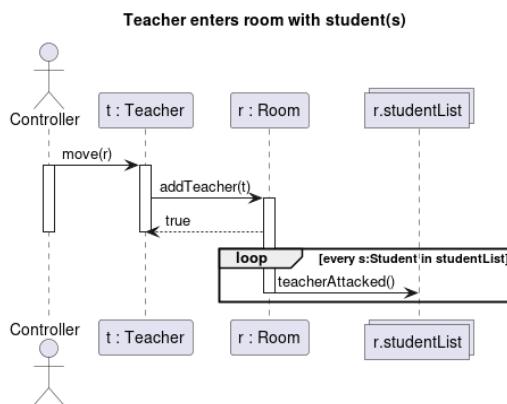
### 5.3.18 Teacher tries to enter full room



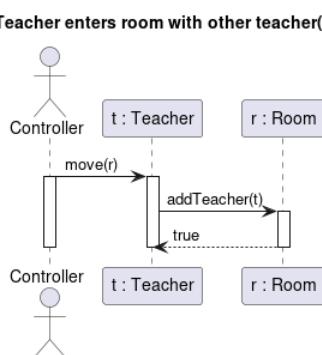
### 5.3.19 Teacher enters poisoned room



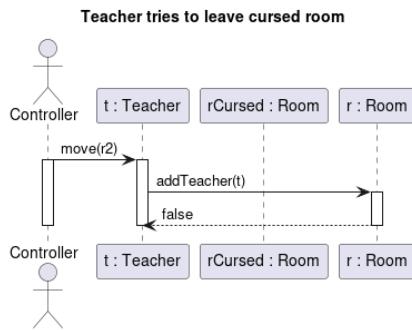
### 5.3.20 Teacher enters room with student(s)



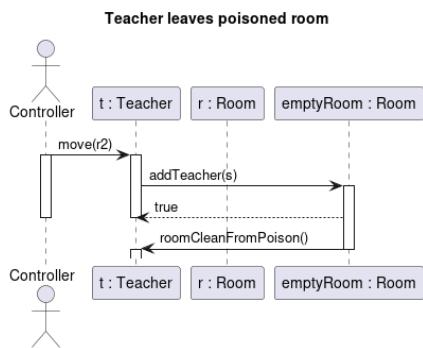
### 5.3.21 Teacher enters room with other teacher(s)



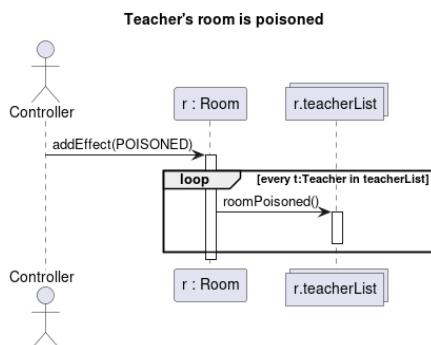
### 5.3.22 Teacher tries to leave cursed room



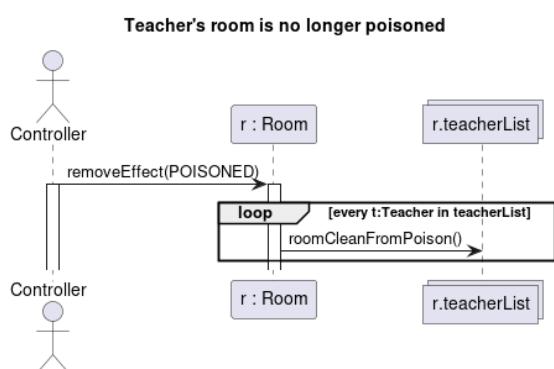
### 5.3.23 Teacher leaves poisoned room



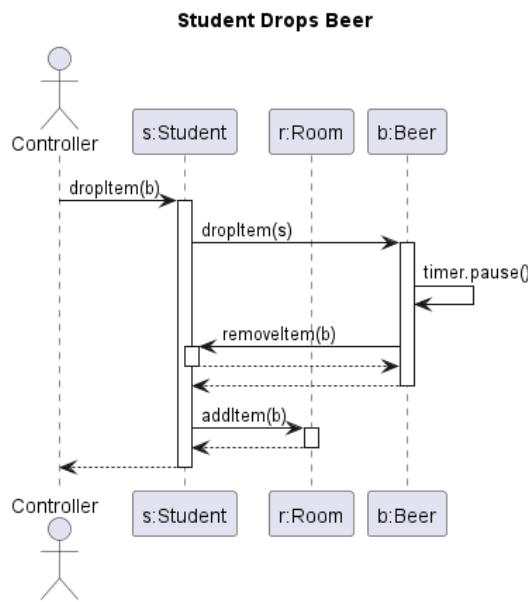
### 5.3.24 Teacher's room is poisoned



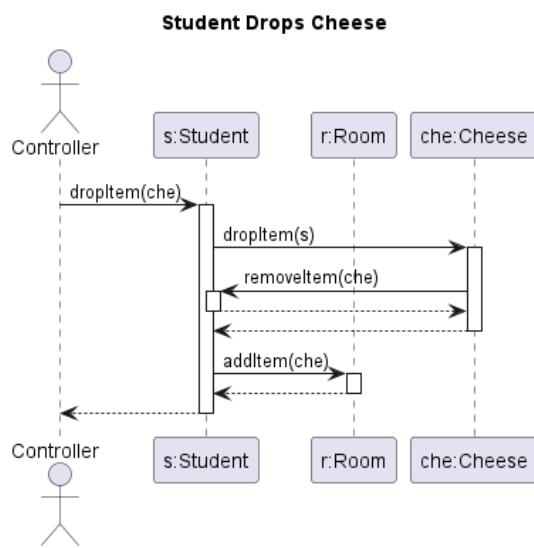
### 5.3.25 Teacher's room is no longer poisoned



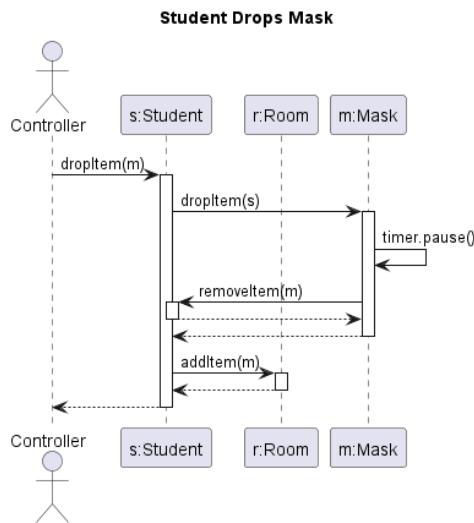
### 5.3.26 Student drops beer



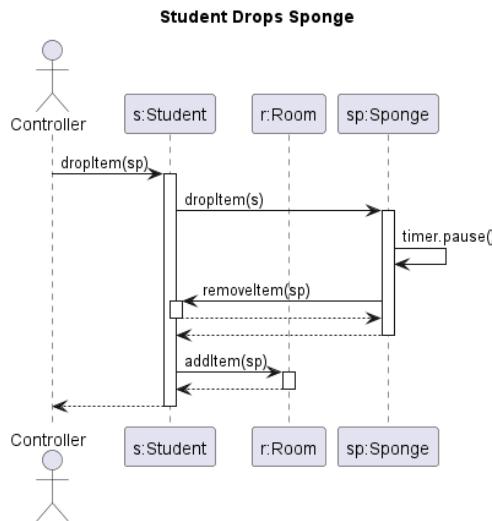
### 5.3.27 Student drops cheese



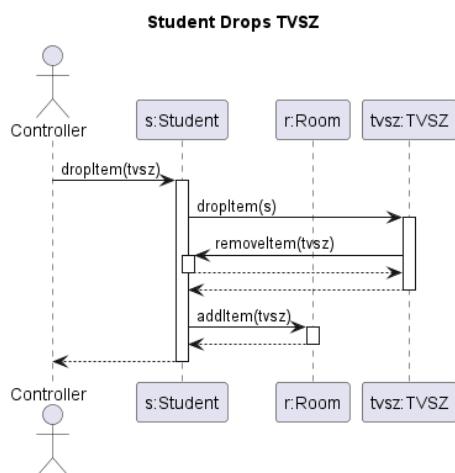
### 5.3.28 Student drops mask



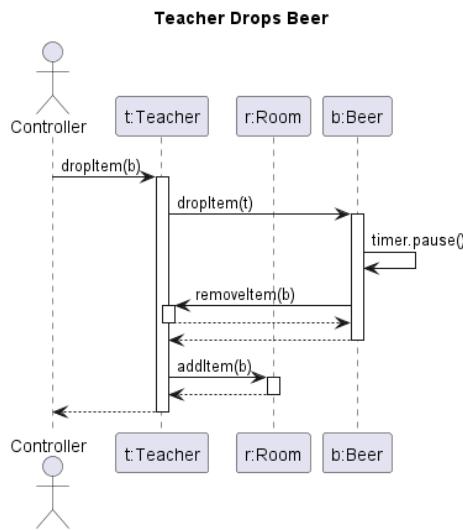
### 5.3.29 Student drops sponge



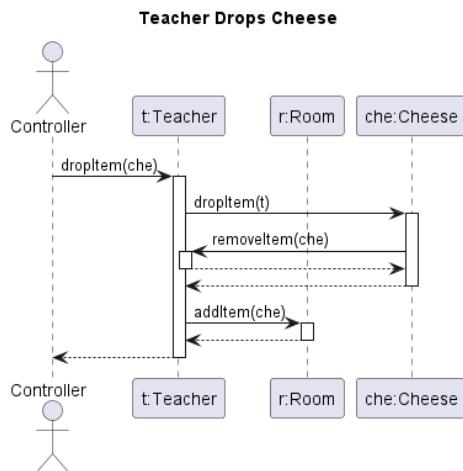
### 5.3.30 Student drops TVSZ



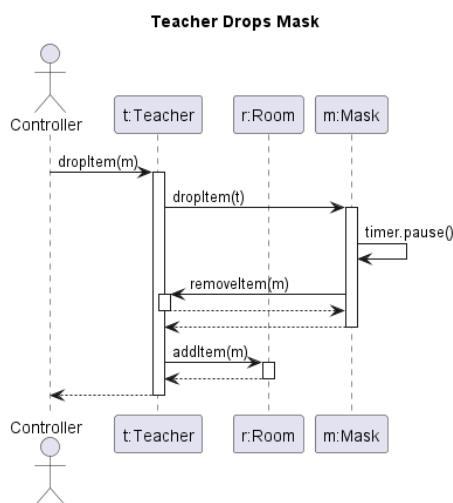
### 5.3.31 Teacher drops beer



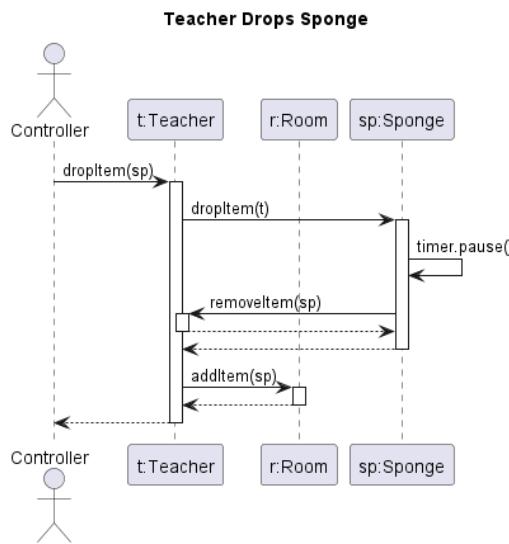
### 5.3.32 Teacher drops cheese



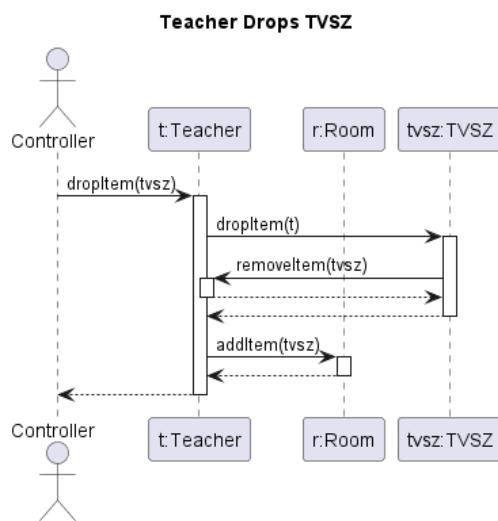
### 5.3.33 Teacher drops mask



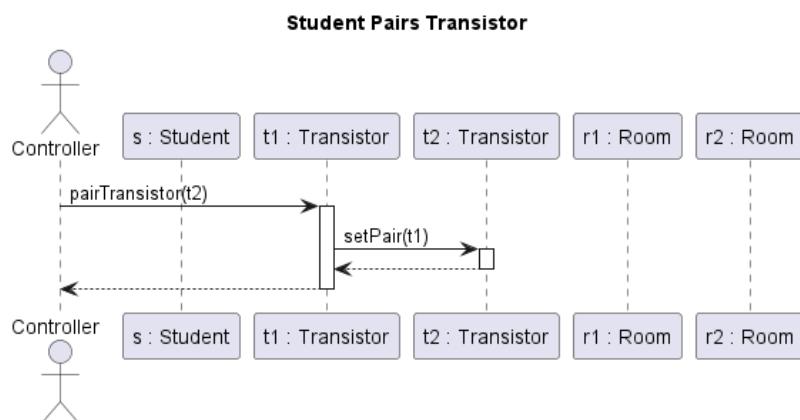
### 5.3.34 Teacher drops sponge



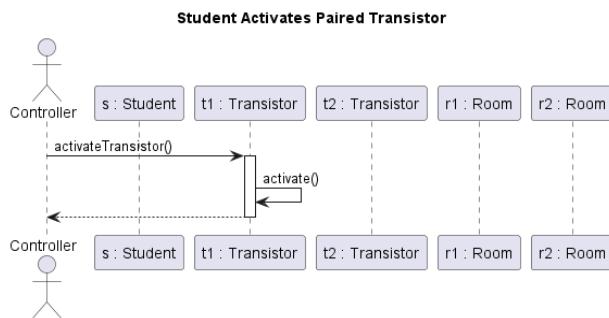
### 5.3.35 Teacher drops TVSZ



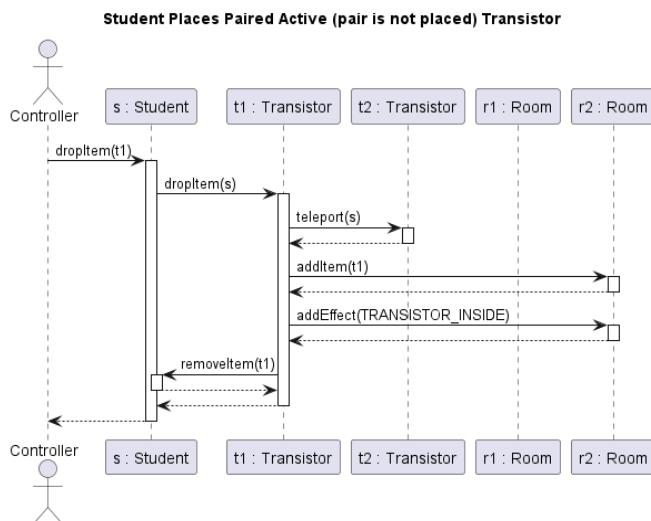
### 5.3.36 Student pairs transistors



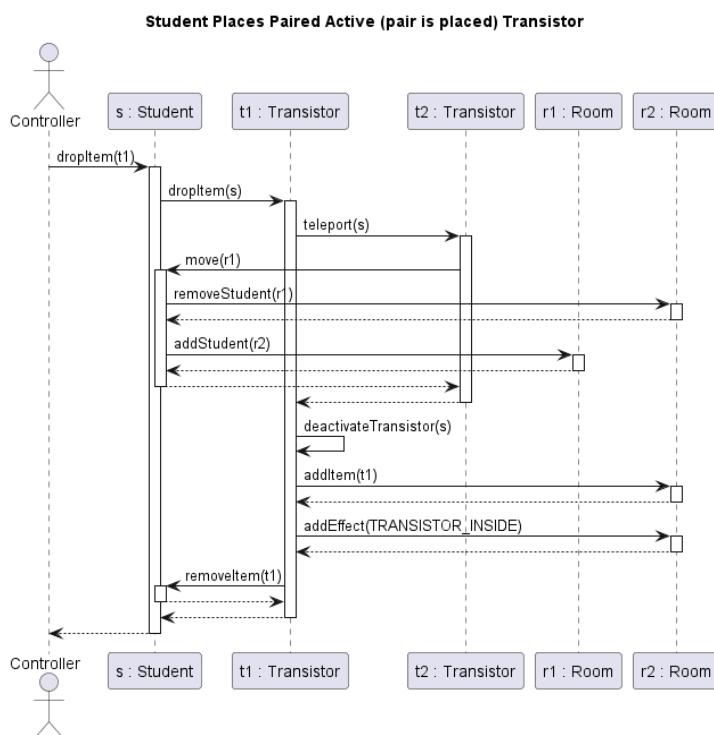
### 5.3.37 Student activates transistor



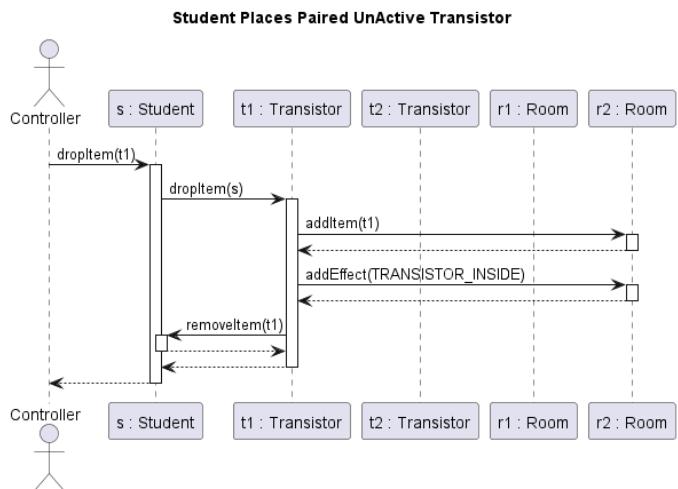
### 5.3.38 Student places activated transistor but pair is not placed



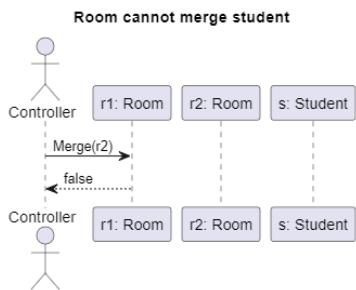
### 5.3.39 Student teleport after dropping it



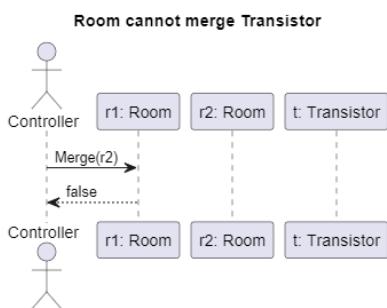
### 5.3.40 Student places inactive or unpaired transistor



