

Szoftver projekt laboratórium

összesített dokumentáció

pickle-plus csapat

Konzulens:
Dr. Simon Balázs

Csapattagok

Görömbey László	I3B5JU	laszlo.gorombey@gmail.com
Héjja Márton	RECW35	mhejja@edu.bme.hu
Sági Péter Pál	KQF28D	sagipali02@gmail.com
Tömöri Péter András	I4RZ0O	tomoripeti2003@gmail.com
Vizhányó Miklós Ferenc	NVY1AG	vmiklos2468@gmail.com

2024.05.22.

Tartalomjegyzék

1.	Követelmény, projekt, funkcionalitás	10
1.1	Bevezetés	10
1.1.1	Cél	10
1.1.2	Szakterület	10
1.1.3	Definíciók, rövidítések	10
1.1.4	Hivatkozások	10
1.1.5	Összefoglalás	10
1.2	Áttekintés	11
1.2.1	Általános áttekintés	11
1.2.2	Funkciók	11
1.2.3	Felhasználók	12
1.2.4	Korlátozások	12
1.2.5	Feltételezések, kapcsolatok	12
1.3	Követelmények	13
1.3.1	Funkcionális követelmények	13
1.3.2	Erőforrásokkal kapcsolatos követelmények	17
1.3.3	Átadással kapcsolatos követelmények	18
1.3.4	Egyéb nem funkcionális követelmények	18
1.4	Lényeges use-case-ek	19
1.4.1	Use-case leírások	19
1.4.2	Use-case diagram	21
1.5	Szótár	21
1.6	Projekt terv	23
1.7	Napló	24
2.	Analízis modell kidolgozása	26
2.1	Objektum katalógus	26
2.1.1	Ajtó	26
2.1.2	Camembert	26
2.1.3	FFP2-es maszk	26
2.1.4	Hallgató	26
2.1.5	Logarléc	26
2.1.6	Oktató	26
2.1.7	Söröspohár	26
2.1.8	Szoba	26
2.1.9	Táblatörlő rongy	26
2.1.10	Tranzisztor	26

2.1.11	TVSZ.....	27
2.2	Statikus struktúra diagramok	27
2.3	Osztályok leírása.....	28
2.3.1	BeerGlass	28
2.3.2	Camembert	28
2.3.3	DangerType	29
2.3.4	IntervalItem	29
2.3.5	<i>Item</i>	29
2.3.6	ItemHandler.....	30
2.3.7	Mask	30
2.3.8	<i>Person</i>	31
2.3.9	Rag	32
2.3.10	Room	33
2.3.11	SlideRule	34
2.3.12	Student.....	35
2.3.13	Teacher	36
2.3.14	TimeSensitive.....	36
2.3.15	Transistor.....	36
2.3.16	TVSZ.....	37
2.4	Szekvencia diagramok	38
2.5	State-chartok.....	50
2.6	Napló	51
3.	Analízis modell refaktorálása.....	52
3.1	Objektum katalógus	52
3.1.1	Ajtó.....	52
3.1.2	Camembert	52
3.1.3	FFP2-es maszk	52
3.1.4	Hallgató	52
3.1.5	Logarléc.....	52
3.1.6	Oktató	52
3.1.7	Söröspohár.....	52
3.1.8	Szoba	52
3.1.9	Táblatörlő rongy	52
3.1.10	Tranzisztor.....	52
3.1.11	TVSZ.....	53
3.2	Statikus struktúra diagramok	53
3.3	Osztályok leírása.....	54

3.3.1	BeerGlass	54
3.3.2	Camembert	55
3.3.3	IntervalItem	55
3.3.4	<i>Item</i>	56
3.3.5	ItemHandler.....	56
3.3.6	Mask.....	57
3.3.7	<i>Person</i>	57
3.3.8	Rag	58
3.3.9	Room	59
3.3.10	SlideRule	61
3.3.11	Student.....	61
3.3.12	Teacher	62
3.3.13	TimeSensitive.....	63
3.3.14	Transistor.....	63
3.3.15	TVSZ.....	64
3.4	Szekvencia diagramok	65
3.5	State-chartok	77
4.	Szkeleton tervezése	80
4.1	A szkeleton modell valóságos use-case-ei.....	80
4.1.1	Use-case diagram	80
4.1.2	Use-case leírások.....	80
4.2	A szkeleton kezelői felületének terve, dialógusok	84
4.3	Szekvencia diagramok a belső működésre	86
4.3.1	Szobába átlépés	86
4.3.2	Oktatók találkozása	87
4.3.3	Hallgatók találkozása	87
4.3.4	Oktató találkozik egy hallgatóval.....	87
4.3.5	Tárgyfelvétel	88
4.3.6	Tárgylerakás	88
4.3.7	Camembert és maszk használata	89
4.3.8	Hallgató elkából	89
4.3.9	Maszk időtelés.....	90
4.3.10	Táblatörölő rongy időtelés	90
4.3.11	Söröspohár időtelés	91
4.3.12	TVSZ használata	92
4.3.13	Tranzisztor használata teleportálásra	93
4.3.14	Szobák egyesítése.....	94

4.3.15	Szoba kettéválása	95
4.3.16	Hallgató felveszi a Logarlécet.....	96
4.3.17	Oktató felveszi a Logarlécet.....	96
4.4	Kommunikációs diagramok.....	97
4.4.1	Szobába átlépés	97
4.4.2	Két oktató találkozik	98
4.4.3	Két hallgató találkozik	98
4.4.4	Oktató találkozik hallgatóval	99
4.4.5	Tárgyfelvétel	99
4.4.6	Tárgylerakás	100
4.4.7	Szoba elgázosítása.....	100
4.4.8	Hallgató elkából	101
4.4.9	Maszk időtelés.....	101
4.4.10	Táblatörlő rongy időtelés	102
4.4.11	Söröspohár időtelés	102
4.4.12	TVSZ használata	103
4.4.13	Tranzisztor használata teleportálásra	103
4.4.14	Szobák egyesítése.....	104
4.4.15	Szoba kettéválasztása	104
4.4.16	Hallgató felveszi a Logarlécet.....	105
4.4.17	Oktató felveszi a Logarlécet.....	105
4.5	Napló	106
5.	Szkeleton beadás	107
5.1	Szkeleton tervezési módosítások	107
5.1.1	Módosított lefutási szekvenciák	107
1 - Szobába átlépés.....	107	
14 - Szobák egyesítése	111	
15 - Szoba kettéválása.....	112	
5.1.2	Inicializáló szekvenciák	114
5.2	Fordítási és futtatási útmutató.....	126
5.2.1	Fájllista	126
5.2.2	Fordítás	127
5.2.3	Futtatás	127
5.3	Értékelés	127
5.4	Napló	128
6.	Prototípus koncepciója.....	129
6.0	Változás hatása a modellre	129

6.0.1	Módosult osztálydiagram	129
6.0.2	Új vagy megváltozó metódusok	130
6.0.3	Szekvencia-diagramok	132
6.1	Prototípus interface-definíciója	145
6.1.1	Az interfész általános leírása	145
6.1.2	Bemeneti nyelv	145
6.1.3	Kimeneti nyelv	151
6.2	Összes részletes use-case	156
6.3	Tesztelési terv	160
6.4	Tesztelést támogató segéd- és fordítóprogramok specifikálása	165
6.5	Napló	165
7.	Részletes tervezet	167
7.0	Prototípus-interface újításai	167
7.0.1	Bemeneti nyelv újítása	167
7.0.2	Kimeneti nyelv újítása	167
7.1	Osztályok és metódusok tervezet	168
7.1.1	AirFresher	168
7.1.2	BeerGlass	168
7.1.3	Camembert	170
7.1.4	Cleaner	171
7.1.5	FalseMask	171
7.1.6	FalseSlideRule	172
7.1.7	FalseTVSZ	172
7.1.8	IntervalItem	172
7.1.9	<i>Item</i>	173
7.1.10	ItemHandler	174
7.1.11	Mask	174
7.1.12	<i>Person</i>	176
7.1.13	Rag	178
7.1.14	Room	180
7.1.15	SlideRule	187
7.1.16	Student	188
7.1.17	Teacher	189
7.1.18	TimeSensitive	190
7.1.19	Transistor	190
7.1.20	TVSZ	192
7.1.21	Interpreter	194

7.2	A tesztek részletes tervei, leírásuk a teszt nyelvén	195
7.2.1	Átlépés oktatóhoz nem védő tárgyakkal 1	195
7.2.2	Átlépés oktatóhoz nem védő tárgyakkal 2	195
7.2.3	Átlépés oktatóhoz TVSZ-szel	196
7.2.4	Átlépés oktatóhoz BeerGlass-szal	197
7.2.5	Átlépés gázba nem védő tárgyakkal 1	197
7.2.6	Átlépés gázba nem védő tárgyakkal 2	198
7.2.7	Átlépés gázba Mask-kal	198
7.2.8	Takarító átléptetése	199
7.2.9	Oktató találkozása személyekkel és tárgyakkal	200
7.2.10	Tárgy felvétele és lerakása	201
7.2.11	Tárgyak időtelése	202
7.2.12	Szobák módosítása	204
7.2.13	Gázhoz kapcsolódó tárgyak aktiválása.....	205
7.2.14	Transistor használata	206
7.2.15	Bénított hallgató	207
7.2.16	Elátkozott szoba	207
7.2.17	Teli szoba	208
7.2.18	Tanár és logarléc	209
7.2.19	Cleaner logarléc.....	209
7.2.20	Cleaner átlép cursed szobába	209
7.2.21	Sikertelen tranzisztor párosítás.....	210
7.2.22	Ragacsos szoba.....	211
7.2.23	Sikertelen kettéválás és egyesülés.....	211
7.2.24	Takarító általi mozgatás	212
7.2.25	Tárgyak eltűnése és aktivált rongy földön viselkedése	213
7.2.26	Ragacsosság és sörösüveg	215
7.2.27	Hallgató megvédi magát és nyer	216
7.2.28	Játék állapotai és hamis Logarléc	218
7.3	A tesztelést támogató programok tervei	219
7.4	Napló	220
8.	Prototípus beadása	222
8.1	Fordítási és futtatási útmutató.....	222
8.1.1	Fájllista	222
8.1.2	Fordítás	224
8.1.3	Futtatás	224
8.2	Tesztek jegyzőkönyvei	225

8.2.1	Átlépés oktatóhoz nem védő tárgyakkal 1	225
8.2.2	Átlépés oktatóhoz nem védő tárgyakkal 2	225
8.2.3	Átlépés oktatóhoz TVSZ-szel	225
8.2.4	Átlépés oktatóhoz BeerGlass-szal	225
8.2.5	Átlépés gázba nem védő tárgyakkal 1	225
8.2.6	Átlépés gázba nem védő tárgyakkal 2	225
8.2.7	Átlépés gázba Mask-kal	225
8.2.8	Takarító átléptetése	225
8.2.9	Oktató találkozása személyekkel és tárgyakkal	226
8.2.10	Tárgy felvétele és lerakása	226
8.2.11	Tárgyak időtelése	226
8.2.12	Szobák módosítása	226
8.2.13	Gázhoz kapcsolódó tárgyak aktiválása.....	226
8.2.14	Transistor használata	226
8.2.15	Bénított hallgató	226
8.2.16	Elátkozott szoba	226
8.2.17	Teli szoba	226
8.2.18	Tanár és logarléc	226
8.2.19	Cleaner és logarléc	227
8.2.20	Cleaner átlép cursed szobába	227
8.2.21	Sikertelen tranzisztor párosítás.....	227
8.2.22	Ragacsos szoba.....	227
8.2.23	Sikertelen kettéválás és egyesülés	227
8.2.24	Takarító általi mozgatás	227
8.2.25	Tárgyak eltűnése és aktivált rongy földön viselkedése	227
8.2.26	Ragacsosság és sörösüveg	228
8.2.27	Hallgató megvédi magát és nyer	228
8.2.28	Játék állapotai és hamis Logarléc	228
8.3	Értékelés	228
8.4	Napló	228
9.	Grafikus felület specifikációja	230
9.1	A grafikus interfész	230
9.1.1	Menü nézete	230
9.1.2	Játék nézete	230
9.1.3	Leírás.....	231
9.2	A grafikus rendszer architektúrája.....	231
9.2.1	A felület működési elve.....	231

9.2.2	A felület osztály-struktúrája	232
9.3	A grafikus objektumok felsorolása.....	232
9.3.1	Controller	232
9.3.2	DoorPanel.....	234
9.3.3	GameWindow.....	234
9.3.4	ItemPanel.....	235
9.3.5	MenuWindow.....	235
9.3.6	PersonPanel	236
9.3.7	PlayerPanel.....	236
9.4	Kapcsolat az alkalmazói rendszerrel	237
9.4.1	Controller	237
9.4.2	GameWindow.....	246
9.4.3	ItemPanel.....	247
9.4.4	PersonPanel	248
9.4.5	MenuWindow.....	248
9.4.6	PlayerPanel.....	249
9.4.7	DoorPanel.....	255
9.5	Napló	256
10.	Grafikus változat beadása.....	257
10.1	Fordítási és futtatási útmutató	257
10.1.1	Fájllista	257
10.1.2	Fordítás és telepítés	258
10.1.3	Futtatás	259
10.2	Értékelés.....	259
10.3	Napló.....	260
11.	Összefoglalás.....	261
11.1	A projektre fordított összes munkaidő	261
11.2	Projekt összegzés	261
11.2.1	Mit tanultak a projektból konkrétan és általában?	261
11.2.2	Mi volt a legnehezebb és a legkönnyebb?.....	261
11.2.3	Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?	261
11.2.4	Ha nem, akkor hol okozott ez nehézséget?	261
11.2.5	Milyen változtatási javaslatuk van?	262
11.2.6	Milyen feladatot ajánlanak a projektre?	262
11.2.7	Egyéb kritika és javaslat.....	262

1. Követelmény, projekt, funkcionalitás

1.1 Bevezetés

1.1.1 Cél

A dokumentum célja, hogy a félév során végrehajtandó projekt kereteit és alapvető céljait definiálja, valamint a megrendelő elvárásait leírja.

1.1.2 Szakterület

A projekt célja egy játékprogram létrehozása, amelyet elsősorban rekreációs célokra lehet használni.

1.1.3 Definíciók, rövidítések

- JVM: A Java programozási nyelvben írt, és bájtkóddá fordított programokat futtatja platformfüggetlen környezetben.
- JRE: Java Runtime Environment: Az a programcsomag, ami tartalmazza a JVM-et és a Java programok futtatásához szükséges standard könyvtárakat.
- JDK: Java Development Kit: A Java programozási nyelv kritikus fejlesztői komponense, egy olyan szoftvercsomag, ami segítségével Java programokat készíthetünk.
- Linter: Szoftver, mely programozási elvekkel szembemenő implementációt és alapvető hibákat jelez statikus analizist használva.
- MVC: Model-View-Controller: Egy szoftvertervezési minta, ami szétválasztja a szoftvert három részre. Ezek közül a 'Model' (Modell) a szoftver struktúráját, belső felépítését és logikáját, a 'View' (Nézet) magát a Modellt a felhasználó számára megjelenítő rendszert és a 'Controller' (Vezérlő) az eseményeket feldolgozó, a Modellt vezérlő rendszert jelenti.
- VCS: Version Control System: A Verziókezelő Rendszer segítségével több fejlesztő dolgozhat egyszerre egy szoftver projekten. Emellett kifejezetten hasznos lehet biztonsági mentések helyett és egyéb fejlesztési folyamatok támogatásához is.

1.1.4 Hivatkozások

- Feladatkiírás: <https://www.iit.bme.hu/file/11582/feladat>
- Feladat ütemezése: <https://www.iit.bme.hu/targyak/BMEVIIIAB02/ütemterv-határidők>
- Objektumorientált tervezési elvek áttekintés: https://www.iit.bme.hu/system/files/uploads/module_files/OoDesignPrinciplesHu.pdf

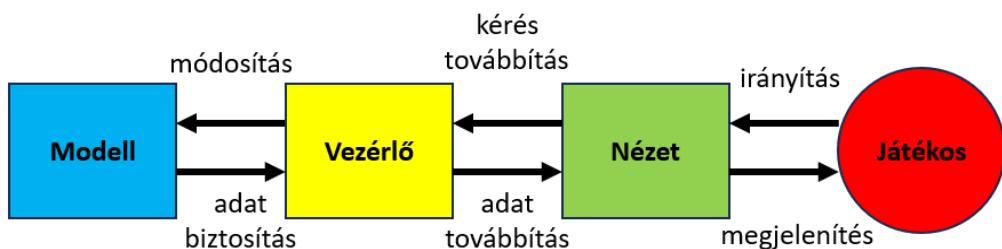
1.1.5 Összefoglalás

A dokumentum legfőképpen az eddig leírtak mellett az elkészítendő szoftver modelljével, annak követelményeivel foglalkozik. A következő részekben helyet kapnak a különböző fejlesztési követelmények pontos leírása és a szoftver architektúrájához, illetve a projekt fejlesztéséhez kapcsolódó információk.

1.2 Áttekintés

1.2.1 Általános áttekintés

A program felépítését MVC minta alapján készítjük el. Az alábbi ábrán látható módon a megjelenítés és logikai modell külön modulként valósulnak meg. A kettő között van az irányító modul, ami a programot vezérli, és minden két másik egységgel interfészken keresztül kommunikál.



1. ábra: MVC modell

1.2.2 Funkciók

A Logarléc nevű játékprogram a Műegyetem Központi épületébe vezeti az azzal egyszerre játszó felhasználókat. A valósághoz hasonlóan a játékban is könnyű eltévedni, ugyanis felépítése egészen labirintusszerű, melynek szobái számos különlegességet rejtenek. A labirintus tehát szobákból épül fel, a játékosoknak, mint hallgatóknak ezek egyikében kell fellelnie a Logarléct a győzelemhez.

A szobák között legegyszerűbben ajtókon keresztül lehet közlekedni. Az ajtók különbözöképpen viselkedhetnek, ugyanis vannak ajtók, melyeket csak az egyik irányba lehet használni. Ennek megfelelően minden szobának van legalább egy ajtaja, amin keresztül át lehet menni egy másik szobába, de ilyenből tetszőlegesen sok is lehet. A szoba rendelkezik egyéb tulajdonságokkal is, az egyik ilyen a szoba befogadóképessége, ami meghatározza, hogy egyszerre legfeljebb hány személy (hallgató és oktató) tartózkodhat az adott szobában. Ezentúl egy szoba elátkozott is lehet, melynek ajtajai időnként eltűnnek, majd később újra előtűnnek. Elátkozott szobánál az összes ajtó eltűnik, akkor is ha van benne ember, de az ajtók csak belülről tűnnek el. Bemenni lehet az éppen elátkozott szobába is, ha nem az elátkozott szobából kifele irányuló egyirányú ajtóról van szó. Egy szoba nem csak elátkozott, hanem akár gázos is lehet. Ezekben mérgező gáz van, és az ide belépő személyek egy időre eszméletüket vesztik, és a náluk lévő tárgyakat elejtik.

A szobák egy, a labirintust még inkább kalandossá tevő képessége, hogy képesek osztódní és egyesülni. Mind az egyesülés, mind az osztódás csak akkor történhet, ha egyik érintett szobában sincs egy személy sem. Egyesülés esetén minden két szomszédos szobából lesz egy, melynek ugyanazok a szomszédai, mint addig a két szobának voltak, egymást kivéve. Az új szoba befogadóképessége a korábbi két szoba közül a nagyobb befogadóképességet öröklí, míg az elátkozott és mérgező gáz tulajdonsághoz elég, ha az egyik korábbi szoba rendelkezett az adott tulajdonsággal. Amikor egy szoba két szobára osztódik, a két szoba befogadóképessége megegyezik az eredeti szobáéval, a többi tulajdonság viszont szétosztódik a két szoba között. Ha az eredeti szoba gázos és elátkozott is volt, akkor a keletkező két szobából az egyik csak gázos, a másik csak elátkozott lesz. Ha csak az egyik tulajdonsággal rendelkezett az eredeti szoba, akkor az osztódás után az egyik szoba rendelkezik az adott tulajdonsággal, a másik viszont nem. A létrejövő két szoba egymással szomszédos lesz (az ajtó kétirányú), ezen kívül az eredeti szoba ajtajait és tárgyait is

megfelezik egymás között, úgy, hogy egy ajtó/tárgy vagy csak az egyik, vagy csak a másik szobába kerül. Az egyirányú ajtóból osztódás után is azok maradnak.

Míg a játékosok hallgatóként igyekeznek megnyerni a játékot, addig az oktatók ezt igyekeznek megakadályozni. Az oktató a vele egy szobában tartózkodó összes hallgató lelkét elveszi, így az adott hallgatók kibuknak az egyetemről. Ezek a játékosok nem tudnak a továbbiakban játszani. Mivel kicsapták őket az egyetemről, ezért testük sem lesz ott többé, így a szoba kapacitásából sem vesznek el. Valamint ezek a hallgatók dühükben magukkal visznek minden, ami lelkük elvesztésekor náluk volt, a tárgyaikat nem dobják el. Amennyiben minden hallgató kibukott, vagy a kalandra szánt idő lejár, a játék kudarccal ér véget.

A játékosokon ugyanakkor segíthetnek a labirintus szobáiban fellelhető tárgyak. A hallgatók a tárgyakat fel tudják venni, tudják aktiválni, és le is tudják rakni. A Logarléc kivételével az oktatók is képesek tárgyak felvételére. Egy személynél egyszerre legfeljebb öt tárgy lehet. A tárgyak között vannak olyanok, melyek védettséget nyújtanak az oktatókkal szemben. A TVSZ denevérbörre nyomtatott példányai például három alkalommal mentik meg a hallgató életét, utána elveszti varázserejüket és eltűnnek. A szent söröspoharak adott ideig jelentenek védelmet, majd kiürülésük után eltűnnek. A nedves táblatörölő a TVSZ denevérbörre nyomtatott példányához hasonlóan megvédi a hallgatót (az oktató lebénításával), viszont a szent söröspoharakhoz hasonlóan csak egyszer, adott ideig fejti ki hatását. Miután lejár az idő a rongy kiszárad, elporlad. A dobozolt káposztás camembert felbontáskor mérges gázt bocsát ki. Tehát az aktiválás helyszínét gázos szobává teszi, majd maga a sajt is elgázosodik, eltűnik. Ha egy szoba mérgezett gázzal teli, az a játék végéig az marad. Az a személy, aki FFP2-es maszkkal rendelkezik adott időre védettséget szerez a mérges gázzal szemben. Mivel a maszk folyamatosan elhasználódik, így a hatásának időtartama egyre csökken. Amint végleg elveszti a hatását, a személy akinél volt a veszélyes hulladékba teszi, ezzel ez is megszűnik. Tehát az idáig felsorolt tárgyak a végső használatuk után eltűnnek (a játékos kezéből és a labirintusból is). Amennyiben egy hallgató szeretne időt spórolva visszatérni egy biztos állomásához, arra is van lehetősége. A szobákban ugyanis elvétve tranzisztorok is találhatók. Ha egy hallgatónál két tranzisztor is van, ezeket összekapcsolhatja. Ekkor összekapcsolt tranzisztorok lesznek, ami azt jelenti, hogy tudnak egymás aktuális tartózkodási helyéről és más tranzisztorral az inaktiválódásig nem tudnak összekapcsolni, de továbbra is két külön tárgy marad. Ha egy hallgató az egyik tranzisztor leteszi, akkor később, a kezében lévő tranzisztor bekapcsolásával abba a szobába kerül, ahol a másik tranzisztor éppen tartózkodik. Ekkor a hallgató eredeti tartózkodási szobájában lévő tranzisztor ott marad a szobában, és inaktiválódik. A másik tranzisztor (ami a hallgatóval egy szobában lesz) inaktív marad. A tranzisztorok tehát nem tűnnek el, így később újra használhatók. Összekötött tranzisztorok a játék során már nem köthetők össze egy harmadik tranzisztorral, csak egy párruk lehet.

1.2.3 Felhasználók

Minden felhasználó ugyanazokhoz a funkciókhöz fér hozzá. Ajánlott kor: 10-100 év. A játék használatához a használati útmutatón kívül nincs szükség előzetes ismeretekre.

1.2.4 Korlátozások

A szoftver x86-os architektúrán, Windows operációs rendszer alatt futtatható JRE segítségével. A szoftver szabadon terjeszthető, jó játékot mindenkinél!

1.2.5 Feltételezések, kapcsolatok

A feladatkiírás a megrendelő szerepét tölti be, ami ott szerepel, azt kell legfőképpen követni programozás során. Az OO irányelvek segítséget nyújtanak a tervezés és fejlesztés során a fejlesztő csapatnak az átláthatóbb, minőségibb kód írásához.

1.3 Követelmények

1.3.1 Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
LOG001	A felhasználó egyszerre több hallgatót irányíthat	Két hallgató létrehozása. Az egyikkel végzett lépések nem befolyásolják a másik állapotát.	Alapvető	Megrendelő	Mozgás	
LOG002	A játék helyszíne több szobából áll	Labirintus létrehozása és karakter mozgatása a szobák között	Alapvető	Megrendelő	Labirintus létrehozása, Mozgás	
LOG003	Az egyik szobában megtalálható a logarléc	Labirintus létrehozása, majd szobákban levő tárgyak listázása	Alapvető	Megrendelő	Labirintus létrehozása	
LOG004	A szobák között lehet közlekedni ajtókon keresztül	Két szoba és köztük egy ajtó létrehozása. Egy karakter átmozgatása egyik szobából a másikba.	Alapvető	Megrendelő	Mozgás	

LOG005	A szoftver ellenőrizze a szoba befogadó képességét belépés előtt	Két szoba és közük egy ajtó létrehozása. Az egyik szoba megtöltése karakterekkel, a másik szobába egy karakter elhelyezése. A másik szobából a karakter mozgatása az egyik szobába.	Alapvető	Megrendelő	Mozgás	
LOG006	Elátkozott szoba ajtajai belülről eltűnnek	Egy elátkozott szoba generálása, és ajtó eltűnés szimulálása	Alapvető	Megrendelő	Ajtó eltűnése	
LOG007	Gázos szobában a karakterek eszméletüket vesztik és elejtik a tárgyaikat	Belépés egy mérgező gázos szobába	Alapvető	Megrendelő	Gázos szobába lépés	
LOG008	Csak akkor egyesül két szoba, ha nincs benne karakter	Mesterséges tesztesettel	Alapvető	A csapat	Szobák egyesítése	
LOG009	Szoba egyesülésénél a két előző szoba tulajdonságai összeadódnak, mérete a nagyobbikéval egyenlő lesz	Mesterséges tesztesettel	Alapvető	Megrendelő	Szobák egyesítése	
LOG0010	Csak akkor osztódik egy szoba, ha nincs benne karakter	Mesterséges tesztesettel	Alapvető	Megrendelő	Szoba szétosztása	

LOG011	A szoba osztódásánál a szomszédai és tulajdonságai elosztódnak a két új szoba között, méretük azonos lesz az eredeti szobáéval	Mesterséges tesztesettel	Alapvető	Megrendelő	Szoba szétosztása	
LOG012	Osztódás után a két új szoba szomszédos egymással minden két irányban	Mesterséges tesztesettel	Alapvető	Megrendelő	Szoba szétosztása	
LOG013	Oktató elveszi a vele egy szobában levő hallgatók lelkét	Játékosként egy szobába kerülés egy oktatóval	Alapvető	Megrendelő	Szobában lévő hallgatók kibuktatása	
LOG014	A játék vége kudarccal, ha nincs túlélő hallgató	Minden játékos találkozik egy oktatóval	Alapvető	Megrendelő	Szobában lévő hallgatók kibuktatása	
LOG015	A játék vége kudarccal, ha lejárt az idő	Megvárni amíg az idő lejár	Alapvető	Megrendelő	A játék nehezítése	
LOG016	Labirintus létrehozása után egyes szobákban tárgyak vannak	Labirintus létrehozása, majd a szobákban lévő tárgyak listázása	Alapvető	Megrendelő	Labirintus létrehozása, Tárgy létrehozása	
LOG017	A tárgyat fel tudják venni a karakterek	Egy tárgy felvétele játékosként	Alapvető	Megrendelő	Tárgy felvétele	
LOG018	A tárgyat le tudják tenni a karakterek	Egy tárgy letétele játékosként	Alapvető	Megrendelő	Tárgy letevése	
LOG019	A tárgyat tudják aktiválni a hallgatók	Egy tárgy aktiválása játékosként	Alapvető	Megrendelő	Tárgy aktiválása	

LOG020	Egy karakternél egyszerre legfeljebb 5 tárgy lehet	Megpróbálni felvenni egy 6. tárgyat játékosként	Alapvető	Megrendelő	Tárgy felvétele	
LOG021	TVSZ denevérbőrre nyomtatott példányainak működése	Egy ilyen tárgy kipróbálása oktatóval szemben négyeszer	Alapvető	Megrendelő	Szobában lévő hallgatók kibuktatása	
LOG022	Szent söröspoharak működése	Egy ilyen tárgy kipróbálása oktatóval szemben az idő letelte előtt és után	Alapvető	Megrendelő	Tárgy aktiválása, Szobában lévő hallgatók kibuktatása	
LOG023	Nedves táblatörlő működése	Egy ilyen tárgy kipróbálása oktatóval szemben az idő letelte előtt és után	Alapvető	Megrendelő	Tárgy aktiválása, Mozgás	
LOG024	Dobozolt káposztás camembert működése	Egy ilyen tárgy kipróbálása	Alapvető	Megrendelő	Tárgy aktiválása	
LOG025	FFP2-es maszk működése	Egy ilyen tárgy kipróbálása gázos szobában	Alapvető	Megrendelő	Gázos szobába lépés	
LOG026	Tranzisztorok működése	Egy ilyen tárgy kipróbálása	Alapvető	Megrendelő	Tárgy aktiválása, Tárgy letevése, Mozgás	
LOG027	Logarléc egy hallgató általi felvételenél győzelem	Logarléc felvétele játékosként	Alapvető	Megrendelő	Tárgy felvétele	
LOG028	Tárgyak - tranzisztor - kivételével használat után (vagy az idő lejárta után) eltűnnék	Mesterséges tesztesettel	Alapvető	A csapat	Tárgy aktiválása	

LOG029	Nincs a tárgyaknak töltési idejük	Mesterséges tesztesettel	Alapvető	A csapat	Tárgy aktiválása	
LOG030	Egy tranzisztor pontosan egy másikkal köthető össze	Mesterséges tesztesettel	Alapvető	A csapat	Tárgy aktiválása	
LOG031	Összekapcsolás után is két tárgy lesz	Mesterséges tesztesettel	Alapvető	A csapat	Tárgy aktiválása	
LOG032	Az oktatók csak felvenni tudják a tárgyakat (kivéve a Logarlécet), lerakni és aktiválni nem (kivéve az FFP2-es maszkot).	Mesterséges tesztesettel	Alapvető	A csapat	Tárgy felvétele	
LOG033	A játékos manuálisan aktiválja a tárgyakat (kivétel a TVSZ és az FFP2, amik automatikusan aktiválódnak)	Mesterséges tesztesettel	Alapvető	A csapat	Tárgy aktiválása	

1.3.2 Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
RES001	A projekt fájlok a Git VCS rendszerrel legyenek nyilvántarva	A fejlesztés során, automatikus	Fontos	Fejlesztők	
RES002	A megfelelő működést a fejlesztés során automatikus tesztek ellenőrizzék	A fejlesztés során, automatikus	Fontos	Fejlesztők	

RES003	A projektet a megfelelő build rendszer segítségével automatikusan le lehessen fordítani	A fejlesztés során, automatikus	Fontos	Fejlesztők	
--------	---	---------------------------------	--------	------------	--

1.3.3 Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
SUB001	A projekt hiba (és jelentős figyelmeztetések) nélkül lefordul	Fordítás során	Magas	Fejlesztők	
SUB002	Minden teszt sikeres	Megfelelő tesztkörnyezet használatával	Alapvető	Fejlesztők	
SUB003	A kész program fut a kari felhő gépein	Kézi teszt (játék)	Alapvető	Megrendelő	
SUB004	A játékot meg lehet nyerni	Kézi teszt (játék)	Alapvető	Megrendelő	

1.3.4 Egyéb nem funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
AUX001	A szoftver ne tartalmazzon rossz minőségű kódot	Linter segítségével	Fontos	Fejlesztők	
AUX002	A szoftver ne tartalmazzon biztonsági kockázatot jelentő kódot	Egyszerűbb tesztek, kézi felülvizsgálat	Fontos	Fejlesztők	
AUX003	A szoftver legyen egyszerűen tesztelhető	A teszesetek írása közben	Fontos	Fejlesztők	
AUX004	A kód legyen jól struktúrált, az előírt tervezési modellnek (MVC) megfelelő	Fejlesztés közben	Alapvető	Fejlesztők	
AUX005	Az elkészült program legyen könnyen használható	Fejlesztés közben	Fontos	Fejlesztők	A felhasználó szempontjából

1.4 Lényeges use-case-ek

1.4.1 Use-case leírások

Use-case neve	Tárgy felvétele
Rövid leírás	Egy karakter felvesz egyet a szobában lévő tárgyak közül.
Aktorok	Karakter kezelő, Labirintus manager
Forgatókönyv	A karakter kezelő kiválaszt egy a szobában lévő tárgyat. Ha a karakternél már van 5 tárgy, akkor a tárgy felvétele sikertelen. Egyébként a labirintus manager eltünteti a szobából a kiválasztott tárgyat eltűnik a szobából és a karakter kezelő hozzáadja a karakter tárgyaihoz.

Use-case neve	Tárgy letevése
Rövid leírás	Egy karakter letesz egyet a nála lévő tárgyakból a szobába.
Aktorok	Karakter kezelő, Labirintus manager
Forgatókönyv	A karakter kezelő kiválaszt egy tárgyat a karakter tárgyai közül és eltünteti azt a karakter tárgyai közül. A labirintus manager hozzáadja a szoba tárgyaihoz a kiválasztott tárgyat.

Use-case neve	Tárgy aktiválása
Rövid leírás	Egy karakter aktivál egyet a nála lévő tárgyakból.
Aktorok	Karakter kezelő
Forgatókönyv	A karakter kezelő kiválaszt egy tárgyat a karakter tárgyai közül és aktiválja azt.

Use-case neve	Tárgy létrehozása
Rövid leírás	Egy új tárgy keletkezik az egyik szobában.
Aktorok	Labirintus manager
Forgatókönyv	A létrejött új tárgyat a labirintus manager elhelyezi a megfelelő szobában.

Use-case neve	Mozgás
Rövid leírás	Egy karakter átmozog egyik szobából egy másikba.
Aktorok	Karakter kezelő, Labirintus manager
Forgatókönyv	A karakter kezelő kiválaszt egy olyan szobát, ahová vezet út a karakter jelenlegi szobájából. A labirintus manager ezután a kiválasztott szobában tartja számon a karaktert.

Use-case neve	Gázos szobába lépés
Rövid leírás	Egy hallgató belép a gázos szobába és eszméletét veszti.
Aktorok	Labirintus manager
Forgatókönyv	A hallgató leteszi az összes tárgyat és egy adott ideig nem lehet irányítani.

Use-case neve	Szobában lévő hallgatók kibuktatása
Rövid leírás	Egy oktató belép egy szobába és megpróbálja kibuktatni az összes itt tartózkodó hallgatót.
Aktorok	Labirintus manager
Forgatókönyv	Az összes a szobában lévő hallgató kibukik, ha nincs nála valamilyen védelmet nyújtó tárgy. A labirintus manager eltávolítja a kibukott hallgatókat.

Use-case neve	Ajtó létrehozása
Rövid leírás	Létrejön egy ajtó két szoba között.
Aktorok	Labirintus manager
Forgatókönyv	A labirintus manager eltárolja a létrejött ajtót.

Use-case neve	Ajtó eltűnése
Rövid leírás	Egy ajtó eltűnik egy időre.
Aktorok	Belső kontroller, Labirintus manager
Forgatókönyv	A belső kontroller megadja az eltüntetendő ajtót. A labirintus manager a megadott időre haználhatatlanná teszi az ajtót. Az idő leteltével a labirintus manager újra használható teszi az ajtót.

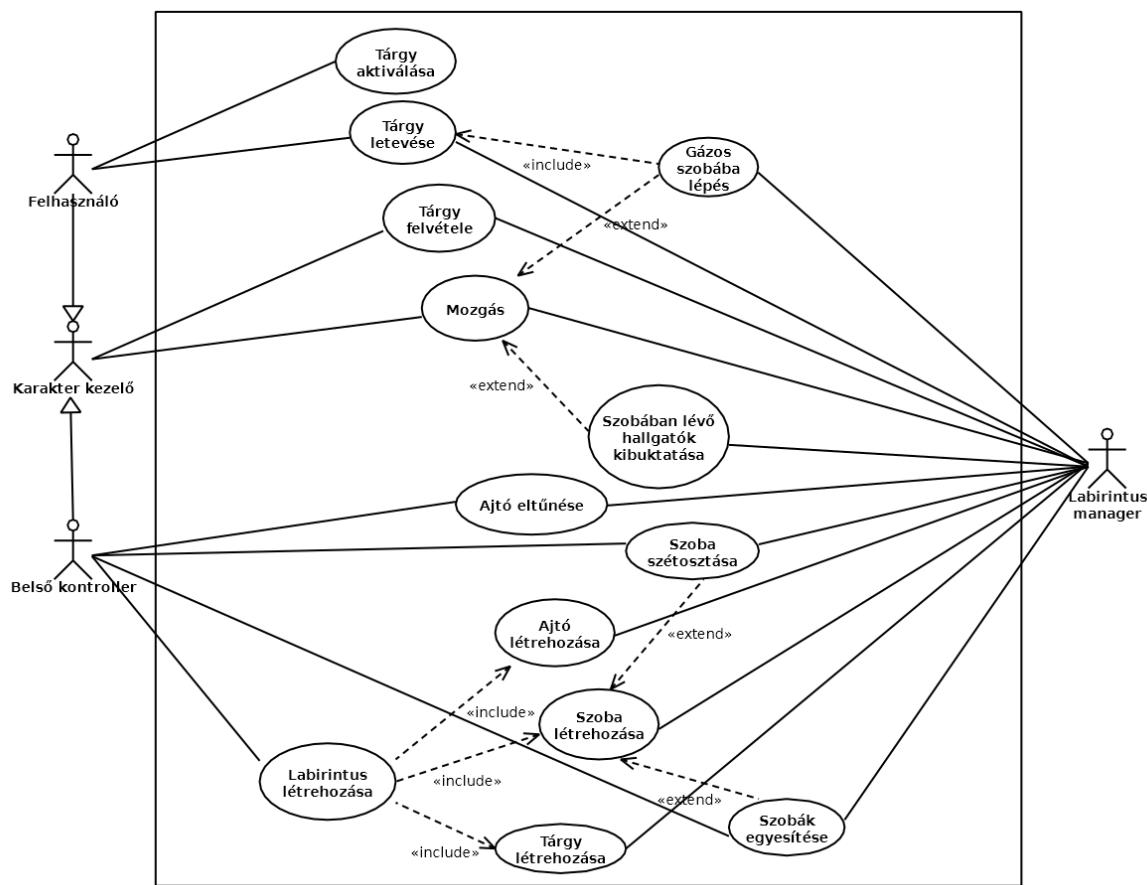
Use-case neve	Szoba létrehozása
Rövid leírás	Létrejön egy új szoba.
Aktorok	Labirintus manager
Forgatókönyv	A labirintus manager eltárolja a létrejött szobát.

Use-case neve	Szoba szétosztása
Rövid leírás	Egy szoba két új szobára válik szét.
Aktorok	Belső kontroller, Labirintus manager
Forgatókönyv	A belső kontorller megadja, hogy mely szobát kell szétosztani. A labirintus manager létrehoz két új szobát a szobafelosztás szabályainak megfelelően és törli az eredeti szobát.

Use-case neve	Szobák egyesítése
Rövid leírás	Kér szoba egyesül egyetlen új szobává.
Aktorok	Belső kontroller, Labirintus manager
Forgatókönyv	A belső kontorller megadja, hogy mely szobákat kell egyesíteni. A labirintus manager létrehoz egyetlen új szobát a szobaegyesítés szabályainak megfelelően és törli az eredeti szobákat.

Use-case neve	Labirintus létrehozása
Rövid leírás	Egy új labirintus jön létre.
Aktorok	Belső kontroller
Forgatókönyv	A belső kontroller létrehoz szobákat, majd létrehoz beléjük tárgyakat és közéjük ajtókat.

1.4.2 Use-case diagram



1.5 Szótár

Fogalom	Leírás
ajtó	Átjáró két szoba között. A közlekedés elsődleges formája a játékban.
befogadóképesség	A szoba kapacitása. Meghatározza, hogy az adott szobában maximum hány hallgató és oktató tartózkodhat egyszerre.
dobozolt káposztás camembert	Egy tárgy, amelyet ha a hallgató felbont akkor mérgező gázt bocsát ki. A szoba, ahol kibontotta, a játék végéig mérgező gázzal lesz teli a játék végéig. Ez a tárgy a letétele után eltűnik.
egyirányú ajtó	Olyan ajtó, amit csak az egyik szobából lehet használni.
elátkozott szoba	Olyan szoba melynek ajtajai időnként eltűnnék, majd később előtűnnék. Eltűnéskor a szoba összes ajtaja eltűnik, előtűnéskor pedig az összes ajtaja megjelenik. Ha az ajtók eltűnnék akkor a szobából nem lehet kijutni, viszont a szobába be lehet lépni, tehát csak belülről tűnnek el az ajtók.
eszméletvesztés	Az oktatók és hallgatók cselevő- és mozgásképtelensége. Sem a hallgató, sem az oktató nem tud ideig mozogni, tárgyakat letenni, illetve az oktató nem tudja kibuktatni a hallgatót.
FFP2-es maszk	Egy olyan tárgy, ami a tulajdonosának védeeltséget ad egy adott időre a mérges gázzal szemben. Többször is lehet használni, viszont egyre rövidebb időre ad védeeltséget, majd végül eltűnik.

gázos szoba	Olyan szoba, amiben mérgező gáz van.
hallgató	Olyan karakter a játékban, amit egy játékos irányít.
kibukás	A hallgató számára a játék vége. Akkor következik be, ha a hallgató és egy oktató egy szobába kerül és a hallgató nem tudja megvédeni magát valamilyen tárgy segítségével.
labirintus	A játéktér, ahol az egész játék menete folyik.
Logarléc	Egy tárgy a játékban, melynek megszerzésével a hallgató(k) nyer(nek)
megbénul	Az oktató cselekvő- és mozgásképtelensége. Az oktató nem tud szobák között mozogni, illetve hallgatókat kibuktatni adott ideig.
mérgező gáz	A szobának egy olyan tulajdonsága, ami megfelelő tárgy(ak) letétele nélkül, eszméletvesztést okoz, illetve a hallgatók és oktatók összes tárgyukat elejtik.
nedves táblatörlő rongy	Egy tárgy, amely megvédi a hallgatót az oktatótól adott ideig. Ha kiszárad, akkor eltűnik.
oktató	Olyan a karakter a játékban, amit a számítógép valamelyen algoritmussal irányít és a hallgatók ellen van.
szent söröspohár	Egy olyan tárgy, amit ha a hallgató aktivál, akkor egy adott időre megvédi a kibuktatástól. Az idő lejárta után ez a tárgy eltűnik.
szoba	A labirintus egy helyisége, melynek van legalább egy ajtaja, ami egy másik szobába vezet. Befogadóképességgel és esetleges tulajdonságokkal rendelkezik, illetve tartalmazhat tárgyat/tárgyakat.
szobaegyesülés	Két szomszédos szoba eggyé válása. Az egyesülés után a keletkező szoba szomszédos lesz az eredeti szobák szomszédaival (egymást kivéve), a befogadóképessége a nagyobb befogadóképességet örököli, illetve a két szoba minden tulajdonságát és tárgyát megörökli. Csak akkor mehet végbe az egyesülés, ha egyik érintett szobában sincs egy személy sem.
szobaosztódás	Egy szoba kettéválása. Mindkét keletkező szoba befogadóképessége megegyezik a z eredeti szoba befogadóképességével. A többi tulajdonság és tárgy szétosztódik a két szoba között. Ha az eredeti szoba gázos és elátkozott volt, akkor keletkező szobák közül az csak az egy lesz gázos a másik pedig csak elátkozott. Ha csak egy tulajdonsága volt a szobának, akkor csak az egyik keletkező szoba fog rendelkezni az adott tulajdonsággal a másik nem. A két szoba szomszédos lesz egymással. Az eredeti szoba ajtajait és tárgyait megfelezik egymás között. Az osztódás csak akkor mehet végbe, ha az érintett szobában nem tartózkodik egy személy sem.
szomszédos szobák	Két szoba szomszédos, ha van közöttük (egyirányú) ajtó.
tárgy	Olyan dolog, ami a labirintus egy szobájában található. Az oktatók és hallgatók fel tudják venni és le tudják tenni. Rendelkezik egy adott képességgel, amit esettől függően a tulajdonosa aktiválhat.
tárgy aktiválása	Olyan cselekvés a hallgató részéről, amikor egy nála lévő tárgy képességét használja.
tárgy elejtése	Olyan történés, amikor egy hallgató vagy egy oktató egy tárgy letételére kényszerül.

tárgy felvétele	Olyan cselekvés egy hallgató vagy egy oktató részéről, amikor magához vesz egy tárgyat a szobából, amiben éppen tartózkodik.
tárgy letétele	Olyan cselekvés, amikor egy hallgató egy nála lévő tárgyat elhelyez a szobában, ahol éppen tartózkodik.
tranzisztor	Egy olyan tárgy, aminek az aktiválásához a hallgatónak szüksége van egy párra. Ha rendelkezik a hallgató két tranzisztorral, akkor azokat összekapcsolhatja. Ha összekapcsolás után az egyiket leteszi egy szobába majd ezután egy másik szobában aktiválja (bekapcsolja) a másik tranzisztorról, akkor visszateleportál az először letett tranzisztor szobájába. Használat után a tranzisztor kikapcsol. A tranzisztorok nem tűnnek el, a játék során újrafelhasználhatóak, viszont ha két tranzisztorról már összekapcsoltak, akkor azok nem köthetőek össze egy harmadikkal.
TVSZ denevérbőrre nyomtatott példányai	Egy olyan tárgy, amit ha a hallgató aktivál, akkor megvédi kibuktatástól. Ezt a tárgyat háromszor lehet használni, mielőtt eltűnik.

1.6 Projekt terv

Lépés	Határidő
csapatlakítás, OO elvek áttekintése	feb. 16. 12:00
Követelmény, projekt, funkcionálisitás	feb. 26. 14:15
Analízis modell (I. változat)	márc. 4. 14:15
Analízis modell (II. változat)	márc. 11. 14:15
Szkeleton tervezése	márc. 18. 14:15
Szkeleton elkészítése	márc. 25. 14:15
Prototípus koncepciója	ápr. 8. 14:15
Részletes tervezek	máj. 15. 14:15
Prototípus elkészítése	ápr. 29. 14:15
Grafikus változat tervezek	máj. 6. 14:15
Grafikus változat elkészítése	máj. 22. labor
Egyesített dokumentáció	máj 24. 12.00

A munka elvégzése saját számítógépeken fog történni. A fejlesztéshez szükséges szoftverek a fejlesztőkre vannak bízva. A kód rendezettsége és a csapatmunka lehetővé tételenek érdekében Git-et, illetve Githubot fogunk használni. A dokumentumok megosztásához, Docs,

illetve Microsoft Sharepoint dokumentum használatát választottuk. Kommunikációs platformként Meta Messenger és Microsoft Teams lesz igénybe véve.

1.7 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2024.02.20. 19:45	2 óra	Görömbey Héjja Sági Tömöri Vizhányó	Értekezlet. Csapat és projekt megismerése. Dokumentum értelmezése a konzultációhoz.
2024.02.21. 10:15	1 óra	Görömbey Héjja Sági Tömöri Vizhányó	Konzultáció
2024.02.21. 11:30	1 óra	Görömbey Héjja Sági Tömöri Vizhányó	Személyes értekezlet, Specifikáció pontosítása, felmerülő kérdések eldöntése
2024.02.21. 15:30	2 óra	Tömöri	2.2.2 Funkciók leírásának elkészítése az értekezlet alapján
2024.02.24. 10:30	1 óra	Vizhányó	Funkcionális követelmények leírása
2024.02.24. 23:00	2 óra	Görömbey	Use-case diagram és leírások készítése
2024.02.24. 23:30	2 óra	Sági	Szótár elkészítése
2024.02.25. 14:30	2 óra	Héjja	Kisebb formátumbeli javítások, lényeges bekezdések és egyéb követelmények kitöltése

2024.02.25. 14:45	1 óra	Vizhányó	Követelmények pontosítása; projekt terv, feltételezések, kapcsolatok megírása,
2024.02.25. 15:30	0,5 óra	Görömbey	Követelmények átfogalmazása, use-case-ekhez rendelés
2024. 02. 25. 17:00	1,5 óra	Tömöri	Funkcionális követelmények kiegészítése, program architektúra leírás, általános áttekintés
2024.02.25. 18:00	1 óra	Vizhányó	Funkcionális követelmények írása

2. Analízis modell kidolgozása

2.1 Objektum katalógus

2.1.1 Ajtó

A szobák közti kapcsolatot reprezentálja. A szoba rendelkezik a saját ajtajaival, azaz számítja, hogy melyik szobákba tud átmenni.

2.1.2 Camembert

A tárgy a használatakor begázosítja a szobát, amiben van, majd eltűnik.

2.1.3 FFP2-es maszk

A tárgy adott ideig védelmet nyújt a mérgező gázzal szemben. A hatása egyre kevesebb, miután végleg elfogy, eltűnik. Oktatók is használhatják.

2.1.4 Hallgató

Egy játékos karaktere, aki tárgyak felvételével és használatával igyekezik megóvni magát a rá leselkedő bajtól, hogy a szobák valamelyikében megtalálja a Logarlécet. Számítja, hogy milyen tárgyak vannak nála, szükség esetén aktiválja őket. Ismeri a tartózkodási helyét és képes annak megváltoztatására.

2.1.5 Logarléc

A tárgy, melynek megszerzése a győzelem kapuja. Oktatók nem vehetik fel.

2.1.6 Oktató

A játék egy gonosz karaktere, aki a labirintusban keringve igyekezik megakadályozni a hallgatók győzelmét. Képes tárgyfelvételre. Számítja, hogy milyen tárgyak vannak nála, szükség esetén aktiválja őket. Ismeri a tartózkodási helyét és képes annak megváltoztatására.

2.1.7 Söröspohár

Ez a tárgy aktiválás után adott ideig nyújt védelmet annak a hallgatónak, aki aktiválta, majd eltűnik.

2.1.8 Szoba

A karakterek és tárgyak tartózkodási helye, itt tudnak egymással interakcióba lépni (karakter-karakter vagy tárgy-karakter). A szobáknak minden van szomszéduk, lehetővé teszik az átjárást. Különleges esetekben megnehezítik a játékosok feladatát, gázos vagy elátkozott tulajdonságukkal, illetve a szobák egyesülésével és osztódásával. Tárolja, hogy mely karakterek és tárgyak találhatók benne. Felel azért, hogy a befogadóképességénél többen ne tartózkodjanak benne.

2.1.9 Táblatörlő rongy

Ez a tárgy aktiválás után a vele egy szobában lévő oktatókat adott ideig megbénítja, majd eltűnik.

2.1.10 Tranzisztor

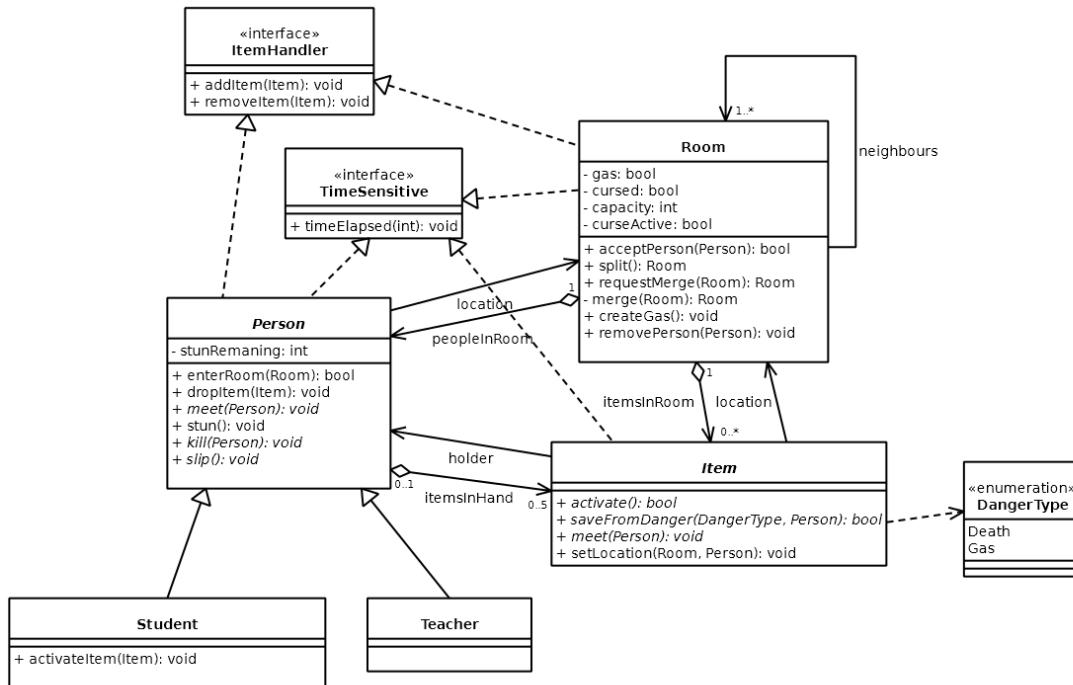
Ennek a tárgynak a használatához a hallgatónak kettő darabra van szüksége. Ha rendelkezik a hallgató két tranzisztorral, akkor azokat összekapcsolhatja. Az összekapcsolás után a játék során

nem lehet összekapcsolni harmadik tranzisztorral. Ha két tranzisztor össze van kapcsolva, akkor azok számontartják egymást, és valamelyik aktiválásával a hallgató átkerül a másik tranzisztor szobájába. Az aktivált tranzisztor az eredeti szobában marad. A tranzisztorok korlátlanul használhatók egymással. A tranzisztor felel a saját tartózkodási helyéért.

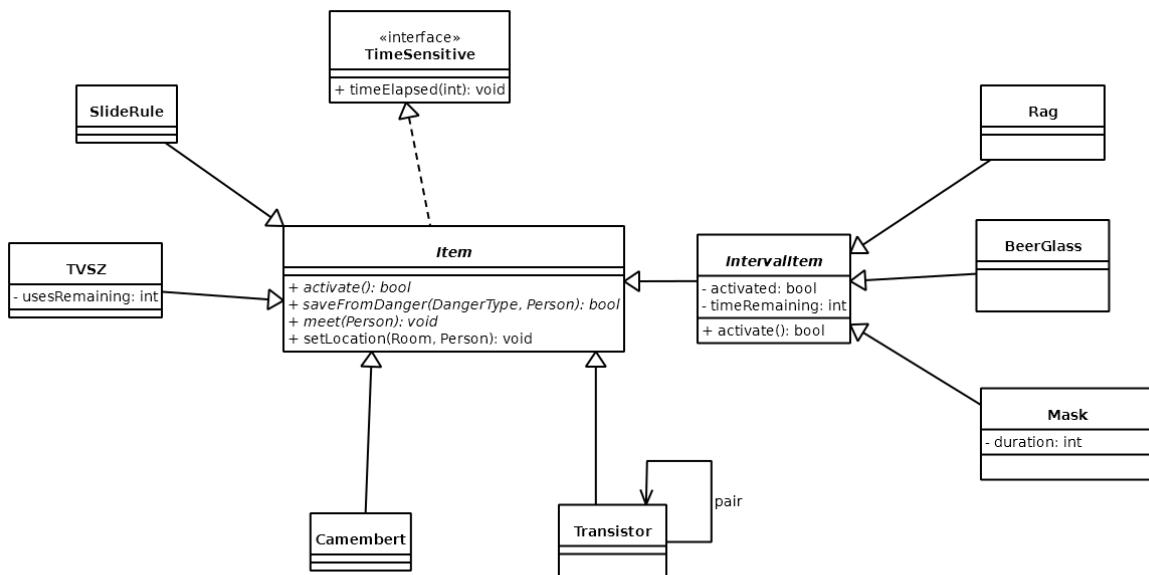
2.1.11 TVSZ

Ez a tárgy három alkalommal használható az oktatók szembeni védelemre, majd eltűnik.

2.2 Statikus struktúra diagramok



1. ábra: Main osztálydiagram



2. ábra: Items osztálydiagram

2.3 Osztályok leírása

2.3.1 BeerGlass

- **Felelősség**

A szent söröspoharak adott ideig védelmet nyújtanak az oktatókkal szemben. A hallgató képes aktiválni, ekkor a söröspohár felelőssége, hogy kezelje magát. Amikor elmúlik a hatása a megszűnésétől kezdeményezi a birtokosának.

- **Ősosztályok**

Item → IntervalItem

- **Interfészek**

- TimeSensitive

- **Attribútumok**

Nem rendelkezik az őséhez képest újabb attribútumokkal.

- **Metódusok**

- **void meet(Person p)**: Nem csinál semmit, mert ha földön van nincs kit megvédenie.
- **bool saveFromDanger(DangerType danger, Person p)**: Amennyiben a danger Death, és aktiválva van a söröspohár, logikai igazzal tér vissza, megvédve a birtokosát. Egyéb esetben logikai hamissal tér vissza.
- **void timeElapsed(int time)**: Ha aktiválva van a tárgy, akkor a timeRemaining értékét csökkenti time-mal. Ha elérte a 0-t, akkor aktuális birtokosánál kezdeményezi a tárgy megsemmisítését.

2.3.2 Camembert

- **Felelősség**

A Camembert sajt aktiválásakor mérgező gázzal árasztja el a szobát. A sajt felelőssége a szoba elgázosításának és utána a megszűnések kezdeményezése.

- **Ősosztályok**

Item

- **Interfészek**

- TimeSensitive

- **Attribútumok**

Nem rendelkezik attribútummal.

- **Metódusok**

- **bool activate()**: A camembert aktiválása, tartózkodási szobáján meghívja a location.createGas() metódust és elgázosítja azt, majd rögtön igényli birtokosánál az önmegsemmisítést.
- **void meet(Person p)**: Nem csinál semmit, mert ha földön van nincs funkciója.

- **bool saveFromDanger(DangerType danger, Person p):** A camembert nem képes hirtelen veszélytől megvédeni a tulajdonosát így mindig logikai hamissal tér vissza.
- **void timeElapsed(int time):** Mivel egyszerhasználatos tárgy, így nem történik vele semmi az idő műlásával.

2.3.3 DangerType

- **Felelősség**

Az enum felsorolja a személyekre leselkedő lehetséges veszélyeket. Az adott személy ennek beállításával tudja jelezni egy tárgynak, hogy mitől kellene őt megvédeni.

- **Konstansok**

- Death: Egy hallgató fenyegeti egy oktató veszélye.
- Gas: A személy mérgező gázzal teli szobába lépett, ettől kell megmenteni.

2.3.4 IntervalItem

- **Felelősség**

Azon tárgyak, melyek adott ideig fejtik ki hatásukat. Felelősségeik tudni, hogy aktiválva vannak-e, illetve, hogy meddig. Amennyiben aktiválva vannak hatással lehetnek a személyekre. Aktiválhatóak. Az IntervalItem osztály absztrakt, nem példányosítható.

- **Ősosztályok**

Item

- **Interfészek**

- TimeSensitive

- **Attribútumok**

- **activated:** Logikai változó. Jelentése: értéke logikai igaz, ha a tárgy aktív, 0 ha még nem.
- **timeRemaining:** Egész szám típus. Jelentése: a tárgy ennyi ideig marad még aktív.

- **Metódusok**

- **bool activate():** Az adott tárgy aktiválása, activated flag bebillentése. Innentől kezdve amíg létezik, képes kifejteni a hatását.

2.3.5 Item

- **Felelősség**

A tárgyak a szereplőket segítő eszközök, melynek használatával különböző előnyökre tehetnek szert. Egy tárgy lehet egy szobában, vagy egy embernél, és a tárgynak a tudás, hogy hol/kinél van, a rendelkezésére áll. Egy tárgy felelőssége megvédeni egy személyt esetleges veszélyektől, amennyiben képes rá, és a saját élettartamát megfelelően vezérelni. A tárgyak továbbá aktiválhatók, amikor a saját képességeket végrehajthatják és felelnek a tartózkodási helyük/birtokosuk aktuálisan tartásáért is. Az Item egy absztrakt osztály, csak a nem absztrakt leszármazottjai példányosíthatók.

- **Ősosztályok**

Nem rendelkezik ősosztályokkal.

- **Interfészek**

- TimeSensitive

- **Attribútumok**

Nem rendelkezik attribútumokkal.

- **Metódusok**

- ***bool activate()***: Az adott tárgy aktiválása.
- ***void meet(Person p)***: Ha a tárgy egy szobában van a földön, akkor új személy belépése esetén a szoba értesíti a tárgyat.
- ***bool saveFromDanger(DangerType danger, Person p)***: A tárgy az adott danger veszéllyel (és az esetleges p személlyel) szemben képes-e védelmet biztosítani.
- ***void setLocation(Room r, Person p)***: Átállítja a tárgy tartózkodási helyét és birtokosát a paraméterként kapottakra.
- ***void timeElapsed(int time)***: Az adott ideig ható, aktivált tárgyak állapotát módosítja.

2.3.6 ItemHandler

- **Felelősség**

Az interfész megvalósítása lehetővé teszi, hogy egy tárgyakkal rendelkezni képes objektum új tárgyhoz férjen hozzá, vagy elhasznált tárgyat töröljön ki.

- **Metódusok**

- ***void addItem(Item it)***: Az it tárgy hozzáadása az objektum tárgynyilvántartásába.
- ***void removeItem(Item it)***: Az it tárgy törlése az objektum tárgynyilvántartásából.

2.3.7 Mask

- **Felelősség**

Az FFP2-es maszkok adott ideig védelmet nyújtanak egy személynek a mérgező gázzal szemben. A hallgató képes manuálisan is aktiválni, de egyébként automatikusan is aktiválódik a személyeknek. A maszk saját felelőssége, hogy amennyiben véget ér a hatása vajon újra használható-e vagy meg kell semmisíteni. Amikor elmúlik a hatása a megszűnésétől kezdeményezi a birtokosának.

- **Ősosztályok**

Item → *IntervalItem*

- **Interfészek**

- TimeSensitive

- **Attribútumok**

- **duration**: Egész szám típus. Tárolja, hogy a következő aktiváláskor mennyi ideig lesz aktív.

- **Metódusok**
 - **void meet(Person p)**: Nem csinál semmit, mert ha földön van nincs kit megvédenie.
 - **bool saveFromDanger(DangerType danger, Person p)**: Amennyiben a danger Gas, és aktiválva van a maszk, logikai igazzal tér vissza, megvédve a birtokosát. Ha a danger Gas, de nincs aktiválva, akkor aktiválódik, és szintén logikai igazzal tér vissza. Egyéb esetben logikai hamissal tér vissza.
 - **void timeElapsed(int time)**: Ha aktiválva van a tárgy, akkor a timeRemaining értékét csökkenti time-mal. Ha elérte a 0-t, akkor adott értékkel csökkenti a durationt és visszabillenti az activated flag-et 0-ra. Ha a duration elérte a 0-t akkor aktuális birtokosánál kezdeményezi a tárgy megsemmisítését.

2.3.8 Person

- **Felelősség**

A játék karaktereinek működéséért felelős, főbb adatait tárolja. Tárolja a személy tartózkodási szobáját és tárgyait. A szereplők képesek találkozni, gyilkolni, elcsúsztani (megbénulni), és elkábulni. Képesek tárgyakat kezelní: felvenni, eldobni, eltávolítani. Tudnak mozogni a szobák között. A Person osztály absztrakt, csak a leszármazottai példányosíthatók.

- **Ősosztályok**

Nem rendelkezik ősosztályokkal.

- **Interfészek**

- ItemHandler
- TimeSensitive

- **Attribútumok**

- **itemsInHand**: [0..5] db Item tárolója. A személy rendelkezésére álló tárgyak, amiket sikeresen felvett. Képes eldobni és eltávolítani.
- **location**: Room referancia. A személy tartózkodási helyét tárolja.
- **stunRemaining**: Egész szám típus. Amennyiben a személy el van kábítva, ennek az értéke adja meg, hogy még mennyi ideig nem mozoghat.

- **Metódusok**

- **void addItem(Item it)**: A paraméterben kapott it tárgy felvétele a személyhez, ha a személy nincs elkábulva. Csak akkor veheti fel, ha még elfér az itemsInHand-ben. Ha felvette, akkor a szobából el kell távolítania és a tárgy birtokosát is beállítja.
- **void dropItem(Item it)**: A paraméterben kapott it tárgy eldobása. Az itemsInHand-ból eltávolítja a tárgyat, hozzáadja a tartózkodási helyének tárgyaihoz és a tárgy birtokosát törli it.setLocation(location, NULL)-lal.
- **bool enterRoom(Room r)**: A p személy mozgását végrehajtó metódus. Ha a személy nincs elkábulva és a jelenlegi szobájának a curseActive tagja hamis, belép a paraméterként kapott r szobába. A szoba felelőssége, hogy a személyt beengedi-e. A visszatérési értéke a beengedés sikeresége, melyről a szobától kap értesítést, az r.acceptPerson(p) visszatérési értékeként. Amennyiben sikeresen átlép a másik szobába, a régi szobájából kilépteti magát a removePerson() metódust használva és frissíti a location-t. Ezután a tárgyainak helyzetét is frissíti setLocation()-nel.
- **void kill(Person p)**: A személy p által végrehajtott kibuktatását/halálát hajtja végre.

- **void meet(Person p):** A személy p-vel való egyirányú találkozását hajtja végre, a személy itt „mutatkozik be” p-nek.
- **void slip():** A személy táblatörlő rongy általi megbénulását hajtja végre.
- **void stun():** A személy mérgező gáz általi megbénulását hajtja végre. Először is végigkérdezi a tárgyait, hogy képesek-e megóvni őt a mérgező gáz általi veszélytől. Ha legalább egy megvédi, akkor nem történik a személlyel semmi. Ha egy sem védi meg, akkor eldobja az összes tárgyat és adott ideig elkábul, a stunRemaining beállítódik.
- **void removeItem(Item it):** A paraméterként kapott it tárgy törlése a személy kezéből. Akkor hívódik meg, ha a tárgy elhasználódott.
- **void timeElapsed(int time):** Továbbítja az eltelt időt (time) a nála lévő tárgyaknak. A személy belső mechanizmusában is részt vesz, például elkábulás esetén csökkenti a stunRemaining-et time értékkel.

2.3.9 Rag

- **Felelősség**

A nedves táblatörlő rongy adott ideig megbénítják az egy helyiségben tartózkodó oktatókat. A hallgató képes aktiválni, ekkor a söröspohár felelőssége, hogy kezelje magát. Amikor elmúlik a hatása a megszűnésétől kezdeményezi a birtokosának.

- **Ősosztályok**

Item → IntervalItem

- **Interfészek**

- TimeSensitive

- **Attribútumok**

Nem rendelkezik az őséhez képest újabb attribútumokkal.

- **Metódusok**

- **void meet(Person p):** Ha egy személlyel találkozik és aktiválva van, akkor kezdeményezi a személy megbénítását a p.slip() metódushívással.
- **bool saveFromDanger(DangerType danger, Person p):** Amennyiben a danger Death, és aktiválva van a rongy, logikai igazzal tér vissza, megvédve a birtokosát és meghívja a p személlyre a slip() metódust, megbénítást kezdeményezve ezzel. Egyéb esetben logikai hamissal tér vissza.
- **void timeElapsed(int time):** Ha aktiválva van a tárgy, akkor a timeRemaining értékét csökkenti time-mal. Ha elérte a 0-t, akkor aktuális birtokosánál kezdeményezi a tárgy megsemmisítését.

2.3.10 Room

- **Felelősség**

Minden szoba számontartja a személyeket és tárgyakat, akik benne tartózkodnak. Ha egy személy belépne a szobába, a szoba felelőssége, hogy csak akkor engedje be, ha befér, valamint a szobák felelnek a személyek kilépéséért is. Amennyiben új személy érkezik a szobába, a szoba mutatja be az új személyt az eddig a szobában tartózkodó személyeknek és tárgyaknak, illetve az új személynek az eddig szobában tartózkodó személyeket. A szoba tudja, hogy melyik szobákkal szomszédos, így a benne lévő személyeknek lehetővé teszik a mozgást. Irányítja a saját viselkedését, az osztódást és egyesülést egy másik szobával, valamint, ha elátkozott, akkor az ajtók eltünését, ha gázos, akkor azt, hogy az új érkezőket elkábítса.

- **Ősosztályok**

Nem rendelkezik ősosztályokkal.

- **Interfészek**

- ItemHandler
- TimeSensitive

- **Attribútumok**

- **capacity:** Egész szám típus. A szoba maximális befogadóképessége. Új személy szobába lépésekor ez alapján dönt arról, hogy beengedi-e vagy sem.
- **cursed:** Logikai változó. Tárolja, hogy a szoba elátkozott-e vagy sem. Ennek lekérdezésével tudja, hogy az ajtajainak el(ő)tünését kell-e módosítani
- **curseActive:** Logikai változó. Amennyiben egy szoba elátkozott, ennek segítségével tudja, hogy jelenleg az ajtajai használhatók-e vagy sem.
- **gas:** Logikai változó. Tárolja, hogy a szoba mérgező-e vagy sem. Új ember szobába lépésekor ennek a lekérdezésével tudja a szoba, hogy elkábítja-e a személyt vagy sem. Inicializáláskor, osztódáskor és camembert hatására állítható.
- **itemsInRoom:** [0..*] db Item tárolója. A szobában található tárgyak, ami nincs senki birtokában, a személyek felvehetik.
- **neighbours:** [1..*]db Room tárolója. Az ajtók megvalósítása: ha egy r1 szobából át lehet menni egy r2 szobába, akkor az r1 neighbours-ben megtalálható r2 referencia. Ha fordítva nem igaz egyirányú ajtóról beszélünk. A személyek ezt használva tudnak közlekedni.
- **peopleInRoom:** [0..capacity] db Person tárolója. A szobában található személyek referenciaja van itt, a személyek interakcióhoz szükséges.

- **Metódusok**

- **bool acceptPerson(Person p):** A paraméterként kapott személyt engedi be a szobába. Amennyiben a szoba kapacitása kimerült nem engedi be a személyt. A visszatérési értéke a beengedés sikeresége. Ha beengedi a személyt, felel az új személy és a szobában tartózkodó személyek kölcsönös találkozásáért, illetve az új személy és szobában levő tárgyak találkozásáért. Ha a szoba mérgezett, felel a belépő játékos elkábításáért.
- **void addItem(Item it):** A paraméterben kapott it tárgy szobához adása. Történhet a játék működése szerint, vagy egy személy által, amennyiben ő a tárgyat eldobta.