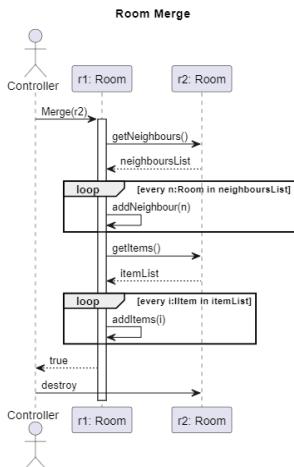
### 5.3.41 Room cannot merge student



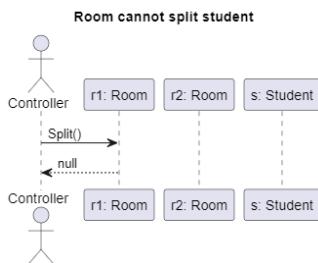
### 5.3.42 Room cannot merge transistor



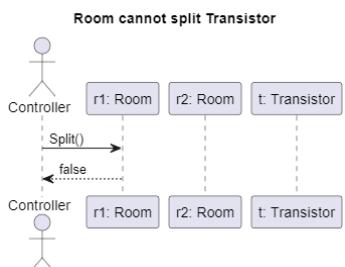
### 5.3.43 Room merge



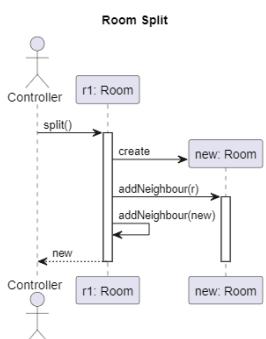
### 5.3.44 Room cannot split student



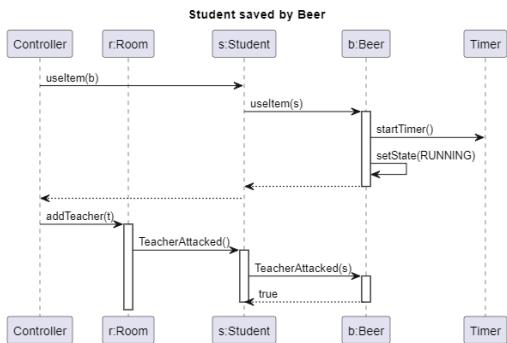
### 5.3.45 Room cannot split transistor



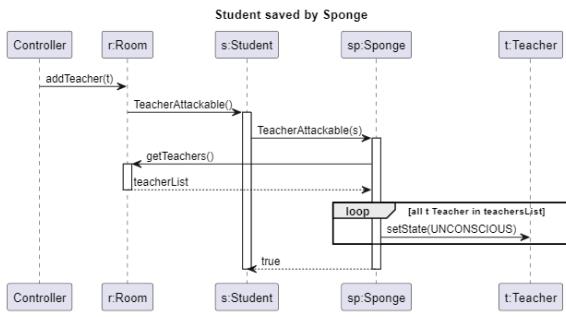
### 5.3.46 Room split



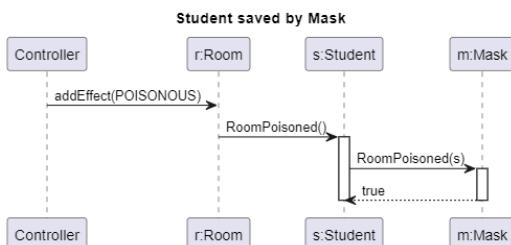
### 5.3.47 Student saved by Beer



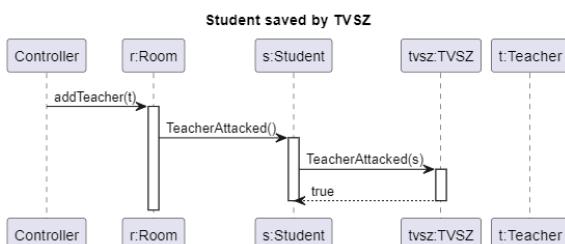
### 5.3.48 Student saved by Sponge



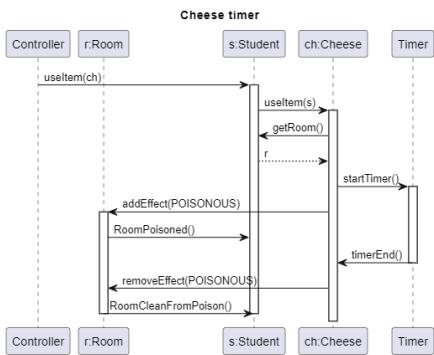
### 5.3.49 Student saved by Mask



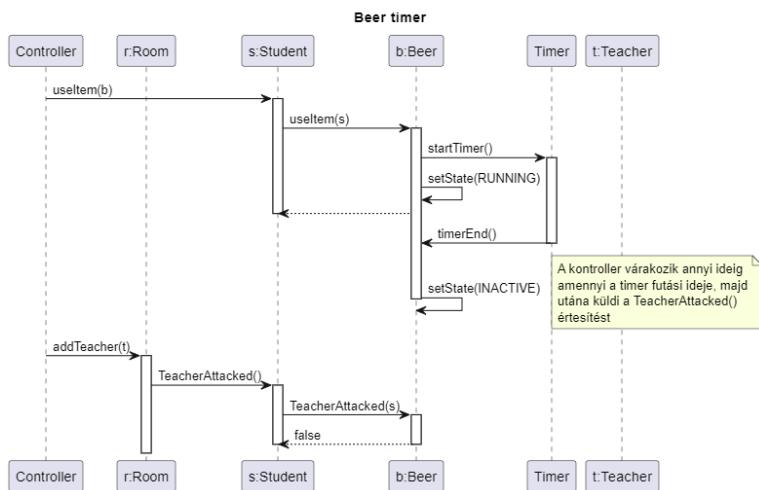
### 5.3.50 Student saved by TVSZ



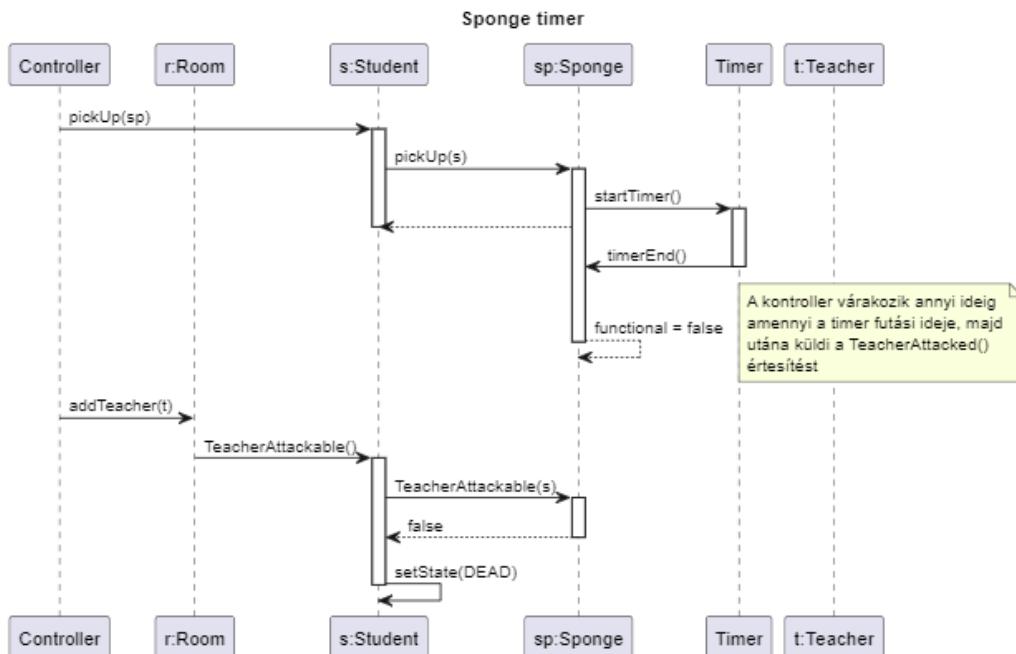
### 5.3.51 Cheese timer



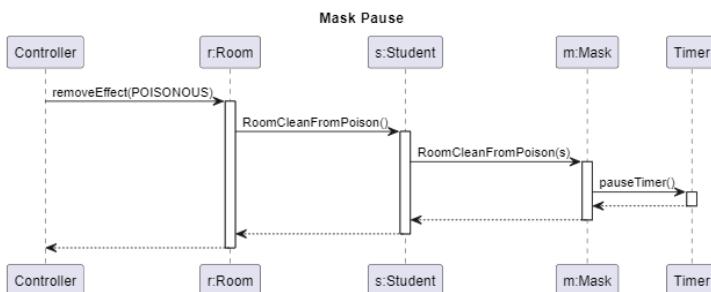
### 5.3.52 Beer timer



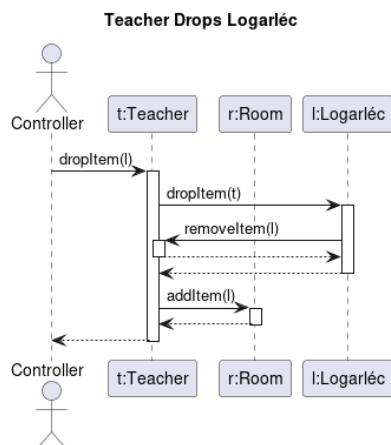
### 5.3.53 Sponge timer



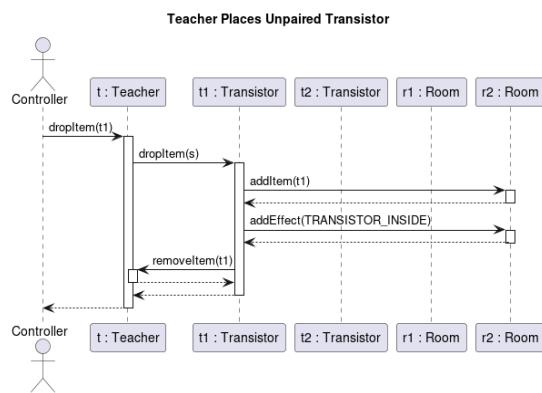
### 5.3.54 Mask pause



### 5.3.55 Teacher Drops Logarléc

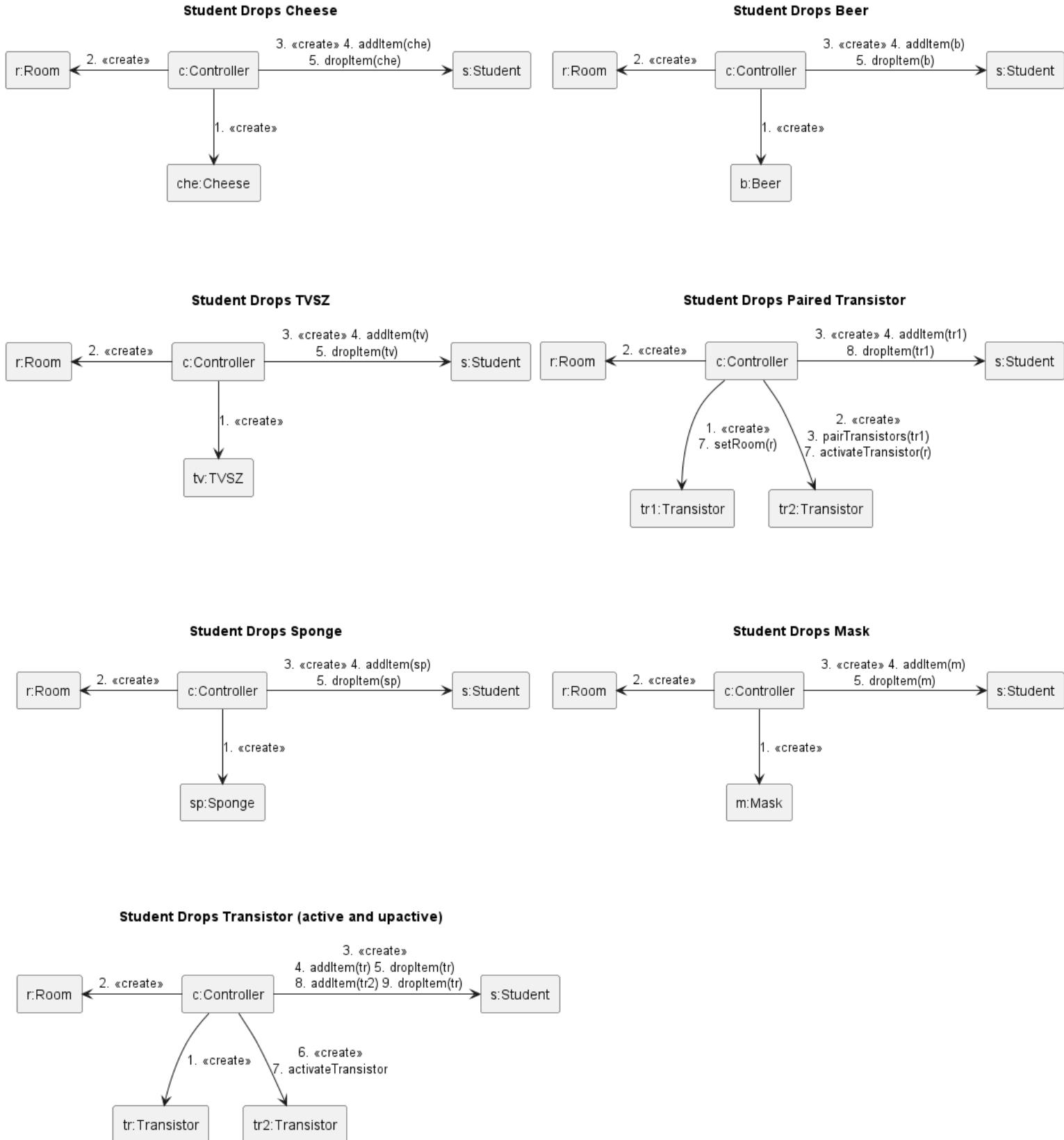


### 5.3.56 Teacher Places Unpaired Transistor

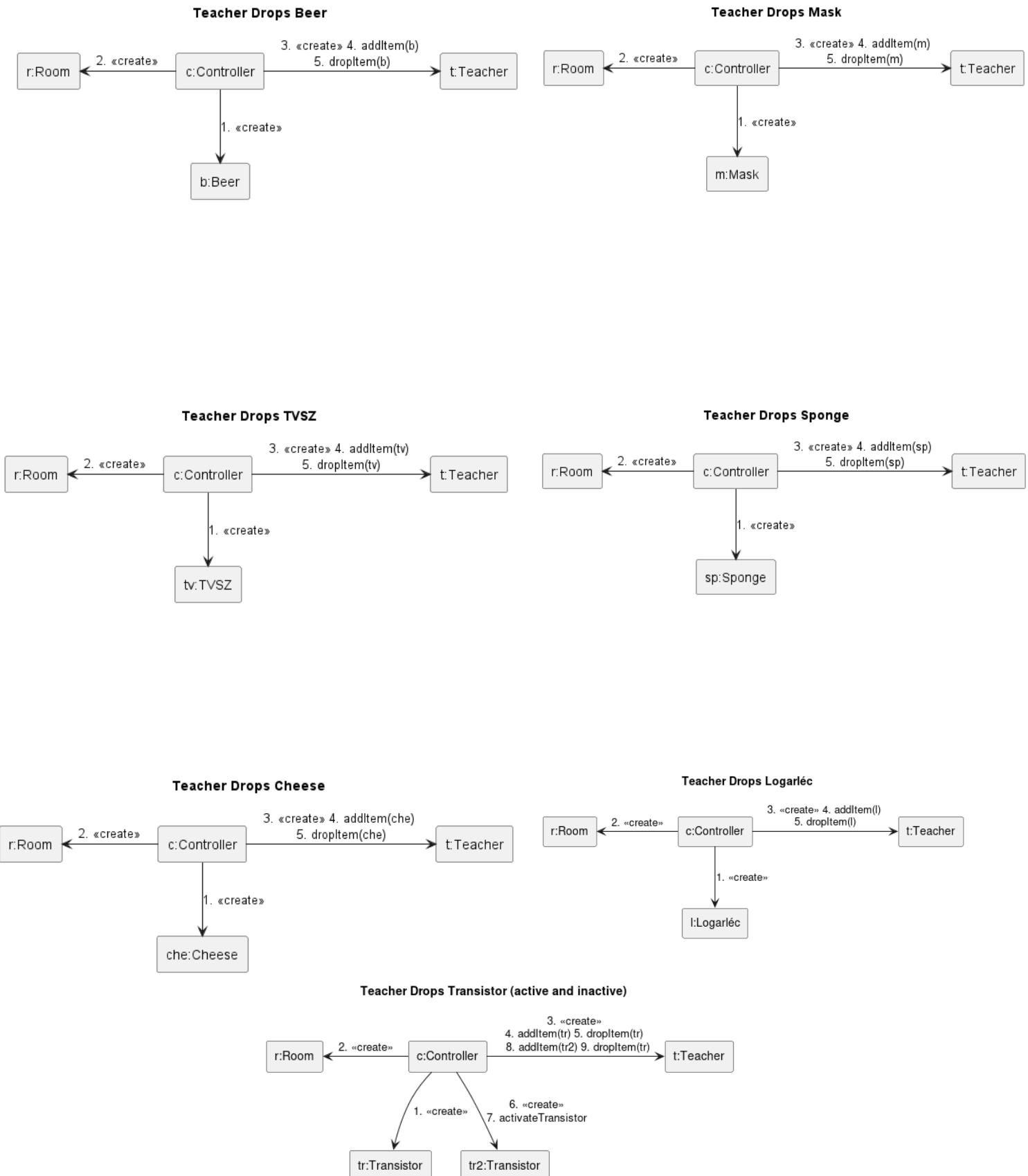


## 5.4 Kommunikációs diagramok

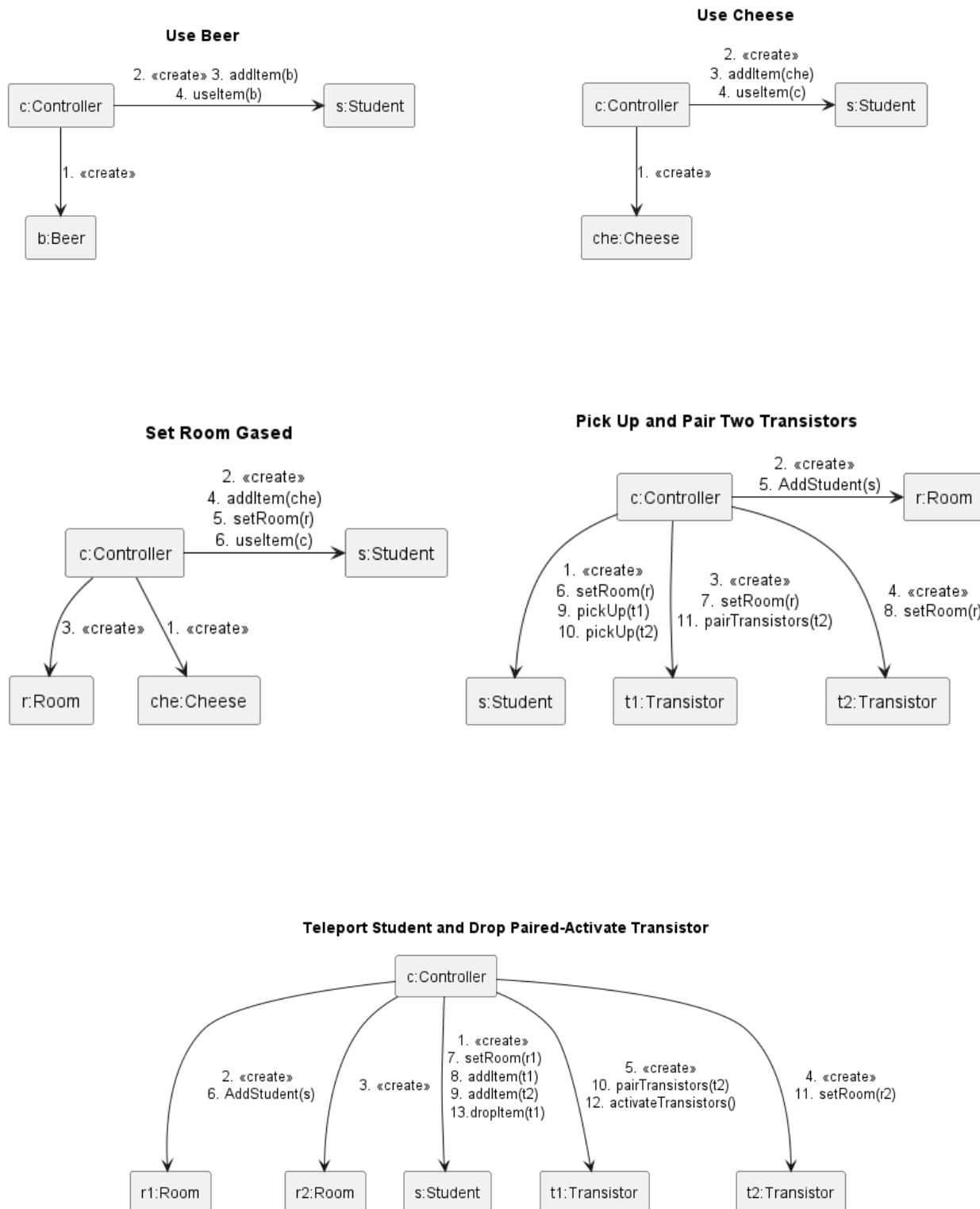
### 5.4.1 Student drops items



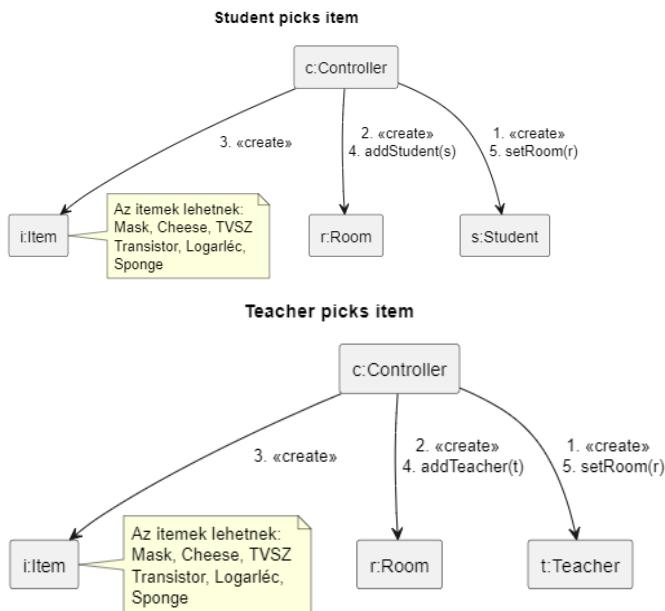
## 5.4.2 Teacher Drops items



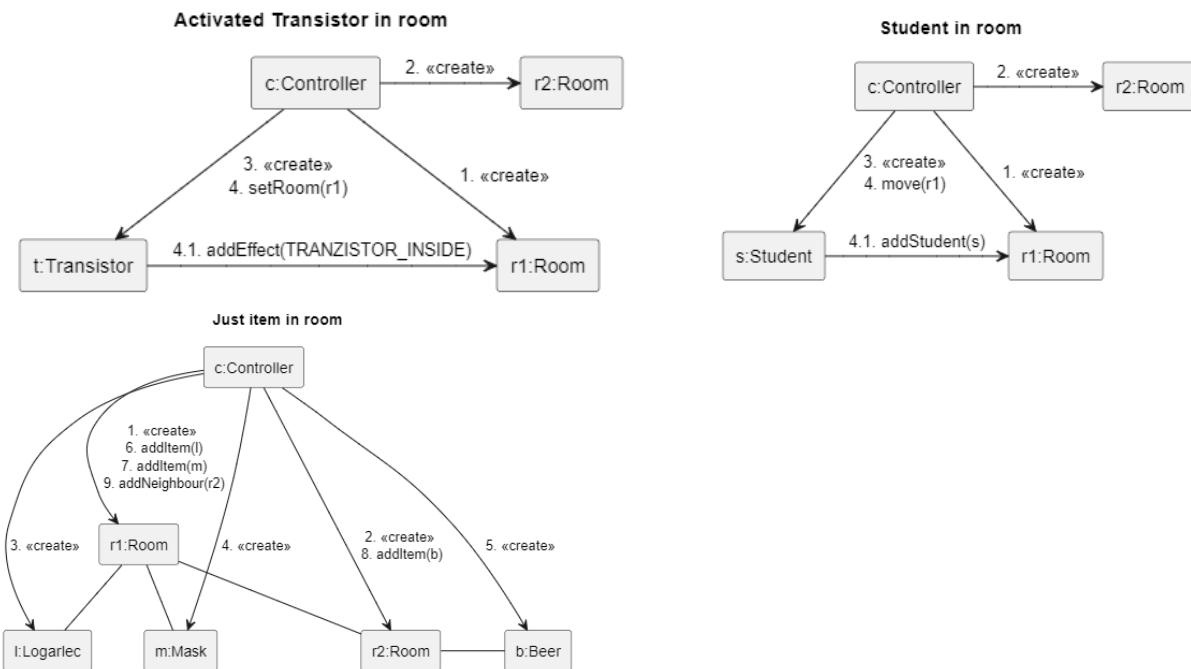
### 5.4.3 Student use



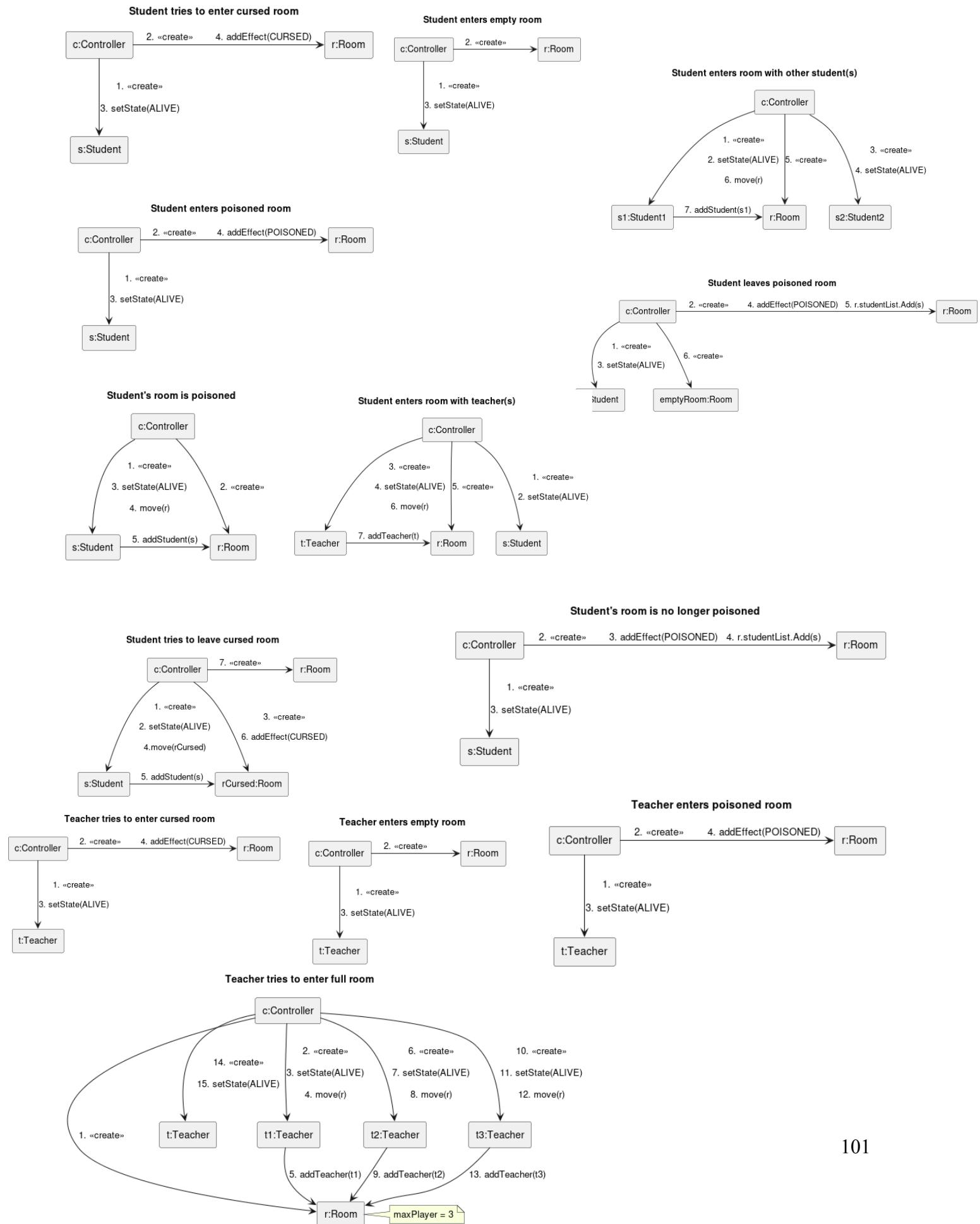
### 5.4.4 Pick Item

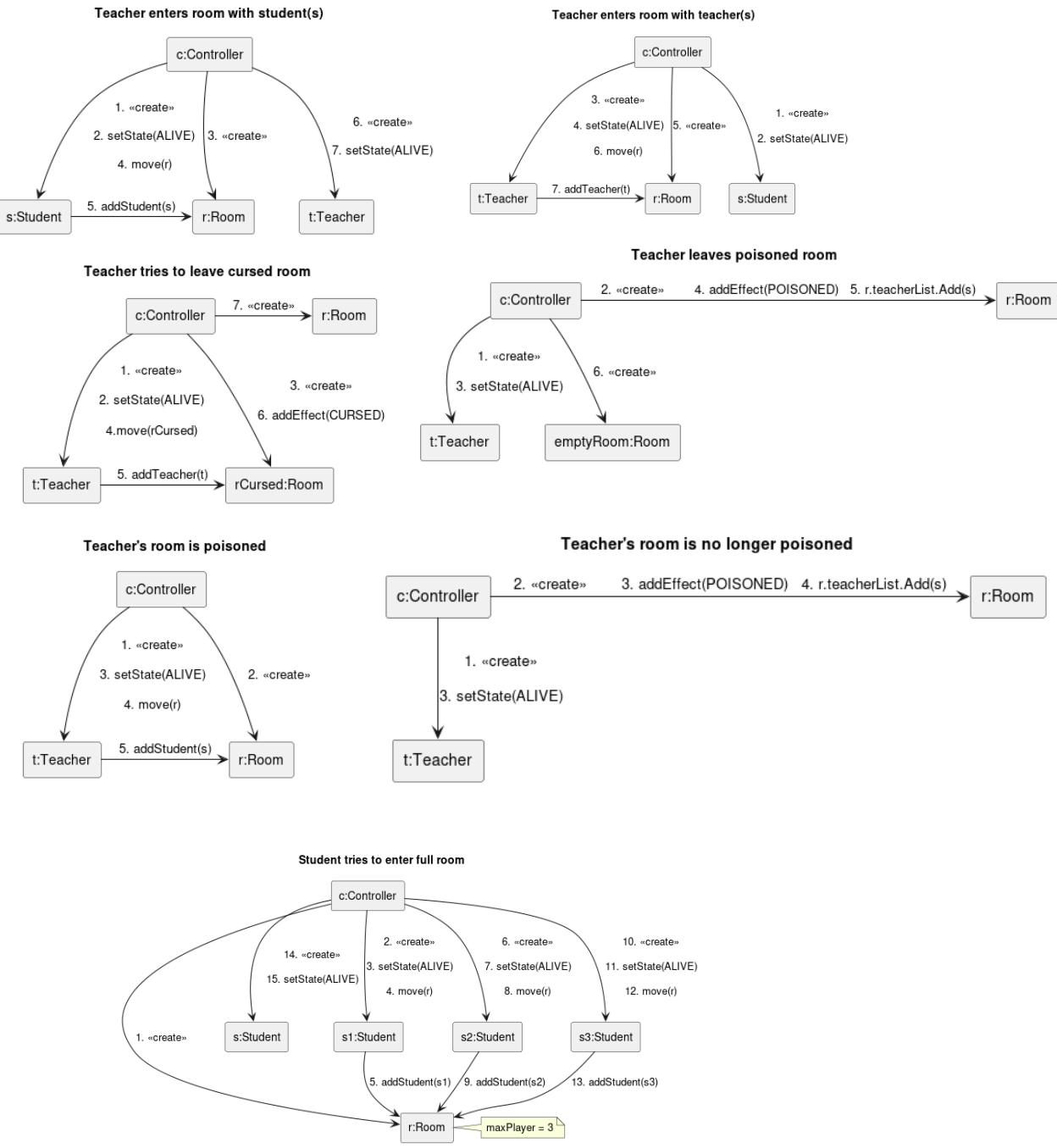


### 5.4.5 Room functionality

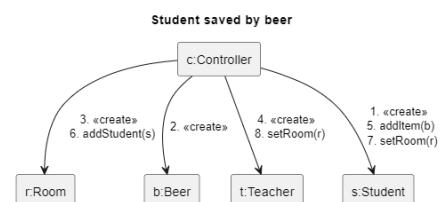


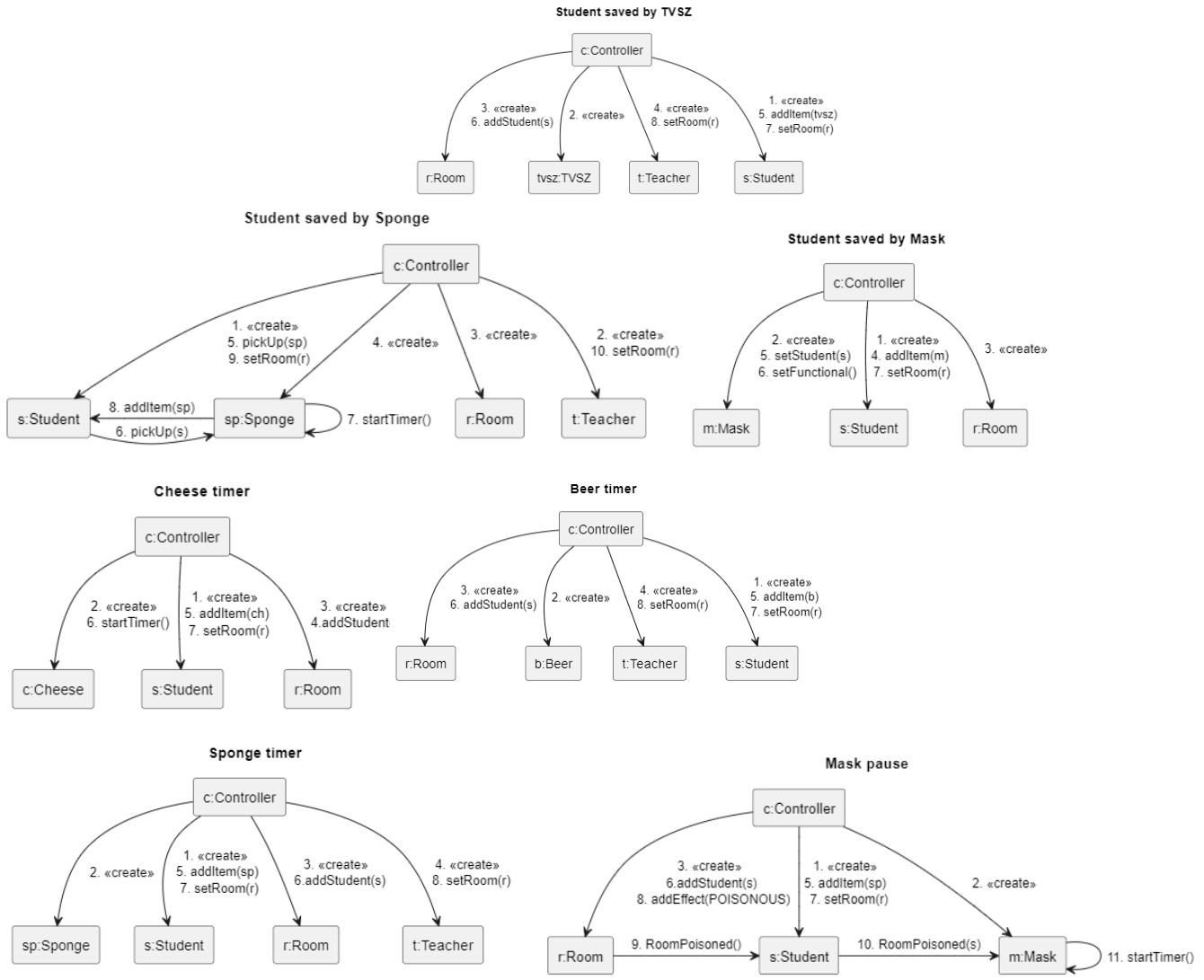
## 5.4.6 Room Interactions





## 5.4.7 Item functionality





## 6. Szkeleton beadás

### Változtatások

- A “Szoba Interakciók” kommunikációs diagrammjaíhoz képest egy bővítés található a szkeletonban: egy alap szobát beállítunk a játékosoknak.
- A Room osztály kibővült `setMax()` és `getMax()` függvényekkel, ezekkel a szoba kapacitását lehet megadni vagy lekérdezni.
- Eldobásnál, a szoba hozzáadásával ki lett egészítve.
- Transistoros példáknál player szobálya hozzá lett adva.
- Use Cheese esetén a Cheese végzi a tárgy eltávolítását.

### Item functionality

- Sponge save kommunikációs diagramban c->r: `addItem(sp)`
- Mask save c->r: `addStudent(s)`
- Cheese timer és cheese\_timer\_end szekvenciadiagram különválasztva

### 6.1 Fordítási és futtatási útmutató

#### 6.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
controller/			
DropItem.java	11 KB		
ItemFunctTests.java	12 KB		
ITestcase.java	1 KB		
Labyrinth.java	1 KB		
Room functionality.java	5 KB		
Room Interactions.java	19 KB		
Skeleton.java	7 KB		
Timer.java	2 KB		
UseItem.java	5 KB		
model/			
ERoomEffects.java	1 KB		
IRoomManager.java	1 KB		
ITimer.java	1 KB		
Room.java	9 KB		
model/items			
Beer.java	3 KB		
Cheese.java	3 KB		
EBeerState.java	1 KB		
IItem.java	1 KB		
Logarlec.java	2 KB		
Mask.java	3 KB		
Sponge.java	3 KB		
Transistor.java	4 KB		
TVSZ.java	2 KB		
model/player			
EPlayerState.java	1 KB		

Player.java	3 KB		
Student.java	3 KB		
Teacher.java	2 KB		

### **6.1.2 Fordítás**

A Skeleton fordításához a Visual Studio Code-ot használjuk, az “Extension Pack for Java” kiegészítővel. VSCode-on belül fordításhoz és futtatáshoz is ezt használjuk. A programot a vscode-ba épített terminálban, a kiegészítő által biztosított automatikus fordítással indítjuk el.

Openjdk 11 szükséges a fordításhoz.

### **6.1.3 Futtatás**

A futtatáshoz openjdk 11 szükséges.

A futtatás Visual Studio Code-ból történik, az előző pontban megfogalmazottak szerint.

## 7. Prototípus koncepciója

### 7.0 Változás hatása a modellre

Söröskorsó új hatása:

Új szabály a korsó osztály használatakor.

Légrissítő (Tárgy):

Új tárgy, új hatás, Itemet interface-t implementálja.

Takarító (Karakter):

Szellőzetés az új Cleaner osztályban, Player leszármazottja.

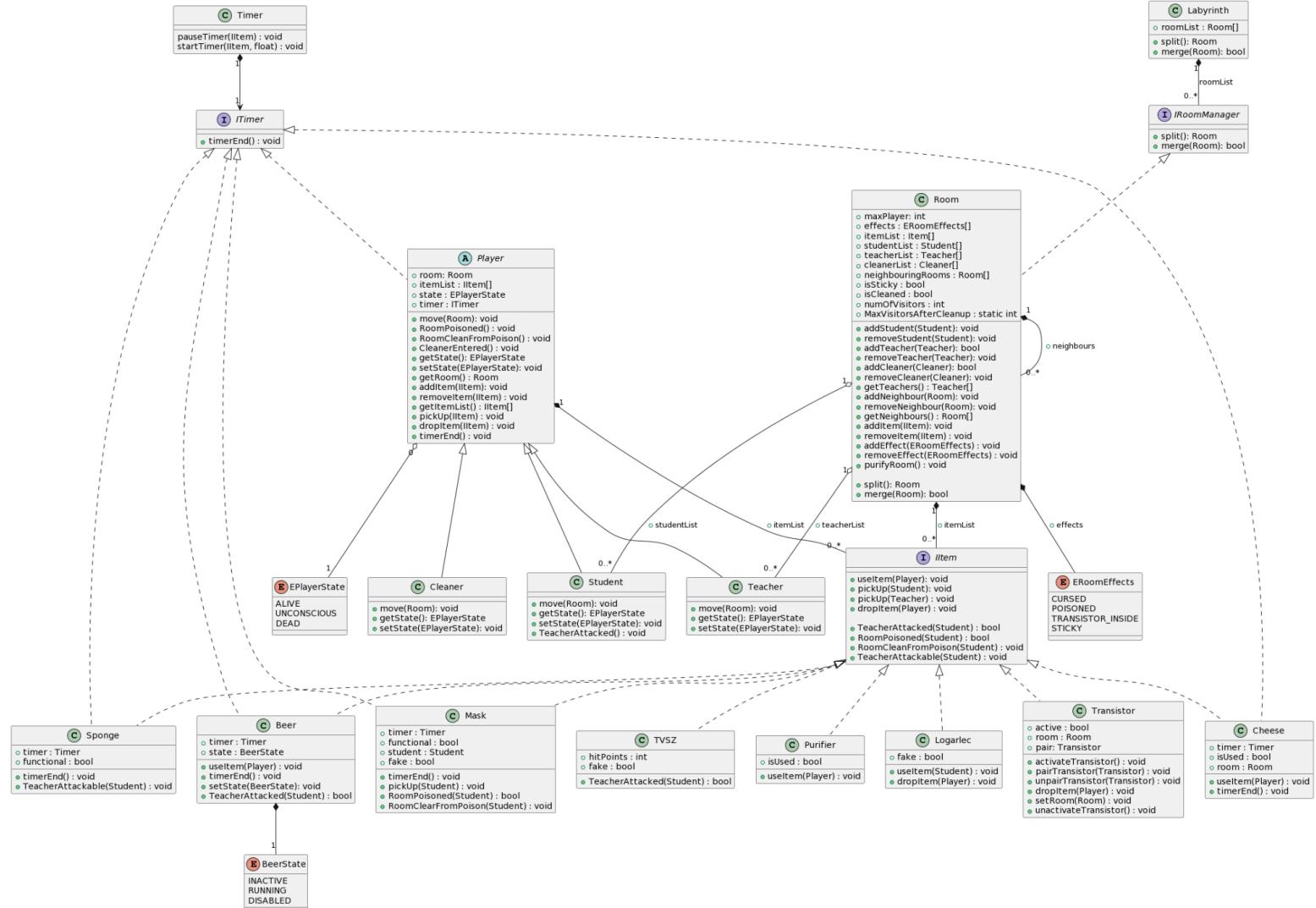
Ragacsosság:

A szobának egy új állapota, miután takarították a szobát.