- **void createGas()**: Mérgező gázzal tölti fel a szobát, beállítja a gas értékét logikai igazra, majd felel a szobában tartózkodó személyek elkábításáért.
- **Room merge(Room)**: Két szoba egyesítését elvégző belső függvény. Egy r1 szoba hívhatja egy r2 szobának ezt a függvényét, ha r1 beleegyezik az egyesülésbe. Az r2 beleegyezik, ha nincs benne egy személy sem. Ekkor létrehoz egy új szobát, melyben végrehajtja a paraméterként kapott r1 és önmaga egyesítését. Az új szoba kettejük összes szomszédjával rendelkezni fog egymást kivéve, átadja a két szoba összes tárgyát, illetve az új cursed és gas értékek a két szoba értékének logikai VAGY kapcsolatából keletkezik. A tárgyak tartózkodási helyét setLocation()-nel frissíti. A curseActive false lesz, a capacity a két szoba kapacitásairól a nagyobbik lesz. Továbbá a metódus felelőssége, hogy megtörténjen a megfelelő szobáknál a szomszédok listájának frissítése, mely egy másik privát függvényben is történhet. A metódus a létrejött új szobával tér vissza, amennyiben pedig nem egyezik bele az egyesülésbe NULL-t ad át.
- **void removeItem(Item it)**: A paraméterként átadott it tárgy törlése a szobából, amennyiben aktiválás után a szobában hagyák, és lejárt a hatásának időtartama.
- **void removePerson(Person p)**: A paraméterként átadott p személy eltávolítása a peopleInRoom-ból.
- **Room requestMerge(Room)**: Ezzel a függvénytel kerül indítványozásra a meghívott szobánál, hogy egyesüljön a paraméterben átadott szobával. Amennyiben a kérényezett szoba beleegyezik (nincs benne egy személy sem), meghívja a merge függvényt a paraméterként átadott szobának, és a merge-ben magát adja át paraméterként. Ekkor a merge visszatérési értékével tér vissza, ha viszont nem egyezik bele az egyesülésbe NULL-lal tér vissza.
- **Room split()**: Ezzel a függvénytel kerül indítványozásra a meghívott szobánál, hogy kettéosztódjon. Amennyiben tartózkodik benne személy NULL-t ad vissza. Különben pedig létrehoz egy új szobát, aminek neighbours-ei a saját neighbours-ei fele és saját maga, az itemsInRoom a saját itemsInRoom-jainak szintén fele lesz. Az átadott szomszédokat és tárgyakat saját magából eltávolítja, az új szobát beállítja saját szomszédjának is. Az átadott tárgyak location-jét frissíti setLocation()-nel. A gas és cursed értékeiből, ha mindenki logikai igaz volt, akkor az erediben a gas hamis lesz, és az újban lesz a gas igaz. Ha a két értékből nem volt mindenki logikai 1, akkor az új szoba mindenki értéke hamis lesz. Az új szoba capacity-je a régiével egyezik meg.
- **void timeElapsed(int time)**: Továbbítja az eltelt időt (time) a benne lévő személyeknek és tárgyaknak. A szoba belső mechanizmusában is részt vesz, például egy elátkozott szoba ajtajainak el(ö)tűnése az eltelt idő alapján. Mivel idő telt el, a továbbra is szobában tartózkodó tárgyakat összetalálkoztatja minden személlyel és minden személyt kölcsönösen összetalálkoztat egymással.

2.3.11 SlideRule

- **Felelősség**

A Logarléc a hallgatók győzelmének kulcsa. Amint egy hallgató felvétel után aktiválja, győznek a hallgatók. Felelőssége, hogy aktiváláskor a játék véget érjen, és hogy oktató ne tudja felvenni.

- **Ősosztályok**

Item

- **Interfészek**

- TimeSensitive

- **Attribútumok**

Nem rendelkezik attribútummal.

- **Metódusok**

- **bool activate()**: A Logarléc aktiválásával győznek a hallgatók és véget ér a játék
- **void meet(Person p)**: Nem csinál semmit, mert ha földön van nincs funkciója.
- **bool saveFromDanger(DangerType danger, Person p)**: A Logarléc nem képes hirtelen veszélytől megvédeni a tulajdonosát így minden logikai hamissal tér vissza.
- **void timeElapsed(int time)**: Mivel egyszerhasználatos tárgy, így nem történik vele semmi az idő műlásával.

2.3.12 Student

- **Felelősség**

A hallgatókat reprezentáló osztály. A játékosok által elvégezhető funkciókat valósítja meg. Felelőssége, hogy a hallgató megfelelően használhassa a tárgyakat a védelem érdekében, és hogy akkor haljon meg vagy kábuljon el, amikor szükséges.

- **Ősosztályok**

A Person absztrakt osztály leszármazottja.

- **Interfészek**

- ItemHandler
- TimeSensitive

- **Attribútumok**

Nem rendelkezik újabb attribútumokkal.

- **Metódusok** (az osztály új, vagy felüldefiniált metódusai)

- **void activateItem(Item it)**: A paraméterben kapott it tárgy használata.
- **void kill(Person p)**: A hallgató megpróbálja megvédeni magát a haláltól azzal, hogy az összes tárgyat megpróbálja felhasználni az oktató elleni védelemre. Amint az egyik tárgy megvédi a hallgatót, a metódus véget ér és a hallgató ép marad. Ha a hallgató meghal, az ő felelőssége törölni magát a tartózkodási helyéről. A p paraméter az a személy, aki megöli a személyt, ezt adj a további tárgyaknak.
- **void meet(Person p)**: Nem történik semmi, mivel egy hallgató nem kezdeményez semmit más személyekkel való találkozáskor.
- **void slip()**: Nem történik semmi. A hallgató biztonságban marad, mert a táblatörlő rongy őt nem veszélyezteti.

2.3.13 Teacher

- **Felelősség**

Az oktatókat reprezentáló osztály. Felelőssége, hogy az oktatók is mozogjanak, és képes legyen fenyegetni a hallgatókat. A személyek alapvető képességein túl a lebénulás, gyilkolás és FFP2-es maszk használatára van lehetősége.

- **Ősosztályok**

A Person absztrakt osztály leszármazottja.

- **Interfészek**

- ItemHandler
- TimeSensitive

- **Attribútumok**

Nem rendelkezik újabb attribútumokkal.

- **Metódusok** (az ősosztály absztrakt metódusainak definiálása)

- **void kill(Person p):** Nem történik semmi, ugyanis oktató nem halhat meg.
- **void meet(Person p):** Az oktató minden p személyt akivel találkozik megpróbál meggyilkolni, amennyiben nincs lebénulva.
- **void slip():** Az oktató a táblatörő rongy hatására lebénül. A stunRemaining adott értékre lesz állítva, és amíg nem lesz nulla, nem képes gyilkolni, tárgyat felvenni és mozogni.

2.3.14 TimeSensitive

- **Felelősség**

Ezen az interfészen keresztül értesülnek a játék objektumai az idő teléséről.

- **Metódusok**

- **void timeElapsed(int time):** Ezen keresztül értesülnek az interfész megvalósító objektumok azzal, hogy a játékban time idő eltelt.

2.3.15 Transistor

- **Felelősség**

A tranzisztorok lehetőséget nyújtanak a hallgatóknak nem szomszédos szobák közti gyorsutazásra. Ahhoz, hogy egy hallgató ezzel élhessen, két tranzisztorra is szüksége van. A tranzisztorok felelőssége, hogy össze tudjanak kapcsolódni egymással, amennyiben minden két hallgatónál van. Az összekapcsolt tranzisztorok számítartják egymást is, így biztosítva a kapcsolatot a két tárgy tartózkodási szobái között. A tranzisztorok örökifjük, korlátlanul használhatók a párjukkal. Ha két tranzisztor egyszer összekapcsolódott, már nem tudnak szétkapcsolódni. Kettőnél több tranzisztor nem csatlakozhat össze.

- **Ősosztályok**

Item

- **Interfészek**
 - TimeSensitive
- **Attribútumok**
 - **pair**: Transistor referencia. Ha nincs összekapcsolva NULL, ha igen, akkor a párja.
- **Metódusok**
 - **bool activate()**: Két tranzisztor összekapcsolását illetve azok használatát is lehetővé tevő metódus. Amennyiben egy t1 tranzisztort úgy aktiválunk, hogy a hallgatónál még nincs másik aktivált tranzisztor és a t1.pair NULL, akkor a t1 párosítási szándékát eltároljuk. Amennyiben egy t2 tranzisztort úgy aktiválunk, hogy a t2.pair értéke NULL, de a hallgató aktuálisan is rendelkezik egy már párosítási szándékra bejegyzett t1 tranzisztorral, akkor a t1-et és a t2-t összekapcsolja, és kiveszi őket a párosítási jegyzékből. Amennyiben egy t1 tranzisztort úgy aktiválunk, hogy a t1.pair értéke nem NULL, akkor t1-et eldobja a játékos az eredeti szobájában és a t1 birtokosát átlépteti az enterRoom() metódus segítségével, t2 tartózkodási helyébe, amit paraméterként ad át a személy metódusába.
 - **void meet(Person p)**: Ilyenkor nem kell semmit se csinálja.
 - **bool saveFromDanger(DangerType danger, Person p)**: A tranzisztor nem képes hirtelen veszélytől megvédeni a tulajdonosát így minden logikai hamissal tér vissza.
 - **void timeElapsed(int time)**: Direktbe nem történik vele semmi az idő műlásával, de a tartózkodási helye és tulajdonosa frissítődik.

2.3.16 TVSZ

- **Felelősség**

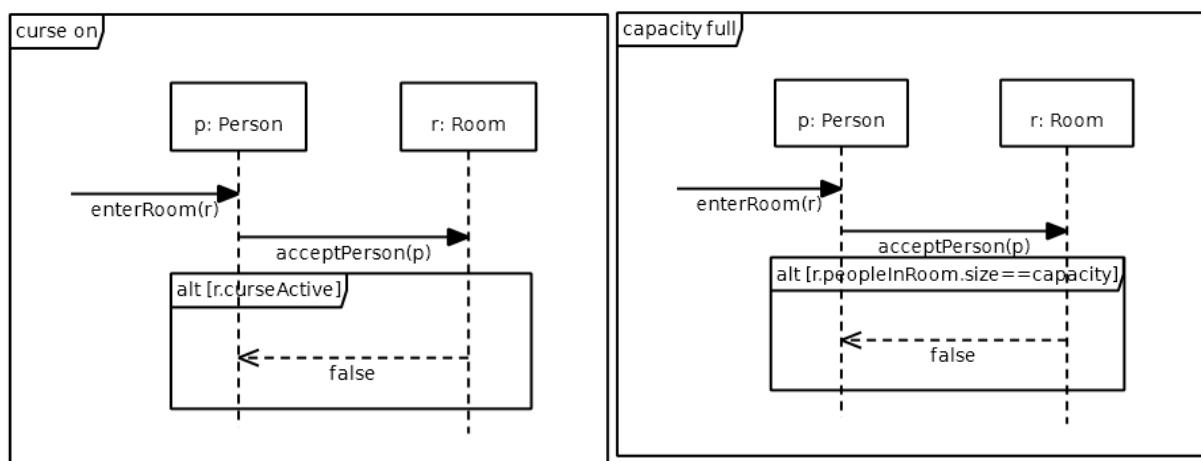
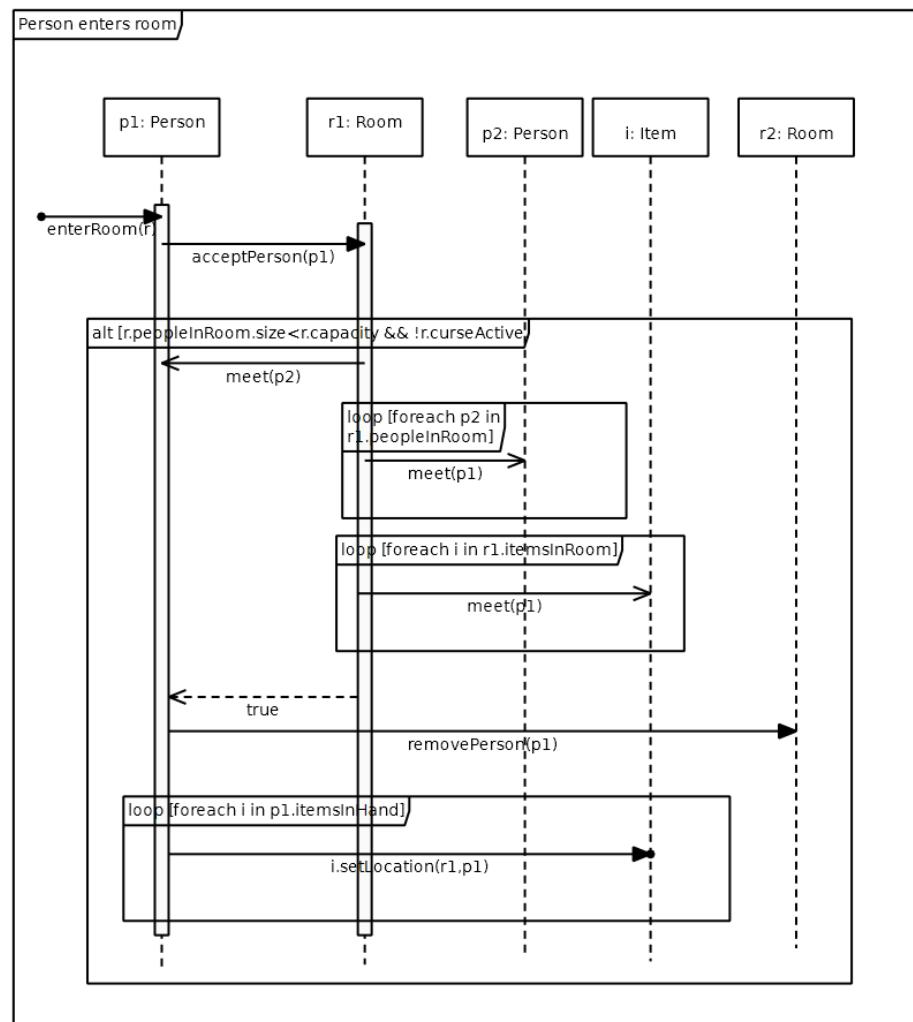
A TVSZ denevérbőrre írt példánya veszély esetén megvédi az oktatótól az őt hordozó hallgatót. Felelőssége, hogy veszély esetén, ha rá kerül a sor automatikusan aktiválódjon. Továbbá, ha többször nem használható kezdeményezi a megsemmisítését birtokosánál.

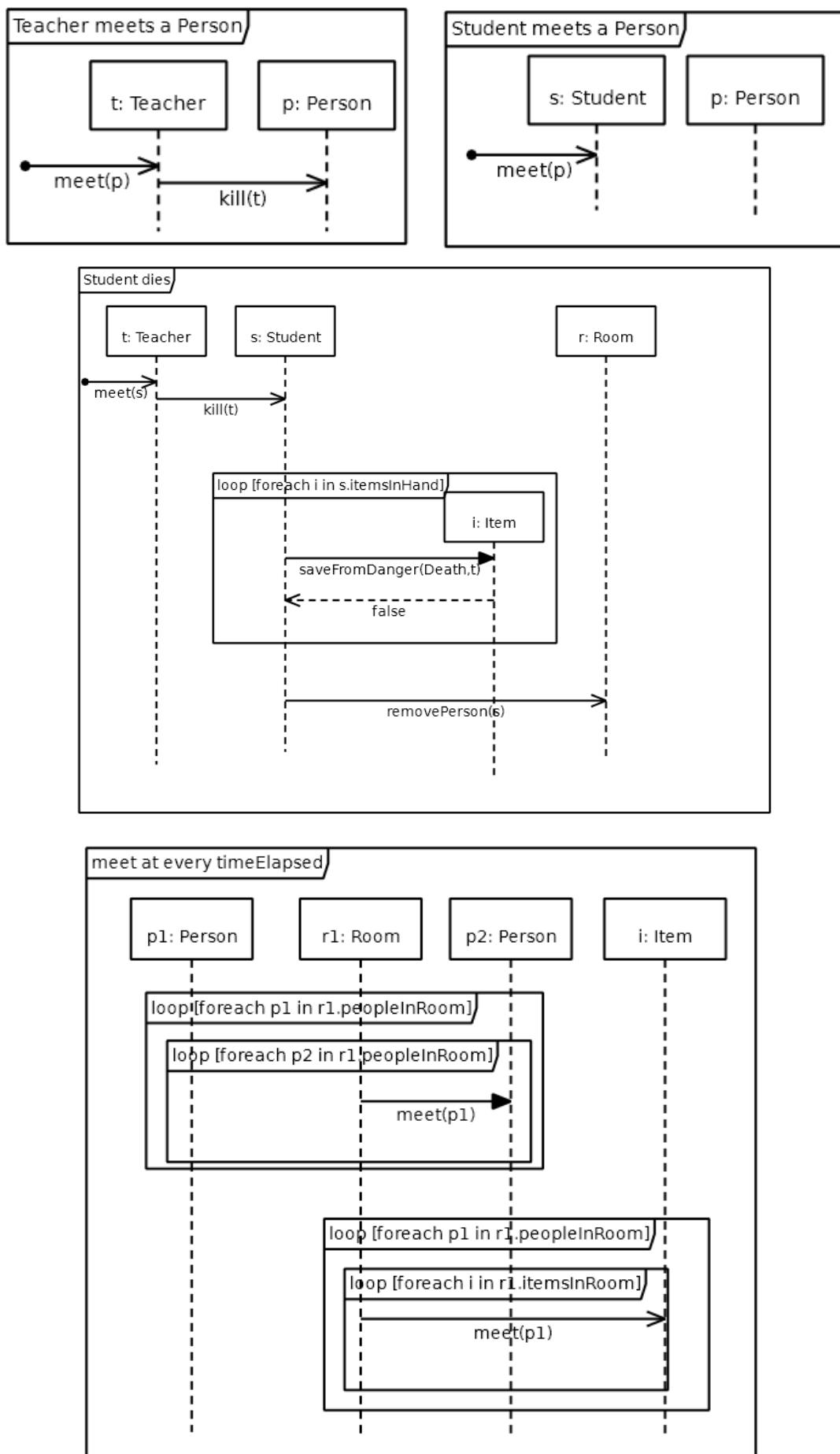
- **Ősosztályok**

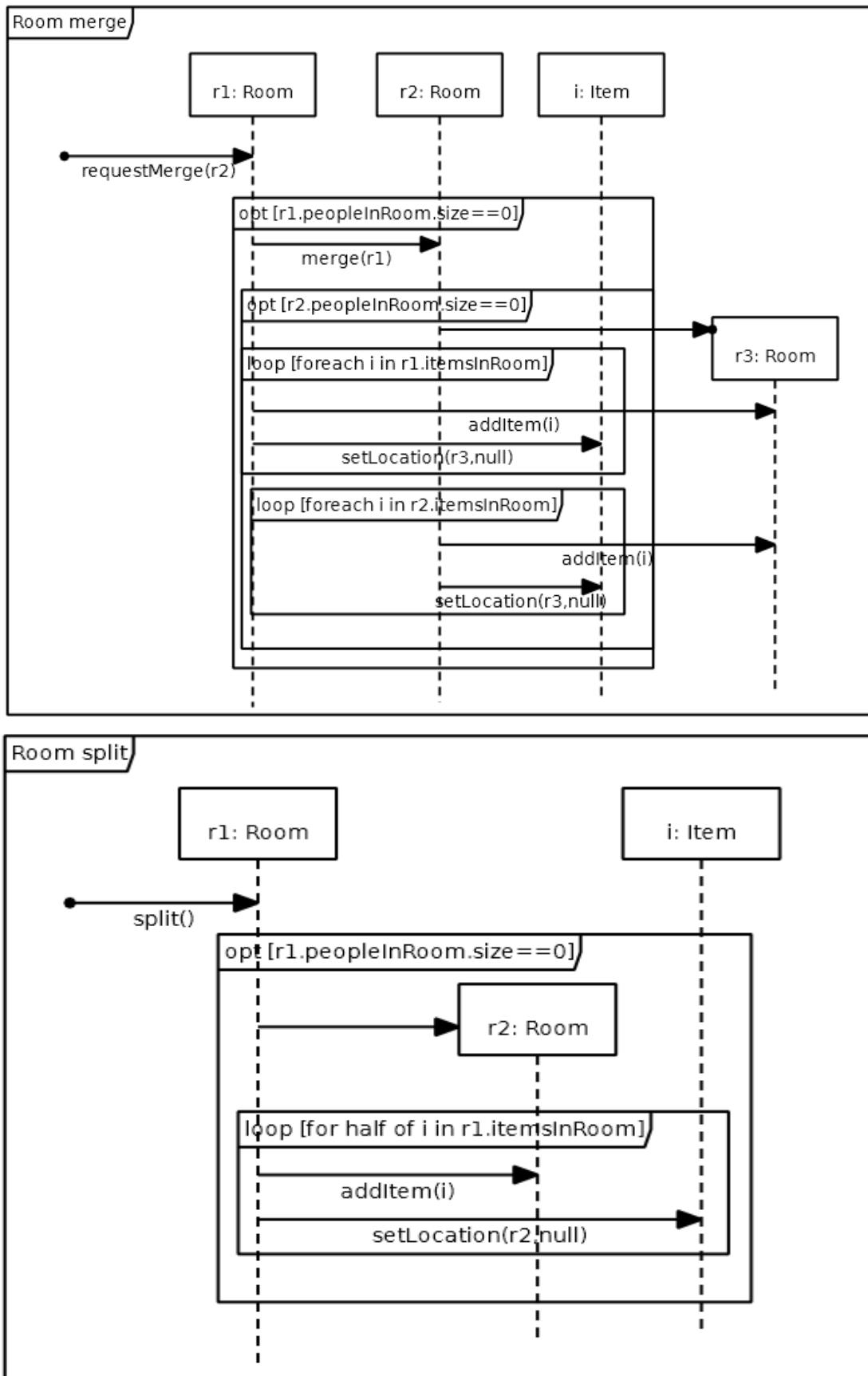
Item

- **Interfészek**
 - TimeSensitive
- **Attribútumok**
 - **usesRemaining**: Egész szám típus. Tárolja, hogy a tárgy még hány alkalommal tud védelmet biztosítani egy oktatóval szemben.
- **Metódusok**
 - **bool activate()**: A TVSZ manuális aktiválása nem lehetséges, így ekkor nem történik semmi.
 - **void meet(Person p)**: Nem csinál semmit, mert ha földön van nincs funkciója.
 - **bool saveFromDanger(DangerType danger, Person p)**: Amennyiben a danger Death, logikai igazzal tér vissza, megvédve a birtokosát és csökkenti a usesRemaining-et 1-gyel. Amennyiben a változó értéke 0 lesz megsemmisül. Ha a danger értéke nem Death, logikai hamissal tér vissza.
 - **void timeElapsed(int time)**: Mivel egyszerhasználatos tárgy, így nem történik vele semmi az idő műlásával.

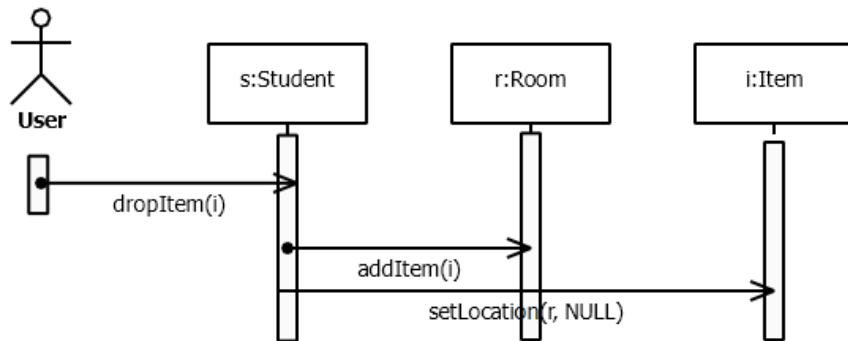
2.4 Szekvencia diagramok



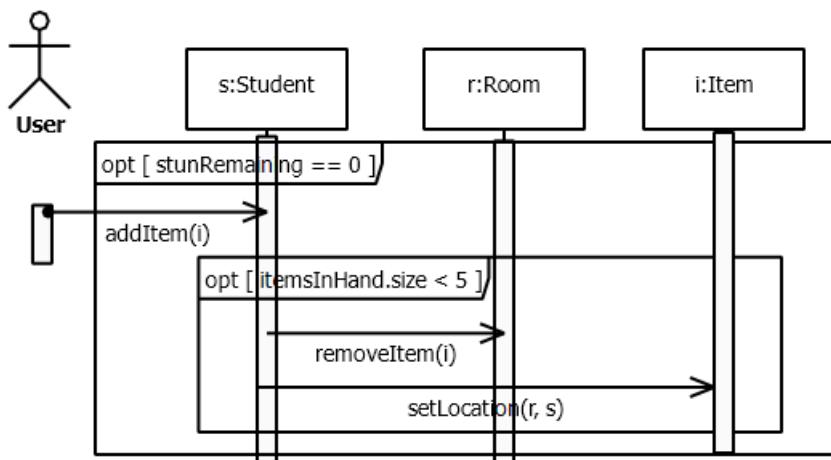




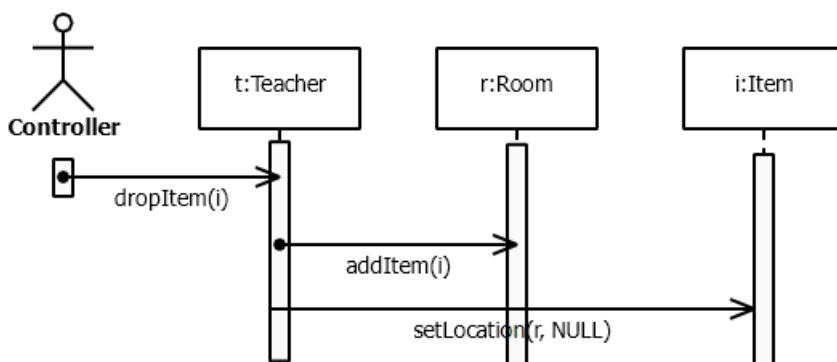
sd Student drop item

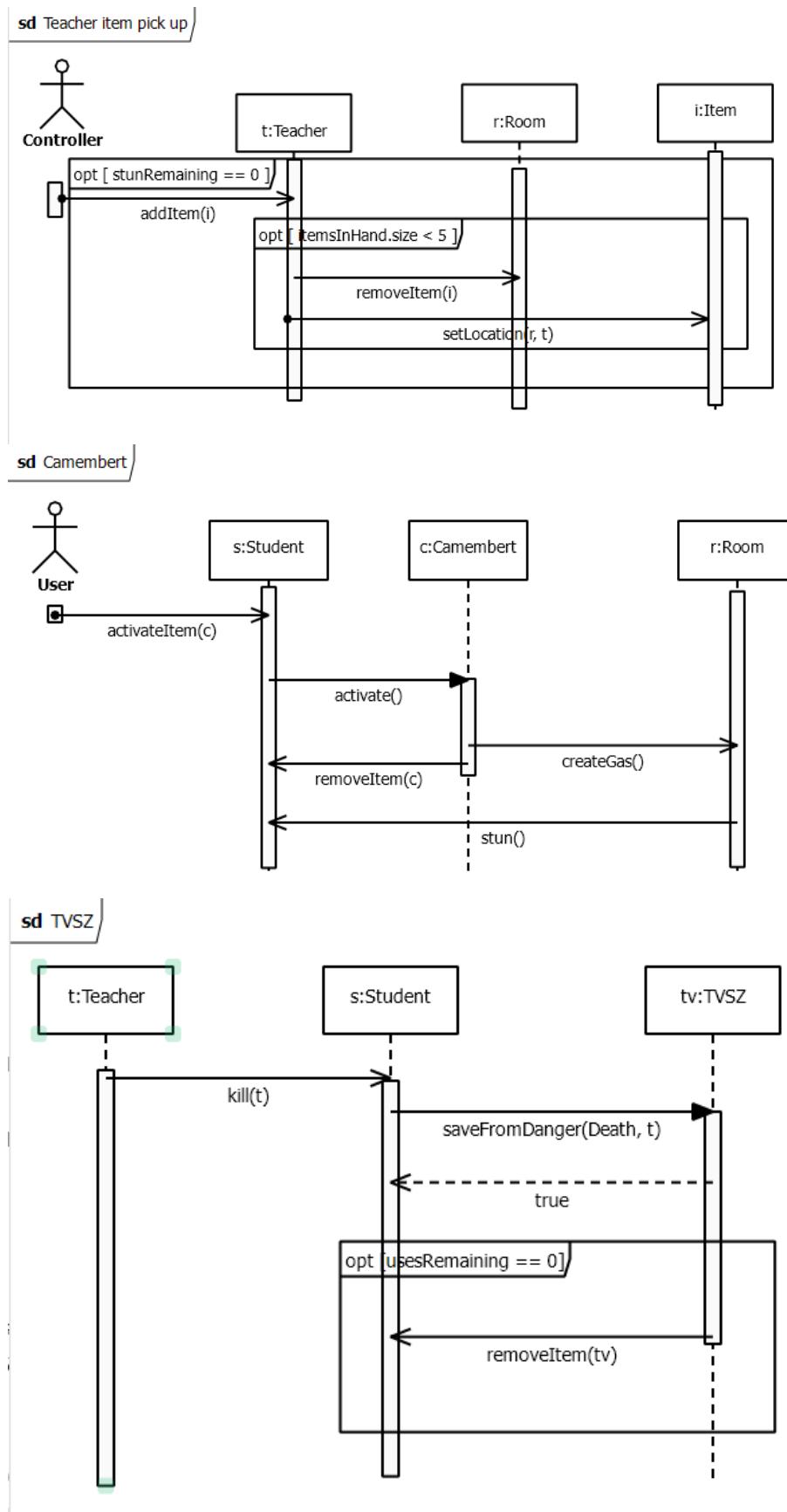


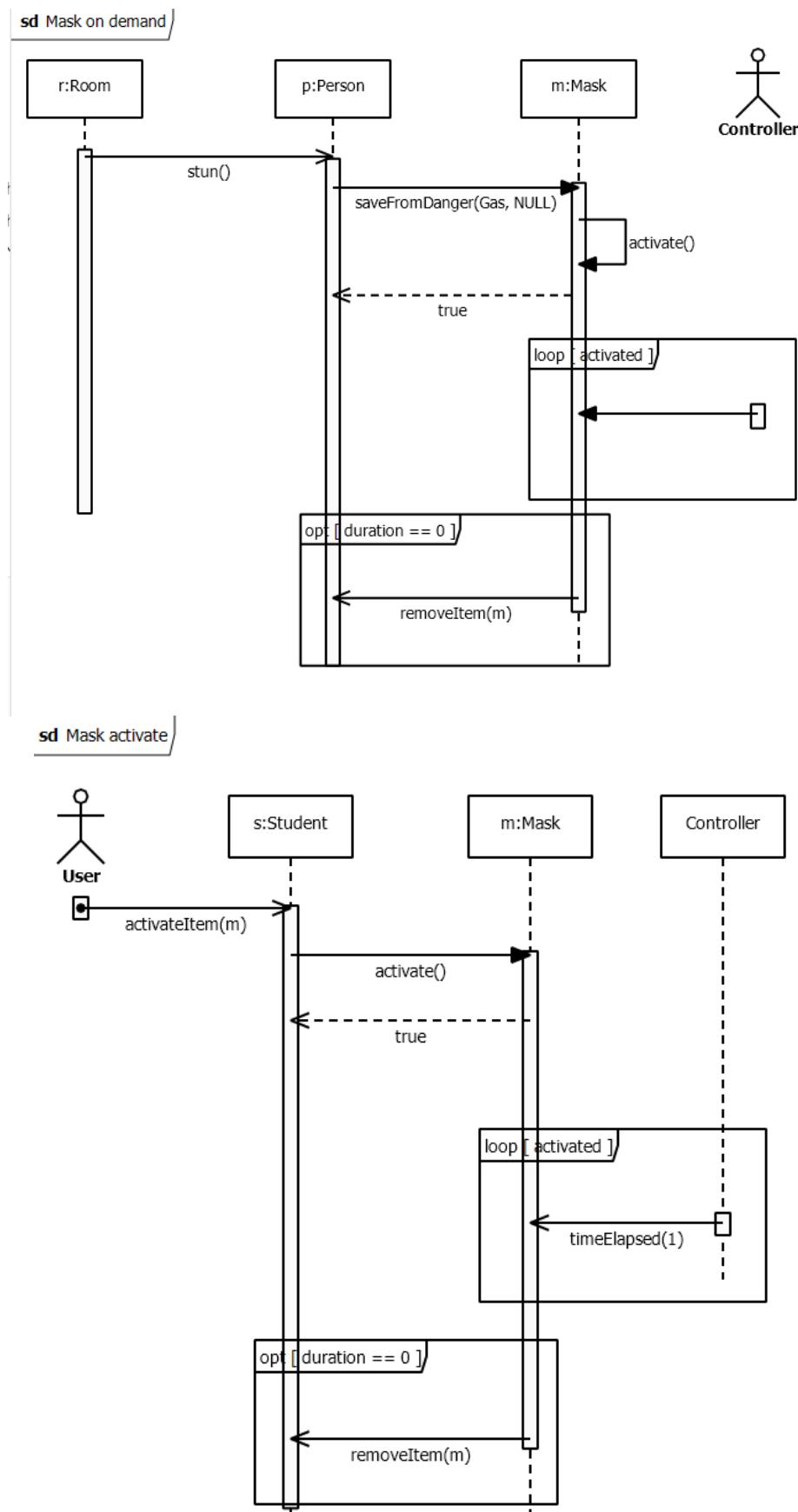
sd Student item pick up

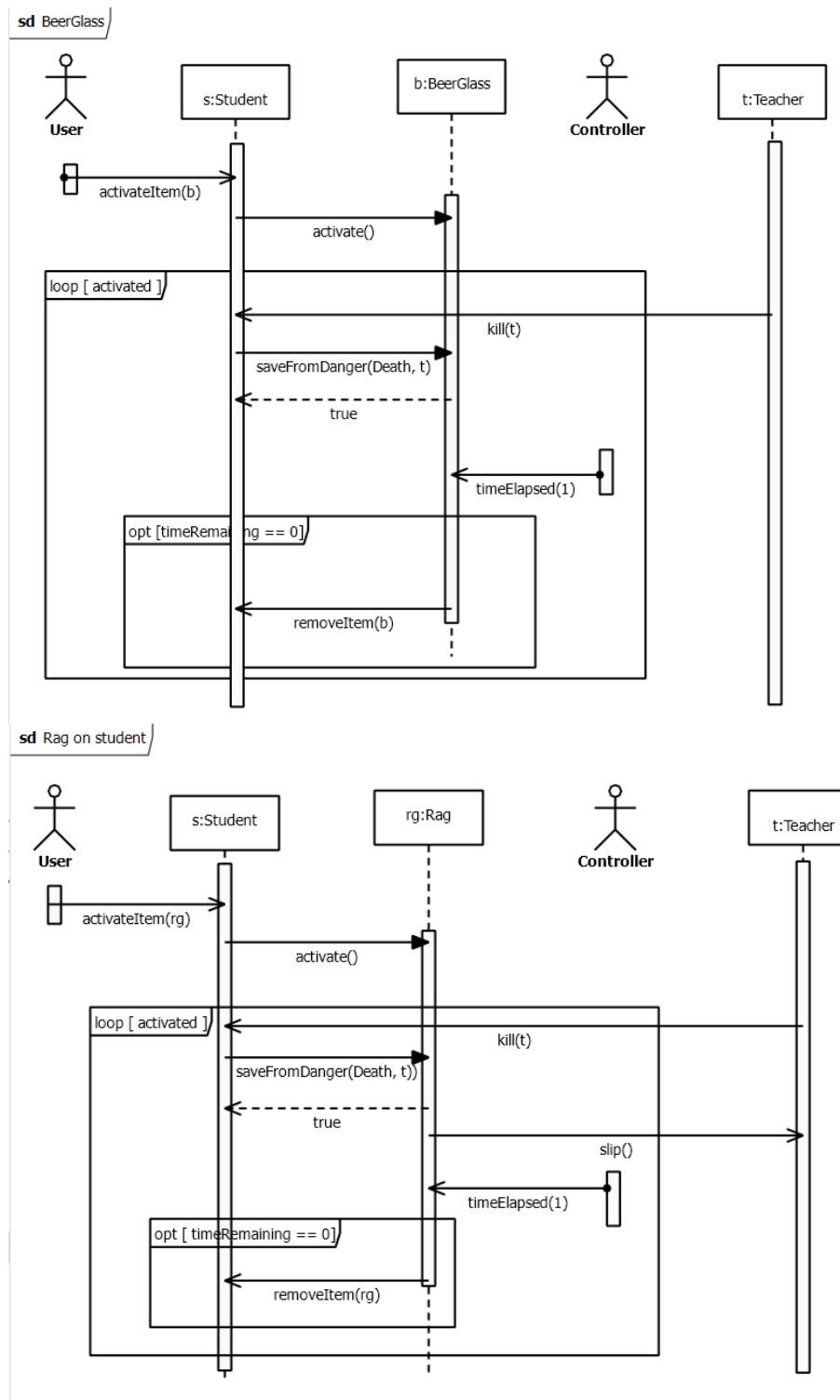


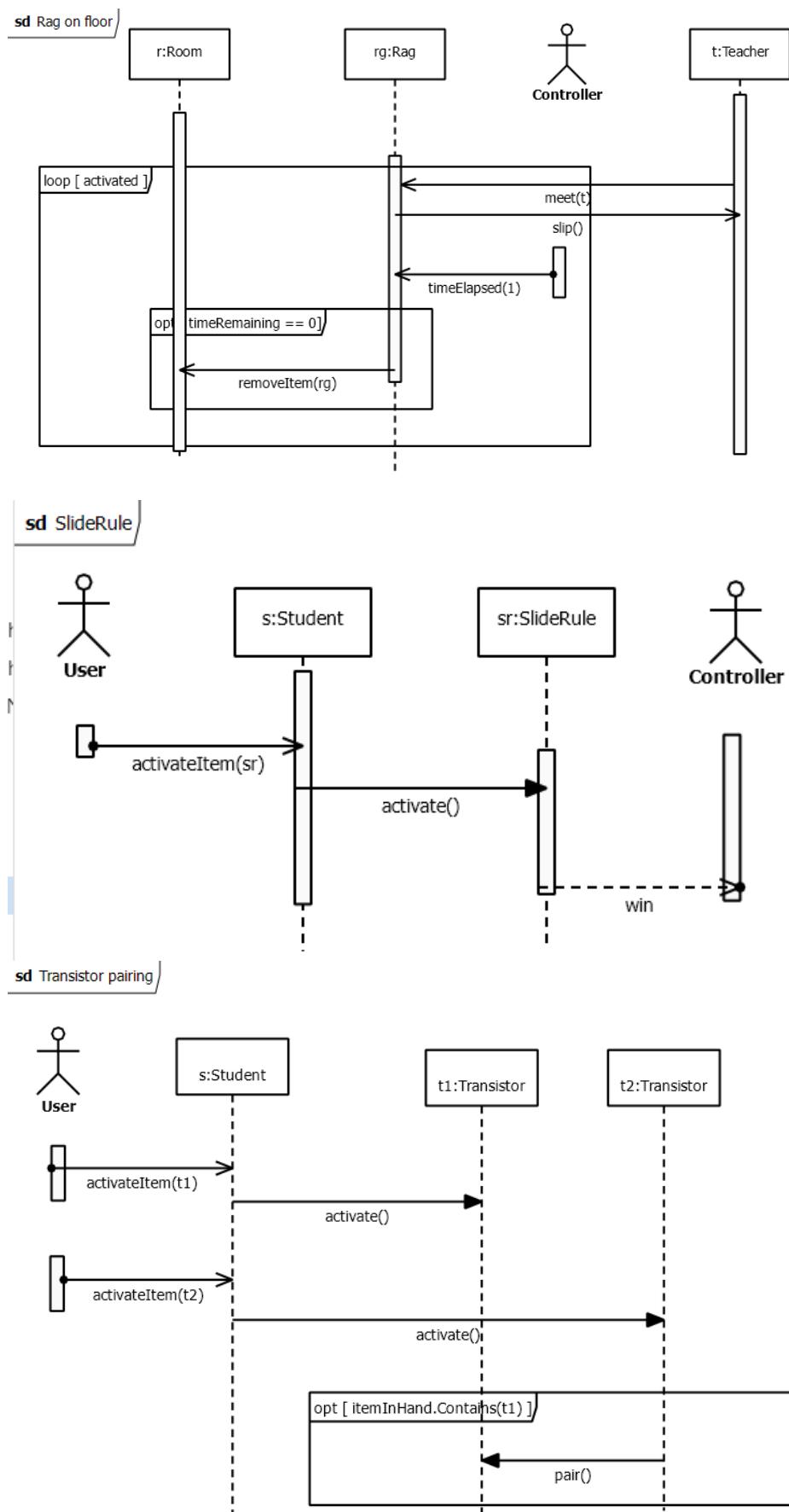
sd Teacher drop item

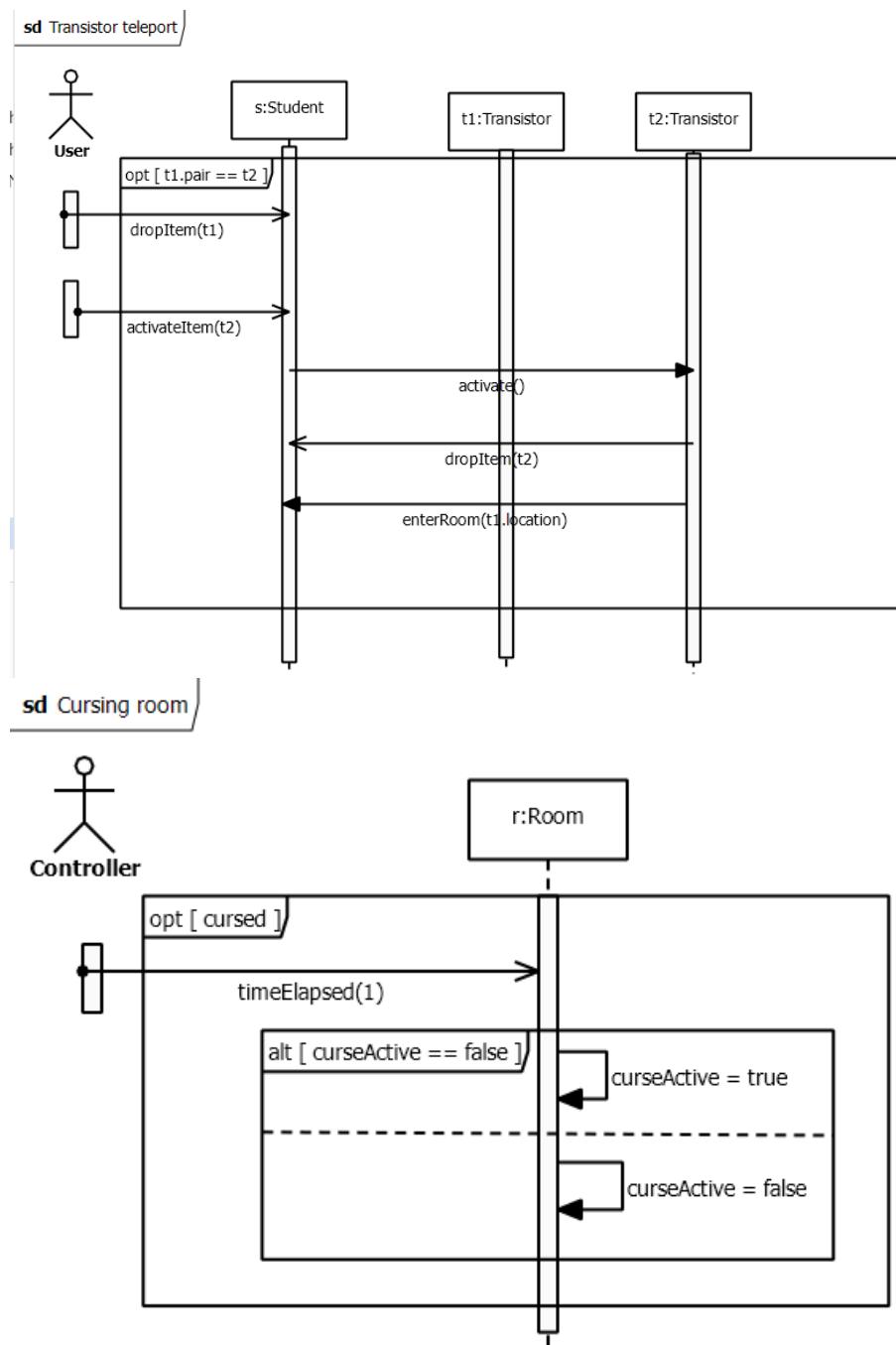


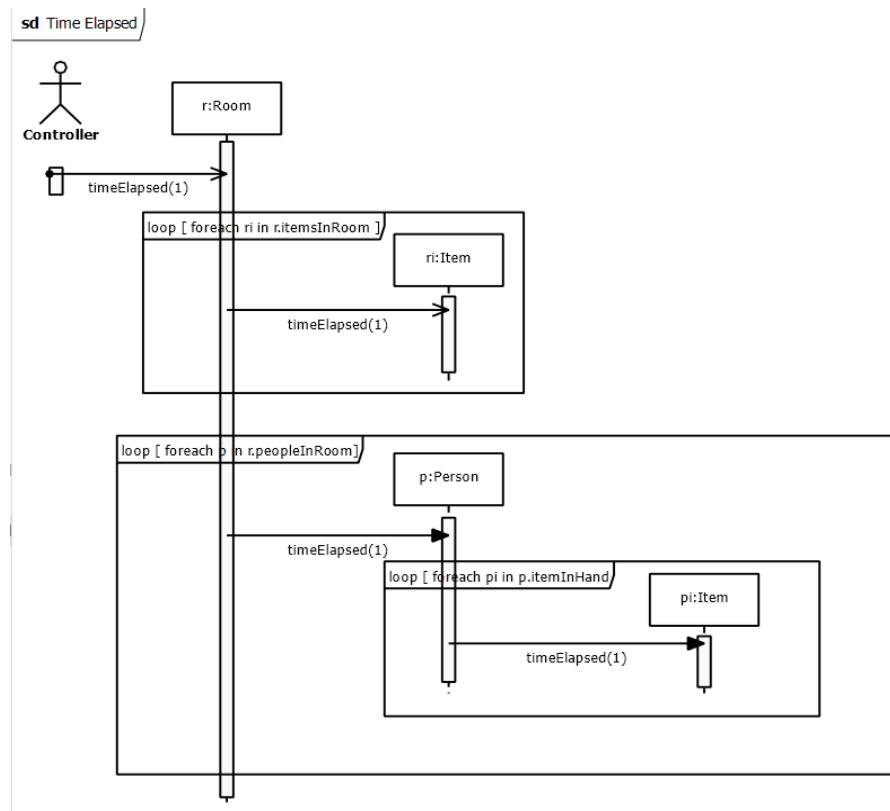


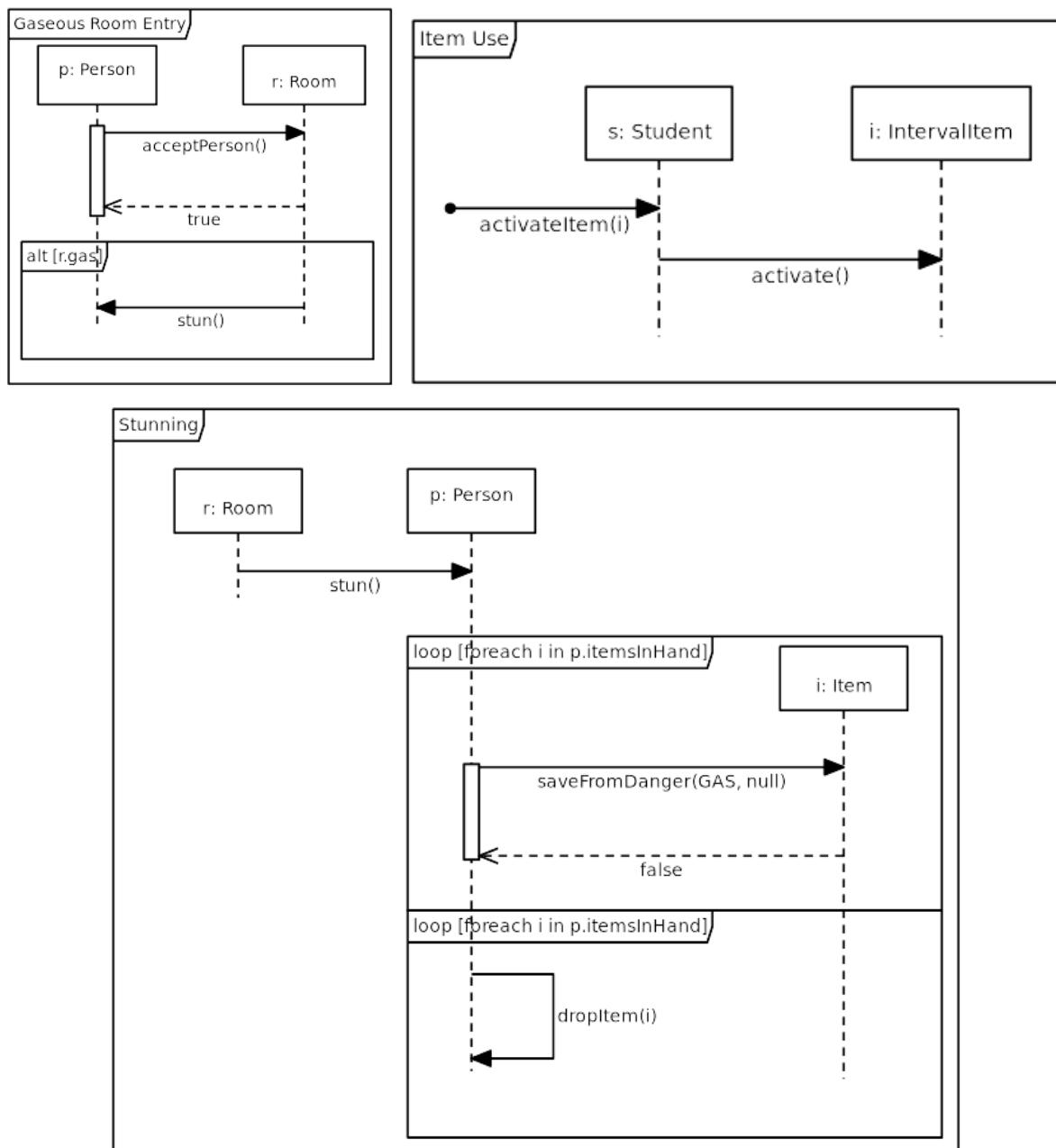


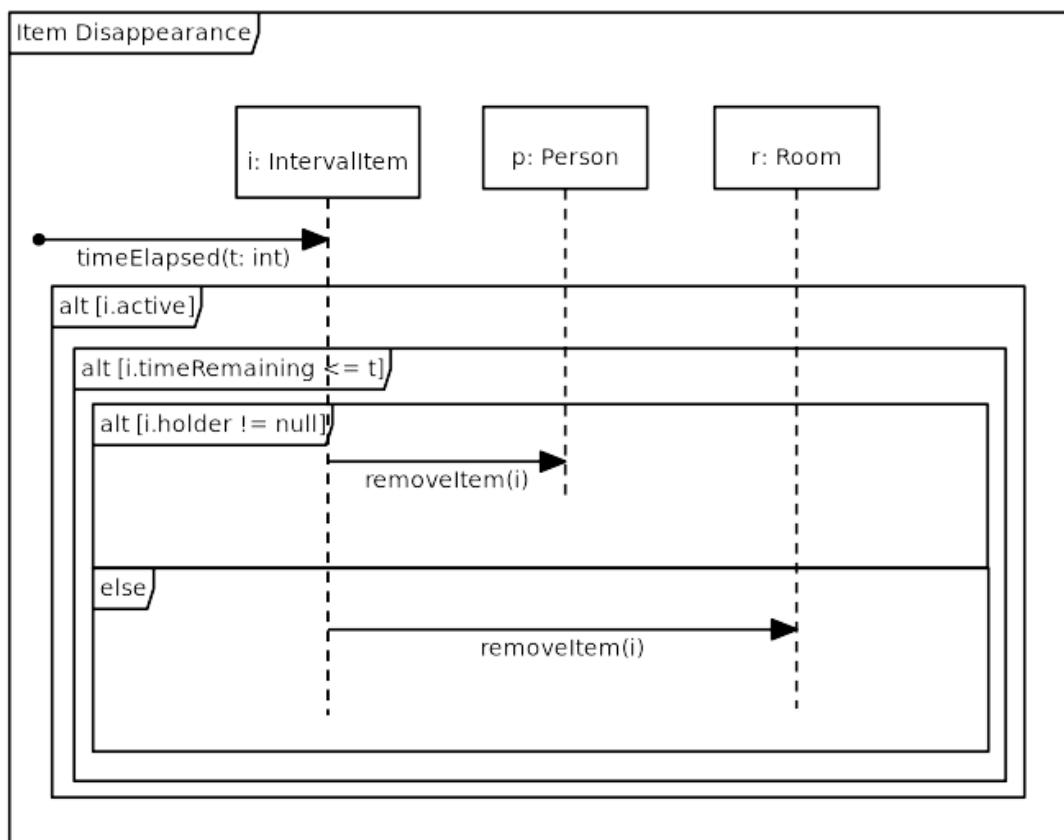
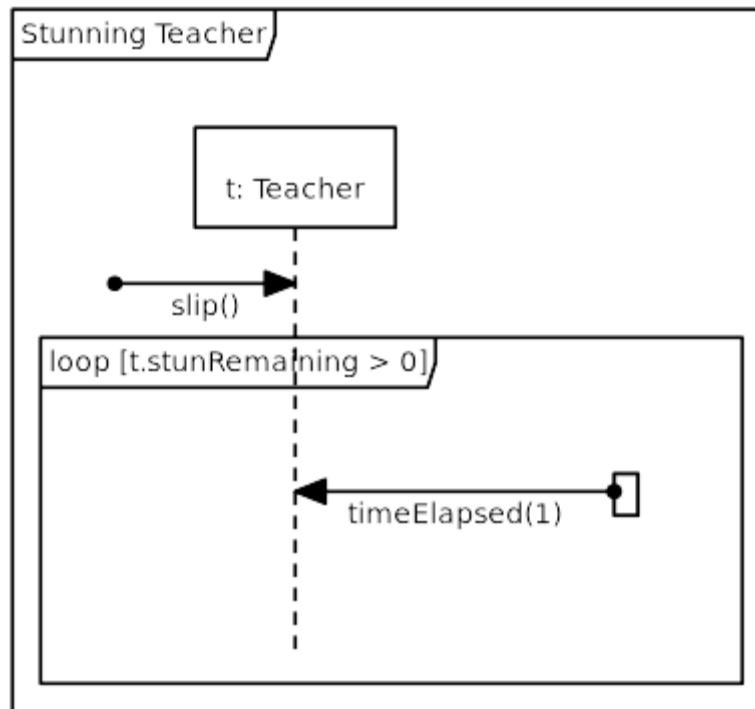




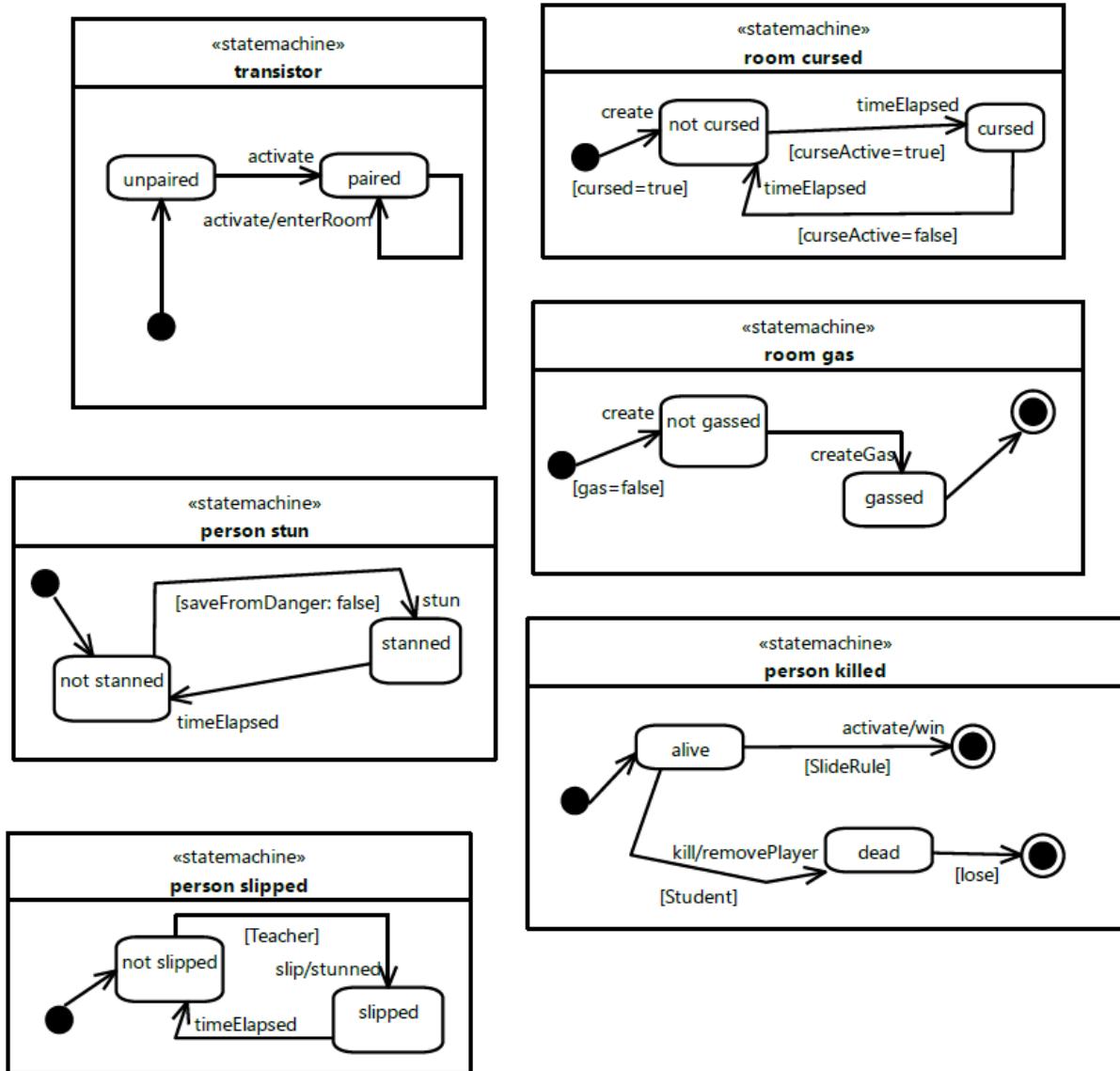








2.5 State-chartok



2.6 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2024. 02. 26. 18:30	1 óra	Tömöri	Feladatkiírás értelmezése, dokumentum létrehozása, objektumkatalógus vázlat készítése.
2024. 02. 26. 20:45	2 óra	Görömbey Sági Tömöri Vizhányó	Értekezlet, Osztálydiagram vázlat
2024. 02. 28. 10:15	1,5 óra	Görömbey Héjja Sági Tömöri Vizhányó	Konzultáció
2024. 02. 28. 18:15	3 óra	Görömbey Héjja Sági Tömöri Vizhányó	Értekezlet
2024. 03. 02. 19:00	3,5 óra	Görömbey Héjja Sági Tömöri Vizhányó	Értekezlet és osztálydiagram elkészítése Görömbey vezetésével, szekvenciadiagramok felsorolása. Döntések: Tömöri elkészíti az osztályleírásokat. Héjja és Sági elkészítik a szekvenciadiagramokat. Vizhányó elkészíti az állapotgépeket. Határidő: 2024. 03. 03. 22:00
2024. 03. 03. 11:00	4 óra	Tömöri	Main diagramhoz tartozó osztályleírások elkészítése.
2024. 03. 03. 14:00	0,5 óra	Vizhányó	State Chartok rajzolása
2024. 03. 03. 15:00	5 óra	Sági	10-25. szekvencia diagram rajzolása (tárgyak)
2024. 03. 03. 18:30	1 óra	Tömöri	Items diagramhoz tartozó, nem megvalósított osztályleírások elkészítése.
2024. 03. 03. 19:45	1,5 óra	Görömbey Sági Tömöri	Értekezlet: UML pontosítás, hiányzó metódusok hozzáadása.
2024. 03. 03. 21:00	4 óra	Vizhányó	1-9. szekvencia diagram rajzolása (mozgás, gyilkolás, szoba split/merge)
2024. 03. 03. 22:00	2,5 óra	Héjja	26-30. szekvencia diagram rajzolása

3. Analízis modell refaktorálása

3.1 Objektum katalógus

3.1.1 Ajtó

A szobák közti kapcsolatot reprezentálja. A szoba rendelkezik a saját ajtajaival, azaz számítja, hogy melyik szobákba tud átmenni.

3.1.2 Camembert

A tárgy a használatakor begázosítja a szobát, amiben van, majd eltűnik.

3.1.3 FFP2-es maszk

A tárgy adott ideig védelmet nyújt a mérgező gázzal szemben. A hatása egyre kevesebb, miután végleg elfogy, eltűnik. Oktatók is használhatják.

3.1.4 Hallgató

Egy játékos karaktere, aki tárgyak felvételével és használatával igyekszik megóvni magát a rá leselkedő bajtól, hogy a szobák valamelyikében megtalálja a Logarlécet. Számítja, hogy milyen tárgyak vannak nála, szükség esetén aktiválja őket. Ismeri a tartózkodási helyét és képes annak megváltoztatására.

3.1.5 Logarléc

A tárgy, melynek megszerzése a győzelem kapuja. Oktatók nem vehetik fel.

3.1.6 Oktató

A játék egy gonosz karaktere, aki a labirintusban keringve igyekszik megakadályozni a hallgatók győzelmét. Képes tárgyfelvételre. Számítja, hogy milyen tárgyak vannak nála, szükség esetén aktiválja őket. Ismeri a tartózkodási helyét és képes annak megváltoztatására.

3.1.7 Söröspohár

Ez a tárgy aktiválás után adott ideig nyújt védelmet annak a hallgatónak, aki aktiválta, majd eltűnik.

3.1.8 Szoba

A karakterek és tárgyak tartózkodási helye, itt tudnak egymással interakcióba lépni (karakter-karakter vagy tárgy-karakter). A szobáknak minden van szomszéduk, lehetővé teszik az átjárást. Különleges esetekben megnehezítik a játékosok feladatát, gázos vagy elátkozott tulajdonságukkal, illetve a szobák egyesülésével és osztódásával. Tárolja, hogy mely karakterek és tárgyak találhatók benne. Felel azért, hogy a befogadóképességénél többen ne tartózkodjanak benne.

3.1.9 Táblatörlő rongy

Ez a tárgy aktiválás után a vele egy szobában lévő oktatókat adott ideig megbénítja, majd eltűnik.

3.1.10 Tranzisztor

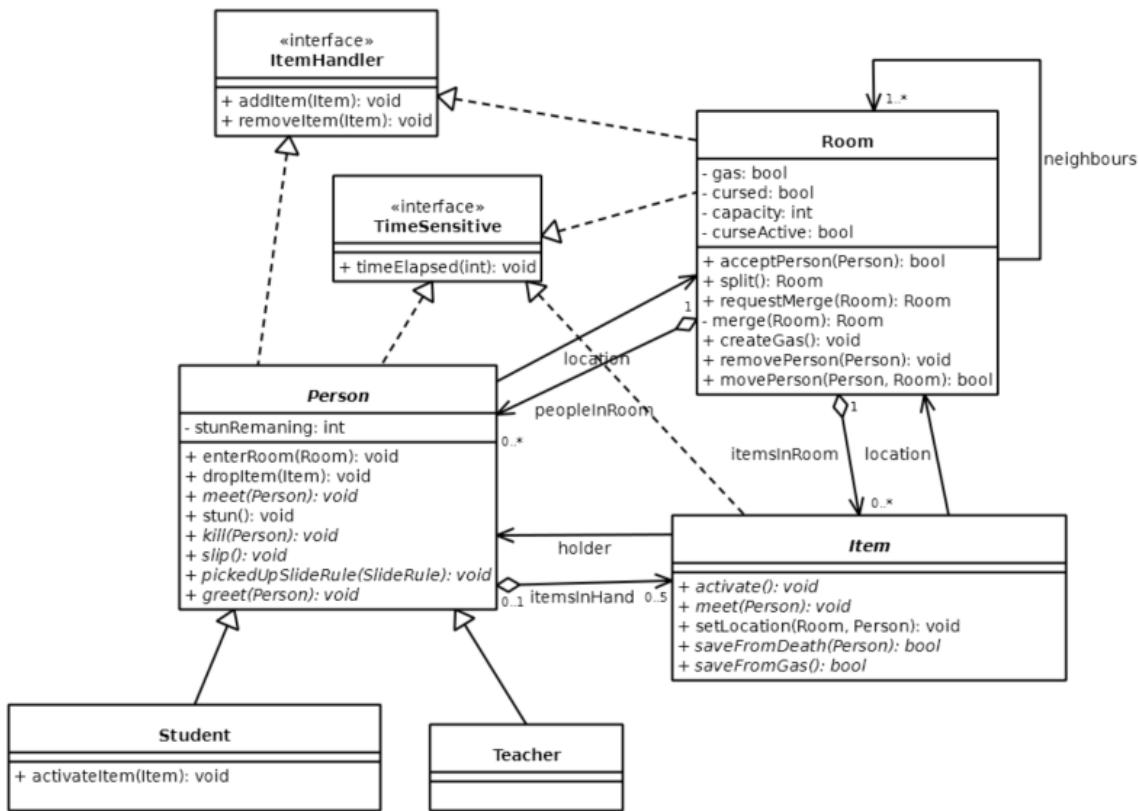
Ennek a tárgynak a használatához a hallgatónak kettő darabra van szüksége. Ha rendelkezik a hallgató két tranzisztorral, akkor azokat összekapcsolhatja. Az összekapcsolás után a játék során

nem lehet összekapcsolni harmadik tranzisztorral. Ha két tranzisztor össze van kapcsolva, akkor azok számítartják egymást, és valamelyik aktiválásával a hallgató átkerül a másik tranzisztor szobájába. Az aktivált tranzisztor az eredeti szobában marad. A tranzisztorok korlátlanul használhatók egymással. A tranzisztor felel a saját tartózkodási helyéért.

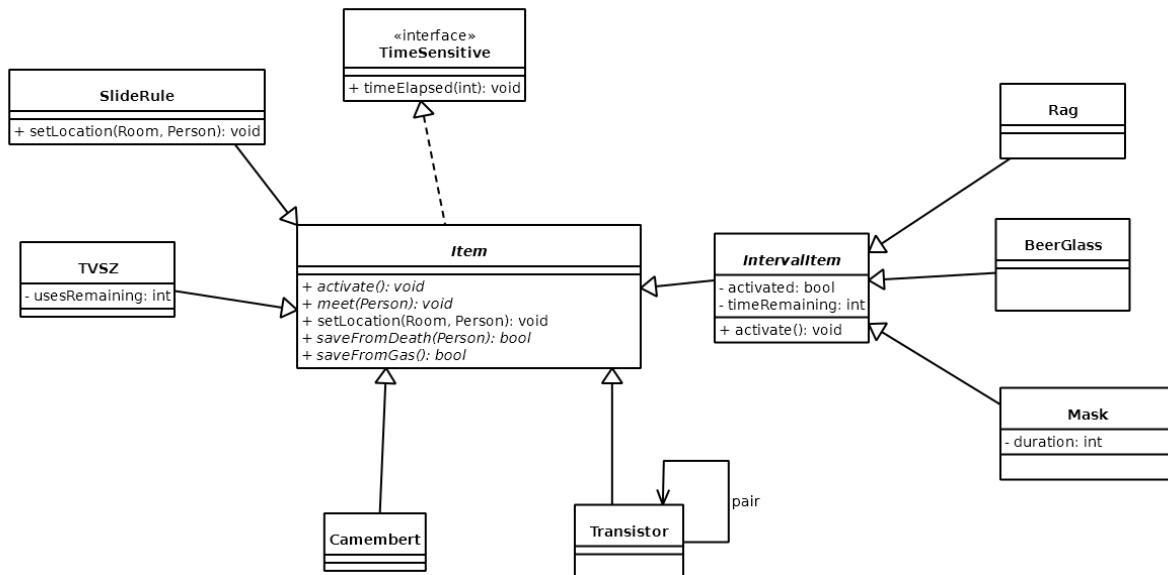
3.1.11 TVSZ

Ez a tárgy három alkalommal használható az oktatók szembeni védelemre, majd eltűnik.

3.2 Statikus struktúra diagramok



1. ábra: Main osztálydiagram



2. ábra: Items osztálydiagram

3.3 Osztályok leírása

3.3.1 BeerGlass

- Felelősség**

A szent söröspoharak adott ideig védelmet nyújtanak az oktatókkal szemben. A hallgató képes aktiválni, ekkor a söröspohár felelőssége, hogy kezelje magát. Amikor elmúlik a hatása a megszűnésétől kezdeményezi a birtokosának.

- Ősosztályok**

Item → *IntervalItem*

- Interfészek**

- *TimeSensitive*

- Attribútumok**

Nem rendelkezik az őséhez képest újabb attribútumokkal.

- Metódusok**

- **void meet(Person p)**: Nem csinál semmit, mert ha földön van nincs kit megvédenie.
- **bool saveFromDeath(Person p)**: Amennyiben aktiválva van a söröspohár, logikai igazzal tér vissza, megvédve a birtokosát. Egyéb esetben logikai hamissal tér vissza.
- **bool saveFromGas()**: A tárgy nem nyújt védelmet a gáz ellen, visszatérési értéke hamis.
- **void timeElapsed(int time)**: Ha aktiválva van a tárgy, akkor a timeRemaining értékét csökkenti time-mal. Ha elérte a 0-t, akkor aktuális birtokosánál kezdeményezi a tárgy megsemmisítését.

3.3.2 Camembert

- **Felelősség**

A Camembert sajt aktiválásakor mérgező gázzal árasztja el a szobát. A sajt felelőssége a szoba elgázosításának és utána a megszűnésének kezdeményezése.

- **Ősosztályok**

Item

- **Interfészek**

- TimeSensitive

- **Attribútumok**

Nem rendelkezik attribútummal.

- **Metódusok**

- **void activate()**: A camembert aktiválása, tartózkodási szobáján meghívja a location.createGas() metódust és elgázosítja azt, majd rögtön igényli birtokosánál az önmegsemmisítést.
- **void meet(Person p)**: Nem csinál semmit, mert ha földön van nincs funkciója.
- **bool saveFromDeath(Person p)**: A tárgy nem nyújt védelmet a kibukás ellen, visszatérési értéke hamis.
- **bool saveFromGas()**: A tárgy nem nyújt védelmet a gáz ellen, visszatérési értéke hamis.
- **void timeElapsed(int time)**: Mivel egyszerhasználatos tárgy, így nem történik vele semmi az idő műlásával.

3.3.3 IntervalItem

- **Felelősség**

Azon tárgyak, melyek adott ideig fejtik ki hatásukat. Felelősségeik tudni, hogy aktiválva vannak-e, illetve, hogy meddig. Amennyiben aktiválva vannak hatással lehetnek a személyekre. Aktiválhatóak. Az IntervalItem osztály absztrakt, nem példányosítható.

- **Ősosztályok**

Item

- **Interfészek**

- TimeSensitive

- **Attribútumok**

- **activated**: Logikai változó. Jelentése: értéke logikai igaz, ha a tárgy aktív, 0 ha még nem.
- **timeRemaining**: Egész szám típus. Jelentése: a tárgy ennyi ideig marad még aktív.

- **Metódusok**

- **void activate()**: Az adott tárgy aktiválása, activated flag bebillentése. Innentől kezdve amíg létezik, képes kifejteni a hatását.

3.3.4 Item

- **Felelősség**

A tárgyak a szereplőket segítő eszközök, melynek használatával különböző előnyökre lehetnek szert. Egy tárgy lehet egy szobában, vagy egy embernél, és a targynak a tudás, hogy hol/kinél van, a rendelkezésére áll. Egy tárgy felelőssége megvédeni egy személyt esetleges veszélyektől, amennyiben képes rá, és a saját élettartamát megfelelően vezérelni. A tárgyak továbbá aktiválhatók, amikor a saját képességeket végrehajthatják és felelnek a tartózkodási helyük/birtokosuk aktuálisan tartásáért is. Az Item egy absztrakt osztály, csak a nem absztrakt leszármazottjai példányosíthatók.

- **Ősosztályok**

Nem rendelkezik ősosztályokkal.

- **Interfészek**

- TimeSensitive

- **Attribútumok**

Nem rendelkezik attribútumokkal.