Hamis tárgyak:

Hamis tárgyat egy boolean változóval különböztetjük meg a valódi tárgyaktól.

### 7.1.1 Módosult osztálydiagram



## 7.1.2 Új vagy megváltozó metódusok

### Room

- **Attribútumok**
  - **cleanerList** - A szobában tartózkodó takarítók listája
  - **isSticky** - Igaz értéknél nem vehetők fel a tárgyak
  - **isCleaned** - Igaz értéknél egy takarító már járt a szobában
  - **numOfVisitors** - Takarítás után eddig belépett hallgatók és oktatók száma
  - **MaxVisitorsAfterCleanup** - statikus érték. A takarítás után a maximum látogatható hallgatók és oktatók száma mielőtt a szoba ragacsossá (**isSticky=true**) válna
- **Metódusok**
  - **void addCleaner(Cleaner)** - Egy takarítót tesz a szobába, majd **purifyRoom**-ot hív
  - **void removeCleaner(Cleaner)** - Eltávolít egy takarítót a szobából
  - **void purifyRoom()** - Az összes gázos hatást leveszi a szobáról, majd értesíti a bent tartózkodó hallgatókat és oktatókat

### Purifier (új)

- **Felelősség**
  - A légrissítő működését valósítja meg.
- **Interfészek**
  - Item
- **Attribútumok**
  - **isUsed** - Ha igaz akkor a tárgy már használva lett így nem használható többé
- **Metódusok**
  - **void useItem(Student)** - Egy hallgató ha használja a tárgyat akkor az aktuális szobájának meghívja a **purifyRoom** függvényét ezzel levéve az összes gázos hatást róla

### Player

- **Metódusok**
  - **void CleanerEntered()** - Akkor hívódik meg ha egy takarító belépett abba a szobába ahol tartózkodik a játékos (A takarító nem implementálja). Ha nem UNCONSCIOUS, akkor megpróbál átlépni egy szomszédos szobába. Ha nem tud akkor ott marad

### Cleaner (új)

- **Felelősség**
  - Megvalósítja a takarító működését. Automatikusan van vezérelve ahogy az oktató is.
- **Interfészek**
  - Player

### • Metódusok

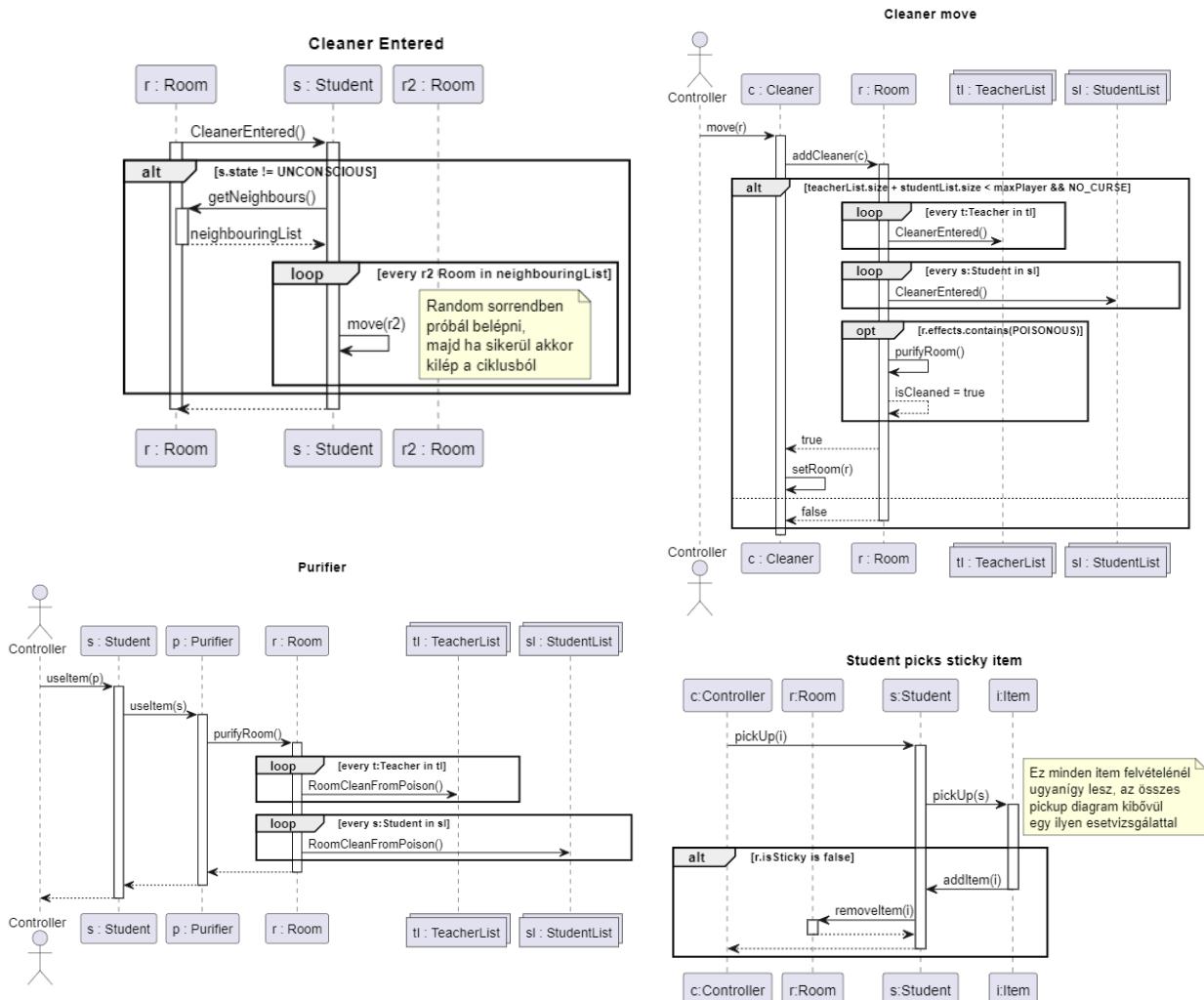
- **void move(Room r)**: A játékos megpróbál lépni az r szobába. Ha nem tud, nem történik semmi.
- **EPlayerState getState()**: Lekérhetjük a játékos jelenlegi állapotát.
- **void setState(EPlayerState s)**: Beállíthatjuk a játékos állapotát.

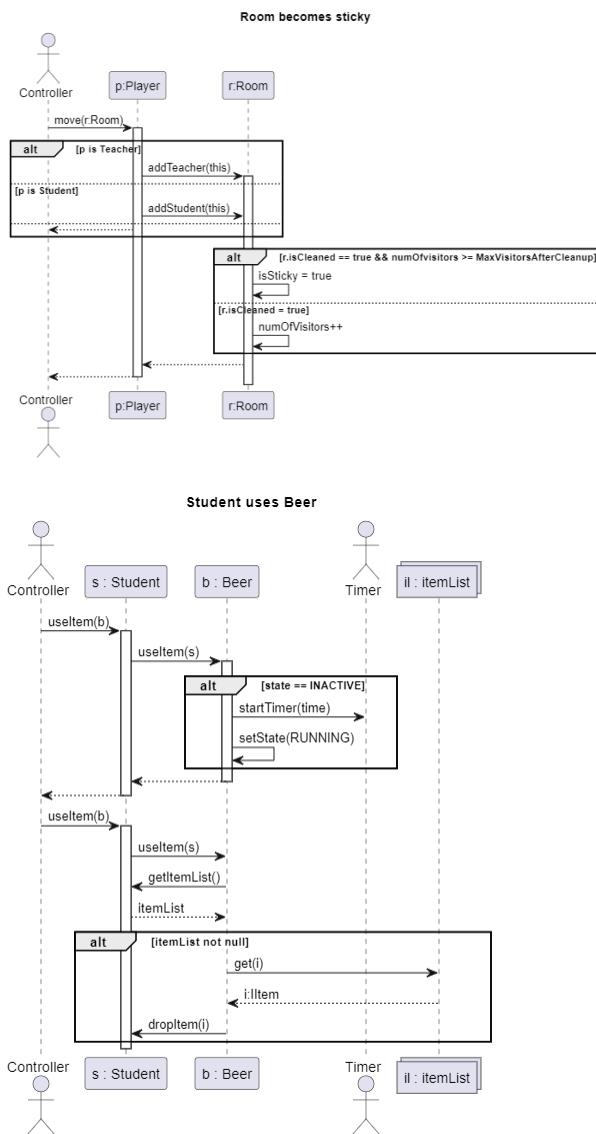
## Mask, TVSZ, Logarléc

### • Attribútumok

- **fake** - Amennyiben ez igaz, a tárgyat nem lehet használni. Egyes tárgyaknál ez azt eredményezi, hogy:
  - **Mask**: A RoomPoisoned-re minden false a válasza. A RoomCleanFromPoison-re nem reagál semmit hiszen időzítője sincs amit elindított volna
  - **TVSZ**: Mintha a hitpointja kezdetben 0 volna
  - **Logarléc**: Felvételnél nem küld a kontrollernek endGame üzenetet

### 7.1.3 Szekvencia-diagramok





## 7.2 Prototípus interface-definíciója

### 7.2.1 Az interfész általános leírása

A prototípus program egy parancssoron keresztül vezérelhető alkalmazás lesz.

A program képes lesz fájlból játék állapotot betölteni, és játék állapotot fájlba menteni.

A következő parancsokat lesz képes fogadni a program, melyek leírása lentebb található:

```
load <<fájlnév>>
save <<fájlnév>>
create <<objektum>> <<név>>
link <<kivel>> <<kivel>>
state <<objektum>> <<állapot>> <<opcionális>>
control <<mit>> <<mivel>>
interact <<drop|pick|use|...>> <<ki>> <<mit>> <<mit>>
stat
```

### 7.2.2 Bemeneti nyelv

#### load [FILE]

##### Leírás:

Egy megadott nevű fájlból lehet a bemenetet beolvasni.

##### Opciók:

FILE: Egy txt fájl teljes neve.

A txt fájl a bemeneti nyelv parancsait tartalmazza, amelyek egymás után fognak sorban lefutni.

##### Kimenetek:

Sikeres beolvasás esetén nem ír ki semmit, amennyiben a fájl nem létezik akkor kiírja ezt

#### save [FILE]

##### Leírás:

Egy megadott nevű fájlba lehet a kimenetet kiírni.

##### Opciók:

FILE: Egy txt fájl teljes neve.

A save parancs a kimeneteinken kívül az összes eddigi beírt parancsot is elmenti egy state.txt nevű fájlba.

##### Kimenetek:

Sikeres írás esetén nem ír ki semmit, amennyiben a fájl már létezik, felülírja azt.

#### create [OBJECT] [NAME]

##### Leírás:

A create parancssal a játék különböző objektumait hozhatjuk létre, saját elnevezéssel.

##### Opciók:

Ha a create parancsot önmagában adjuk meg, akkor kilistázza a létrehozható objektumokat.

Lehetséges opciók az [OBJECT] helyére: Room, Student, Teacher, Cleaner, Sponge, Beer, Mask, TVSZ, Purifier, Logarlec, Transistor, Cheese.

Ha nem ezek közül választunk, akkor a parancs hibát fog dobni.

Névnek bármilyen egybefüggő karakter sorozatot megadhatunk.

##### Kimenetek:

Opciók listázása create parancs kiadása után:

```
>create
Room
Student
```

...

A create parancs sikeres lefutása után kiírásra kerül a létrehozott objektum neve és típusa. Példa egy szoba létrehozására:

```
>create Room LetrehozottSzoba
Room LetrehozottSzoba created
```

## link [NAME1] [NAME2]

### Leírás:

A link parancsot két általunk létrehozott objektum összekapcsolásához használhatjuk.

### Opciók:

hogy Egy NAME egy általunk létrehozott objektum általunk megadott neve kell, legyen.

Ha [NAME1] helyére egy játékost, [NAME2] helyére pedig egy szobát írunk, akkor megtörténik a játékos mozgatása az adott szobába.

### Kimenetek:

A parancs eredménye kerül kiírásra.

Példa egy hallató egy szobába mozgatására:

```
>link Hallgato1 Szoba1
Hallgato1 moved to Szoba1
```

Példa egy hallató mozgatására egy szobába úgy, hogy a mozgás sikertelen:

```
>link Hallgato2 Szoba2
Hallgato2 move failed
```

## state [NAME] [STATE] [OPTION]

### Leírás:

A state parancccsal játékosok, szobák és tárgyak különböző állapotait és beállításait kezelhetjük.

### Opciók:

A [NAME] helyére a create parancccsal létrehozott és általunk elnevezett objektumok nevei kerülhetnek.

Ha kihagyjuk a parancs [STATE] részét, akkor kilistázhatjuk az összes lehetséges értéket, ami a [STATE] helyére kerülhet, a hozzájuk tartozó opciókkal.

A [STATE] megadásával az adott objektumot valamelyen állapotba vihetjük át, vagy az adott tulajdonságát állíthatjuk.

### Példák:

A "state szoba1 poisoned" parancs kiadásával a szoba1 nevű Room objektumunk gázos állapotú lesz.

A "state sajt1 timer 3" parancs kiadásával a sajt1 nevű camembert tárgyunk időzítője 3-ra állítódik.

### Kimenetek:

[STATE] megadása nélkül sortörésekkel kiírjuk az adott objektum lehetséges állapotait.

[STATE] megadásával kiírjuk az objektum új állapotát.

Példa egy hallató állapotának halottra állítására:

```
>state Hallgato1
student Hallgato1 states :
ALIVE
UNCONSCIOUS
DEAD
>state Hallgato1 DEAD
student Hallgato1 state set to DEAD
```

**control [OPTION1] [OPTION2]****Leírás:**

A kontroller parancs, a működésébe való beavatkozások kezelése érdekében. Idő megállítása és léptetése, véletlenszerűség kikapcsolása.

**Opciók:**

[OPTION1] irányítási opció kiválasztása, timer vagy random. Időzítési vagy véletlenszerűség manipulálása érdekében.

[OPTION2] timer esetén ki- bekapsolás, léptetés egy számmal.

pl.: timer on;

timer off;

timer 5

random esetén ki- bekapsolás,

pl.: random on;

random off;

**Kimenetek:**

[OPTION1] [OPTION2] ok

pl.: timer 5 ok

vagy rövid hibaüzenet

**interact [OPTION] [SUBJECT] [OBJECT1] [OBJECT2]****Leírás:**

A különböző interakciók vezérelhetők a parancssal, a karakterek és tárgyak között.

**Opciók:**

[OPTION] a különböző interakciók közül választhatunk: use, drop, pickup, illetve transistor esetén még activate, deactivate, pair, unpair. Emellett a szoba merge és split metódusai is ezzel hívhatóak meg.

[SUBJECT] ki/mi hajtja végre az adott interakciót.

[OBJECT1] melyik objektumon hajtjuk végre az adott interakciót.

[OBJECT2] két tranzisztor párosításakor itt adhatjuk meg a második tranzisztort.

pl.: interact pickup student5 transistor3

**Kimenetek:**

Siker esetén:

[CARACTER] [OPTION] [OBJECT1] ok

pl.: student1 pickup mask5 ok

student2 activate transistor3 ok

egyébként rövid hibaüzenet.

**stat****Leírás:**

Kiírja az összes létező objektum tulajdonságát

**Kimenetek:**

Az objektumok és azok tulajdonságai, sortöréssel köztük.

### 7.3 Összes részletes use-case

<b>Use-case neve</b>	Create listázás
<b>Rövid leírás</b>	Create parancs kilistázza a létrehozható objektumokat
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiadjuk a create parancsot, ezután a parancssorban megjelennek sortöréssel a létrehozható objektumok.

<b>Use-case neve</b>	Create
<b>Rövid leírás</b>	Egy választott objektum létrehozása az adott névvel.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kiadjuk a create parancsot egy objektum névvel, amit a fent említett listából választunk. Harmadik opciónak egy saját nevet adhatunk meg, aminek egy egybefüggő karaktersorozatnak kell lennie. EZ UTÁN VISSZAJELZÉST KAPUNK, HOGY SIKERESEN LÉTREJÖTT-E AZ OBJEKTUM A VÁLASZTOTT NÉVVEL. Helytelen objektumnév esetén hibát ír ki a program.

<b>Use-case neve</b>	Link szobaváltás
<b>Rövid leírás</b>	Link használata játékosok szobaváltására.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A link parancsot úgy adjuk ki, hogy utána írjuk egy általunk létrehozott játékos nevét majd egy általunk létrehozott szoba nevét. Ha valódi objektumok nevét adtuk meg, akkor a játékos megpróbál a játék szabályai szerint az adott szobába mozogni. A mozgás eredménye kiírásra kerül a parancssorra.  HA "f"-ET ÍRUNK A PARANCS UTÁN, AKkor ÚGY ADJUK HOZZÁ A SZOBÁHOZ A JÁTEKOST, HOGY NEM VESSZÜK FIGYELEMBE A JÁTEKSZABÁLYOKAT.

<b>Use-case neve</b>	State listázás
<b>Rövid leírás</b>	State parancs listázza az adott objektumra vonatkozó opciókat.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Úgy adjuk ki a state parancsot, hogy utána csak egy általunk létrehozott szoba, játékos vagy tárgy nevét adjuk meg. Ennek hatására sortöréssel listázásra kerül az összes opció, amely alkalmazható az adott objektumra..

<b>Use-case neve</b>	State játékosokra
<b>Rövid leírás</b>	Effektus alkalmazása egy választott játékosra.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az általunk választott állapotba helyezi az általunk választott játékest, vagy egy adott tulajdonságot tudunk megváltoztatni.

<b>Use-case neve</b>	State tárgyakra
<b>Rövid leírás</b>	Tárgyak állítása
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az általunk választott tárgyaknak a tulajdonságait tudjuk átállítani.

<b>Use-case neve</b>	Aktiválás, activate
<b>Rövid leírás</b>	A felhasználónál lévő transistor aktiválása
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Karakter interakció a meglévő transistorttal. A transistor aktivitás állapota megváltozhat. pl.: interact activate player1 transistor2.

<b>Use-case neve</b>	Párosítása, pair
<b>Rövid leírás</b>	A felhasználónál lévő 2 transistor párosítása
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Karakter interakció a meglévő transistorttal. A két transistor párt kapnak. pl.: interact pair player1 transistor2 transistor3

<b>Use-case neve</b>	Használat, use
<b>Rövid leírás</b>	A választott tárgy használata
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A játékos tárgya használatba lép. A futásra képes tárgyak állapota futóval változik, időzítő elindul. Hatását kifejti a környezetére, szobára.

<b>Use-case neve</b>	Felvétel, pickup
<b>Rövid leírás</b>	Szobából a felhasználó megkíséri a tárgy felvételét.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Átkerül a tárgy a játékoshoz.

<b>Use-case neve</b>	Eldobás, drop
<b>Rövid leírás</b>	Szobába tárgy lerakásának kísérlete.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Átkerül a tárgy a szobához, transistor teleportálhat. Időzítő felfüggesztheti futását.

<b>Use-case neve</b>	Kontroller időzítő állítása
<b>Rövid leírás</b>	Időzítő ki és bekapcsolása, idő léptetése
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Futó időzítő leáll, bekapcsol. Időzítő előreugorik megadott mennyiséget.

<b>Use-case neve</b>	Kontroller véletlenszerűség állítása
<b>Rövid leírás</b>	Véletlenszerűség ki és bekapcsolása.
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Kikapcsolt állapotban a véletlenszerű kiosztás

<b>Use-case neve</b>	Szoba Állapot lista
<b>Rövid leírás</b>	Kilistázza a szoba lehetséges állapotait
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A state parancsot úgy adjuk ki, hogy átadjuk neki a létrehozott szoba nevét

<b>Use-case neve</b>	Szoba állapot
<b>Rövid leírás</b>	Beállítja a szoba állapotát a megadottra
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A state parancsot úgy adjuk ki, hogy átadjuk neki a létrehozott szoba nevét, valamint a kívánt állapotát

<b>Use-case neve</b>	Szoba split
<b>Rövid leírás</b>	A szoba kettéválik
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az interact parancsot úgy adjuk ki, hogy megadjuk neki a split opciót valamint a szoba nevét.

<b>Use-case neve</b>	Szoba merge
<b>Rövid leírás</b>	Két szoba egybeolvad
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az interact parancsot úgy adjuk ki, hogy megadjuk neki a merge opciót valamint a szobák nevét, az elsőnek megadott szoba fog megmaradni.

<b>Use-case neve</b>	Stat
<b>Rövid leírás</b>	Összes objektum tulajdonságainak listázása
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A stat parancs kiadására a jelenlegi játék össze objektuma és az objektumok összes tulajdonsága is listázásra kerül.

<b>Use-case neve</b>	Save
<b>Rövid leírás</b>	Mentés
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	A jelenlegi állapot az adott txt fájlba mentésre kerül. A txt fájl a jelenlegi állapotig beírt összes parancsot fogja tartalmazni. A parancsok kimenete egy output.txt fájlba fog íródni.

<b>Use-case neve</b>	Load
<b>Rövid leírás</b>	Betöltés
<b>Aktorok</b>	Felhasználó
<b>Forgatókönyv</b>	Az adott txt fájlból beolvassunk egy elmentett játékállapotot.

## 7.4 Tesztelési terv

<b>Teszt-eset neve</b>	Transistor teleportál
<b>Rövid leírás</b>	A két nem szomszédos szobában, az egyikben egy párosított transistor lent van, a másikban pedig eldobásra kerül az aktivált párja. A játékos átkerül a elhelyezett transistoros szobába.
<b>Teszt célja</b>	Megmutassa hogy lehet teleportálni a transistort használva

<b>Teszt-eset neve</b>	Transistor (aktiválatlan) lehelyezés
<b>Rövid leírás</b>	A két nem szomszédos szobában, az egyikben egy párosított transistor lent van, a másikban pedig eldobásra kerül a nem aktivált párja.
<b>Teszt célja</b>	Aktiválás szükségességének vizsgálata

<b>Teszt-eset neve</b>	Transistor (nem párosított) lehelyezés
<b>Rövid leírás</b>	A két nem szomszédos szobában, az egyikben egy párosítatlan transistort helyez le a játékos, a másikban pedig eldobásra kerül egy másik transistor.
<b>Teszt célja</b>	Lehetőség van egy transistort eldobni, teleportáláshoz párosítani kell.

<b>Teszt-eset neve</b>	Transistor aktiválás
<b>Rövid leírás</b>	Két nála lévő transistort a játékos aktiválni próbál.
<b>Teszt célja</b>	Játékosnál lévő két transistora párosítás előtt nem aktiválható.

<b>Teszt-eset neve</b>	Transistor párosítása
<b>Rövid leírás</b>	Két nála lévő transistort a játékos párosítani próbál. A pár állapota változik.
<b>Teszt célja</b>	Játékosnál lévő két transistora párosításítható.

<b>Teszt-eset neve</b>	Sajt gázosít
<b>Rövid leírás</b>	A játékos egy szobában állva azt elgázosítja, ha használja a nála lévő sajtot.
<b>Teszt célja</b>	Működik a sajt használati hatása.

<b>Teszt-eset neve</b>	Légrissítő tisztít
<b>Rövid leírás</b>	A játékos egy gázos szobában állva, használja a nála lévő légrissítőt.

<b>Teszt-eset neve</b>	Légrissítő tisztít
<b>Teszt célja</b>	Működik a légrissítő használati hatása.

<b>Teszt-eset neve</b>	Sör megvéd
<b>Rövid leírás</b>	A játékos egy szobában állva, ha használja a nála lévő sört, és a sör időkeretén belül találkozik oktatóval.
<b>Teszt célja</b>	Működik a sör használati hatása.

<b>Teszt-eset neve</b>	Sör lejár.
<b>Rövid leírás</b>	A játékos egy szobában állva, használja a nála lévő sört, előtte és a sör időkeretén kívül is találkozik oktatóval.
<b>Teszt célja</b>	A sör hatása lejár és sör lejárta után nem véd meg.

<b>Teszt-eset neve</b>	Logarléc nyer
<b>Rövid leírás</b>	A játékos egy szobában állva, felveszi a ott lévő logarlécet.
<b>Teszt célja</b>	Megmutassa, hogy a logarléc felvétele a játék vége.

<b>Teszt-eset neve</b>	Logarléc hamis
<b>Rövid leírás</b>	A játékos egy szobában állva, felveszi a ott lévő hamis logarlécet.
<b>Teszt célja</b>	Hamis logarléc nem okoz játék végét.

<b>Teszt-eset neve</b>	Maszk véd
<b>Rövid leírás</b>	A játékos egyik szobában használja a nála lévő maszkot, a hatás időtartamán belül átlép egy elgázosított szobába.
<b>Teszt célja</b>	Maszk megvédi viselőjét a gázosított szoba hatásától.

<b>Teszt-eset neve</b>	Maszk lejár
<b>Rövid leírás</b>	A játékos egyik szobában használja a nála lévő maszkot, a hatás időtartamán belül átlép egy elgázosított szobába. A maszk hatása lejár, míg a szoba gázos marad.
<b>Teszt célja</b>	Maszk hatása le tud járni.

<b>Teszt-eset neve</b>	Maszk hamis
<b>Rövid leírás</b>	A játékos egyik szobában használja a nála lévő hamis maszkot, a hatás időtartamán belül átlép egy elgázosított szobába.
<b>Teszt célja</b>	Hamis maszk nem nyújt védelmet.

<b>Teszt-eset neve</b>	Szivacs működik
<b>Rövid leírás</b>	A játékos egy szobában állva, mikor van nála szivacs, oktató belép a szobába. Oktató lebénül. Szivacs elhasználódik.
<b>Teszt célja</b>	Szivacs hatása érvényesül, megvédi a hallgatót.

<b>Teszt-eset neve</b>	TVSZ megvéd
<b>Rövid leírás</b>	A játékos egy szobában állva, mikor van nála TVSZ aminek van még ereje, oktató belép a szobába.
<b>Teszt célja</b>	TVSZ megvédi a hallgatót az oktatótól

<b>Teszt-eset neve</b>	TVSZ lejár
<b>Rövid leírás</b>	A játékos egy szobában állva, mikor van nála TVSZ aminek van még egy ereje, oktató belép a szobába. Oktató kilép a szobából majd újra belép.
<b>Teszt célja</b>	TVSZ védelmi hatása csökken a tanárral való találkozáskor.

<b>Teszt-eset neve</b>	TVSZ hamis
<b>Rövid leírás</b>	A játékos egy szobában állva, mikor van nála hamis TVSZ, oktató belép a szobába.
<b>Teszt célja</b>	Hamis TVSZ nem védi meg a hallgatót az oktatótól.

<b>Teszt-eset neve</b>	Mozgásteszт
<b>Rövid leírás</b>	Az összes játékosból egyet különböző szobákból egy közös szobába mozgatunk. Sorban egy hallgatót, egy oktatót és egy takarítót fogunk a közös szobába mozgatni.
<b>Teszt célja</b>	Az összes játékos mozgásának működésének és egymásra való hatásának tesztelése.

<b>Teszt-eset neve</b>	Szoba split
<b>Rövid leírás</b>	Egy adott szobát splittelünk és a szabályoknak megfelelő szobákat fogunk kapni split után.
<b>Teszt célja</b>	Szoba splitelés működésének tesztelése.

<b>Teszt-eset neve</b>	Szoba merge
<b>Rövid leírás</b>	Egy adott szobát mergelünk egy másikkal, és a szabályoknak megfelelően jön létre egy új szoba.
<b>Teszt célja</b>	Merge függvény tesztelése.

<b>Teszt-eset neve</b>	Gázos szoba
<b>Rövid leírás</b>	Egy hallgatót egy gázos szobába helyezünk. A szoba hatására a hallgató elájul.
<b>Teszt célja</b>	Gázos szoba hatására a benne lévő hallgató eszméletét veszti.

<b>Teszt-eset neve</b>	Átkozott szoba
<b>Rövid leírás</b>	Csinálunk egy átkozott szobát amiben van egy hallgató. A hallgatót megpróbáljuk kiléptetni, míg egy oktatót megpróbálunk beléptetni. Egyikük se fog tudni sikeresen mozogni.
<b>Teszt célja</b>	Átkozott szoba nem engedi a ki vagy be mozgást.

<b>Teszt-eset neve</b>	Gázos kilépés
<b>Rövid leírás</b>	Gázos szobából kilépve a hallgatónak az aktiválódott maszkja deaktiválódik.
<b>Teszt célja</b>	Gázos szobából való kilépés deaktiválja a maszkot.

<b>Teszt-eset neve</b>	Felkelés
<b>Rövid leírás</b>	Egy hallgató gázos szobába kerül és egy időre eszméletét veszti. A gázt kitisztítjuk a szobából. Az idő lejárta után felkel a hallgató.
<b>Teszt célja</b>	Timer rendesen működik, lejártával a hallgató már nem eszméletvesztett.

<b>Teszt-eset neve</b>	Felvétel/Eldobás
<b>Rövid leírás</b>	Az összes tárgyból egyet adunk egy hallgatónak és egy oktatónak is. A tárgyak a működésüknek megfelelően aktiválódnak felvételkor, és bekerülnek az oktató/hallgató tárgyai közé. Felvétel után az összes tárgyat eldobjuk, melyek a játékosok szobájába fognak kerülni.
<b>Teszt célja</b>	Tárgyfelvétel, pickUp függvények és tárgyeldobás tesztelése.