- **Metódusok**

- **void activate()**: Az adott tárgy aktiválása.
- **void meet(Person p)**: Ha a tárgy egy szobában van a földön, akkor új személy belépése esetén a szoba értesíti a tárgyat.
- **bool saveFromDeath(Person p)**: A metódus visszatérési értéke igaz, ha a tárgy a kibukással (és az esetleges p személlyel) szemben képes védelmet biztosítani és hamis, ha nem.
- **bool saveFromGas()**: A metódus visszatérési értéke igaz, ha a tárgy a mérgező gázzal védelmet biztosít és hamis, ha nem.
- **void setLocation(Room r, Person p)**: Átállítja a tárgy tartózkodási helyét és birtokosát a paraméterként kapottakra.
- **void timeElapsed(int time)**: Az adott ideig ható, aktivált tárgyak állapotát módosítja.

3.3.5 ItemHandler

- **Felelősség**

Az interfész megvalósítása lehetővé teszi, hogy egy tárgyakkal rendelkezni képes objektum új tárgyhoz férjen hozzá, vagy elhasznált tárgyat töröljön ki.

- **Metódusok**

- **void addItem(Item it)**: Az it tárgy hozzáadása az objektum tárgynyilvántartásába.
- **void removeItem(Item it)**: Az it tárgy törlése az objektum tárgynyilvántartásából.

3.3.6 Mask

- **Felelősség**

Az FFP2-es maszkok adott ideig védelmet nyújtanak egy személynek a mérgező gázzal szemben. A hallgató képes manuálisan is aktiválni, de egyébként automatikusan is aktiválódik a személyeknek. A maszk saját felelőssége, hogy amennyiben véget ér a hatása vajon újra használható-e vagy meg kell semmisíteni. Amikor elmúlik a hatása a megszűnésétől kezdeményezi a birtokosának.

- **Ősosztályok**

Item → IntervalItem

- **Interfészek**

- TimeSensitive

- **Attribútumok**

- **duration:** Egész szám típus. Tárolja, hogy a következő aktiváláskor mennyi ideig lesz aktív.

- **Metódusok**

- **void meet(Person p):** Nem csinál semmit, mert ha földön van nincs kit megvédenie.
- **bool saveFromDeath(Person p):** A tárgy nem nyújt védelmet a kibukás ellen, visszatérési értéke hamis.
- **bool saveFromGas():** Amennyiben aktiválva van a maszk, logikai igazzal tér vissza, megvédve a birtokosát. Ha nincs aktiválva, akkor aktiválódik, és szintén logikai igazzal tér vissza.
- **void timeElapsed(int time):** Ha aktiválva van a tárgy, akkor a timeRemaining értékét csökkenti time-mal. Ha elérte a 0-t, akkor adott értékkel csökkenti a durationt és visszabillent az activated flag-et 0-ra. Ha a duration elérte a 0-t akkor aktuális birtokosánál kezdeményezi a tárgy megsemmisítését.

3.3.7 Person

- **Felelősség**

A játék karaktereinek működéséért felelős, főbb adatait tárolja. Tárolja a személy tartózkodási szobáját és tárgyait. A szereplők képesek találkozni, gyilkolni, elcsúsztani (megbénulni), és elkábulni. Képesek tárgyat kezelní: felvenni, eldobni, eltávolítani. Tudnak mozogni a szobák között. A Person osztály absztrakt, csak a leszármazottai példányosíthatók.

- **Ősosztályok**

Nem rendelkezik ősosztályokkal.

- **Interfészek**

- ItemHandler
- TimeSensitive

- **Attribútumok**
 - **itemsInHand**: [0..5] db Item tárolója. A személy rendelkezésére álló tárgyak, amiket sikeresen felvett. Képes eldobni és eltávolítani.
 - **location**: Room referancia. A személy tartózkodási helyét tárolja.
 - **stunRemaining**: Egész szám típus. Amennyiben a személy el van kábítva, ennek az értéke adja meg, hogy még mennyi ideig nem mozoghat.
- **Metódusok**
 - **void addItem(Item it)**: A paraméterben kapott it tárgy felvétele a személyhez, ha a személy nincs elkábulva. Csak akkor veheti fel, ha még elfér az itemsInHand-ben. Ha felvette, akkor a szobából el kell távolítania és a tárgy birtokosát is beállítja.
 - **void dropItem(Item it)**: A paraméterben kapott it tárgy eldobása. Az itemsInHand-ből eltávolítja a tárgyat és hozzáadja a tartózkodási helyének tárgyaihoz.
 - **void enterRoom(Room r)**: A p személy mozgását végrehajtó metódus. Ha a személy nincs elkábulva, továbbítja a jelenlegi szobájának az átlépés igényét. A két szoba felelőssége, hogy a személyt beengedi-e. Amennyiben sikeresen átlép a másik szobába, frissíti a saját és a tárgyainak tartózkodási helyét is.
 - **void greet(Person p)**: A személynek köszönt a másik p személy.
 - **void kill(Person p)**: A személy p által végrehajtott kibuktatását/halálát hajtja végre.
 - **void meet(Person p)**: A személy találkozik p-vel.
 - **void pickedUpSlideRule(SlideRule sr)**: A személy felvette a Logarlécet.
 - **void removeItem(Item it)**: A paraméterként kapott it tárgy törlése a személy kezéből. Akkor hívódik meg, ha a tárgy elhasználódott.
 - **void slip()**: A személy táblatörölő rongy általi megbénulását hajtja végre.
 - **void stun()**: A személy mérgező gáz általi megbénulását hajtja végre. Először is végigkérdezi a tárgyait, hogy képesek-e megóvni őt a mérgező gáz általi veszélytől. Ha legalább egy megvédi, akkor nem történik a személlyel semmi. Ha egy sem védi meg, akkor eldobja az összes tárgyat és adott ideig elkábul, a stunRemaining beállítódik.
 - **void timeElapsed(int time)**: Továbbítja az eltelt időt (time) a nála lévő tárgyaknak. A személy belső mechanizmusában is részt vesz, például elkábulás esetén csökkenti a stunRemaining-et time értékkel.

3.3.8 Rag

- **Felelősség**

A nedves táblatörölő rongy aktivált állapotban adott ideig megbénítja a vele egy helyiségen tartózkodó oktatókat. A hallgató képes aktiválni, ekkor a táblatörölő rongy felelőssége, hogy kezelje magát. Amikor elmúlik a hatása a megszűnésétől kezdeményezi a birtokosának.

- **Ősosztályok**

Item → IntervalItem

- **Interfészek**

- TimeSensitive

- **Attribútumok**

Nem rendelkezik az őséhez képest újabb attribútumokkal.

- **Metódusok**
- **void meet(Person p)**: Ha egy személlyel találkozik és aktiválva van, akkor kezdeményezi a személy megbénítását.
- **bool saveFromDeath(Person p)**: Amennyiben aktiválva van a rongy, logikai igazzal tér vissza, megvédve a birtokosát és p személyre megbénítást kezdeményez. Egyéb esetben logikai hamissal tér vissza.
- **bool saveFromGas()**: A tárgy nem nyújt védelmet a gáz ellen, visszatérési értéke hamis.
- **void timeElapsed(int time)**: Ha aktiválva van a tárgy, akkor a timeRemaining értékét csökkenti time-mal. Ha elérte a 0-t, akkor aktuális birtokosánál kezdeményezi a tárgy megsemmisítését.

3.3.9 Room

- **Felelősség**

Minden szoba számontartja a személyeket és tárgyakat, akik benne tartózkodnak. Ha egy személy belépne a szobába, a szoba felelőssége, hogy csak akkor engedje be, ha befér, valamint a szobák felelnek a személyek kilépéséért is. Amennyiben új személy érkezik a szobába, a szoba mutatja be az új személyt az eddig a szobában tartózkodó személyeknek és tárgyaknak, illetve az új személynek az eddig szobában tartózkodó személyeket. A szoba tudja, hogy melyik szobákkal szomszédos, így a benne lévő személyeknek lehetővé teszik a mozgást. Irányítja a saját viselkedését, az osztódást és egyesülést egy másik szobával, valamint, ha elátkozott, akkor az ajtók eltűnését, ha gázos, akkor azt, hogy az új érkezőket elkábítja.

- **Ősosztályok**

Nem rendelkezik ősosztályokkal.

- **Interfészek**

- ItemHandler
- TimeSensitive

- **Attribútumok**

- **capacity**: Egész szám típus. A szoba maximális befogadóképessége. Új személy szobába lépésekor ez alapján dönt arról, hogy beengedi-e vagy sem.
- **cursed**: Logikai változó. Tárolja, hogy a szoba elátkozott-e vagy sem. Ennek lekérdezésével tudja, hogy az ajtajainak el(ő)tűnését kell-e módosítani
- **curseActive**: Logikai változó. Amennyiben egy szoba elátkozott, ennek segítségével tudja, hogy jelenleg az ajtajai használhatók-e vagy sem.
- **gas**: Logikai változó. Tárolja, hogy a szoba mérgező-e vagy sem. Új ember szobába lépésekor ennek a lekérdezésével tudja a szoba, hogy elkábítja-e a személyt vagy sem. Inicializáláskor, osztódáskor és camembert hatására állítható.
- **itemsInRoom**: [0..*] db Item tárolója. A szobában található tárgyak, ami nincs senki birtokában, a személyek felvehetik.
- **neighbours**: [1..*] db Room tárolója. Az ajtók megvalósítása: ha egy r1 szobából át lehet menni egy r2 szobába, akkor az r1 neighbours-ben megtalálható r2 referencia. Ha fordítva nem igaz egyirányú ajtóról beszélünk. A személyek ezt használva tudnak közlekedni.

- **peopleInRoom:** [0..capacity] db Person tárolója. A szobában található személyek referenciája van itt, a személyek interakcióihoz szükséges.
- **Metódusok**
- **bool acceptPerson(Person p):** A paraméterként kapott személyt engedi be a szobába. Amennyiben a szoba kapacitása kimerült nem engedi be a személyt. A visszatérési értéke a beengedés sikeressége. Ha beengedi a személyt, felel az új személy és a szobában tartózkodó személyek kölcsönös találkozásáért, illetve az új személy és szobában levő tárgyak találkozásáért. Ha a szoba mérgezett, felel a belépő játékos elkábtásáért.
- **void addItem(Item it):** A paraméterben kapott it tárgy szobához adása. És a tárgy tartózkodási helyének beállítása. Történhet a játék működése szerint, vagy egy személy által, amennyiben ő a tárgyat eldobta.
- **void createGas():** Mérgező gázzal tölti fel a szobát, beállítja a gas értékét logikai igazra, majd felel a szobában tartózkodó személyek elkábtásáért.
- **Room merge(Room):** Két szoba egyesítését elvégző belső függvény. Egy r1 szoba hívhatja egy r2 szobának ezt a függvényét, ha r1 beleegyezik az egyesülésbe. Az r2 beleegyezik, ha nincs benne egy személy sem. Ekkor létrehoz egy új szobát, melyben végrehajtja a paraméterként kapott r1 és önmaga egyesítését. Az új szoba kettejük összes szomszédjával rendelkezni fog egymást kivéve, átadja a két szoba összes tárgyát, illetve az új cursed és gas értékek a két szoba értékének logikai VAGY kapcsolatából keletkezik. A tárgyak tartózkodási helyét setLocation()-nel frissíti. A curseActive false lesz, a capacity a két szoba kapacitásairól a nagyobbik lesz. Továbbá a metódus felelőssége, hogy megtörténjen a megfelelő szobáknál a szomszédok listájának frissítése, mely egy másik privát függvényben is történhet. A metódus a létrejött új szobával tér vissza, amennyiben pedig nem egyezik bele az egyesülésbe NULL-t ad át.
- **bool movePerson(Person p, Room r):** Ha a szoba jelenleg nincs aktívan elátkozva, a paraméterként kapott p személy r szobába léptetésének igényét továbbítja r-nek. Az r értesíti ennek sikerességéről és ő is ezzel tér vissza. Ha igazzal tér vissza, akkor eltávolítja a p személyt önmagából. Ha a szoba aktívan elátkozott, rögtön hamissal tér vissza, r értesítése nélkül.
- **void removeItem(Item it):** A paraméterként átadott it tárgy törlése a szobából, amennyiben aktiválás után a szobában hagyák, és lejárt a hatásának időtartama.
- **void removePerson(Person p):** A paraméterként átadott p személy eltávolítása a peopleInRoom-ból.
- **Room requestMerge(Room):** Ezzel a függvénytel kerül indítványozásra a meghívott szobánál, hogy egyesüljön a paraméterben átadott szobával. Amennyiben a kérényezett szoba beleegyezik (nincs benne egy személy sem), meghívja a merge függvényt a paraméterként átadott szobának, és a merge-ben magát adja át paraméterként. Ekkor a merge visszatérési értékével tér vissza, ha viszont nem egyezik bele az egyesülésbe NULL-lal tér vissza.

- **Room split()**: Ezzel a függvényel kerül indítványozásra a meghívott szobánál, hogy kettéosztódjon. Amennyiben tartózkodik benne személy NULL-t ad vissza. Különben pedig létrehoz egy új szobát, aminek neighbours-ei a saját neighbours-ei fele és saját maga, az itemsInRoom a saját itemsInRoom-jainak szintén fele lesz. Az átadott szomszédokat és tárgyakat saját magából eltávolítja, az új szobát beállítja saját szomszédjának is. Az átadott tárgyak location-jét frissíti setLocation()-nel. A gas és cursed értékeiből, ha mindenki logikai igaz volt, akkor az erediben a gas hamis lesz, és az újban lesz a gas igaz. Ha a két értékből nem volt mindenki logikai 1, akkor az új szoba mindenki értéke hamis lesz. Az új szoba capacity-je a régiével egyezik meg.
- **void timeElapsed(int time)**: Továbbítja az eltelt időt (time) a benne lévő személyeknek és tárgyaknak. A szoba belső mechanizmusában is részt vesz, például egy elátkozott szoba ajtajainak el(ő)tűnése az eltelt idő alapján. Mivel idő telt el, a továbbra is szobában tartózkodó tárgyakat összetalálkoztatja minden személlyel és minden személyt kölcsönösen összetalálkoztat egymással.

3.3.10 SlideRule

- **Felelősség**

A Logarléc a hallgatók győzelmének kulcsa. Amint egy hallgató felvétel után aktiválja, győznek a hallgatók. Felelőssége, hogy aktiváláskor a játék véget érjen, és hogy oktató ne tudja felvenni.

- **Ősosztályok**

Item

- **Interfészek**

- TimeSensitive

- **Attribútumok**

Nem rendelkezik attribútummal.

- **Metódusok**

- **void activate()**: Nem történik semmi.
- **void meet(Person p)**: Nem csinál semmit, mert ha földön van nincs funkciója.
- **bool saveFromDeath(Person p)**: A tárgy nem nyújt védelmet a kibukás ellen, visszatérési értéke hamis.
- **bool saveFromGas()**: A tárgy nem nyújt védelmet a gáz ellen, visszatérési értéke hamis.
- **void timeElapsed(int time)**: Mivel egyszerhasználatos tárgy, így nem történik vele semmi az idő műlásával.

3.3.11 Student

- **Felelősség**

A hallgatókat reprezentáló osztály. A játékosok által elvégezhető funkciókat valósítja meg. Felelőssége, hogy a hallgató megfelelően használhassa a tárgyakat a védelem érdekében, és hogy akkor haljon meg vagy kábuljon el, amikor szükséges.

- **Ősosztályok**

A Person absztrakt osztály leszármazottja.

- **Interfészek**

- ItemHandler
- TimeSensitive

- **Attribútumok**

Nem rendelkezik újabb attribútumokkal.

- **Metódusok** (az osztály új, vagy felüldefiniált metódusai)

- **void activateItem(Item it):** A paraméterben kapott it tárgy használata.
- **void greet(Person p):** Nem csinál semmit a hallgató.
- **void kill(Person p):** A hallgató megpróbálja megvédeni magát a haláltól azzal, hogy az összes tárgyat megpróbálja felhasználni az oktató elleni védelemre. Amint az egyik tárgy megvédi a hallgatót, a metódus véget ér és a hallgató ép marad. Ha a hallgató meghal, az ő felelőssége törölni magát a tartózkodási helyéről. A p paraméter az a személy, aki megöli a személyt, ezt adja tovább a tárgyaknak.
- **void meet(Person p):** Köszön p-nek.
- **void pickedUpSlideRule(SlideRule sr):** Nem történik semmi.
- **void slip():** Nem történik semmi. A hallgató biztonságban marad, mert a táblatörlő rongy ót nem veszélyezteti.

3.3.12 Teacher

- **Felelősség**

Az oktatókat reprezentáló osztály. Felelőssége, hogy az oktatók is mozogjanak, és képes legyen fenyegetni a hallgatókat. A személyek alapvető képességein túl a lebénulás, gyilkolás és FFP2-es maszk használatára van lehetősége.

- **Ősosztályok**

A Person absztrakt osztály leszármazottja.

- **Interfészek**

- ItemHandler
- TimeSensitive

- **Attribútumok**

Nem rendelkezik újabb attribútumokkal.

- **Metódusok** (az ősosztály absztrakt metódusainak definiálása)

- **void greet(Person p):** Az oktató minden p személyt aki köszön neki megpróbál meggyilkolni, amennyiben nincs lebénulva.
- **void kill(Person p):** Nem történik semmi, ugyanis oktató nem halhat meg.
- **void meet(Person p):** Az oktató minden p személyt akivel találkozik megpróbál meggyilkolni, amennyiben nincs lebénulva.
- **void pickedUpSlideRule(SlideRule sr):** Az oktató kidobja a Logarléket.

- **void slip()**: Az oktató a táblatörölő rongy hatására lebénül. A stunRemaining adott értékre lesz állítva, és amíg nem lesz nulla, nem képes gyilkolni, tárgyat felvenni és mozogni.

3.3.13 TimeSensitive

- **Felelősség**

Ezen az interfészen keresztül értesülnek a játék objektumai az idő teléséről.

- **Metódusok**

- **void timeElapsed(int time)**: Ezen keresztül értesülnek az interfész megvalósító objektumok azzal, hogy a játékban time idő eltelt.

3.3.14 Transistor

- **Felelősség**

A tranzisztorok lehetőséget nyújtanak a hallgatóknak nem szomszédos szobák közti gyorsutazásra. Ahhoz, hogy egy hallgató ezzel élhessen, két tranzisztorra is szüksége van. A tranzisztorok felelőssége, hogy össze tudjanak kapcsolódni egymással, amennyiben minden kettő a hallgatónál van. Az összekapcsolt tranzisztorok számítartják egymást is, így biztosítva a kapcsolatot a két tárgy tartózkodási szobái között. A tranzisztorok örökifjűk, korlátlanul használhatók a párjukkal. Ha két tranzisztor egyszer összekapcsolódott, már nem tudnak szétkapcsolódni. Kettőnél több tranzisztor nem csatlakozhat össze.

- **Ősosztályok**

Item

- **Interfészek**

- TimeSensitive

- **Attribútumok**

- **pair**: Transistor referencia. Ha nincs összekapcsolva NULL, ha igen, akkor a párja.

- **Metódusok**

- **void activate()**: Két tranzisztor összekapcsolását illetve azok használatát is lehetővé tevő metódus. Amennyiben egy t1 tranzisztort úgy aktiválunk, hogy a hallgatónál még nincs másik aktivált tranzisztor és a t1.pair NULL, akkor a t1 párosítási szándékát eltároljuk. Amennyiben egy t2 tranzisztort úgy aktiválunk, hogy a t2.pair értéke NULL, de a hallgató aktuálisan is rendelkezik egy már párosítási szándékra bejegyzett t1 tranzisztorral, akkor a t1-et és a t2-t összekapcsolja, és kiveszi őket a párosítási jegyzékből. Amennyiben egy t1 tranzisztort úgy aktiválunk, hogy a t1.pair értéke nem NULL, akkor t1-et eldobja a játékos az eredeti szobájában és a t1 birtokosát áltépteti az enterRoom() metódus segítségével, t2 tartózkodási helyébe, amit paraméterként ad át a személy metódusába.
- **void meet(Person p)**: Ilyenkor nem kell semmit se csinálja.
- **bool saveFromDeath(Person p)**: A tárgy nem nyújt védelmet a kibukás ellen, visszatérési értéke hamis.
- **bool saveFromGas()**: A tárgy nem nyújt védelmet a gáz ellen, visszatérési értéke hamis.

- **void timeElapsed(int time):** Direktbe nem történik vele semmi az idő műlásával, de a tartózkodási helye és tulajdonosa frissítődik.

3.3.15 TVSZ

- **Felelősség**

A TVSZ denevérbőrre írt példánya veszély esetén megvédi az oktatótól az őt hordozó hallgatót. Felelőssége, hogy veszély esetén, ha rá kerül a sor automatikusan aktiválódjon. Továbbá, ha többször nem használható kezdeményezi a megsemmisítését birtokosánál.

- **Ősosztályok**

Item

- **Interfészek**

- TimeSensitive

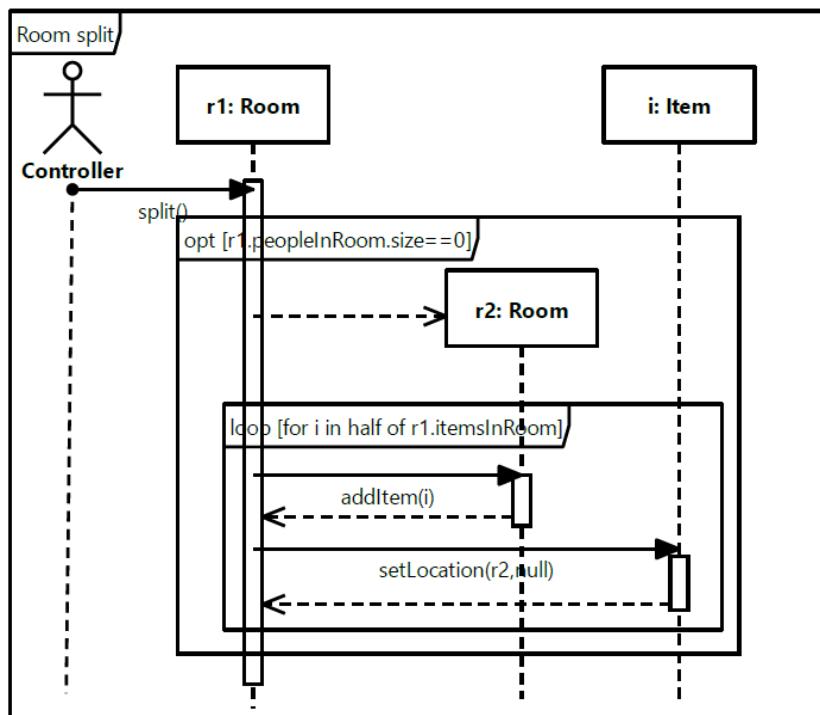
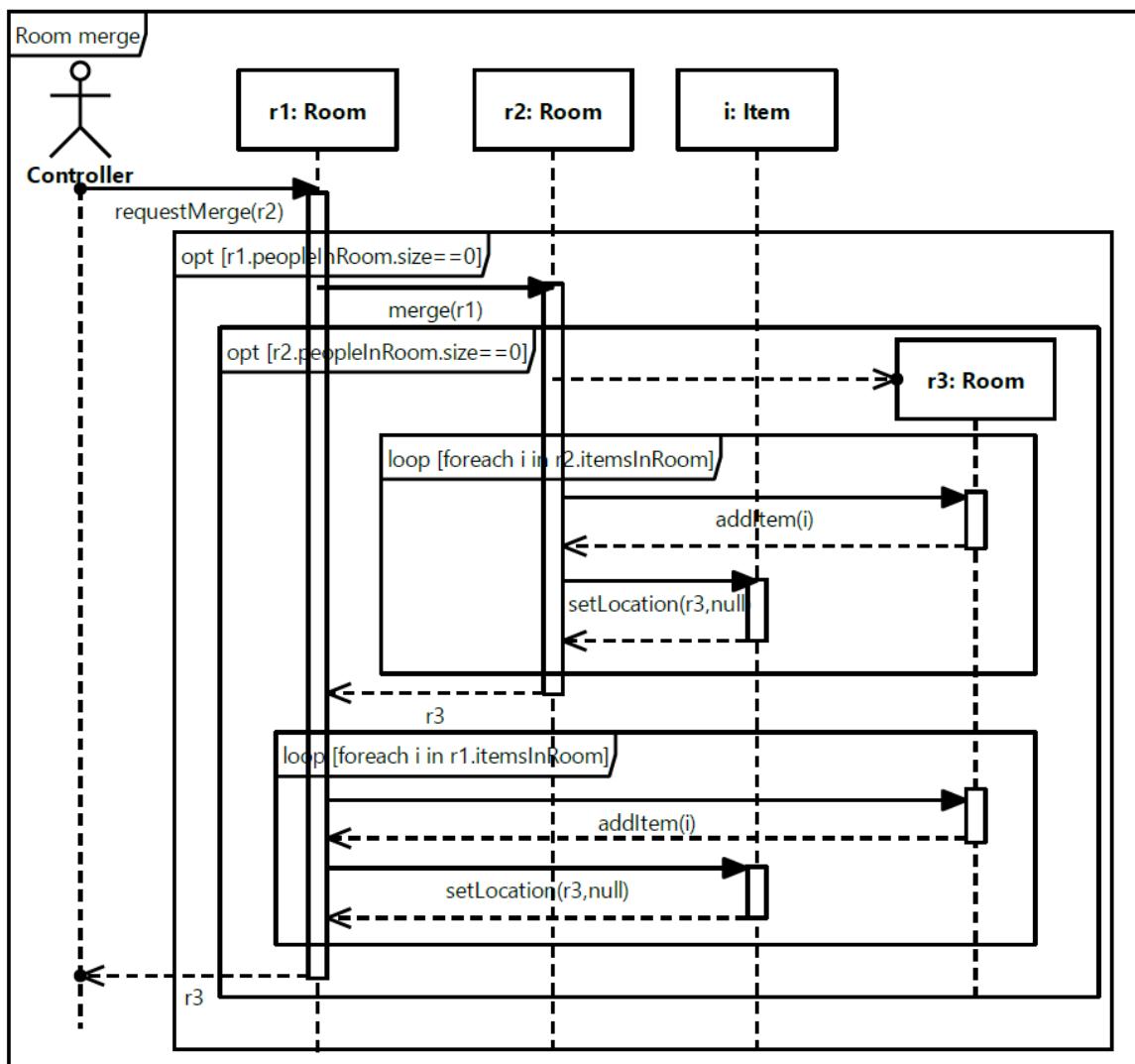
- **Attribútumok**

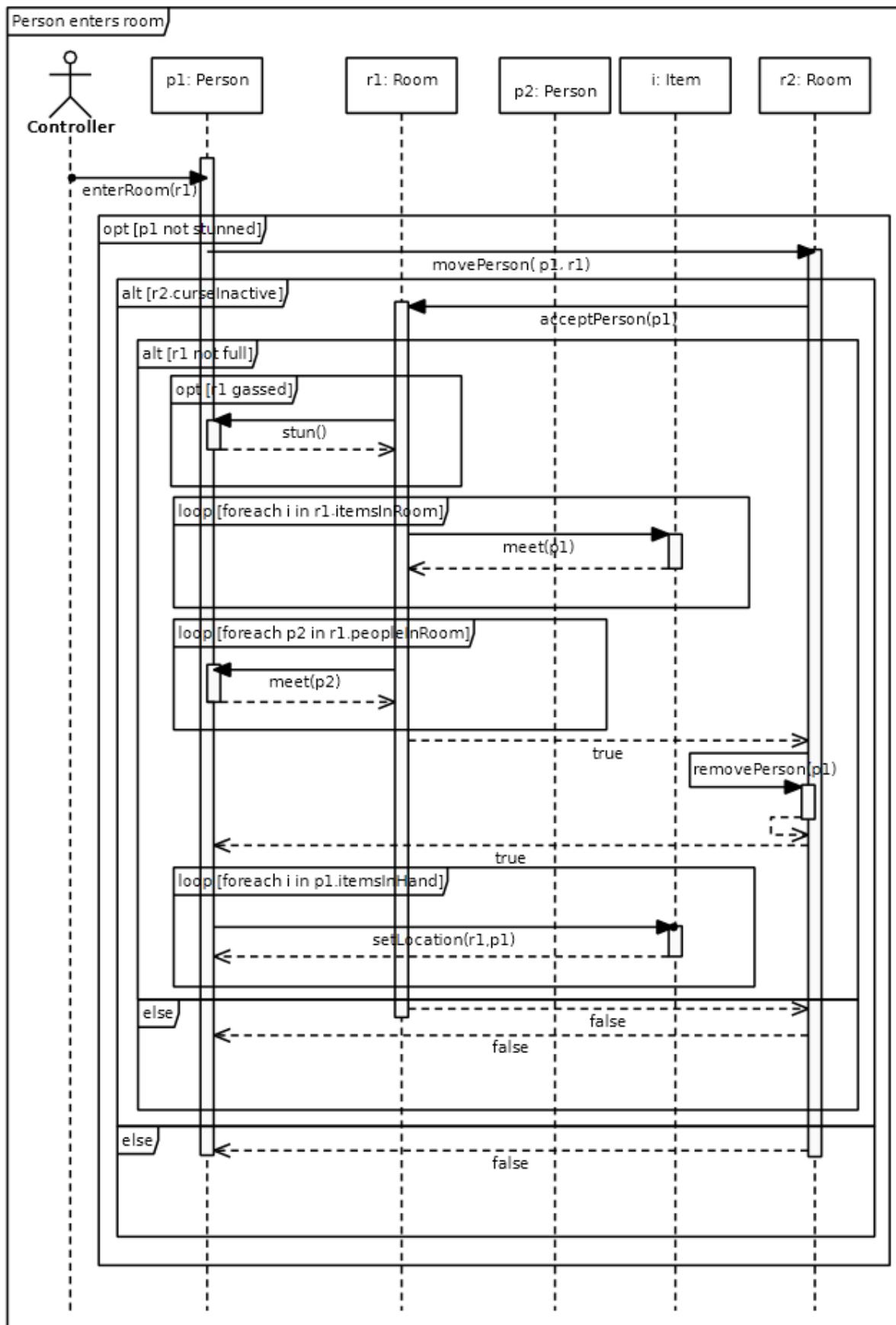
- **usesRemaining:** Egész szám típus. Tárolja, hogy a tárgy még hány alkalommal tud védelmet biztosítani egy oktatóval szemben.

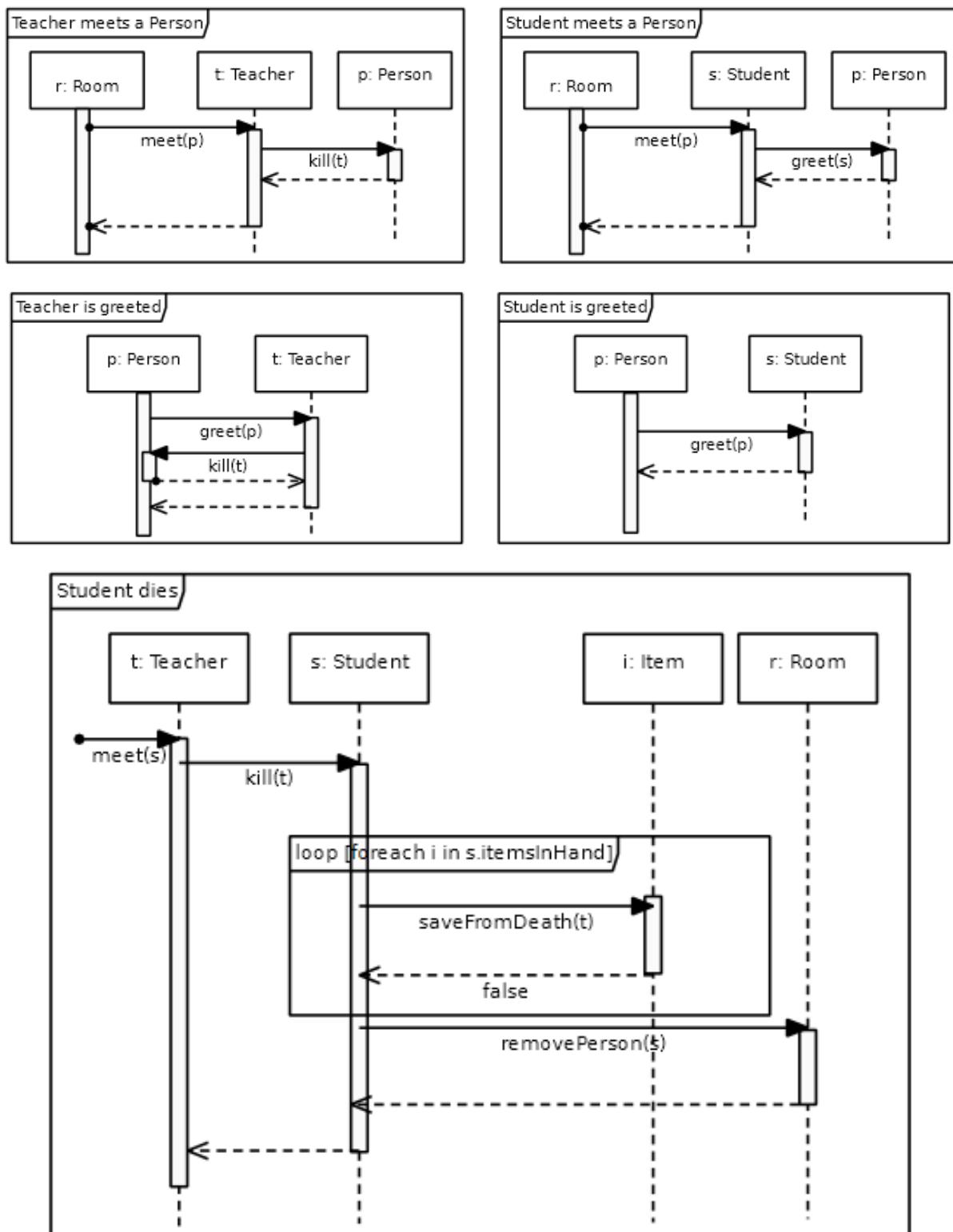
- **Metódusok**

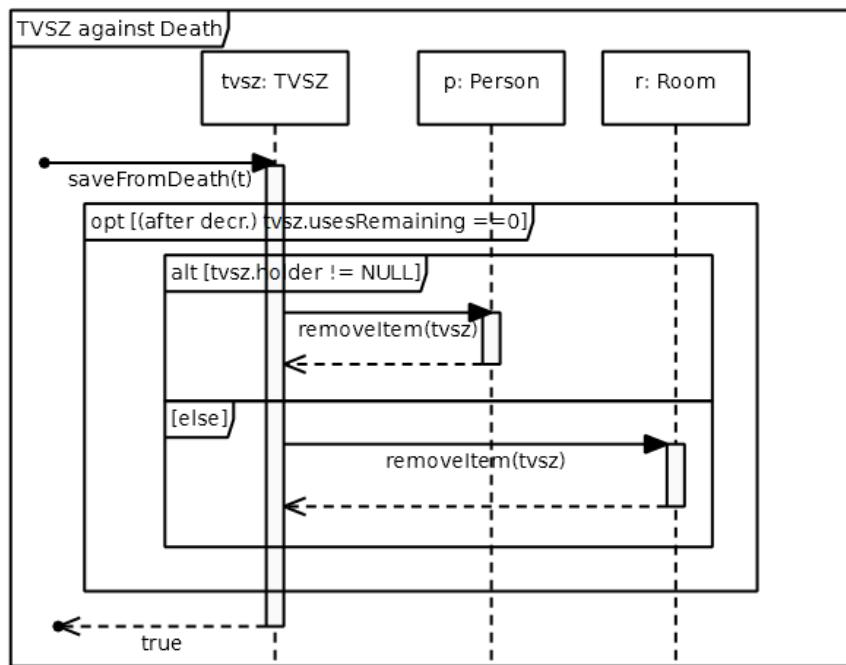
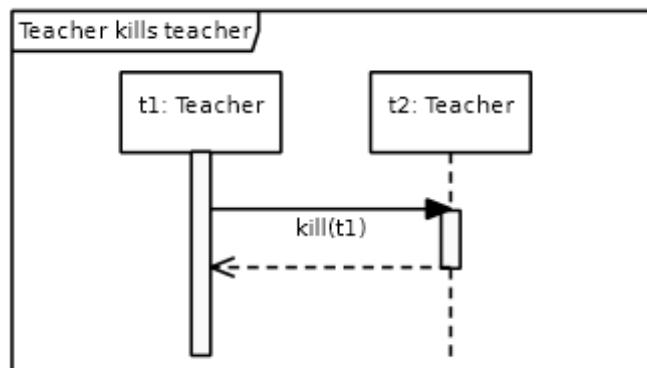
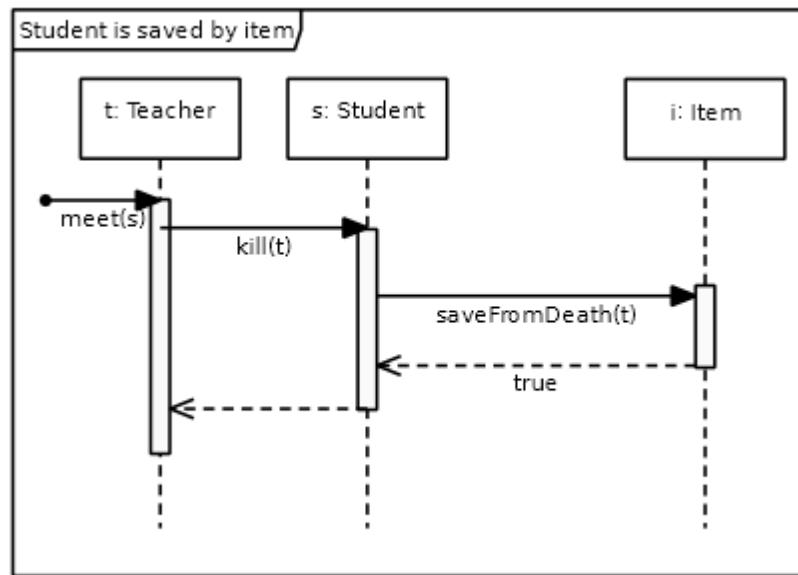
- **void activate():** A TVSZ manuális aktiválása nem lehetséges, így ekkor nem történik semmi.
- **void meet(Person p):** Nem csinál semmit, mert ha földön van nincs funkciója.
- **bool saveFromDeath(Person p):** A tárgy megvédve a birtokosát és csökkenti a usesRemaining-et 1-gyel. Amennyiben a változó értéke 0 lesz megsemmisül. Logikai igazzal tér vissza.
- **bool saveFromGas():** A tárgy nem nyújt védelmet a gáz ellen, visszatérési értéke hamis.
- **void timeElapsed(int time):** Mivel egyszerhasználatos tárgy, így nem történik vele semmi az idő műlásával.

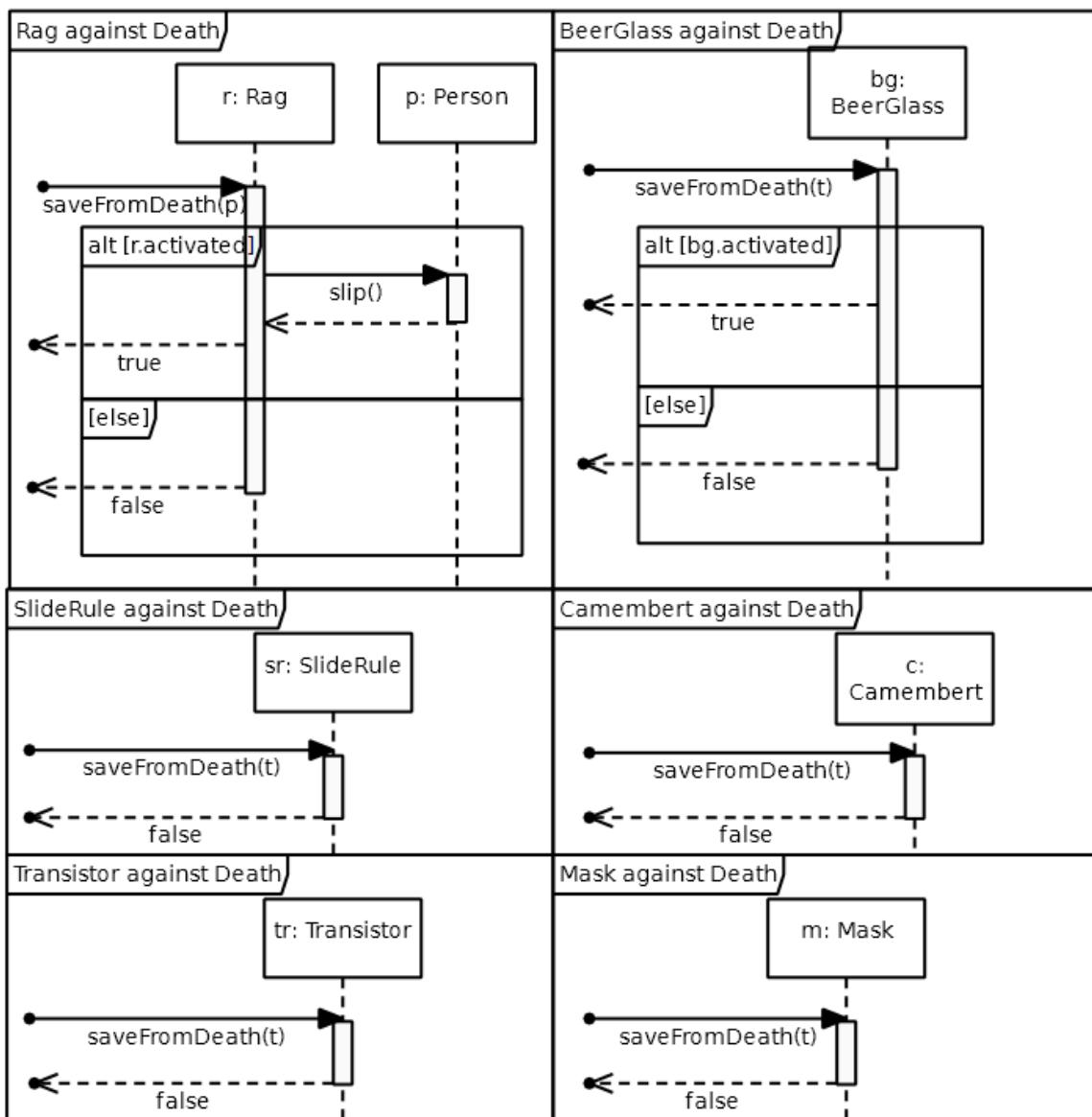
3.4 Szekvencia diagramok

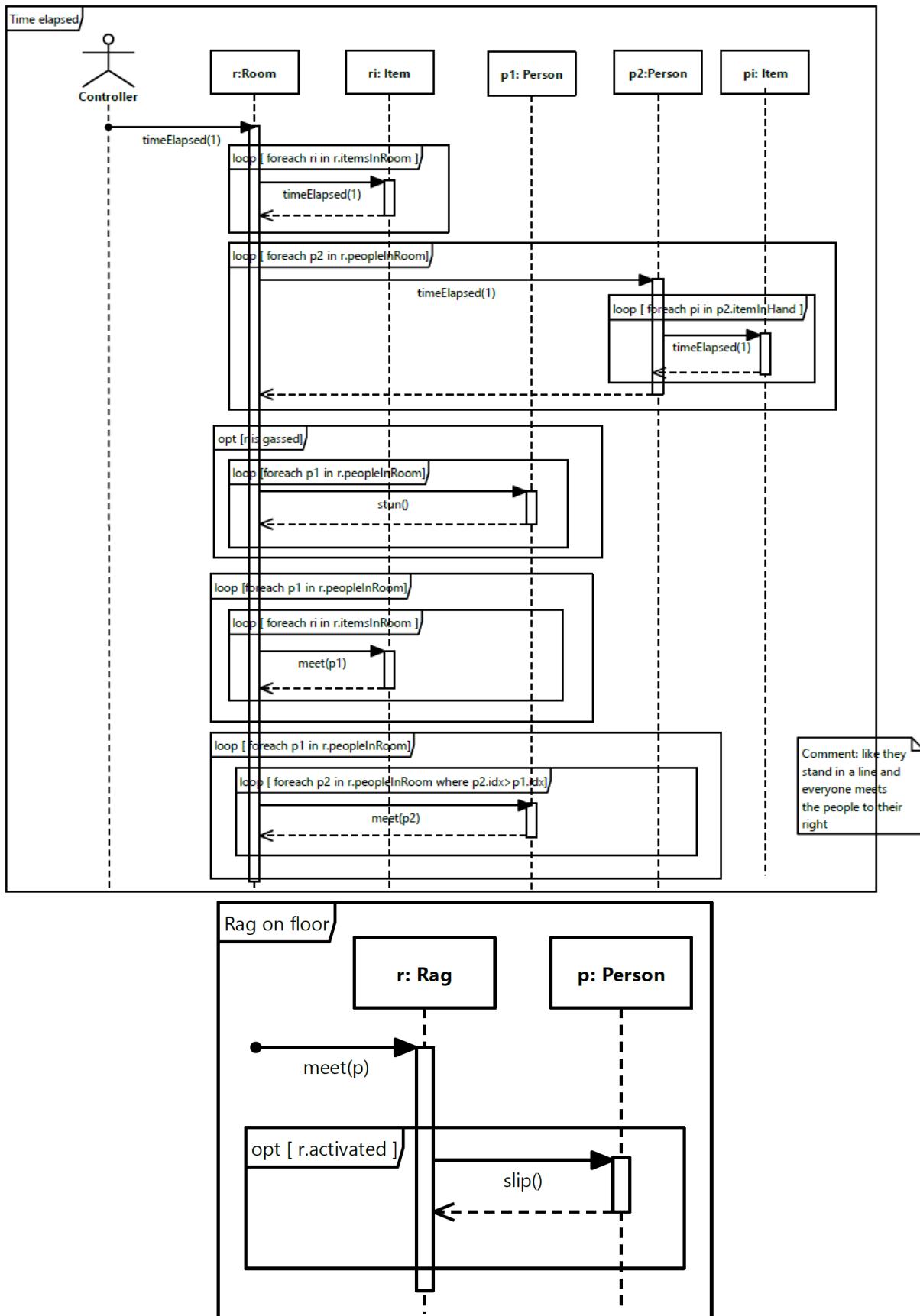


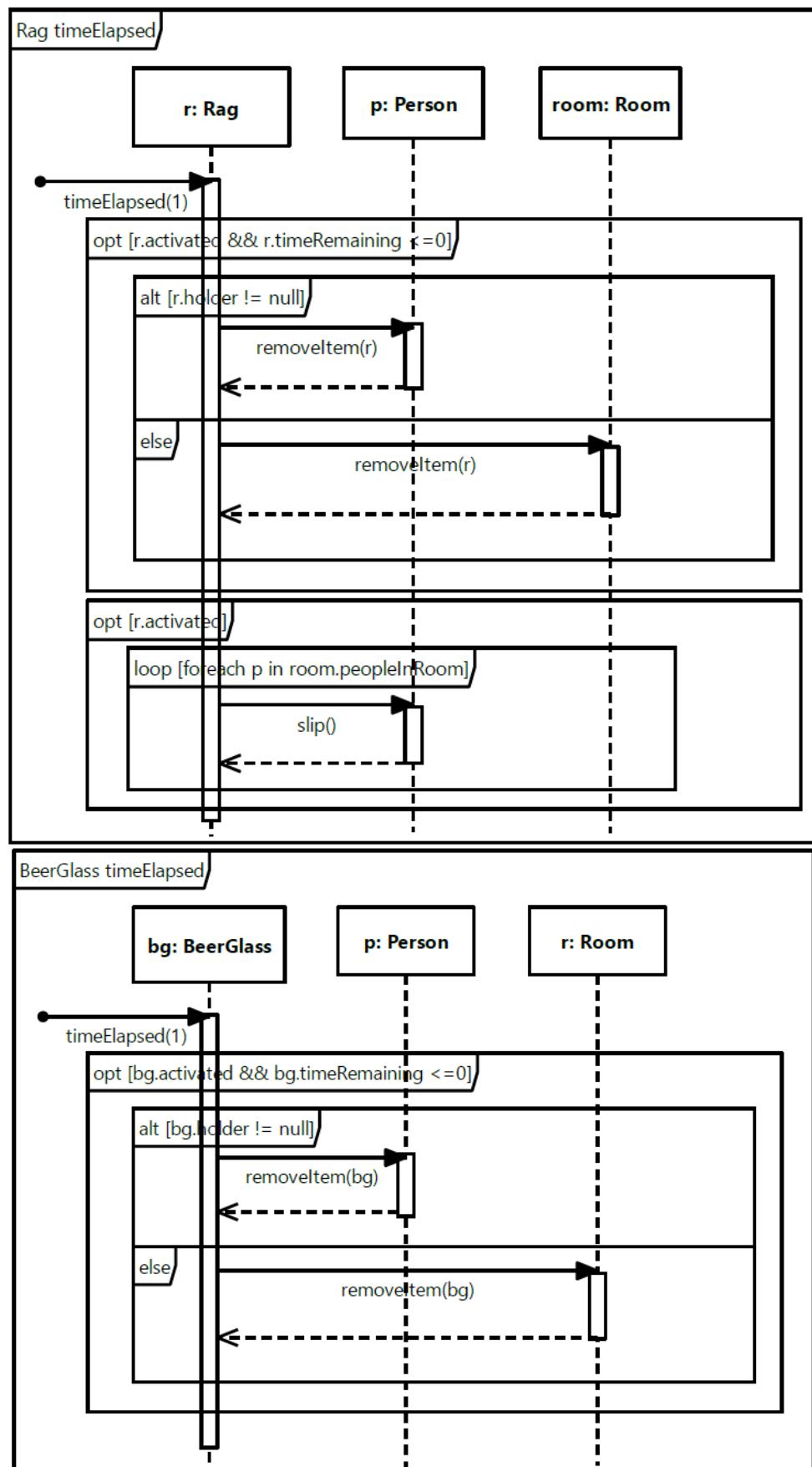


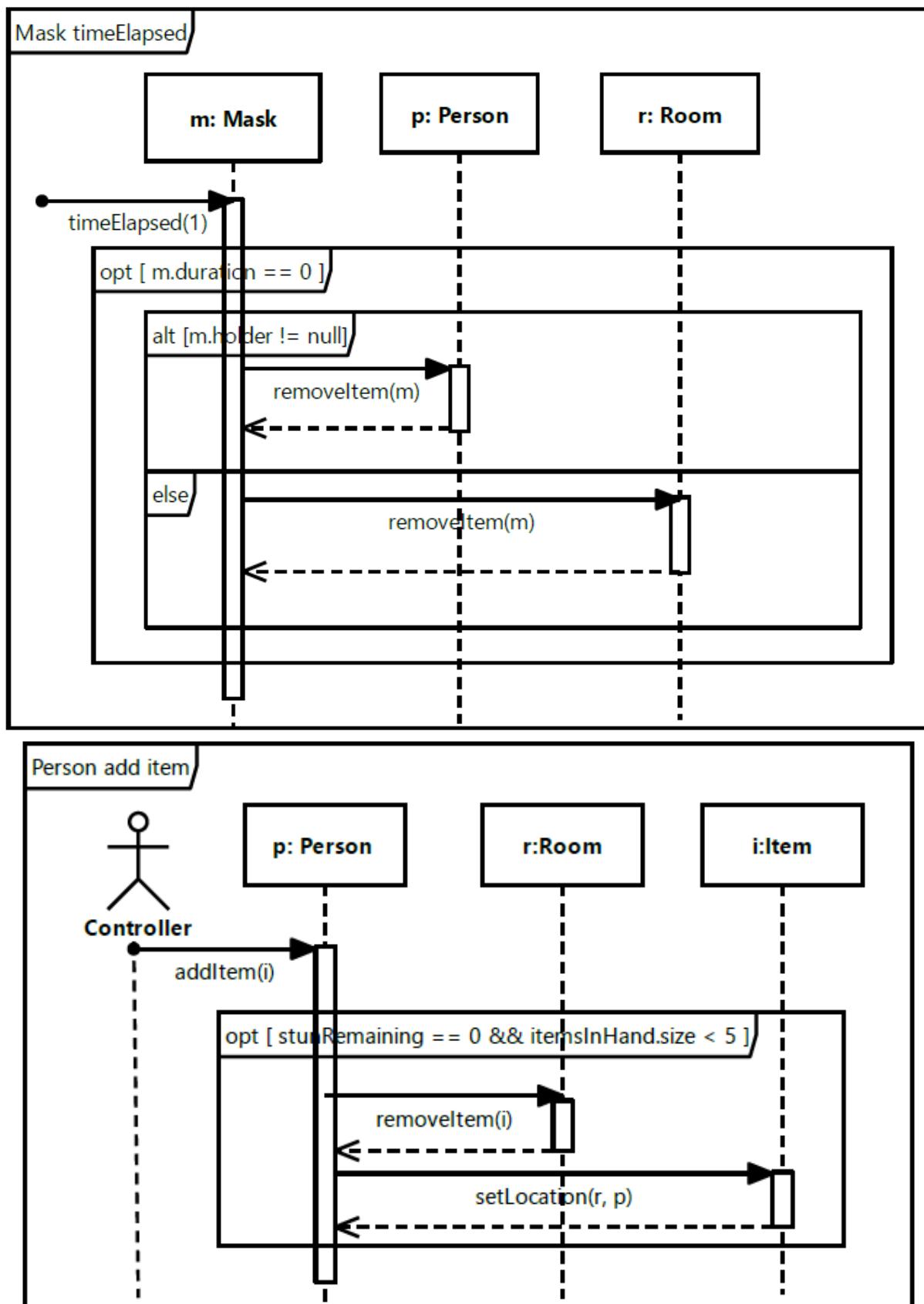


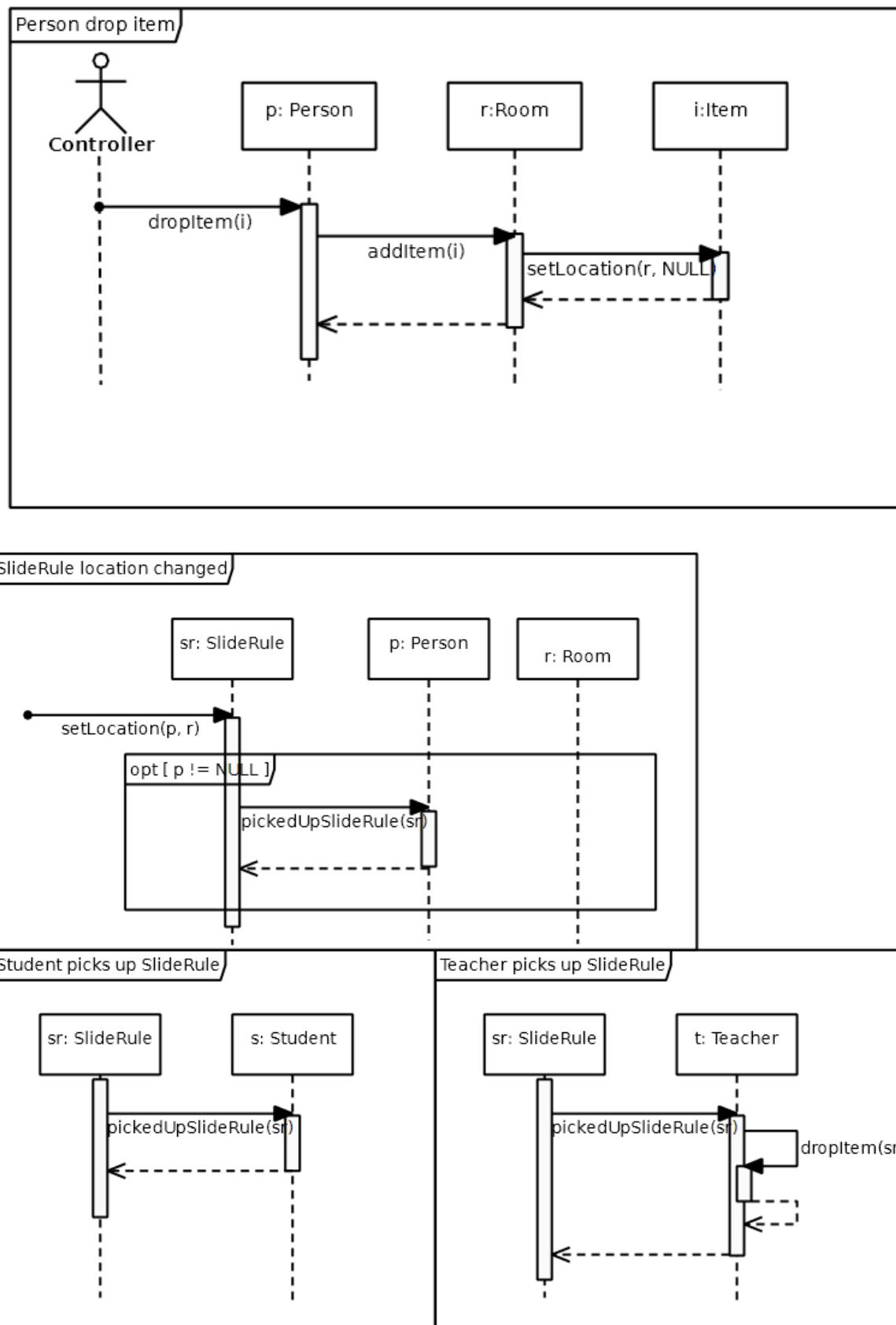


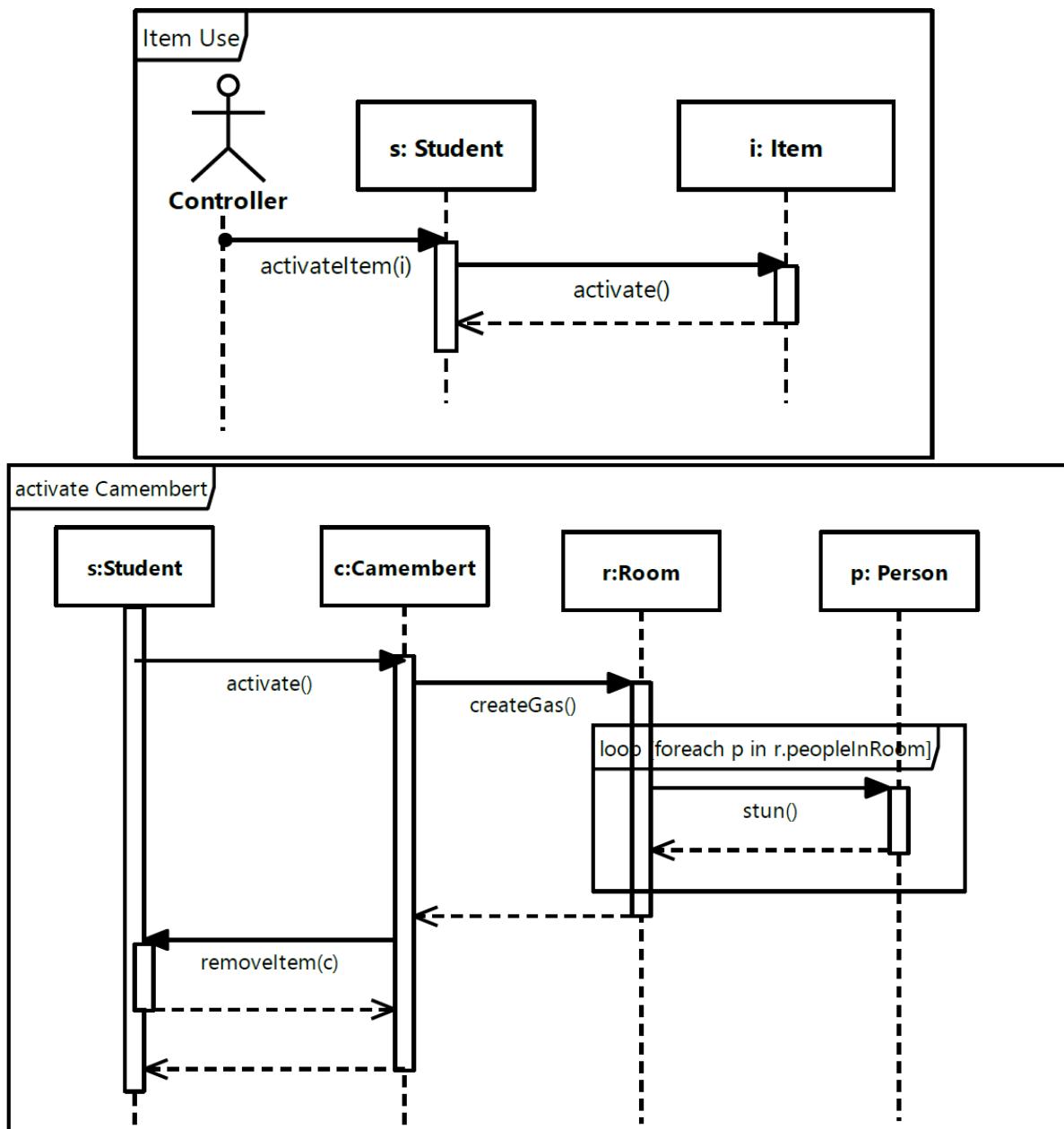


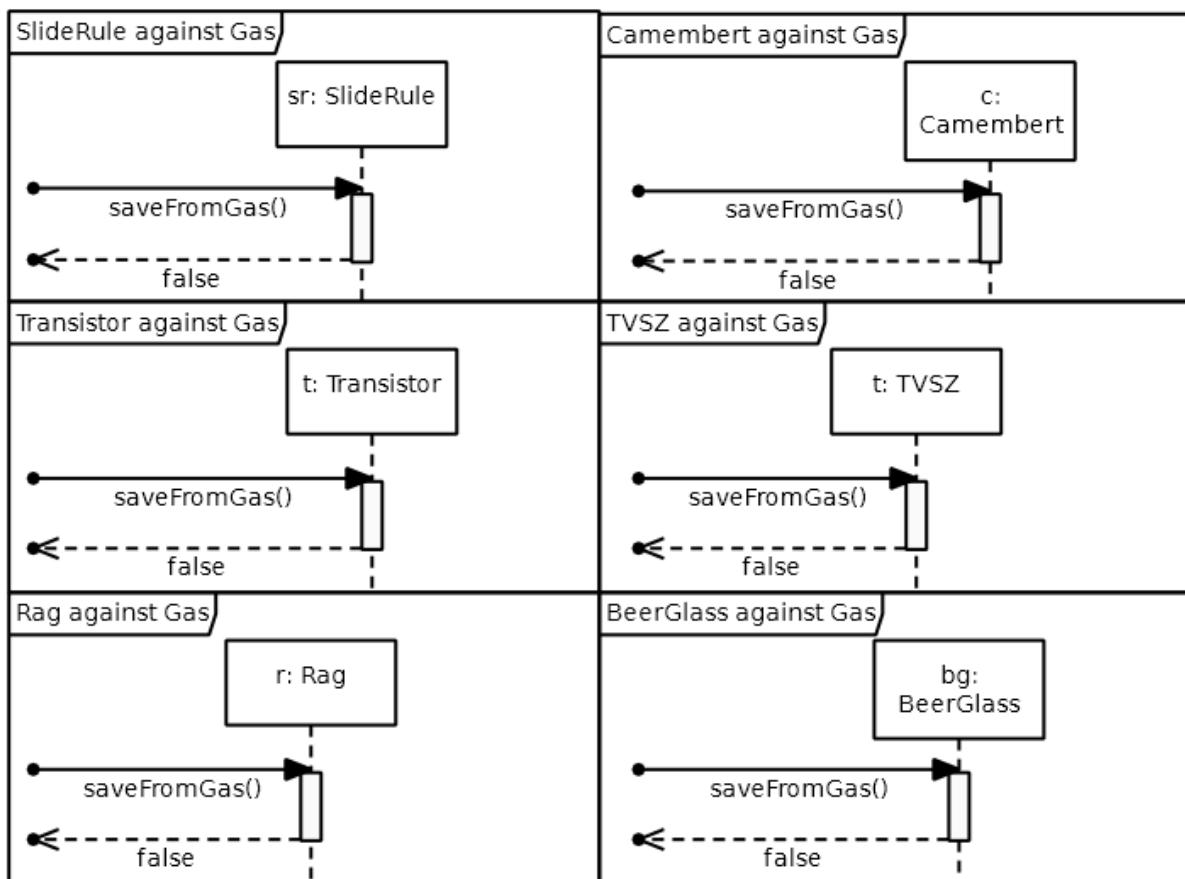
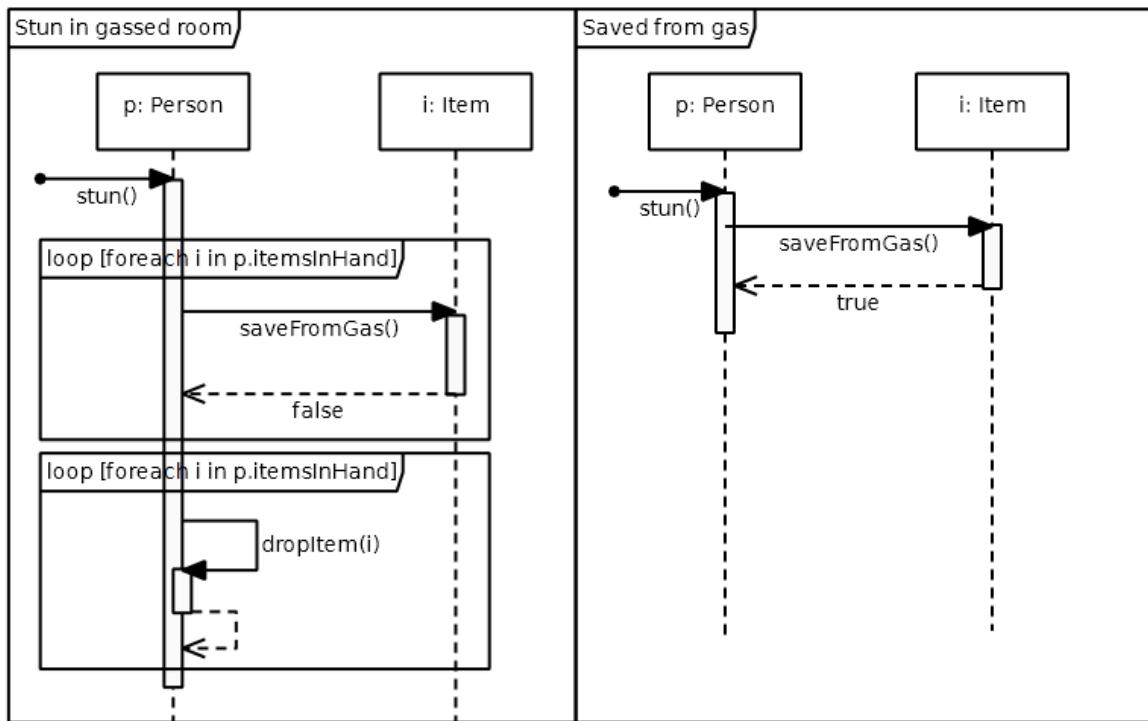


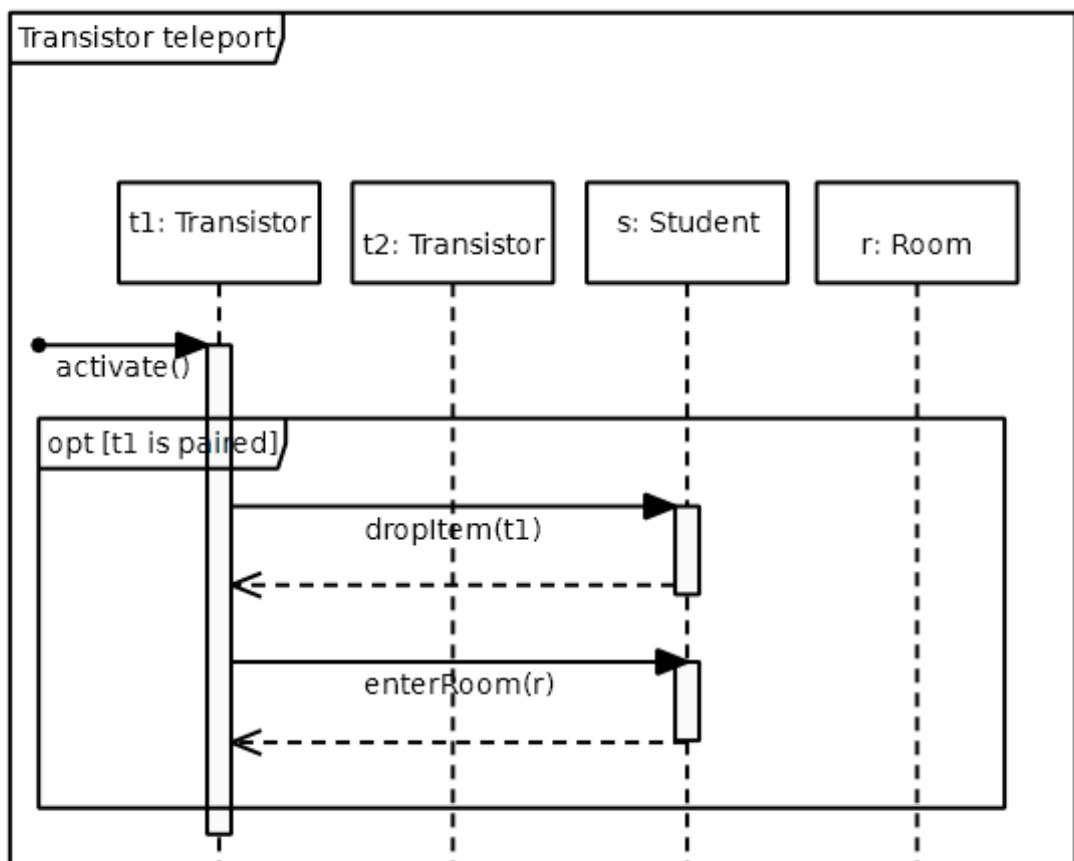
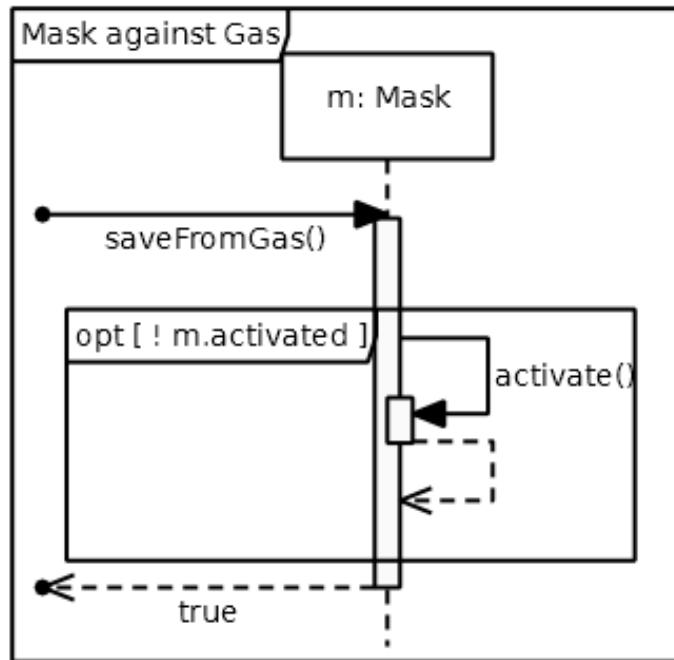