## **7.5 Tesztelést támogató segéd- és fordítóprogramok specifikálása**

A tesztelés elvárt kimenetét összehasonlítjuk a teszt valós kimenetével. Az összehasonlításhoz a git program használható.

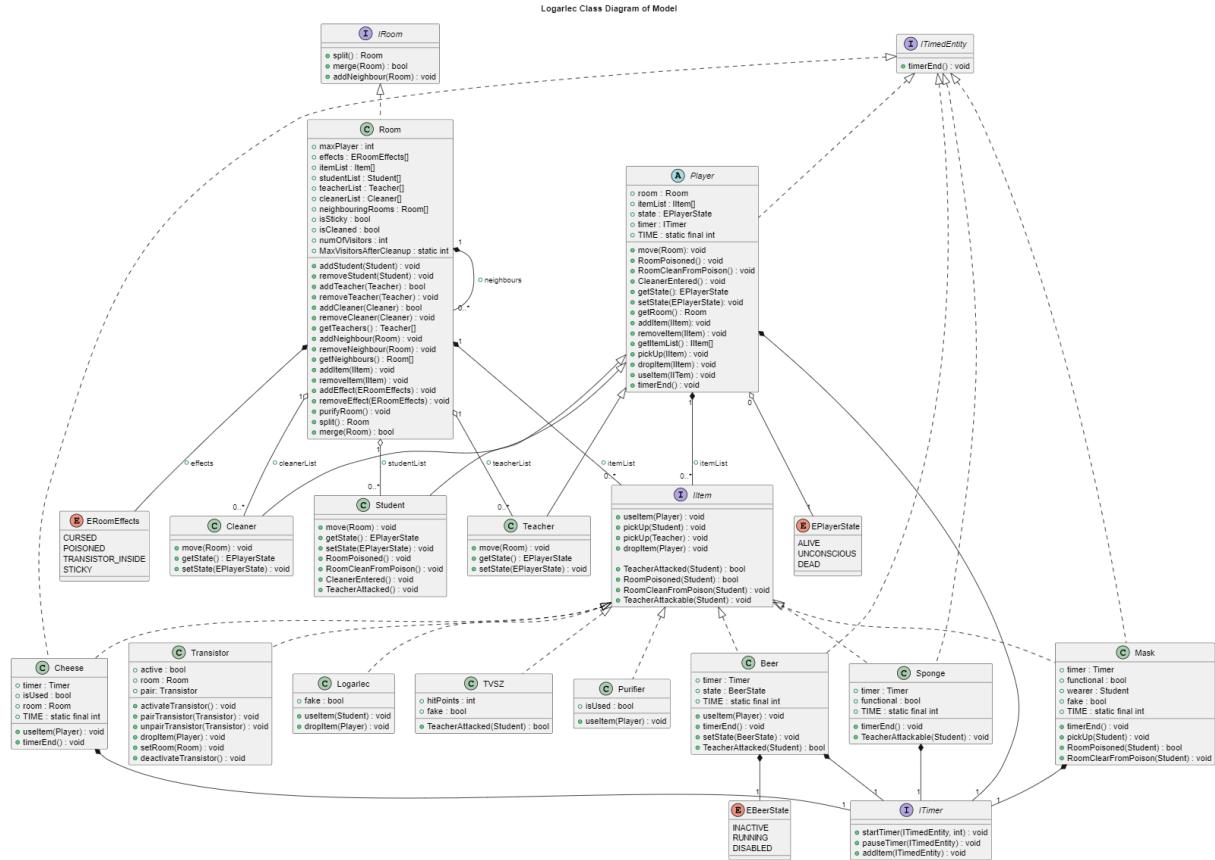
program használata: git diff --output=<file> --no-index <path> <path>  
A <file> helyére a kimeneti fájl neve kerül, a két útvonal pedig az elvárt és a kapott kimenet txtkre mutatnak.

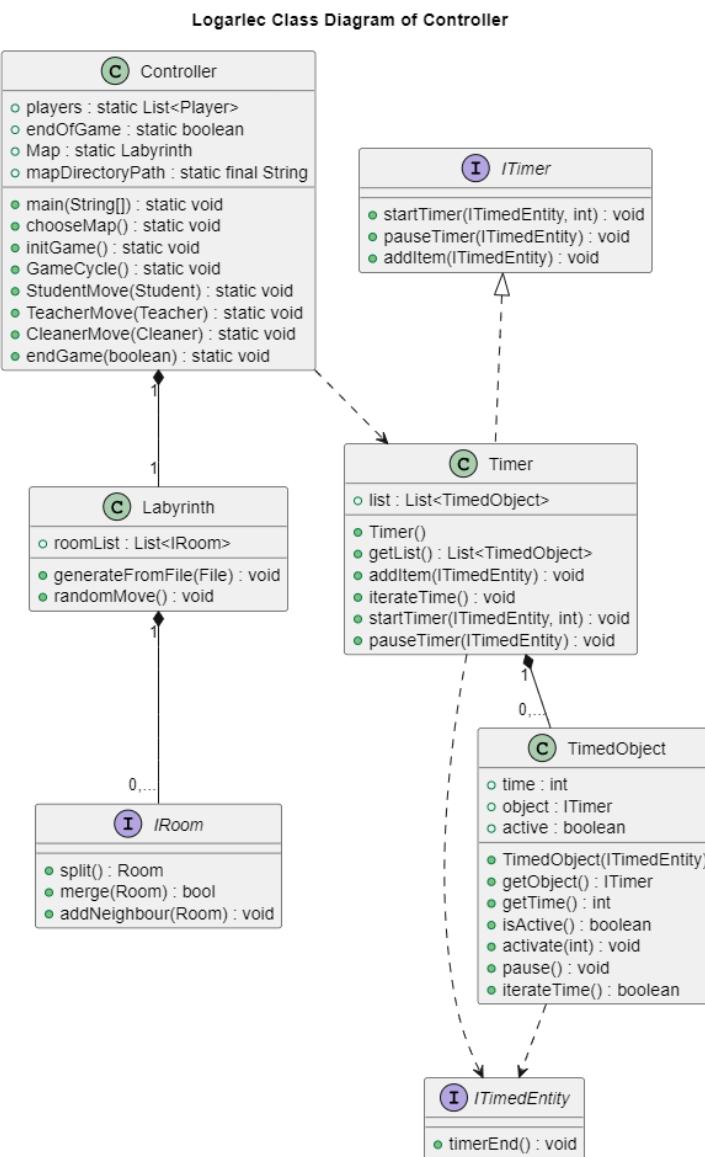
Ha a kimeneti fájl üres, nincs különbség a két file tartalma között, a teszt sikeresen futott le.

A parancsok batch fájlba összefoglalhatóak.

## **8. Részletes tervezés**

## **8.1 Osztályok és metódusok tervezése.**



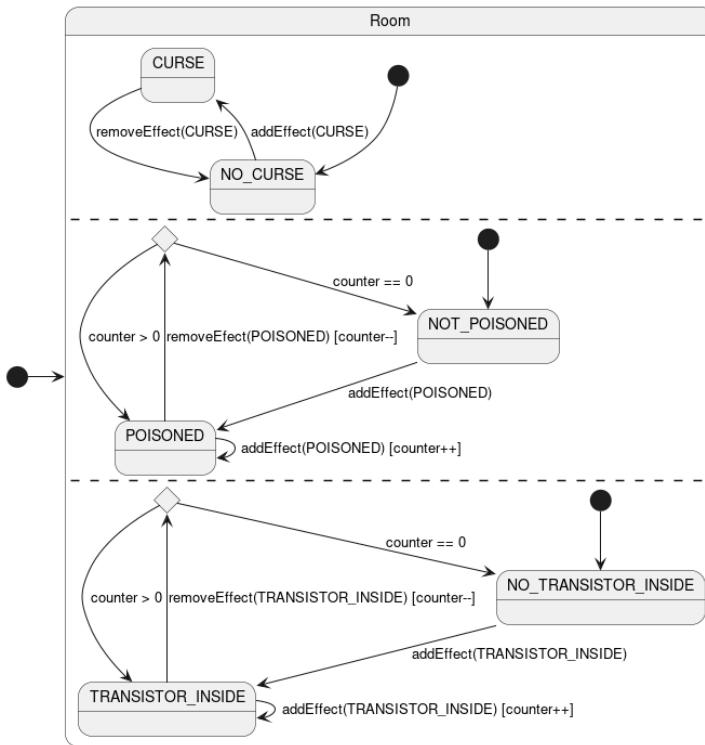


## Model

### 8.1.1 Room

- **Felelősség**
  - Az osztály felelőssége, hogy számon tartsa egy-egy szoba állapotát, valamint a bennük lévő játékosokat és tárgyakat.
- **Ősosztályok**
  - Nincs
- **Interfészek**
  - IRoom
- **Attribútumok**
  - - *maxPlayer*: int - Tárolja, hogy az adott szobában hány játékos lehet egyszerre maximum
  - - *effects* : *ERoomEffects*[] - Tárolja a szoba aktuális állapotait (pl. átkozott)
  - - *itemList*: *IItem*[] - Tárolja a szobában lévő tárgyakat
  - - *studentList* : *Student*[] - Tárolja a szobában lévő diákokat

- - *teacherList: Teacher[]* - Tárolja a szobában lévő tanárokat
- - *cleanerList: Cleaner[]* - Tárolja a szobában lévő takarítókat
- - *neighbouringRooms: Room[]* - Tárolja a szoba szomszédait
- + *isSticky : bool* - Tárolja, hogy a szoba ragacsos-e
- + *isCleaned : bool* - Tárolja, hogy a szoba ki lett-e tisztítva
- + *numOfVisitors : int* - Tárolja a szobában lévő látogatókat
- + *MaxVisitorsAfterCleanup : static int* - Tárolja, hogy a szobában legfeljebb hány látogató lehet a takarítás után



## • Metódusok

- + *addStudent(Student) : bool*
  - HA *student.getRoom() = null* akkor:  
Rakd bele a student-et a *studentList*-be  
Térj vissza igazzal  
KÜLÖNBEN HA a szoba átkozott VAGY a játékos szobája átkozott:  
Térj vissza hamissal  
KÜLÖNBEN HA a játékos szobája mérgezett ÉS a szoba nem mérgezett:  
Hívd meg a *CleanFromPoison* metódusát  
KÜLÖNBEN HA a szoba mérgezett:  
Hívd meg a *RoomPoisoned* metódusát  
Vedd ki a játékosat a szobából  
Rakd bele a játékosat a *studentList*-be  
Hívd meg a *checkAttacks* metódust  
Térj vissza igazzal
- + *removeStudent(Student) : bool*
  - Kiveszi a diádot a szobából. Azsal tér vissza, hogy benne volt-e.

- `+ getStudents() : Student[]`
  - Visszaadja a studentList értékét
- `+ addTeacher(Teacher) : bool`
  - HA `teacher.getRoom() = null` akkor:
    - Rakd bele a teachert a teacherList-be
    - Térj vissza igazzal
  - KÜLÖNBEN HA a szoba átkozott VAGY a játékos szobája átkozott:
    - Térj vissza hamissal
  - KÜLÖNBEN HA a játékos szobája mérgezett ÉS a szoba nem mérgezett:
    - Hívd meg a CleanFromPoison metódusát
  - KÜLÖNBEN HA a szoba mérgezett:
    - Hívd meg a RoomPoisoned metódusát
    - Vedd ki a játékost a szobájából
    - Rakd bele a játékost a teacherList-be
    - Hívd meg a checkAttacks metódust
    - Térj vissza igazzal
- `+ removeTeacher(Teacher) : bool`
  - Eltávolítja a játékost a teacherList-ből, azzal tér vissza, hogy benne volt-e
- `+ getTeachers() : Teacher[]`
  - Visszaadja a teacherList értékét
- `+ addCleaner(Cleaner) : bool`
  - HA `teacher.getRoom() = null` akkor:
    - Rakd bele a cleanert a cleanerList-be
    - Térj vissza igazzal
  - KÜLÖNBEN HA a szoba átkozott VAGY a játékos szobája átkozott:
    - Térj vissza hamissal
  - KÜLÖNBEN HA a játékos szobája mérgezett ÉS a szoba nem mérgezett:
    - Hívd meg a CleanFromPoison metódusát
  - KÜLÖNBEN HA a szoba mérgezett:
    - Hívd meg a RoomPoisoned metódusát
    - Vedd ki a játékost a szobájából
    - Rakd bele a játékost a cleanerList-be
    - Hívd meg a purifyRoom metódust
    - Térj vissza igazzal
- `+ removeCleaner(Cleaner) : bool`
  - Eltávolítja a játékost a cleanerList-ből, azzal tér vissza, hogy benne volt-e
- `+ getCleaners() : Cleaner[]`
  - Visszaadja a cleanerList értékét
- `+ addNeighbour(Room) : void`
  - Hozzáadja a szomszédot a neighbouringRooms-hoz
- `+ removeNeighbour(Room) : void`
  - Eltávolítja a szomszédot a neighbouringRooms-ból
- `+ getNeighbours() : Room[]`
  - Visszaadja a neighbouringRooms értékét
- `+ addItem(IItem) : void`
  - Hozzáadja a tárgyat a itemList-hoz

- + *removeItem(IItem)* : void
  - Eltávolítja a tárgyat a itemList-ból
- + *getItems()* : *IItem[]*
  - Visszaadja a itemList értékét
- + *addEffect(ERoomEffects)* : void
  - Hozzáadja az effect-et az effektekhez, majd ha ez a POISONED, akkor meghívja a játékosok RoomPoisoned metódusát
- + *removeEffect(ERoomEffects)* : void
  - Eltávolítja az effectet, majd ha nem maradt többé mérgezett, akkor meghívja a játékosok CleanFromPoison metódusát
- + *getEffects()* : *ERoomEffects[]*
  - Visszaadja a szoba aktuális effectjeit
- + *purifyRoom()* : void
  - Megtisztítja a szobát úgy hogy addig hívja a removeEffect metódusát, amíg van benne mérgezés
- + *split(): Room*
  - HA a szobában vannak játékosok VAGY az effektjei között van TRANSISTOR\_INSIDE:
    - Térj vissza nullal
    - Hozz létre új szobát newRoom néven
    - Add hozzá az új szobát a szomszédokhoz
    - Add hozzá a szobát az új szoba szomszédjaihoz
    - Térj vissza az új szobával
- + *merge(Room)* : bool
  - HA bármelyik szobában vannak játékosok VAGY az effektjei között van TRANSISTOR\_INSIDE:
    - Térj vissza hamissal
    - Add hozzá a szoba szomszédjait a sajátjaidhoz
    - Add hozzá a szoba tárgyait a sajátjaidhoz
    - Térj vissza igazzal

### 8.1.2 Player

- **Felelősség**
  - Az osztály felelőssége, hogy megvalósítsa azokat a metódusokat, melyek minden játékosra azonosan működnek
- **Ősosztályok**
  - Nincs
- **Interfészek**
  - ITimer
- **Attribútumok**
  - # room: Room - Tárolja a játékos aktuális szobáját
  - # itemList: *IItem[]* - Tárolja a játékos tárgyait
  - # state: *EPlayerState* - Tárolja a játékos állapotát
  - # timer : ITimer - Tárolja az időt mérő timert
  - # TIME: static final int - Tárolja, hogy az unconscious hatás meddig tart
- **Metódusok**
  - + *move(Room)* : void - Absztrakt metódus, minden leszármazott maga valósítja meg.
  - + *RoomPoisoned()* : void

■ CIKLUS i végig az itemList-en:

HA i.RoomPoisoned igaz:

Térj vissza

Állítsd be az állapotod mérgezettre

- + RoomCleanFromPoison(): void - Visszaállítja a játékos állapotát mérgezettről élőre
- + CleanerEntered(): void - Kiküldi az összes bent lévő játékost
- + getState() : EPlayerState - Visszaadja a játékos aktuális állapotát
- + setState(EPlayerState): void - Beállítja a játékos aktuális állapotát
- + getRoom(): Room - Visszatér a játékos aktuális szobájával
- + addItem(IItem) : void - Hozzáadja a tárgyat a itemList-hoz
- + removeItem(IItem) : void - Eltávolítja a tárgyat a itemList-ból
- + getItems() : IItem[] - Visszaadja a itemList értékét
- + pickUp(IItem) : void - Meghívja az addItem metódust, majd elveszi a szobától
- + dropItem(IItem) : void - Meghívja az addItem metódust, majd hozzáadja a szoba itemjeihez
- + useItem(IItem) : void - Meghívja az item use metódusát
- + timerEnd() : void - Visszaállítja a játékos állapotát ÉLŐre

### 8.1.3 Student

● Felelősség

- Az osztály felelőssége a személyek által irányított játékosok tárolása

● Ősztályok

- Player

● Interfészek

- Nincs

● Attribútumok

- Nincs

● Metódusok

- + move(Room) : void - Meghívja a szoba addStudent metódusát
- + TeacherAttacked() : void

■ CIKLUS i végig az itemList-en:

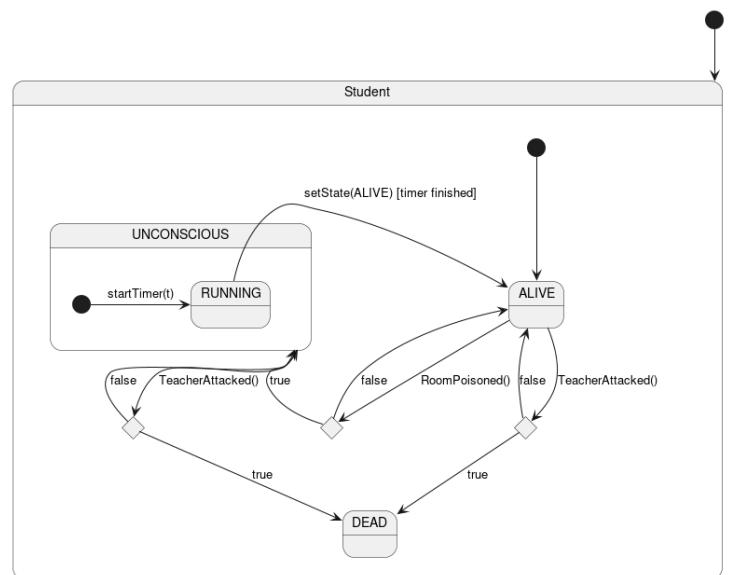
HA i.TeacherAttackable igaz:

Térj vissza

HA i.TeacherAttacked igaz:

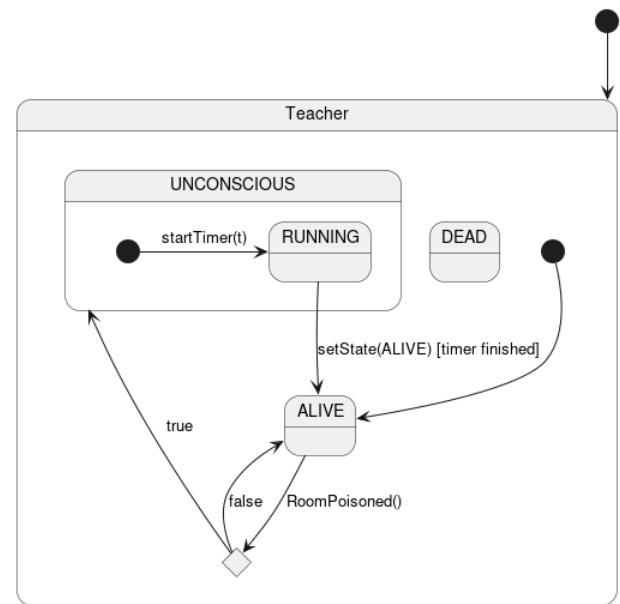
Térj vissza

Állítsd be az állapotod halottra



### 8.1.4 Teacher

- **Felelősség**
  - Az osztály felelőssége a nem játékos által irányított tanárok kezelése
- **Ősosztályok**
  - Player
- **Interfészek**
  - Nincs
- **Attribútumok**
  - Nincs
- **Metódusok**
  - `+ move(Room) : void` - Meghívja a szoba addTeacher metódusát



### 8.1.5 Cleaner

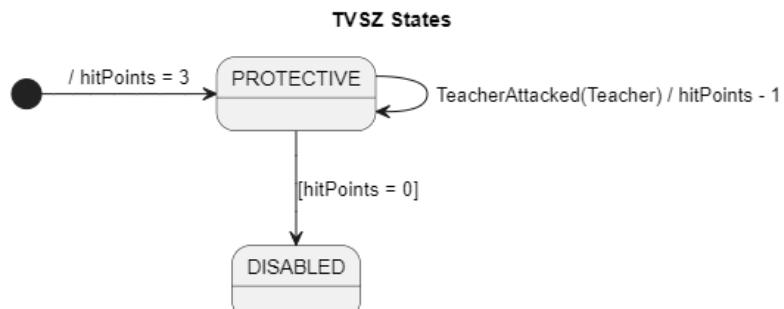
- **Felelősség**
  - Az osztály felelőssége a takarítók kezelése
- **Ősosztályok**
  - Player
- **Interfészek**
  - Nincs
- **Attribútumok**
  - Nincs
- **Metódusok**
  - `+ move(Room) : void` - Meghívja a szoba addCleaner metódusát

### 8.1.6 TVSZ

- **Felelősség**

A felelőssége, hogy eltárolja az éleponját és, hogy ha a hallgató egy esemény hatására meghívja akkor el tudja dönteni a válaszát tehát igaz vagy hamissal tér vissza. Emellett felelőssége, hogy belerakja/kiveszi magát a játékos (Player) tárgy listájába/listájából

- **Interfészek**
  - IIItem
- **Attribútumok**
  - `- hitpoints: int`
  - `- fake : bool`
- **Metódusok**
  - `+ TVSZ0:`  
Konstruktor. Beállítja a hitpoints attribútum értékét 3-ra és egy random számot generál 0 és 1 között. Ha az a szám kisebb, mint 0.2 akkor igazra ellenben hamisra állítja a fake attribútumot. Ezt a java.util.Random osztály segítségével valósítja meg

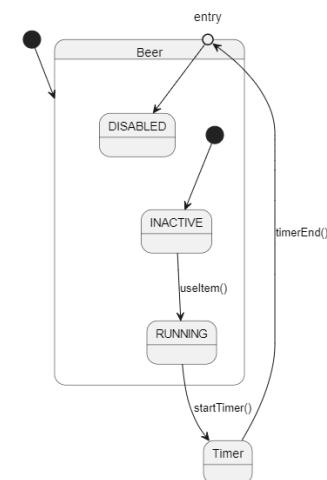


- **+ void pickUp(Student s):** Meghívja a kapott s hallgatóra az addItem függvényét és átadja magát.
- **+ void pickUp(Teacher t):** Meghívja a kapott t oktatóra az addItem függvényét és átadja magát.
- **+ dropItem(Player p):** Meghívja a kapott p player-re a removeItem függvényét és átadja magát.
- **+ bool TeacherAttacked(Student s):** Ha a hitpoint attribútuma nagyobb, mint 0 és a fake attribútuma hamis akkor levon a hitpointból egyet, majd igazzal tér vissza. Ellenben hamis a visszatérési értéke
- Egyéb IItem-ből implementált metódusok törzse üres, illetve ha az adott metódusnak van, visszatérési értéke false

### 8.1.7 Beer

- **Felelősség**

A felelőssége, hogy eltárolja az állapotát és az időzítőjét. Ha a hallgató egy esemény hatására meghívja akkor el tudja dönten a választ tehát igaz vagy hamissal tér vissza. Emellett felelőssége, hogy belerakja/kiveszi magát a játékos (Player) tárgy listájába/listájából illetve, hogy az időzítő lejártát fel tudja dolgozni és a tárgy használatakor el tudja indítani



- **Interfészek**

IItem

ITimedEntity

- **Attribútumok**

- - **timer:** ITimer
- - **state:** BeerState
- - **TIME:** static final int

- **Metódusok**

- **+ Beer(ITimer t):** Konstruktur. A kapott ITimer interface-ű objektumot beállítja a saját timerjének. Majd a t-re meghívja az addItem-et átadva magát.
- **+ void pickUp(Student s):** Meghívja a kapott s hallgatóra az addItem függvényét és átadja magát.
- **+ void pickUp(Teacher t):** Meghívja a kapott t oktatóra az addItem függvényét és átadja magát.
- **+ dropItem(Player p):** Meghívja a kapott p player-re a removeItem függvényét és átadja magát.
- **+ void useItem(Student):** Meghívja a timer-re a startTimer-t, átadja magát és a TIME paraméterét majd az állapotát (state) RUNNING-ra állítja
- **+ bool TeacherAttacked(Student s):** Ha INACTIVE az állapota akkor meghívja a timer-re a startTimer-t és átadja magát és a TIME paraméterét. Ha RUNNING az állapota akkor igazzal tér vissza. Egyébként hamissal tér vissza
- **+ void timerEnd():** Átállítja az állapotát DISABLED-re.
- Egyéb IItem-ből implementált metódusok törzse üres, illetve ha az adott metódusnak van, visszatérési értéke false

### 8.1.8 EBeerState (Enum)

- **Attribútumok**
  - *INACTIVE*
  - *RUNNING*
  - *DISABLED*

### 8.1.9 Transistor

- **Felelősség**

Eltárolja a jelenlegi szobát amennyiben le van helyezve ott, a páját amennyiben párosítva van illetve, hogy jelenleg aktív-e. Képes ezen attribútumokat beállítani egy-egy hívásra és teleportálni a játékost. Emellett felelőssége, hogy belerakja/kiveszi magát a játékos (Player) tárgy listájába/listájából

- **Interfészek**

IItem

- **Attribútumok**

- - **active**: bool
- - **room** : Room
- - **pair**: Transistor

- **Metódusok**

- + **Transistor()** : Konstruktor. active-t 0-ra, room-ot és pair-t null-ra állítja
- + **void pickUp(Student s)**: Ha a pair attribútuma nem null akkor a szoba (room) attribútumára meghívja a removeEffect-et TRANSISTOR\_INSIDE paraméterrel majd meghívja a kapott s hallgatóra az addItem függvényét és átadja magát.
- + **void pickUp(Teacher t)**: Meghívja a kapott t oktatóra az addItem függvényét és átadja magát.
- + **dropItem(Player p)**: Ha a pair attribútuma nem null és aktív akkor meghívja a teleport függvényt p paraméterrel, majd a deactivateTransistor-t p paraméterrel. Beállítja a room attribútumot a p szobájára.  
Végül meghívja a kapott p player-re a removeItem függvényét és átadja magát.
- + **void activateTransistor()**: Igazra állítja az active értékét
- + **void pairTransistor(Transistor t2)**: A pair értékét a kapott t2-re állítja, majd meghívja t2-re a setPair függvényt és átadja magát
- + **void unpairTransistor(Transistor t2)**: null-ra állítja a pair-t.
- + **void setRoom(Room r)**: Beállítja a room-ot r-re
- + **void deactivateTransistor()**: Hamisra állítja az active-ot
- - **void teleport(Player p)**: Meghívja p-re a move függvényt és átadja a pair attribútum alatt lévő tranzisztor szobát.
- Egyéb IItem-ból implementált metódusok törlése üres, illetve ha az adott metódusnak van, visszatérési értéke false

### 8.1.10 Mask

- **Felelősség**

A felelőssége, hogy eltárolja, hogy funkcionális-e, az időzítőjét és a hallgatót aki épp használja. El tudja dönteni, hogy megmenti-e a hallgatót egy gázos hatástól. Emellett felelőssége, hogy belerakja/kiveszi magát a játékos (Player) tárgy listájába/listájából

- **Interfészek**

IItem

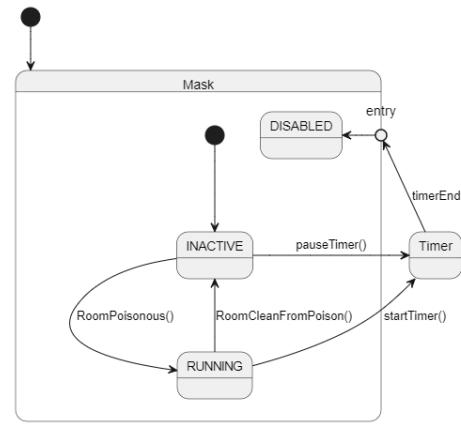
ITimedEntity

- **Attribútumok**

- - **timer**: ITimer
- - **TIME**: static final int
- - **wearer**: Student
- - **functional**: bool
- - **fake** : bool

- **Metódusok**

- + **Mask(Timer t)**: Konstruktor. Beállítja a timert a kapott t-re. A t-re meghívja az addItem-et, átadva magát és igazra váltja a functional attribútumát. Végül egy random számot generál 0 és 1 között. Ha az a szám kisebb, mint 0.2 akkor ellenben hamisra állítja a fake attribútumot. Ezt a java.util.Random osztály segítségével valósítja meg
- + **setWearer(Student s)**: A wearer attribútumot beállítja s-re.
- + **void pickUp(Student s)**: A wearer attribútumot beállítja s-re, majd meghívja a kapott s hallgatóra az addItem függvényét és átadja magát.
- + **void pickUp(Teacher t)**: Meghívja a kapott t oktatóra az addItem függvényét és átadja magát.
- + **dropItem(Player p)**: Meghívja a timer attribútumára a pauseTimer és átadja magát Meghívja a kapott p player-re a removeItem függvényét és átadja magát.
- + **bool RoomPoisoned(Student s)**: Ha a functional false vagy a fake true akkor false-al tér vissza. Ellenben meghívja a timer-re a startTimer átadva magát és a TIME attribútumot majd igazzal tér vissza
- + **void RoomClearFromPoison(Student s)**: Meghívja a timer-re a pauseTimer-t és átadja magát
- + **void timerEnd**: Hamisra állítja a functional attribútumot
- Egyéb IItem-ből implementált metódusok törlése üres, illetve ha az adott metódusnak van, visszatérési értéke false



### 8.1.11 Sponge

- **Felelősség**

A felelőssége, hogy eltárolja, hogy funkcionális-e és az időzítőjét. El tudja dönteni, hogy megmenti-e a hallgatót egy támadástól illetve ha igen akkor képes a hallgató szobájában lévő oktatók állapotát UNCONSCIOUS-ra rakni. Emellett felelőssége, hogy belerakja/kiveszi magát a játékos (Player) tárgy listájába/listájából