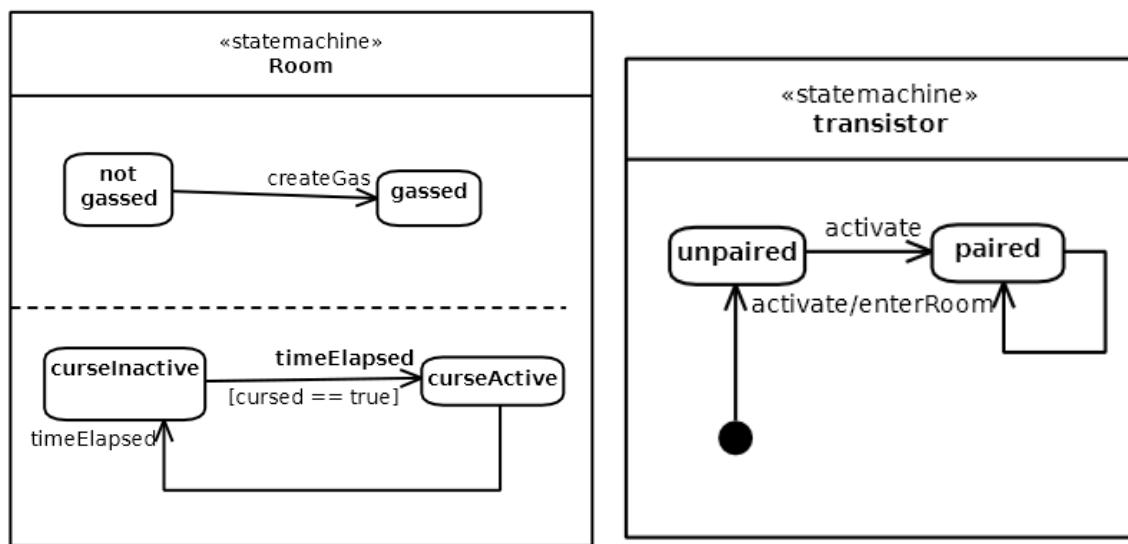
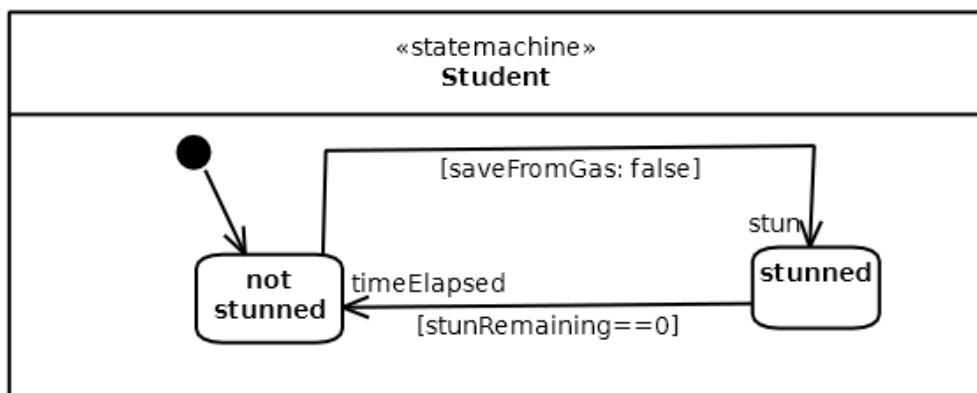
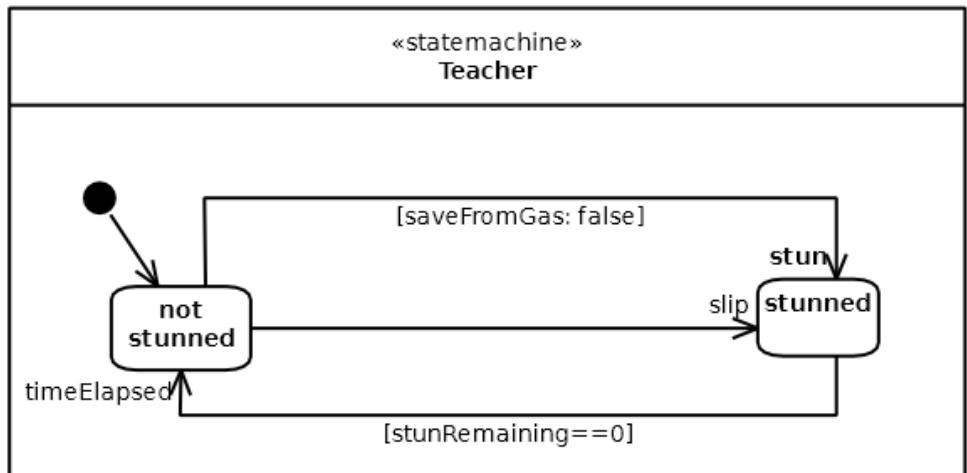


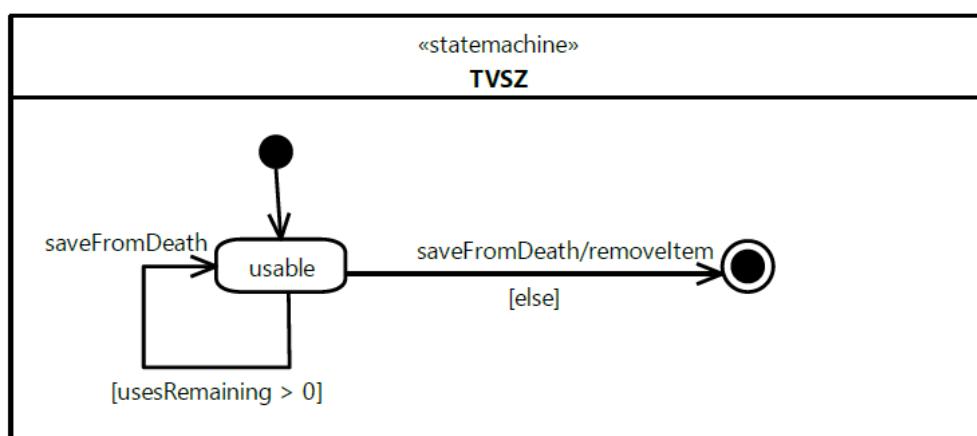
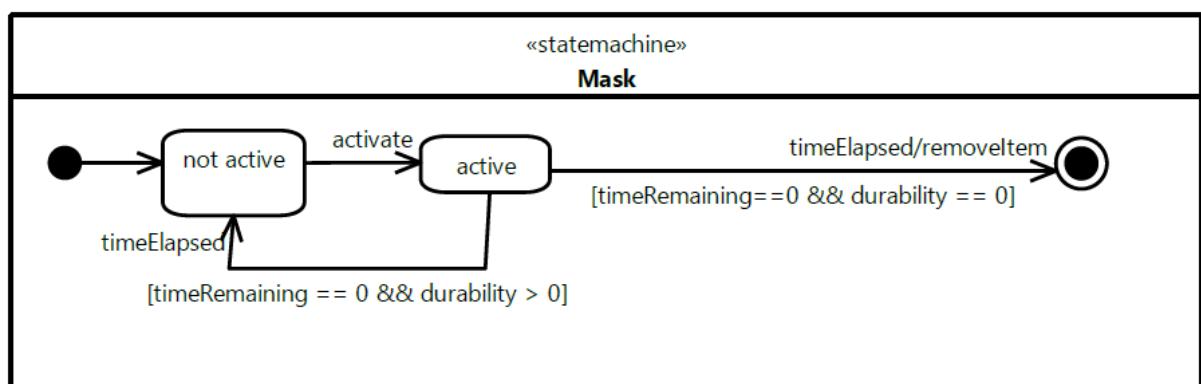
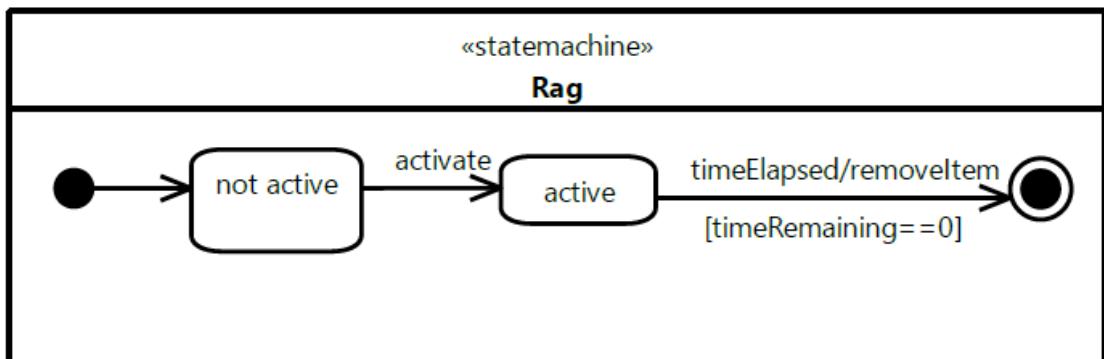
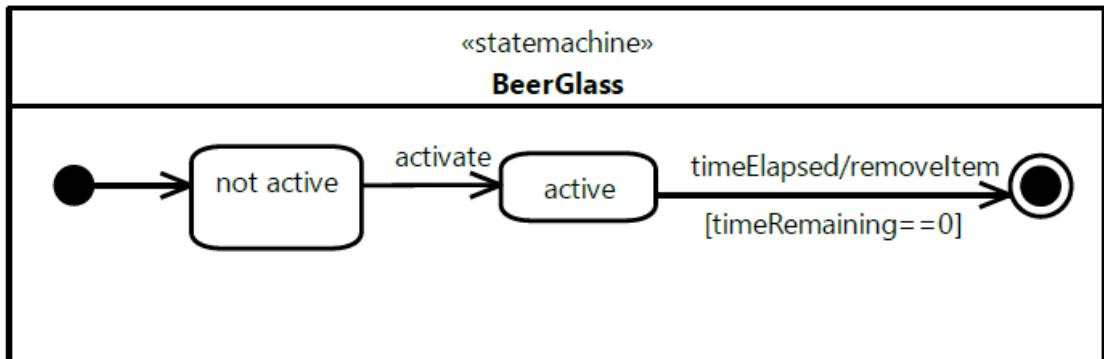






3.5 State-chartok





Napló

Kezdet	Időtartam	Résztvevők	Leírás
2024. 03. 06. 10:15	1,5 óra	Görömbey Héjja Sági Tömöri Vizhányó	Konzultáció
2024. 03. 06. 12:00	2 óra	Görömbey Tömöri	Osztálymodell apró módosítása. Görömbey átírta az osztálydiagram-ot. Tömöri átírta az osztálykatalógust.
2024. 03. 06. 14:00	3 óra	Görömbey Sági Tömöri	Módosítások konziszenessé tétele az érintett szekvenciadiagramokon, azok szintaktikai javítása.
2024. 03. 06. 14:00	2 óra	Vizhányó	Állapotgépek refaktorálása
2024. 03. 06. 17:00	2 óra	Sági Tömöri	Az első 9 szekvenciadiagram javítása, elkészítése
2024. 03. 07. 14:00	2 óra	Görömbey Tömöri Vizhányó	Értekezlet: GitHub karbantartás, hátralevő szekvenciadiagramok átbeszélése.
2024. 03. 07. 16:00	4 óra	Tömöri	Maradék szekvenciadiagramok javítása, elkészítése
2024.03.07. 21:00	0,5 óra	Görömbey	Elkészült diagramok ellenőrzése, bemásolása
2024.03.08. 12:00	2 óra	Héjja	Még nem átnézett dolgok ellenőrzése, diagrammok további ellenőrzése
2024.03.08. 14:00	0,5 óra	Vizhányó	A dokumentum átnézése, diagrammokhoz javítások kezdeményezése
2024.03.08. 21:00	0,5 óra	Sági	A dokumentum átnézése, esetleges javítások kezdeményezése

4. Szkeleton tervezése

4.1 A szkeleton modell valóságos use-case-ai

4.1.1 Use-case diagram



4.1.2 Use-case leírások

Use-case neve	Szobába átlépés
Rövid leírás	Egy hallgató átlép egy másik szobába, ahol már van egy tanár, ott megvédi magát vagy meghalhat.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz egy hallgatót, egy söröspoharat, egy TVSZ-t, egy maszkot egy tanárt és két szobát, melyek szomszédosak, majd a hallgatót és a tanárt elhelyezi egy-egy szobában. A söröspoharat és a maszkot a hallgató kezébe adja, a TVSZ-t a tanár szobájába rakja. Ezután az egyik szobából a hallgató átlépteti a másikba, amennyiben a hallgató nincs elkábítva és a jelenlegi szobája jelenleg nem elátkozott. Itt megkérdezi a felhasználót, hogy a szoba tele van-e, és ha igen, a hallgató nem tud belépni, ha nem, megkérdezi, aktiválva van-e a söröspohár, és ha nem, a hallgató megbukik a tanár által. Egyébként átlép.

Use-case neve	Két oktató találkozik
Rövid leírás	Két oktató találkozik.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz két oktatót, akik találkoznak.

Use-case neve	Két hallgató találkozik
Rövid leírás	Két hallgató találkozik.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz két hallgatót, akik találkoznak.

Use-case neve	Egy oktató találkozik egy hallgatóval
Rövid leírás	Egy oktató és egy hallgató találkoznak.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz egy hallgatót, egy oktatót és egy szobát, ahova mindenket belerakja. Az oktató ezután megbuktatja a hallgatót.

Use-case neve	Tárgyfelvétel
Rövid leírás	Egy hallgató felvesz tárgyakat egy szobából.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz egy hallgatót és egy TVSZ-t, majd hozzáadja egy újonnan létrehozott szobához. Ezután a hallgató felveszi a TVSZ-t. A felvett tárgy eltűnik a szobából, és a hallgató kezébe kerül, ha van a hallgatonál hely és nincsen mégbénulva. Ezt a két feltételt a felhasználó döntheti el. Ha valamelyik feltétel nem áll fenn, akkor az tárgy a szobában marad.

Use-case neve	Tárgylerakás
Rövid leírás	Egy hallgató lerak egy kezében lévő tárgyat.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz egy hallgatót, egy szobát és egy maszkot, amit a hallgató kezébe ad. Ezután a hallgató a szobába rakja, és eldobatja a tárgyat. Az eldobott tárgy a hallgató kezéből eltűnik és a szobába kerül.

Use-case neve	Camembert és maszk használata
Rövid leírás	Egy hallgató használ egy Camembert-et és elgázosítja a szobát, majd használja a maszkját.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz egy szobát és egy hallgatót és a kezébe ad egy Camembert-et, egy TVSZ-t és egy maszkot, majd belehelyezi a szobába. Ezután a hallgató használja a Camembert-et, ami elgázosítja a szobát. Amikor a hallgató elkábulna a gáztól, használja a maszkot, ami megmenti őt az elkábulástól.

Use-case neve	Hallgató elkábul
Rövid leírás	Egy szobában használnak egy Camembert-et és az megbénítja a szobában lévő összes embert.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz két hallgatót és egy szobát, aminek a kapacitása legalább 2, majd minden hallgatót a szobába rakja. Az egyik hallgatónak ezután ad egy Camembert-et, és aktiválta vele. A hallgató megpróbál védekezni, de a TVSZ nem védi meg a gáztól. Mivel egyik hallgatónál sincs olyan tárgy, ami megvédené őket a keletkező gáztól, mindenkitől elkábul.

Use-case neve	Maszk időtelés
Rövid leírás	Egy hallgatónál lévő maszk aktív ideje lejár, ezért eltűnik.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz egy hallgatót és egy maszkot, amit a hallgató kezébe ad. Ezután a maszkot aktiválta és megfelelő számú alkalommal meghívja rajta a timeElapsed-et, ami ennek hatására lejár és törlődik.

Use-case neve	Táblatörlő rongy időtelés
Rövid leírás	Az időtelésére a táblatörlő rongy megbénítja a vele egy szobában lévő oktatót, ha kiszárad, akkor eltűnik
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz egy szobát és elhelyez benne egy táblatörlő rongyot és egy oktatót. Az időtelésére, ha a rongy aktív és nincsen kiszáradva, akkor az oktatót megbénítja. Ha a rongy kiszárad, akkor eltűnik a szobából. A felhasználó tudja eldönteni, hogy a rongy aktív-e illetve, hogy ki van-e száradva.

Use-case neve	Söröspohár időtelés
Rövid leírás	Az időtelésével a söröspohár megvédi a hallgatót az oktatótól, ha aktív és nem járt le a hatása. Ha lejárt a hatása, akkor eltűnik és nem nyújt további védelmet.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz egy szobát elhelyez benne egy hallgatót és oktatót. A hallgató kezébe ad egy söröspoharat. A felhasználó eldöntheti, hogy a söröspohár aktív-e illetve, hogy a még van-e még használható-e. Ha nem használható, akkor a eltűnik a játékos kezéből és az oktató megöli a hallgatót. Ha aktív és használható, akkor megvédi a hallgatót a bukástól.

Use-case neve	TVSZ használata
Rövid leírás	Egy oktató megpróbálja megölni hallgatót, de az megvédi magát egy TVSZ-szel.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz egy oktatót és egy hallgatót. A hallgató kezébe ad egy TVSZ példányt. Az oktató találkozik a hallgatóval. Ha az oktató nincs megbénulva (amit a felhasználó dönt el), akkor megpróbálja a hallgatót megölni. A hallgató megvédi magát a TVSZ-szel. A felhasználó eldöntheti, hogy ez volt-e az utolsó használata a tárgynak. Ha igen, akkor a tárgy eltűnik a hallgató kezéből.

Use-case neve	Tranzisztor használata
Rövid leírás	Egy hallgató aktivál egy nála lévő tranzisztort és átteleportál a pájjához.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz egy hallgatót és két szobát, illetve két tranzisztort. A tranzisztorokat összpárosítja és az egyiket elhelyezi az egyik szobában, míg a másikat a hallgató kezébe adja és hallgatót elhelyezi a másik szobában. A hallgatónál lévő tranzisztort aktiválva a hallgató eldobja a tranzisztorát a szobában és átteleportál a másik szobába, ahol a másik tranzisztor van. A felhasználó felelőssége a teszteset helyes működése érdekében a páros működés kiválasztása.

Use-case neve	Szobák egyesítése
Rövid leírás	Két szoba egyesül.
Aktorok	Felhasználó
Forgatókönyv	A szkeleton létrehoz két szobát, melyek egymással szomszédosak, amikbe egyenként berak egy-egy tárgyat, ezután egyesíti őket. Ha nincs ember egyik szobában se, akkor sikerül az egyesülés. Az eddig a két külön szobában található TVSZ és maszk mind az új szobában lesznek. Ha van ember a bármelyik szobában, akkor nem történik meg az egyesülés.

Use-case neve	Szoba kettéválása
Rövid leírás	Egy szoba kettéosztódik.
Aktorok	Felhasználó
Forgatókönyv	A szkeleton létrehoz egy szobát, amibe belerak egy maszkot és egy táblatörő rongyot. Ezután meghívja a szobán a split() metódust. Ha nincs ember a szobában, akkor a szoba kettéválik. Az új szobák szomszédosak lesznek, az eredeti szobában lévő tárgyak megoszlanak a két új szoba között. A táblatörő rongy a második szobába kerül, a maszk az elsőben marad. Ha van ember a szobában, akkor nem fog osztódni.

Use-case neve	Hallgató felveszi a logarlécet
Rövid leírás	Egy hallgató felveszi a logarlécet és megnyeri a játéket.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó választása után a szkeleton létrehoz egy hallgatót és egy szobát, ezután a szobába berakja a hallgatót és egy logarlécet. A hallgató felveszi és ezzel megnyerke a játéket.

Use-case neve	Oktató felveszi a logarlécet
Rövid leírás	Egy oktató felveszi a logarlécet és el is dobja.
Aktorok	Felhasználó
Forgatókönyv	A szkeleton létrehoz egy oktatót, egy szobát és egy logarlécet. Egy oktató felveszi a logarlécet. Ezután el is dobja azonnal.

4.2 A szkeleton kezelői felületének terve, dialógusok

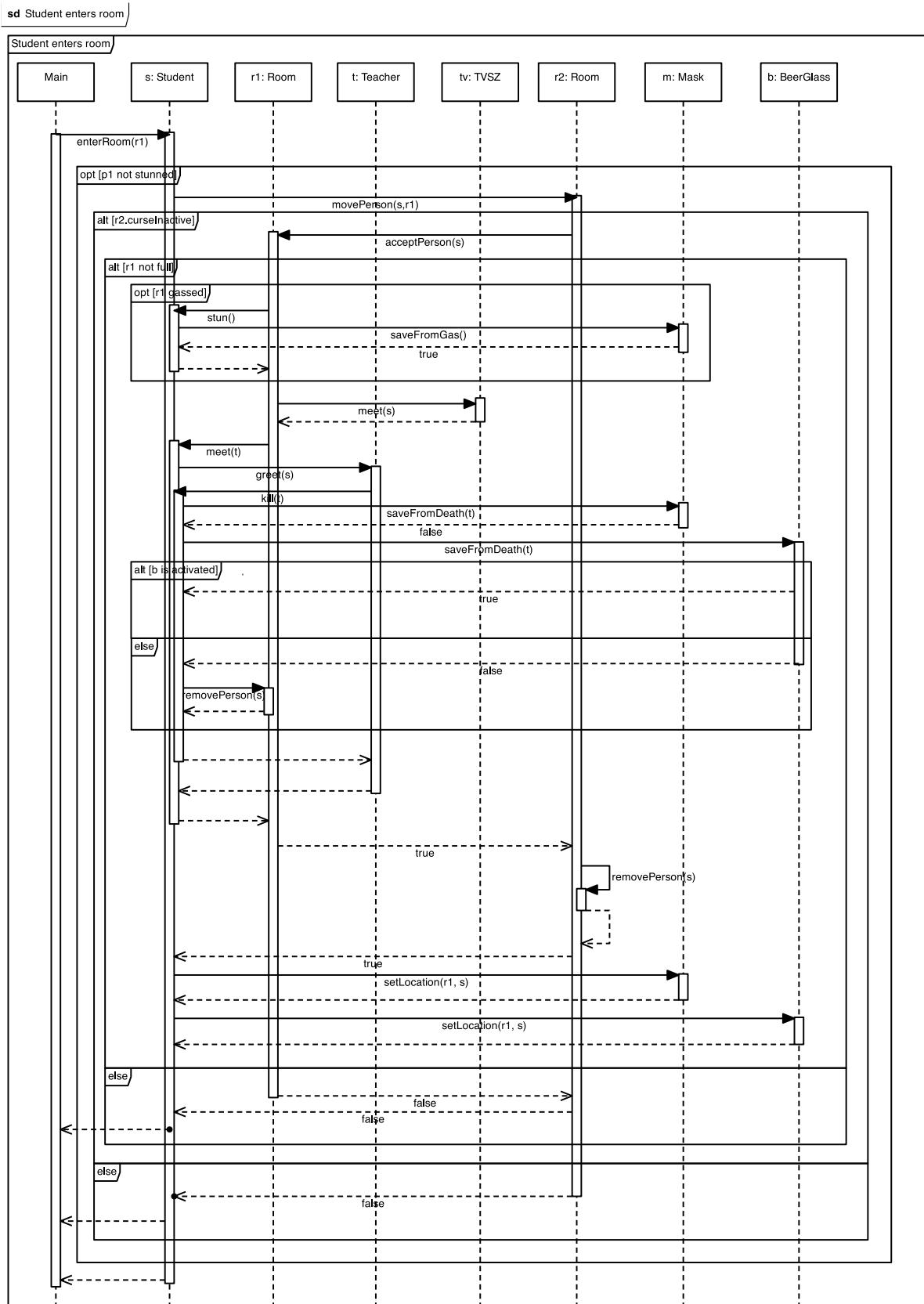
A szkeleton alapvetően egy menüvezérelt konzolos program, ami segítségével ellenőrizni lehet a különböző osztályok alapvető megvalósításait és az elkészítendő szoftver egyéb funkcióit. A teszteseteket számmal látjuk el, ezekből lehet választani a menüben. Választás (konkrétan a kiválasztott teszteset száma) után látni fogjuk azt a hívásfolyamot (függvényhívás és -visszatérés), ami a teszteset közben lefut a különböző objektumok egymással való kommunikációja során. A hívásfolyam a függvényhívásokat jobbra mutató nyíllal (->), míg a függvény visszatéréseket balra mutató nyíllal jelöli (<-). A nyíl után függvényhívás esetén feltünteti, hogy melyik objektum mely tagfüggvényét hívta meg. Visszatérés esetén feltünteti a visszatérési értéket, ha van. A függvényhívások sorrendjének megfelelő követhetősége érdekében indentálva jeleníti meg az egyes műveleteket. Ha egy függvény egy másik függvényt meghív, akkor a második függvényhívás nagyobb behúzással jelenik meg a konzolon. Alapesetben nincsen behúzása a soroknak. Egy függvény működéséhez szükség lehet az objektum egy tulajdonságának értékére. Ezt az értéket a szkeleton programban a felhasználó döntheti el. Ha ilyen függvény kerül meghívásra, akkor a konzolon megkérdezi a tesztelőt, hogy milyen értékkel fusson le a teszteset. Valamennyi kérdés eldöntendő, így a felhasználónak elég csak egy betűt begépelnie (y/n) az adott tagváltozó beállításához. A kérdések is hasonlóan indentáltak, mint a függvényhívások.

Konzol szemlétetése:

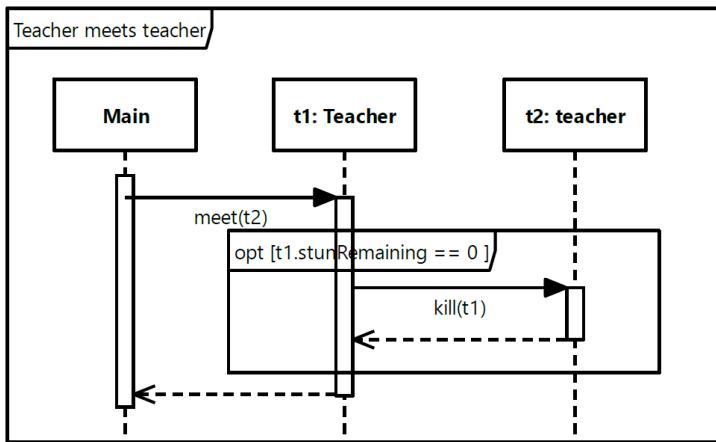
```
->Student()
<-Student
h created
->Room()
<-Room
r1 created
->Room()
<-Room
r2 created
->h.setRoom(r1)
<-h.setRoom
->h.enterRoom(r2)
    Is h stunned? [y/n] n
    ->r1.movePerson(h, r2)
        Is r1 cursed? [y/n] y
        <-r1.movePerson: false
<-h.enterRoom
```

4.3 Szekvencia diagramok a belső működésre

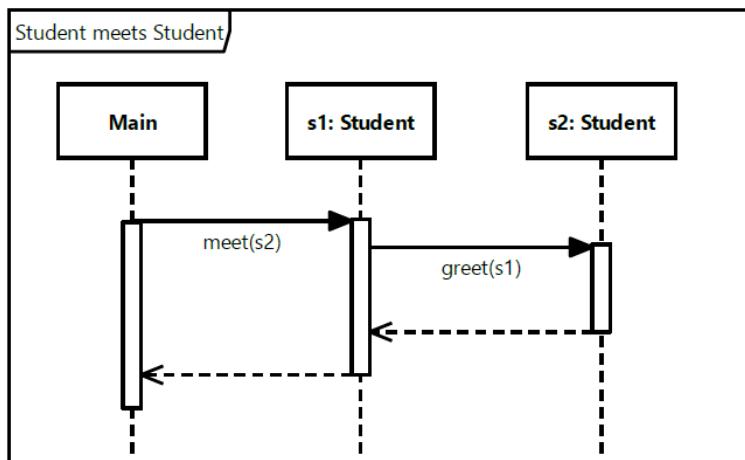
4.3.1 Szobába átlépés



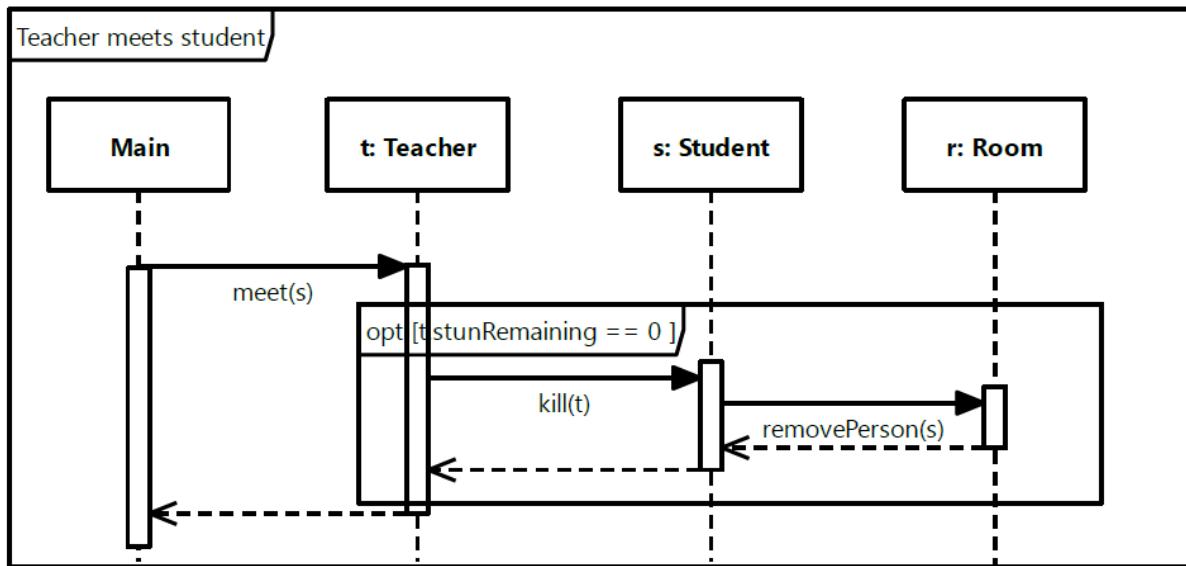
4.3.2 Oktatók találkozása



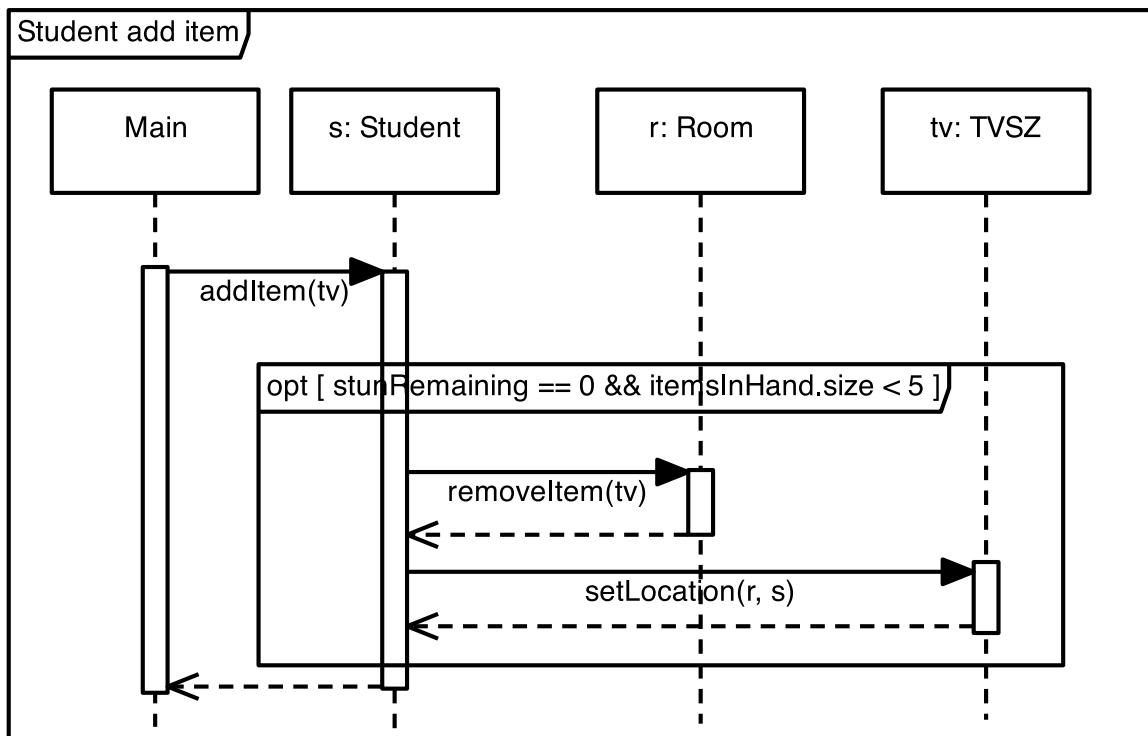
4.3.3 Hallgatók találkozása



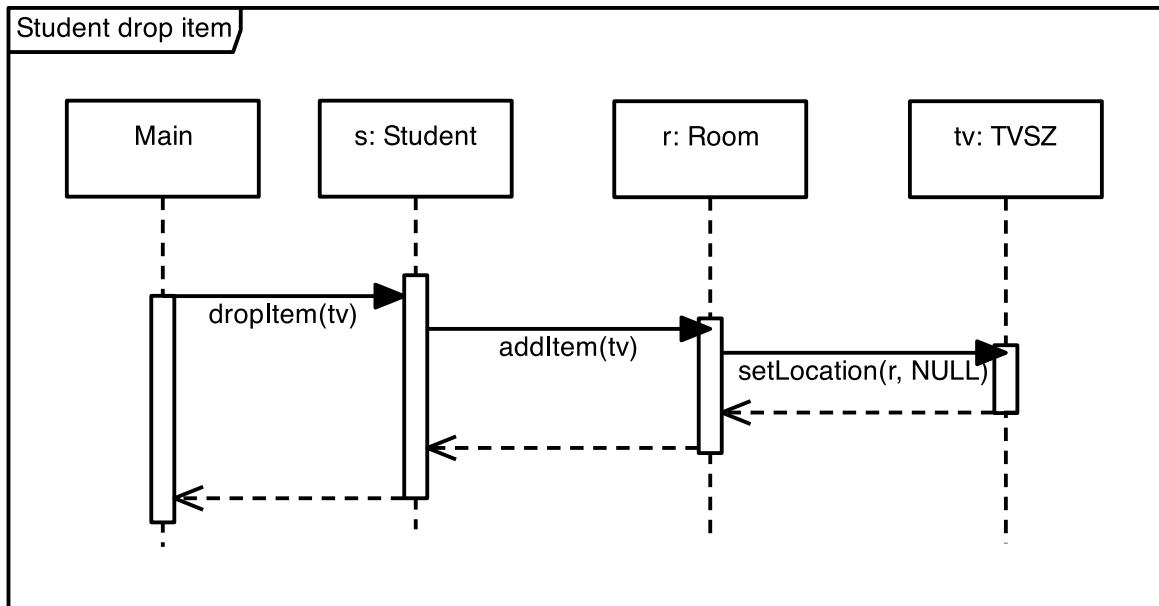
4.3.4 Oktató találkozik egy hallgatóval



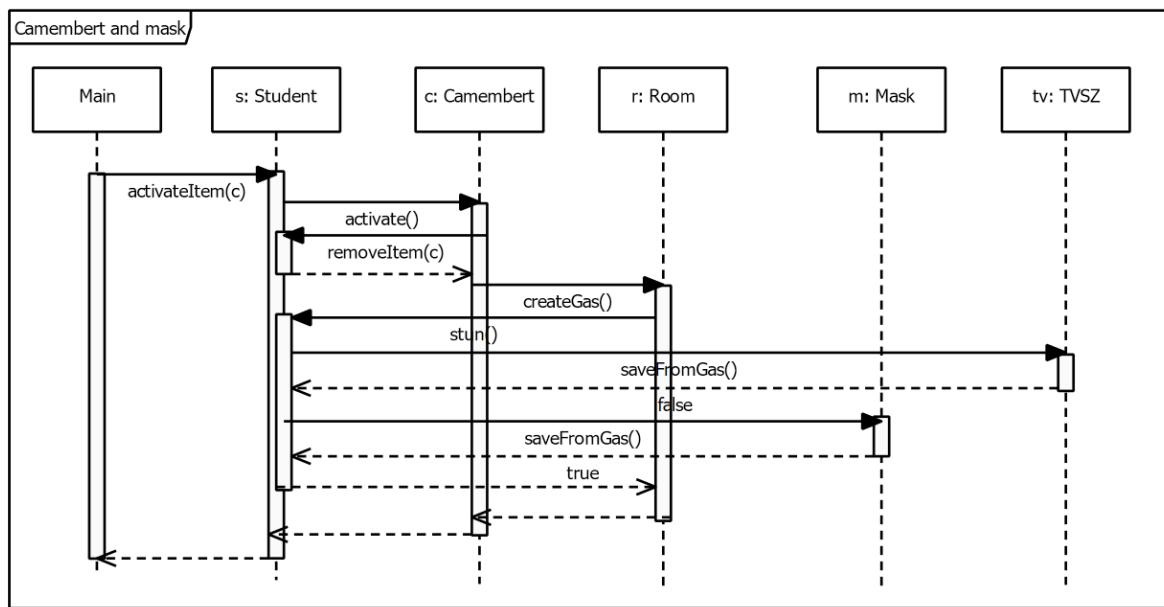
4.3.5 Tárgyfelvétel



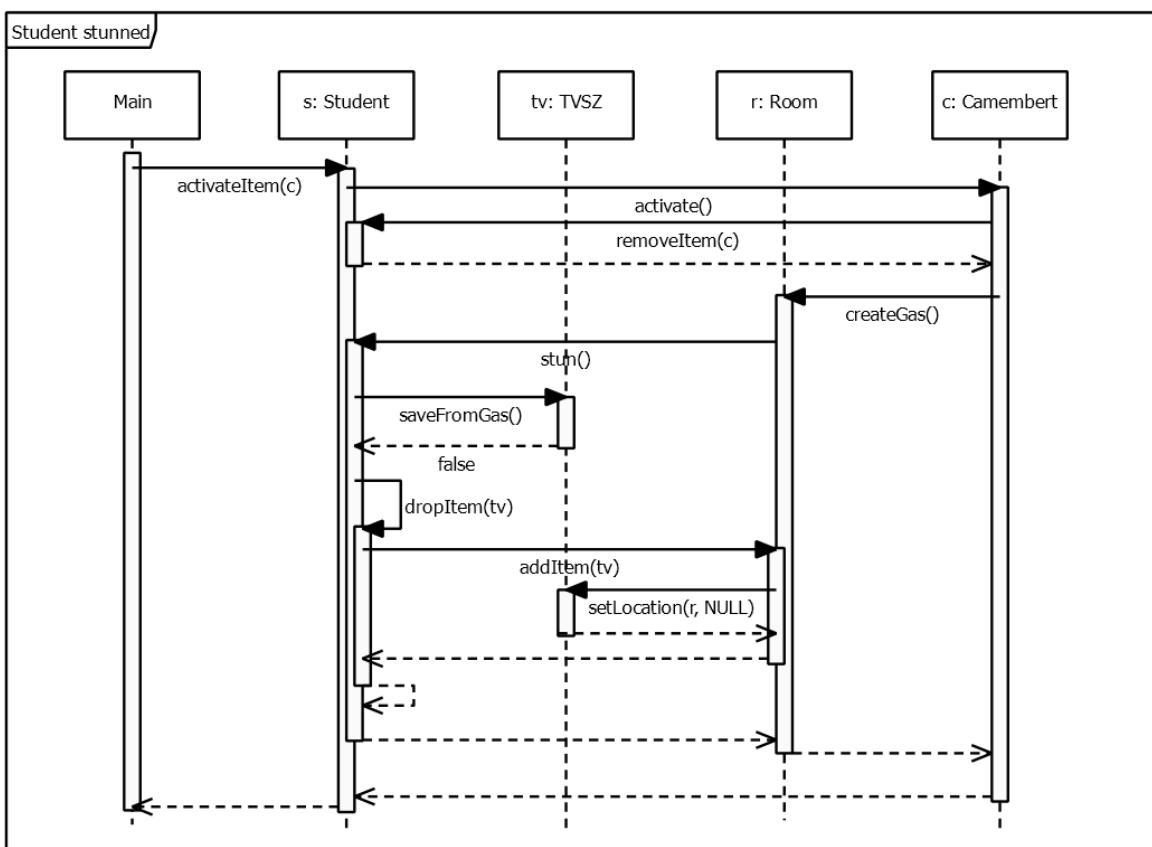
4.3.6 Tárgylerakás



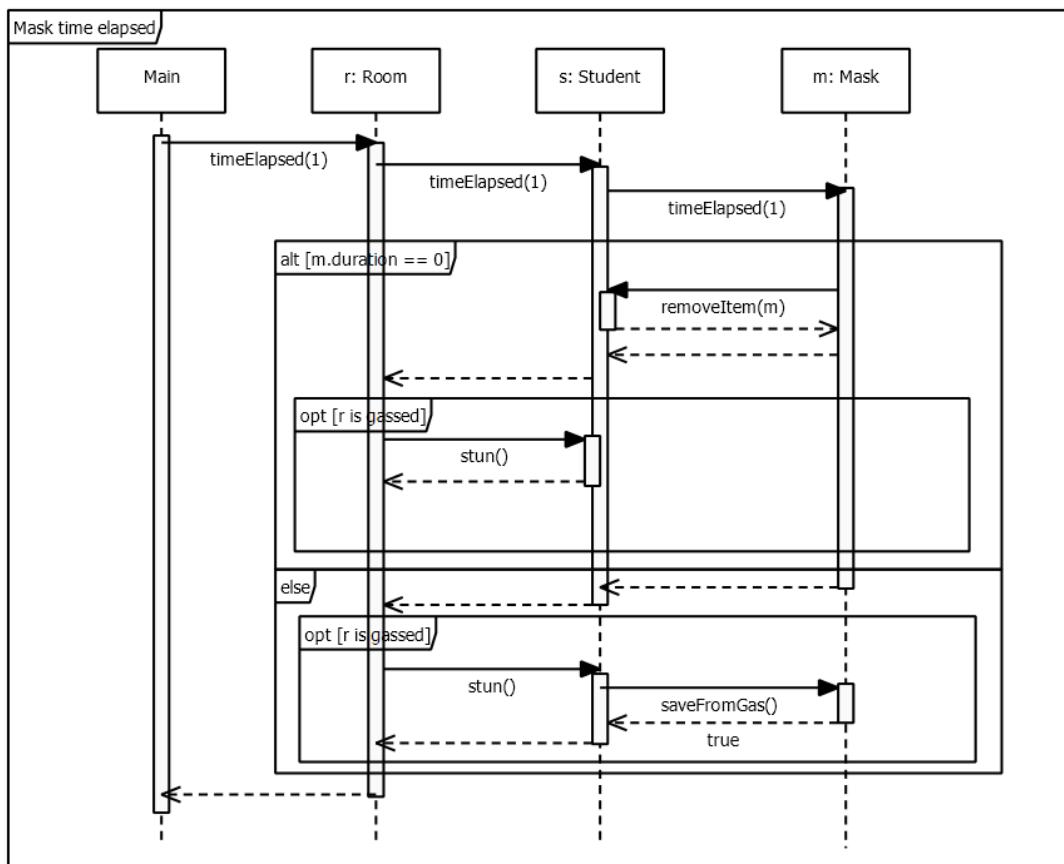
4.3.7 Camembert és maszk használata



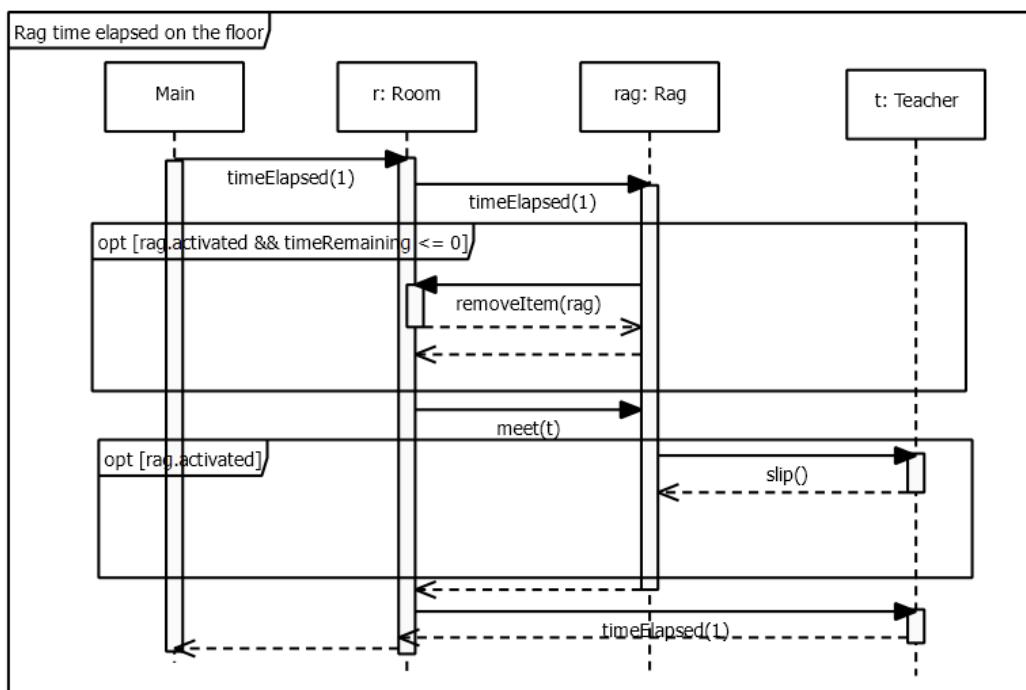
4.3.8 Hallgató elkábul



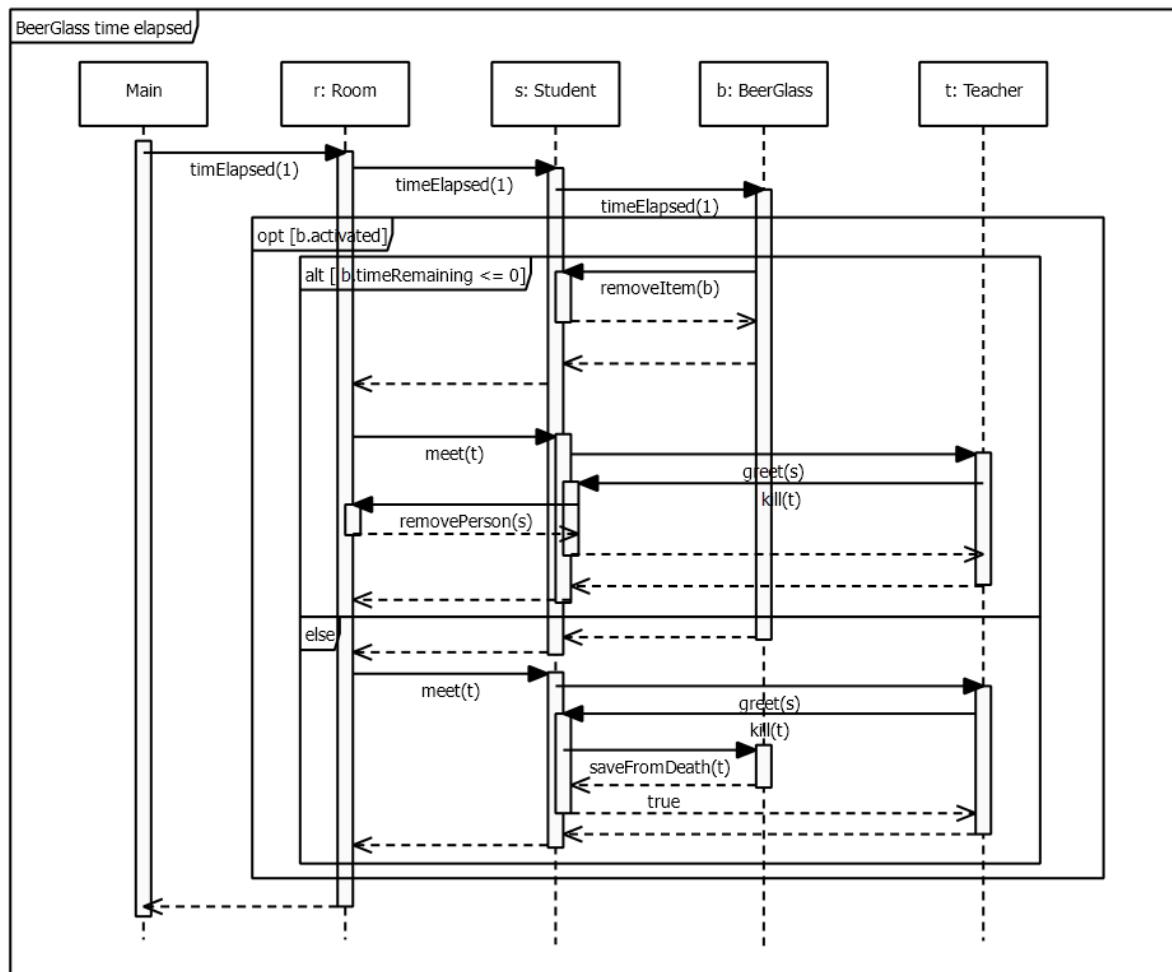
4.3.9 Maszk időtelés



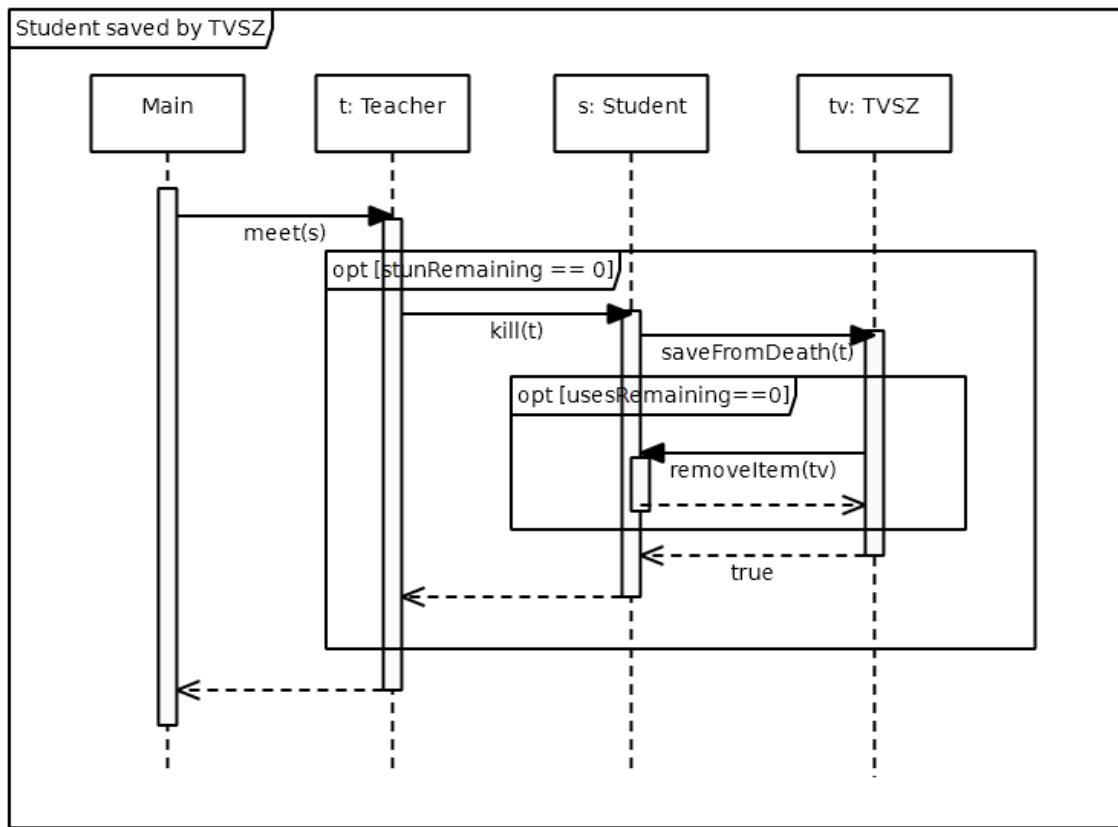
4.3.10 Táblatörlő rongy időtelés



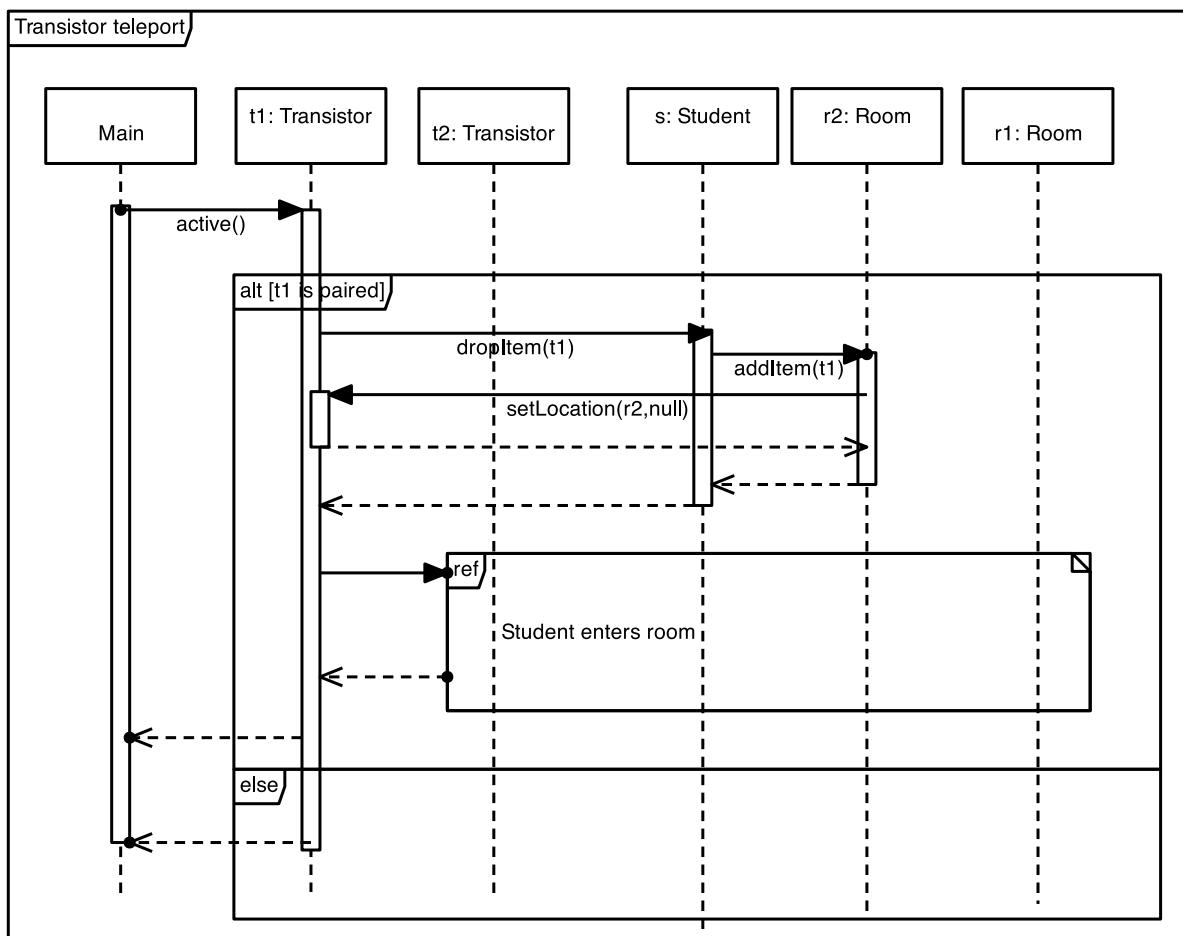
4.3.11 Söröspohár időtelés



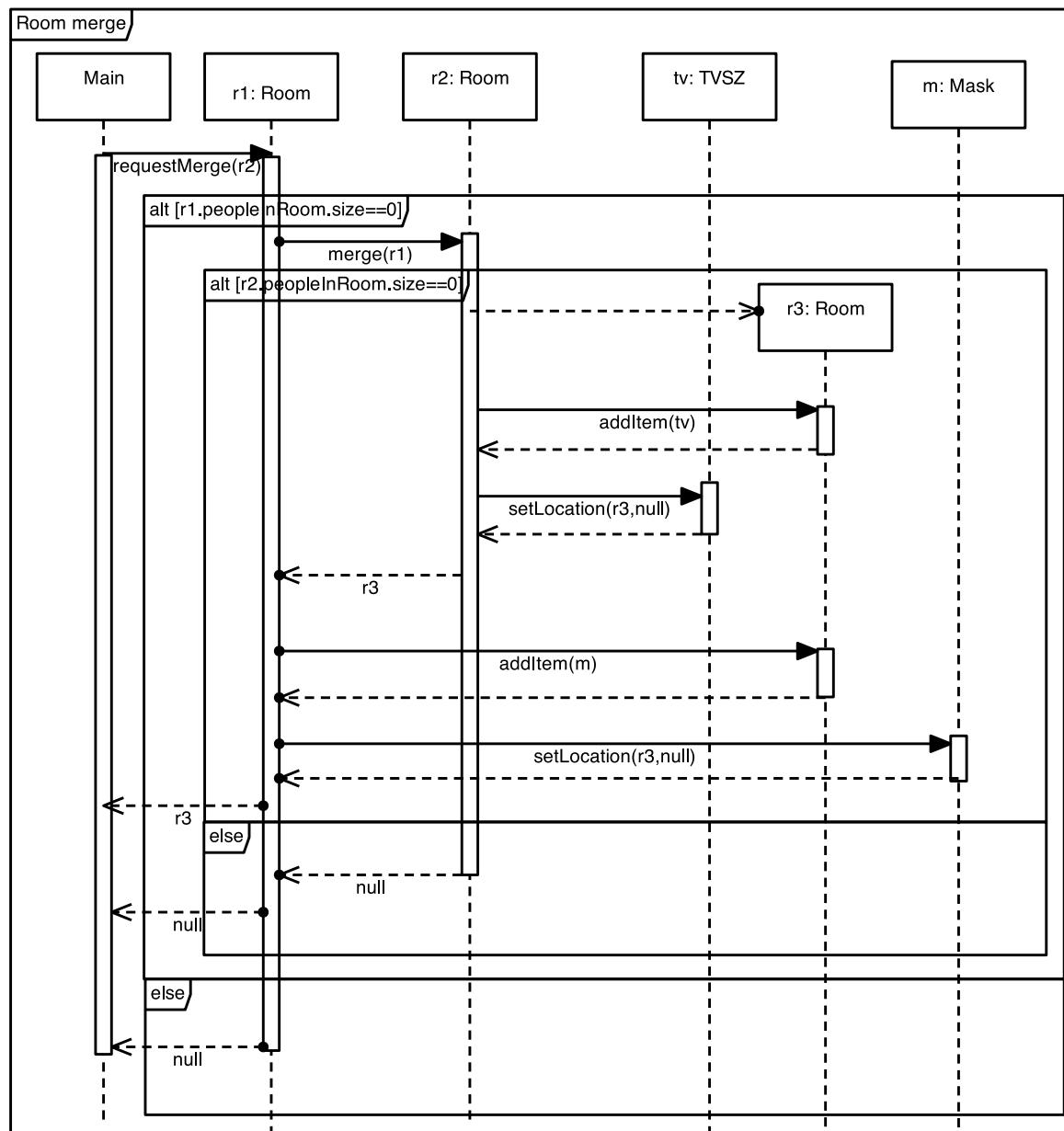
4.3.12 TVSZ használata



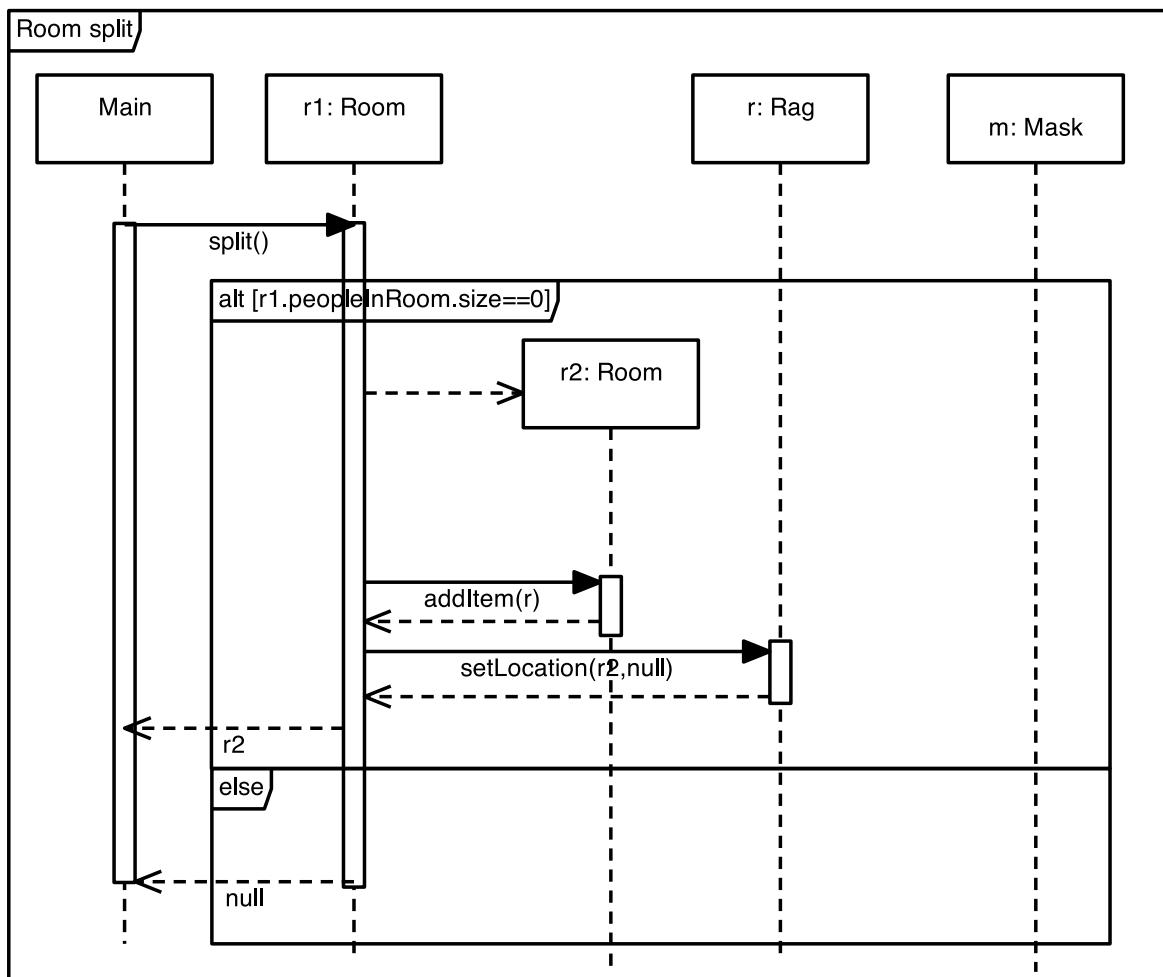
4.3.13 Tranzisztor használata teleportálásra