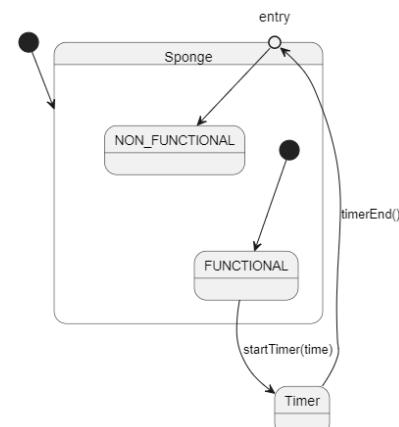
- **Interfészek**

IItem

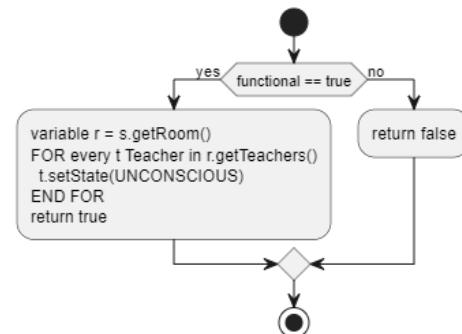
ITimedEntity

- **Attribútumok**

- - **timer**: ITimer



- - **TIME**: static final int
- - **functional**: bool
- **Metódusok**
  - + **Sponge(Timer t)**: Konstruktor. Beállítja a timert a kapott t-re. és meghívja rá az addItem-et.
  - + **void pickUp(Student s)**: Meghívja a timer-re a startTimer-t átradva magát és a TIME attribútumát. A functional-t igazra állítja, végül meghívja a kapott s hallgatóra az addItem függvényét és átadja magát.
  - + **void pickUp(Teacher t)**: Meghívja a kapott t oktatóra az addItem függvényét és átadja magát.
  - + **void dropItem(Player p)**: Meghívja a timer attribútumára a pauseTimert és átadja magát. Meghívja a kapott p player-re a removeItem függvényét és átadja magát.
  - + **boolean TeacherAttackable(Student s)**:



A működését az aktivitás diagram szemlélteti:

- + **void timerEnd()**: Hamisra állítja a functional attribútumot
- Egyéb IItem-ból implementált metódusok törzse üres, illetve ha az adott metódusnak van, visszatérési értéke false

### 8.1.12 Cheese

- **Felelősség**

A felelőssége, hogy eltárolja, hogy elhasznált-e és az időzítőjét. Képes a hallgató szobájára egy POISONOUS effectet helyezni, illetve levenni azt az idő lejártakor. Emellett felelőssége, hogy belerakja/kiveszi magát a játékos (Player) tárgy listájába/listájából

- **Interfészek**

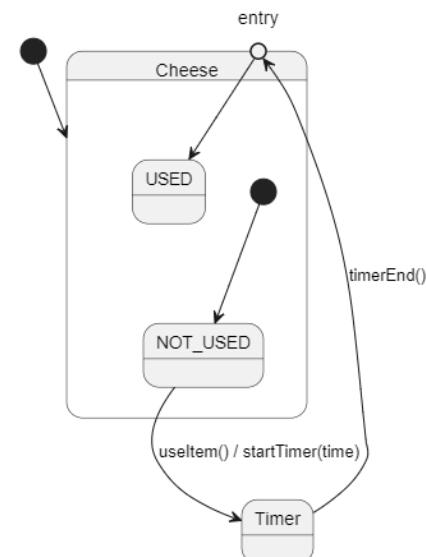
IItem

ITimedEntity

- **Attribútumok**

- - **timer**: ITimer
- - **TIME**: static final int
- - **isUsed**: bool

- **Metódusok**



- + **Cheese(Timer t)**: Konstruktor. Beállítja a timert a kapott t-re. és meghívja rá az addItem-et.
- + **void pickUp(Student s)**: Meghívja a kapott s hallgatóra az addItem függvényét és átadja magát.
- + **void pickUp(Teacher t)**: Meghívja a kapott t oktatóra az addItem függvényét és átadja magát.
- + **void dropItem(Player p)**: Meghívja a kapott p player-re a removeItem függvényét és átadja magát.
- + **void useItem(Player p)**: Ha az isUsed true akkor visszatér. Ellenben meghívja a timer-re a startTimer-t átradva magát és a TIME attribútumát, majd a room paraméterére beállítja a p szobáját, erre a szobára meghívja az addEffects(POISONED) függvényt, majd a p-re a removeItem-et átradva magát, illetve a room-ra az addItem-et ahol szintén átadja magát
- + **timerEnd()**: Meghívja a room attribútumára a removeEffect(POISONED)-et.
- Egyéb IItem-ből implementált metódusok törlése üres, illetve ha az adott metódusnak van, visszatérési értéke false

### 8.1.13 Purifier

- **Felelősség**

Eltárolja magáról, hogy használható-e illetve ha használják akkor képes egy szobát megtisztítani. Emellett felelőssége, hogy belerakja/kiveszi magát a játékos (Player) tárgy listájába/listájából

- **Interfészek**

IItem

- **Attribútumok**

- - **isUsed**: bool

- **Metódusok**

- + **Purifier()**: Konstruktor. isUsed-ot false-ra állítja
- + **void pickUp(Student s)**: Meghívja a kapott s hallgatóra az addItem függvényét és átadja magát.
- + **void pickUp(Teacher t)**: Meghívja a kapott t oktatóra az addItem függvényét és átadja magát.
- + **void dropItem(Player p)**: Meghívja a kapott p player-re a removeItem függvényét és átadja magát.
- + **void useItem(Student s)** - A kapott s hallgató szobájára meghívja a purifyRoom() függvényt, majd az isUsed-ot true-ra állítja
- Egyéb IItem-ből implementált metódusok törlése üres, illetve ha az adott metódusnak van, visszatérési értéke false

## 8.1.14 Logarlec

- **Felelősség**

Eltárolja a fake attribútumát. Felelős azért, hogy ha egy hallgató veszi fel akkor eldöntse, hogy vége-e a játéknak illetve erről értesítse a kontrollert. Emellett felelőssége, hogy belerakja/kiveszi magát a játékos (Player) tárgy listájába/listából

- **Interfészek**

IItem

- **Attribútumok**

- **fake : bool**

- **Metódusok**

- **+ Logarlec():** Konstruktor. Egy random számot generál 0 és 1 között. Ha az a szám kisebb, mint 0.2 akkor igazra ellenben hamisra állítja a fake attribútumot. Ezt a java.util.Random osztály segítségével valósítja meg.
- **+ void pickUp(Student s):** Meghívja a kapott s hallgatóra az addItem függvényét és átadja magát. Ha a fake hamis akkor meghívja a Controller osztály endGame(true) függvényét.
- **+ void pickUp(Teacher t):** Meghívja a kapott t oktatóra az addItem függvényét és átadja magát.
- **+ void dropItem(Player p):** Meghívja a kapott p player-re a removeItem függvényét és átadja magát.

## Controller

---

### 8.1.15 Labyrinth

- **Felelősség**

Nyilvántartja az összes szobát, képes egy random szobára random split-et vagy merge-t hívni. Illetve képes egy fájlból beolvasni egy előre megírt labirintust, ez alapján feltölteni a listáját és minden szobára a szomszédjait

- **Attribútumok**

- **roomList:** IRoom

- **Metódusok**

- **+ void generateFromFile(File file):** A szobákat létrehozza és belerakja a roomList attribútumba, emellett feltölti az egyes szobák szomszédsági listáját. A következő pszeudokód szemlélteti a működését:

```

"file" alatt lévő file megnyitása
CIKLUS minden 'l' sor olvasása a file végéig
    'l' sor parse-olása
        HA 'l' sor üres akkor ugord át
        HA 'l' sor "create room"-al kezdődik akkor
            - "rx" név kigyűjtése
            - HA nem található akkor "file beolvasása során hiba
                történt: *Szoba neve nem található* kivétel dobása
            - "rx" szoba létrehozása és roomListhez hozzáadás
        HA 'l' sor "link"-el kezdődik akkor
            - "rx" és "ry" név kigyűjtése
            - HA nem található akkor "file beolvasása során hiba
                történt: *Szobák neve nem található* kivétel dobása
            - HA létezik "rx" és "ry" szoba a roomListben akkor "rx"
                szomszédsági listájához add hozzá "ry"-t
            - KÜLÖNBEN "file beolvasása során hiba történt: *Szoba nem
                létezik* kivétel dobása
        KÜLÖNBEN
            file beolvasása során hiba történt: *parancs nem
            értelmezett* hiba dobása
        VÉGE
    CIKLUS VÉGE

```

- **+ void randomMove():** Egy véletlenszerű szobára meghívja vagy a split-et vagy a merge-t. Arra, hogy ez a split lesz 50% esély van. A működését a következő pszeudokód mutatja be:  
HA roomList == null vagy üres AKKOR kivétel dobása: \*Üres lista\*

EGYÉBKÉNT

```

Random szám generáló inicializálása
'i' egész szám generálása {0 < i < roomList.size - 1}
i-edik elem megkeresése a listában
'x' lebegőpontos szám generálása {0 < x < 1}
HA
    x <= 0.5 akkor roomList[i].split() hívása
EGYÉBKÉNT
    'i2' egész szám generálása {0 < i2 < roomList.size - 1, i2 != i}
    i-edik elem megkeresése a listában, majd roomList[i].merge(i2) hívás
VÉGE

```

### 8.1.16 TimedObject

- **Felelősség**

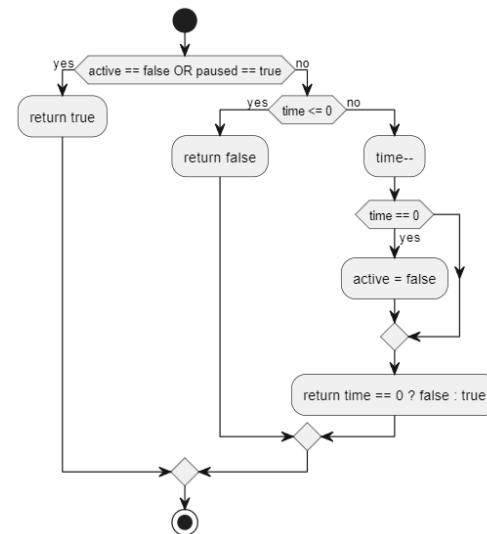
Ez a Timer segédosztálya. minden ITimedEntity egy TimedObject-é konvertálódik a Timerben, és ezeken van végrehajtva az iterálás. minden TimedObject eltárolja magáról a jelenlegi idejét és állapotát, hogy melyik ITimedEntity-hez tartozik illetve, hogy meg van-e állítva.

- **Attribútumok**

- - time : int
- - object : ITimedEntity
- - active : boolean
- - paused : boolean

- **Metódusok**

- + **TimedObject(ITimedEntity o)**: Konstruktor. A time-ot '-1'-re, objectet 'o'-ra, active-ot és paused-et pedig 'false'-ra állítja
- + **ITimer getObject()** : Visszatér az object ITimer-rel
- + **int getTime()**: Visszatér a time integerrel
- + **boolean isActive()**: Visszatér az active boolean-al
- + **boolean isPaused()**: Visszatér az paused boolean-al
- + **void activate(int t)**: time-ot t-re, active-ot igazra állítja
- + **void pause()**: Paused-et igazra állítja
- + **void resume()**: Paused-et hamisra állítja
- + **boolean iterate()**: Hamissal tér vissza ha lejárt, igazzal ha nem. A működését az



alábbi aktivitás diagram szemlélteti:

### 8.1.17 Timer

- **Felelősség**

Eltárolja a TimedObjecteket. Ezzel interaktálva lehet az egyes ITimedEntity-k időzítőjét iterálni. Ez az osztály felelős az elindításukra és megállításukra is, illetve ő küldi az üzenetet ha lejár az egyik ITimedEntity időzítője.

- **Interfészek**

ITimer

- **Attribútumok**

- - list : List<TimedObject>

- **Metódusok**

- + Timer(): Konstruktor. Inicializálja a list-et.
- + List<TimedObject> getList(): Visszatér a list referenciájával
- + void addItem(ITimedEntity o): A listához hozzáad egy new TimedObject(o)-t
- + void iterateTime(): Végigmegy az egész list-en, majd minden 't' lista elemre megvizsgálja, hogy ha 't' aktív (isActive) és az iterateTime hamissal tér vissza akkor t object-ére meghívja a timerEnd()-et.
- + void startTimer(ITimedEntity o, int time): Végigmegy az egész list-en és megkeresi azt a 't' TimedObject-et aminek object attribútuma megegyezik 'o'-val, majd ha ez paused akkor meghívja 't'-re a resume()-ot. Ellenben ha nem aktív (tehát t.isActive == false) akkor meghívja rá az activate()-et és átadja a time-ot paraméterként
- + void pauseTimer(ITimedEntity): Végigmegy az egész list-en és megkeresi azt a 't' TimedObject-et aminek object attribútuma megegyezik 'o'-val, majd meghívja rá a pause() függvényt

### 8.1.18 Controller

- **Felelősség**

Ez a fő osztály, innen indul a program. Eltárolja a játékban lévő játékosokat, a labirintust, a játék állapotát nyilvántartja illetve tudja melyik mappában vannak a készített pályák. Felelőssége, hogy interaktáljon a felhasználókkal és inicializálja a játékot a standard inputról kapott adatok alapján. Ezen felül, hogy megvalósítsa a hallgatók konzolos ablakon történő akciók kiírását és ezek lekezelését. Mozgatja a teacher és cleaner játékosokat. Illetve a játék ciklusa is ebben található. Végül a játék vége üzenetet is ő írja ki

- **Attribútumok**

- - players : static List<Player>
- - Map : static Labyrinth

- - **endOfGame : static boolean**
- - **mapDirectoryPath : static final boolean** - Kezdőértéke a mappa címe ahol találhatók a létrehozott map-ok

- **Metódusok**

- + **static void main()**: A program indulási pontja. Meghívja az initGame-et, majd a GameCycle-t.

- + **static void endGame(boolean victory)**: Beállítja az endOfGame boolean-t true-ra, majd ha victory == true akkor kiírja a képernyőre a VICTORY, ellenben pedig a DEFEAT szöveget

- - **static void chooseMap()**: Kílistázza a mapDirectoryPath alatt lévő fájlokat és a felhasználó tud választani közülük egyet. A működését a következő pszeudokód mutatja be:

```

"directory" nevű File objektum létrehozása a mapDirectoryPath címmel
"files" nevű File típusú tömb inicializálása
"files" feltöltése "directory" alatt lévő fájlokkal
HA "files" == null AKKOR kivétel dobása: *Nem létezik pálya. Kérlek
    készíts egyet magadnak és törlsd fel a "directory" mappába*
KÜLÖNBEN
    "file" tartalmának indexelt kílistázása
    'i' szám kérése a felhasználótól
    HA 0 <= 'i' < files.size AKKOR a Map generateFromFile nevű
        függvényének meghívása az 'i' index alatt lévő File objektum
        paraméterrel
    KÜLÖNBEN
        "Nem létezik ilyen file" kiírás és szám bekérésének megismétlése
    VÉGE
VÉGE
  
```

- - **static void initGame()**: Kiírja a főmenüt a standard input-ra. Itt tudjuk megadni, hogy hányan játszanak a játékkal (student darabszáma) illetve, hogy hány teacher-t és cleanert szeretnénk a játékba. Majd meghívja a chooseMap()-ot ahol a mapot is kiválaszthatjuk amennyiben létrehoztuk azt előre egy külön fájlban. A működését a következő pszeudokód mutatja be:

```

Főmenü megjelenítése
CIKLUS
    's' szám bekérése a felhasználótól
    HA 0 < 's' < 6 AKKOR
        's' db Student létrehozása és a players listához adása
        BREAK
    CIKLUS VÉGE
    CIKLUS
        't' szám bekérése a felhasználótól
        HA 0 < 't' < 6 AKKOR
            't' db Teacher létrehozása és a players listához adása
            BREAK
    CIKLUS VÉGE
    CIKLUS
        'c' szám bekérése a felhasználótól
        HA 0 < 'c' < 6 AKKOR
            'c' db Cleaner létrehozása és a players listához adása
  
```

```

    BREAK
    CIKLUS VÉGE
    players lista megkeverése
    chooseMap() hívása

```

- **- static void GameCycle():** Ebben a metódusban van a teljes játék ciklus. Ez hívja meg a StudentMove, TeacherMove, CleanerMove és Map.randomMove függvényeket. A működését a következő pszeudokód szemlélteti:

```

    CIKLUS AMÍG endOfGame == false
        CIKLUS minden 'p' Player objektumra players listában
            HA 'p' Student AKKOR StudentMove(p) hívása
            HA 'p' Teacher AKKOR TeacherMove(p) hívása
            HA 'p' Cleaner AKKOR CleanerMove(p) hívása
            Map randomMove() függvényének hívása
        CIKLUS VÉGE
    CIKLUS VÉGE

```

- **- static void StudentMove(Student s):** A felhasználóval való interakció itt valósul meg. Itt tudja mozgatni a hallgatót. A működését az alábbi pszeudokód szemlélteti:

```

    Aktuális szoba kiírása
    Szomszédos szobák listázása
    A szobában lévő tárgyak listázása
    moved: bool változó létrehozása
    moved = false
    CIKLUS
        'l' Sor beolvasása a standard bemenetről
        'l' sor parse-olása
            HA l[0] == "move"
                HA moved == false AKKOR
                    r : Room változó létrehozása
                    r = null
                    CIKLUS minden ri szobára s.getRoom.getNeighbours()-ben
                        HA l[1] == ri
                        r = ri
                        BREAK
                    CIKLUS VÉGE
                    HA r != null AKKOR s.move(r)
                    KÜLÖNBEN "Nem létezik ilyen szoba" kiírása
            HA l[0] == "interact"
                validItem: bool változó létrehozása
                it: IItem változó létrehozása
                it = null
                CIKLUS minden 'i' itemre s.getRoom.getItemList()-ben
                    HA l[2] == 'i'
                        it = i
                        BREAK
                CIKLUS VÉGE
                HA it != null
                    HA l[1] == "pickUp" AKKOR it.pickUp(s)
                    HA l[1] == "drop" AKKOR it.dropItem(s)
                    HA l[1] == "use" AKKOR it.useItem(s)
                    HA l[1] == "activate" ÉS it tranzisztor AKKOR
                        it.ActivateTransistor()
                    HA l[1] == "deactivate" ÉS it tranzisztor AKKOR
                        it.deactivateTransistor()
                    HA l[1] == "pair" ÉS it tranzisztor AKKOR
                        it.PairTransistor(l[3])

```

```

    HA l[1] == "unpair" ÉS it tranzisztor AKKOR
        it.UnpairTransistor(l[3])
    KÜLÖNBEN "Nem létezik ilyen item" kiírása
    HA l[0] == "end" AKKOR BREAK
    KÜLÖNBEN "Rossz input" üzenet küldése
    CIKLUS VÉGE

```

- - static void TeacherMove(Teacher t):

Az oktató a számára elérhető ajtók közül véletlenszerűen választ egyet amikor sorra kerül, meghívódik a játék összes tanárára.

Pszeudokód:

```

method TeacherMove is
    input:T: teacher, Tr: room of teacher, R: list of connecting rooms of Tr
    NUM is a random number from 0 to number of rooms in R
    if NUM = 0
        return
    else
        move T to the NUMth room in R Call corresponding method of T
        with argument: NUMth room in R

```

- - static void CleanerMove(Cleaner c):

Az oktatóhoz hasonlóan a takarító is véletlenszerűen választ a lehetséges mozgások közül.

Pszeudokód:

```

method CleanerMove is
    input:C: cleaner, Cr: room of cleaner, R: list of connecting rooms of Cr
    NUM is a random number from 0 to number of rooms in R
    if NUM == 0
        return
    else
        move C to the NUMth room in R. Call corresponding method of C
        with argument: NUMth room in R

```

## 8.2 A tesztek részletes tervei, leírásuk a teszt nyelvén

### 8.2.1 \* teszt

#### Leírás

A konzultáción előírt kötelező eset.

#### Ellenőrzött funkcionálitás, várható hibahelyek

Szoba létrehozása.	Játékos létrehozása	Transistor létrehozása.
Logarléc létrehozása	Játékos nyer	Maszk létrehozása
TVSZ használata	Játékos szobába lépése.	Tanár lép
Tanár felvesz tárgyat	Tanár létrehozása	

- **Bemenet**

create room r1 create teacher t link r1 te create room r2 link r1 r2 create mask m link m r2 create room r3 link r3 r2	create logar l link l r3 create room r4 link r3 r4 create room r5 link r4 r5 link r2 r5 create room r6 link r5 r6	link r4 r6 link r2 r6 create room r7 link r2 r7 create student st link r7 st create tvsz tvsz1 state tvsz1 hitPoint 1 link tvsz1 st
--	---	---

create room r8  
link r7 r8  
link te r2

### • Elvárt kimenet

room r1 created  
teacher t created  
t moved to r1  
room r2 created  
r1 connected to r2  
mask m created  
m added to r2  
room r3 created  
r3 connected to r2  
logar l created  
l added to r3  
room r4 created  
r3 connected to r4  
room r5 created  
r4 connected to r5  
r2 connected to r5  
room r6 created  
r5 connected to r6  
r4 connected to r6  
r2 connected to r6  
room r7 created  
student st created  
st added to r7  
tvzs tvsz1 created  
st hitPoint set to 1  
room r8 created  
r7 connected to r8  
te moved to r2  
interact pickup te m ok  
st moved to r2  
st moved to r3  
interact st pickup l ok  
stat:  
st room r3  
st EPlayerState ALIVE  
st itemList tvsz m l  
m functional true

interact pickup te m  
link st r2  
link st r3

m fake false  
l fake false  
tvzs1 hitPoint 0  
tvzs1 fake false  
r1 effects  
r1 itemList  
r1 studentList  
r1 teacherList  
r1 cleanerList  
r1 neighbouringRooms r2  
r1 sticky false  
r1 cleaned false  
r1 numofVisitors 1  
r2 effects  
r2 itemList  
r2 studentList  
r2 teacherList te  
r2 cleanerList  
r2 neighbouringRooms r1 r3 r5 r6 r7  
r2 sticky false  
r2 cleaned false  
r2 numofVisitors 2  
r3 effects  
r3 itemList  
r3 studentList st  
r3 teacherList  
r3 cleanerList  
r3 neighbouringRooms r2 r4  
r3 sticky false  
r3 cleaned false  
r3 numofVisitors 1  
r4 effects  
r4 itemList  
r4 studentList  
r4 teacherList  
r4 cleanerList  
r4 neighbouringRooms r3 r5 r6  
r4 sticky false

interact st pickup l  
stat

r4 cleaned false  
r4 numofVisitors 0  
r5 effects  
r5 itemList  
r5 studentList  
r5 teacherList  
r5 cleanerList  
r5 neighbouringRooms r2 r4 r6  
r5 sticky false  
r5 cleaned false  
r5 numofVisitors 0  
r6 effects  
r6 itemList  
r6 studentList  
r6 teacherList  
r6 cleanerList  
r6 neighbouringRooms r2 r4 r5  
r6 sticky false  
r6 cleaned false  
r6 numofVisitors 0  
r7 effects  
r7 itemList  
r7 studentList  
r7 teacherList  
r7 cleanerList  
r7 neighbouringRooms r2 r6 r8  
r7 sticky false  
r7 cleaned false  
r7 numofVisitors 1  
r8 effects  
r8 itemList  
r8 studentList  
r8 teacherList  
r8 cleanerList  
r8 neighbouringRooms r7  
r8 sticky false  
r8 cleaned false  
r8 numofVisitors 0

## 8.2.2 Transistor teleportál

### • Leírás

A két nem szomszédos szobában, az egyikben egy párosított transistor lerakása, a másikban pedig eldobásra kerül az aktivált párja. A játékos átteleportál az első elhelyezett transistoros szobába.

### • Ellenőrzött funkcionalitás, várható hibahelyek

Szoba létrehozása.

Játékos létrehozása

Transistor létrehozása.

Transistor szobához állítása.

Transistor eldobása.

Transistor aktiválása.

Teleportálás

Játékos szobába lépése.

Transistor párosítása

### • Bemenet

create room r1  
create room r2  
create student s  
create transistor t1  
create transistor t2

link r1 s  
link s t1  
link s t2  
interact pair s t1 t2  
interact activate s t1

interact drop s t1  
link -f r2 s  
interact activate s t2  
interact drop s t2  
stat

### • Elvárt kimenet

room r1 created room r2 created student s created transistor t1 created transistor t2 created s moved to r1 t1 added to s t2 added to s interact pair s t1 t2 ok interact activate s t1 ok interact drop s t1 ok s moved to r2 interact activate s t2 ok interact drop s t2 ok	Stat: s room r1 s EPlayerState ALIVE s itemList t1 t1 pair t2 t1 active true t1 room r1 t2 pair t1 t2 active false t2 room r2 r1 effects TRANSISTOR_IN r1 itemList t1 r1 studentList s r1 teacherList	r1 cleanerList r1 neighbouringRooms r1 sticky false r1 cleaned false r1 numOfVisitors 1 r2 effects TRANSISTOR_IN r2 itemList t1 r2 studentList r2 teacherList r2 cleanerList r2 neighbouringRooms r2 sticky false r2 cleaned false r2 numOfVisitors 1
---	--	--

### 8.2.3 Transistor (aktiválatlan) lehelyezés

- Leírás

A két nem szomszédos szobában, az egyikben egy párosított transistor lent van, a másikban pedig eldobásra kerül a nem aktivált párja. Az aktiválatlan transistor nem vált ki teleportálási hatást, a tárgy lekerül a szobába.

- Ellenőrzött funkcionalitás, várható hibahelyek

Szoba létrehozása.	Játékos létrehozása	Transistor létrehozása.
Transistor szobához állítása.	Transistor eldobása.	Játékos szobába lépése.
Transistor párosítása		

- Bemenet

create room r1 create room r2 create student s create transistor t1	create transistor t2 link r1 s link s t1 link s t2 interact pair s t1 t2	interact drop s t1 link -f r2 s interact drop s t2 stat
--	--	--

- Elvárt kimenet

room r1 created room r2 created student s created transistor t1 created transistor t2 created s moved to r1 t1 added to s t2 added to s interact pair s t1 t2 ok interact drop s t1 ok s moved to r2 interact drop s t2 ok Stat:	s room r2 s EPlayerState ALIVE s itemList t1 pair t2 t1 active true t1 room r1 t2 pair t1 t2 active false t2 room r2 r1 effects TRANSISTOR_IN r1 itemList t1 r1 studentList r1 teacherList r1 cleanerList	r1 neighbouringRooms r1 sticky false r1 cleaned false r1 numOfVisitors 0 r2 effects TRANSISTOR_IN r2 itemList t1 r2 studentList s r2 teacherList r2 cleanerList r2 neighbouringRooms r2 sticky false r2 cleaned false r2 numOfVisitors 1
--	--	--

### 8.2.4 Transistor (nem párosított) lehelyezés

- Leírás

A két nem szomszédos szobában, az egyikben egy párosítatlan transistort helyez le a játékos, a másikban pedig eldobásra kerül egy másik transistor.

- Ellenőrzött funkcionalitás, várható hibahelyek

Szoba létrehozása.	Játékos létrehozása	Transistor létrehozása.
Transistor eldobása.	Játékos szobába lépése.	