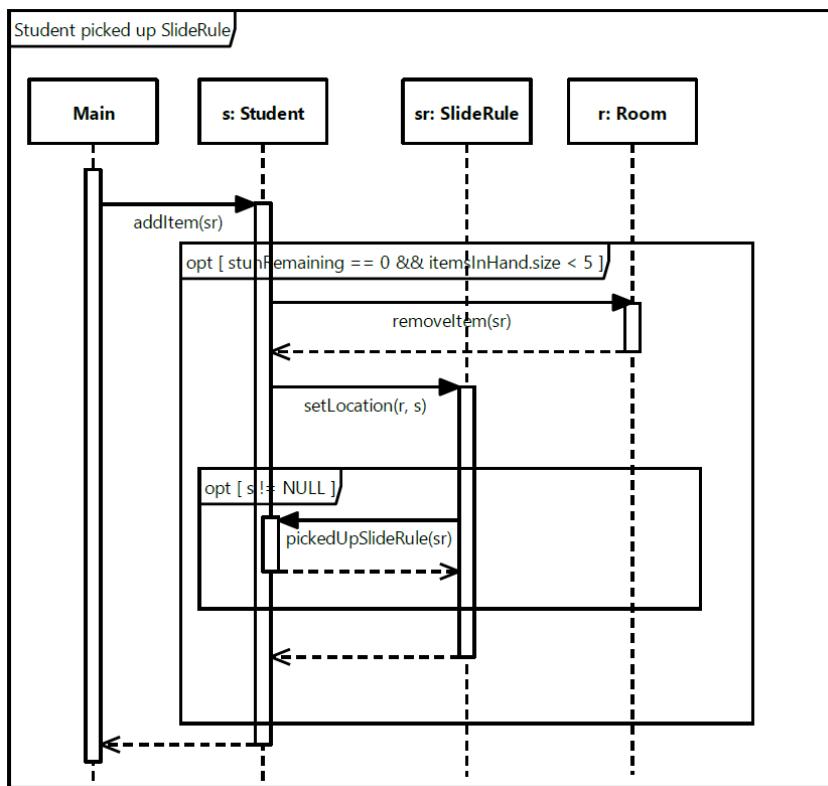
4.3.14 Szobák egyesítése



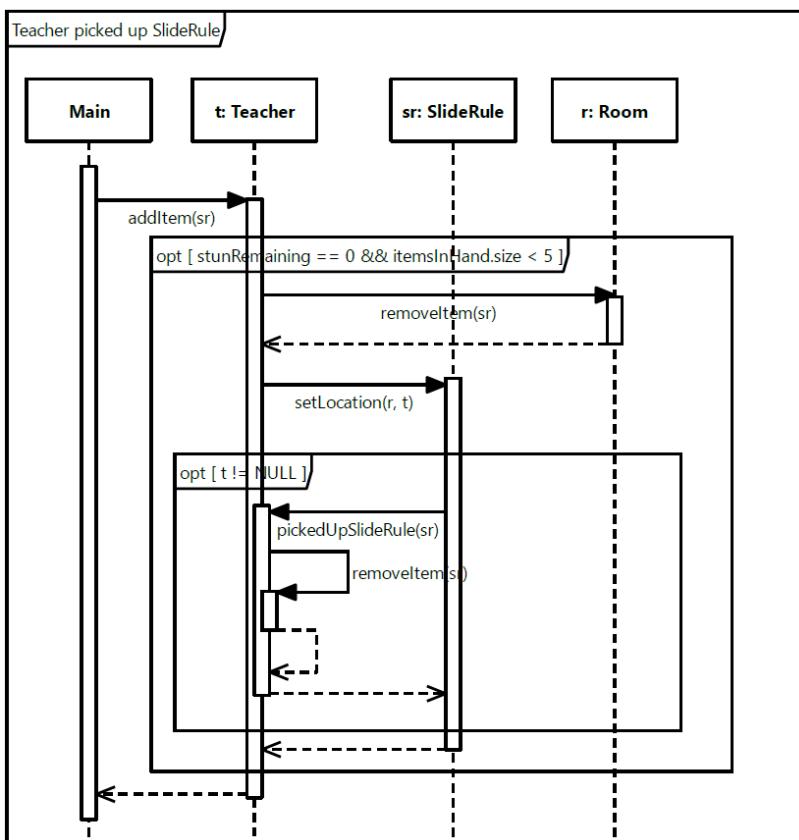
4.3.15 Szoba kettéválása



4.3.16 Hallgató felveszi a Logarlécet

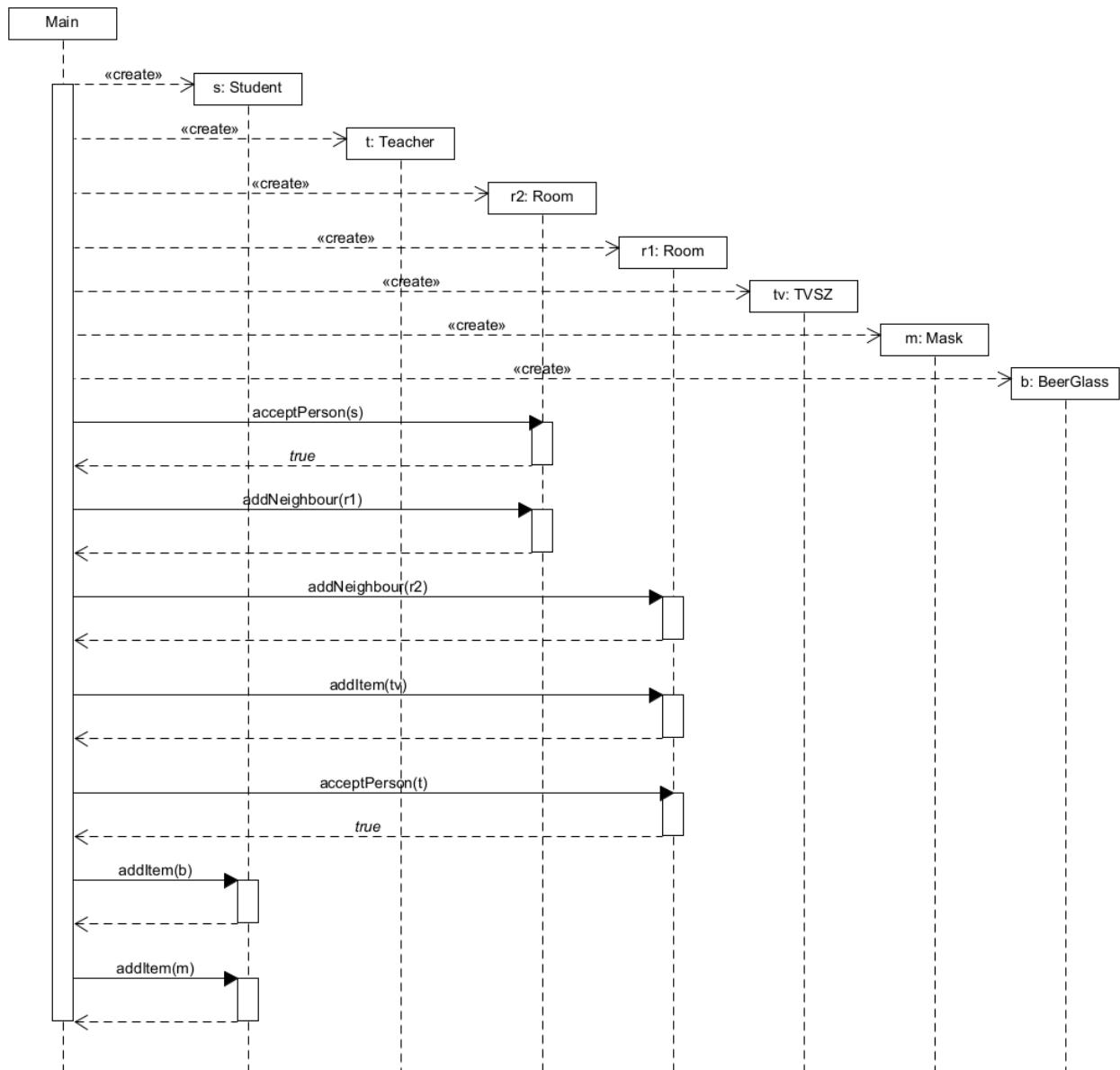


4.3.17 Oktató felveszi a Logarlécet

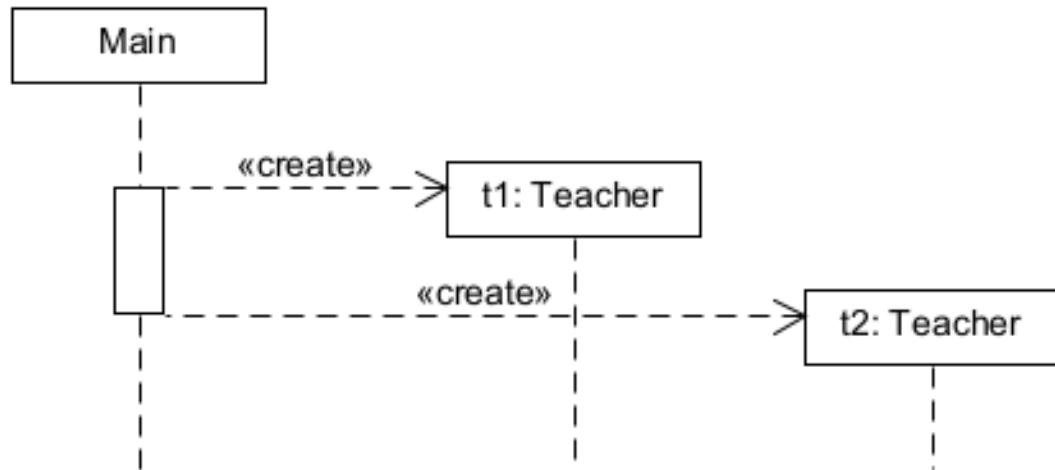


4.4 Kommunikációs diagramok

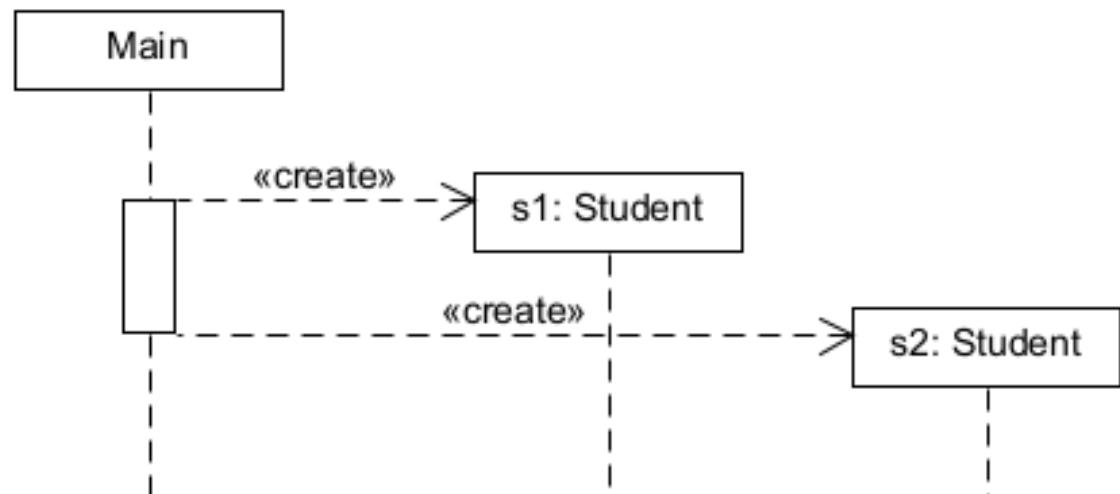
4.4.1 Szobába átlépés



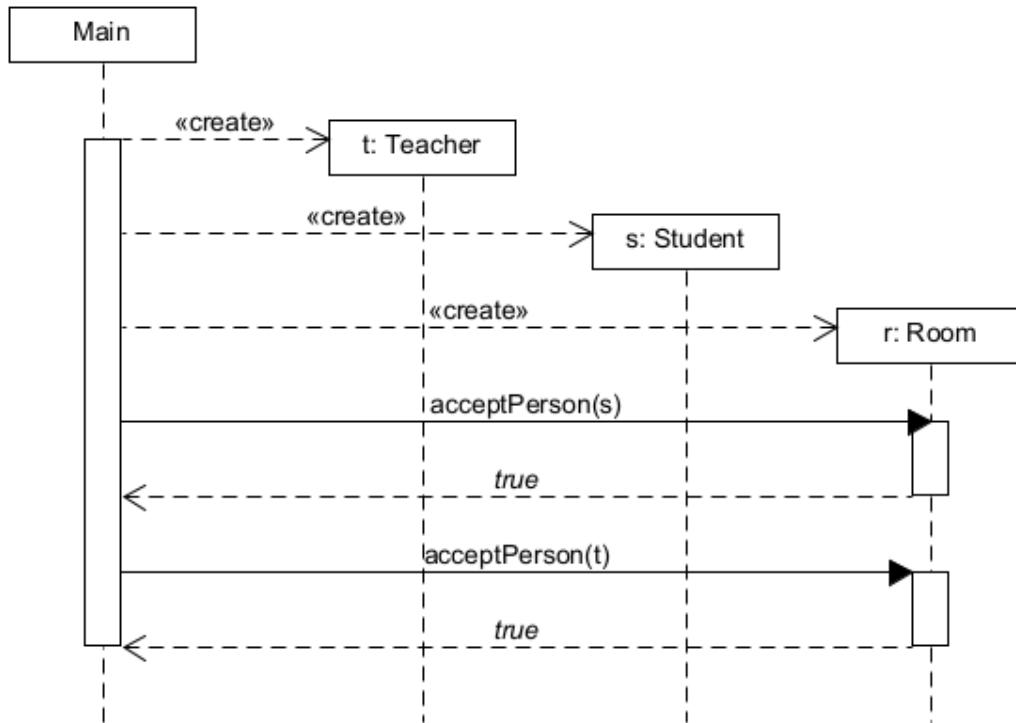
4.4.2 Két oktató találkozik



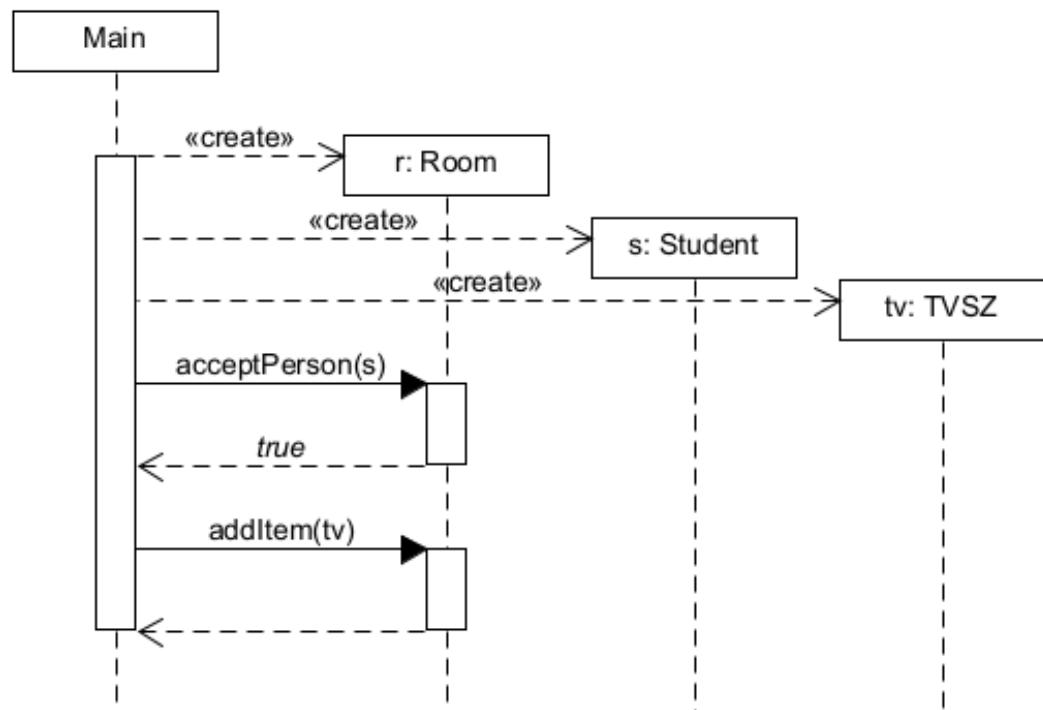
4.4.3 Két hallgató találkozik



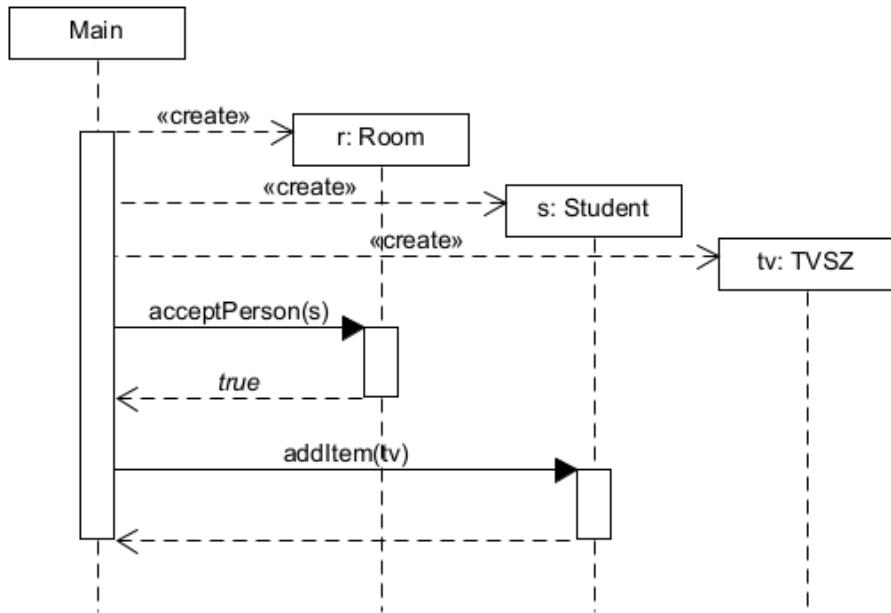
4.4.4 Oktató találkozik hallgatóval



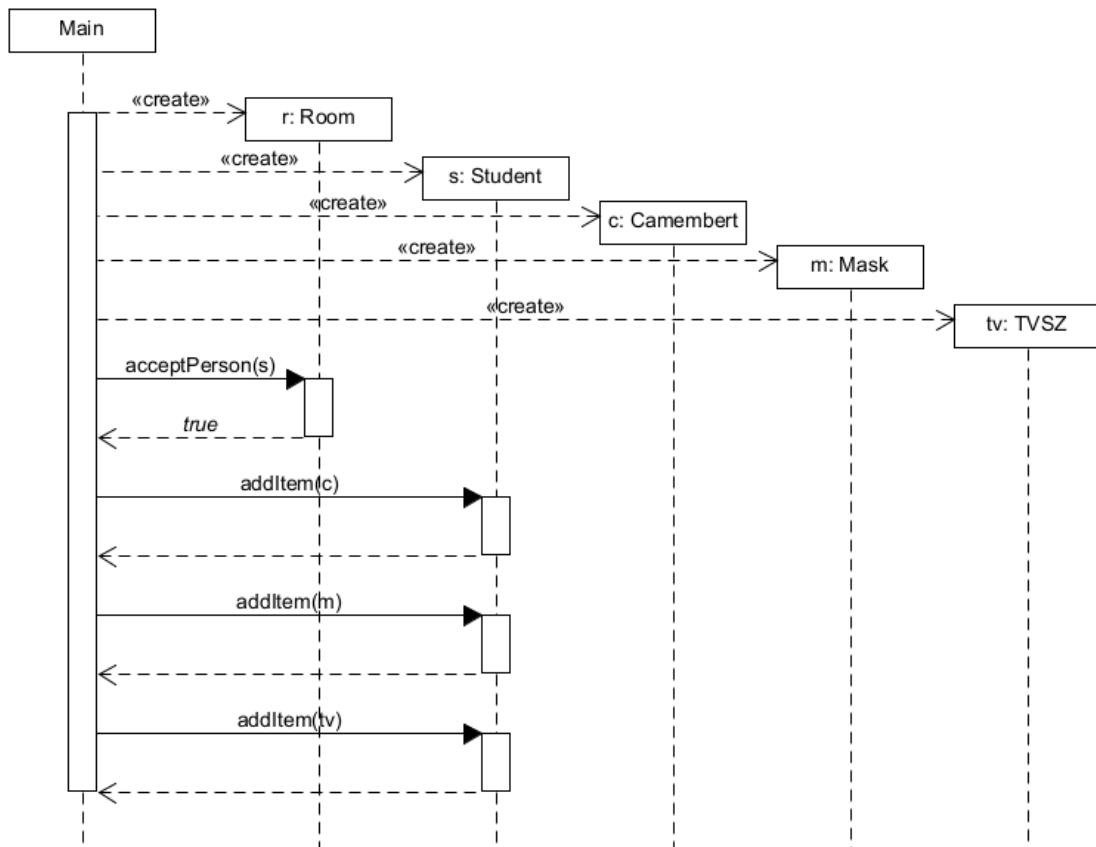
4.4.5 Tárgyfelvétel



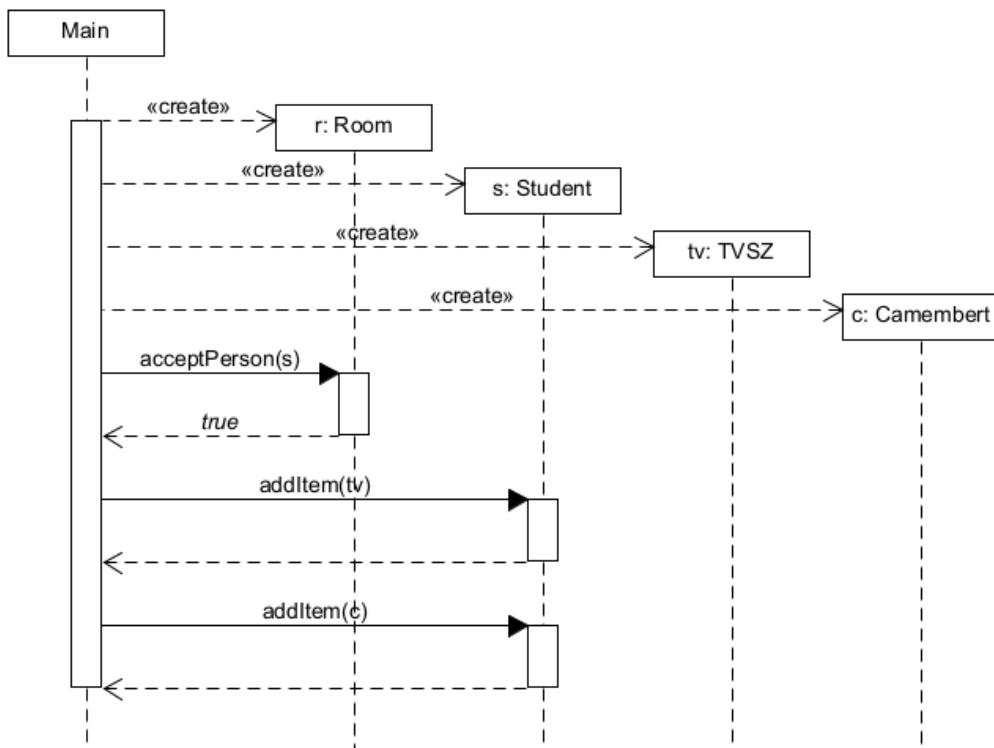
4.4.6 Tárgylerakás



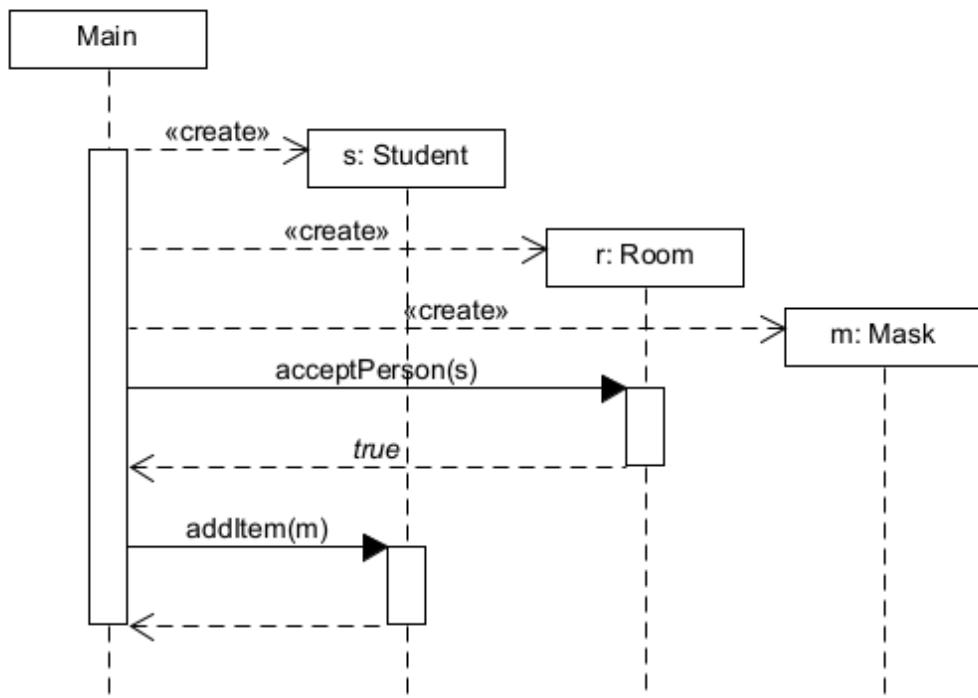
4.4.7 Szoba elgázosítása



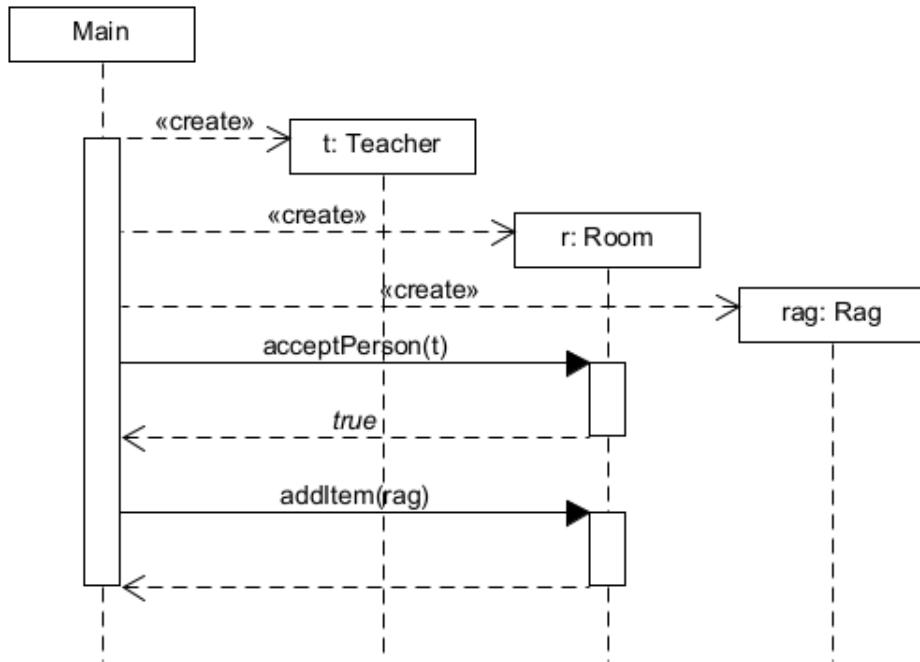
4.4.8 Hallgató elkábul



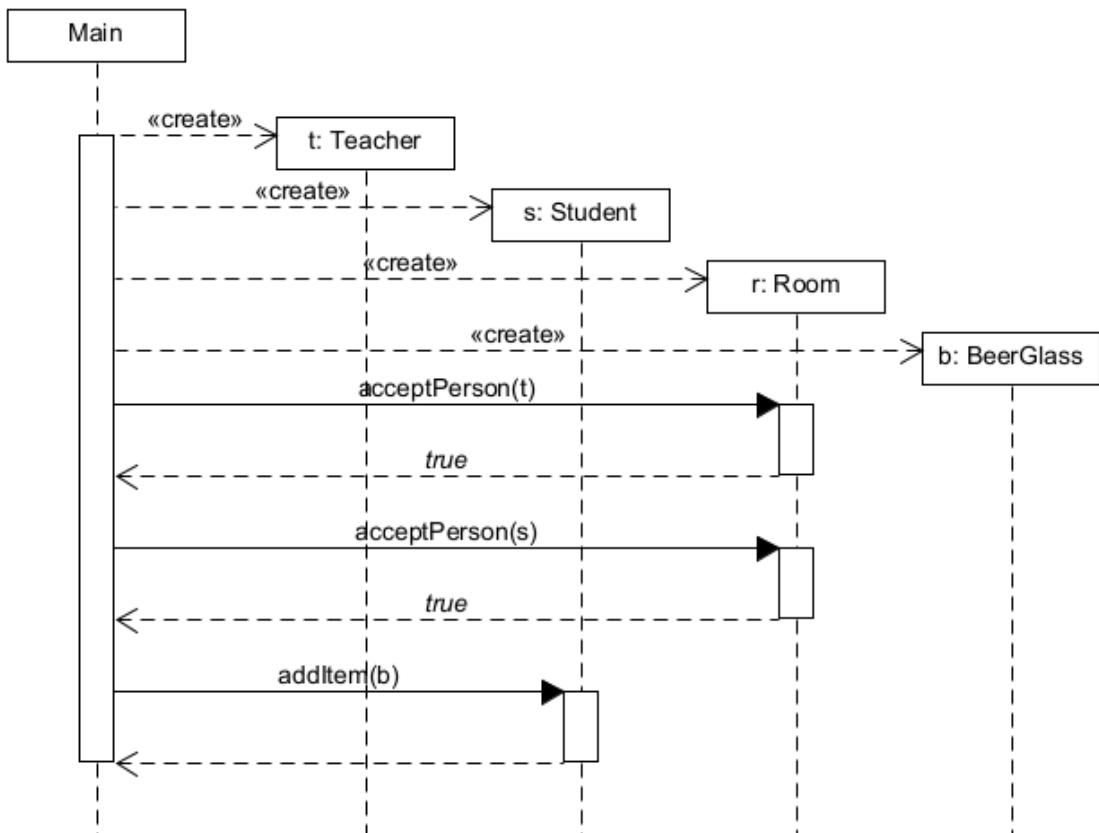
4.4.9 Maszk időtelés



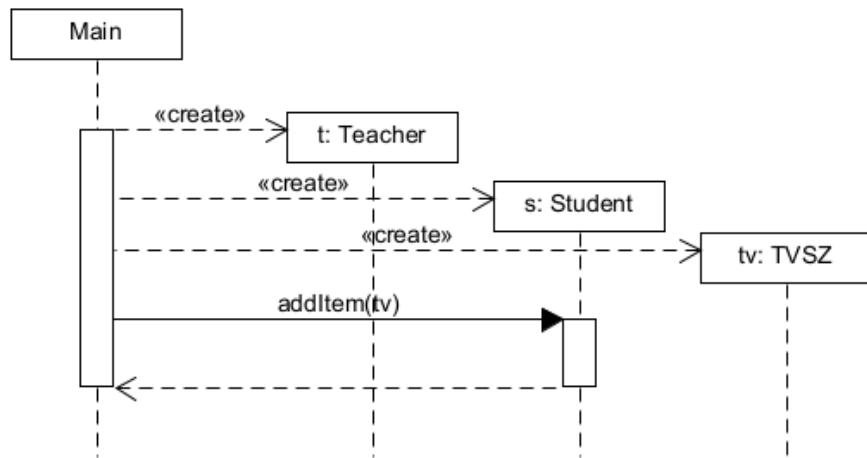
4.4.10 Táblatörlő rongy időtelés



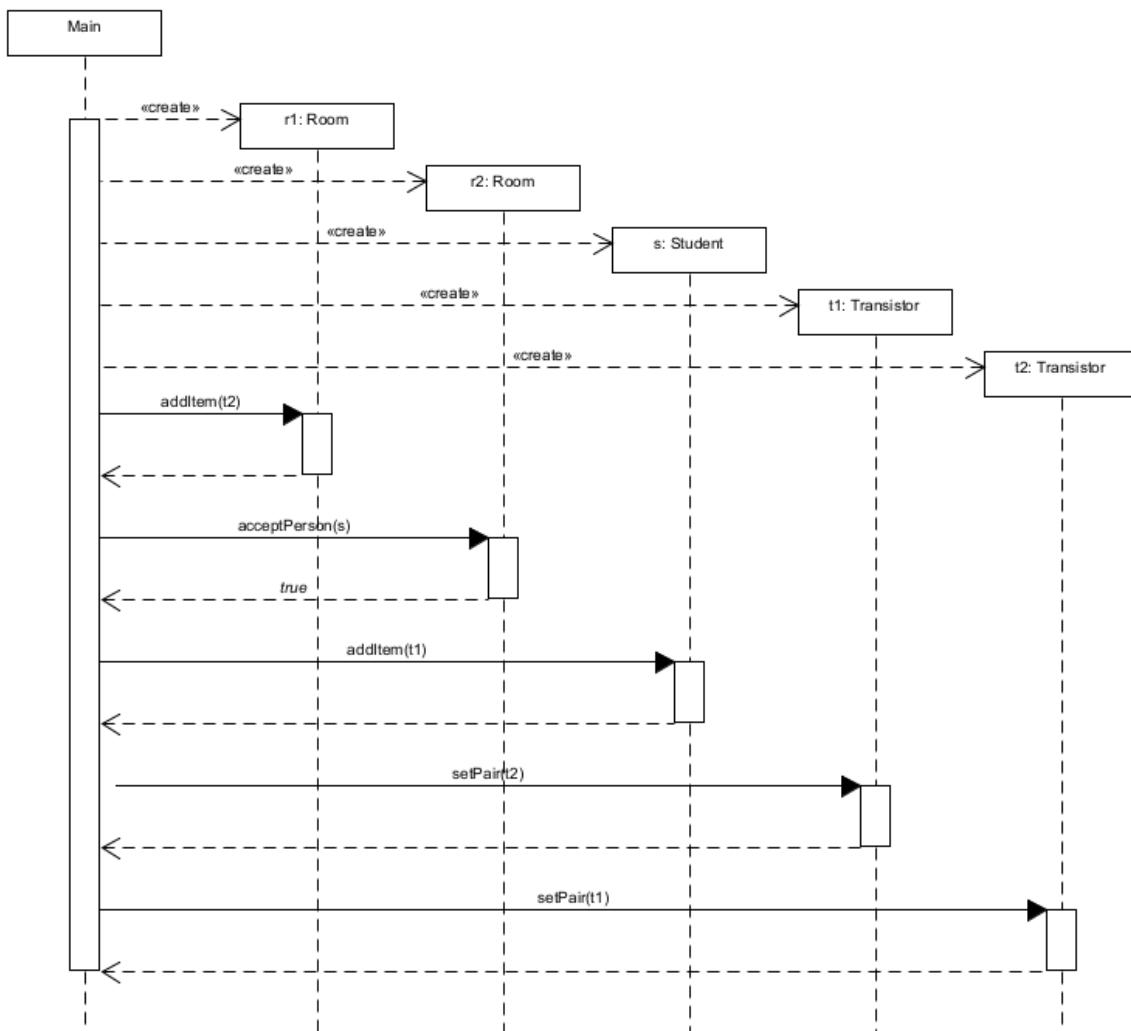
4.4.11 Söröspohár időtelés



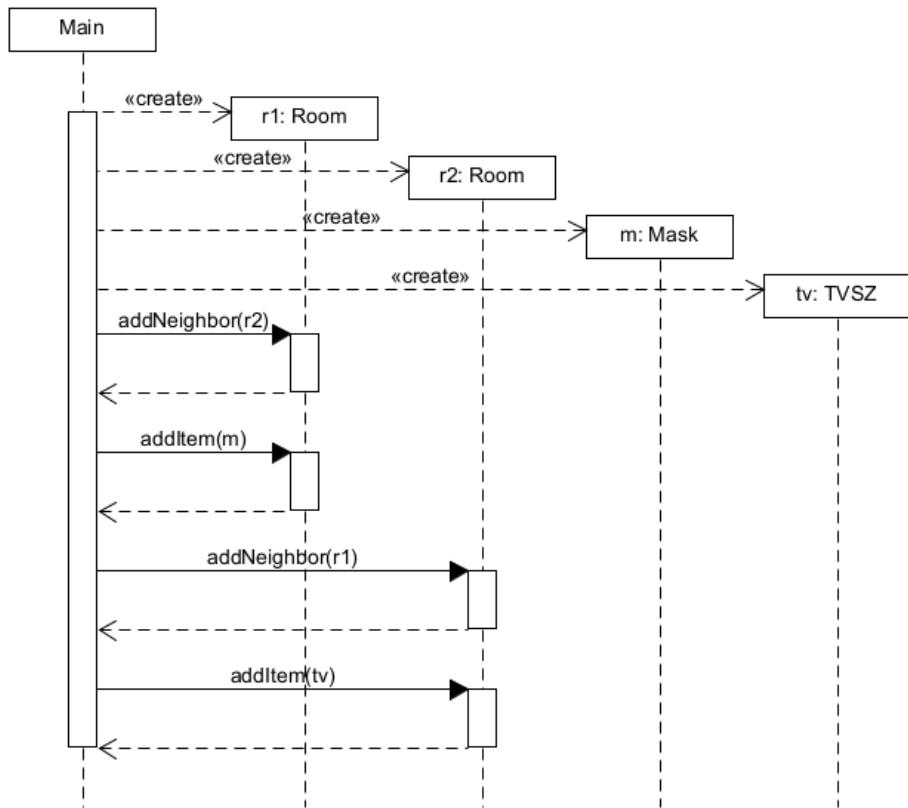
4.4.12 TVSZ használata



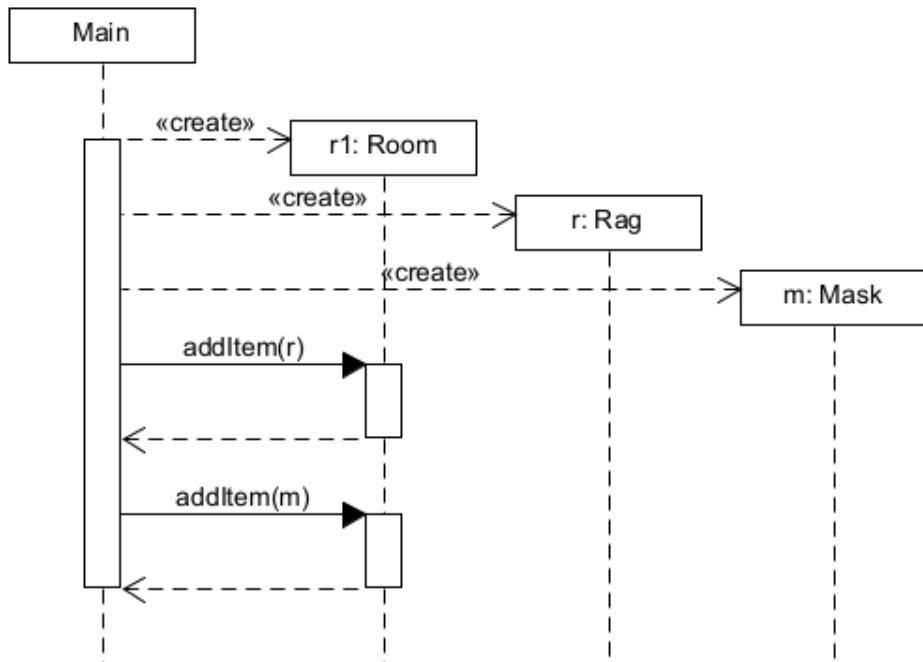
4.4.13 Tranzisztor használata teleportálásra



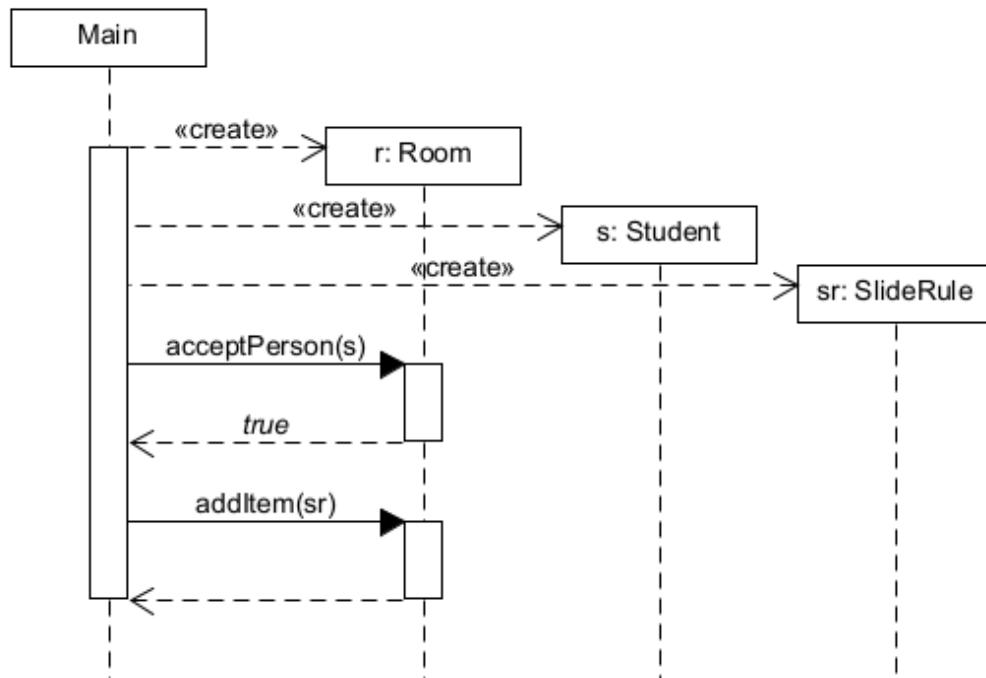
4.4.14 Szobák egyesítése



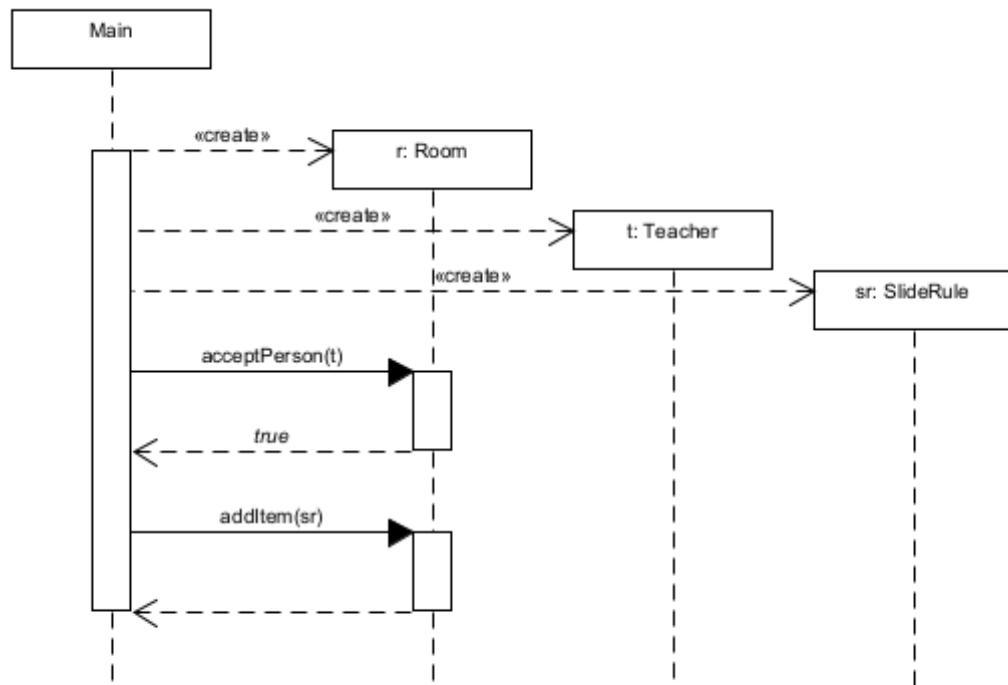
4.4.15 Szoba kettéválasztása



4.4.16 Hallgató felveszi a Logarlécet



4.4.17 Oktató felveszi a Logarlécet



4.5 Napló

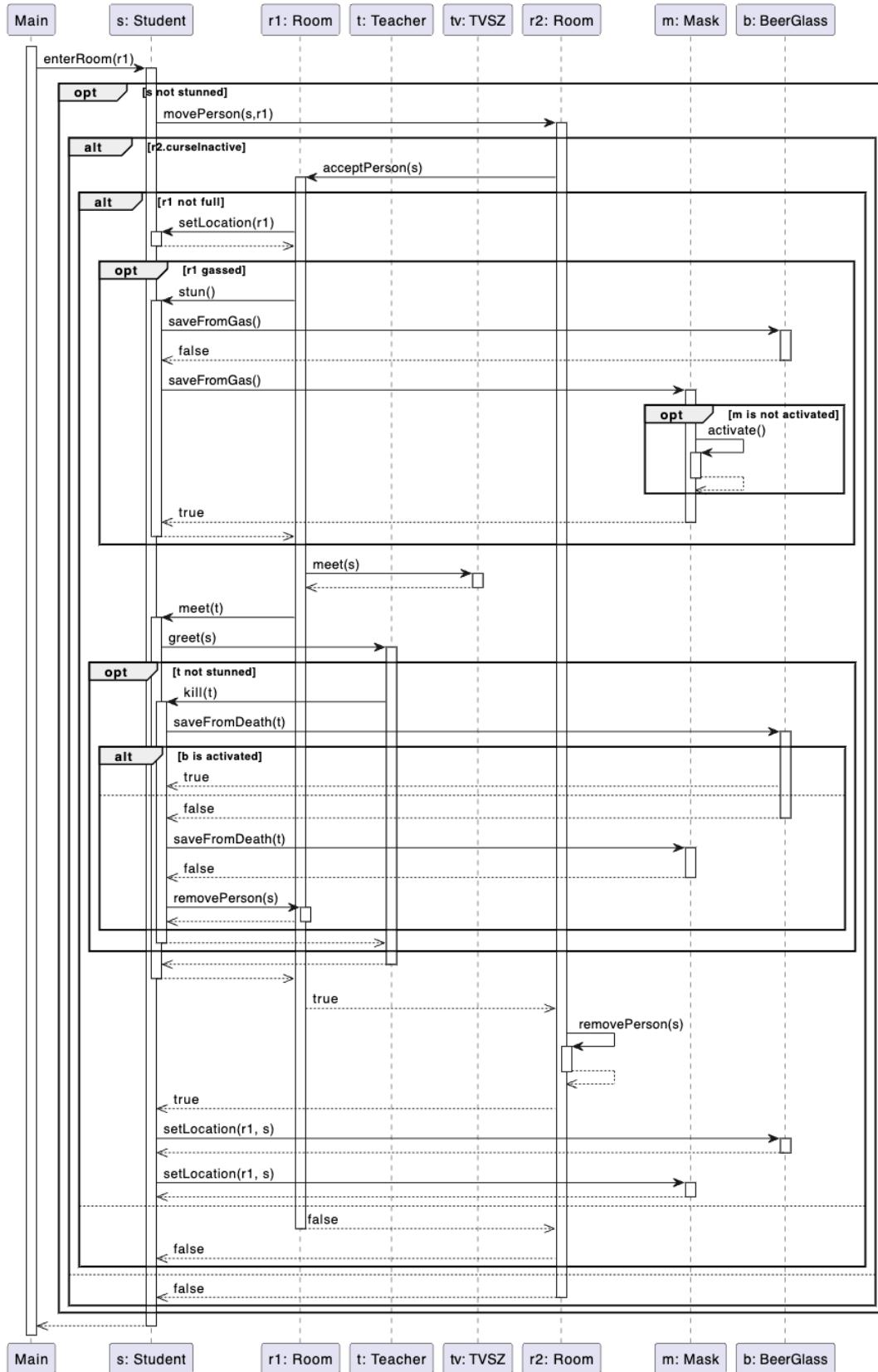
Kezdet	Időtartam	Résztvevők	Leírás
2024. 03. 13. 10:15	1,5 óra	Görömbey Héjja Sági Tömöri Vizhányó	Konzultáció
2024. 03. 14. 13:20	4 óra	Héjja	Use-case-ek megtervezése, use-case diagram létrehozása
2024. 03. 16. 11:30	2 óra	Vizhányó	Szekvencia diagramok készítése
2024. 03. 16. 17:30	4 óra	Vizhányó	Szekvencia diagramok készítése
2024. 03. 16. 22:45	2,5 óra	Héjja	Use-case-ek javítása, szkeleton bemenet-kimenet leírása, inicializációs diagramok készítése
2024. 03. 17.	0,5 óra	Tömöri	Use-case felülvizsgálat
2024. 03. 17. 11:00	4 óra	Vizhányó	Szekvencia diagramok módosítása, use case leírások pontosítása
2024.03.17. 13:00	1 óra	Sági	Kezelői felület tervének elkészítése
2024. 03. 17. 15:00	6 óra	Vizhányó	Szekvencia diagramok módosítása (split, merge, tranzisztor), use case leírások pontosítása
2024. 03. 17. 15:00	6 óra	Tömöri	Szobába lépés, sörösüveg használata, Logarléc felvétele, szekvencia diagramok elkészítése
2024. 03. 17. 16:00	6 óra	Sági	Tárgyak használata és időtelés szekvenciák készítése
2024. 03. 17. 16:00	6 óra	Görömbey	Szekvencia diagramok egyeztetése, dokumentum formázása, felülvizsgálás
2024. 03. 17. 18:00	6 óra	Héjja	Inicializáló szekvenciák frissítése, use-case leírásokhoz illesztés

5. Szkeleton beadás

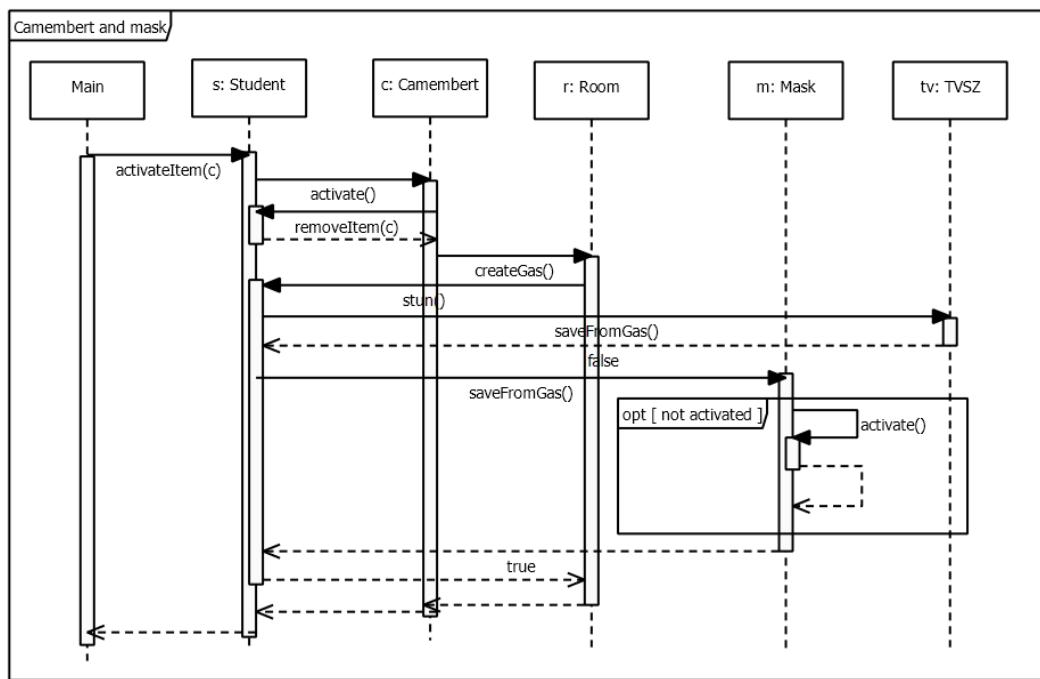
5.1 Szkeleton tervezési módosítások

5.1.1 Módosított lefutási szekvenciák

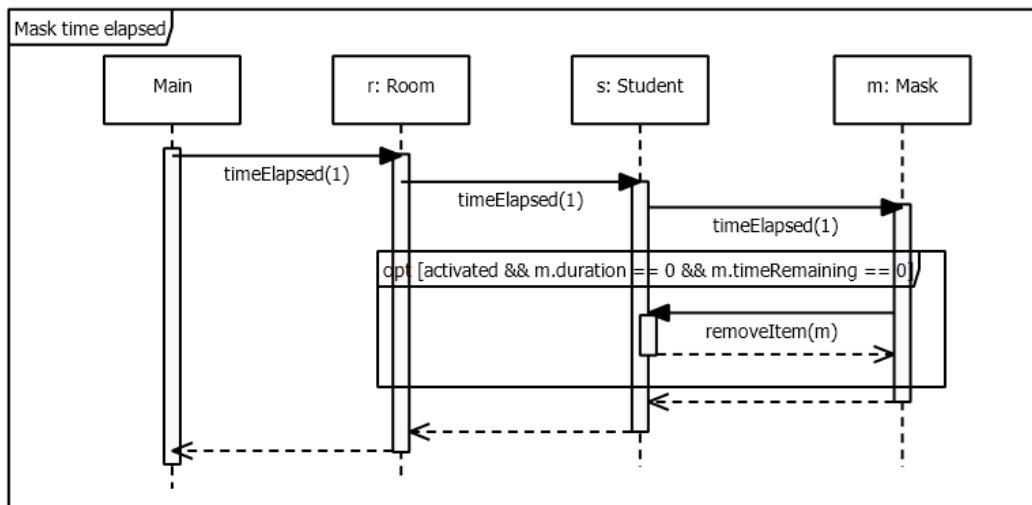
1 - Szobába átlépés



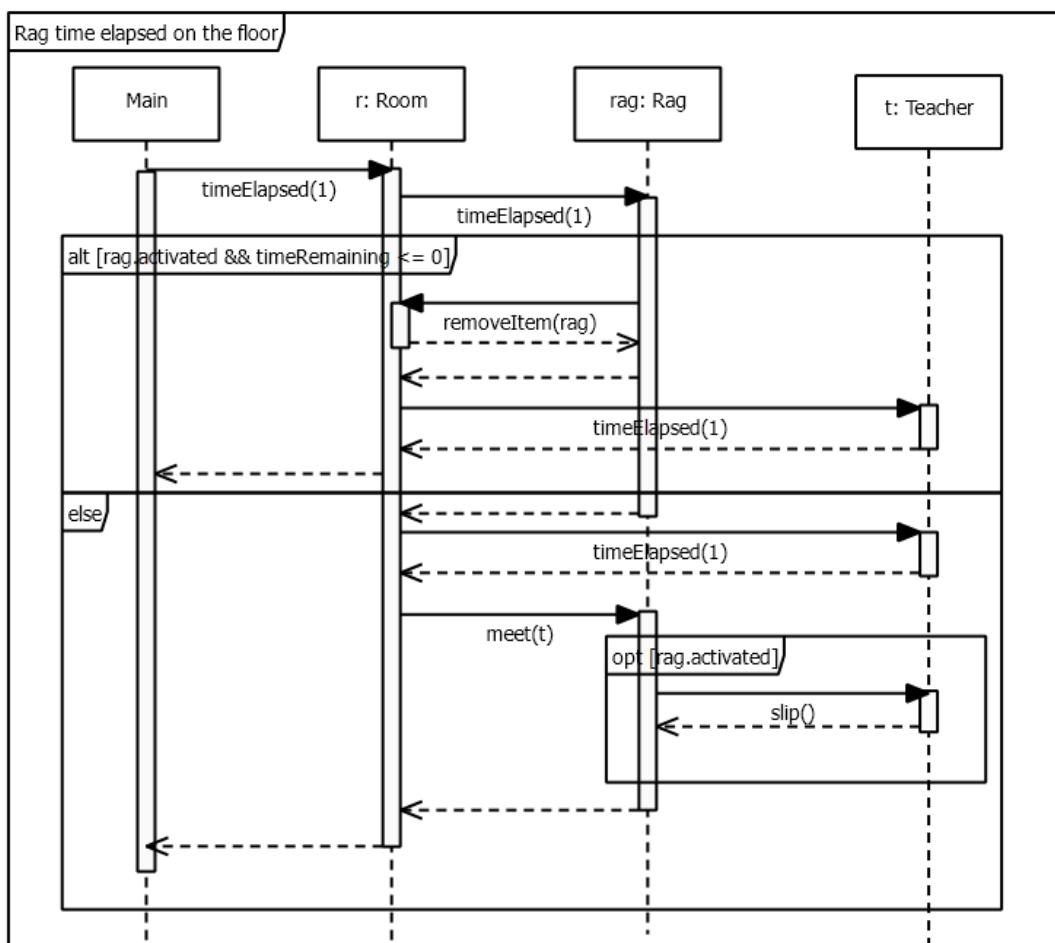
7 - Camembert és maszk használata



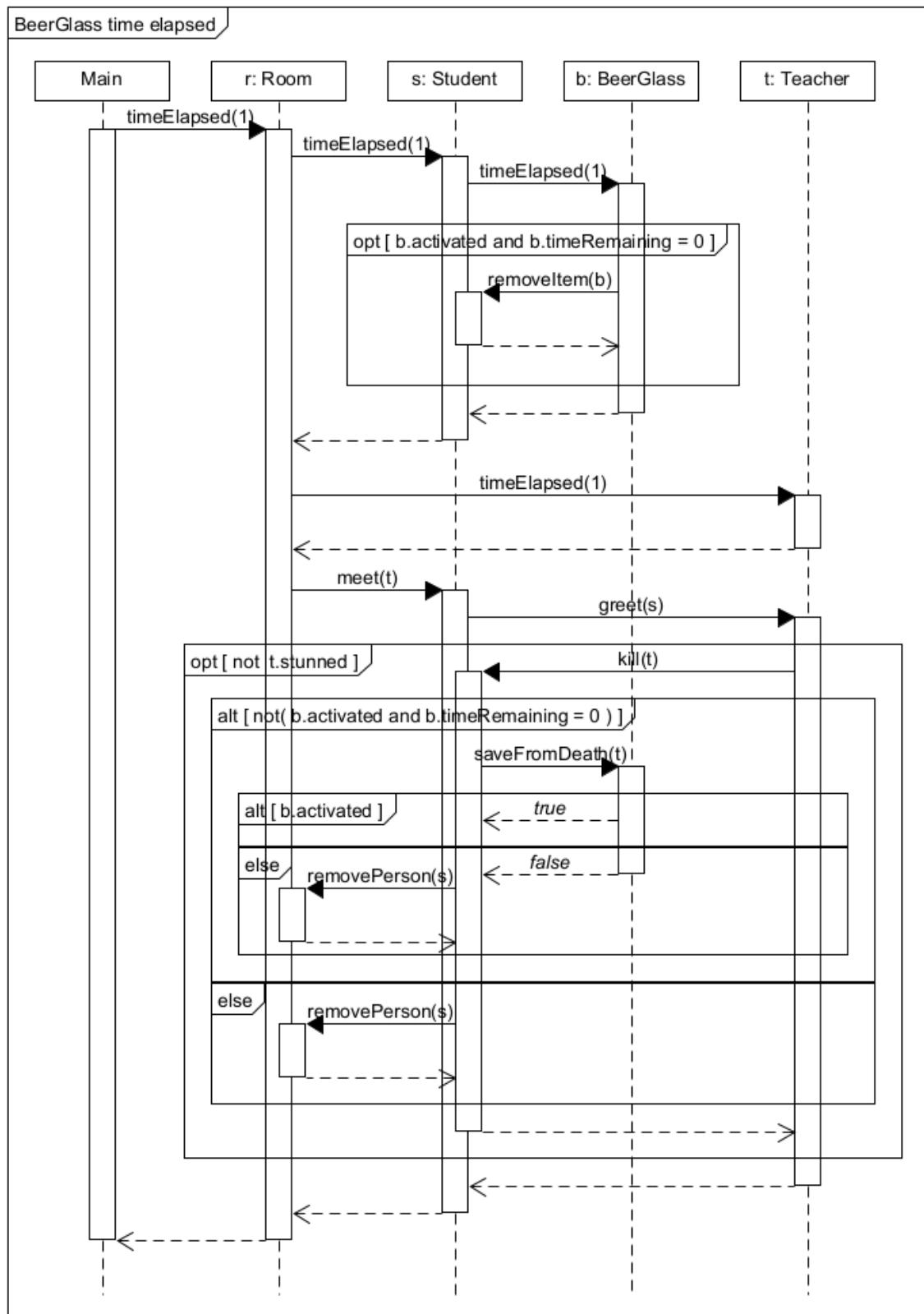
9 - Maszk időtelés



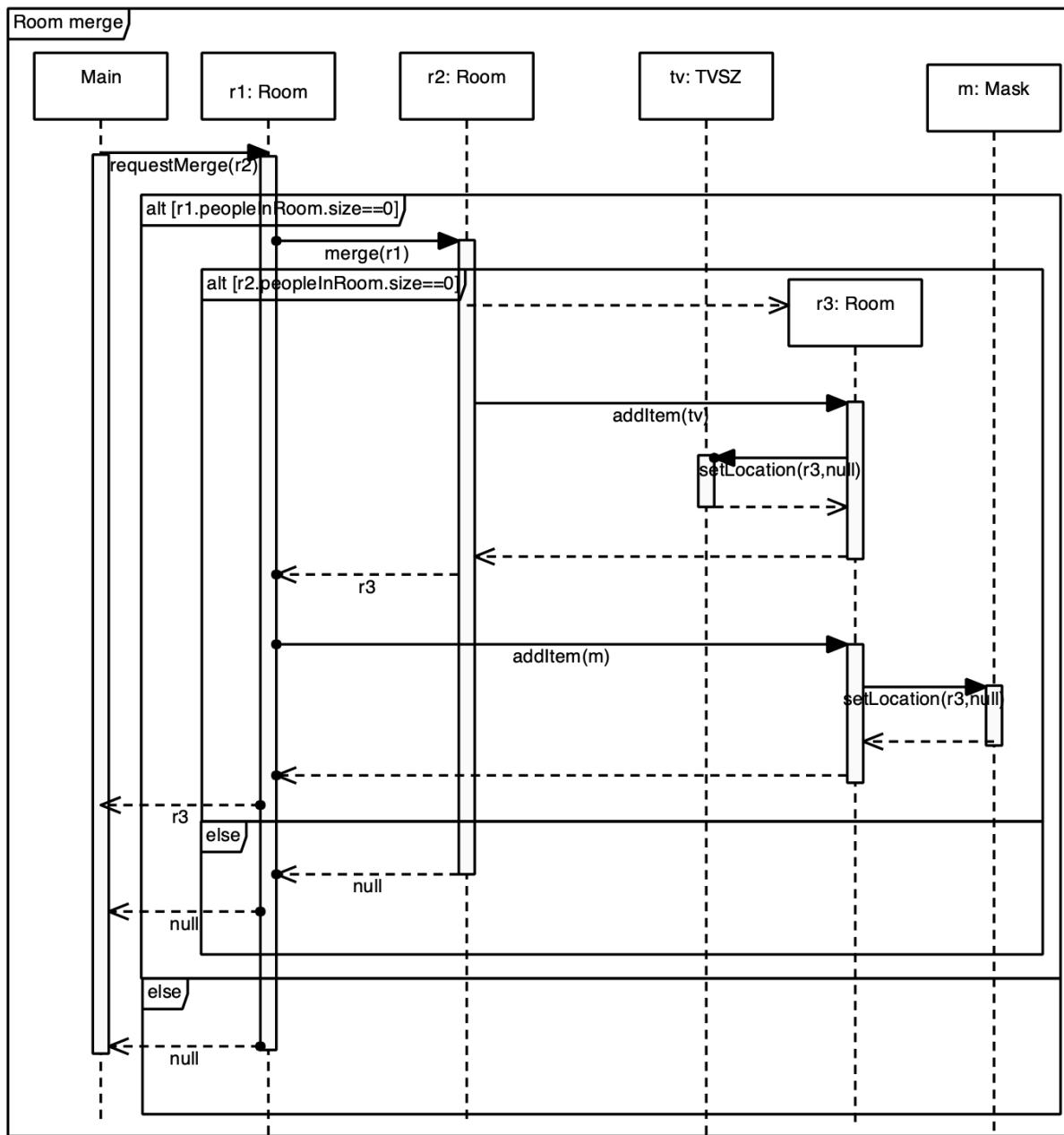
10 - Táblatörő rongy időtelés



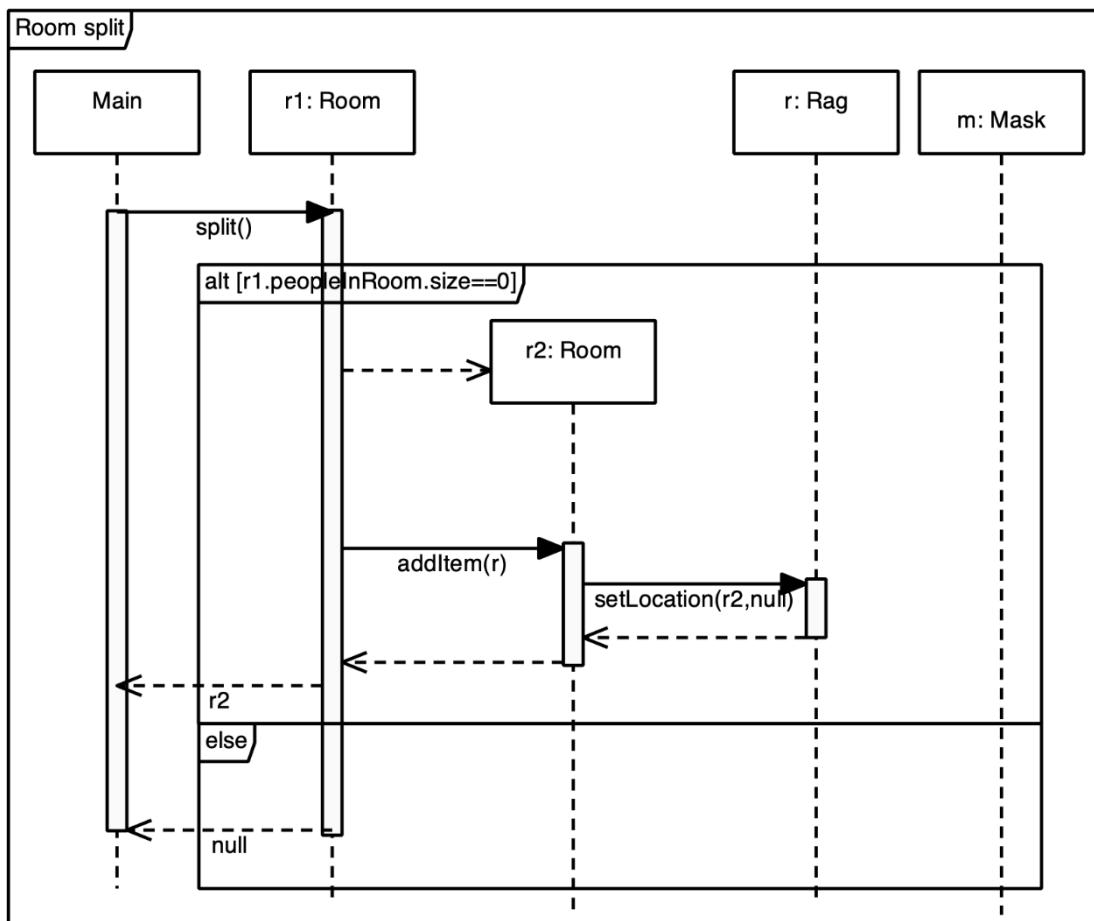
11 – Söröspohár időtelés



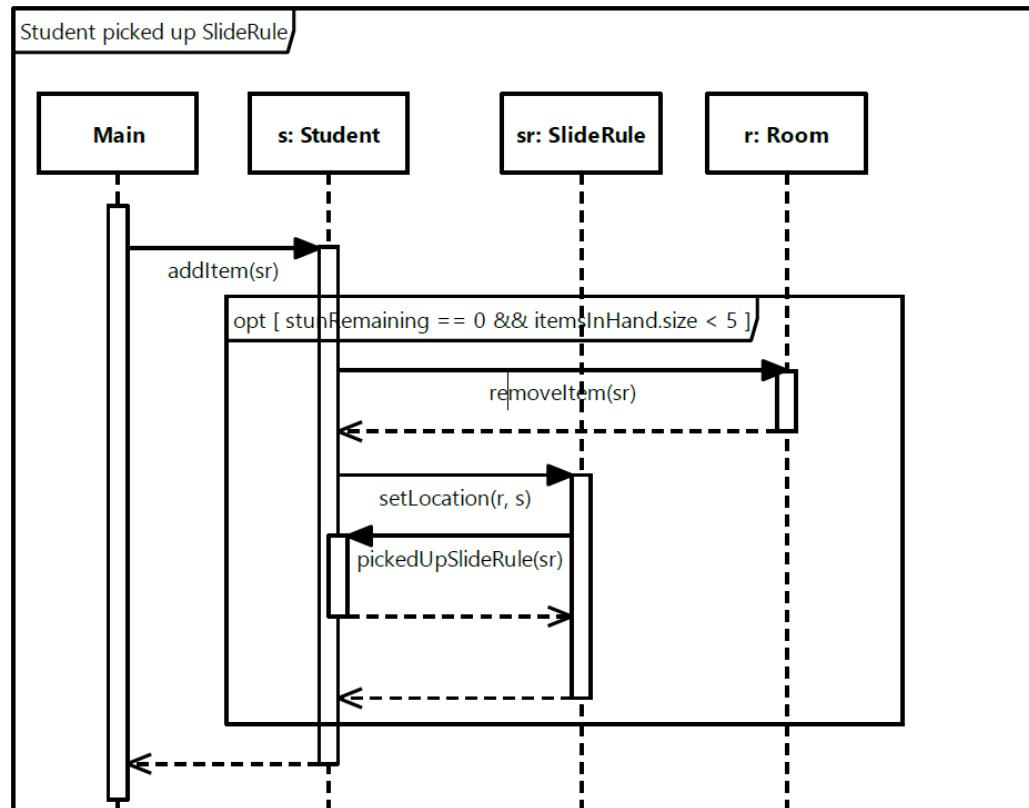
14 - Szobák egyesítése



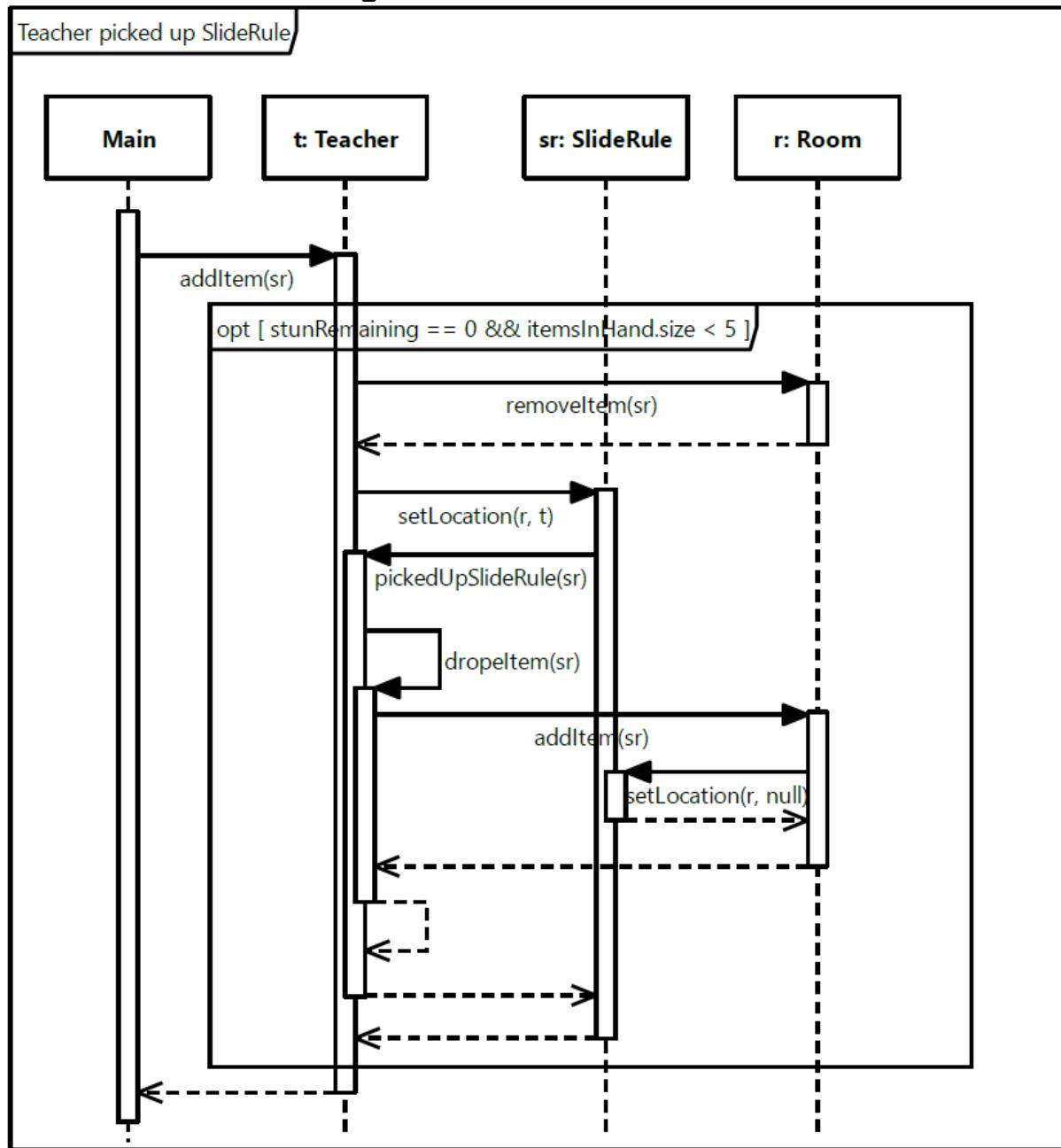
15 - Szoba kettéválása



16 - Hallgató felveszi a Logarlécet

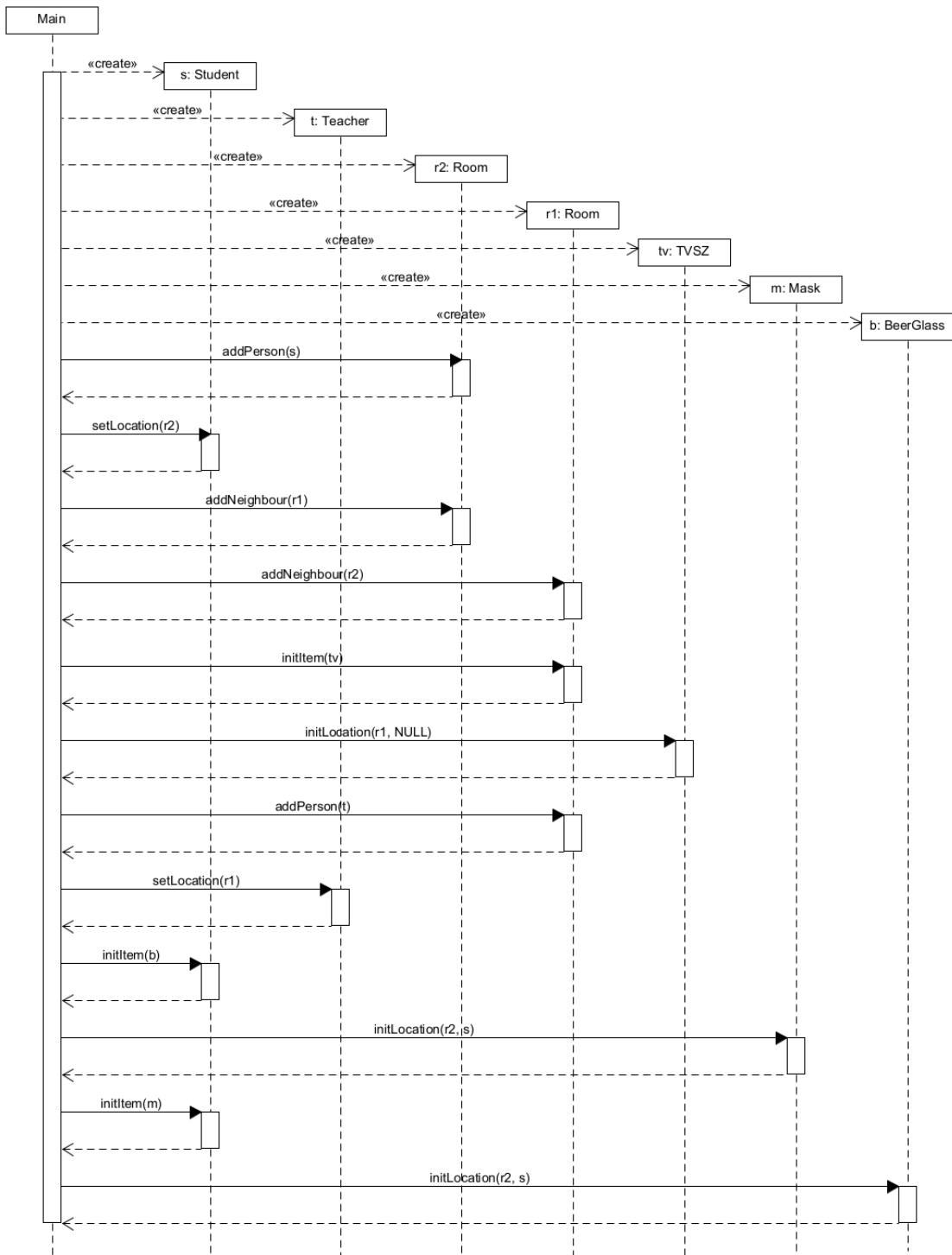


17 – Oktató felveszi a Logarlécet

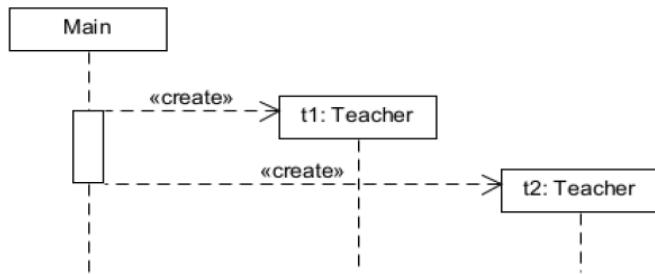


5.1.2 Inicializáló szekvenciák

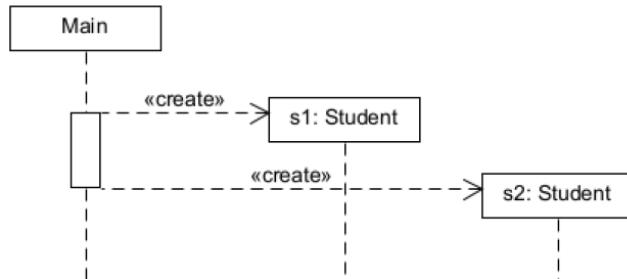
5.1.2.1 Szobába átlépés



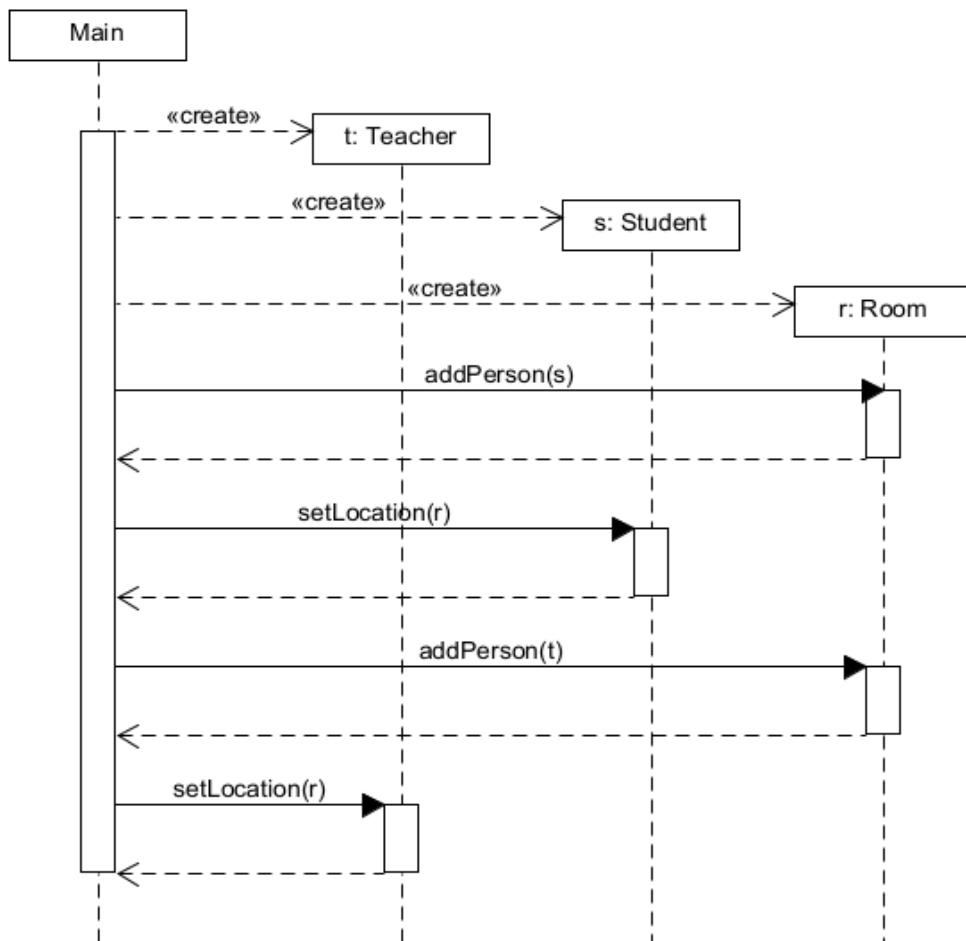
5.1.2.2 Két oktató találkozik



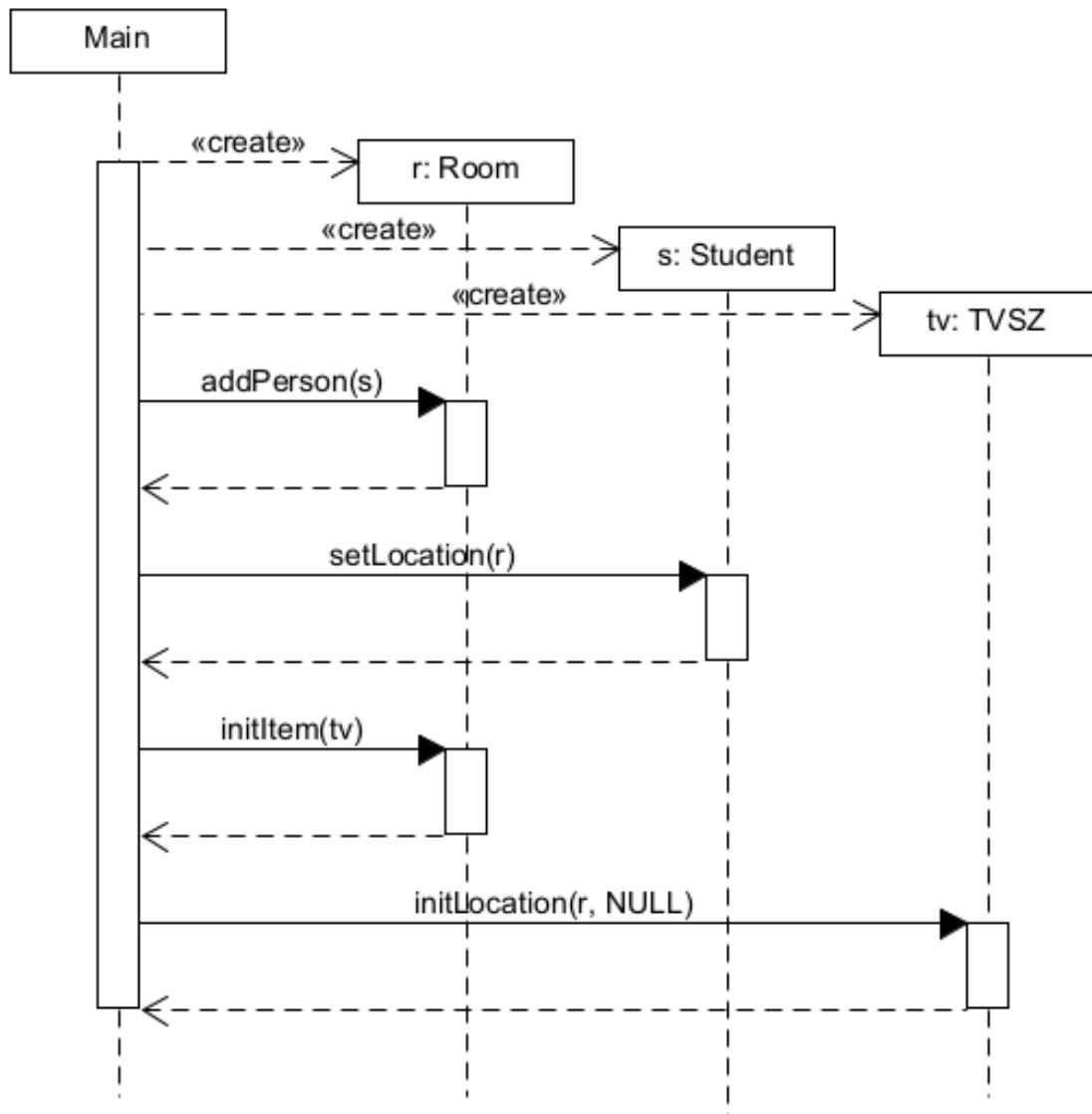
5.1.2.3 Két hallgató találkozik



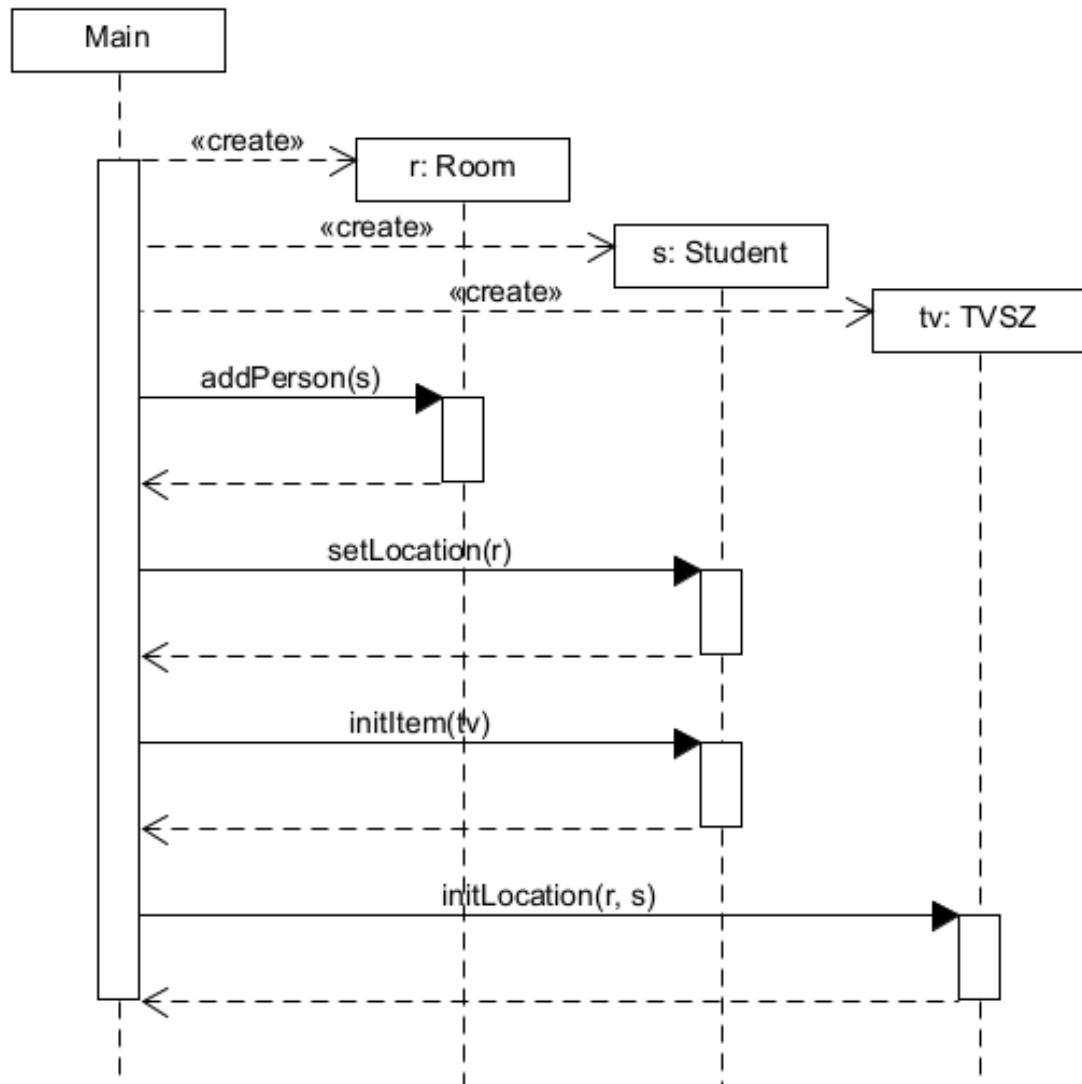
5.1.2.4 Oktató találkozik hallgatóval



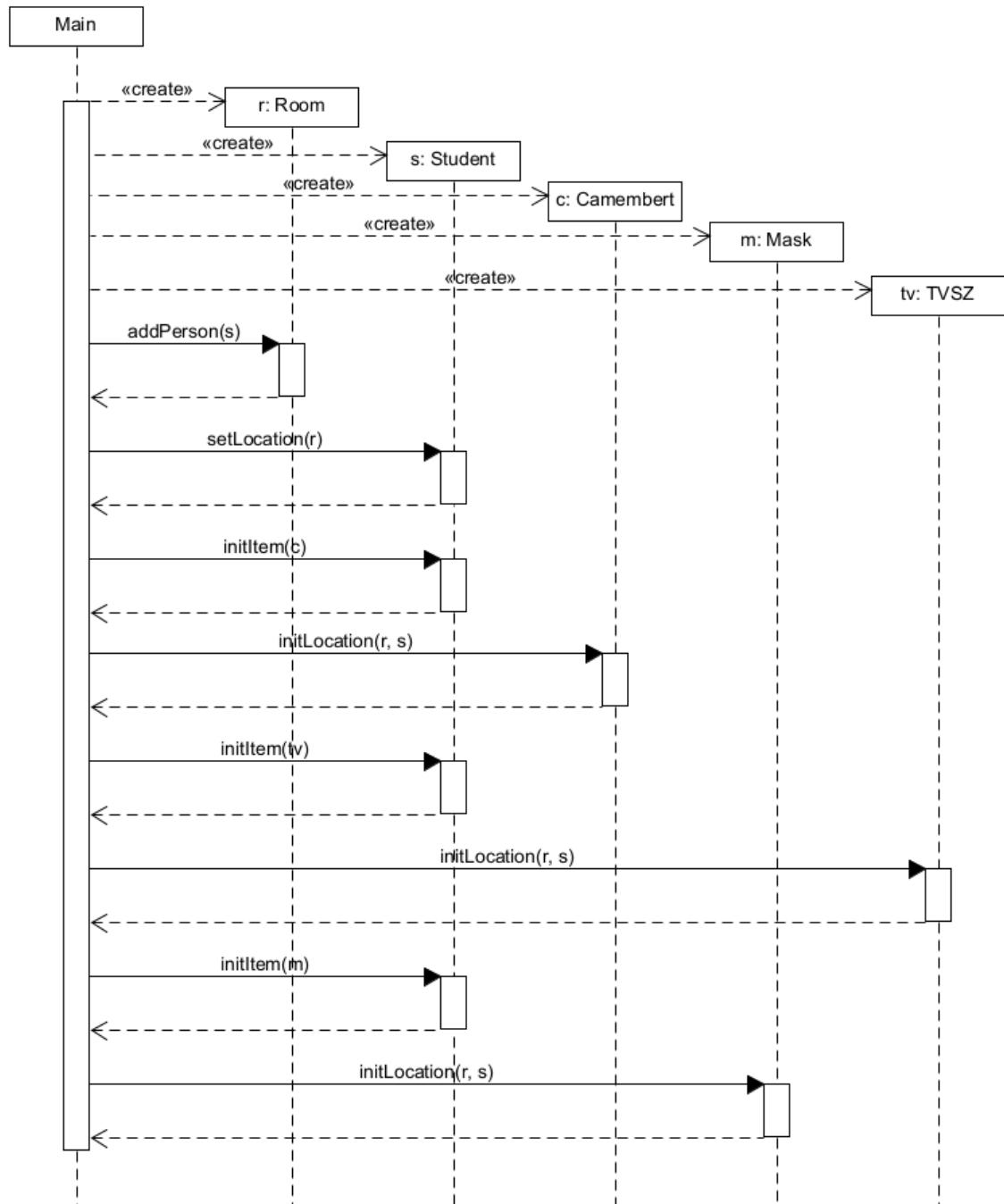
5.1.2.5 Tárgyfelvétel



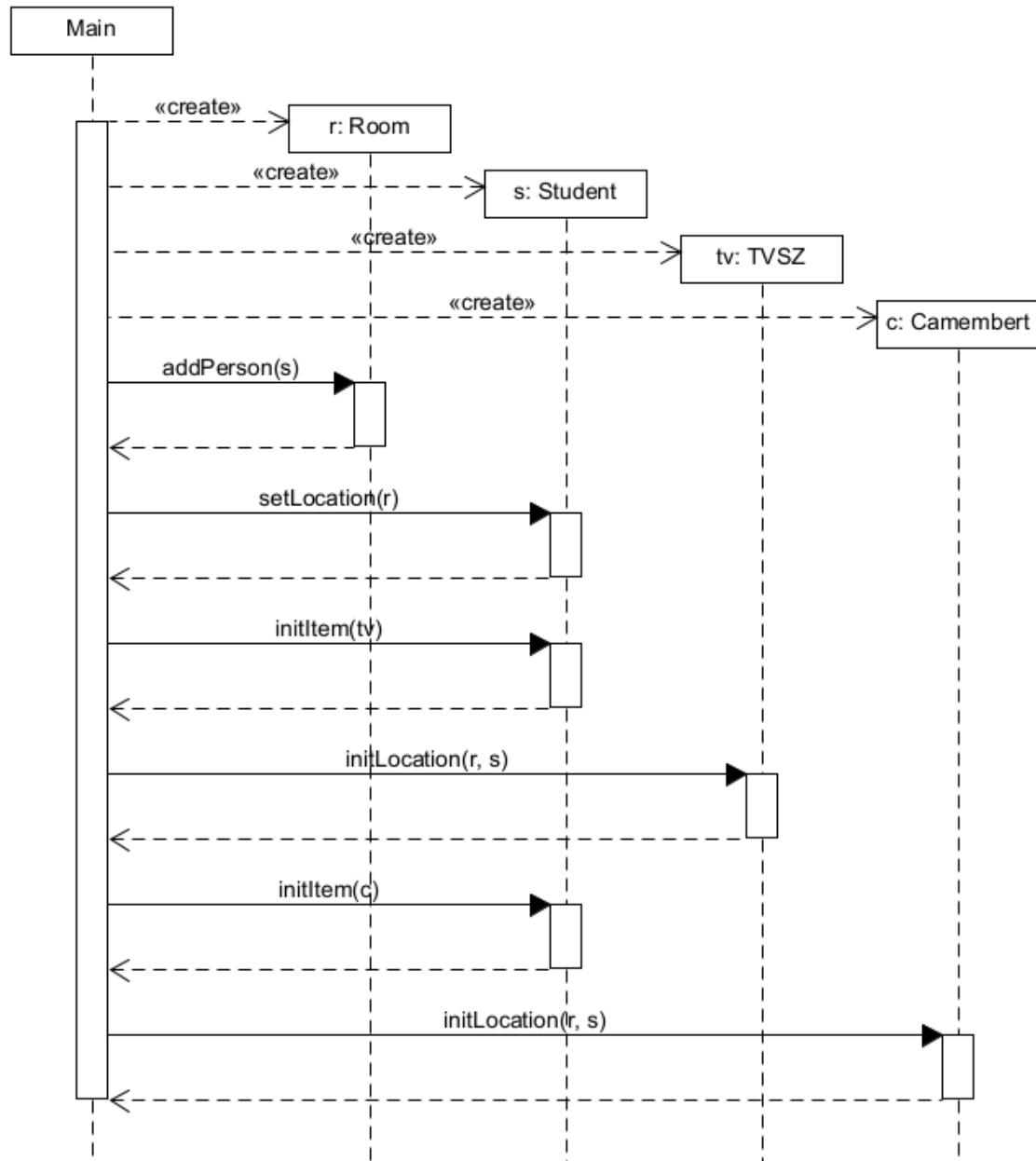
5.1.2.6 Tárgylerakás



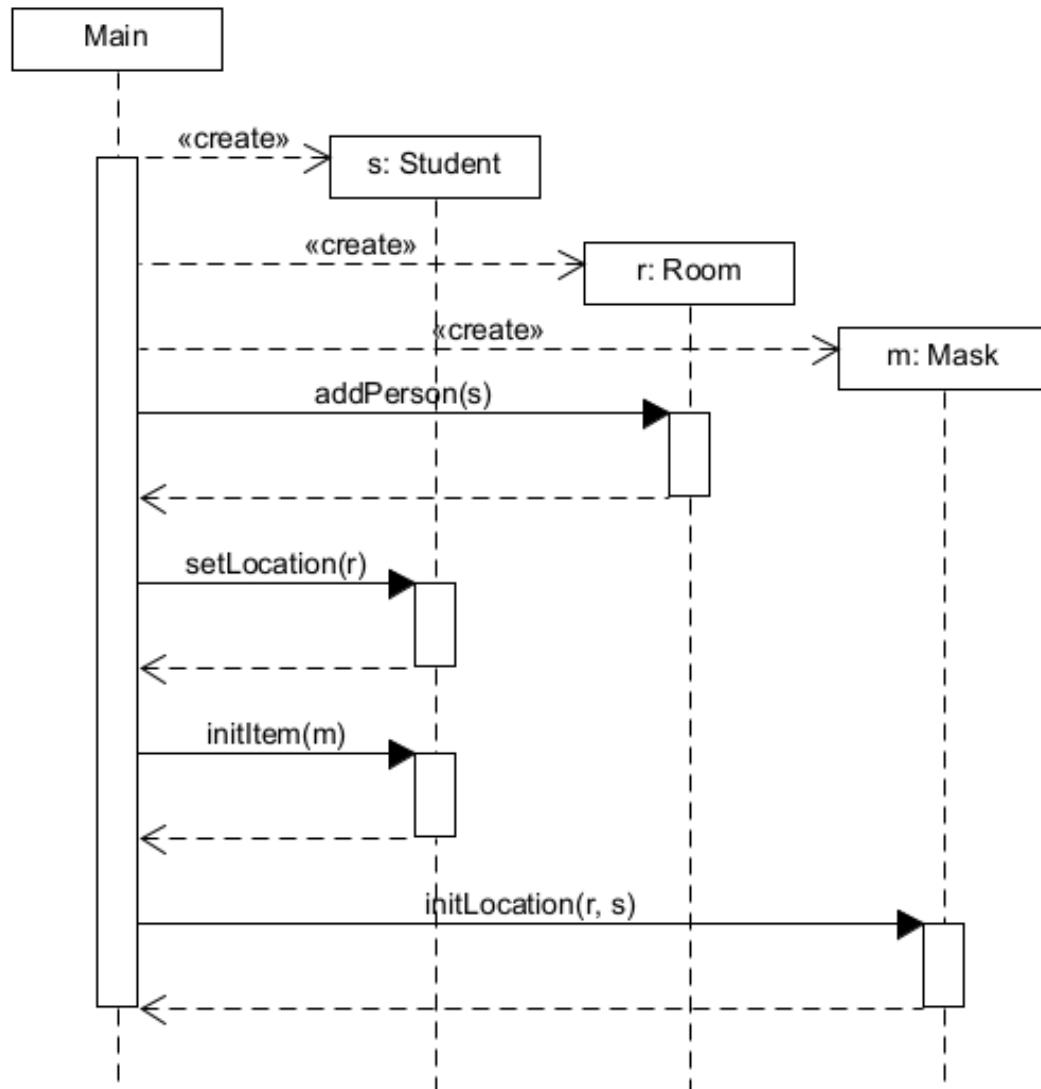
5.1.2.7 Szoba elgázosítás



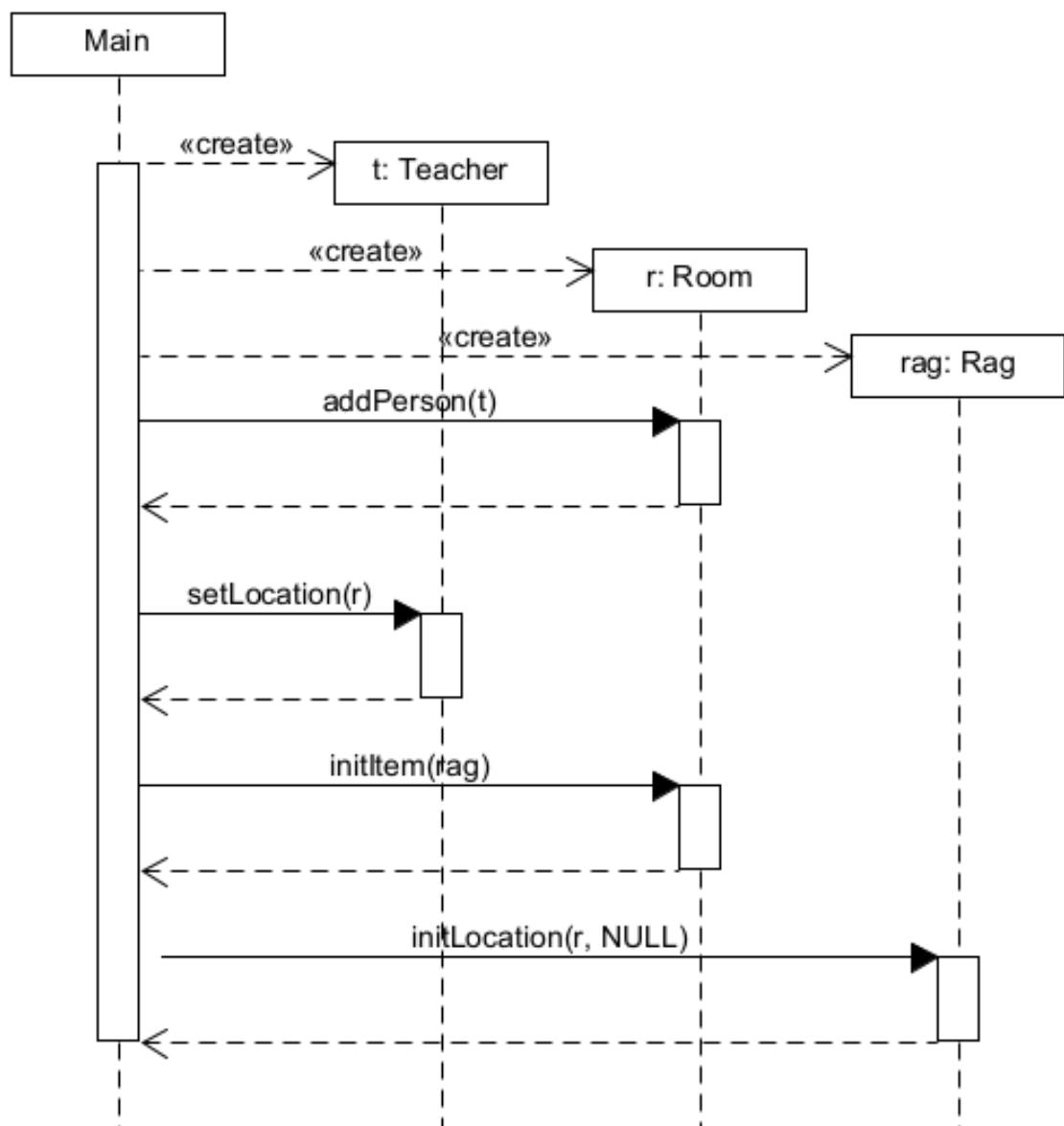
5.1.2.8 Hallgató elkábul



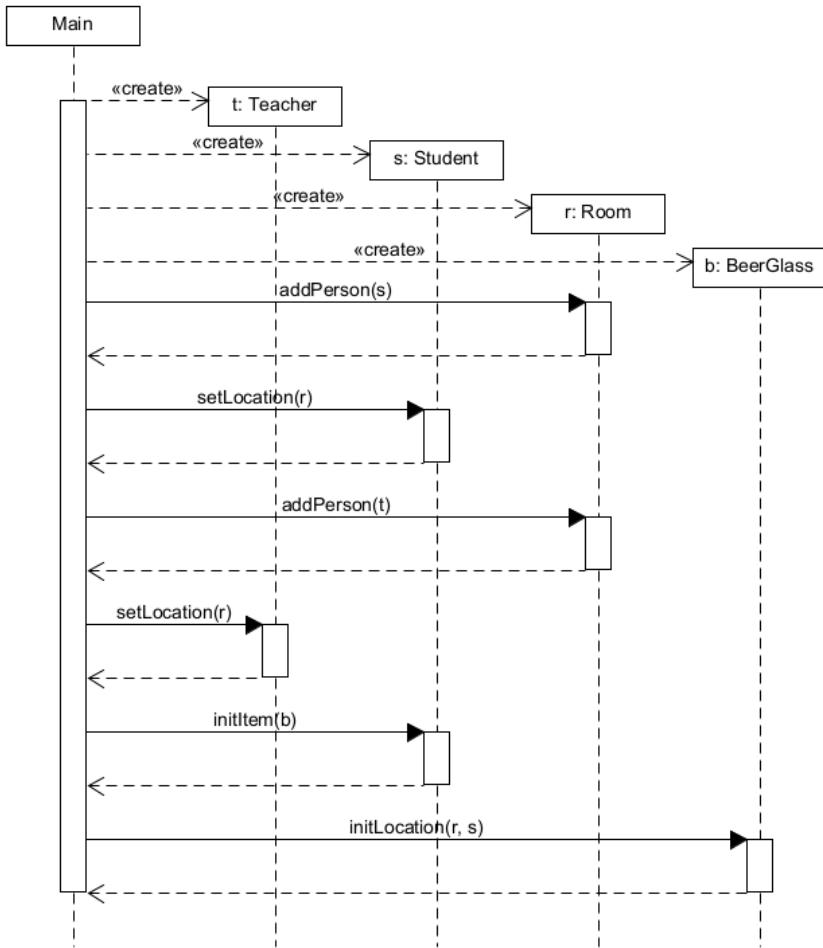
5.1.2.9 Maszk időtelés



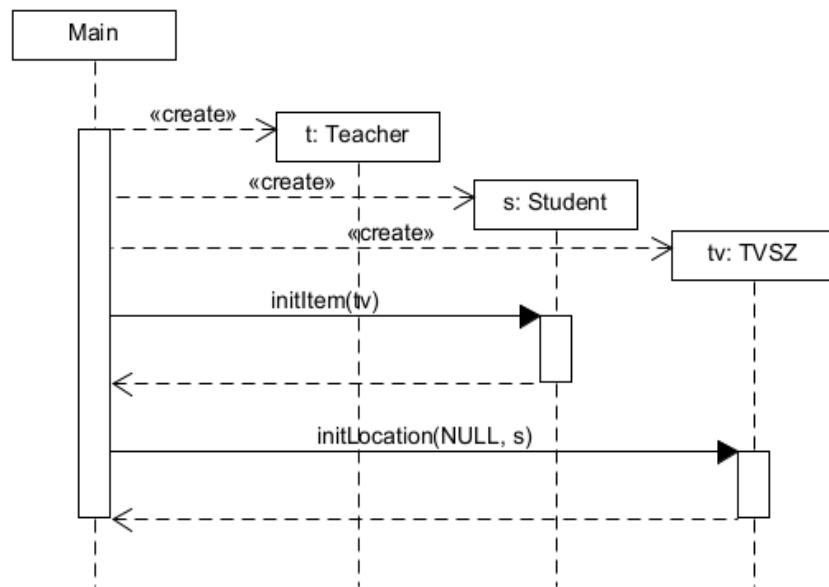
5.1.2.10 Táblatörlő rongy időtelés



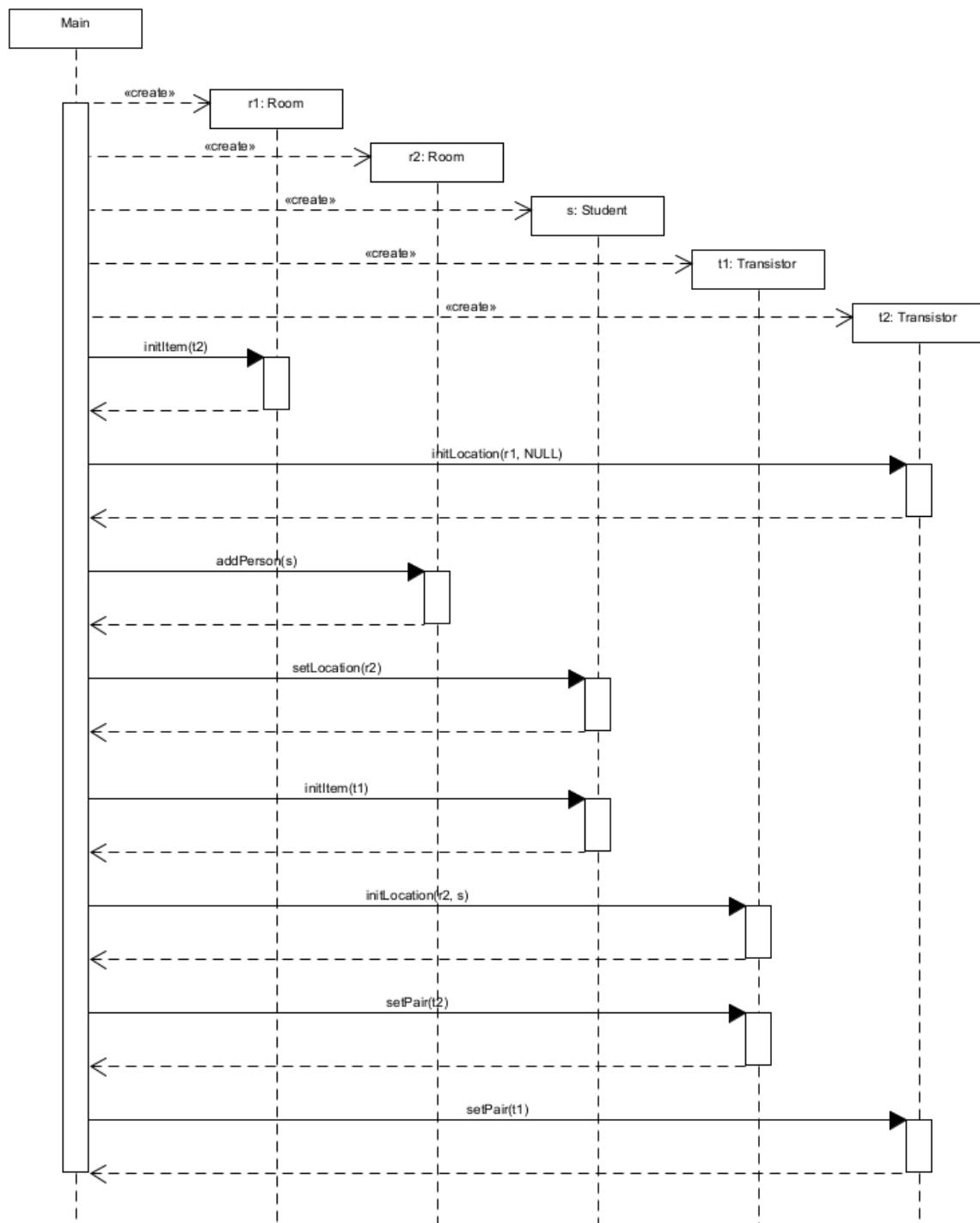
5.1.2.11 Söröspohár időtelés



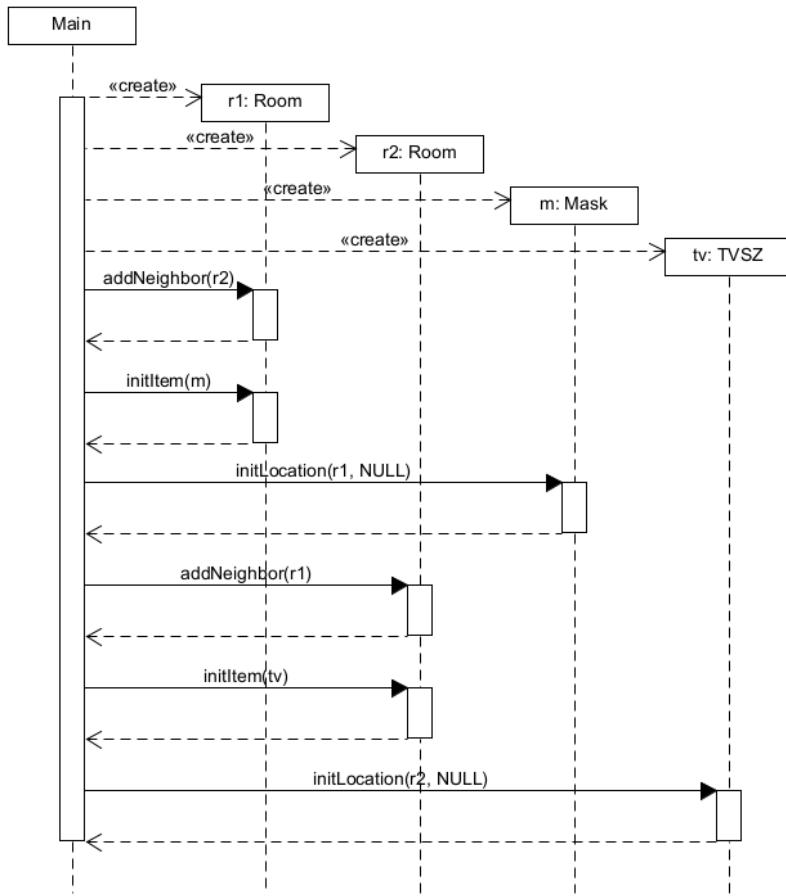
5.1.2.12 TVSZ használata



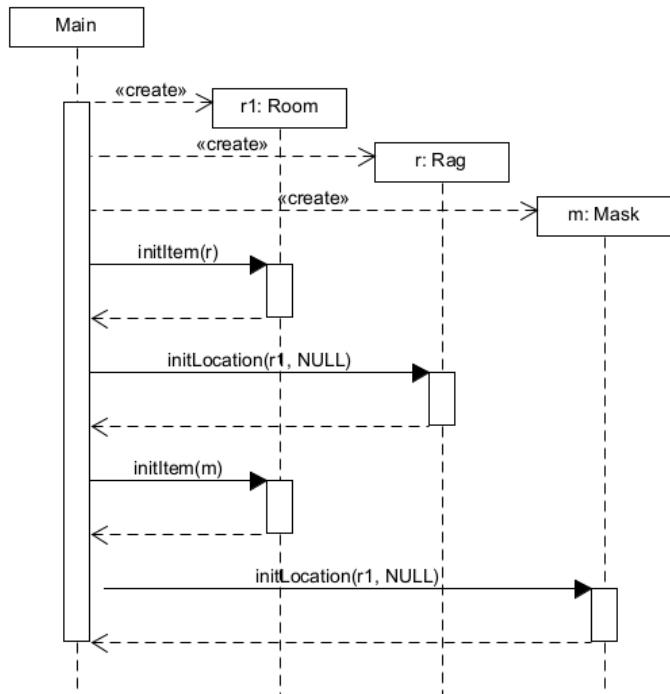
5.1.2.13 Tranzisztor használata teleportálásra



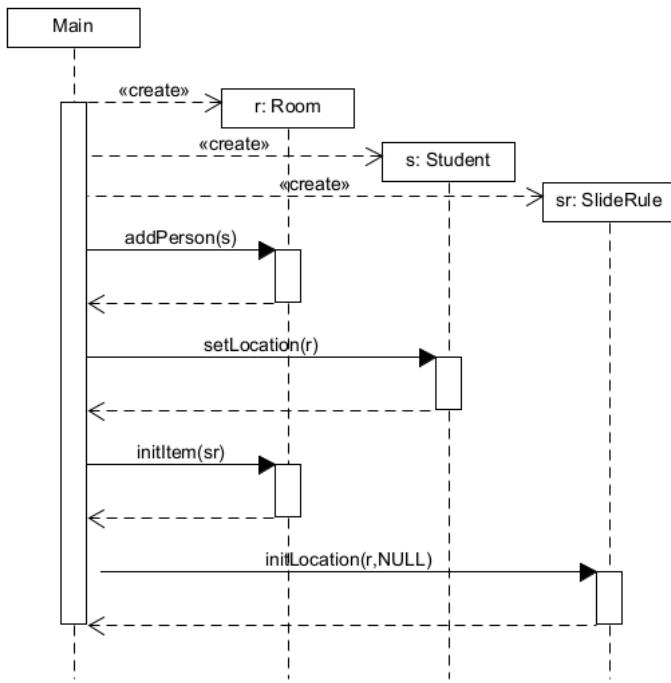
5.1.2.14 Szobák egyesítése



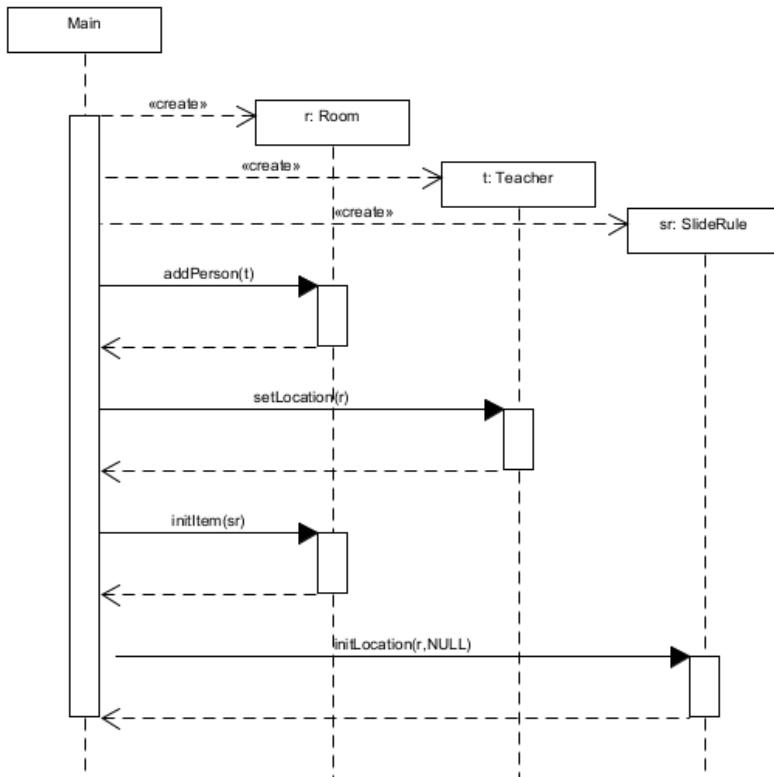
5.1.2.15 Szoba kettéválasztása



5.1.2.16 Hallgató felveszi a Logarlécet



5.1.2.17 Oktató felveszi a Logarlécet



5.2 Fordítási és futtatási útmutató

5.2.1 Fájllista

Fájl neve	Méret (kB)	Keletkezés ideje	Tartalom
BeerGlass.java	4	2024. 03. 23. 21:25	BeerGlass osztály működésének logikája
Camembert.java	3	2024. 03. 23. 21:25	Camembert osztály működésének logikája
IntervalItem.java	1	2024. 03. 23. 21:25	IntervalItem osztály működésének logikája
Item.java	3	2024. 03. 23. 21:25	Item osztály működésének logikája
ItemHandler.java	1	2024. 03. 23. 21:25	ItemHandler interface deklarációja
ITest.java	1	2024. 03. 23. 21:25	ITest interface deklarációja
Logger.java	8	2024. 03. 23. 21:25	Logger osztály működésének logikája
Main.java	4	2024. 03. 23. 21:25	Main osztály, a program innen indul
Mask.java	4	2024. 03. 23. 21:25	Mask osztály működésének logikája
Person.java	6	2024. 03. 23. 21:25	Person osztály működésének logikája
Rag.java	4	2024. 03. 23. 21:25	Rag osztály működésének logikája
Room.java	11	2024. 03. 23. 21:25	Room osztály működésének logikája
SlideRule.java	3	2024. 03. 23. 21:25	SlideRule osztály működésének logikája
Student.java	3	2024. 03. 23. 21:25	Student osztály működésének logikája
Teacher.java	3	2024. 03. 23. 21:25	Teacher osztály működésének logikája
Test1.java	2	2024. 03. 23. 21:25	Az 1. teszt kódja
Test10.java	1	2024. 03. 23. 21:25	A 10. teszt kódja
Test11.java	2	2024. 03. 23. 21:25	A 11. teszt kódja
Test12.java	1	2024. 03. 23. 21:25	A 12. teszt kódja
Test13.java	2	2024. 03. 23. 21:25	A 13. teszt kódja
Test14.java	2	2024. 03. 23. 21:25	A 14. teszt kódja
Test15.java	1	2024. 03. 23. 21:25	A 15. teszt kódja
Test16.java	1	2024. 03. 23. 21:25	A 16. teszt kódja
Test17.java	1	2024. 03. 23. 21:25	A 17. teszt kódja
Test2.java	1	2024. 03. 23. 21:25	A 2. teszt kódja
Test3.java	1	2024. 03. 23. 21:25	A 3. teszt kódja
Test4.java	1	2024. 03. 23. 21:25	A 4. teszt kódja
Test5.java	1	2024. 03. 23. 21:25	Az 5. teszt kódja
Test6.java	1	2024. 03. 23. 21:25	A 6. teszt kódja
Test7.java	2	2024. 03. 23. 21:25	A 7. teszt kódja

Test8.java	2	2024. 03. 23. 21:25	A 8. teszt kódja
Test9.java	1	2024. 03. 23. 21:25	A 9. teszt kódja
TimeSensitive.java	1	2024. 03. 23. 21:25	TimeSensitive interface deklarációja
Transistor.java	5	2024. 03. 23. 21:25	Transistor osztály működésének logikája
TVSZ.java	3	2024. 03. 23. 21:25	TVSZ osztály működésének logikája

5.2.2 Fordítás

A szoftver x86-os architektúrán, Windows operációs rendszer alatt fordítható. Pontosabban a forrásprogramnak a kari felhőben (<https://niif.cloud.bme.hu/> vagy <https://fured.cloud.bme.hu/>), a „Windows 10 20H2 - JDK-Eclipse-WSU” sablonnal meghatározott környezetben fordítható és futtatható. Ehhez belépési segédlet itt található: <https://www.mit.bme.hu/node/11462>. Egy ilyen típusú virtuális gépre csatlakozás után másolja fel (pl.: vágólapon keresztül) a rendelkezésre álló *skeleton.zip*-ből kicsomagolt *skeleton* mappát.

Megjegyzés: Ha a kicsomagolás során egy új, a zip-pel megegyező mappába teszi a fájlokat, akkor kettő skeleton mappa lesz (az egyikben lesz a másik, ami eredetileg a zip-ben volt). Ebben az esetben, ha *skeleton* mappáról beszélünk, a struktúrában lejjebb lévő mappát értjük.

A kód fordítását a *skeleton* mappában megnyitott cmd shellben teheti meg a következő parancssal:

```
javac -encoding utf8 -d ./out/ ./src/main/java/model/*.java ./src/main/java/testing/*.java  
./src/main/java/Main.java
```

5.2.3 Futtatás

A szoftver x86-os architektúrán, Windows operációs rendszer alatt futtatható. Pontosabban a forrásprogramnak a kari felhőben (<https://niif.cloud.bme.hu/> vagy <https://fured.cloud.bme.hu/>), a „Windows 10 20H2 - JDK-Eclipse-WSU” sablonnal meghatározott környezetben fordítható és futtatható. Ehhez belépési segédlet itt található: <https://www.mit.bme.hu/node/11462>. Amennyiben egy ilyen gépen elvégezte az előző pontban (6.1.2 Fordítás) megadott lépéseket és fordította a programfájlokat, a program futtatható a *skeleton* mappában megnyitott cmd shellben a következő parancssal:

```
java -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -cp ./out/ Main
```

5.3 Értékelés

Tag neve	Tag neptun	Munka százalékban és Aláírás
Görömbey László	I3B5JU	20
Héjja Márton	RECW35	20
Sági Péter Pál	KQF28D	20
Tömöri Péter András	I4RZ0O	20
Vizhányó Miklós Ferenc	NVY1AG	20

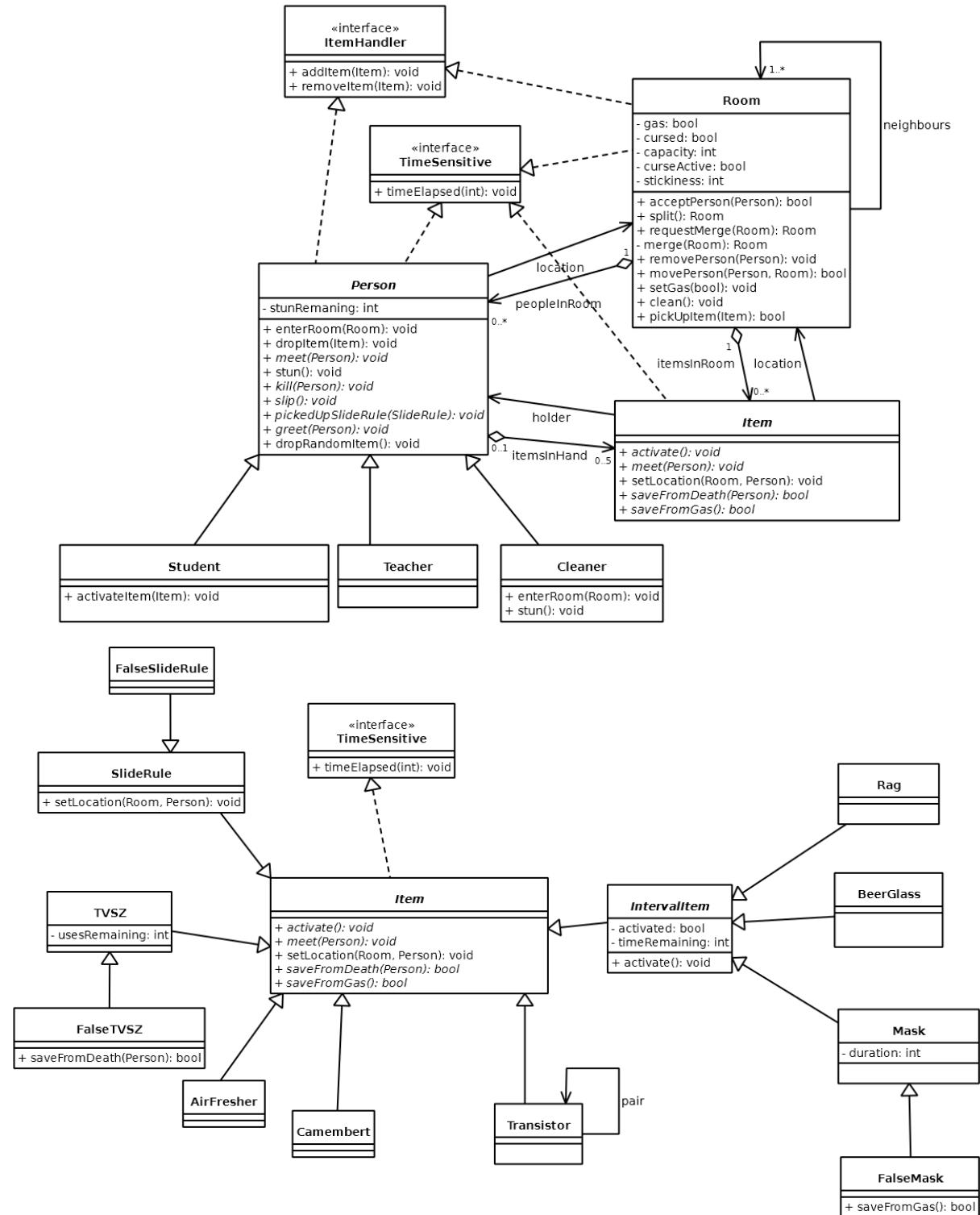
5.4 Napló

Kezdet	Időtartam	Résznevők	Leírás
2024. 03. 18. 19:00	2 óra	Görömbey	Logger elkészítése.
2024. 03. 19. 14:00	1.5 óra	Tömöri	Kezelői felület elkészítése.
2024. 03. 19. 16:00	3 óra	Görömbey Tömöri	Az enterRoom teszteléshez szükséges metódusok programozása
2024. 03. 20. 10:00	1 óra	Görömbey Héjja Sági Tömöri Vizhányó	Konzultáció
2024. 03. 20. 11:00	4 óra	Tömöri	Tesztosztályok elkészítése, tesztek inicializálása
2024. 03. 20. 11:00	3.5 óra	Görömbey	Logger továbbfejlesztése, Room, Person, Student metódusainak implementálása
2024. 03. 20. 11:00	3 óra	Vizhányó	Tárgyak metódusainak implementálása
2024. 03. 20. 12:00	1.5 óra	Sági	Teacher és Tranzisztor implementálása
2024. 03. 21. 12:00	0.5 óra	Vizhányó	Lefutási szekvenciák frissítése
2024. 03. 21. 22:00	3 óra	Héjja	Inicializációs diagramok frissítése és ellenőrzése
2024. 03. 22. 12:00	2 óra	Vizhányó	Forduláshoz és futtatáshoz szükséges parancsok megírása, kipróbálása a VM-ben, útmutatók leírása
2024. 03. 22. 14:00	0.5 óra	Tömöri	Dokumentumformázás
2024. 03. 23. 15:00	3 óra	Sági	Javadoc írása, a modell osztályainak dokumentálása
2024. 03. 23. 20:30	1.5 óra	Sági	Javadoc írása, a tesztelés osztályainak dokumentálása
2024. 03. 24. 9:00	4 óra	Sági	Tesztelés, hibák keresése a teszesetekben
2024. 03. 24. 18:00	2 óra	Héjja	Hibák javítása az inicializáló szekvenciákon
2024. 03. 24. 18:00	2 óra	Görömbey	Hibák javítása a programkódban
2024. 03. 24. 18:00	3 óra	Tömöri	Hibák javítása a lefutási szekvenciákban
2024. 03. 24. 19:00	2 óra	Vizhányó	Szobába lépési szekvencia korrekció
2024. 03. 24. 21:00	0.5 óra	Görömbey	Virtuális környezet végső tesztje

6. Prototípus koncepciója

6.0 Változás hatása a modellre

6.0.1 Módosult osztálydiagram



6.0.2 Új vagy megváltozó metódusok

AirFresher osztály:

- **Felelősség:** Ez az egyszerhasználatos tárgy aktiváláskor megszünteti a szoba gázosságát.
- **Ősosztály:** *Item*
- **Interfész:** TimeSensitive

Metódusok:

- **void activate():** A légrissítő aktiválása, a location-jére meghívja a setGas(false) metódust, ami beállítja a gas attribútumot a kívánt értékre. Ezután eltávolítja magát a holder-jétől removeItem(this)-szel
- **void meet(Person p):** Nem csinál semmit, mert ha földön van nincs funkciója.
- **bool saveFromDeath(Person p):** A tárgy nem nyújt védelmet a kibukás ellen, visszatérési értéke hamis.
- **bool saveFromGas():** A tárgy nem nyújt védelmet a gáz ellen, csak aktiválással, visszatérési értéke tehát hamis.
- **void timeElapsed(int time):** Mivel egyszerhasználatos tárgy, így nem történik vele semmi az idő műlásával.

BeerGlass módosítások:

- **bool saveFromDeath(Person p):** Amennyiben aktiválva van a söröspohár meghívja a holder-je dropRandomItem() függvényét, majd logikai igazzal tér vissza, megvédve a birtokosát. Egyéb esetben logikai hamissal tér vissza.

Camembert módosítások:

- **void activate():** A camembert aktiválása, igényli birtokosánál az önmegsemmisítést removeItem(this)-szel, majd a tartózkodási szobáján meghívja a setGas(true) metódust így elgázosítva azt.

Cleaner osztály:

- **Felelősség:** Ez a személy felel a takarításért minden új szobába lépéskor. Ekkor amennyiben lehetősége van, kitessékel mindenkit az aktuális szobából. A takarítással megszünteti a szoba gázos hatását és kinullázza a ragacsosságot.
- **Ősosztály:** *Person*
- **Interfész:** ItemHandler, TimeSensitive

Metódusok:

- **void enterRoom(Room r):** A takarító mozgási metódusa ugyanazzal kezdődik, mint az eredeti, ősből definiált (super.enterRoom(r) metódushívás). Majd amennyiben a location-je módosult, meghívja a location-ön a setGas(false) és a clean() metódusait, és frissíti a tárgyainak tartózkodási helyét.
- **void greet(Person p):** Nem csinál semmit azokkal akikkel találkozik.
- **void kill(Person p):** Nem történik semmi, ugyanis takarító nem halhat meg.
- **void meet(Person p):** Nem történik semmi, üres metódus.
- **void pickedUpSlideRule(SlideRule sr):** A takarító kidobja a Logarlécet.
- **void slip():** A takarítót nem kábítja el a rongy, nem történik semmi.
- **void stun():** A takarító nem ájul el gáz hatására, nem történik semmi.

FalseMask osztály:

- **Felelősség:** A tárgy látszatával ellentétben haszontalan, nem véd meg a mérgező gáztól.
- **Ősosztályok:** *Item → IntervalItem → Mask*
- **Interfész:** TimeSensitive

Metódusok:

- **bool saveFromGas():** Nem történik semmi, hamissal tér vissza.

FalseSlideRule osztály:

- **Felelősség:** A tárgy látszatával ellentében haszontalan, nem jelent győzelmet.
- **Ősosztályok:** *Item → SlideRule*
- **Interfész:** TimeSensitive

FalseTVSZ osztály:

- **Felelősség:** A tárgy látszatával ellentében haszontalan, nem véd meg a kibukástól.
- **Ősosztályok:** *Item → TVSZ*
- **Interfész:** TimeSensitive

Metódusok:

- **bool saveFromDeath(Person p):** Nem történik semmi, hamissal tér vissza.

Person módosítások:

- **void addItem(Item it):** A paraméterben kapott it tárgy felvétele a személyhez, ha a személy nincs elkábulva. Csak akkor veheti fel, ha még elfér az itemsInHand-ben. Ekkor a location-jének a pickUpItem(Item) metódusát hívja meg, kezdeményezve a tárgy felvételét. Ha ez igaz értékkel tér vissza, felveszi, beállítja a tárgy birtokosát. Ha hamissal tér vissza nem történik semmi.
- **void dropRandomItem():** Az ember kezéből egy véletlenszerűen választott tárgyat eldob a szobába. A véletlenszerű kiválasztás pszeudo-random módon történik, a seed-je állítható (teszteléshez).

Room módosítások:

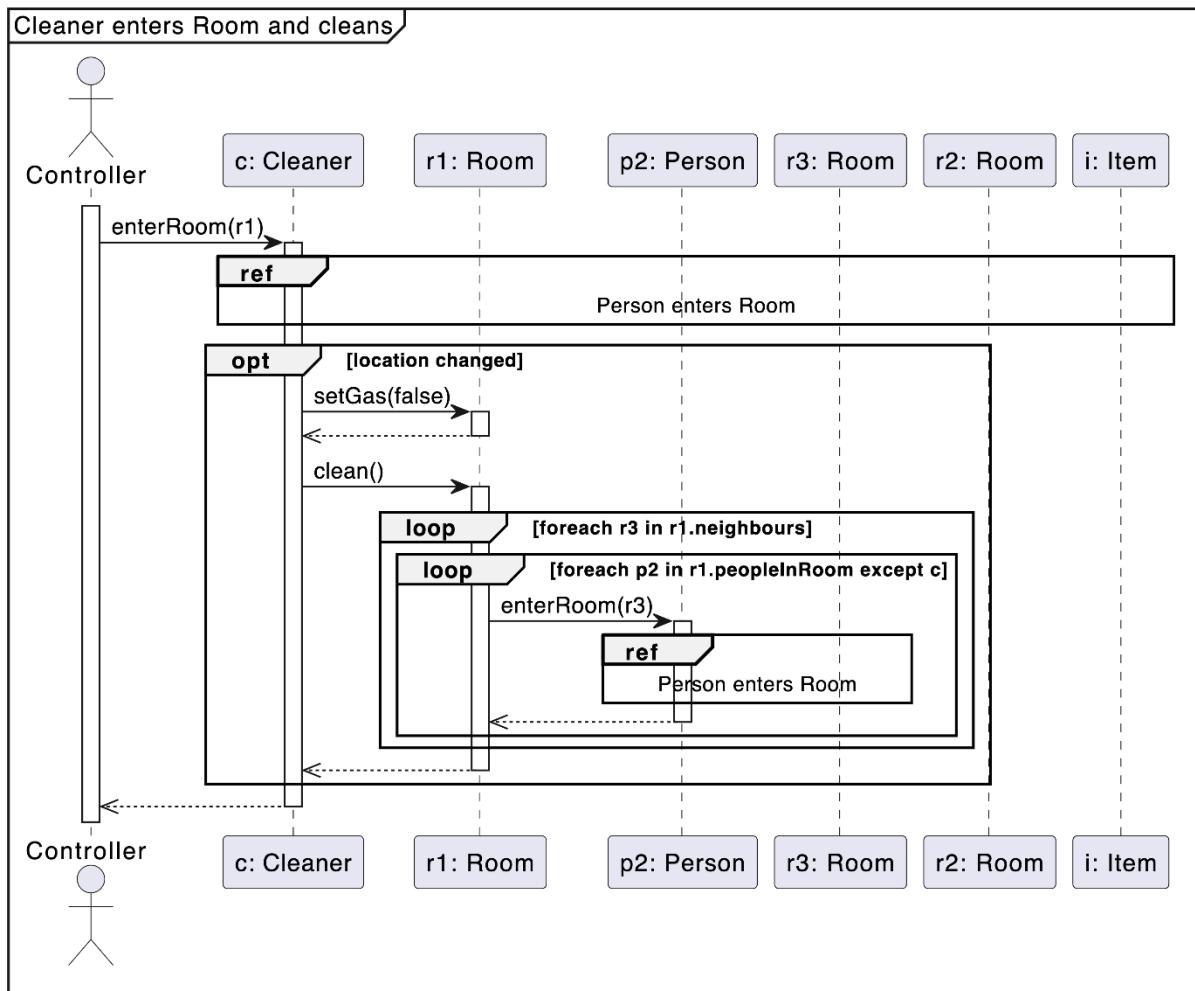
- **int stickiness:** attribútum, mely számon tartja, hogy az utolsó takarítás óta hányan léptek be a szobába – a ragacsosság reprezentálásaképp.
- **bool acceptPerson(Person p):** A paraméterként kapott személyt engedi be a szobába. Amennyiben a szoba kapacitása kimerült, nem engedi be a személyt. A visszatérési értéke a beengedés sikeressége. Ha beengedi a személyt, ***felel a stickiness érték növeléséért***, az új személy és a szobában tartózkodó személyek kölcsönös találkozásáért, illetve az új személy és szobában levő tárgyak találkozásáért. Ha a szoba mérgezett, felel a belépő játékos elkábításáért.
- **void clean():** A szoba takarítása, stickiness nullázása. A legutóbb érkezett ember (ez a takarító, aki a szobába jött) kivételével összes szobában tartózkodó embert átteszi egy másik szobába, amennyiben teheti. A szomszékok listájában előlről indul, és ameddig tudja tenni az embereket, addig oda teszi (meghívja az adott emberre az enterRoom(r3) metódust az adott r3 szomszédot átadva), ha pedig nem tudja, akkor a következő szomszéddal próbálkozik. Ha az összes szomszédon végig ment és még mindig maradt ember a szobában, akkor ők ott maradhatnak.
- **void createGas():** Mérgező gázzal tölti fel a szobát. ~~A gas értéke igaz lesz és minden szobában levő embert megmérgez.~~ (törölve, helyette setGas(bool))
- **bool pickUpItem(Item):** a removeItem()-hez hasonlóan eltávolítja a tárgyat a szobából, amennyiben a stickiness még nem érte el a küszöböt, majd igazzal visszatér. Ha viszont már elérte, akkor nem történik semmi csak false-sal visszatér.
- **void setGas(bool b):** A szoba gázosítása vagy annak megszüntetése. Amennyiben a b igaz, mérgező gázzal tölti fel a szobát, beállítja a gas értékét logikai igazra, majd felel a szobában tartózkodó személyek elkábításáért stun() hívásával. Ha a b értéke hamis, akkor a gas értékét is hamisra állítja.

6.0.3 Szekvencia-diagramok

6.0.3.1 Új szekvencia-diagramok

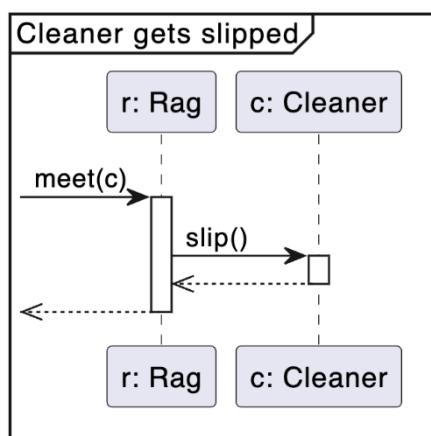
6.0.3.1.1 Cleaner enters room and cleans

Célja: Takarító enterRoom() és szoba clean() metódusának bemutatása



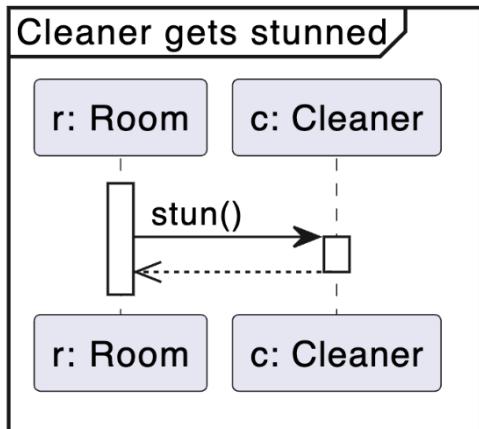
6.0.3.1.2 Cleaner gets slipped

Célja: Takarító slip() metódusának bemutatása

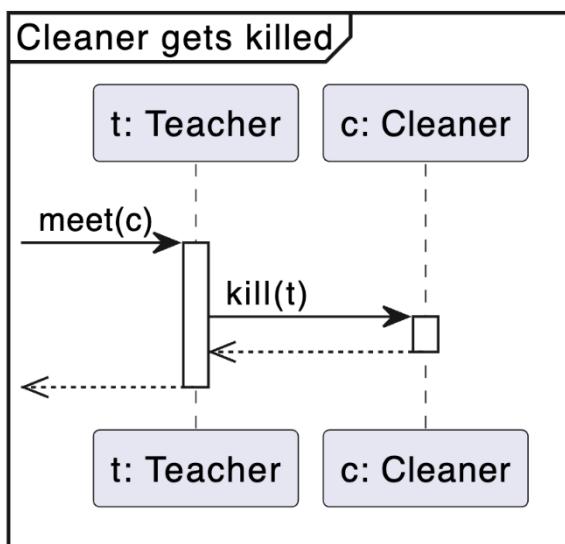


6.0.3.1.3 Cleaner gets stunned

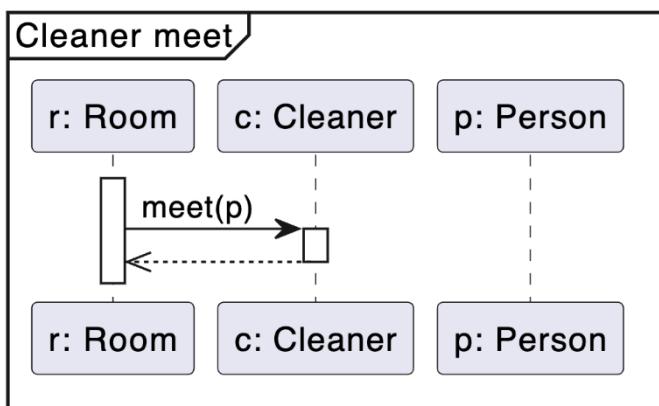
Célja: Takarító stun() metódusának bemutatása

**6.0.3.1.4 Cleaner gets killed**

Célja: Takarító kill(t) metódusának bemutatása

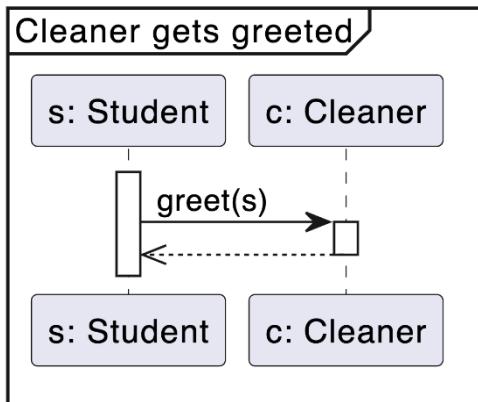
**6.0.3.1.5 Cleaner meet**

Célja: Takarító meet(p) metódusának bemutatása



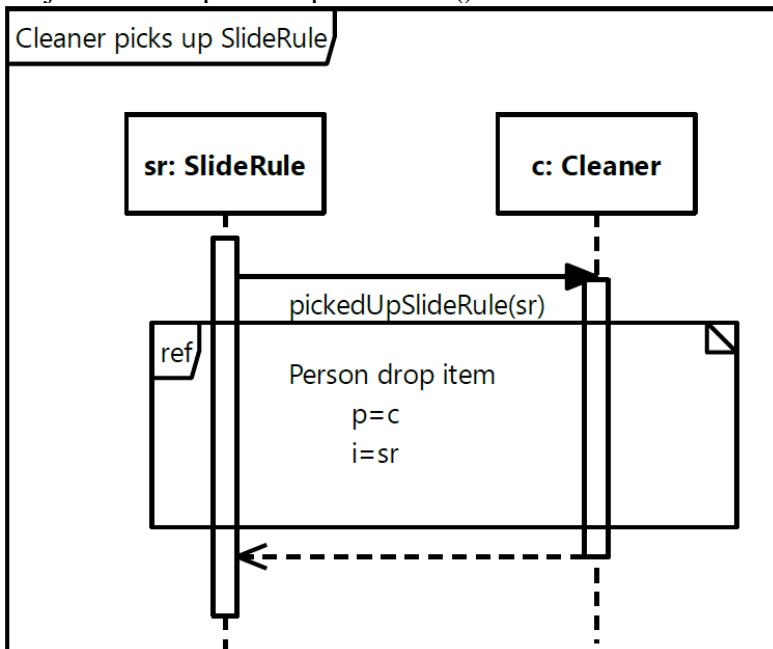
6.0.3.1.6 Cleaner is greeted

Célja: Takarító greet(p) metódusának bemutatása



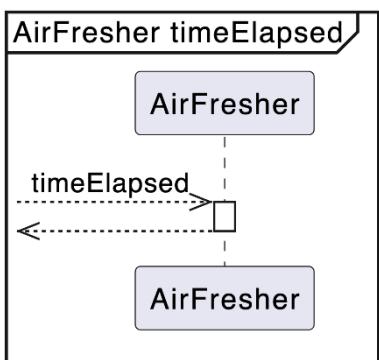
6.0.3.1.7 Cleaner picks up SlideRule

Célja: Takarító pickedUpSlideRule() metódusának bemutatása



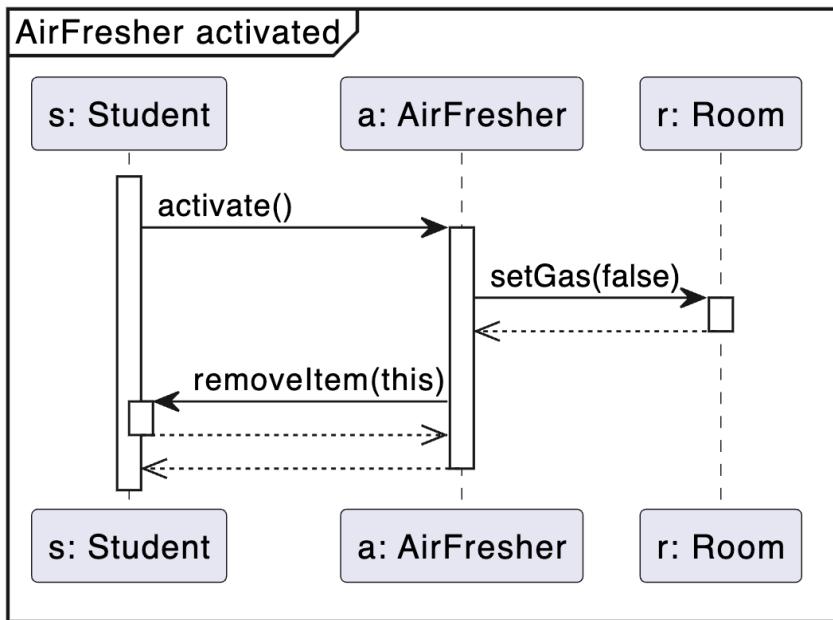
6.0.3.1.8 Time elapsed on AirFresher

Célja: Léggördítő timeElapsed() metódusának bemutatása



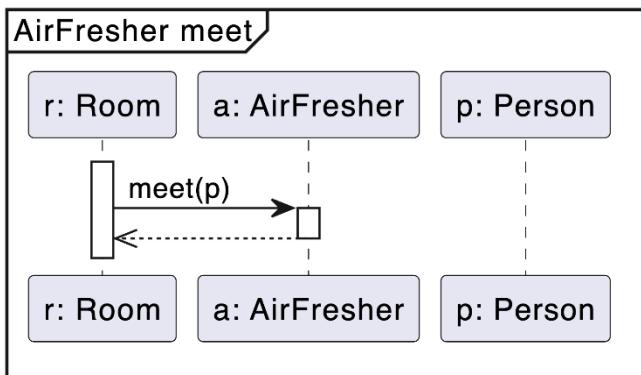
6.0.3.1.9 AirFresher is activated

Célja: Légfrissítő activate() metódusának bemutatása



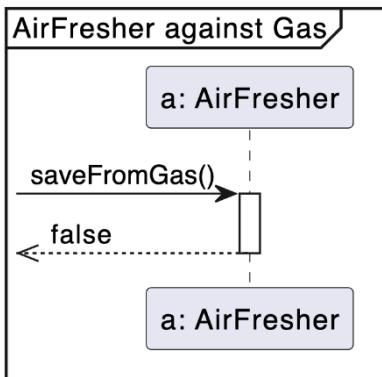
6.0.3.1.10 AirFresher meet

Célja: Légfrissítő meet(p) metódusának bemutatása



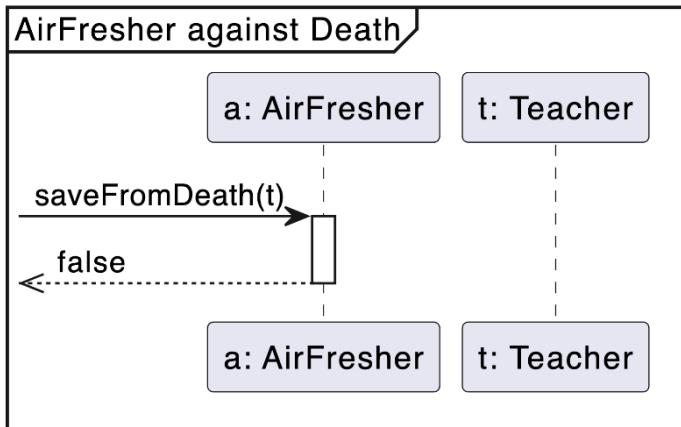
6.0.3.1.11 AirFresher against Gas

Célja: Légfrissítő saveFromGas() metódusának bemutatása



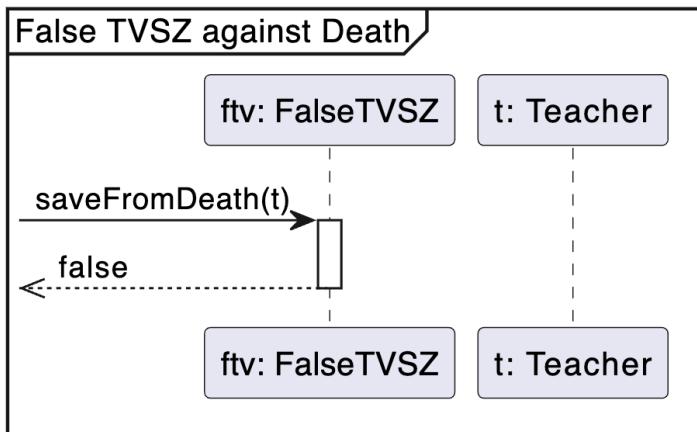
6.0.3.1.12 AirFresher against Death

Célja: Légfrissítő saveFromDeath(t) metódusának bemutatása



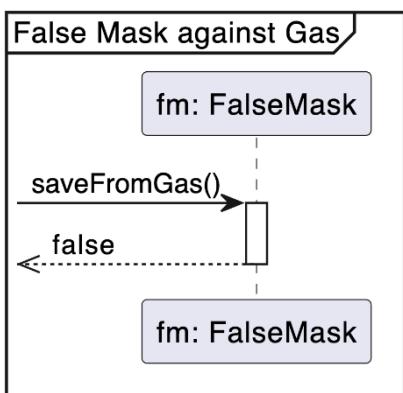
6.0.3.1.13 FalseTVSZ against Death

Célja: FalseTVSZnek az ōhéhez képesti változása.