- Bemenet

create room r1	create room r2 create student s	create transistor t1 create transistor t2
----------------	------------------------------------	--

link r1 s link s t1 link s t2	interact drop s t1 link -f r2 s interact drop s t2	stat
<ul style="list-style-type: none"> <li>● <b>Elvárt kimenet</b></li> </ul> room r1 created room r2 created student s created transistor t1 created transistor t2 created s moved to r1 t1 added to s t2 added to s interact drop s t1 ok s moved to r2 interact drop s t2 ok Stat: s room r2	s EPlayerState ALIVE s itemList t1 pair t1 active false t1 room r1 t2 pair t2 active false t2 room r2 r1 effects TRANSISTOR_IN r1 itemList t1 r1 studentList r1 teacherList r1 cleanerList r1 neighbouringRooms	r1 sticky false r1 cleaned false r1 numOfVisitors 0 r2 effects TRANSISTOR_IN r2 itemList t1 r2 studentList s r2 teacherList r2 cleanerList r2 neighbouringRooms r2 sticky false r2 cleaned false r2 numOfVisitors 1

## 8.2.5 Transistor aktiválás

- **Leírás**

Két nála lévő transistort a játékos aktiválni próbál. Az első aktiválódik, a másodikat nem tudja aktiválni, amíg nincs párja. Párosítás után tudja aktiválni.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Játékos létrehozása. Transistor párosítása.	Transistor létrehozása.	Transistor aktiválása.
--	-------------------------	------------------------

- **Bemenet**

create student s  
create transistor t1  
create transistor t2

- **Elvárt kimenet**

student s created  
transistor t1 created  
transistor t2 created  
t1 added to s  
t2 added to s  
interact activate s t1 ok  
stat:  
s room r2  
s EPlayerState ALIVE

link s t1  
link s t2  
interact activate s t1  
stat

interact pair s t1 t2  
interact activate s t1  
stat

s itemList  
t1 pair  
t1 active false  
t1 room  
t2 pair  
t2 active false  
t2 room  
interact pair s t1 t2 ok  
interact activate s t1 ok  
stat:

s room r2  
s EPlayerState ALIVE  
s itemList  
t1 pair t2  
t1 active true  
t1 room  
t2 pair t1  
t2 active false  
t2 room

## 8.2.6 Sajt gázosít

- **Leírás**

A játékos egy szobában állva azt elgázosítja, ha használja a nála lévő sajtot.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Szoba létrehozás

Hallgató létrehozás

Sajt tárgy létrehozás

Sajt tárgy használata

- **Bemenet**

create room r1 create student s	create cheese c link r1 s link s c	interact use s c stat
● <b>Elvárt kimenet</b>  room r1 created student s created cheese c created s moved to r1 c added to s interact use s c ok Stat:	s room r1 s EPlayerState ALIVE s itemList c c timer 2 c isUsed true c room r1 r1 effects POISONED r1 itemList	r1 studentList s r1 teacherList r1 cleanerList r1 neighbouringRooms r1 sticky false r1 cleaned false r1 numOfVisitors 1

## 8.2.7 Légfrissítő tisztít, maszk megvéd

- **Leírás**

A játékos egy gázos szobában lépés után használja a nála lévő légfrissítőt. A játékosnál van egy maszk, ami megvédi a gáztól.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Légrissítő működése	Légrissítő létrehozása	Hallgató létrehozása
Szoba létrehozása	Maszk működése	Maszk létrehozása
Időzítő működése	Játékos lépés	

- **Bemenet**

create room r1 create room r2 link r1 r2 create student s	create mask m create purifier p link r1 s link s p	link s m state r2 POISONED link r2 s stat
● <b>Elvárt kimenet</b>  room r1 created room r2 created r1 linked to r1 student s created mask m created purifier p created s moved to r1 p added to s m added to s r2 state set to POISONED s moved to r2 Stat:	s room r2 s EPlayerState ALIVE s itemList p, m m timer 2 m functional true m student s m fake false p isUsed true r1 effects r1 itemList r1 studentList r1 teacherList r1 cleanerList	r1 neighbouringRooms r2 r1 sticky false r1 cleaned false r1 numOfVisitors 0 r2 effect r2 itemList r2 studentList s r2 teacherList r2 cleanerList r2 neighbouringRooms r1 r2 sticky false r2 cleaner true r2 numOfVisitors 1

## 8.2.8 Sör megvéd, majd lejár

- **Leírás**

A játékos egy szobában állva, ha használja a nála lévő sört, és a sör időkeretén belül találkozik oktatóval. Ez után lejár a sör időzítője és az oktató megöli a hallgatót.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Sör létrehozása	Sör működése	Hallgató létrehozása
Játékos lépés	Oktató létrehozása	Időzítő működése
Szoba létrehozása	Játékos lépés	Hallgató halál
Időzítő lejárta	Timer parancs	

### ● Bemenet

create room r1  
create room r2  
link r1 r2  
create student s

create teacher t  
create beer b  
link r1 s  
link r2 t  
link s b

interact use s b  
link r2 s  
control timer 2  
stat

### ● Elvárt kimenet

room r1 created  
room r2 created  
r1 linked to r2  
student s created  
teacher t created  
beer b created  
s moved to r1  
t moved to r2  
b added to s  
interact use b ok  
s moved to r2  
timer 2 ok  
Stat:

s room r2  
s EPlayerState DEAD  
s itemList b  
t room r2  
t EPlayerState ALIVE  
t itemList  
b timer 0  
b state DISABLED  
r1 effects  
r1 itemList  
r1 studentList  
r1 teacherList  
r1 cleanerList

r1 neighbouringRooms r2  
r1 sticky false  
r1 cleaned false  
r1 numOfVisitors 0  
r2 effect  
r2 itemList  
r2 studentList s  
r2 teacherList t  
r2 cleanerList  
r2 neighbouringRooms r1  
r2 sticky false  
r2 cleaner true  
r2 numOfVisitors 2

## 8.2.9 Logarléc nyer

### ● Leírás

A játékos egy szobában állva, felveszi a ott lévő logarlécet

### ● Ellenőrzött funkcionalitás, várható hibahelyek

Tárgyfelvétel funkció  
Hallgató létrehozása

Logarléc működése  
Szoba létrehozása

Logarléc létrehozása

### ● Bemenet

create room r1  
create student s

create logarlec l  
link r1 s

link s l  
stat

### ● Elvárt kimenet

room r1 created  
student s created  
logarlec l created  
s moved to r1  
l added to s  
Victory

Stat:  
s EPlayerState ALIVE  
s itemList l  
l fake false  
r1 effects  
r1 itemList  
r1 studentList s

r1 teacherList  
r1 cleanerList  
r1 neighbouringRooms  
r1 sticky false  
r1 cleaned false  
r1 numOfVisitors 1

## 8.2.10 Logarléc hamis

### ● Leírás

A játékos egy szobában állva, felveszi a ott lévő hamis logarlécet

### ● Ellenőrzött funkcionalitás, várható hibahelyek

Tárgyfelvétel funkció  
Hallgató létrehozása

Hamis Logarléc működése  
Szoba létrehozása

Logarléc létrehozása

### ● Bemenet

create room r1  
create student s

create logarlec l  
state l fake true  
link r1 s

link s l  
stat

- Elvárt kimenet

```
room r1 created
student s created
logarlec l created
l state fake set to true
s moved to r1
l added to s
```

Stat:  
 s EPlayerState ALIVE  
 s itemList l  
 l fake true  
 r1 effects  
 r1 itemList  
 r1 studentList s

r1 teacherList  
 r1 cleanerList  
 r1 neighbouringRooms  
 r1 sticky false  
 r1 cleaned false  
 r1 numOfVisitors 1

## 8.2.11 Maszk lejár

- Leírás

A játékos egyik szobában használja a nála lévő maszkot, a hatás időtartamán belül átlép egy elgázosított szobába. A maszk hatása lejár, míg a szoba gázos marad. A hallgatót a gáz miatt elájul.

- Ellenőrzött funkcionalitás, várható hibahelyek

Timer működése

Maszk működése

Maszk létrehozása

Szoba létrehozása

Játékos lépés

Gázos szoba működése

- Bemenet

```
create room r1
create room r2
link r1 r2
```

create student s
create mask m
state r2 POISONED
link r1 s

link s m
link r2 s
controller timer 2
stat

- Elvárt kimenet

```
room r1 created
room r2 created
r1 linked to r2
student s created
mask m created
s moved to r1
m added to s
r2 state set to POISONED
s moved to r2
timer 2 ok
Stat:
s room r2
```

s EPlayerState
UNCONSCIOUS
s itemList m
m timer 0
m functional false
m student s
m fake false
r1 effects
r1 itemList
r1 studentList
r1 teacherList
r1 cleanerList
r1 neighbouringRooms r2

r1 sticky false
r1 cleaned false
r1 numOfVisitors 0
r2 effect POISONED
r2 itemList
r2 studentList s
r2 teacherList
r2 cleanerList
r2 neighbouringRooms r1
r2 sticky false
r2 cleaner true
r2 numOfVisitors 1

## 8.2.12 Maszk hamis

- Leírás

A játékos egyik szobában felvesz egy hamis maszkot, a hatás időtartamán belül átlép egy elgázosított szobába.

- Ellenőrzött funkcionalitás, várható hibahelyek

Gázos szoba működése

Hamis Maszk működése

Maszk létrehozása

Szoba létrehozása

Játékos lépés

- Bemenet

```
create room r1
create room r2
link r1 r2
```

create student s
create mask m
state r2 POISONED
link r1 s

link s m
link r2 s
controller timer 2
stat

- Elvárt kimenet

```
room r1 created
room r2 created
r1 linked to r2
student s created
```

mask m created
s moved to r1
m added to s
r2 state set to POISONED

s moved to r2
timer 2 ok
Stat:
s room r2

s EPlayerState UNCONSCIOUS s itemList m m timer 0 m functional false m student s m fake false r1 effects r1 itemList	r1 studentList r1 teacherList r1 cleanerList r1 neighbouringRooms r2 r1 sticky false r1 cleaned false r1 numOfVisitors 0 r2 effect POISONED r2 itemList	r2 studentList s r2 teacherList r2 cleanerList r2 neighbouringRooms r1 r2 sticky false r2 cleaner true r2 numOfVisitors 1
--	---	---

## 8.2.13 Szivacs működik

- Leírás

A játékos egy szobában áll és van nála szivacs, oktató belép a szobába. Oktató lebénül. Szivacs elhasználódik

- Ellenőrzött funkcionalitás, várható hibahelyek

Játékos létrehozása	Szoba létrehozása	Játékos mozgás
Szivacs létrehozása	Sziva működése	

- Bemenet

create room r1 create room r2 link r1 r2	create student st create teacher t create sponge sp link r1 st	link r2 t link st sp link r1 t stat
--	---	--

- Elvárt kimenet

room r1 created room r2 created r1 linked to r2 student st created teacher t created sponge sp created st moved to r1 t moved to r2 sp added to st t moved to r1 Stat: s room r1 s EPlayerState ALIVE	s itemList sp t room r1 t EPlayerState UNCONSCIOUS t itemList sp timer 2 sp functional true r1 effects r1 itemList r1 studentList s r1 teacherList t r1 cleanerList r1 neighbouringRooms r2 r1 sticky false	r1 cleaned false r1 numOfVisitors 2 r2 effect r2 itemList r2 studentList r2 teacherList r2 cleanerList r2 neighbouringRooms r1 r2 sticky false r2 cleaner true r2 numOfVisitors
---	--	---

## 8.2.14 TVSZ megvéd

- Leírás

A játékos egy szobában állva, mikor van nála TVSZ aminek van még ereje, oktató belép a szobába.

- Ellenőrzött funkcionalitás, várható hibahelyek

TVSZ létrehozása	TVSZ működése	Játékos lépés
Oktató létrehozása	Hallgató létrehozása	Szoba létrehozása

- Bemenet

create room r1 create room r2 link r1 r2	create student st create teacher t create TVSZ tvsz link r1 st	link r2 t link st tvsz link r1 t stat
--	---	--

- Elvárt kimenet

room r1 created room r2 created	r1 linked to r2 student s created	teacher t created TVSZ tvsz created
------------------------------------	--------------------------------------	--

s moved to r1 t moved to r2 tvsz added to s t moved to r1 Stat: s room r1 s EPlayerState ALIVE s itemList tvsz t room r1 t EPlayerState UNCONSCIOUS t itemList	tvsz hitpoints 2 tvsz fake false r1 effects r1 itemList r1 studentList s r1 teacherList t r1 cleanerList r1 neighbouringRooms r2 r1 sticky false r1 cleaned false r1 numOfVisitors 2 r2 effect	r2 itemList r2 studentList r2 teacherList r2 cleanerList r2 neighbouringRooms r1 r2 sticky false r2 cleaner true r2 numOfVisitors
---	---	--

## 8.2.15 TVSZ lejár

- Leírás

Egy hallgató egy szobában áll és a nála lévő TVSZnek már nincs ereje, oktató belép a szobába. A TVSZ nem védi meg a hallgatót.

- Ellenőrzött funkcionalitás, várható hibahelyek

TVSZ létrehozása	TVSZ működése	Játékos lépés
Oktató létrehozása	Hallgató létrehozása	Szoba létrehozása

- Bemenet

create room r1  
create room r2  
link r1 r2  
create student st

create teacher t  
create TVSZ tvsz  
state TVSZ hitPoints 0  
link r1 st

link r2 t  
link st tvsz  
link r1 t  
stat

- Elvárt kimenet

room r1 created  
room r2 created  
r1 linked to r2  
student s created  
teacher t created  
TVSZ tvsz created  
TVSZ hitPoints set to 0  
s moved to r1  
t moved to r2  
tvsz added to s  
t moved to r1  
Stat:

s room r1  
s EPlayerState DEAD  
s itemList tvsz  
t room r1  
t EPlayerState ALIVE  
t itemList  
tvsz hitpoints 0  
tvsz fake false  
r1 effects  
r1 itemList  
r1 studentList s  
r1 teacherList t  
r1 cleanerList

r1 neighbouringRooms r2  
r1 sticky false  
r1 cleaned false  
r1 numOfVisitors 2  
r2 effect  
r2 itemList  
r2 studentList  
r2 teacherList  
r2 cleanerList  
r2 neighbouringRooms r1  
r2 sticky false  
r2 cleaner true  
r2 numOfVisitors 0

## 8.2.16 TVSZ hamis

- Leírás

Egy hallgató egy szobában áll és a nála lévő TVSZ hamis, egy oktató belép a szobába.

- Ellenőrzött funkcionalitás, várható hibahelyek

TVSZ létrehozása	Hamis TVSZ működése	Játékos lépés
Oktató létrehozása	Hallgató létrehozása	Szoba létrehozása

- Bemenet

create room r1  
create room r2  
link r1 r2  
create student st

create teacher t  
create TVSZ tvsz  
state TVSZ fake true  
link r1 st

link r2 t  
link st tvsz  
link r1 t  
stat

- Elvárt kimenet

```

room r1 created
room r2 created
r1 linked to r2
student s created
teacher t created
TVSZ tvsz created
TVSZ fake set to true
s moved to r1
t moved to r2
tvsz added to s
t moved to r1
Stat:
s room r1

```

```

s EPlayerState DEAD
s itemList tvsz
t room r1
t EPlayerState ALIVE
t itemList
tvsz hitpoints 3
tvsz fake true
r1 effects
r1 itemList
r1 studentList s
r1 teacherList t
r1 cleanerList
r1 neighbouringRooms r2
r1 sticky false

```

```

r1 cleaned false
r1 numOfVisitors 2
r2 effect
r2 itemList
r2 studentList
r2 teacherList
r2 cleanerList
r2 neighbouringRooms r1
r2 sticky false
r2 cleaned false
r2 numOfVisitors 0

```

## 8.2.17 Mozgásteszт

- Leírás

Az összes játékosból egyet különböző szobákból egy közös szobába mozgatunk. Sorban egy hallgatót, egy oktatót és egy takarítót fogunk a közös szobába mozgatni.

- Ellenőrzött funkcionalitás, várható hibahelyek

Játékosok létrehozása

Játékosok egymásra hatása

Játékosok mozgása

- Bemenet

```

create room r1
create room r2
create room r3
create room r4
link r1 r4
link r2 r4

```

```

link r3 r4
create student s
create teacher t
create cleaner c
link r1 s
link r2 t
link r3 c

```

```

link r4 s
link r4 t
link r4 c
stat

```

- Elvárt kimenet

```

room r1 created
room r2 created
room r3 created
room r4 created
r1 linked to r4
r2 linked to r4
r3 linked to r4
student s created
teacher t created
cleaner c created
s moved to r1
t moved to r2
c moved to r3
s moved to r4
t moved to r4
c moved to r4
Stat:
s room r4
s EPlayerState DEAD
s itemList

```

```

t room r1
t EPlayerState ALIVE
t itemList
c room r4
c EPlayerState ALIVE
c itemList
r1 effects
r1 itemList
r1 studentList
r1 teacherList t
r1 cleanerList
r1 neighbouringRooms r4
r1 sticky false
r1 cleaned false
r1 numOfVisitors 1
r2 effect
r2 itemList
r2 studentList
r2 teacherList
r2 cleanerList
r2 neighbouringRooms r4

```

```

r2 sticky false
r2 cleaned false
r2 numOfVisitors 0
r3 effect
r3 itemList
r3 studentList
r3 teacherList
r3 cleanerList
r3 neighbouringRooms r4
r3 sticky false
r3 cleaned false
r3 numOfVisitors 0
r4 effect
r4 itemList
r4 studentList s
r4 teacherList
r4 cleanerList c
r4 neighbouringRooms r1,r2,r3
r2 sticky false
r2 cleaned true
r2 numOfVisitors 2

```

## 8.2.18 Szoba split

- Leírás

Egy adott szobát splittelünk és a szabályoknak megfelelő szobákat fogunk kapni split után.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Szoba létrehozás

Hallgató létrehozás

Logarléc létrehozás

Szoba split

Random kikapcsolása

- **Bemenet**

```
create room r1
create student s
create logarlec L1
```

```
create logarlec L2
link r1 s
link r1 L1
link r1 L2
```

```
control random off
interact split r1
stat
```

- **Elvárt kimenet**

```
room r1 created
student s created
logarlec L1 created
logarlec L2 created
s moved to r1
L1 moved to r1
L2 moved to r1
random off ok
split r1 ok
Stat
```

```
s room r1_1
s EPlayerState ALIVE
s itemList
r1_1 effects
r1_1 itemList L1,L2
r1_1 studentList s
r1_1 teacherList
r1_1 cleanerList
r1_1 neighbouringRooms r1_2
r1_1 sticky false
r1_1 cleaned false
r1_1 cleaned false
```

```
r1_1 numOfVisitors 1
r1_2 effects
r1_2 itemList
r1_2 studentList
r1_2 teacherList
r1_2 cleanerList
r1_2 neighbouringRooms r1_1
r1_2 sticky false
r1_2 cleaned false
r1_2 numOfVisitors 0
```

## 8.2.19 Szoba merge

- **Leírás**

Egy adott szobát mergelünk egy másikkal, és a szabályoknak megfelelően jön létre egy új szoba.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Szoba létrehozás

Hallgató létrehozás

Logarléc létrehozás

Szoba split

- **Bemenet**

```
create room r1
create room r2
create student s
create logarlec L1
```

```
create logarlec L2
link r1 s
link r1 L1
link r1 L2
```

```
interact merge r1 r2
stat
```

- **Elvárt kimenet**

```
room r1 created
room r2 created
student s created
logarlec L1 created
logarlec L2 created
s moved to r1
L1 moved to r1
L2 moved to r1
```

```
merge r1 r2 ok
Stat
s room r1r2
s EPlayerState ALIVE
s itemList
r1r2 effects
r1r2 itemList L1,L2
r1r2 studentList s
```

```
r1r2 teacherList
r1r2 cleanerList
r1r2 neighbouringRooms
r1r2 sticky false
r1r2 cleaned false
r1r2 numOfVisitors 1
```

## 8.2.20 Átkozott szoba

- **Leírás**

Csinálunk egy átkozott szobát amiben van egy hallgató. A hallgatót megpróbáljuk kiléptetni, míg egy oktatót megpróbálunk beléptetni. Egyikük se fog tudni sikeresen mozogni.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Átkozott szoba hatása

Szoba létrehozás

Hallgató létrehozás

## Oktató létrehozás

**• Bemenet**

```
create room r1
create room r2
link r1 r2
create student s
create teacher t
```

```
link r1 s
link r2 t
state r1 CURSED
link r1 t
link r2 s
```

stat

**• Elvárt kimenet**

```
room r1 created
room r2 created
r1 linked to r2
student s created
teacher t created
s moved to r1
t moved to r2
r2 state set to CURSED
t failed to move to r1
s failed to move to r2
Stat
```

```
s room r1
s EPlayerState ALIVE
s itemList
r1 effects CURSED
r1 itemList
r1 studentList s
r1 teacherList
r1 cleanerList
r1 neighbouringRooms r2
r1 sticky false
r1 cleaned false
r1 numofVisitors 1
```

```
r2 effects
r2 itemList
r2 studentList
r2 teacherList t
r2 cleanerList
r2 neighbouringRooms r1
r2 sticky false
r2 cleaned false
r2 numofVisitors 1
```

**8.2.21 Gázos kilépés****• Leírás**

Gázos szobából kilépve a hallgatónak az aktiválódott maszkja deaktiválódik.

**• Ellenőrzött funkcionalitás, várható hibahelyek**

Időzítő működése

Játékos lépés

Hallgató létrehozása

Szoba létrehozása

Maszk működése

Maszk létrehozása

Maszk deaktiválása

**• Bemenet**

```
create room r1
create room r2
link r1 r2
create student s
```

```
create mask m
link r1 s
link s m
state r2 POISONED
link r2 s
```

```
link r1 s
time 5 ok
stat
```

**• Elvárt kimenet**

```
room r1 created
room r2 created
r1 linked to r2
student s created
mask m created
s moved to r1
m added to s
r2 state set to POISONED
s moved to r2
s moved to r1
control time 5
Stat:
s room r1
```

```
s EPlayerState ALIVE
s itemList m
m timer 1
m functional true
m student s
m fake false
r1 effects
r1 itemList
r1 studentList s
r1 teacherList
r1 cleanerList
r1 neighbouringRooms r2
r1 sticky false
```

```
r1 cleaned false
r1 numofVisitors 1
r2 effect POISONED
r2 itemList
r2 studentList
r2 teacherList
r2 cleanerList
r2 neighbouringRooms r1
r2 sticky false
r2 cleaner true
r2 numofVisitors 0
```

**8.2.22 Felkelés****• Leírás**

Egy hallgató gázos szobába kerül és egy időre eszméletét veszti. A gázt kitisztítjuk a szobából. Az idő lejárta után felkel a hallgató. Állapotkérdezés a gázosítás előtt után és az időzítő lejárta után.

### ● Ellenőrzött funkcionalitás, várható hibahelyek

Szoba létrehozás

Hallgató létrehozás

Szoba kitisztulása

Szoba gázosítása

Bénítás

Időzítő lejártja

### ● Bemenet

control timer off

create room r1

create student s

```
link s c
stat
state r1 effects POISONED
```

```
stat
timer 50
stat
```

### ● Elvárt kimenet

control timer off

room r1 created

student s created

s added to r1

stat:

s room r1

s EPlayerState ALIVE

s itemList

r1 effect

r1 itemList

r1 studentList s

r1 teacherList

r1 cleanerList

r1 neighbouringRooms

```
r1 sticky false
r1 cleaned false
r1 numOfVisitors 1
r1 effects set to POISONED
stat:
s room r1
s EPlayerState UNCONSCIOUS
s itemList
r1 effect POISONED
r1 itemList
r1 studentList s
r1 teacherList
r1 cleanerList
r1 neighbouringRooms
r1 sticky false
```

```
r1 cleaned false
r1 numOfVisitors 1
stat:
s room r1
s EPlayerState ALIVE
s itemList
r1 effect
r1 itemList
r1 studentList s
r1 teacherList
r1 cleanerList
r1 neighbouringRooms
r1 sticky false
r1 cleaned false
r1 numOfVisitors 1
```

## 8.2.23 Felvétel/Eldobás

### • Leírás

Az összes tárgyból egyet adunk egy hallgatónak és egy oktatónak is. A tárgyak a működésének megfelelően aktiválódnak felvételkor, és bekerülnek az oktató/hallgató tárgyai közé. Felvétel után az összes tárgyat eldobjuk, melyek a játékosok szobájába fognak kerülni.

### • Ellenőrzött funkcionalitás, várható hibahelyek

Transistor létrehozása	Mask létrehozása (igaz/hamis)	TVSZ létrehozása (igaz/hamis)
Sajt létrehozása	Logarléc létrehozása (igaz/hamis)	Légtisztító létrehozása
Sör létrehozása	Szivacs létrehozása	Transistor felvétele
Mask felvétele (igaz/hamis)	TVSZ felvétele (igaz/hamis)	Sajt felvétele
Logarléc felvétele (igaz/hamis)	Légtisztító felvétele	Sör felvétele
Szivacs felvétele	Transistor eldobása	Mask eldobása (igaz/hamis)
TVSZ eldobása (igaz/hamis)	Sajt eldobása	Logarléc eldobása (igaz/hamis)
Légtisztító eldobása	Sör eldobása	Szivacs eldobása
Tanár létrehozása	Játékos létrehozása	Szoba létrehozása

### • Bemenet

```
create teacher te1
create room r1
link r1 te1

create cheese ch1
link ch1 r1
interact pickup te1 ch1
interact drop te1 ch1
create transistor t1
link t1 r1
interact pickup te1 t1
interact drop te1 t1
create logar l1
link l1 r1
interact pickup te1 l1
interact drop te1 l1
create logar l1f
state l1f fake true
link l1f r1
interact pickup te1 l1f
interact drop te1 l1f
create tvsz tvsz1
link tvsz1 r1
interact pickup te1 tvsz1
interact drop te1 tvsz1
create tvsz tvsz1f
state tvsz1f fake true
link tvsz1f r1
interact pickup te1 tvsz1f
interact drop te1 tvsz1f
create purifier p1
link p1 r1
interact pickup te1 p1
interact drop te1 p1
create beer b1
```

### • Elvárt kimenet

```
teacher te1 created
room r1 created
te1 added to r1
cheese ch1 created
ch1 added to r1
interact pickup te1 ch1 ok
interact drop te1 ch1 ok
transistor t1 created
```

```
link b1 r1
interact pickup te1 b1
interact drop te1 b1
create sponge sp1
link sp1 r1
interact pickup te1 sp1
interact drop te1 sp1
create mask m1
link m1 r1
interact pickup te1 m1
interact drop te1 m1
create mask m1f
state m1f fake true
link m1f r2
interact pickup te1 m1f
interact drop te1 m1f

create student s1
create room r2
link r2 s1

create cheese ch2
link ch2 r2
interact pickup st1 ch2
interact drop st1 ch2
create transistor t2
link t2 r2
interact pickup st1 t2
interact drop st1 t2
create logar l2
link l2 r2
interact pickup st1 l2
interact drop st1 l2
create logar l2f
state l2f fake true

create mask m2
link m2 r2
interact pickup st1 m2
interact drop st1 m2
create mask m2f
state m2f fake true
link m2f r2
interact pickup st1 m2f
interact drop st1 m2f
stat
```

```
t1 added to r1
interact pickup te1 t1 ok
interact drop te1 t1 ok
logar l1 created
l1 added to r1
interact pickup te1 l1 ok
interact drop te1 l1 ok
logar l1f created
```

```
l1f state set to true
l1f added to r1
interact pickup te1 l1f ok
interact drop te1 l1f ok
tvsz tvsz1 created
tvsz1 added to r1
interact pickup te1 tvsz1 ok
interact drop te1 tvsz1 ok
```

```

tvsz tvsz1f created
tvsz1f state set to true
tvsz1f added to r1
interact pickup te1 tvsz1f ok
interact drop te1 tvsz1f ok
purifier p1 created
p1 added to r1
interact pickup te1 p1 ok
interact drop te1 p1 ok
beer b1 created
b1 added to r1
interact pickup te1 b1 ok
interact drop te1 b1 ok
sponge sp1 created
sp1 added to r1
interact pickup te1 sp1 ok
interact drop te1 sp1 ok
mask m1 created
m1 added to r1
interact pickup te1 m1 ok
interact drop te1 m1 ok
mask m1f created
m1f fake set to true
m1f added to r2
interact pickup te1 m1f ok
interact drop te1 m1f ok
student st1 created
room r2 created
st1 added to r2
cheese ch2 created
ch2 added to r2
interact pickup st1 ch2 ok
interact drop st1 ch2 ok
transistor t2 created
t2 added to r2
interact pickup st1 t2 ok
interact drop st1 t2 ok
logar l2 created
l2 added to r2
interact pickup st1 l2 ok
interact drop st1 l2 ok
logar lif created
lif state set to true
l2f added to r2
interact pickup st1 l2f ok
interact drop st1 l2f ok
tvsz tvsz2 created
tvsz2 added to r2
interact pickup st1 tvsz2 ok

```

```

interact drop st1 tvsz2 ok
tvsz tvsz2f created
tvsz2f state set to true
tvsz2f added to r2
interact pickup st1 tvsz2f ok
interact drop st1 tvsz2f ok
purifier p2 created
p2 added to r2
interact pickup st1 p2 ok
interact drop st1 p2 ok
beer b2 created
b2 added to r2
interact pickup st1 b2 ok
interact drop st1 b2 ok
sponge sp2 created
sp2 added to r2
interact pickup st1 sp2 ok
interact drop st1 sp2 ok
mask m2 created
m2 added to r2
interact pickup st1 m2 ok
interact drop st1 m2 ok
mask m2f created
m2f state set to true
m2f added tp r2
interact pickup st1 m2f ok
interact drop st1 m2f ok
stat:
te room r1
te EPlayerState ALIVE
te itemList
st room r2
st EPlayerState ALIVE
st itemList
ch1 isUsed false
ch1 room r1
t1 active fals
t1 room r1
t1 pair null
l1 fake false
l1f fake true
tvsz1 hitpoint 3
tvsz1 fake fasle
tvsz1f
tvsz1 hitpoint 3
tvsz1 fake true
p1 used false
b1 state INACTIVE
sp1 funcional false

```

```

m1 functional true
m1 student null
m1 fake false
m1f functional true
m1f student null
m1f fake true
ch2 isUsed false
ch2 room r1
t2 active fals
t2 room r1
t2 pair null
l2 fake false
l1f fake true
tvsz2 hitpoint 3
tvsz2 fake fasle
tvsz2 hitpoint 3
tvsz2 fake true
p2 used false
b2 state INACTIVE
sp2 funcional false
m2 functional true
m2 student null
m2 fake false
m2f functional true
m2f student null
m2f fake truer1 effects
TRANSISTOR_IN
r1 itemList ch1 t1 l1 l1f tvsz1
tvsz1f p1 b1 sp1 m1 m1f
r1 studentList
r1 teacherList te
r1 cleanerList
r1 neighbouringRooms
r1 sticky false
r1 cleaned false
r1 numOfVisitors 1
r2 effect TRANSISTOR_IN
r2 itemList ch2 t2 l2 l2f tvsz2
tvsz2f p2 b2 sp2 m2 m2f
r2 studentList st
r2 teacherList
r2 cleanerList
r2 neighbouringRooms
r2 sticky false
r2 cleaned false
r2 numOfVisitors 1

```

### 8.3 A tesztelést támogató programok tervezése

A tesztek kiértékelésére windows alatt futtatható batch fájlokat fogunk létrehozni.

Minden tesztesethez külön fájl fog tartozni, a fájlok a teszteset bemeneti parancsait továbbítják a prototípus programnak. A tesztek a kimenetetüket minden saját névvel látják el (pl. test1.txt, test2.txt...). minden tesztesethez tartozik egy előre összeállított expected.txt is (expected1.txt, stb), ami az elvárt kimenetet tartalmazza.

A parancsok lefutása után a batch fájl összehasonlítja az elvárt txt fájlt a kapottall git diff parancs segítségével. Ez egy resultX.txt fájlt fog elkészíteni. Ha a resultX.txt üres, az azt jelenti hogy nincs különbség az elvárt és a kapott kimenetek között. Ha van eltérés, akkor azok kerülnek a fájlba.

Létezik egy olyan batch fájl is, ami az összes tesztesetet futtatja.

A prototípus programban a tesztekben létrehozott objektumokhoz nevet egy `<string, object>` típusú map segítségével rendelünk.

A programban a különböző tesztek függvényei az objektumok állapotait fogják ellenőrizni, és ez alapján írnak a parancssorra kimenetet.

A különböző bemeneti parancsok rendeltetésszerű működéséhez `instanceof` operátort használunk. Így például a link parancsot szobák közti mozgásra vagy tárgyak játékosoknak adására is lehet használni.

A parancsok értelmezéséhez a parancssoron megadott bemeneti stringet először space-eknél feldaraboljuk. Az első szó alapján parancsot választunk. Létrehozásnál egy `<string, object>` map-be tesszük az objektumainkat. A többi parancsnál ha egy objektum nevét adjuk meg, akkor ebben a mapben fogjuk keresni az objektumot, aminek a publikus metódusait fogjuk meghívni. A prototípus összes parancsa egy osztályban található.

## 10. Prototípus beadása

### 10.1 Fordítási és futtatási útmutató

#### 10.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
<b>controller/</b>			
Controller.java	1695 B	Apr 28 23:32	Kontroller
Labyrinth.java	3396 B	Apr 28 23:32	Labirintus
TimedObject.java	1108 B	Apr 28 23:32	Időzítő objektum
Timer.java	1298 B	Apr 28 23:32	Időzítés
<b>model/</b>			
ERoomEffects.java	114 B	Apr 28 23:32	Hatások enumerátor
ICRoom.java	285 B	Apr 28 23:32	Szoba controller interface
ITimedEntity.java	83 B	Apr 28 23:32	Időzített interfész
ITimer.java	190 B	Apr 28 23:32	Időzítő interfésze
IVRoom.java	275 B	Apr 28 23:32	Szoba view interface
Room.java	13250 B	Apr 28 23:32	Szoba
<b>model/items</b>			
Beer.java	2746 B	Apr 28 23:32	Sör tárgy
Cheese.java	2654 B	Apr 28 23:32	Sajt tárgy
EBeerState.java	99 B	Apr 28 23:32	Sör állapotának enumerátora
IItem.java	2409 B	Apr 28 23:32	Tárgy interfésze
Logarlec.java	2106 B	Apr 28 23:32	Logarléc
Mask.java	3152 B	Apr 28 23:32	Maszk
Purifier.java	1759 B	Apr 28 23:32	Légfrissítő
Sponge.java	2612 B	Apr 28 23:32	Szivacs
Transistor.java	3234 B	Apr 28 23:32	Tranzisztor
TVSZ.java	2424 B	Apr 28 23:32	TVSZ
<b>model/player</b>			
Cleaner.java	1067 B	Apr 28 23:32	Takarító
EPlayerState.java	96 B	Apr 28 23:32	Játékos állapot enumerátora
Player.java	2594 B	Apr 28 23:32	Játékos
Student.java	1659 B	Apr 28 23:32	Tanuló
Teacher.java	1052 B	Apr 28 23:32	Oktató
<b>teszt/</b>			
CommandData.java	136 B	Apr 28 23:32	Lehetséges parancsok struktúra
ECommand.java	142 B	Apr 28 23:32	Parancsok Enumerátor
FileHandling.java	5966 B	Apr 29 00:21	Filekezelés
IPrintStat.java	126 B	Apr 28 23:32	Státusz kimenet interfész
TestRunner.java	12397 B	Apr 29 00:20	Tesztek futtatása
<b>teszt/test batch</b>			
runAllTest.bat	501 B	Apr 28 23:32	Összes tesztet indító batch file.
test1.bat	591 B	Apr 29 00:21	test1 futtatása
test10.bat	581 B	Apr 29 00:23	test10 futtatása
test11.bat	581 B	Apr 29 00:23	test11 futtatása
test12.bat	581 B	Apr 29 00:23	test12 futtatása

test13.bat	581 B	Apr 29 00:23	test13 futtatása
test14.bat	581 B	Apr 29 00:23	test14 futtatása
test15.bat	581 B	Apr 29 00:23	test15 futtatása
test16.bat	581 B	Apr 29 00:23	test16 futtatása
test17.bat	581 B	Apr 29 00:23	test17 futtatása
test18.bat	581 B	Apr 29 00:23	test18 futtatása
test19.bat	581 B	Apr 29 00:23	test19 futtatása
test2.bat	578 B	Apr 29 00:23	test2 futtatása
test20.bat	581 B	Apr 29 00:23	test20 futtatása
test21.bat	583 B	Apr 29 00:23	test21 futtatása
test22.bat	583 B	Apr 29 00:23	test22 futtatása
test23.bat	583 B	Apr 29 00:23	test23 futtatása
test3.bat	578 B	Apr 29 00:23	test3 futtatása
test4.bat	578 B	Apr 29 00:23	test4 futtatása
test5.bat	578 B	Apr 29 00:23	test5 futtatása
test6.bat	577 B	Apr 29 00:23	test6 futtatása
test7.bat	577 B	Apr 29 00:23	test7 futtatása
test8.bat	577 B	Apr 29 00:23	test8 futtatása
test9.bat	577 B	Apr 29 00:23	test9 futtatása
<b>test/test txt/test input</b>			
test1.txt	505 B	Apr 28 23:32	test1 bemenet
test10.txt	97 B	Apr 28 23:32	test10 bemenet
test11.txt	152 B	Apr 28 23:32	test11 bemenet
test12.txt	152 B	Apr 28 23:32	test12 bemenet
test13.txt	149 B	Apr 28 23:32	test13 bemenet
test14.txt	151 B	Apr 28 23:32	test14 bemenet
test15.txt	175 B	Apr 28 23:32	test15 bemenet
test16.txt	173 B	Apr 28 23:32	test16 bemenet
test17.txt	224 B	Apr 28 23:32	test17 bemenet
test18.txt	152 B	Apr 28 23:32	test18 bemenet
test19.txt	151 B	Apr 28 23:32	test19 bemenet
test2.txt	256 B	Apr 28 23:32	test2 bemenet
test20.txt	145 B	Apr 28 23:32	test20 bemenet
test21.txt	154 B	Apr 28 23:32	test21 bemenet
test22.txt	116 B	Apr 28 23:32	test22 bemenet
test23.txt	1988 B	Apr 28 23:32	test23 bemenet
test3.txt	208 B	Apr 28 23:32	test3 bemenet
test4.txt	185 B	Apr 28 23:32	test4 bemenet
test5.txt	165 B	Apr 28 23:32	test5 bemenet
test6.txt	94 B	Apr 28 23:32	test6 bemenet
test7.txt	161 B	Apr 28 23:32	test7 bemenet
test8.txt	177 B	Apr 28 23:32	test8 bemenet

test9.txt	78 B	Apr 28 23:32	test9 bemenet
<b>test/test txt/test expected</b>			
test1_exp.txt	2017 B	Apr 28 23:32	test1 elvárt kimenete
test10_exp.txt	2017 B	Apr 28 23:32	test10 elvárt kimenete
test11_exp.txt	2017 B	Apr 28 23:32	test11 elvárt kimenete
test12_exp.txt	2017 B	Apr 28 23:32	test12 elvárt kimenete
test13_exp.txt	2017 B	Apr 28 23:32	test13 elvárt kimenete
test14_exp.txt	1512 B	Apr 28 23:32	test14 elvárt kimenete
test15_exp.txt	1512 B	Apr 28 23:32	test15 elvárt kimenete
test16_exp.txt	1512 B	Apr 28 23:32	test16 elvárt kimenete
test17_exp.txt	1512 B	Apr 28 23:32	test17 elvárt kimenete
test18_exp.txt	1512 B	Apr 28 23:32	test18 elvárt kimenete
test19_exp.txt	1512 B	Apr 28 23:32	test19 elvárt kimenete
test2_exp.txt	2017 B	Apr 28 23:32	test2 elvárt kimenete
test20_exp.txt	1512 B	Apr 28 23:32	test20 elvárt kimenete
test21_exp.txt	1512 B	Apr 28 23:32	test21 elvárt kimenete
test22_exp.txt	1512 B	Apr 28 23:32	test22 elvárt kimenete
test23_exp.txt	1512 B	Apr 28 23:32	test23 elvárt kimenete
test3_exp.txt	2017 B	Apr 28 23:32	test3 elvárt kimenete
test4_exp.txt	2017 B	Apr 28 23:32	test4 elvárt kimenete
test5_exp.txt	2017 B	Apr 28 23:32	test5 elvárt kimenete
test6_exp.txt	1512 B	Apr 28 23:32	test6 elvárt kimenete
test7_exp.txt	1512 B	Apr 28 23:32	test7 elvárt kimenete
test8_exp.txt	1512 B	Apr 28 23:32	test8 elvárt kimenete
test9_exp.txt	2017 B	Apr 28 23:32	test9 elvárt kimenete

### 10.1.2 Fordítás

A fordításhoz a Visual Studio Code, “Language Support for Java by Red Hat” nevű extensiont használjuk. Ennek segítségével az Explorer fulben a Java Projects alatt található egy Export Jar nevű automatikus jar file generálására szolgáló gomb. Ezt megnyomva a feljövő opciók közül a FileHandling osztály main metódusából kell generálni a jar filet. Majd miután ezt megtettük, át kell nevezni a létrejövő “echo.jar”-t “projlab.jar”-ra.

### 10.1.3 Futtatás

A teszt a fordítás utáni jar file segítségével indítandó, neve “projlab.jar”, más név esetén átnevezendő és a src mappával egy szintre helyezendő. Az elkészített .bat fájl így hivatkozik a programra.

A “src/test/test\_batch” mappában a felsorolt egyes tesztek mellett egy tömbösített runAllTest.bat file. Ezt konzolos felület segítségével indítva, visszajelzésekkel nézhetjük meg a tesztek lefutását. Az elvárt és a kimeneti szöveges állományok különbségeit a “src/test/test\_batch/diff\_output” mappában látható fileokban találjuk, üres file, teljes egyezést,

sikeres futást jelent. A runAllTest.bat file a program futtatása után ezt ellenőrzi, visszajelzést ad.

## **10.2 Tesztek jegyzőkönyvei**

### **10.2.1 \* -os teszt**

Teszteleő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

### **10.2.2 Transistor teleportál teszt**

Teszteleő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

### **10.2.3 Transistor (aktiválatlan) lehelyezés teszt**

Teszteleő neve	Rékó Márton
Teszt időpontja	2024.04.29 0:33

### **10.2.4 Transistor (nem párosított) lehelyezés teszt**

Teszteleő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

### **10.2.5 Transistor aktiválás teszt**

Teszteleő neve	Rékó Márton
Teszt időpontja	2024.04.29 0:33

### **10.2.6 Sajt teszt**

Teszteleő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

### **10.2.7 Légrissítő tisztít, maszk megvéd teszt**

Teszteleő neve	Rékó Márton
Teszt időpontja	2024.04.29 0:33

### **10.2.8 Sör megvéd, majd lejár teszt**

Teszteleő neve	Rékó Márton
Teszt időpontja	2024.04.29 0:33

### **10.2.9 Logarléc teszt**

Teszteleő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

### **10.2.10 Logarléc teszt**

Teszteleő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

**10.2.11 Maszk lejár teszt**

Teszteleő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

**10.2.12 Maszk hamis teszt**

Teszteleő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

**10.2.13 Szivacs működik teszt**

Teszteleő neve	Rékó Márton
Teszt időpontja	2024.04.29 0:33

**10.2.14 TVSZ megvéd teszt**

Teszteleő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

**10.2.15 TVSZ lejár teszt**

Teszteleő neve	Rékó Márton
Teszt időpontja	2024.04.29 0:33

**10.2.16 TVSZ hamis teszt**

Teszteleő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

**10.2.17 Mozgásteszt teszt**

Teszteleő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

**10.2.18 Szoba split teszt**

Teszteleő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

**10.2.19 Szoba merge teszt**

Teszteleő neve	Rékó Márton
Teszt időpontja	2024.04.29 0:33

**10.2.20 Átkozott szoba teszt**

Teszteleő neve	Rékó Márton
Teszt időpontja	2024.04.29 0:33

**10.2.21 Gázos kilépés teszt**

Teszteleő neve	Rékó Márton
Teszt időpontja	2024.04.29 0:33

**10.2.22 Felkelés teszt**

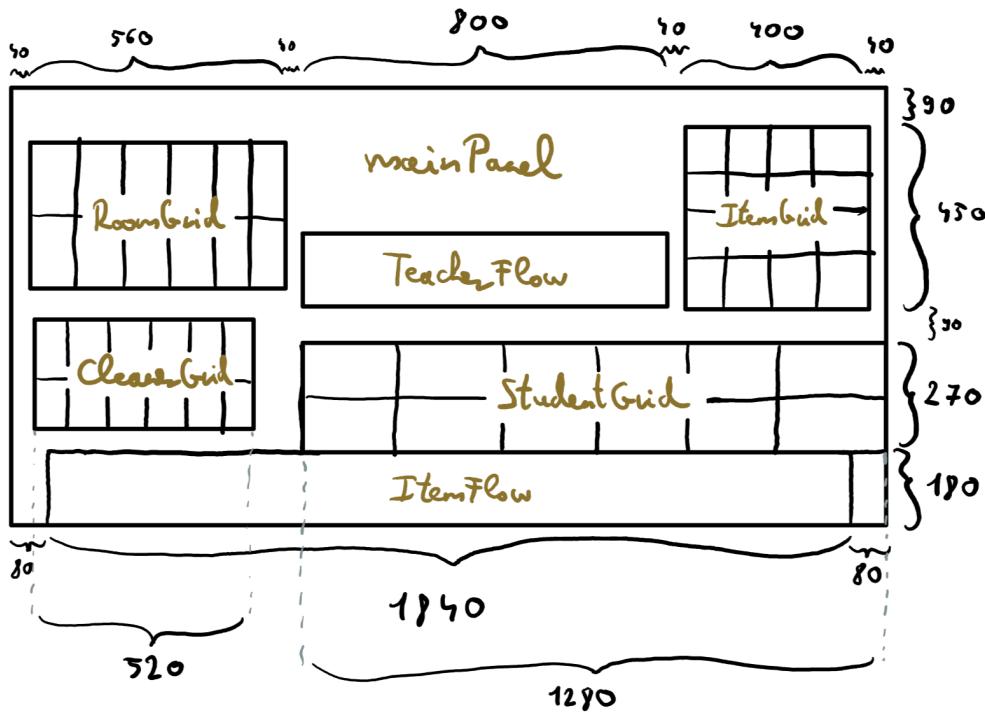
Tesztelő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

**10.2.23 Felvétel/Eldobás teszt**

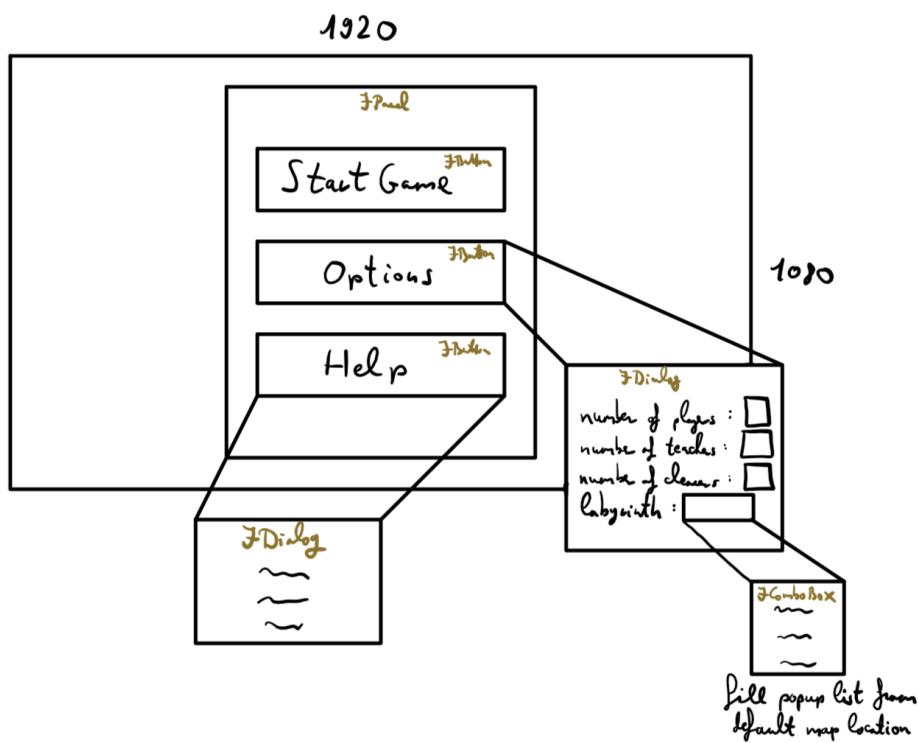
Tesztelő neve	Csirmaz Gáspár
Teszt időpontja	2024.04.29 0:33

## 11. Grafikus felület specifikációja

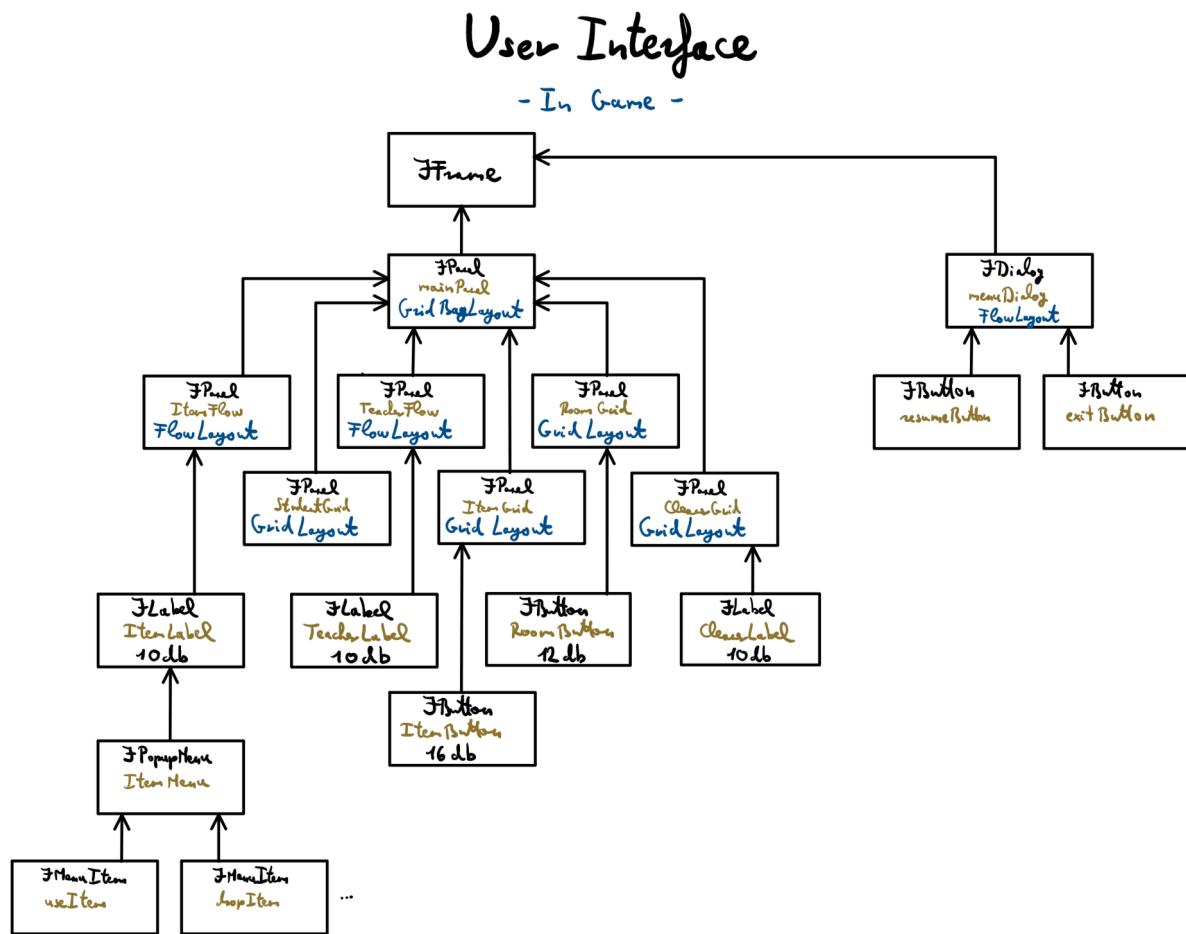
### 11.1 A grafikus interfész



Minden itemhez egyedi grafika készül amik majd ImageIcon-ként kerülnek rá a JLabel-ekre és a JButtonokra.



## 11.2 A grafikus rendszer architektúrája



Containerek feltöltése:

- ItemFlow ← VItemsOfCP
- TeacherFlow ← teachersInRoom
- RoomGrid ← neighbouringRoom
- StudentGrid ← studentsInRoom
- ItemGrid ← itemsInRoom
- CleanerGrid ← cleanersInRoom

A MainWindow attribútumainak az értékei a currentPlayeren keresztül állítódnak be.

A currentPlayert pedig a RefreshView metódusban kéri le a Controllertől.ű

A MainMenu StartGame gombja a MainWindow.startGame()-et hívja (Az pedig ahogyan a szekvenciadiagramon is látszik, a controllernek szól az IController interfacen keresztül)

Az Options beállítása majd a JDialo... lezárása az IController setParameters függvényét hívja, átadva az értékeket

A map beállításához az IController getMapList függvényét használhatjuk ami a default mappából kiolvassa a létrehozott pályák listáját és átadja

A help a mainWindow drawHelp() függvényét hívja

### 11.2.1 A felület működési elve

A View az aktuális játékos szemszögéből megfigyelt környezetet rajzolja le. A játékos jelenlegi szobáját, tárgyait, szereplőit, hatásait. Ezen felül látja, interaktálhat a saját tárgyaival, a szoba tárgyaival és a szomszédos szobákba vezető ajtókkal.

Az adott SWING gombra kattintva a **view** a **controller** a gombnak megfelelő parancsát hívja meg, a MainWindow controller által megvalósított ICommands interface-en keresztül. Ennek hatására a **controller** továbbítja a megfelelő körökre osztott parancsokat a **modell** felé, a parancs függvényében, a Controller osztában tárolt Player interfacen, vagy a Labyrinth osztály IRoom interface-én keresztül. Így a **modell** módosításai végbe mehetnek.

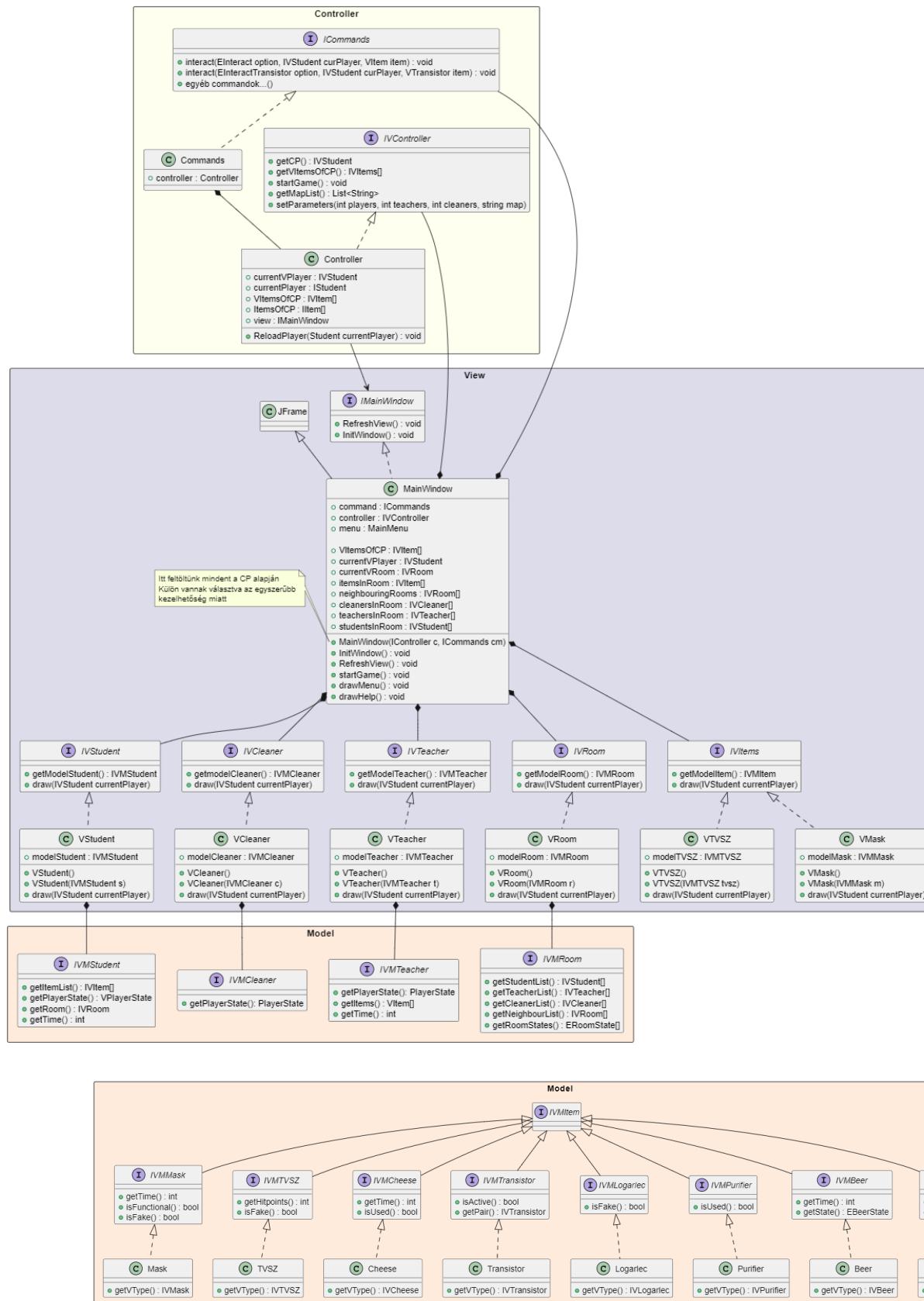
A **modell** módosításainak lefutása után a **controller** értesíti a **view-t**, az IMainWindow interfacen keresztül, hogy a modell potenciálisan megváltozott, és rajzolja újra a megfelelő területeket. Itt minden komponens draw függvényét végig hívja ami egyenként lekérdezi a modelból a változott adatokat (pull alapú). Itt felhasználja a currentPlayert, hiszen az tartalmazza az aktuális IVM osztályok referenciáit (A controller minden körben belerakja az curPlayer IVStudentbe a modelben lévő Student referenciáját IVMStudent interfaceként). A currentPlayert a MainWindow kéri le minden frissítéskor az IVController interfacen keresztül.

Így tudják a MainWindow osztály felületi elemei, az értesítés hatására lekérdezni a felhasználói-felületi elemek **modell**beli tulajdonságait. Az elemek IVM-előtagú interface-ei segítségével.

### 11.2.2 A felület osztály-struktúrája

Megjegyzés: Az IVItems-ból az összes itemnek megfelelő VItem leszármazik, rendre: VTVSZ, VMask, VCheese, VTransistor, VLogarlec, VPurifier, VBeer és Vsponge.

Mindegyik a neki megfelelő model interfacet tárol el. Azért nem jelöltetem az osztálydiagramban mert óriási lenne a mérete



## 11.3 A grafikus objektumok felsorolása

### 11.3.1 Commands

- **Felelősség**

Az osztály felelőssége, hogy a különböző kattintások hatására a modellbe végrehajtsa a megfelelő változást.

- **Ősosztályok**

Nincs

- **Interfészek**

ICommands

- **Attribútumok**

**controller:** Hivatkozás a kontrollerre

- **Metódusok**

**+void interact():** Az itemekre vonatkozó különböző események kezelése a modellben. Pl. item felvétele, eldobás, használat.

### 11.3.2 Controller

- **Felelősség**

Ez az osztály felel a játék logikájáért, hogy a parancsok a modellen a megfelelő hatást váltsák ki. Kapcsolatot teremt a nézet és a modell között.

- **Ősosztályok**

- **Interfészek**

IVController

- **Attribútumok**

- **currentVPlayer:** a jelenlegi játékos nézete
- **currentPlayer:** a jelenlegi játékos modellje
- **VItemsOfCP:** A jelenlegi játékos tárgyainak nézete
- **ItemsOfCP:** A jelenlegi játékos tárgyainak modellje
- **view:** A nézet

- **Metódusok**

- **+void ReloadPlayer(currentPlayer: Student):** A jelenlegi játékos frissítése a paraméterben kapotttra.
- **+IVStudent getCP():** Aktuális játékos lekérdezése
- **+IVItems[] getVItemsOfCP():** Aktuális játékos tárgyai
- **+void startGame():** Játék indítása
- **+List<String> getMapList():** Létrehozott pályák neveinek lekérdezése a default mappából
- **+void setParameters(int, int, string):** Adatok átadása a kontrollernek az options által beállítottakból

### 11.3.3 MainWindow

- **Felelősség**

Ez az osztály felel a játék megjelenítéséért.

- **Ősosztályok**

- **Interfészek**

IMainWindow

- **Attribútumok**

- **command:** a parancsok
- **controller:** a controller
- **menu:** főmenü view
- **VItemsOfCP:** A jelenlegi játékos tárgyainak nézete
- **currentVPlayer:** A jelenlegi játékos nézete
- **currentVRoom:** A jelenlegi szoba nézete
- **itemsInRoom:** A szobában lévő itemek nézetei
- **neighbouringRooms:** A szomszédos szobák nézetei
- **cleanersInRoom:** A szobában lévő cleanerek nézetei
- **teachersInRoom:** A szobában lévő tanárok nézetei
- **studentsInRoom:** A szobában lévő hallgatók nézetei

- **Metódusok**

- **+void RefreshView():** A nézet frissítése
- **+ void startGame():** Ha elindítjuk a játékot a főmenüből akkor ez hívódik meg és inicializálja az attribútumokat
- **+void InitWindow():** Első megjelenítésnél az adatok lekérése és az ablak kirajzolása
- **+void drawMenu():** Főmenü kirajzolása
- **+void drawHelp():** Súgó kirajzolása

### 11.3.4 VTeacher

- **Felelősség**

Ez az osztály felel az oktatók nézetéért

- **Ősosztályok**

JLabel

- **Interfészek**

IVTeacher

- **Attribútumok**

- **modelTeacher:** az oktató modellje

- **Metódusok**

- **+void draw(IVStudent curPlayer):** A nézet kirajzolása
- **+VTeacher(IVMTeacher t):** konstruktur, melyben a nézethez hozzá csatolunk egy olyan interface-t megvalósító osztályt, mellyel a modellbeli adatokat lekérdezhetjük.

### 11.3.5VCleaner

- **Felelősség**

Ez az osztály felel az takarítók nézetéért

- **Ősosztályok**

JLabel

- **Interfészek**

IVCleaner

- **Attribútumok**

- **modelCleaner:** a takarító modellje

- **Metódusok**

- **+void draw(IVStudent curPlayer):** A nézet kirajzolása
- **+VTeacher(IVMCleaner c):** konstruktor, melyben a nézethez hozzá csatolunk egy olyan interface-t megvalósító osztályt, mellyel a modellbeli adatokat lekérdezhetjük.

### 11.3.6VStudent

- **Felelősség**

Ez az osztály felel az hallgatók nézetéért

- **Ősosztályok**

JLabel

- **Interfészek**

IVStudent

- **Attribútumok**

- **modelStudent:** a hallgató modellje

- **Metódusok**

- **+void draw(IVStudent curPlayer):** A nézet kirajzolása
- **+VStudent(IVMStudent s):** konstruktor, melyben a nézethez hozzá csatolunk egy olyan interface-t megvalósító osztályt, mellyel a modellbeli adatokat lekérdezhetjük.