6.0.3.1.14 FalseMask against Gas

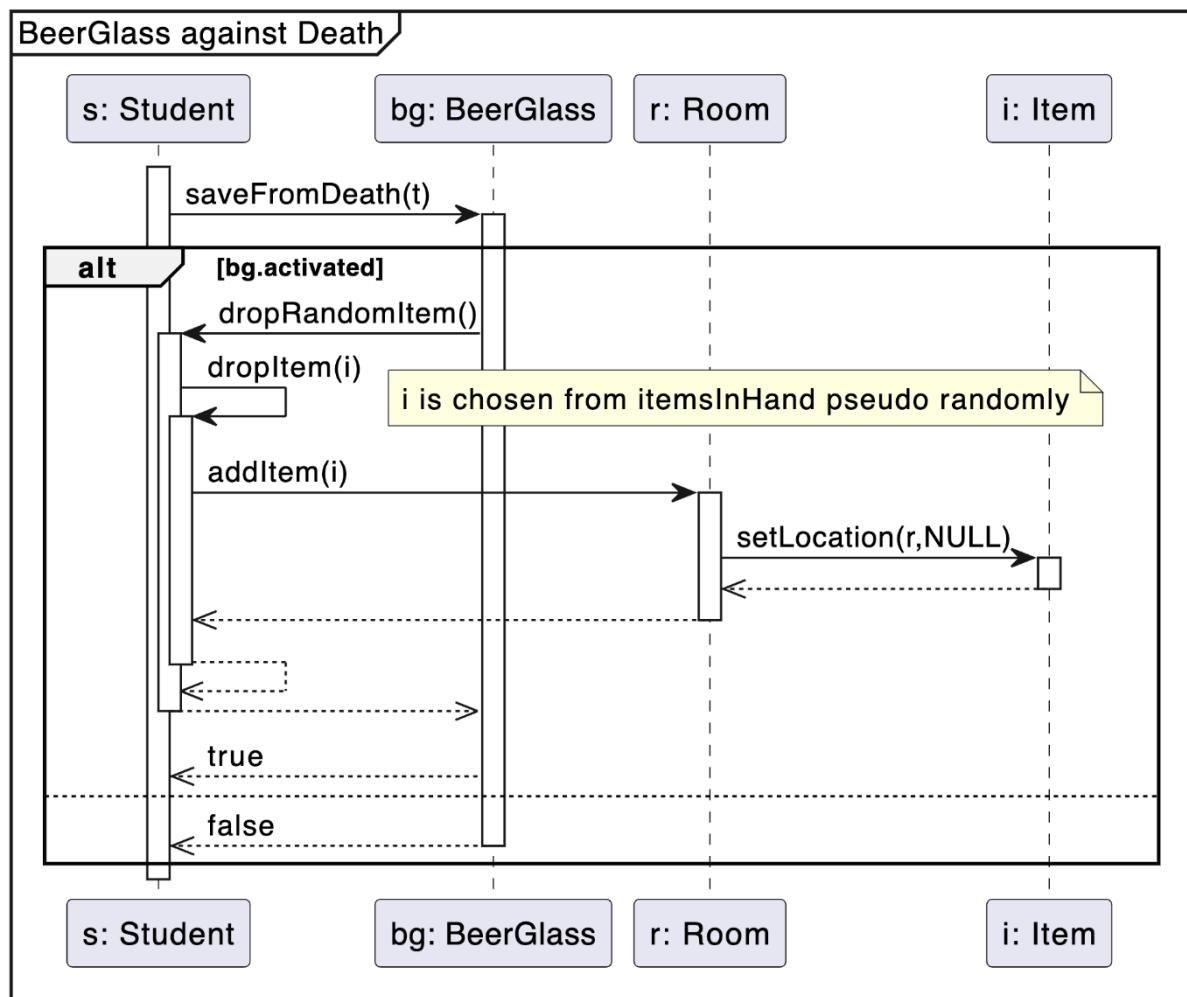
Célja: FalseMasknek az ōhéhez képesti változása.



6.0.3.2 Újítások miatt módosított szekvencia-diagramok

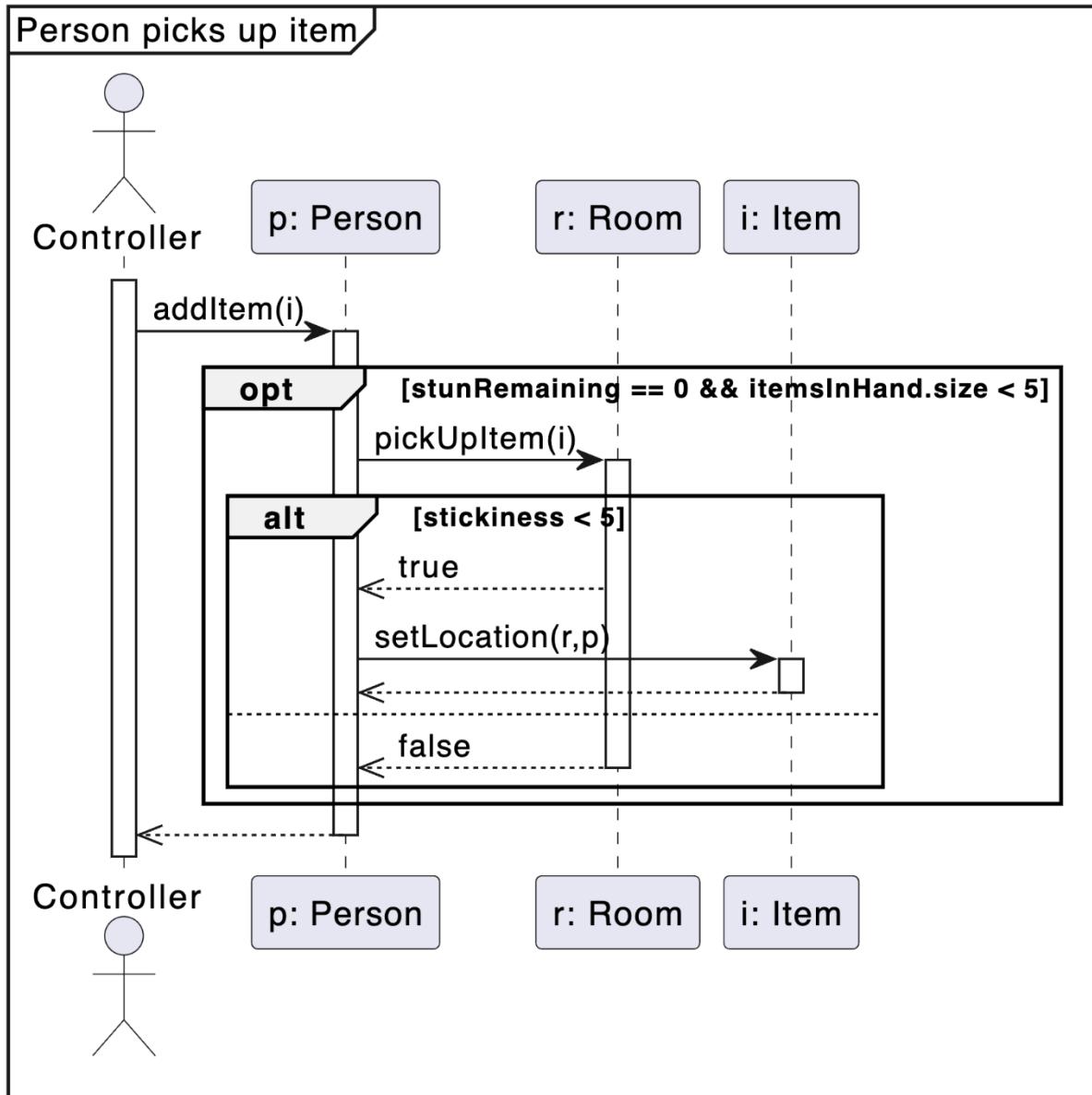
6.0.3.2.1 BeerGlass against Death

Módosítás: dropRandomItem() meghívása, bemutatása.



6.0.3.2.2 Person picks up item

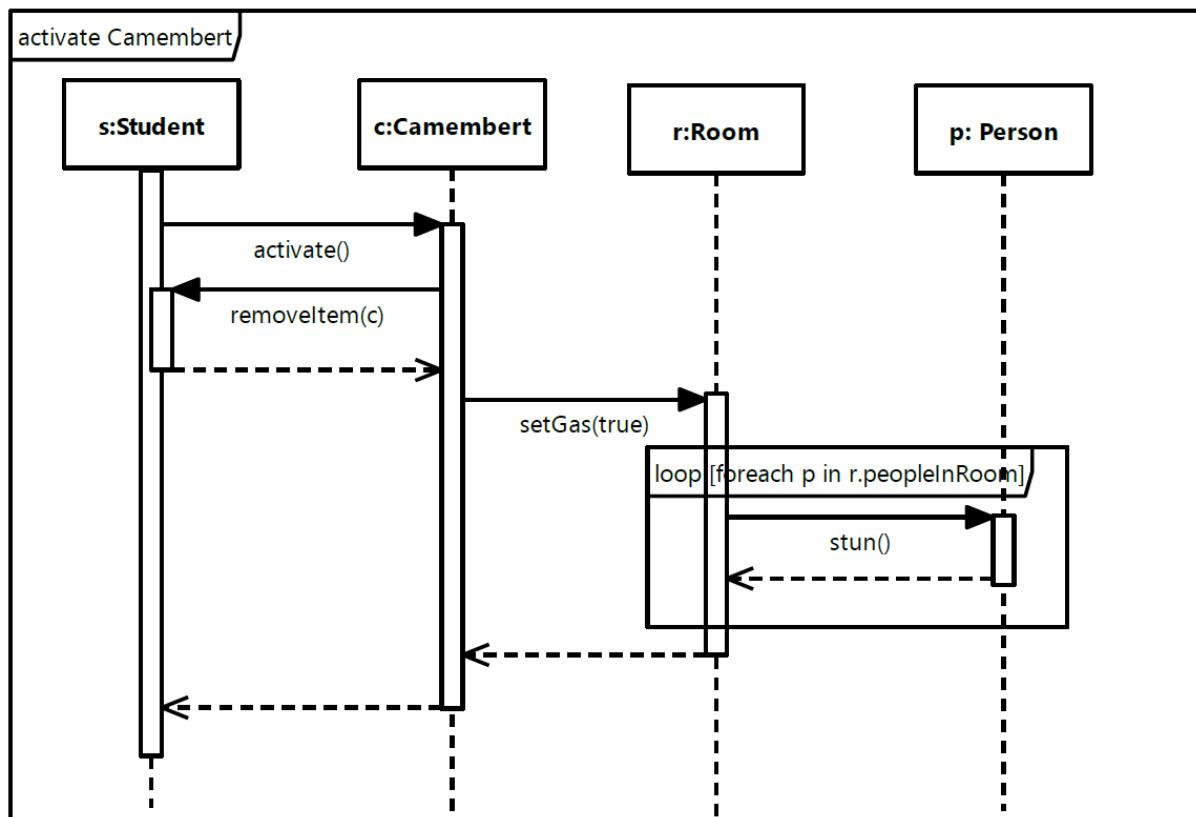
Módosítás: removeItem helyett pickUpItem, ragacsosság vizsgálata.



6.0.3.2.3 activate Camembert

Módosítás: createGas() helyett setGas(true)

Javítás: előbb hívódik meg a removeItem, mint a setGas (ahogy már a szkeleton tervben is)

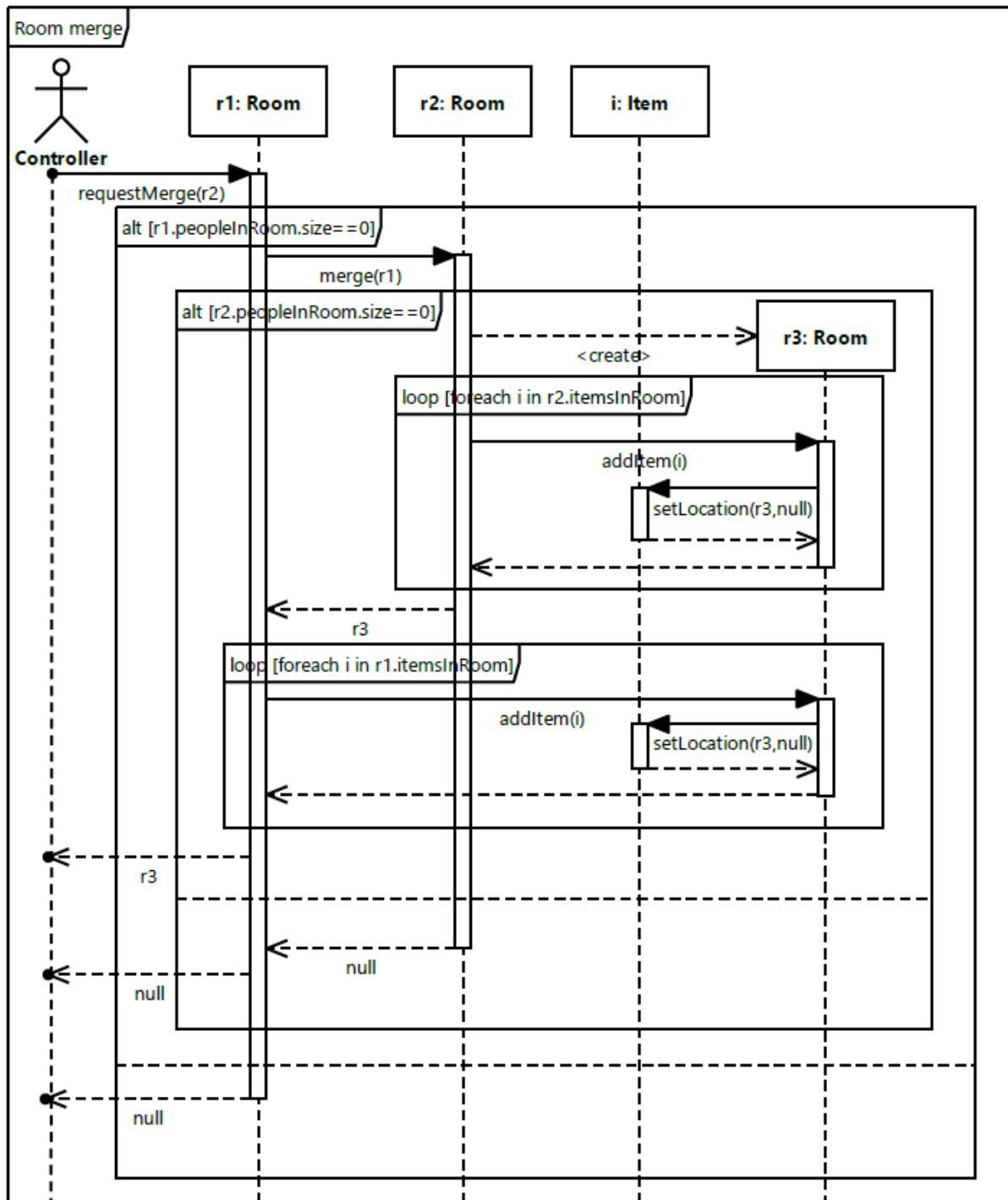


6.0.3.3 Refaktorálás miatt módosított szekvencia-diagramok

Szekvencia-diagramok, melyek az analízis modellben minimális hibát tartalmaztak. Ezek a szkeleton terveiben már helyesen voltak implementálva, de most, hogy az analízis modell változott, kijavítottuk ott is.

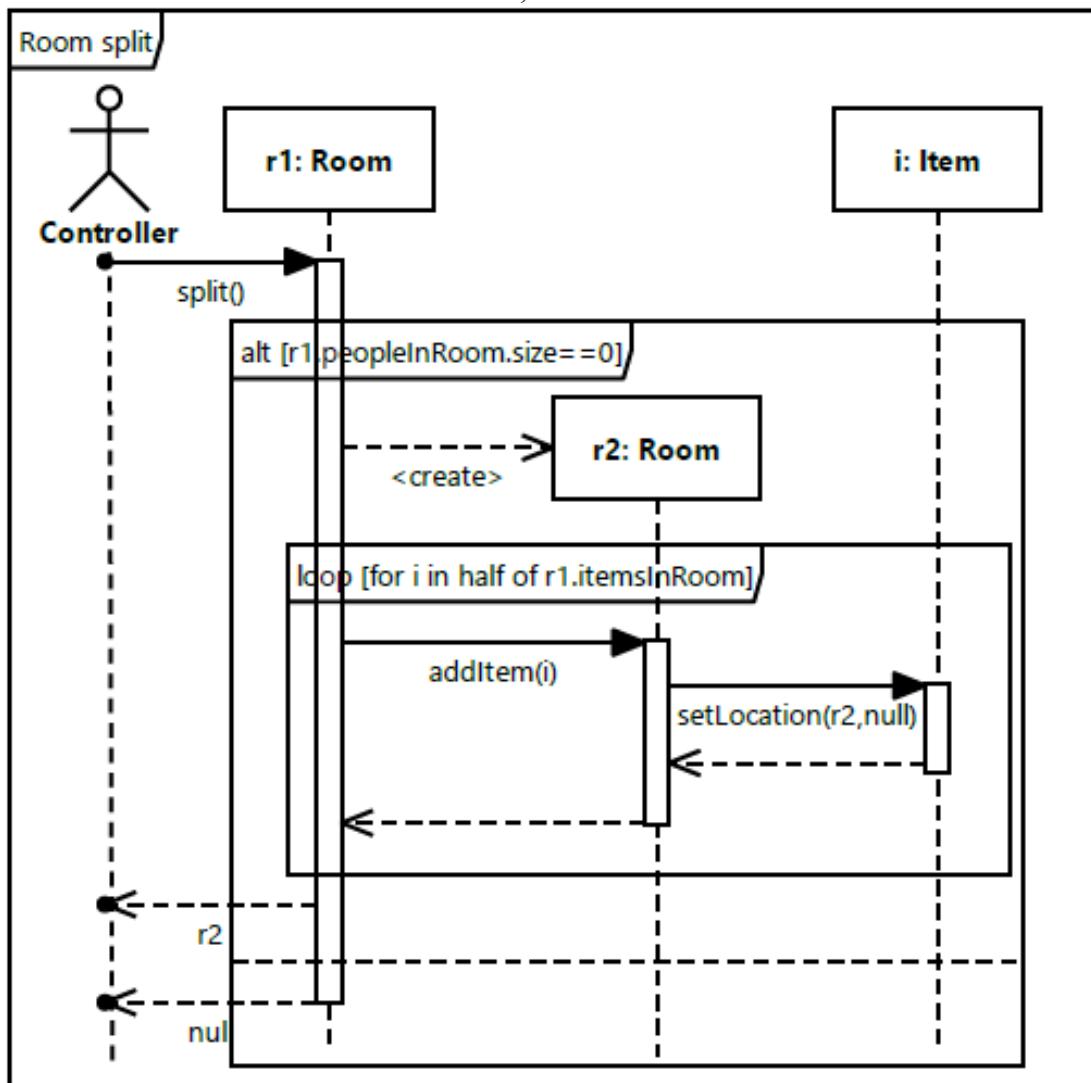
6.0.3.3.1 Room merge

Javítás: setLocation nem az addItem után, hanem abból hívódik.



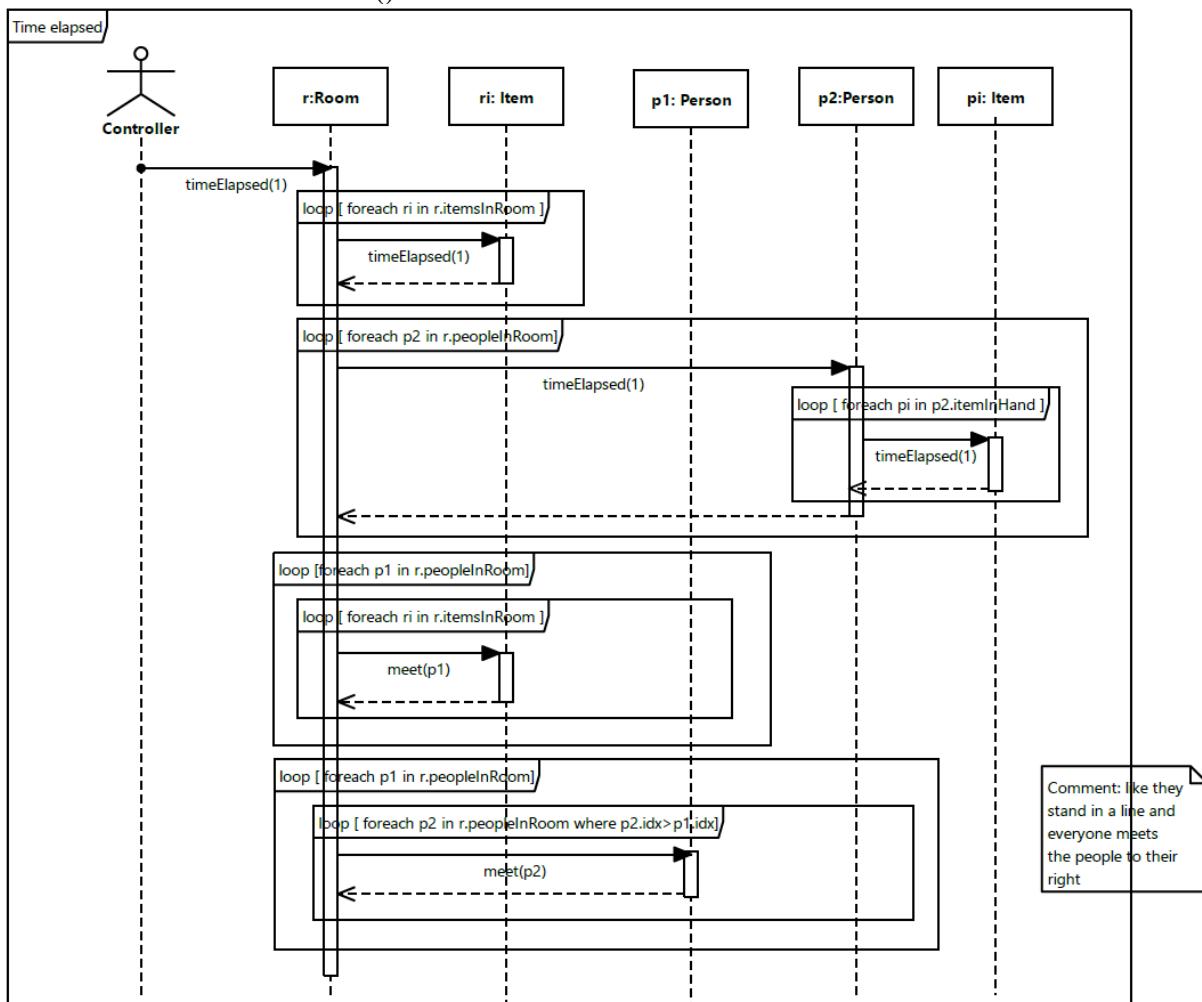
6.0.3.3.2 Room split

Javítás: setLocation nem az addItem után, hanem abból hívódik.



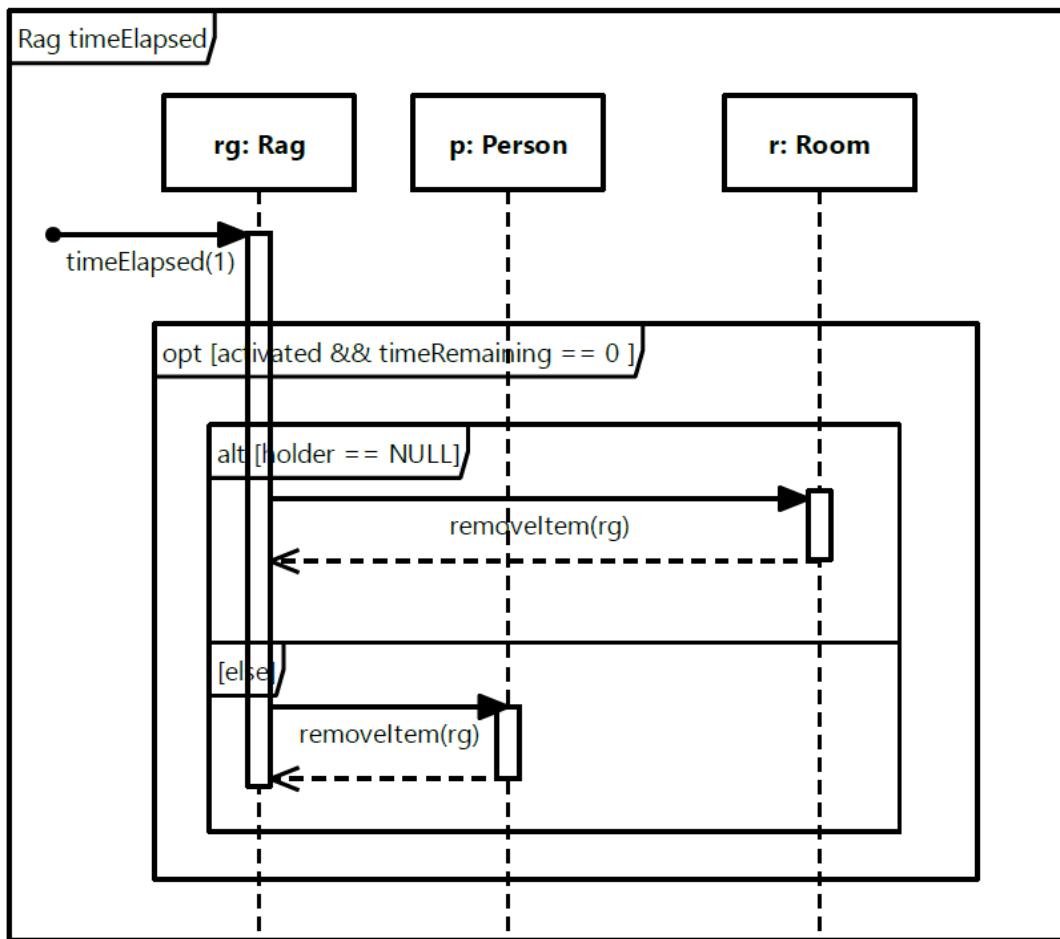
6.0.3.3.3 Time elapsed

Javítás: a szoba nem hív stun()-t időteléskor



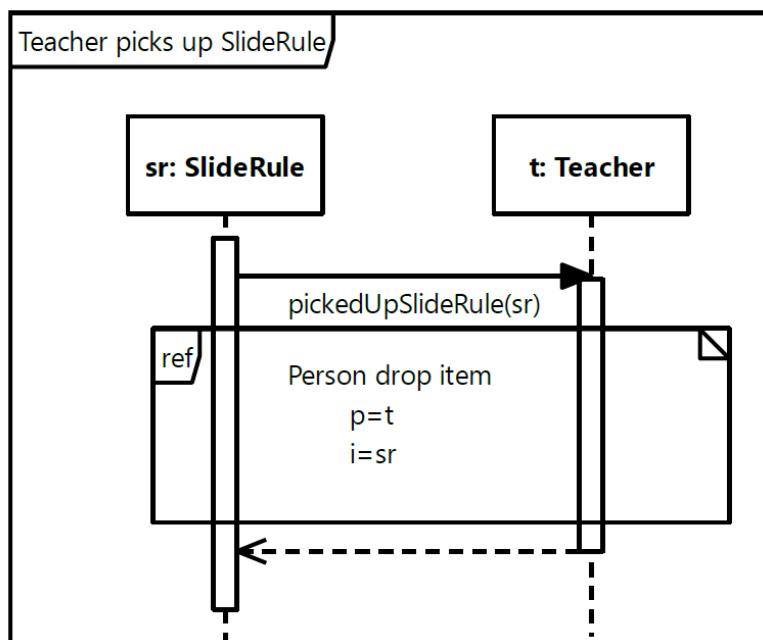
6.0.3.3.4 Rag timeElapsed

Javítás: a rongy nem slip()-el közvetlenül az időtelés miatt, ez a meet() miatt történik.



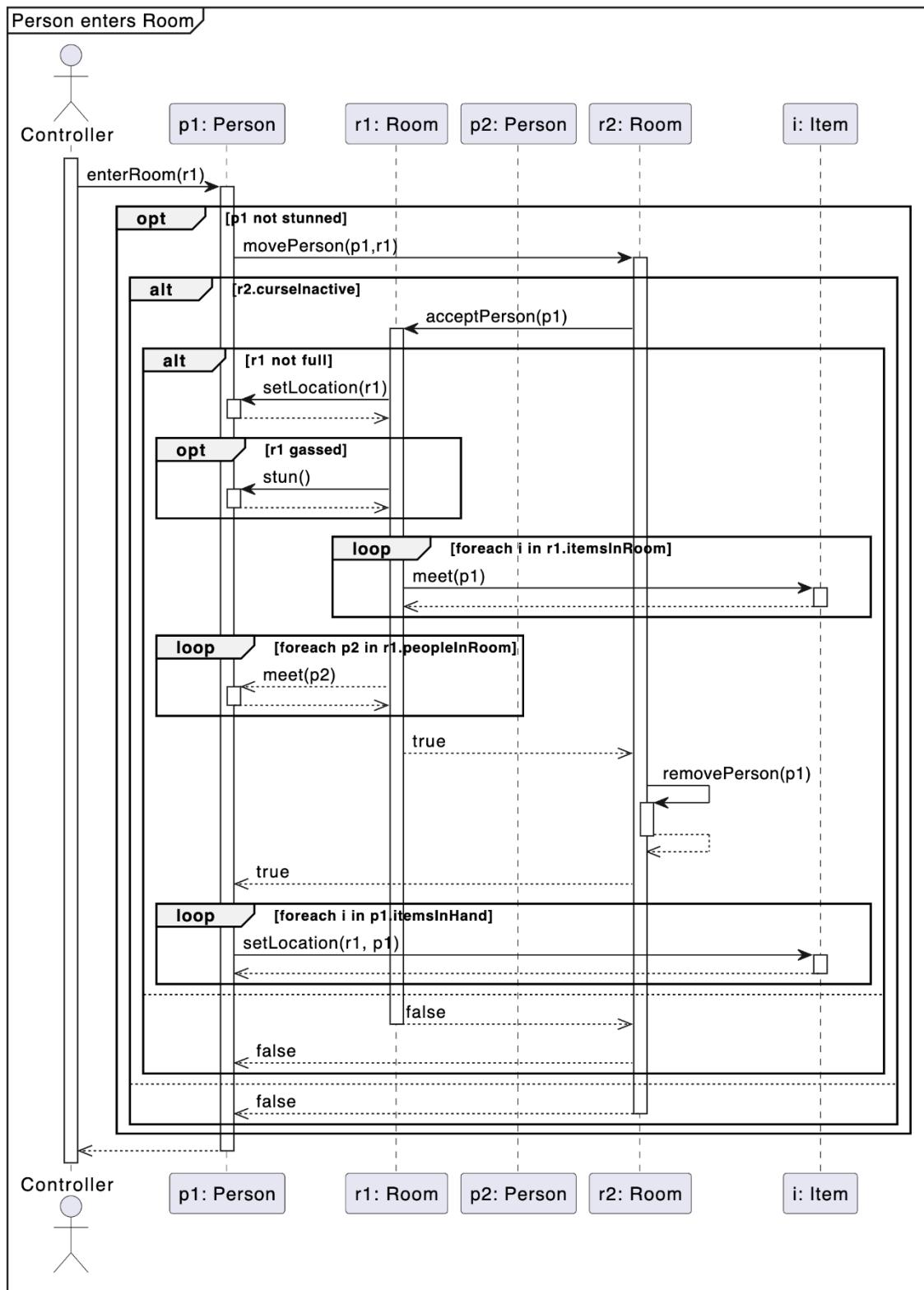
6.0.3.3.5 Teacher picks up SlideRule

Javítás: a dropItem-re referenciát hívunk hogy ne ismételjük a szekvenciákat



6.0.3.3.6 Person enters room

Javítás: Student enters room-ban lévő apró eltérések módosítása, ami a szkeleton tervnél már helyesen szerepelt. További általánosítás történt, a konkrét metódusok (stun, meet) az objektumok típusától függően eltérően viselkednek, de ezek más szekvenciadiagramokon külön szerepelnek.



6.1 Prototípus interface-definíciója

6.1.1 Az interfész általános leírása

A prototípus program interfésze a szabványos be- és kimenetet használja. A felhasználó a bemeneten megadhat parancsokat, azok működését kapcsolókkal és paraméterekkel pontosíthatja. A kapcsolókat és paramétereket a parancs után kell felsorolni egy-egy szóközzel elválasztva. Program az egyes parancsokra visszajelzést adhat a felhasználó számára a szabványos kimeneten. A programot lehet egyszerű konzolos felületen irányítani az egyes utasítások begépelésével, illetve a parancsokat képes egy input fájlból is venni. A parancsok kimenete továbbá átirányítható egy output fájlba is

6.1.2 Bemeneti nyelv

6.1.2.1 Inicializáló parancsok

create <típus> <név>

Leírás: A create parancs segítségével objektumokat lehet létrehozni. A parancs után szóközzel elválasztva meg kell adni az objektum típusát, utána pedig szintén szóközzel elválasztva az objektum nevét, amivel hivatkozni lehet rá.

Név opciók:

Az a hivatkozási név, amivel a felhasználó el szeretné érni a létrehozandó objektumot. Nem tartalmazhat szóközt.

Típus opciók:

- **room** - egy Room objektumot hoz létre úgy, hogy a listái üresek, a bool tagváltozói false értékűek és int tagváltozói 3 értékűek.
- **student** – egy Student objektumot hoz létre úgy, hogy az itemsInHand listája üres, a stunRemaining tagváltozója 0 értékű, illetve a location tagváltozója NULL értékű
- **teacher** - egy Teacher objektumot hoz létre úgy, hogy az itemsInHand listája üres, a stunRemaining tagváltozója 0 értékű, illetve a location tagváltozója NULL értékű
- **cleaner** – egy Cleaner objektumot hoz létre úgy, hogy az itemsInHand listája üres, a stunRemaining tagváltozója 0 értékű, illetve a location tagváltozója NULL értékű
- **rag** – egy Rag objektumot hoz létre úgy, hogy az activated tagváltozója false értékű és a timeRemaining tagváltozója 5 értékű, emellett a holder és location tagváltoozói egyaránt NULL értékűek
- **beerglass** – egy BeerGlass objektumot hoz létre úgy, hogy az activated tagváltozója false értékű és a timeRemaining tagváltozója 5 értékű, emellett a holder és location tagváltoozói egyaránt NULL értékűek
- **mask** – egy Mask objektumot hoz létre úgy, hogy az activated tagváltozója false értékű, a timeRemaining tagváltozó értéke 6, a duration tagváltozó értéke 4, emellett a holder és location tagváltozói egyaránt NULL értékűek
- **falsemask** – egy FalseMask objektumot hoz létre úgy, hogy az activated tagváltozója false értékű, a duration tagváltozó értéke 4, emellett a holder és location tagváltozói egyaránt NULL értékűek
- **sliderule** – egy SlideRule objektumot hoz létre úgy, hogy a holder és location tagváltozói egyaránt NULL értékűek

- **falsesliderule** – egy FalseSlideRule objektumot hoz létre úgy, hogy a holder és location tagváltozói egyaránt NULL értékűek
- **tvsz** – egy TVSZ objektumot hoz létre úgy, hogy a usesRemaining tagváltozója 3 értékű, emellett a holder és location tagváltoozói egyaránt NULL értékűek
- **falsetvsz** - egy FalseTVSZ objektumot hoz létre úgy, hogy a usesRemaining tagváltozója 3 értékű, emellett a holder és location tagváltoozói egyaránt NULL értékűek
- **airfresher** – egy AirFresher objektumot hoz létre úgy, hogy a holder és location tagváltozói egyaránt NULL értékűek
- **transistor** – egy Transistor objektumot hoz létre úgy, hogy a pair tagváltozója NULL értékű, emellett a holder és location tagváltozói egyaránt NULL értékűek
- **camembert** – egy Camembert objektumot hoz létre úgy, hogy a holder és location tagváltozói egyaránt NULL értékűek

addperson <személynév> <szobanév>

Leírás: Az első paraméterben megadott Person-leszármazott objektumot elhelyezi a második paraméterben kapott Room objektum peopleInRoom listájában. A Person objektum location tagváltozóját pedig a Room objektumra állítja.

Opciók:

Személynév: egy olyan objektum hivatkozási neve, ami a Person alaposztályból származik.

Szobanév: egy olyan objektum hivatkozási neve, ami a Room típusú

add <tárgynév> <személy/szoba név>

Leírás: Az add parancs az első paraméterként megadott Item-leszármazott objektumot hozzáadja a második paraméterként kapott Room vagy Person-leszármazott objektum tárhelyéhez.

Ha Room objektum tárhelyéhez adja hozzá, akkor az Item-leszármazott objektum location tagváltozóját beállítja a Room objektumra, míg holder tagváltozóját NULL-ra állítja.

Ha Person-leszármazott objektum tárhelyéhez adja hozzá, akkor az Item-leszármazott objektum location tagváltozóját beállítja a Room objektumra, amiben Person-leszármazott objektum van, míg holder tagváltozóját a Person-leszármazott objektumra állítja.

A paramétereket szóközökkel elválasztva kell megadni a parancs után.

Opciók:

Tárgynév: egy olyan objektum hivatkozási neve, ami az Item alaposztályból származik.

Személy/szoba név: egy olyan objektum hivatkozási neve, ami vagy a Person alaposztályból származik, vagy Room típusú

remove <tárgynév> <személy/szoba név>

Leírás: A remove parancs az első paraméterként megadott Item-leszármazott objektumot törli a második paraméterként kapott Room vagy Person-leszármazott objektum tárhelyéből. A paramétereket szóközökkel elválasztva kell megadni a parancs után.

Opciók:

Tárgynév: egy olyan objektum hivatkozási neve, ami az Item alaposztályból származik.

Személy/szoba név: egy olyan objektum hivatkozási neve, ami vagy a Person alaposztályból származik, vagy Room típusú

setroom <szobanév> <állapot> <érték>

Leírás: Az első paraméterként kapott Room objektum egyik tagváltozóját változtatja meg. A második paraméter adja meg a változtatni kívánt tagváltozót, a harmadik paraméter pedig az értéket, amire változnia kell. A paramétereket szóközökkel elválasztva kell megadni a parancs után.

Szobanév opciók: egy olyan objektum hivatkozási neve, ami Room típusú

Állapot opciók:

- **gas** – a Room objektum gas tagváltozóját változtatja meg
- **cursed** – a Room objektum cursed tagváltozóját változtatja meg
- **curseactive** - a Room objektum curseActive tagváltozóját változtatja meg
- **capacity** - a Room objektum capacity tagváltozóját változtatja meg
- **stickiness** - a Room objektum stickiness tagváltozóját változtatja meg

Érték opciók:

capacity és **stickiness**: nemnegatív egész szám

gas, curse és **curseactive**: logikai érték: “true” - igaz, “false” – hamis

setstun <személynév> <érték>

Leírás: Az első paraméterként kapott Person-leszármazott objektum stunRemaining tagváltozóját állítja be a második paraméterként kapott értékre. A paramétereket szóközökkel elválasztva kell megadni a parancs után.

Opciók:

Személynév: egy olyan objektum hivatkozási neve, ami a Person alaposztályból származik.

Érték: nemnegatív egész szám

neighbour <szobanév1> <szobanév2>

Leírás: A neighbour parancs az első paraméterként kapott Room objektum neighbours listájához hozzáadja a második paraméterben kapott Room objektumot. A paramétereket szóközökkel elválasztva kell megadni a parancs után.

Szobanév opciók: egy olyan objektum hivatkozási neve, ami Room típusú

setpair <tranzisztornév1> <tranzisztornév2>

Leírás: Kezdeményezi a két, paraméterben kapott párosítatlan tranzisztor párosítását. A paramétereket szóközökkel elválasztva kell megadni a parancs után.

Opciók:

- Tranzisztornév1: egy olyan objektum hivatkozási neve, ami Transistor típusú.
- Tranzisztornév2: egy olyan objektum hivatkozási neve, ami Transistor típusú.

setremaining <Mask/FalseMask/TVSZ/FalseTVSZ név> <érték>

Leírás: Az első paraméterben megadott Mask, FalseMask, TVSZ vagy FalseTVSZ objektum megfelelő tagváltozójának értékét állítja át a második paraméterben megadott értékre. Mask és FalseMask objektum esetén a duration tagváltozó értékét állítja, míg TVSZ és FalseTVSZ objektum esetén a usesRemaining tagváltozóét.

Opciók:

- Mask/FalseMask/TVSZ/FalseTVSZ név: egy olyan objektum hivatkozási neve, ami Mask, FalseMask, TVSZ vagy FalseTVSZ típusú

Érték: TVSZ és False TVSZ: pozitív egész szám, Mask és FalseMask: nemnegatív egész szám

setinterval <tulajdonság> <IntervalItem név> <érték>

Leírás: A második paraméterként megadott IntervalItem osztályból származó objektum egy tagváltozóját változtatja meg. Az első paraméter határozza meg, hogy melyik tagváltozót kell megvátoztatni, a harmadik paraméter pedig az értéket adja meg, amire meg kell változtatni.

Állapot opciók:

- activated – az IntervalItem objektum activated tagváltozóját változtatja meg
- timeremaining – az IntervalItem objektum timeRemaining tagváltozóját változtatja meg

IntervalItem név opciók: egy olyan objektum hivatkozási neve, ami az IntervalItem osztályból származik.

Érték opciók:

- activated: logikai érték: “true” - igaz, “false” - hamis
- timeremaining: pozitív egész szám

setseed <személy név> <érték>

Leírás: A paraméterként kapott Person-leszármazott véletlenszám generátotának seed-jének beállítása a kapott értékre.

load <fájlnév>

Leírás: A paraméterben megadott elérési útvonalon talalható érvényes parancsokat tartalmazó szöveges fájlból (.txt formátumú) tartalmát beolvassa és végrehajtja az utasításokat.

Fájlnév opciók: az érvényes parancsokat tartalmazó szöveges (.txt formátumú) fájl elérési útvonala

6.1.2.2 Játékvezérlő parancsok

drop <tárgynév> <személynév>

Leírás: Kezdeményezi a második paraméterben megadott Person-leszármazott objektumnál lévő, első paraméterben kapott Item-leszármazott objektum eldobását.

Opciók:

Tárgynév: egy olyan objektum hivatkozási neve, ami az Item alaposztályból származik.

Személynév: egy olyan objektum hivatkozási neve, ami a Person alaposztályból származik.

pickup <tárgynév> <személynév>

Leírás: Kezdeményezi a második paraméterként megadott Person-leszármazott objektumnál a vele egy szobában lévő, első paraméterként megadott Item-leszármazott objektum felvételét. A paramétereket szóközökkel elválasztva kell megadni a parancs után.

Opciók:

Tárgynév: egy olyan objektum hivatkozási neve, ami az Item alaposztályból származik.

Személynév: egy olyan objektum hivatkozási neve, ami a Person alaposztályból származik.

enter <szobanév> <személynév>

Leírás: Kezdeményezi a második paraméterként kapott Person-leszármazott objektum átléptetését az első paraméterként megadott szomszédos Room objektumba. A paramétereket szóközökkel elválasztva kell megadni a parancs után.

Opciók:

Szobanév: egy olyan objektum hivatkozási neve, ami Room típusú

Személynév: egy olyan objektum hivatkozási neve, ami a Person alaposztályból származik.

merge <szobanév1> <szobanév2> <új szobanév>

Leírás: Kezdeményezi az első és második paraméterekben megadott szomszédos Room objektumok összeolvadását egy újonnan létrehozott, a harmadik paraméterben megadott nevű Room objektumba. A paramétereket szóközökkel elválasztva kell megadni a parancs után.

Opciók:

Szobanév1: egy olyan objektum hivatkozási neve, ami Room típusú

Szobanév2: egy olyan objektum hivatkozási neve, ami Room típusú

Új szobanév: Nem tartalmazhat szóközöket.

split <szobanév> <új szobanév>

Leírás: Kezdeményezi a paraméterben megadott Room objektum osztódását, így létrehozva egy újonnan létrehozott, a második paraméterben megadott nevű Room objektumot. A paramétert szóközzel elválasztva kell megadni a parancs után.

Opciók:

Szobanév: egy olyan objektum hivatkozási neve, ami Room típusú
Új szobanév: Nem tartalmazhat szóközöket.

activate <tárgynév> <hallgatónév>

Leírás: Kezdeményezi a második paraméterben kapott Student objektum kezében lévő, első paraméterként kapott Item-leszármazott objetum aktiválását. A paramétereket szóközökkel elválasztva kell megadni a parancs után.

Opciók:

Tárgynév: egy olyan objektum hivatkozási neve, ami az Item alaposztályból származik.
Hallgatónév: egy olyan objektum hivatkozási neve, ami Student típusú

elapse time <érték>

Leírás: A paraméterben megadott értékkel telik az idő és ezt az objektumok felé továbbítja.

Érték opciók: pozitív egész szám, ami az eltelt időegységek számát jelenti

6.1.2.3 Játékállapot-ellenőrző parancsok

status <objektumnév>

Leírás: A szabványos kimenetre kiírja a paraméterben kapott objetum összes tagváltozójának értékét.

Objektumnév opciók: egy Item vagy Person alaposztályból származó, vagy Room típusú objektum hivatkozási neve

6.1.3 Kimeneti nyelv

Általánosságban:

Sikeress lefutás esetén a parancsok nem adnak kimenetet.

Ha a parancs nem helyes, tehát

- A parancsnév nem létezik
- Nem megfelelő paraméterszám
- Nem létező objektumra való hivatkozás paraméterben
- Nem megfelelő típusú vagy értékű paraméter

Esetén a kimenet: “Incorrect command”

A következőkben ezt egészítjük ki parancsspecifikusan.

create <típus> <név>

- Ha a kért név már foglalt: “*Error: Name <név> already exists*”

addperson <személynév> <szobanév>

- Ha a személy már van egy szobában: “*Error: Person is already in a Room*”

add <tárgynév> <személy/szoba név>

- Ha a tárgy már van egy szobában vagy valakinek a kezében van: “*Error: Item is already placed*”

remove <tárgynév> <szemény/szoba név>

- Ha a tárgy nem a személynél/szobánál van: “*Error: Item <tárgynév> is not held by <személy/szoba név>*.”

drop <tárgynév> <személynév>

- Ha a tárgy nem a személynél van: *Error: Item <tárgynév> is not held by <személy/szoba név>*.

pickup <tárgynév> <személynév>

- Ha a tárgy és a személy nem ugyanabban a szobában van: “*Error: Item <tárgynév> and Person <személynév> are not in the same room.*”

setroom <szobanév> <állapot> <érték>

setstun <személynév> <érték>

neighbour <szobanév1> <szobanév2>

- Ha a szobák már szomszédosak: “*Error: Rooms <szobanév> and <szobanév> are already neighbours.*”
- Ha a két szoba megegyezik: “*Error: Room objects must differ.*”

enter <szobanév> <személynév>

- Ha a paraméterben kapott szoba nem szomszédos a paraméterben kapott személy szobájával: “*Error: Rooms are not neighbours.*”

status <objektumnév>

Kimenet sikeres lefutás esetén:

Ha egy tagváltozó értéke NULL, akkor “NULL”-t ír ki a megfelelő értéknek.

Room objektum esetén:

“

<objektumnév>

neighbours: <szomszédos szoba 1>, <szomszédos szoba 2>, ...

peopleInRoom: <szobában lévő személy 1>, <szobában lévő személy 2>, ...

itemsInRoom: <szobában lévő tárgy 1>, <szobában lévő tárgy 2>, ...

gas: <gas tagváltozó értéke>

cursed: <cursed tagváltozó értéke>

capacity: <capacity tagváltozó értéke>

curseActive: <curseActive tagváltozó értéke>

stickiness: <stickiness tagváltozó értéke>

“

Student objektum esetén:

“

<objektumnév>

location: <location tagváltozó értéke>

itemsInHand: <kézben lévő tárgy 1>, <kézben lévő tárgy 2>, ...

stunRemaining: <stunRemaining tagváltozó értéke>

“

Teacher objektum esetén:

“

<objektumnév>

location: <location tagváltozó értéke>

itemsInHand: <kézben lévő tárgy 1>, <kézben lévő tárgy 2>, ...

stunRemaining: <stunRemaining tagváltozó értéke>

“

Cleaner objektum esetén:

“

<objektumnév>

location: <location tagváltozó értéke>

itemsInHand: <kézben lévő tárgy 1>, <kézben lévő tárgy 2>, ...

stunRemaining: <stunRemaining tagváltozó értéke>

“

Camembert objektum esetén:

“

<objektumnév>

location: <location tagváltozó értéke>

holder: <holder tagváltozó értéke>

“

AirFresher objektum esetén:

“

<objektumnév>

location: <location tagváltozó értéke>

holder: <holder tagváltozó értéke>

“

Transistor objektum esetén:

“

<objektumnév>

location: <location tagváltozó értéke>

holder: <holder tagváltozó értéke>

pair: <pair tagváltozó értéke>

“

TVSZ objektum esetén:

“

<objektumnév>

location: <location tagváltozó értéke>

holder: <holder tagváltozó értéke>

usesRemaining: <usesRemaining tagváltozó értéke>

“

FalseTVSZ objektum esetén:

“

<objektumnév>

location: <location tagváltozó értéke>

holder: <holder tagváltozó értéke>

usesRemaining: <usesRemaining tagváltozó értéke>

“

SlideRule objektum esetén:

“

<objektumnév>

location: <location tagváltozó értéke>

holder: <holder tagváltozó értéke>

“

FalseSlideRule objektum esetén:

```
“
<objektumnév>
location: <location tagváltozó értéke>
holder: <holder tagváltozó értéke>
“
```

Rag objektum esetén:

```
“
<objektumnév>
location: <location tagváltozó értéke>
holder: <holder tagváltozó értéke>
activated: <activated tagváltozó értéke>
timeRemaining: <timeRemaining tagváltozó értéke>
“
```

BeerGlass objektum esetén:

```
“
<objektumnév>
location: <location tagváltozó értéke>
holder: <holder tagváltozó értéke>
activated: <activated tagváltozó értéke>
timeRemaining: <timeRemaining tagváltozó értéke>
“
```

Mask objektum esetén:

```
“
<objektumnév>
location: <location tagváltozó értéke>
holder: <holder tagváltozó értéke>
activated: <activated tagváltozó értéke>
timeRemaining: <timeRemaining tagváltozó értéke>
duration: <duration tagváltozó értéke>
“
```

FalseMask objektum esetén:

```
“
<objektumnév>
location: <location tagváltozó értéke>
holder: <holder tagváltozó értéke>
activated: <activated tagváltozó értéke>
timeRemaining: <timeRemaining tagváltozó értéke>
duration: <duration tagváltozó értéke>
“
```

merge <szobanév1><szobanév2><új szobanév>

- Ha a szobák nem szomszédosak: “*Error: Rooms <szobanév> and <szobanév> are not neighbours.*”
- Ha az egyesítendő két szoba megegyezik: “*Error: Room objects must differ.*”

- Ha a kért név már foglalt: “*Error: Name <új szobanév> already exists.*”

split <szobanév> <új szobanév>

- Ha a kért név már foglalt: “*Error: Name <új szobanév> already exists.*”

activate <tárgynév>

setpair <tranzisztornév1> <tranzisztornév2>

- Ha bármelyik tranzisztornak van már párja (mindkét tranzisztorra, ha mindeneketőnek van): “*Error: Transistor is already paired.*”
- Ha a két tranzisztor megegyezik: “*Error: Transistor objects must differ.*”

elapsetime <érték>

setremaining <maszk/TVSZ név> <érték>

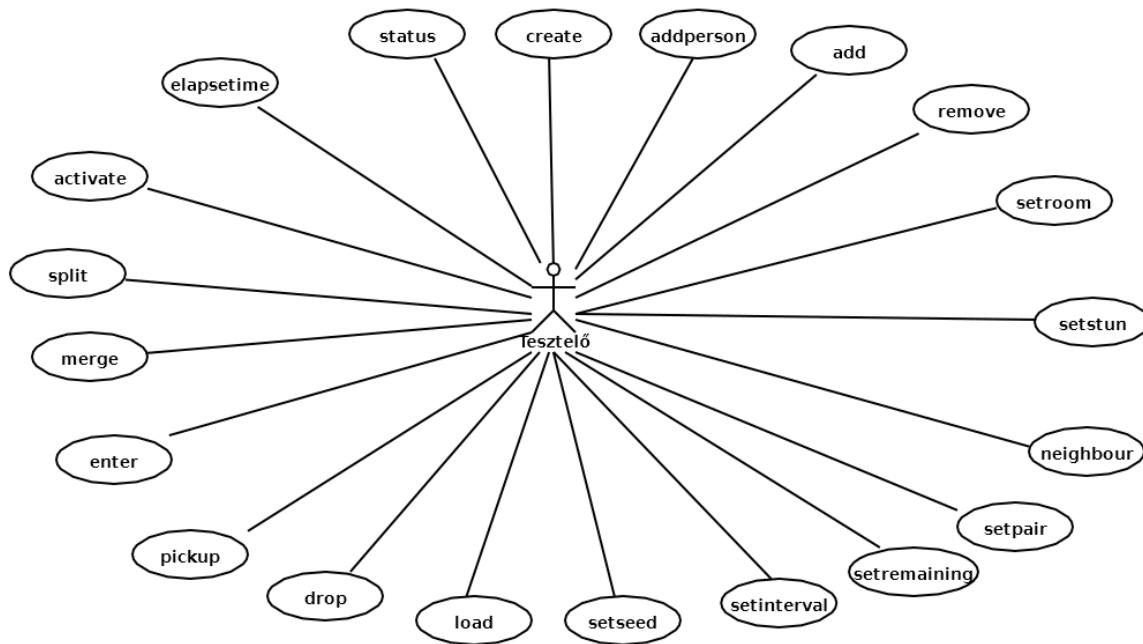
setinterval <tulajdonság> <IntervalItem név> <érték>

setseed <személy név> <érték>

load <fájlnév>

Kimenet sikeres lefutás esetén: A betöltött parancsok kimenete.

6.2 Összes részletes use-case



Use-case neve	create
Rövid leírás	Egy objektum létrehozása adott névvel és típussal.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program létrehozza a kért objektumot a kért névvel és típussal, majd annak attribútumait az alapértékekre állítja. Szükség esetén hibaüzenetet küld.

Use-case neve	addperson
Rövid leírás	Adott személy hozzáadása egy létező szobához.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program hozzáadja a parancsban megadott személyt a megadott szobához. Amennyiben a személy vagy a szoba nem létezik, vagy a személy már tartózkodik egy szobában, hibaüzenetet küld a tesztelőnek.

Use-case neve	add
Rövid leírás	Egy adott tárgy odaadása egy adott ItemHandler-nek.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program odaadja a kiválasztott tárgyat a kiválasztott ItemHandler-nek, ha a tárgy és az ItemHandler is léteznek és a tárgy még nem rendelkezik birtokossal vagy tartózkodási helyvel, különben megfelelő hibaüzenetet küld a tesztelőnek.

Use-case neve	remove
----------------------	--------

Rövid leírás	Egy adott tárgy elvétele egy adott ItemHandler-től
Aktorok	Tesztelő
Forgatókönyv	A prototípus program a kért tárgyat eltávolítja az adott ItemHandler birtokából, ha a tárgy és az ItemHandler is léteznek és a tárgy az ItemHandler-nél van, egyébként megfelelő hibaüzenetet küld a tesztelőnek.

Use-case neve	setroom
Rövid leírás	Beállítja a megadott szoba egy állapotváltozóját a kért értékre.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program beállítja a megadott szoba megadott állapotváltozóját, ha az adott szoba létezik, van ilyen állapotváltozója és azt a megadott értékre lehet állítani, egyébként megfelelő hibaüzenetet küld a tesztelőnek.

Use-case neve	setstun
Rövid leírás	Egy adott Person elkábítása vagy kábításának megszüntetése.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program az adott Person-nél beállítja a megadott érték szerint, hogy az elkábult-e, vagy sem, ha a Person létezik, és a beállítani kívánt érték érvényes, egyébként megfelelő hibaüzenetet küld a tesztelőnek.

Use-case neve	neighbour
Rövid leírás	Két szoba közt egyirányú szomszédság bevezetése
Aktorok	Tesztelő
Forgatókönyv	A prototípus program a két adott szoba közötti megfelelő irányú szomszédságot állítja be, ha az értelmezhető, minden szoba létezik, és a konkrét szomszédság nem létezett eddig. Különben hibaüzenetet küld a tesztelőnek.

Use-case neve	setpair
Rövid leírás	Két tranzisztor párosítása.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program párosítja a két megadott tranzisztor, ha azok léteznek és még egyiknek sincs párja, egyébként megfelelő hibaüzenetet küld a tesztelőnek.

Use-case neve	setremaining
Rövid leírás	Hátralévő használatok számának beállítása.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program beállítja a hátralévő használatok számát a megadott tárgynál, ha az létezik és helyes, és a beállítandó érték érvényes, egyébként megfelelő hibaüzenetet küld a tesztelőnek.

Use-case neve	setinterval
Rövid leírás	Aktív állapot és hátralévő aktív idő mennyiségének beállítása.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program beállítja a megadott létező tárgy aktív állapotát és a megmaradt aktív idő mennyiségét, szintaktikailag helyes parancs esetén, egyébként megfelelő hibaüzenetet küld a tesztelőnek.

Use-case neve	setseed
Rövid leírás	Aktív állapot és hátralévő aktív idő mennyiségének beállítása.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program beállítja a megadott létező tárgy aktív állapotát és a megmaradt aktív idő mennyiségét, szintaktikailag helyes parancs esetén, egyébként megfelelő hibaüzenetet küld a tesztelőnek.

Use-case neve	load
Rövid leírás	Tesztelési állapot betöltése.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program egy fájlból tölt be parancsokat, ha a fájl létezik és olvasható, egyébként hibaüzenetet küld a tesztelőnek. A parancsokat lefuttatja, ezáltal azoknak a kimeneteit illetve hibaüzeneteit is megjeleníti.

Use-case neve	drop
Rövid leírás	Egy adott tárgy eldobatása egy Person kezéből.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program eldobat egy kért tárgyat a megadott Person kezéből, ha a tárgy és a Person is léteznek és a tárgy a Person kezében van, egyébként megfelelő hibaüzenetet küld a tesztelőnek.

Use-case neve	pickup
Rövid leírás	Felvetet egy adott szobában lévő Person-nal egy tárgyat.
Aktorok	Tesztelő
Forgatókönyv	A prototípus felveteti a kért tárgyat a megadott Person-nal, ha a tárgy és a Person is léteznek és emellett azonos szobában is vannak, egyébként megfelelő hibaüzenetet küld a tesztelőnek.

Use-case neve	enter
Rövid leírás	Egy Person beléptetése egy adott szobába.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program kezdeményezi a megadott Person beléptetését a megadott szobába, ha a Person és a szoba is léteznek és a Person tartózkodási helye megfelelő irányban szomszédos a megadott szobával. Egyébként megfelelő hibaüzenetet küld a tesztelőnek.

Use-case neve	merge
Rövid leírás	Két szoba egyesítése egy szobává.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program a kért szobát egy újonnan létrehozott szobába egyesíti amennyiben a szobák léteznek és szomszédosak voltak, egyébként megfelelő hibaüzenetet küld a tesztelőnek.

Use-case neve	split
Rövid leírás	Egy szoba két szobává választása.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program a megadott szobát két szobává választja, ha a kért szoba létezik, egyébként hibaüzenetet küld a tesztelőnek.

Use-case neve	activate
Rövid leírás	Egy megadott tárgy aktiválása.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program aktiválja a megadott tárgyat, ha az létezik, egyébként hibaüzenetet küld a tesztelőnek.

Use-case neve	elapsetime
Rövid leírás	Adott számú játékbeli időegység eltelése.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program megadott számú időegység eltelését szimulálja.

Use-case neve	status
Rövid leírás	A választott objektum adatainak lekérdezése.
Aktorok	Tesztelő
Forgatókönyv	A prototípus program lekérdezi a megadott objektum attribútumait, és kiírja azokat a standard kimenetre, ha a keresett objektum létezik, egyébként hibaüzenetet küld a tesztelőnek.

6.3 Tesztelési terv

Teszt-eset neve	Átlépés oktatóhoz nem védő tárgyakkal 1
Rövid leírás	Két szomszédos szoba, az egyikbe egy hallgató, a másikba egy oktató létrehozása. A hallgatónak adni egy FalseTVSZ-t, egy AirFresher-t, egy Camambert-et, egy Transistor és egy Rag-et. A hallgató átléptetése a másik szobába.
Teszt célja	A szobák közti mozgás, a kibuktatás és tárgyak kibuktatás elleni védelmének tesztelése.

Teszt-eset neve	Átlépés oktatóhoz nem védő tárgyakkal 2
Rövid leírás	Kétszomszédos szoba, az egyikbe egy hallgató, a másikba egy oktató létrehozása. A hallgatónak adni egy aktiválatlan BeerGlass-t és egy Mask-ot. A hallgató átléptetése a másik szobába.
Teszt célja	A szobák közti mozgás, a kibuktatás és tárgyak kibuktatás elleni védelmének tesztelése.

Teszt-eset neve	Átlépés oktatóhoz TVSZ-szel
Rövid leírás	Kétszomszédos szoba, az egyikbe egy hallgató, a másikba egy oktató létrehozása. A hallgatónak adni egy TVSZ-t. A hallgató átléptetése a másik szobába.
Teszt célja	A szobák közti mozgás, a kibuktatás és TVSZ kibuktatás elleni védelmének tesztelése.

Teszt-eset neve	Átlépés oktatóhoz BeerGlass-szal
Rövid leírás	Kétszomszédos szoba, az egyikbe egy hallgató, a másikba egy oktató létrehozása. A hallgatónak adni egy BeerGlass-t. A BeerGlass aktiválása. A hallgató átléptetése a másik szobába.
Teszt célja	A szobák közti mozgás, a kibuktatás és BeerGlass kibuktatás elleni védelmének tesztelése.

Teszt-eset neve	Átlépés gázba nem védő tárgyakkal 1
Rövid leírás	Két szomszédos szoba, melyek közül az egyik gázos és a másikba egy hallgató létrehozása. A hallgatónak adni egy TVSZ-t, egy AirFresher-t, egy Camembert-t, egy Transistor-t és egy Rag-et. A hallgató átléptetése a másik szobába.
Teszt célja	A szobák közti mozgás és a tárgyak gáz elleni védelmének tesztelése.

Teszt-eset neve	Átlépés gázba nem védő tárgyakkal 2
Rövid leírás	Két szomszédos szoba, melyek közül az egyik gázos és a másikba egy hallgató létrehozása. A hallgatónak adni egy BeerGlass-t és egy FalseMask-ot. A hallgató átléptetése a másik szobába.
Teszt célja	A szobák közti mozgás és a tárgyak gáz elleni védelmének tesztelése.

Teszt-eset neve	Átlépés gázba Mask-kal
Rövid leírás	Két szomszédos szoba, melyek közül az egyik gázos és a másikba egy hallgató létrehozása. A hallgatónak adni egy Mask-ot. A hallgató átléptetése a másik szobába.
Teszt célja	A szobák közti mozgás és a Mask gáz elleni védelmények tesztelése.

Teszt-eset neve	Takarító átléptetése
Rövid leírás	Két szomszédos szoba létrehozása, melyekből az egyik gázos és bele egy hallgató, a másikba egy takarító létrehozása. A gázos szoba stickiness-ének beállítása. A takarító átléptetése a szomszédos szobába.
Teszt célja	A takarító mozgásának és a szoba tisztításának tesztelése.

Teszt-eset neve	Oktató találkozása személyekkel és tárgyakkal
Rövid leírás	Két szomszédos szoba létrehozása. A második szoba kapacitásának beállítása 5-re. Az egyikbe egy oktató és hallgató, a másikba egy hallgató, egy oktató és egy takarító létrehozása. Az első szobában lévő hallgató kezébe egy rongy létrehozása, hozzáadása és aktiválása. A második szobába továbbá minden tárgyból egy példány létrehozása és hozzáadása. Az első szobában lévő hallgató átléptetése a másik szobába, majd az egyedül lévő oktató átléptetése a másik szobába.
Teszt célja	A különböző személyek megbuktatásra, találkozására való reagálásának tesztelése. Személyek és tárgyak találkozásának tesztelése.

Teszt-eset neve	Tárgy felvétele és lerakása
Rövid leírás	Egy szoba létrehozása benne egy hallgatóval és 2 TVSZ-szel. A hallgató kezébe 4 TVSZ létrehozása. A hallgató megpróbálja felvenni először egyik, majd a másik a földön lévő TVSZ-t, majd lerak egyet a nála lévő TVSZ-ekből.
Teszt célja	A Room és a Student tárgykezeléssel kapcsolatos függvényeinek tesztelése.

Teszt-eset neve	Tárgyak időtelése
Rövid leírás	Egy szoba és benne két hallgató és egy oktató létrehozása. Az egyik hallgató kezébe egy 2 időegység múlva lejáró Rag, a másik kezébe egy a Rag bénítási idejénél később lejáró aktivált BeerGlass és egy tartóssággal már nem rendelkező, nemsokára lejáró aktivált Mask létrehozása. Az idő léptetése addig, amire már a BeerGlass ideje is lejár.
Teszt célja	Az időérzékeny objektumok viselkedésének tesztelése.

Teszt-eset neve	Szobák módosítása
Rövid leírás	Egy szoba létrehozása 3 tárggyal benne, gázos tulajdonság igazra állítása. 3, az eredeti szobával szomszédos szoba létrehozása. A szoba felosztása, majd egyesítése a felosztás által újonnan létrehozott szobával.
Teszt célja	Szoba szétválásának és szobák összeolvadásának tesztelése.

Teszt-eset neve	Gázhoz kapcsolódó tárgyak aktiválása
Rövid leírás	Egy szoba és benne egy hallgató létrehozása. A hallgatónak a kezébe egy Mask, egy Camembert és egy AirFresher létrehozása. A Camembert, majd az AirFresher aktiválása.
Teszt célja	A Camembert és az AirFresher működésének tesztelése.

Teszt-eset neve	Transistor használata
Rövid leírás	Három szoba létrehozása, melyek közül a középső szomszédos a két szélsővel. Egy szélsőbe egy hallgató létrehozása kezében két Transistor-ral. A Transistorok aktivállással való párosítása, majd az egyik kidobása. A hallgató átléptetése a középső, majd a másik szélső szobába és a nála lévő Transistor aktiválása.
Teszt célja	A Transistor működésének tesztelése.

Teszt-eset neve	Bénított hallgató
Rövid leírás	Két szomszédos szoba és az egyikbe egy hallgató létrehozása. A hallgató bénítása majd átléptetése a másik szobába.
Teszt célja	Bénultság tesztelése.

Teszt-eset neve	Elátkozott szoba
Rövid leírás	Két szomszédos szoba létrehozása, melyek közül az egyik elátkozott. Az elátkozott szobába két hallgató létrehozása. Ez egyik hallgató átléptetése a másik szobába. Az idő lejtése, hogy aktívvá váljon az átok. A másik hallgató átléptetés a másik szobába. Az idő lejtése, hogy inaktívvá váljon az átok. A másik hallgató átléptetésének megismétlése.
Teszt célja	Elátkozódás tesztelése.

Teszt-eset neve	Teli szoba
Rövid leírás	Két szomszédos szoba és az egyikbe egy hallgató létrehozása. A másik szoba kapacitásának 0-ra állítása. A hallgató átléptetése a másik szobába.
Teszt célja	Teli szoba tesztelése.

Teszt-eset neve	Tanári logarléc
Rövid leírás	Egy szoba, benne egy oktató és egy SlideRule létrehozása. Az oktató felveszi a SlideRule-t.
Teszt célja	Teacher és SlideRule interakciójának tesztelése.

Teszt-eset neve	Cleaner logarléc
Rövid leírás	Egy szoba, benne egy takarító és egy SlideRule létrehozása. A takarító felveszi a SlideRule-t.
Teszt célja	Cleaner és SlideRule interakciójának tesztelése.

Teszt-eset neve	Cleaner átlép cursed szobába
Rövid leírás	Két szomszédos szoba létrehozása, melyekből az egyik gázos és elátkozott, és bele egy hallgató, a másikba egy takarító létrehozása. A gázos szoba stickiness-ének beállítása. A takarító átléptetése a szomszédos szobába.
Teszt célja	A takarító mozgásának tesztelése.

Teszt-eset neve	Sikertelen tranzisztor párosítás
Rövid leírás	Két szomszédos szoba létrehozása. Az egyikben egy hallgató és a kezében két Transistor létrehozása. Egy Transistor aktiválása, majd eldobása. A másik Transistor aktiválása. A hallgató átmozgatása a másik szobába, majd a nála lévő Transistor aktiválása.
Teszt célja	Transistor-ok párosíthatóságának ellenőrzése.

Teszt-eset neve	Ragacsos szoba
Rövid leírás	Egy szoba és benne egy hallgató és egy TVSZ létrehozása. A szoba ragacsosságának határérték fölé állítása. A TVSZ hallgató általi felvétele.
Teszt célja	Ragacsosság hatásának a tárgyfelvételre tesztelése.

Teszt-eset neve	Sikertelen kettéválás és osztódás
Rövid leírás	Két szomszédos szoba létrehozása, az egyikbe egy hallgatóval. A hallgatót tartalmazó szoba felosztása, majd a két szoba egyesítése minden paraméterező sorrenddel.
Teszt célja	Szobaegyesülés és szétválás sikertelen lefutásának tesztelése.

Teszt-eset neve	Takarító általi mozgatás
Rövid leírás	Három szoba létrehozása, a középső kölcsönösen szomszédos a két szélsszel. A középső szobához 2 hallgató adása. Az egyik szélsszobához egy takarító adása. A takarító szobájának kapacitása 1-re állítása. A másik szélsszoba kapacitásának 0-ra állítása. A takarító középső szobába lépése.
Teszt célja	Takarító szobába lépésének és ezáltal más szereplők mozgatásának tesztelése.

Teszt-eset neve	Tárgyak eltűnése és aktivált rongy földön viselkedése
Rövid leírás	Két szomszédos szoba létrehozása. Az egyikhez egy oktató hozzáadása. Az oktató kezébe egy maszk létrehozása. A maszk aktiválása, timeRemaining-jének 1-re és duration-jének 0-ra állítása. Az oktató szobájába egy söröspohár tétele, aktiválása és timeRemaining-jének 1-re állítása. A másik szobába egy rongy rakása, aktiválása és timeRemaining-jének 1-re állítása. Az oktató másik szobába lépése, majd 1 egységnyi idő telésének jelzése. Az oktató eredeti szobába való (sikertelen) visszaléptetése.
Teszt célja	Oktató és rongy találkozásának, kábult ember mozgásának, valamint tárgyak eltűnésének tesztelése.

Teszt-eset neve	Ragacsosság és sörösüveg
Rövid leírás	Két szomszédos szoba létrehozása. Az egyikbe egy hallgató hozzáadása és egy sör a földre. A sör aktiválása. Egy tranzisztor földre rakása. A másik szobába egy oktató rakása. A hallgató szobájának ragacsosság beállítása éppen a határérték alá. Hallgató seed-jének beállítása. Tranzisztor és sör sikeres felvétele. Oktató szobába lépése. Hallgató megvédi magát, de elejti a tranzisztort (pszeudo-randomitás). Tranzisztor felvétele (sikertelenül).
Teszt célja	Stickiness általi különböző viselkedés, sörösüveg használatának tesztelése

6.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása

A tesztelést egy Windows parancssor alatt futtatható batch script fájl fogja segíteni. Ez megkérdezi a felhasználótól, melyik tesztet (vagy esetleg az összeset) kívánja futtatni, majd futtatja a tesztet a kért teszthez tartozó bemeneti fájl megadásával, majd a kimenetet egy ideiglenes fájlba irányítja. Ezután ellenőrzi az FC parancs segítségével az átirányított kimeneti fájl egyenlőségét az elvárt kimeneti fájlhoz képest. Egy teszt akkor sikeres, ha az FC kimenete az, hogy a két fájl között nincs különbség.