### 11.3.7VRoom

- **Felelősség**

Ez az osztály felel a szobák nézetéért

- **Ősosztályok**

JButton

- **Interfészek**

IVRoom

- **Attribútumok**

- **modelRoom:** a szoba modellje

- **Metódusok**

- **+void draw(IVStudent curPlayer):** A nézet kirajzolása
- **+VRoom(IVMRoom r):** konstruktor, melyben a nézethez hozzá csatolunk egy olyan interface-t megvalósító osztályt, mellyel a modellbeli adatokat lekérdezhetjük.

### 11.3.8 VMask

- **Felelősség**

Ez az osztály felel a maszkok nézetéért

- **Ősosztályok**

JButton

- **Interfészek**

IVItems

- **Attribútumok**

- **modelMask:** a maszk modellje

- **Metódusok**

- **+void draw(IVStudent curPlayer):** A nézet kirajzolása
- **+VMask(IVMMask m):** konstruktor, melyben a nézethez hozzá csatolunk egy olyan interface-t megvalósító osztályt, mellyel a modellbeli adatokat lekérdezhetjük.

### 11.3.9 VTVSZ

- **Felelősség**

Ez az osztály felel a TVSZ nézetéért

- **Ősosztályok**

JButton

- **Interfészek**

IVItems

- **Attribútumok**

- **modelTVSZ:** a TVSZ modellje

- **Metódusok**

- **+void draw(IVStudent curPlayer):** A nézet kirajzolása
- **+VTVSZ(IVMTVSZ t):** konstruktor, melyben a nézethez hozzá csatolunk egy olyan interface-t megvalósító osztályt, mellyel a modellbeli adatokat lekérdezhetjük.

### 11.3.10 VCheese

- **Felelősség**

Ez az osztály felel a sajtok nézetéért

- **Ősosztályok**

JButton

- **Interfészek**

IVItems

- **Attribútumok**

- **modelCheese:** a sajt modellje

- **Metódusok**

- **+void draw(IVStudent curPlayer):** A nézet kirajzolása
- **+VCheese(IVMCheese c):** konstruktor, melyben a nézethez hozzá csatolunk egy olyan interface-t megvalósító osztályt, mellyel a modellbeli adatokat lekérdezhetjük.

### 11.3.11 VLogarlec

- **Felelősség**

Ez az osztály felel a logarlécek nézetéért

- **Ősosztályok**

JButton

- **Interfészek**

IVItems

- **Attribútumok**

- **modelLogalec:** a logarléc modellje

- **Metódusok**

- **+void draw(IVStudent curPlayer):** A nézet kirajzolása
- **+VLogarlec(IVMLogarlec l):** konstruktor, melyben a nézethez hozzá csatolunk egy olyan interface-t megvalósító osztályt, mellyel a modellbeli adatokat lekérdezhetjük.

### 11.3.12 VPurifier

- **Felelősség**

Ez az osztály felel a légtisztítók nézetéért

- **Ősosztályok**

JButton

- **Interfészek**

IVItems

- **Attribútumok**

- **modelPurifier**: a légtisztító modellje

- **Metódusok**

- **+void draw(IVStudent curPlayer)**: A nézet kirajzolása
- **+VPurifier(IVMPurifier p)**: konstruktor, melyben a nézethez hozzá csatolunk egy olyan interface-t megvalósító osztályt, mellyel a modellbeli adatokat lekérdezhetjük.

### 11.3.13 VSponge

- **Felelősség**

Ez az osztály felel a szivacsok nézetéért

- **Ősosztályok**

JButton

- **Interfészek**

IVItems

- **Attribútumok**

- **modelSponge**: a szivacs modellje

- **Metódusok**

- **+void draw(IVStudent curPlayer)**: A nézet kirajzolása
- **+VSponge(IVMSponge s)**: konstruktor, melyben a nézethez hozzá csatolunk egy olyan interface-t megvalósító osztályt, mellyel a modellbeli adatokat lekérdezhetjük.

### 11.3.14 VTransistor

- **Felelősség**

Ez az osztály felel a tranzisztor nézetéért

- **Ősosztályok**

JButton

- **Interfészek**

IVItems

- **Attribútumok**

- **modelTransistor**: a tranzisztor modellje

- **Metódusok**

- **+void draw(IVStudent curPlayer):** A nézet kirajzolása
- **+VTransistor(IVMTransistor t):** konstruktor, melyben a nézethez hozzá csatolunk egy olyan interface-t megvalósító osztályt, mellyel a modellbeli adatokat lekérdezhetjük.

## Modell bővülés:

### 11.3.15 Mask

- **Felelősség**

Az osztály a modellbeli Mask működéséért felelős

- **Ősosztályok**

- **Interfészek**

IVMMask

- **Attribútumok**

- **Metódusok**

- **+IVMask getVType():** Az osztály nézetbeli típusát adja vissza.
- **+int getTime():** Mask maradék idejének lekérdezése
- **+bool isFunctional():** Működik-e még a Mask
- **+bool isFake():** Hamis-e a Mask

### 11.3.16 TVSZ

- **Felelősség**

Az osztály a modellbeli TVSZ működéséért felelős

- **Ősosztályok**

- **Interfész**

IVMTVSZ

- **Attribútumok**

- **Metódusok**

- **+IVTVSZ getVType():** Az osztály nézetbeli típusát adja vissza.
- **+int getHitpoints():** TVSZ életpontjainak lekérdezése
- **+bool isFake():** Hamis-e a TVSZ

### 11.3.17 Cheese

- **Felelősség**

Az osztály a modellbeli Cheese működéséért felelős

- **Ősosztályok**

- **Interfészek**

IVMCheese

- **Attribútumok**

- **Metódusok**

- **+IVCheese getVType()**: Az osztály nézetbeli típusát adja vissza.
- **+int getTime()**: Mask maradék idejének lekérdezése
- **+bool isUsed()**: Elhasználták-e már a sajtot

### 11.3.18 Transistor

- **Felelősség**

Az osztály a modellbeli Transistor működéséért felelős

- **Ősosztályok**

- **Interfész**

IVMTransistor

- **Attribútumok**

- **Metódusok**

- **+IVTransistor getVType()**: Az osztály nézetbeli típusát adja vissza.
- **+IVTransistor getPair()**: A tranzisztor párnak lekérdezése.
- **+bool isActive()**: Aktiválták-e már a tranzisztort

### 11.3.19 Logarléc

- **Felelősség**

Az osztály a modellbeli Logarléc működéséért felelős

- **Ősosztályok**

- **Interfész**

IVMLogarlec

- **Attribútumok**

- **Metódusok**

- **+IVLogarlec getVType()**: Az osztály nézetbeli típusát adja vissza.
- **+bool isFake()**: Hamis-e a Logarléc

### 11.3.20 Purifier

- **Felelősség**

Az osztály a modellbeli Purifier működéséért felelős

- **Ősosztályok**

- **Interfészek**

IVMPurifier

- **Attribútumok**

- **Metódusok**

- **+IVPurifier getVType()**: Az osztály nézetbeli típusát adja vissza.
- **+bool isUsed()**: Elhasználták-e a purifert.

### 11.3.21 Beer

- **Felelősség**

Az osztály a modellbeli Beer működéséért felelős

- **Ősosztályok**

- **Interfészek**

IVMBeer

- **Attribútumok**

- **Metódusok**

- **+IVBeer getVType()**: Az osztály nézetbeli típusát adja vissza.
- **+int getTime()**: Beer maradék idejének lekérdezése
- **+EBeerState getState()**: A sör állapotát kérdezi le.

### 11.3.22 Sponge

- **Felelősség**

Az osztály a modellbeli Sponge működéséért felelős

- **Ősosztályok**

- **Interfészek**

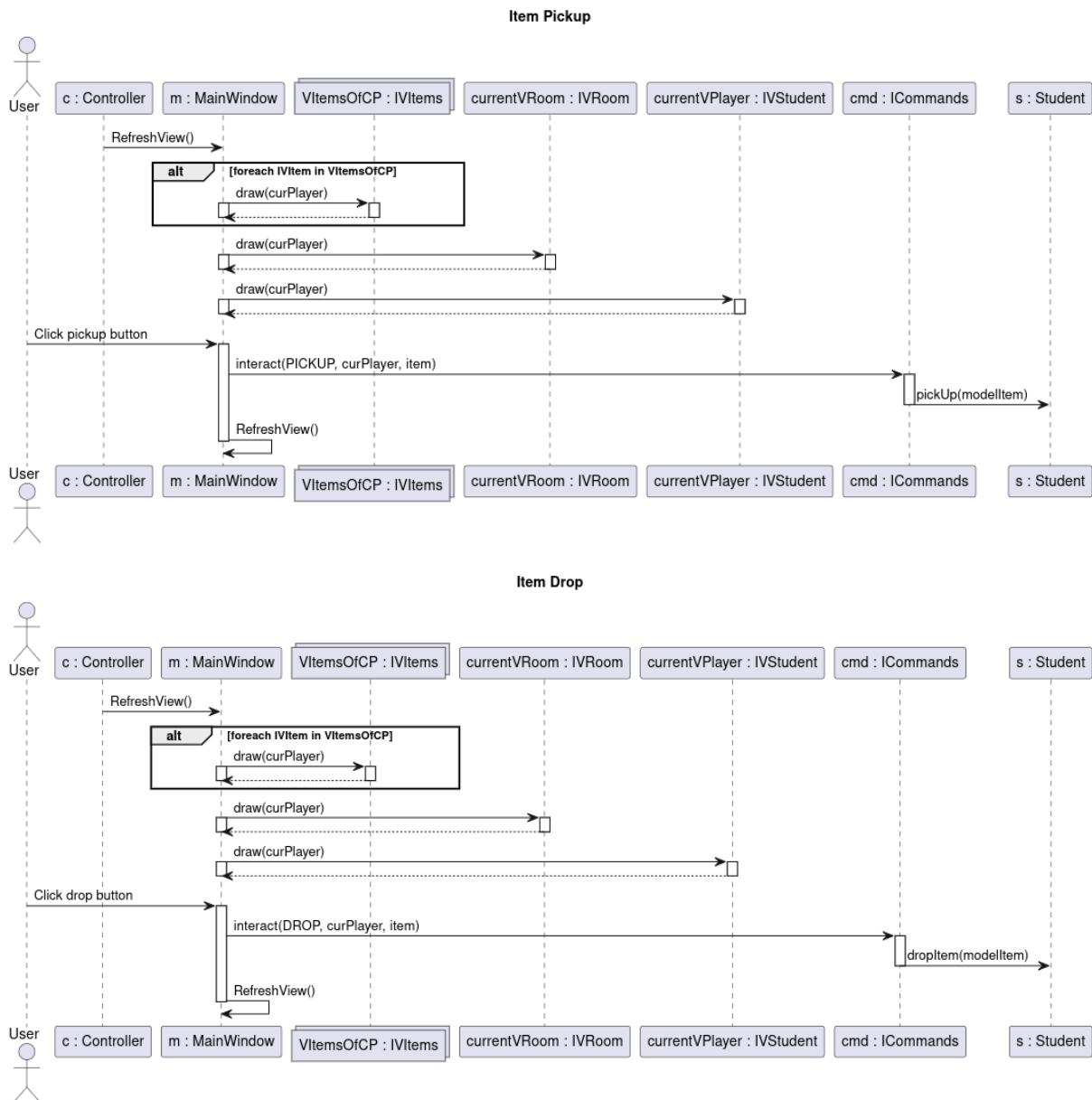
IVMSponge

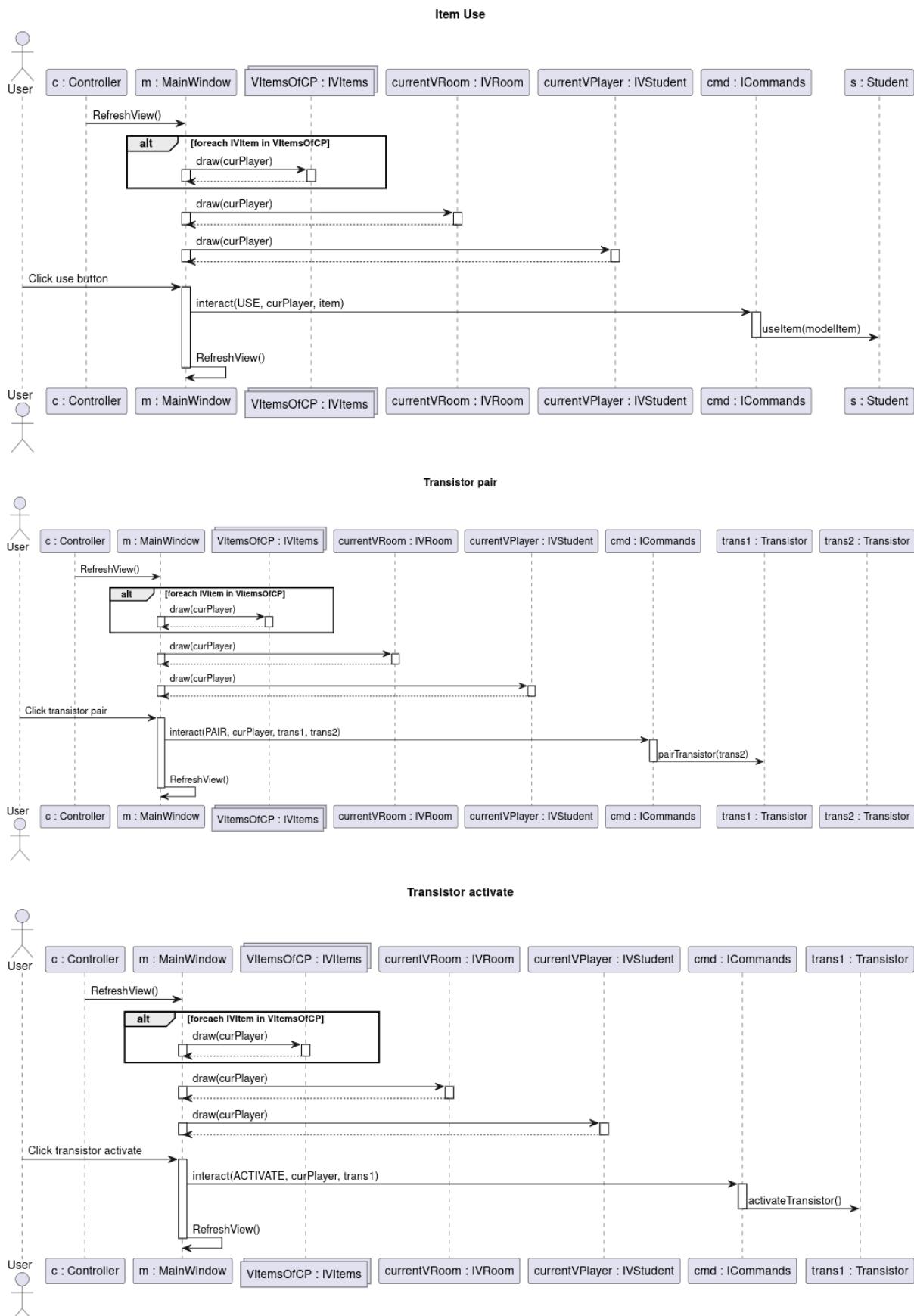
- **Attribútumok**

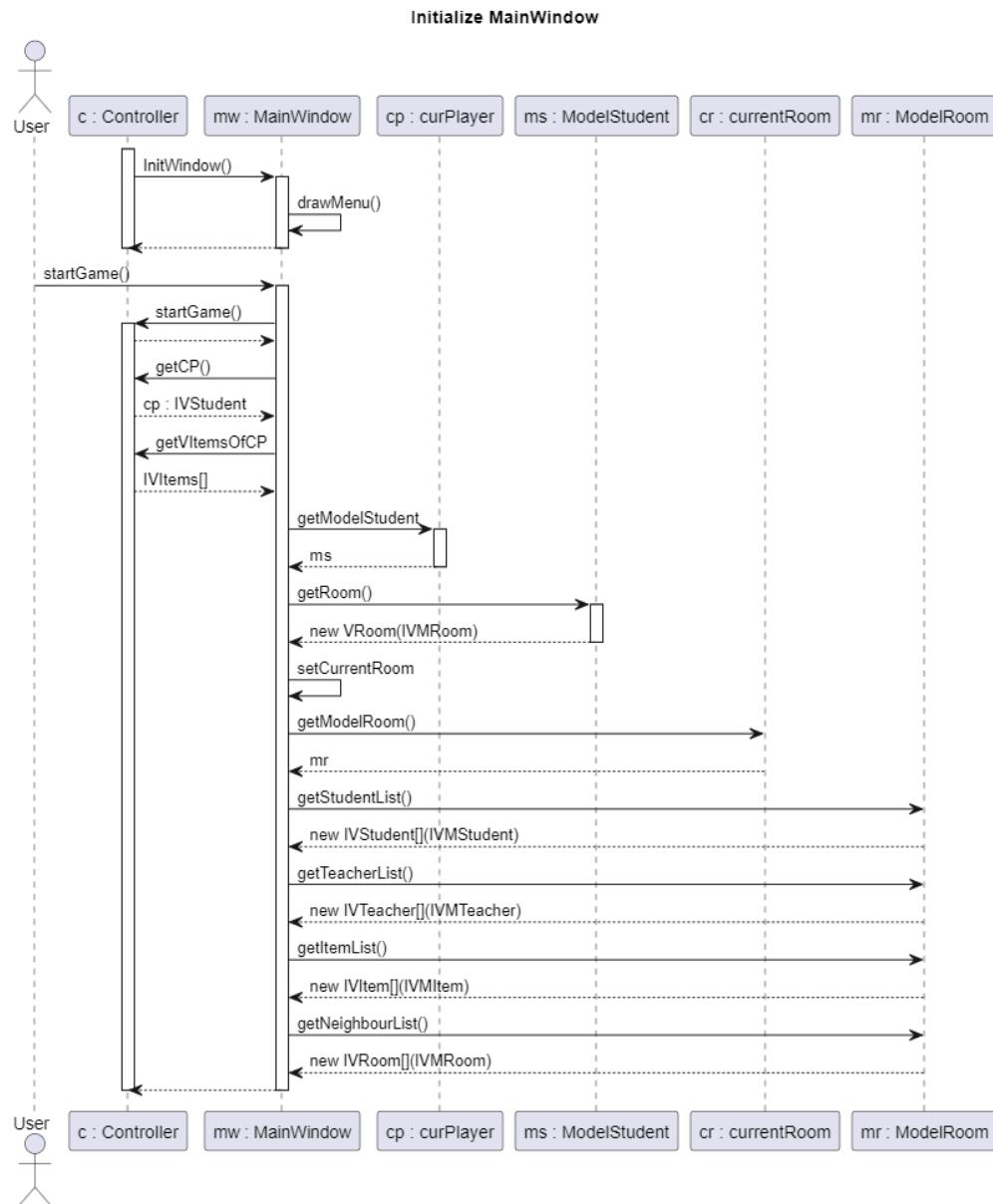
- **Metódusok**

- **+IVSponge getVType()**: Az osztály nézetbeli típusát adja vissza.
- **+int getTime()**: Sponge maradék idejének lekérdezése
- **+bool isFunctional()**: Működik-e még a szivacs.

## 11.4 Kapcsolat az alkalmazói rendszerrel







## 13. Grafikus változat beadása

### 13.1 Fordítási és futtatási útmutató

#### 13.1.1 Fájllista

Fájl neve	Méret(byte)	Keletkezés ideje	Tartalom
Main.java	371	May 20 15:24	
./controller:			
Commands.java	3794	May 21 15:47	
Controller.java	9212	May 21 15:47	
ICommands.java	480	May 21 15:47	
IController.java	530	May 21 15:47	
Labyrinth.java	5648	May 21 16:42	
TimedObject.java	1108	May 3 20:47	
Timer.java	1300	May 13 20:30	
./controller/tests:			
DropItem.java	10586	Apr 25 22:20	
ItemFunctTests.java	11499	May 3 20:47	
ItemPickUp.java	3945	May 3 20:47	
ITestcase.java	122	Apr 25 22:20	
Room_functionality.java	4811	May 3 20:47	
Room_Interactions.java	19388	Apr 25 22:20	
Skeleton.java	6275	Apr 25 22:20	
UseItem.java	4630	May 3 20:47	
./model:			
ERoomEffects.java	114	May 3 20:47	
ICRoom.java	285	May 3 20:47	
ITimedEntity.java	83	May 3 20:47	
ITimer.java	192	May 13 20:30	
IVMRoom.java	435	May 21 15:47	
Room.java	15199	May 21 15:47	
./model/items:			
Beer.java	3085	May 21 15:47	
Cheese.java	2929	May 21 15:47	
EBeerState.java	454	May 21 15:47	
IlItem.java	2571	May 20 15:03	
IVMBeer.java	119	May 21 15:47	

Fájl neve	Méret(byte)	Keletkezés ideje	Tartalom
IVMCheese.java	117	May 21 15:47	
IVMItems.java	62	May 21 15:47	
IVMLogarlec.java	98	May 21 15:47	
IVMMask.java	144	May 21 15:47	
IVMPurifier.java	98	May 21 15:47	
IVMSponge.java	123	May 21 15:47	
IVMTransistor.java	153	May 21 15:47	
IVMTVSZ.java	127	May 21 15:47	
Logarlec.java	2577	May 21 15:47	
Mask.java	3793	May 21 15:47	
Purifier.java	2047	May 20 15:03	
Sponge.java	2910	May 20 15:03	
Transistor.java	3803	May 21 16:40	
TVSZ.java	2888	May 21 15:47	
./model/player:			
Cleaner.java	1085	May 21 15:47	
EPlayerState.java	96	Mar 21 23:52	
IVMCleaner.java	102	May 20 15:03	
IVMStudent.java	282	May 20 15:24	
IVMTeacher.java	227	May 20 15:24	
Player.java	2721	May 21 16:31	
Student.java	2491	May 21 15:47	
Teacher.java	1638	May 21 15:47	
./test:			
CommandData.java	136	Apr 28 23:32	
ECommand.java	142	Apr 28 23:32	
FileHandling.java	5966	May 3 20:47	
IPrintStat.java	126	Apr 28 23:32	
TestRunner.java	12677	May 20 15:24	
./test/test_batch:			
runAllTest.bat	501	Apr 28 23:32	
test1.bat	591	Apr 29 00:21	
test10.bat	581	Apr 29 00:23	
test11.bat	581	Apr 29 00:23	
test12.bat	581	Apr 29 00:23	
test13.bat	581	Apr 29 00:23	

Fájl neve	Méret(byte)	Keletkezés ideje	Tartalom
test14.bat	581	Apr 29 00:23	
test15.bat	581	Apr 29 00:23	
test16.bat	581	Apr 29 00:23	
test17.bat	581	Apr 29 00:23	
test18.bat	581	Apr 29 00:23	
test19.bat	581	Apr 29 00:23	
test2.bat	578	Apr 29 00:23	
test20.bat	581	Apr 29 00:23	
test21.bat	583	Apr 29 00:23	
test22.bat	583	Apr 29 00:23	
test23.bat	583	Apr 29 00:23	
test3.bat	578	Apr 29 00:23	
test4.bat	578	Apr 29 00:23	
test5.bat	578	Apr 29 00:23	
test6.bat	577	Apr 29 00:23	
test7.bat	577	Apr 29 00:23	
test8.bat	577	Apr 29 00:23	
test9.bat	577	Apr 29 00:23	
./test/test_txt/test_expected:			
test1_exp.txt	2046	May 3 20:47	
test10_exp.txt	255	May 3 20:47	
test11_exp.txt	610	May 3 20:47	
test12_exp.txt	610	May 3 20:47	
test13_exp.txt	622	May 3 20:47	
test14_exp.txt	549	May 3 20:47	
test15_exp.txt	549	May 3 20:47	
test16_exp.txt	549	May 3 20:47	
test17_exp.txt	1032	May 3 20:47	
test18_exp.txt	255	May 3 20:47	
test19_exp.txt	422	May 3 20:47	
test2_exp.txt	749	May 3 20:47	
test20_exp.txt	565	May 3 20:47	
test21_exp.txt	625	May 3 20:47	
test22_exp.txt	658	May 3 20:47	
test23_exp.txt	3642	May 3 20:47	
test3_exp.txt	677	May 3 20:47	

Fájl neve	Méret(byte)	Keletkezés ideje	Tartalom
test4_exp.txt	645	May 3 20:47	
test5_exp.txt	793	May 3 20:47	
test6_exp.txt	342	May 3 20:47	
test7_exp.txt	642	May 3 20:47	
test8_exp.txt	602	May 3 20:47	
test9_exp.txt	255	May 3 20:47	
./test/test_txt/test_input:			
test1.txt	502	May 3 20:47	
test10.txt	97	Apr 28 23:32	
test11.txt	152	Apr 28 23:32	
test12.txt	152	Apr 28 23:32	
test13.txt	149	Apr 28 23:32	
test14.txt	151	Apr 28 23:32	
test15.txt	175	Apr 28 23:32	
test16.txt	173	Apr 28 23:32	
test17.txt	224	Apr 28 23:32	
test18.txt	152	Apr 28 23:32	
test19.txt	151	Apr 28 23:32	
test2.txt	256	Apr 28 23:32	
test20.txt	145	Apr 28 23:32	
test21.txt	154	Apr 28 23:32	
test22.txt	116	Apr 28 23:32	
test23.txt	1989	May 3 20:47	
test3.txt	208	Apr 28 23:32	
test4.txt	185	Apr 28 23:32	
test5.txt	192	May 3 20:47	
test6.txt	94	Apr 28 23:32	
test7.txt	161	Apr 28 23:32	
test8.txt	177	Apr 28 23:32	
test9.txt	78	Apr 28 23:32	
./view:			
BackgroundPanel.java	535	May 21 15:47	
IMainWindow.java	172	May 21 15:47	
IVCleaner.java	279	May 21 15:47	
IVItems.java	297	May 21 15:47	
IVRoom.java	526	May 20 15:24	

Fájl neve	Méret(byte)	Keletkezés ideje	Tartalom
IVStudent.java	291	May 21 15:47	
IVTeacher.java	277	May 21 15:47	
MainWindow.java	20684	May 21 15:47	
VBeer.java	2514	May 21 15:47	
VCheese.java	2371	May 21 15:47	
VCleaner.java	607	May 21 15:47	
VLogarlec.java	1191	May 21 15:47	
VMask.java	2484	May 21 15:47	
VPurifier.java	2409	May 21 15:47	
VRoom.java	826	May 20 18:54	
VSponge.java	2470	May 21 15:47	
VStudent.java	736	May 21 15:47	
VTeacher.java	647	May 21 15:47	
VTransistor.java	3674	May 21 15:47	
VTVSZ.java	2162	May 21 15:47	

### 13.1.2 Fordítás és telepítés

A Program fordításához a Visual Studio Code-ot használjuk, az “Extension Pack for Java” kiegészítővel. VSCode-on belül fordításhoz és futtatáshoz is ezt használjuk. A programot a bővítmény által nyújtott “Run Java” gombbal indítjuk el.

Openjdk 11 szükséges a fordításhoz.

A program indulási pontja a Main.java main függvénye.

### 13.1.3 Futtatás

A programot Visual Studio Code-ból tudjuk elindítani a fent megadott kiegészítő segítségével. Ezen kívül a projekthez generálva van egy futtatható .jar kiterjesztésű file amivel fordítás nélkül is indítható a játék

## 14. Összefoglalás

### 14.1 A projektre fordított összes munkaidő

Tag neve	Munkaidő (óra)
Mészáros Barna	130
Rékó Márton	86
Csiszár Alex Gergő	48
Csirmaz Gáspár	79
Hegedüs Péter	0
<b>Összesen</b>	<b>343</b>

- A feltöltött programok forrássorainak száma**

Fázis	Kódsorok száma
Szkeleton	~1500
Prototípus	~2000
Grafikus változat	~1200
<b>Összesen</b>	<b>~4700</b>

### 14.2 • Projekt összegzés

#### 14.2.1 Mit tanultak a projektből konkrétan és általában?

-Kód írása nélkül nem tudunk pontosan tervezni, azt tapasztaltuk, hogy mindenkor jöttek elő a problémák a tervezében. Szóval a tervezések inkább csak egy irányvonalat adtak, és végül is nem tükrözik teljesen a programkódot, miközben sok időt töltöttünk különböző diagramok rajzolásával.

-Nagyon nehéz rendesen felosztani a feladatot, és ha sikerül is felosztani, később problémák akadnak a részek összeállításánál.

-Jó tapasztalatot adott a projekt karbantartható szoftver készítésének a nehézségeiről.

#### 14.2.2 Mi volt a legnehezebb és a legkönnyebb?

-Nehéz volt kitalálni egy jó osztálydiagramot az első fázisokban, és kód nélkül különböző diagramokat készíteni.

-A kitalált modelljeink alapján nehéz volt működő kódot írni.

-A szkeleton készítése viszonylag könnyű volt

-Nehéz volt egymás között felosztani a feladatot

#### 14.2.3 Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?

Nem, pl. a grafikus változat elkészítése sokkal több időt vett igénybe, mint a tervezések elkészítése.

#### **14.2.4 Ha nem, akkor hol okozott ez nehézséget?**

A grafikus változat elkészítése sok időt vett igénybe, és itt sok hiba előjött a kóddal.

#### **14.2.5 Milyen változtatási javaslatuk van?**

Én úgy érzem nekünk problémát okozott, hogy sokáig nem kellett semmilyen kódot írnunk. Ennek az lett az eredménye, hogy létrehoztunk egy csomó diagramot, amiket végül nem tudtunk felhasználni, mert kiderült, hogy így nem tudjuk megvalósítani a programot. A napló nem volt túl hasznos, mivel inkább a feladaton dolgoztunk minthogy azt töltögessük. Előre pedig csak nagy ívekben volt tervezés, akkor dolgoztunk a feladaton, amikor volt rá időnk.

#### **14.2.6 Milyen feladatot ajánlanának a projektre?**

Minecraft remake

#### **14.2.7 Egyéb kritika és javaslat**

A dokumentumok félrevezetők voltak. A laborvezető nélkül alig lehetett pontosan megérteni mit vár el a feladat.

A feladatok a hetek előrehaladtával nőttek, azaz amikor a többi tárgyból is jöttek a ZH-k és határidők akkor kellett többet foglalkozni a projekttel is. Emellett az előkészület nagyon túlspecifikálta az amúgy nem túl bonyolult feladatot, mely olyan UML tudást várt el, amit az új rendszerben nem tanítottak ennyire mélyen.

Ezekben felül a tárgy nagyon nem áll összhangban a kredit értékével még így sem.

Konklúzió: Az elején rengeteg felesleges dolgot kellett újra és újra diagramra venni, majd a végén túl kevés idő alatt várták el, hogy összerakjuk az egészet.