6.5 Napló

Kezdet	Időtartam	Részttvevők	Leírás
2024. 03. 27. 10:00	2 óra	Görömbey Héjja Sági Tömöri Vizhányó	Konzultáció, másik csapat modelljének tesztelése
2024. 03. 27. 14:30	2 óra	Görömbey	Változások dokumentálása az osztálydiagramon.
2024. 03. 27. 14:30	2 óra	Tömöri	Változások dokumentálása a leírásban.
2024. 04. 04. 10:00	3 óra	Vizhányó	Szekvenciadiagramok frissítése 7.0.3.2 fele
2024. 04. 05. 15:00	3 óra	Tömöri	Változások dokumentálása és az alapján a módosítandó és elkészítendő szekvenciák felsorolása rövid leírással.
2024. 04. 06. 8:30	5 óra	Tömöri	Elavult szekvenciák módosítása. 7.0.3.2 fele és 7.0.3.3
2024. 04. 06. 9:00	5 óra	Vizhányó	Újítások miatti szekvenciák létrehozása 7.0.3.1
2024. 04. 06. 21:00	2 óra	Sági	A bemeneti nyelv tervezésének megkezdése
2024. 04. 07. 10:30	1,5 óra	Héjja	A kimeneti nyelv tervezésének megkezdése
2024. 04. 07. 13:00	1 óra	Sági	A bemeneti nyelv tervezése

2024. 04. 07. 14:00	2,5 óra	Héjja	A kimeneti nyelv tervezése, use-case diagram elkészítése, use-case leírások elkészítésének megkezdése
2024.04.07 18:00	2 óra	Görömbey	Tesztek írása
2024. 04. 07. 18:00	5 óra	Sági	A bemeneti nyelv tervezése és pontosítása, kimeneti nyelv pontosítása
2024. 04. 07. 18:45	2 óra	Héjja	Use-case-ek leírása, kimeneti nyelv és use-case diagram ellenőrzése és javítása
2024. 04. 07. 19:00	3 óra	Tömöri	Bemeneti nyelv és use-case javítások.
2024. 04. 07. 20:00	1.5 óra	Vizhányó	Szekvencia diagramok javítása, tesztek átnézése, hiányzó tesztek hozzáadása

7. Részletes tervezettségek

7.0 Prototípus-interface újításai

7.0.1 Bemeneti nyelv újítása

7.0.1.1 create <típus> <név> parancs módosítása

Új szoba létrehozásakor a stickiness-t nullával inicializálja.

7.0.1.2 status <objektumnév> parancs kiegészítése

Objektumnév megadása opcionális: amennyiben csak a status szó kerül beírásra, a parancs kiírja a játék állapotát. Egyéb esetben az eddig definiált módon viselkedik a parancs.

7.0.2 Kimeneti nyelv újítása

7.0.2.1 status <objektumnév> parancs kiegészítése

Amikor az objektum egy listájának kiírása történik meg, akkor az elemek a program listájának tárolási sorrendjében kerülnek megjelenítésre. Ez a sorrend megegyezik a listába hozzáadás sorrendjével. Ez történik a szobák állapotánál a neighbours, peopleInRoom és itemsInRoom listáknál, valamint a személyek itemsInHand listájánál. Ha a lista üres, csak a lista neve és közvetlen utána “:” jelenik meg a sorban. Egyéb esetben minden elem után beszűrunk egy vessző+szóköz elválasztást “,” úgy, hogy az utolsó elem után is még kiírásra kerül ez az elválasztó.

Amennyiben nem adtak meg objektumnevet a játék állapotát kérdezték le. Ez alapján 3 üzenet jelenhet meg:

„Win”

Jelentése: A játék szobáinak egyikében található egy hallgató logarléccel a kezében.

„Game in progress”

Jelentése: A játék szobáiban található legalább egy hallgató, de egyik kezében sincs logarléc.

„Game over”

Jelentése: A játék szobáinak egyikében sem található egy hallgató sem. minden hallgató kibukott.

7.1 Osztályok és metódusok tervezése

7.1.1 AirFresher

- **Felelősség**

A légfrissítő aktiváláskor megszünteti a szobájának gázosságát. A tárgy felelőssége továbbá önmaga megszűnésének kezdeményezése.

- **Ősosztályok**

Item

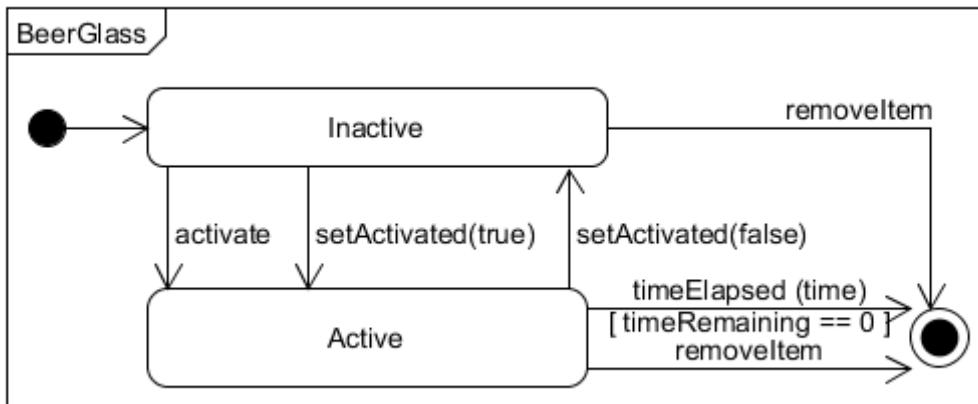
- **Metódusok**

- **+AirFresher(Room location, Person holder)**: Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az űsének a konstruktörét, ezzel beállítva tagváltozóinak értékét.
- **+void activate()**: A légfrissítő aktiválása. A tartózkodási szobáján meghívja a setGas(false) metódust ezzel megszüntetve az esetleges mérgező gázt. Ezután igényli birtokosánál a tárgy eltávolítását.
- **+void meet(Person person)**: Egy személlyel való találkozás kezelése. A metódus üres, mert minden tárgy csak akkor találkozik személlyel, ha földön van, és a légfrissítőnek ekkor nincs teendője.
- **+boolean saveFromDeath(Person killer)**: Kibukás elleni védelem kezelése. Mivel a légfrissítő nem véd a kibukástól, a metódus logikai hamissal tér vissza minden esetben, és nem történik más.
- **+bool saveFromGas()**: Mérgező gáz elleni védelem kezelése. Mivel a légfrissítő nem nyújt védelmet a gáz ellen, a metódus logikai hamissal tér vissza minden esetben, és nem történik más.
- **+void timeElapsed(int time)**: Idő telésének kezelése camemberten. Mivel a légfrissítő egyszerhasználatos tárgy, így nem történik vele semmi az idő műlásával, a metódus üres.

7.1.2 BeerGlass

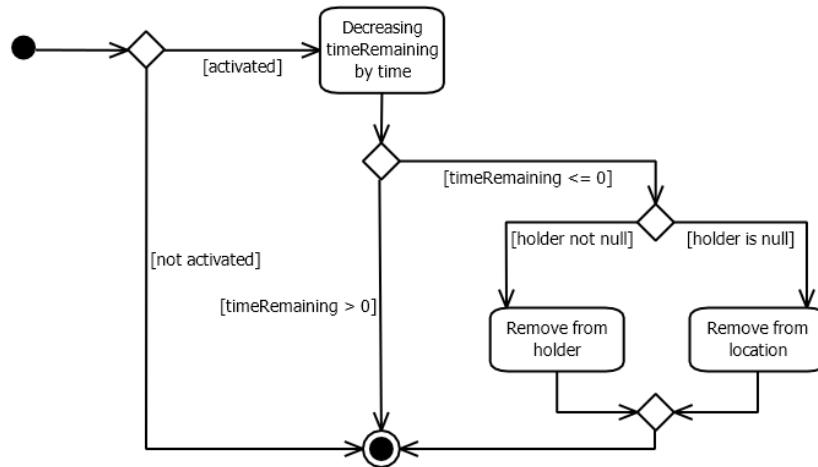
- **Felelősség**

A szent söröspoharak adott ideig védelmet nyújtanak az oktatókkal szemben. A hallgató képes aktiválni, innentől a söröspohár felelőssége, hogy kezelje magát. Védelem nyújtása esetén a birtokosa véletlenszerűen eleji az egyik tárgyat a kezéből. Amikor elmúlik a hatása a megszűnésétől kezdeményezi a birtokosánál.



1. állapotgép: A BeerGlass osztály állapotgépe

- **Ősosztályok**
Item → IntervalItem
- **Metódusok**
 - **+BeerGlass(Room location, Person holder, boolean activated, int timeRemaining)**: Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az ősének a konstrukturát, ezzel beállítva tagváltozónak értékét.
 - **+void meet(Person person)**: Egy személlyel való találkozás kezelése. A metódus üres, mert minden tárgy csak akkor találkozik személlyel, ha földön van, és a söröspohárnak ekkor nincs teendője.
 - **+boolean saveFromDeath(Person killer)**: Kibuktatástól való megmentés kezelése. Amennyiben aktiválva van a söröspohár, kezdeményezi a tulajdonosánál egy véletlen tárgy elejtését a holder-je dropRandomItem() metódusának meghívásával, majd logikai igazzal tér vissza, megvédve a birtokosát. Egyéb esetben logikai hamissal tér vissza, nem nyújt védelmet.
 - **+boolean saveFromGas()**: Mérgező gáz elleni védelem kezelése. Mivel a söröspohár nem nyújt védelmet a gáz ellen, a metódus logikai hamissal tér vissza minden esetben, és nem történik más.
 - **+void timeElapsed(int time)**: Idő telésének kezelése a söröspoháron. Ha a tárgy nincs aktiválva, nem történik semmi. Ha aktiválva van a tárgy, akkor a timeRemaining értékét csökkenti time-mal. Ekkor ha a timeRemaining értéke kisebb vagy egyenlő mint nulla, akkor az objektum az aktuális birtokosánál kezdeményezi a tárgy megsemmisítését. Ha a tárgynak a holder-e nullpointer, akkor a birtokosa a location-je, egyéb esetben a holder-je a birtokos.

1. aktivitás diagram: `BeerGlass timeElapsed(int time)` metódusa

7.1.3 Camembert

- **Felelősség**

A Camembert sajt aktiváláskor mérgező gázzal árasztja el a szobát. A sajt felelőssége a szoba elgázosításának és önmaga megszűnésenek kezdeményezése.

- **Ősosztályok**

Item

- **Metódusok**

- **+Camembert(Room location, Person holder):** Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az ősének a konstruktort, ezzel beállítva tagváltozóinak értékét.
- **+void activate():** A camembert aktiválása. Először igényli birtokosánál a tárgy eltávolítását, és utána a tartózkodási szobáján meghívja a setGas(true) metódust ezzel elgázosítva azt.
- **+void meet(Person person):** Egy személlyel való találkozás kezelése. A metódus üres, mert minden tárgy csak akkor találkozik személlyel, ha földön van, és a camembertnek ekkor nincs teendője.
- **+boolean saveFromDeath(Person killer):** Kibukás elleni védelem kezelése. Mivel a camembert nem véd a kibukástól, a metódus logikai hamissal tér vissza minden esetben, és nem történik más.
- **+bool saveFromGas():** Mérgező gáz elleni védelem kezelése. Mivel a camembert nem nyújt védelmet a gáz ellen, a metódus logikai hamissal tér vissza minden esetben, és nem történik más.
- **+void timeElapsed(int time):** Idő telésének kezelése camemberten. Mivel a camembert egyszerhasználatos tárgy, így nem történik vele semmi az idő műlásával, a metódus üres.

7.1.4 Cleaner

- **Felelősség**

A takarítókat reprezentáló osztály. Felelőssége, hogy a takarítók is mozognak, ezáltal takarítva a szobákat. Megszüntetik a mérgező gázt a szobákban, és kitakarításuk következményeképp nullázódik a ragacsosság, és elmozgatja a többi személyt. Nem lehet nála Logarléc, nem tud elkábulni.

- **Ősosztályok**

Person

- **Metódusok**

- **+Cleaner(int stunRemaining, Room location)**: Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az űsének a konstruktort, ezzel beállítva tagváltozónak értékét.
- **+void enterRoom(Room roomTo)**: A takarító mozgását végrehajtó metódus. Lokálisan eltárolja a tartózkodási helyét. A takarító mozgása ugyanúgy kezdődik, mint a többi személyé, ezért meghívja az űsének ugyanilyen nevű metódusát a paramétert is átadva. Ezután ha a tartózkodási helye nem egyezik meg a metódushívás előtt eltárolttal (tehát másik szobába lépett), kiszellőzeti az esetleges mérgező gázt a location-jének setGas(false) metódushívásával, majd kitakarítja a szobát annak clean() metódusát meghívva.
- **+void meet(Person person)**: Egy személlyel való találkozást kezeli le. Nem csinál semmit, üres metódus.
- **+void greet(Person greeter)**: A takarító köszönésre reagálása. Nem csinál semmit, üres metódus.
- **+void kill(Person killer)**: A takarító megtámadását kezeli le. Mivel a takarítók nem bukhatnak ki, a metódus üres.
- **+void slip()**: Rongy miatti elkábulás. Mivel a takarítókat a rongy nem fenyegeti, a metódus üres.
- **+void stun()**: Mérgező gáz miatti elkábulás felüldefiniálása. Mivel a takartók nem kábulhatnak el, a metódus üres.
- **+void pickedUpSlideRule(SlideRule slideRule)**: Logarléc felvétele. A takarítók nem vehetik fel a Logarlécet, ezért kidobják a paraméterként kapott slideRule-t a kezüköböl a szobába.

7.1.5 FalseMask

- **Felelősség**

A FalseMask osztály a hamis FFP2-es maszk működését és viselkedését modellezzi. A Maskhoz hasonlóan működik, azt leszámítva, hogy nem nyújt védelmet a mérgező gázzal szemben.

- **Ősosztályok**

Item → IntervalItem → Mask

- **Metódusok**
- **+FalseMask(Room location, Person holder, boolean activated, int timeRemaining)**: Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az ōsének a konstruktorát, ezzel beállítva tagváltozónak értékét.
- **+boolean saveFromGas()**: Mérgező gáz elleni védelem kezelése. Meghívja az ōsének azonos nevű metódusát, hogy amennyiben eddig nem volt aktiválódva, most aktiválódjon, ugyanakkor a hamis FFP2-es maszk nem véd meg a kibukástól, a metódus visszatérési értéke mindenhamis.

7.1.6 FalseSlideRule

- **Felelősség**

A FalseSlideRule osztály a hamis Logarléc működését és viselkedését modellezzi. A hamis Logarléc átveri a hallgatókat, ennek birtoklásával nem lehet gyözni. Ezen kívül ugyanúgy működik, mint a SlideRule.

- **Ősosztályok**

Item → SlideRule

- **Metódusok**

- **+FalseSlideRule(Room location, Person holder)**: Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az ōsének a konstruktorát, ezzel beállítva tagváltozónak értékét.

7.1.7 FalseTVSZ

- **Felelősség**

A TVSZ denevérbőrre írt példányának másolata a tárgy látszatával ellentében haszontalan, nem véd meg a kibukástól. A többi funkcióját tekintve megegyezik az ōsével.

- **Ősosztályok**

2. *Item → TVSZ*

- **Metódusok**

- **+FalseTVSZ(Room location, Person holder)**: Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az ōsének a konstruktorát, ezzel beállítva a megfelelő tagváltozónak értékét.
- **+boolean saveFromDeath(Person killer)**: Kibukás elleni védelem kezelése. Meghívja az ōsének azonos nevű metódusát, hogy a usesRemaining csökkenjen, ugyanakkor a hamis TVSZ nem véd meg a kibukástól, a metódus visszatérési értéke mindenhamis.

7.1.8 IntervalItem

- **Felelősség**

Azon tárgyak, melyek aktiválhatók és csak adott ideig fejtik ki hatásukat. Felelősségeik tudni, hogy aktiválva vannak-e, és kezelni a hatásidéjét. Amennyiben aktiválva vannak hatással lehetnek a személyekre. Az IntervalItem osztály absztrakt, nem példányosítható.

- **Ősosztályok**
 - Item*
- **Attribútumok**
 - **#boolean activated**: Ez a logikai változó tárolja, hogy a tárgy aktiválva van-e vagy sem. Értéke logikai igaz, ha a tárgy aktív és hamis, ha nincs aktiválva.
 - **#int timeRemaining**: Ez az egész szám tárolja, hogy mennyi a hátralévő idő, amíg a tárgy aktiválva használható.
- **Metódusok**
 - **+IntervalItem(Room location, Person holder, boolean activated, int timeRemaining)**: Az osztály konstruktora. A kapott location és holder paraméterekkel meghívja az űsének a konstruktörét, ezzel beállítva a tartózkodási helyét és birtokosát, majd az activated és timeRemaining változóinak értékét beállítja a paraméterként kapottakra.
 - **+void activate()**: Az adott tárgy aktiválása. Az activated változót igazra állítja be.
 - **+boolean isActivated()**: Az aktiváltság lekérdezése. Az activated változó értékével tér vissza.
 - **+void setActivated(boolean activated)**: Az aktiváltság beállítása. Az objektum átállítja az activated változójának értékét a paraméterként kapotttra.
 - **+int getTimeRemaining()**: A hátralévő hatásidő lekérdezése. A timeRemaining változó értékével tér vissza.
 - **+void setTimeRemaining(int timeRemaining)**: A hátralévő hatásidő beállítása. Az objektum átállítja a timeRemaining változójának értékét a paraméterként kapotttra.

7.1.9 Item

- **Felelősség**

A tárgyak a szereplőket segítő eszközök, melynek használatával különböző előnyökre lehetnek szert. Egy tárgy lehet egy szobában, vagy egy embernél, ezt az információt a tárgy is tárolja. Egy tárgy felelőssége az aktiválásra, a veszélyektől való védelmi kérésekre és az idő telésére a megfelelő reagálás. Az Item egy absztrakt osztály, csak a nem absztrakt leszármazottjai példányosíthatók.
- **Interfészek**
 - TimeSensitive
- **Attribútumok**
 3. **#Room location**: Ez a Room típusú változó tárolja, hogy mi a tárgy tartózkodási szobája. Ha egy ember kezében van, akkor az ember tartózkodási helyével egyezik meg.
 4. **#Person holder**: Ez a Person típusú változó tárolja, hogy ki a tárgy birtokosa. Ha a tárgy a földön van egy szobában, akkor értéke nullpointer.
- **Metódusok**
 - **+Item(Room location, Person holder)**: Az osztály konstruktora. A kapott location és holder paraméterekkel beállítja a tartózkodási helyét és birtokosát.
 - **+void activate()**: Az adott tárgy aktiválása.

- **+void meet(Person person)**: Ha a tárgy egy szobában van a földön, akkor új személy belépése esetén a szoba értesíti a tárgyat a találkozásról, viselkedését ebben definiálja.
- **+boolean saveFromDeath(Person killer)**: Tárgy megkérésé, hogy védje meg birtokosát a kibukás ellen. A metódus visszatérési értéke igaz, ha a tárgy a kibukással szemben képes védelmet biztosítani és hamis, ha nem.
- **+boolean saveFromGas()**: Tárgy megkérésé, hogy védje meg birtokosát mérgező gáz ellen. A metódus visszatérési értéke igaz, ha a tárgy a mérgező gázzal védelmet biztosít és hamis, ha nem.
- **+void setLocation(Room location, Person holder)**: Átállítja a tárgy tartózkodási helyét és birtokosát a paraméterként kapottakra.
- **+Room getLocation()**: A tárgy tartózkodási helyének lekérdezése. A location változó értékével tér vissza.
- **+Person getHolder()**: A tárgy birtokosának lekérdezése. A holder változó értékével tér vissza.

7.1.10 ItemHandler

- **Felelősség**

Az interfész megvalósítása lehetővé teszi, hogy egy tárgyakkal rendelkezni képes objektum új tárgyhoz férjen hozzá, vagy elhasznált tárgyat töröljön ki a tárhelyéből.

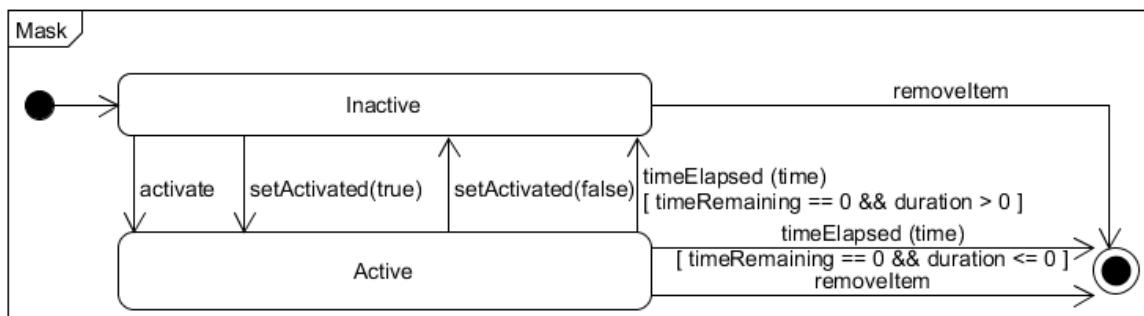
- **Metódusok**

- **+void addItem(Item item)**: Az item tárgy hozzáadása az objektum tárgynyilvántartásába.
- **+void removeItem(Item item)**: Az item tárgy eltávolításából az objektum tárgynyilvántartásából.

7.1.11 Mask

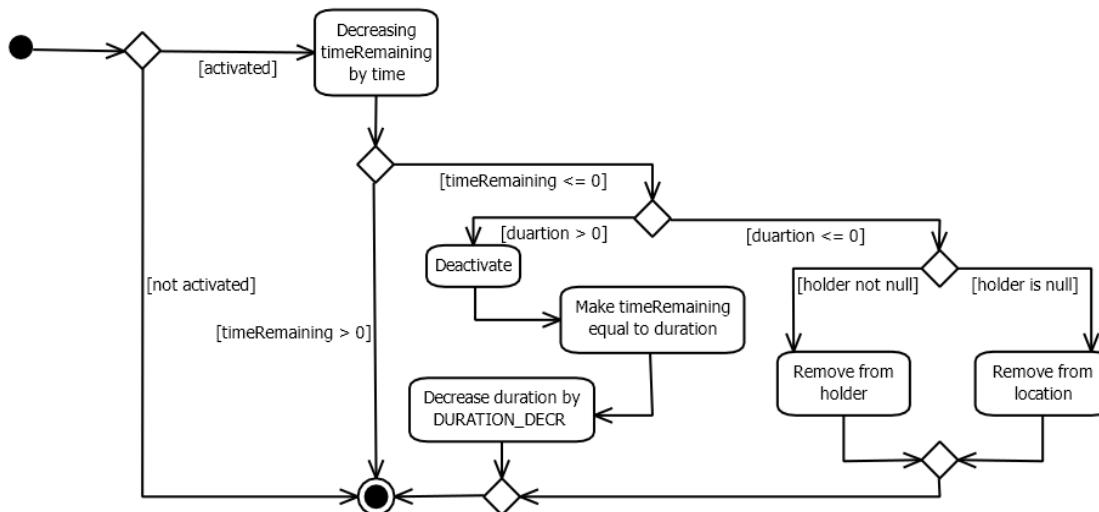
- **Felelősség**

Az FFP2-es maszkok adott ideig védelmet nyújtanak egy személynek a mérgező gázzal szemben. A hallgató képes manuálisan is aktiválni, de egyébként automatikusan is aktiválódik a személyeknek. A maszk saját felelőssége, hogy amennyiben véget ér a hatása vajon újra használható-e vagy meg kell semmisíteni. Amikor elmúlik a hatása a megszűnésétől kezdeményezi a birtokosánál.



2. állapotgép: A Mask osztály állapotgépe

- **Ősosztályok**
Item → IntervalItem
- **Attribútumok**
 - **-int duration:** Ez az egész szám változó tárolja a maszk tartósságát, vagyis, hogy a maszk a következő aktiválás után mennyi ideig lesz aktív.
 - **-int DURATION_DECR:** Statikus végleges (konstans) egész szám. Értéke 2. Amikor lejár a maszk aktív időtartama, de még nem szűnik meg, ezzel az értékkel csökken a következő használati idejének időtartama.
- **Metódusok**
 - **+Mask(Room location, Person holder, boolean activated, int timeRemaining):** Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az ősének a konstruktort, ezzel beállítva tagváltozónak értékét. A duration változójának értéke DURATION_DECR értékével lesz kisebb, mint a timeRemaining.
 - **+void meet(Person person):** Egy személlyel való találkozás kezelése. A metódus üres, mert minden tárgy csak akkor találkozik személlyel, ha földön van, és a maszknak ekkor nincs teendője.
 - **+boolean saveFromDeath(Person killer):** Kibukás elleni védelem kezelése. Mivel a maszk nem véd a kibukástól, a metódus logikai hamissal tér vissza minden esetben, és nem történik más.
 - **+boolean saveFromGas():** Mérgező gáz elleni védelem kezelése. Amennyiben aktiválva van a maszk, logikai igazzal tér vissza, megvéve a birtokosát. Ha nincs aktiválva, akkor aktiválja magát, és szintén logikai igazzal tér vissza.
 - **+void timeElapsed(int time):** Idő telésének kezelése a maszon. Ha a tárgy nincs aktiválva, nem történik semmi. Ha aktiválva van a tárgy, akkor a timeRemaining értékét csökkenti time-mal. Ekkor, ha a timeRemaining értéke kisebb vagy egyenlő, mint nulla, de a duration még pozitív a tárgy deaktiválja magát, beállítja a timeRemaining-jének értékét a durationre, majd csökkenti a duration értékét DURATION_DECR-vel. Ha nem csak a timeRemaining értéke lett kisebb vagy egyenlő, mint nulla, hanem a duration értéke is, akkor kezdeményezi aktuális birtokosánál a tárgy megsemmisítését. Ha a tárgynak a holder-e nullpointer, akkor a birtokosa a location-je, egyéb esetben a holder-je a birtokos.



2. aktivitás diagram: Mask timeElapsed(int time) metódusa

- **+int getDuration()**: A maszk tartósságának lekérdezése. A duration változó értékével tér vissza.
- **+void setDuration(int duration)**: A tartósság beállítása. Az objektum átállítja a duration változójának értékét a paraméterként kapotttra.

7.1.12 Person

- **Felelősség**

A játék karaktereinek működéséért felelős, főbb adatait és cselekvéseiiknek mechanizmusát tárolja. Tárolja a személy tartózkodási szobáját és tárgyait. A szereplők képesek találkozni, gyilkolni, elcsúszní (membénulni), és elkábólni. Képesek tárgyakat kezelni: felvenni, eldobni, eltávolítani. Tudnak mozogni a szobák között. A Person osztály absztrakt, csak a leszármazottai példányosíthatók.

- **Interfészek**

- ItemHandler
- TimeSensitive

- **Attribútumok**

- **#Room location**: A személy tartózkodási helyét tároló Room típus.
- **#List<Item> itemsInHand**: [0..ITEMSINHANDLIMIT] db Item tárolására alkalmas referencia. A személy rendelkezésére álló tárgyakat tárolja, melyeket képes aktiválni, használni, eldobni és eltávolítani. Létrehozáskor a tároló üres. A tárolóba a tárgyak sorrendje a felvételük idejének sorrendjével egyezik meg.
- **#int ITEMSINHANDLIMIT**: Statikus végeleges (konstans) egész szám. Értéke 5. Egy személy kezében maximum elférő tárgyak számát tárolja.
- **#int stunRemaining**: Ez az egész szám változó tárolja, hogy amennyiben a személy el van kábítva, még mennyi ideig nem mozoghat és nem vehet fel tárgyakat. Ha az értéke nulla, azt jelenti, hogy nincs elkábítva.
- **#int STUNSTART**: Statikus végeleges (konstans) egész szám. Értéke 3. Amikor a személyt elkábítják, ennek az értékére állítja a stunRemaining értékét.
- **#Random random**: Véletlenszám előállítására alkalmas statikus változó. A sörösvég miatti tárgyeldobásnál van szerepe, seed-je állítható.

- **Metódusok**

- **+Person(int stunRemaining, Room location)**: Az osztály konstruktora. A megfelelő változók értékét beállítja a hozzájuk tartozó paraméterekre.
- **+void setLocation(Room room)**: A személy tartózkodási helyének beállítása. A location értékét a paraméterre állítja be.
- **+void initItem(Item item)**: A személy kezébe adja a megadott tárgyat amennyiben az elfér nála. Inicializáláskor használandó. A tárgy a tárolójában a jelenlegi utolsó elem lesz.
- **+void addItem(Item item)**: A paraméterben kapott item tárgy felvétele a személyhez, ha a személy nincs elkábólva és a kezében lévő tárgyak száma még kisebb mint a konstans limit. Ekkor még a location-jének a pickUpItem(item) metódusát hívja meg, kezdeményezve a tárgy felvételét. Ha ez igaz értékkal tér vissza akkor a szoba eltávolította magából a tárgyat. Ekkor a személy felveszi, és a tárgynak beállítja az új birtokosát, elhelyezkedését. A tárgy a személy tárolójában a jelenlegi utolsó elem lesz. Ha a pickUpItem hamissal tér vissza nem történik semmi.

- **+void removeItem(Item item)**: A paraméterként kapott item tárgy eltávolítása a személy kezéből.
- **+void dropItem(Item item)**: A paraméterben kapott item tárgy eldobása. A saját kezéből eltávolítja a tárgyat és hozzáadja a tartózkodási helyének tárgyaihoz.
- **+void dropRandomItem()**: Egy véletlenszerű tárgy eldobása. A person osztály random változójával kisorsol egy véletlenszerű egész számot, aminek az értéke 0 és itemsInHand.size()-1 közé eshet, a felső határ tehát, amit már nem sorsolhat ki, az a szám, hogy hány tárgy van a személynél. A személy a kisorsolt sorszámmal megegyező indexű tárgyat eldobja a kezéből dropItem hívással.
- **+void enterRoom(Room roomTo)**: A személy mozgását végrehajtó metódus. Ha a személy nincs elkábulva, továbbítja a jelenlegi szobájának az átlépés igényét a movePerson(this, roomTo) metódus hívásával. A továbbiakban a két szoba felelőssége, hogy a személyt beengedi-e, erről az előbbi metódus visszatérési értéke értesíti a személyt. Amennyiben a visszatérési érték logikai igaz, sikeresen átlépett a másik szobába, frissíti a tárgyainak tartózkodási helyét. Ha nem sikerült átlépni, nem csinál semmit.
- **+void greet(Person greeter)**: Kezeli, hogy a személy mit csinál, ha köszönnek neki.
- **+void kill(Person killer)**: Kezeli, hogy a személy mit csinál, ha valaki kibuktatná.
- **+void meet(Person person)**: Kezeli, hogy a személy mit csinál, ha találkozik valakivel
- **+void pickedUpSlideRule(SlideRule slideRule)**: Kezeli, hogy a személy reagálhasson a Logarléc felvételére.
- **+void slip()**: A táblatörlő rongy megbénítására történő reakciót hajtja végre.
- **+void stun()**: A személy mérgező gáz általi megbénulását hajtja végre. Lokálisan tárolja, hogy már megmenekült-e vagy sem. Sorban végigmegy a kezében levő tárgyakon, és mindegyiktől igényli a mérgező gáztól való védelmet a saveFromGas() metódusuk meghívásával. Ha az egyik tárgy igazzal tér vissza, tehát megmenti, akkor megjegyzi, hogy megmenekült és abbahagyja a tárgyak kérdezgetését. Ha a kérdezés véget ért és megmenekült, akkor nem történik semmi más. Amennyiben nem menekült meg a gáztól, akkor a stunRemaining értékét beállítja a STARTSTUN értékére, majd sorban végighaladva az összes tárgyán, kidobja őket a dropItem() hívásával.

```

saved = hamis
ciklus i=0-tól itemsInHand.size-ig 1-gyel
    ha (itemsInHand[i].saveFromGas()) akkor
        saved = igaz
        kilépés a ciklusból
    ha (nem saved) akkor
        stunRemaining = STUNSTART
        ciklus i=0-tól itemsInHand.size-ig 1-gyel
            dropitem(itemsInHand[i])
visszatérés

```

1. pszeudo-kód: Person stun() metódusa

- **+void timeElapsed(int time)**: Idő telésének szimulálása az emberen. Ha a személy el van kábulva, csökkenti a stunRemaining értékét time-mal. Ha így negatív értékű lett, akkor beállítja 0-ra, vagyis az elkábulás ideje lejárt. Ezután sorban továbbítja az eltelt időt (time) a nála lévő összes tárgynak, függetlenül attól, hogy el van-e kábulva vagy sem.

```

ha (stunRemaining > 0) akkor
    stunRemaining = stunRemaining - time
    ha(stunRemaining < 0) stunRemaining = 0
ciklus i=0-tól itemsInHand.size-ig 1-gyel
    itemsInHand[i].timeElapsed(time)
visszatérés

```

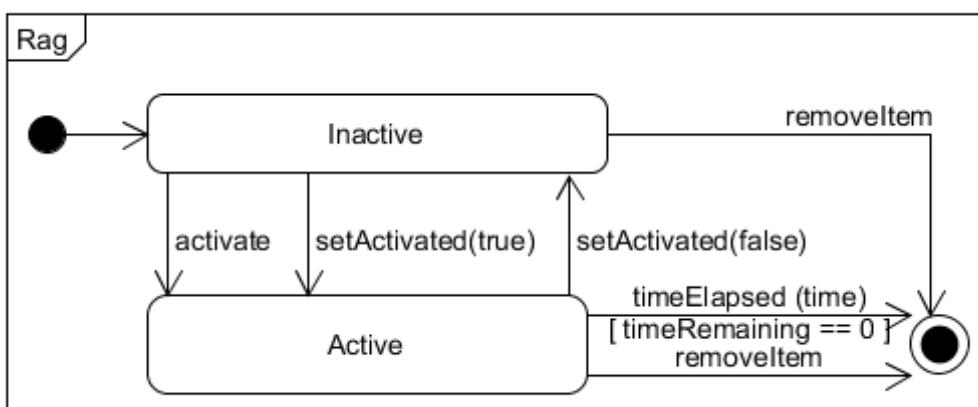
2. pszeudo-kód: Person timeElapsed(int time) metódusa

- **+int getStunRemaining()**: A kábultság hátralévő idejének lekérdezése. A stunRemaining változó értékével tér vissza.
- **+void setStunRemaining (int stunRemaining)**: A kábultság hátralévő idejének beállítása. Az objektum átállítja a stunRemaining változójának értékét a paraméterként kapotttra.
- **+Room getLocation()**: A személy tartózkodási helyének lekérdezése. A location változó értékével tér vissza.
- **+List<Item> getItemsInHand()**: A személynél lévő tárgyak lekérdezése. Az itemsInHand tárolóval tér vissza.
- **+void setSeed (long seed)**: A véletlenszám-generátor seed-jének beállítása. Az objektum átállítja a random seed-jének értékét a paraméterként kapotttra.

7.1.13 Rag

- **Felelősség**

A nedves táblatörölő rongy működéséért és viselkedéséért felel. Aktivált állapotban adott ideig megbénítja a vele egy helyiségen tartózkodó oktatókat. A hallgató képes aktiválni, ekkor a táblatörölő rongy felelőssége, hogy kezelje hátralévő használati idejét. Amikor elmúlik a hatása a megszűnésétől kezdeményezi a birtokosánál.

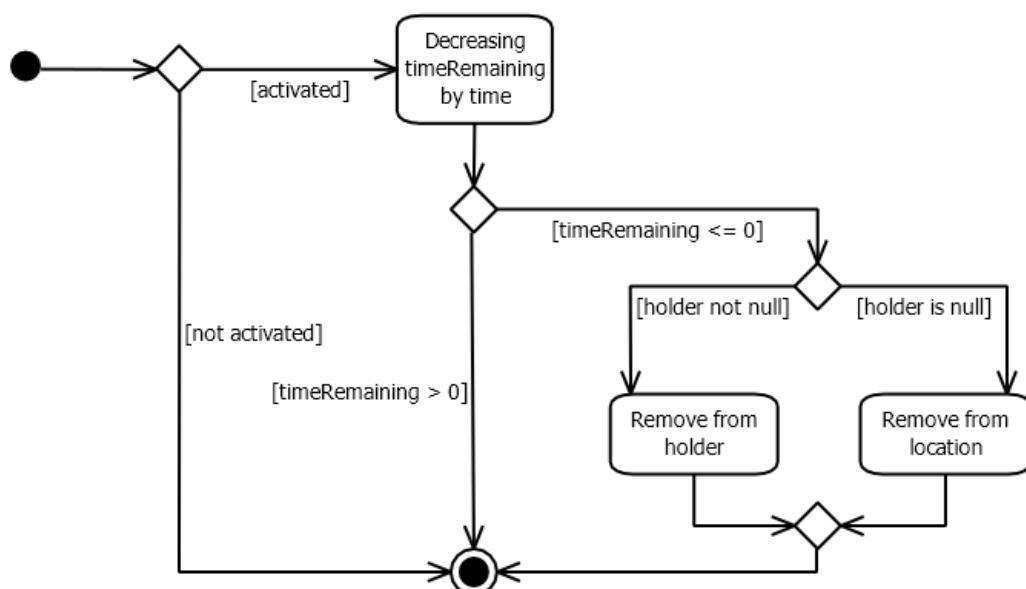


3. állapotgép: A Rag osztály állapotgépe

- **Ősosztályok**

Item → IntervalItem

- **Metódusok**
- **+Rag(Room location, Person holder, boolean activated, int timeRemaining)**: Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az ōsének a konstrukturát, ezzel beállítva tagváltozónak értékét.
- **+void meet(Person person)**: Egy személlyel való találkozás kezelése. Ha aktiválva van, akkor kezdeményezi a person személy megbénítását slip() hívásával, egyébként nem történik semmi.
- **+boolean saveFromDeath(Person killer)**: Kibuktatástól való megmentés kezelése. Amennyiben aktiválva van a rongy, kezdeményezi a killer személy megbénítását slip() hívásával, majd logikai igazzal tér vissza, megvédve a birtokosát. Egyéb esetben logikai hamissal tér vissza, nem nyújt védelmet.
- **+boolean saveFromGas()**: Mérgező gáz elleni védelem kezelése. Mivel a rongy nem nyújt védelmet a gáz ellen, a metódus logikai hamissal tér vissza minden esetben, és nem történik más.
- **+void timeElapsed(int time)**: Idő telésének kezelése a táblatörlő rongyon. Ha a tárgy nincs aktiválva, nem történik semmi. Ha aktiválva van a tárgy, akkor a timeRemaining értékét csökkenti time-mal. Ekkor ha a timeRemaining értéke kisebb vagy egyenlő mint nulla, akkor az objektum az aktuális birtokosánál kezdeményezi a tárgy megsemmisítését. Ha a tárgynak a holder-e nullpointer, akkor a birtokosa a location-je, egyéb esetben a holder-je a birtokos.

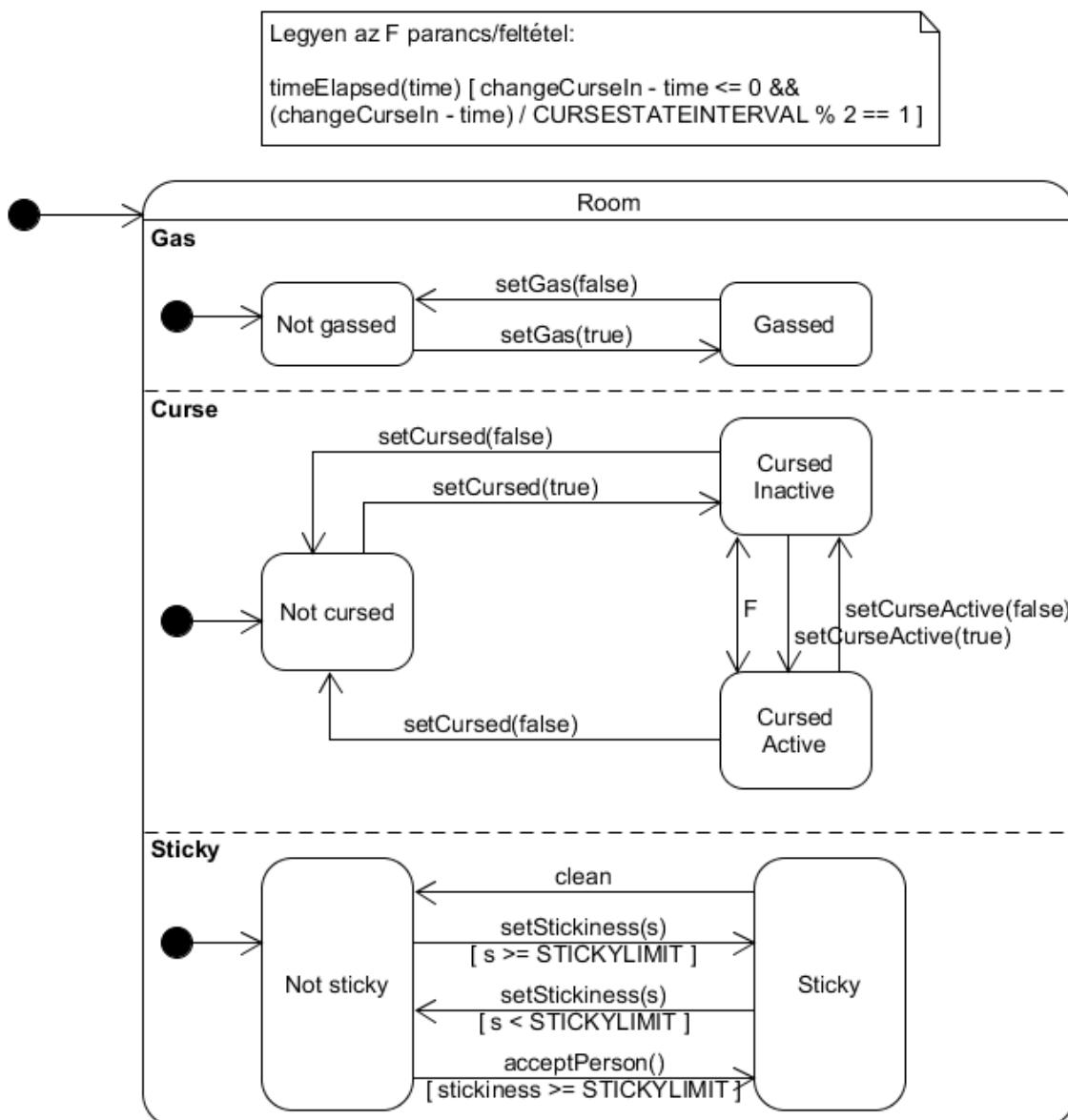


3. aktivitás diagram: Rag timeElapsed(int time) metódusa

7.1.14 Room

- Felelősség**

Minden szoba számon tartja a személyeket és tárgyakat, akik benne tartózkodnak. Ha egy személy belépne a szobába, a szoba felelőssége, hogy csak akkor engedje be, ha befér, valamint a szobák felelnek a személyek kiléptetéséért is. Amennyiben új személy érkezik a szobába, a szoba találkoztatja össze az új személyt a szobában levő tárgyakkal, illetve az új személyt az eddig szobában tartózkodó személyekkel. A szoba tudja, hogy melyik szobákkal szomszédos, így a benne lévő személyeknek lehetővé teszik a mozgást, amennyiben nem aktívan elátkozottak. Irányítja a saját viselkedését, kérésre az osztódást és egyesülést egy másik szobával.



4. állapotgép: A Room osztály állapotgépe

- **Interfészek**
 - ItemHandler
 - TimeSensitive
- **Attribútumok**
 - **-int capacity:** Ez az egész változó tárolja a szoba maximális befogadóképességét. Új személy szobába lépésekor ez alapján dönt arról, hogy beengedi-e vagy sem.
 - **-boolean gas:** Ez a logikai változó tárolja, hogy a szoba mérgező-e vagy sem. Új ember szobába lépésekor ennek a lekérdezésével tudja a szoba, hogy elkábítja-e a személyt vagy sem. Inicializáláskor, osztódáskor, légfrissítő és camembert hatására állítható.
 - **-boolean cursed:** Ez a logikai változó tárolja, hogy a szoba elátkozott-e vagy sem. Ennek lekérdezésével tudja, hogy az ajtajainak el(ő)tűnését kell-e módosítani.
 - **-boolean curseActive:** Amennyiben egy szoba elátkozott, ennek a logikai változónak segítségével tudja, hogy jelenleg az ajtajai használhatók-e vagy sem.
 - **-int changeCurseIn:** Amennyiben egy szoba elátkozott, ez a változó tárolja, hogy mennyi idő múlva kell módosítania a curseActive értékén. Az idő telésével együtt változik.
 - **-int CURSESTATEINTERVAL:** Statikus végleges (konstans) egész szám. Értéke 5. Ez a változó tárolja, hogy az elátkozott szobák mennyi időnként változtatják a curseActive értékét, tehát megadja, hogy az ajtók mennyi időnként tűnnek el és jelennek meg.
 - **-int stickiness:** Ez az egész változó tárolja a szoba ragacsosságának állapotát. Új személy szobába lépésekor növekszik, takarításkor nullázódik. Ez alapján dönt a szoba arról, hogy a tárgyak felvehetők-e vagy sem.
 - **-int STICKYLIMIT:** Statikus végleges (konstans) egész szám. Értéke 5. Ez a változó tárolja, hogy a ragacsosság mikortól érvényesül. Ha eléri ennek az értékét, a tárgyak nem vehetők fel.
 - **-List<Person> peopleInRoom:** [0..capacity] db Person tárolására alkalmas referencia. A szobában található személyeket tárolja, a személyek interakcióihoz, mozgatásához és az időtelés továbbításához szükséges. Létrehozáskor a tároló üres. A tárolóba a személyek sorrendje a tárolóba kerülés idejének sorrendjével egyezik meg.
 - **-List<Item> itemsInRoom:** [0..*] db Item tárolására alkalmas referencia. A szobában található tárgyakat tárolja, melyek találkozhatnak személyekkel így kifejtve hatásukat, illetve ezeket a személyek felvehetik. Létrehozáskor a tároló üres. A tárolóba a tárgyak sorrendje a tárolóba kerülés idejének sorrendjével egyezik meg.
 - **-List<Room> neighbours:** Room-ok tárolására alkalmas referencia. Az ajtók megvalósítása: ha az adott szobából át lehet menni egy másik szobába, akkor a másik szoba megtalálható ebben a tárolóban. Ha fordítva nem igaz egyirányú ajtóról beszélünk. A személyek ezt használva tudnak közlekedni. Létrehozáskor a tároló üres. Az inicializáláskor a szobák összefüggő hálózatot fognak képezni. A játék során minden minden szobának lesz legalább egy szomszédja és az összefüggőség megmarad. A tárolóba a szomszédok sorrendje a tárolóhoz adás idejének sorrendjével egyezik meg.

- **Metódusok**
- **+Room(int capacity, boolean gas, boolean cursed, int stickiness):** Az osztály konstruktora, létrehoz és inicializál egy szoba objektumot. A megfelelő nevű változónak értékét megfelelteti a megegyező nevű paraméterek értékével. A curseActive értékét hamisra állítja, a changeCurseIn értékét a CURSESTATEINTERVAL-lal teszi egyenlővé.
- **+void addNeighbour(Room room):** Új szoba felvétele szomszédok közé inicializáláshoz. Hozzáadja a room-ot a neighbours tárolóhoz.
- **+void addPerson(Person person):** Új személy hozzáadása a szobához inicializáláskor. A peopleInRoom tárolóhoz hozzáadja a person személyt, amennyiben a tárolóban lévő személyek száma még nem érte el a capacity értékét.
- **+removePerson(Person person):** A paraméterként kapott személy eltávolítása a személyek tárolójából.
- **+void initItem(Item item):** Tárgy elhelyezése a szobába inicializáláskor. Hozzáadja a tárgyak listájához a paraméterben kapott item-et, de az item tulajdonságainak beállításáért nem felel.
- **+void addItem(Item item):** A paraméterben kapott item tárgy szobához adása és a tárgy tartózkodási helyének beállítása setLocation(this, null)-al.
- **+boolean pickUpItem(Item item):** Egy tárgy felvételének kezdeményezése a szobánál. Amennyiben a stickiness még nem érte el a STICKYLIMIT értékét, eltávolítja az item tárgyat a tárolójából, majd igazzal visszatér. Egyébként hamis visszatérési értékkel jelzi a személynek a sikertelen felvételt.
- **+void removeItem(Item item):** Tárgy törlése a szobából. Eltávolítja item-et a tárgyak tárolójából.
- **+void timeElapsed(int time):** Idő telésének szimulálása. Amennyiben a szoba elátkozott, itt kezeli ajtajai el(ő)tűnését. Ha a szoba elátkozott csökkenti a changeCurseIn értékét time-mal, és ezután addig amíg a changeCurseIn nem pozitív, megváltoztatja a curseActive értékét és hozzáad a changeCurseIn-hez CURSESTATEINTERVAL-t. Függetlenül az elátkozottságtól, felel az idő telésének továbbításáért a benne lévő személyeknek és tárgyaknak. Először sorban végigmegy az összes szobában található tárgyon és továbbítja az idő telését, majd sorban végigmegy a személyek tárolóján is továbbítva az idő telését. Ezután még egyszer végigmegy sorban az összes személyen, aki a szobában van és személyenként végigmegy az összes szobában található tárgyon, összetálkoztatva az adott tárgyat az adott személlyel, a tárgy meet(személy) metódusát hívva. Végül a személyeket is összetálkoztatja egymással a következőképpen: Sorban végighalad az összes emberen, aki a szobában van, és emberenként végighalad minden emberen, aki a személyek tárolójában mögötte helyezkedik el. Ezekkel az emberekkel fog találkozni, tehát az első ciklusban végigiterált ember metódusát hívjuk meg, a belső ciklusban végigiterált személyt, mint paramétert átadva.

```

ha (cursed) akkor
    changeCurseIn = changeCurseIn - time
    ciklus amíg changeCurseIn <= 0
        curseActive = !curseActive
        changeCurseIn = changeCurseIn + CURSESTATEINTERVAL
    ciklus i = 0-tól itemsInRoom.size-ig 1-gyel
        itemsInRoom[i].timeElapsed(time)
    ciklus i = 0-tól peopleInRoom.size-ig 1-gyel
        peopleInRoom[i].timeElapsed(time)
    ciklus i = 0-tól peopleInRoom.size-ig 1-gyel
        ciklus j = 0-tól itemsInRoom.size-ig 1-gyel
            itemsInRoom[j].meet(peopleInRoom[i])
    ciklus i = 0-tól peopleInRoom.size-ig 1-gyel
        ciklus j = i+1-től peopleInRoom.size-ig 1-gyel
            peopleInRoom[i].meet(peopleInRoom[j])
visszatérés

```

4. pszeudo-kód: *Room timeElapsed(int time)* metódusa

- **+bool acceptPerson(Person person):** A paraméterként kapott személyt szobába engedése. Amennyiben a szobában tartózkodó emberek száma megegyezik a kapacitással, nem engedi be a személyt, hamissal tér vissza. Ha beengedi a személyt, először is frissíti a személy tartózkodási helyét, hozzáadja a személyt a saját tárolójához és a ragacsosság értékét növeli eggyel. Ha a szoba gázos állapotú, akkor elkábítja a személyt a stun() hívásával. Ezután a személy összetalálkoztatja az összes földön levő tárggyal, ami a szobában van. A tárgyak tárolójában sorban végighalad a tárgyakon és meghívja rájuk a meet(person) metódust. Ezután a személyekkel is találkozik. A személyek tárolójában sorban végighalad az embereken és mindegyiknél meghívja a person-nek a meet metódusát paraméterként átadva az aktuális személyt, aki vel találkozik. A visszatérési értéke a beengedés sikeresége, tehát a végén igazzal tér vissza.

```

ha (peopleInRoom.size = capacity) akkor
    visszatérés hamissal
    person.setLocation(önmaga)
    peopleInRoom.add(person)
    stickiness = stickiness + 1
    ha(gas) akkor
        person.stun()
    ciklus i = 0-tól itemsInRoom.size-ig 1-gyel
        itemsInRoom[i].meet(person)
    ciklus i = 0-tól peopleInRoom.size-ig 1-gyel
        ha (peopleInRoom[i] != person) akkor
            person.meet(peopleInRoom[i])
    visszatérés igazzal

```

5. pszeudo-kód: *Room acceptPerson(Person person)* metódusa

- **+bool movePerson(Person person, Room roomTo):** Személy továbbküldése egy másik szobába. Ha a szoba jelenleg aktívan el van átkozva, nem engedi tovább a személyt és hamissal tér vissza. Egyébként pedig a paraméterként kapott személy mozgási igényét továbbítja a paraméterként kapott szobának az acceptPerson() metódus meghívásával. Ha ez a metódus igazzal tér vissza, akkor eltávolítja a személyt a tárolójából. Ha a szoba nincs aktívan elátkozva, a metódus visszatérési értéke az acceptPerson() metódus visszatérési értékével egyezik meg.

```

ha (curseActive) akkor
    visszatérés hamissal
moveSucceed = roomTo.acceptPerson(person)
ha (moveSucceed) akkor
    peopleInRoom.remove(person)
visszatérés moveSucceed-del

```

6. pszeudo-kód: Room movePerson(Person person, Room roomTo) metódusa

- **+void clean():** A szoba takarítása. Stickiness értékének nullázása. A legutóbb érkezett ember (ez a takarító, aki a szobába jövetelkor hívta a függvényt) kivételével összes szobában tartózkodó embert átteszi egy másik szobába, amennyiben lehet. A szomszédok listájában előlről indul, és szomszédonként végigmegy a szobában levő embereken az utolsó személyt kivéve. Ekkor meghívja az aktuális ember enterRoom() metódusát az aktuális szomszédot paraméterként átadva.

```

stickiness = 0
ciklus i = 0-tól neighbours.size-ig eggyel
    ciklus j = 0-tól peopleInRoom.size()-1-ig 1-gyel
        peopleInRoom[j].enterRoom(neighbours[i])
visszatérés

```

7. pszeudo-kód: Room clean() metódusa

- **+Room split():** Ezzel a függvénytel kerül indítványozásra a meghívott szobánál, hogy kettéosztódjon. Amennyiben tartózkodik benne személy NULL a visszatérési értéke. Különben pedig létrehoz egy új szobát és az inicializálása után azzal tér vissza. Az új szoba kapacitása és stickinessse megegyezik az eredeti szobáéval. Ha az eredeti szoba gázos és elátkozott is, akkor az új szoba csak gázos lesz a cursed értéke hamis, az eredeti szoba pedig csak elátkozott marad, a gas értéke hamis lesz. Ha az eredeti szoba nem rendelkezett minden előbb említett tulajdonsággal, akkor az új szoba egyikkel sem fog, a gas és a cursed is hamis lesz. Az új szoba létrehozása után a tárgyak és szomszédok megosztása is megtörténik. Az eredeti szoba tárolói tartalmának első fele átkerül az új szoba tárolóiba, a saját szobából eltávolítva és az újhoz hozzáadva. Ha páratlan számú tárgy vagy szomszéd van összesen, akkor az új szoba fog eggyel többel rendelkezni. A szomszédok átadásakor figyelni kell azt is, hogy amennyiben a két szoba közötti ajtó kétirányú volt (tehát a szomszéd neighbours listájában szerepel az eredeti szoba), akkor onnan az eredeti szobát el kell távolítani, és az új szobát hozzá kell adni. Végül az eredeti szoba és az új szoba kölcsönösen hozzáadják egymást a saját szomszédaik listájához, és a metódus visszatér az új szobával.

```

ha (peopleInRoom.size != 0) akkor
    visszatérés null értékkel
newGas = hamis
newCursed = hamis
ha (gas és cursed) akkor
    gas = hamis
    newGas = igaz
newRoom = Room(capacity, newGas, newCursed, stickiness)
desiredItemSize = itemsInRoom.size / 2
ciklus amíg itemsInRoom.size > desiredItemSize
    newRoom.addItem(itemsInRoom[0])
    itemsInRoom.remove(0)
desiredNeighbourSize = neighbours.size / 2
ciklus amíg neighbours.size > desiredNeighbourSize
    neighbour = neighbours[0]
    ha (neighbour.neighbours.contains(önmaga)) akkor
        neighbour.neighbours.remove(önmaga)
        neighbour.neighbours.add(newRoom)
    newRoom.neighbours.add(neighbour)
    neighbours.remove(0)
neighbours.add(newRoom)
newRoom.neighbours.add(önmaga)
visszatérés newRoom-mal

```

8. pszeudo-kód: Room split() metódusa

- **+Room requestMerge(Room room2):** Ezzel a függvénytel kerül indítványozásra a meghívott szobánál, hogy egyesüljön a paraméterben átadott szobával. Ha a szobában van legalább egy személy, nullpointerrel tér vissza. Egyébként pedig meghívja a paraméterként kapott szobának a merge() függvényét, ahol magát adja át paraméterként. Ekkor a merge() visszatérési értékével tér vissza. Amennyiben viszont a merge() visszatérési értéke nem nullpointer a visszatérés előtt még inicializálja az merge() visszatérési értékeként kapott új szobát a saját tárgyai és szomszédai hozzáadásával. Először végigmegy a tárgyain és mindegyiket hozzáadja az új szobához, majd sorban végigmegy az összes szomszédján, és amennyiben nem a másik egyesülő szoba az aktuális szomszéd, akit vizsgál, két dolgot csinál. Egyszerű hozzáadja az új szoba szomszédjaihoz, másrészről megnézi, hogy a szomszédnak a neighbours-ei között megtalálja-e magát, és ha igen, akkor magát kitörli onnan, és hozzáadja az új szobát.

```

ha (peopleInRoom.size != 0) akkor
    visszatérés null értékkel
newRoom = room2.merge(önmaga)
ha (newRoom != null)
    ciklus i= 0-tól itemsInRoom.size-ig
        newRoom.addItem(itemsInRoom[i])
    ciklus amíg neighbours.size != 0
        neighbour = neighbours[0]
        ha (neighbour != room2) akkor
            ha (neighbour.neighbours.contains(önmaga)) akkor
                neighbour.neighbours.remove(önmaga)
                neighbour.neighbours.add(newRoom)
            newRoom.neighbours.add(neighbour)
            neighbours.remove(0)
    visszatérés newRoom-mal

```

9. pszeudo-kód: Room requestMerge(Room room2) metódusa

- **-Room merge(Room room1):** Két szoba egyesítését elvégző függvény. Egy olyan szoba hívhatja egy másik szobának ezt a függvényét, amelyik már beleegyezett az egyesülésbe. A másik szoba nem egyezik bele, ha található benne legalább egy személy, ekkor nullpointer-rel tér vissza. Különben pedig létrehozza az új szobát, mely az egyesülésükből keletkezik. Az új szoba cursed és gas értékei a két szoba értékének logikai VAGY kapcsolatából keletkezik, míg az új capacity és stickiness a két szoba értékeiből a nagyobbik lesz. Ezután végigmegy a tárgyain és mindegyiket hozzáadja az új szobához, majd sorban végigmegy az összes szomszédján, és amennyiben nem a másik egyesülő szoba az aktuális szomszéd, akit vizsgál, két dolgot csinál. Egyrészt hozzáadja az új szoba szomszédjaihoz, másrészt megnézi, hogy a szomszédnak a neighbours-ei között megtalálja-e magát, és ha igen, akkor magát kitörli onnan, és hozzáadja az új szobát. Végül a létrejött új szobával visszatér.

```

ha (peopleInRoom.size != 0) akkor
    visszatérés null értékkel
newCapacity = max(room1.capacity, capacity)
newGas = room1.gas vagy gas
newCurse = room1.cursed vagy cursed
newStickiness = max(room1.stickiness, stickiness)
newRoom = Room(newCapacity, newGas, newCurse, newStickiness)
ciklus i= 0-tól itemsInRoom.size-ig
    newRoom.addItem(itemsInRoom[i])
ciklus amíg neighbours.size != 0
    neighbour = neighbours[0]
    ha (neighbour != room1) akkor
        ha (neighbour.neighbours.contains(önmaga)) akkor
            neighbour.neighbours.remove(önmaga)
            neighbour.neighbours.add(newRoom)
        newRoom.neighbours.add(neighbour)
        neighbours.remove(0)
    visszatérés newRoom-mal

```

10. pszeudo-kód: Room merge(Room room1) metódusa

- **+void setGas(boolean gas):** A szoba gázosítása vagy annak megszüntetése. Beállítja a gas értékét a paraméterben kapottra. Továbbá amennyiben a kapott paraméter igaz, a szobában tartózkodó személyeken sorban végighaladva meghívja rájuk a stun() metódust.
- **+boolean isGas():** A szoba gázos állapotának lekérdezése. A gas változó értékével tér vissza.
- **+boolean isCursed():** A szoba elátkozottságának lekérdezése. A cursed változó értékével tér vissza.
- **+void setCursed(boolean cursed):** Az elátkozottság beállítása. Az objektum átállítja a cursed változójának értékét a paraméterként kapottra.
- **+int getCapacity():** A szoba kapacitásának lekérdezése. A capacity változó értékével tér vissza.
- **+void setCapacity(int capacity):** A kapacitás beállítása. Az objektum átállítja a capacity változójának értékét a paraméterként kapottra.
- **+boolean isCurseActive():** A szoba aktív elátkozottságának lekérdezése. A curseActive változó értékével tér vissza.
- **+void setCurseActive(boolean curseActive):** Az aktív elátkozottság beállítása. Az objektum átállítja a curseActive változójának értékét a paraméterként kapottra.
- **+int getStickiness():** A szoba ragacsosságának lekérdezése. A stickiness változó értékével tér vissza.
- **+void setStickiness(int stickiness):** A ragacsosság beállítása. Az objektum átállítja a stickiness változójának értékét a paraméterként kapottra.
- **+List<Person> getPeopleInRoom():** A szobában lévő emberek lekérdezése. A peopleInRoom tárolóval tér vissza.
- **+List<Item> getItemsInRoom():** A szobában lévő tárgyak lekérdezése. Az itemsInRoom tárolóval tér vissza.
- **+List<Room> getNeighbours():** A szoba szomszédjainak lekérdezése. A neighbours tárolóval tér vissza.

7.1.15 SlideRule

- **Felelősség**

A Logarléc tárgy a hallgatók győzelmének kulcsa. Ha egy hallgató birtokolja, győznek a hallgatók. Felelőssége, hogy jelezze a személynek, hogy logarlécet vett fel, ezzel biztosítva, hogy tanulón kívül más ne vehesse fel.

- **Ősosztályok**

Item

- **Metódusok**

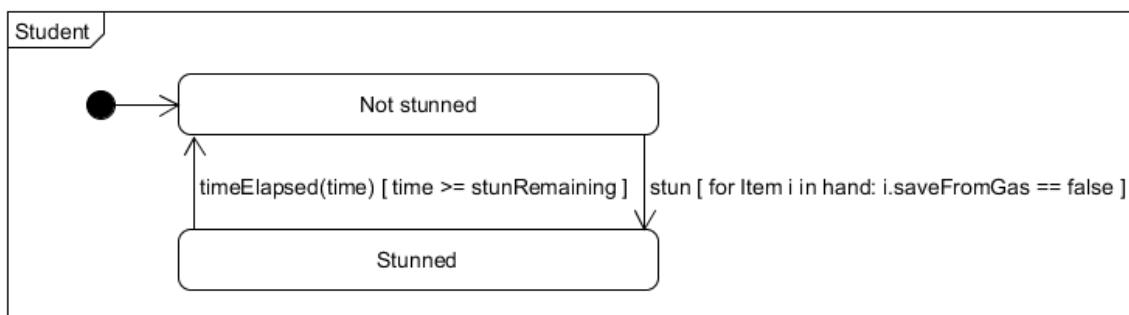
- **+SlideRule(Room location, Person holder):** Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az ősének a konstruktort, ezzel beállítva tagváltozóinak értékét.
- **+void setLocation(Room room, Person person):** A tárgy tartózkodási helyének beállítása. Felüldefiniálja a metódust, mert ekkor tudja értesíteni birtokosát, hogy logarlécet vett fel. Beállítja a paraméterként kapott értékeket a megfelelő változóiba. Ha a person értéke nem nullpointer, akkor meghívja a pickedUpSlideRule(this) metódust rá.
- **+void activate():** A Logarléc aktiválásakor nem történik semmi, üres metódus.

- **+void meet(Person person):** Egy személlyel való találkozás kezelése. A metódus üres, mert minden tárgy csak akkor találkozik személlyel, ha földön van, és a logarlécnek ekkor nincs teendője.
- **+boolean saveFromDeath(Person killer):** Kibukás elleni védelem kezelése. Mivel a logarléc nem véd a kibukástól, a metódus logikai hamissal tér vissza minden esetben, és nem történik más.
- **+bool saveFromGas():** Mérgező gáz elleni védelem kezelése. Mivel a logarléc nem nyújt védelmet a gáz ellen, a metódus logikai hamissal tér vissza minden esetben, és nem történik más.
- **+void timeElapsed(int time):** Idő telésének kezelése. Mivel a logarléccel nem történik semmi az idő műlásával, a metódus üres.

7.1.16 Student

- **Felelősség**

A hallgatókat reprezentáló osztály. A játékosok által elvégezhető funkciókat valósítja meg. Felelőssége, hogy a hallgató mozoghasson, megfelelően használhassa a tárgyakat a védelem érdekében, és hogy akkor haljon meg vagy kábuljon el, amikor szükséges.



5. állapotgép: A Student osztály állapotgépe

- **Ősosztályok**
Person
- **Metódusok**
 - **+Student(int stunRemaining, Room location):** Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az ősének a konstruktorát, ezzel beállítva tagváltozóinak értékét.
 - **+void meet(Person person):** Egy személlyel való találkozást kezeli le. Köszön a személynek, meghívva rajta a greet(this) metódust.
 - **+void greet(Person greeter):** A hallgató köszönésre reagálása. Nem csinál semmit, üres metódus.
 - **+void kill(Person killer):** A hallgató megtámadását kezeli le. Lokálisan tárolja, hogy már megmenekült-e vagy sem. Sorban végigmegy a kezében levő tárgyakon, és mindegyiktől igényli a kibukástól való védelmet a saveFromDeath(killer) metódusuk meghívásával. Ha az egyik tárgy igazzal tér vissza, tehát megmenti, akkor megjegyzi, hogy megmenekült és abbahagyja a tárgyak kérdezgetését. Ha a kérdezés véget ért és megmenekült, akkor nem történik semmi más. Amennyiben nem menekült meg a kibukástól, eltávolítja magát a tartózkodási szobájából.

```

saved = false
ciklus i=0-tól itemsInHand.size-ig 1-gyel
    ha (itemsInHand[i].saveFromDeath(killer)) akkor
        saved = igaz
        kilépés a ciklusból
    ha (nem saved) akkor
        loaction.removePerson(önmaga)
    visszatérés

```

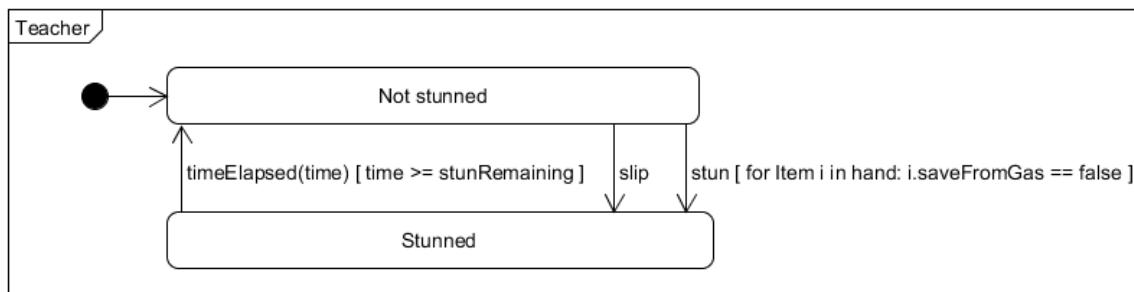
11. pszeudo-kód: Student kill(Person killer) metódusa

- **+void slip():** Rongy miatti elkábulás. A táblatörölő rongy a hallgatókat nem veszélyezteti, ezért ez egy üres metódus.
- **+void pickedUpSlideRule(SlideRule slideRule):** Logarléc felvétele. A hallgatók megtarthatják a logarlécet, nem történik semmi.
- **void activateItem(Item item):** A paraméterben kapott item tárgy aktiválása.

7.1.17 Teacher

- **Felelősség**

Az oktatókat reprezentáló osztály. Felelőssége, hogy az oktatók is mozogjanak, és képes legyen fenyegetni a hallgatókat találkozásukkor. A személyek alapvető képességein túl a lebénulásra és gyilkolásra van lehetősége. Nem lehet nála Logarléc.



6. állapotgép: A Teacher osztály állapotgépe

- **Ősosztályok**
Person
- **Metódusok**
 - **+Teacher(int stunRemaining, Room location):** Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az ősének a konstrukturát, ezzel beállítva tagváltozóinak értékét.
 - **+void meet(Person person):** Egy személlyel való találkozást kezeli le. Ha a tanár nincs elkábulva, kezdeményezi a person kibuktatását a kill(this) metódust meghívva.
 - **+void greet(Person greeter):** Az oktató köszönésre reagálása. Ha a tanár nincs elkábulva, kezdeményezi a greeter kibuktatását a kill(this) metódust meghívva.
 - **+void kill(Person killer):** A tanár megtámadását kezeli le. Mivel a tanárok nem bukhatnak ki, a metódus üres.
 - **+void slip():** Rongy miatti elkábulás. A stunRemaining értékét beállítja STUNSTART-ra, elkábítva az oktatót.

- **+void pickedUpSlideRule(SlideRule slideRule):** Logarléc felvétele. Az oktatók nem vehetik fel a Logarlécet, ezért kidobják a paraméterként kapott slideRule-t a kezüköből a szobába.

7.1.18 TimeSensitive

- **Felelősség**

Ezen az interfészen keresztül értesülnek a játék objektumai az idő teléséről.

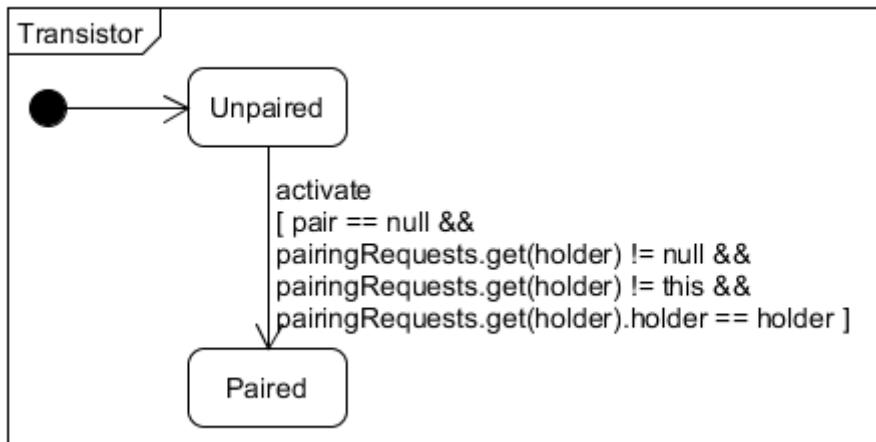
- **Metódusok**

- **void timeElapsed(int time):** Ezen keresztül értesülnek az interfészt megvalósító objektumok azzal, hogy a játékban time idő eltelt, valamint kezelhetik az ennek megfelelő állapotváltozásaikat, mechanizmusaiat.

7.1.19 Transistor

- **Felelősség**

A tranzisztorok lehetőséget nyújtanak a hallgatóknak nem szomszédos szobák közti gyorsutazásra. Ahhoz, hogy egy hallgató ezzel élhessen, két tranzisztorra is szüksége van. A tranzisztorok felelőssége, hogy össze tudjanak kapcsolódni egymással, amennyiben minden kettő a hallgatónál van. Az összekapcsolt tranzisztorok számítartják egymást is, így biztosítva a kapcsolatot a két tárgy tartózkodási szobái között. A tranzisztorok örökifjűk, korlátlanul használhatók a párjukkal. Ha két tranzisztor egyszer összekapcsolódott, már nem tudnak szétkapcsolódni. Kettőnél több tranzisztor nem csatlakozhat össze egymással.



7. állapotgép: A Transistor osztály állapotgépe

- **Ősosztályok**

Item

- **Attribútumok**

- **-Transistor pair:** Ez a Transistor referencia tárolja a tranzisztor párját. Ha nincs összekapcsolva, az értéke nullpointer.
- **-HashMap<Person, Transistor> pairingRequests:** Statikus végleges (konstans) kétdimenziós tároló. Amennyiben egy személy aktivál egy tranzisztor, aminek még nincs párja, és a kezében ez az első ilyen tranzisztor amit aktivált, az ide kerül be az adott személlyel. A második tranzisztor kapcsolódásakor ezt használva párosítja össze őket. minden személy párosítási igényét tárolja. minden személy csak egyszer szerepelhet benne.

- **Metódusok**

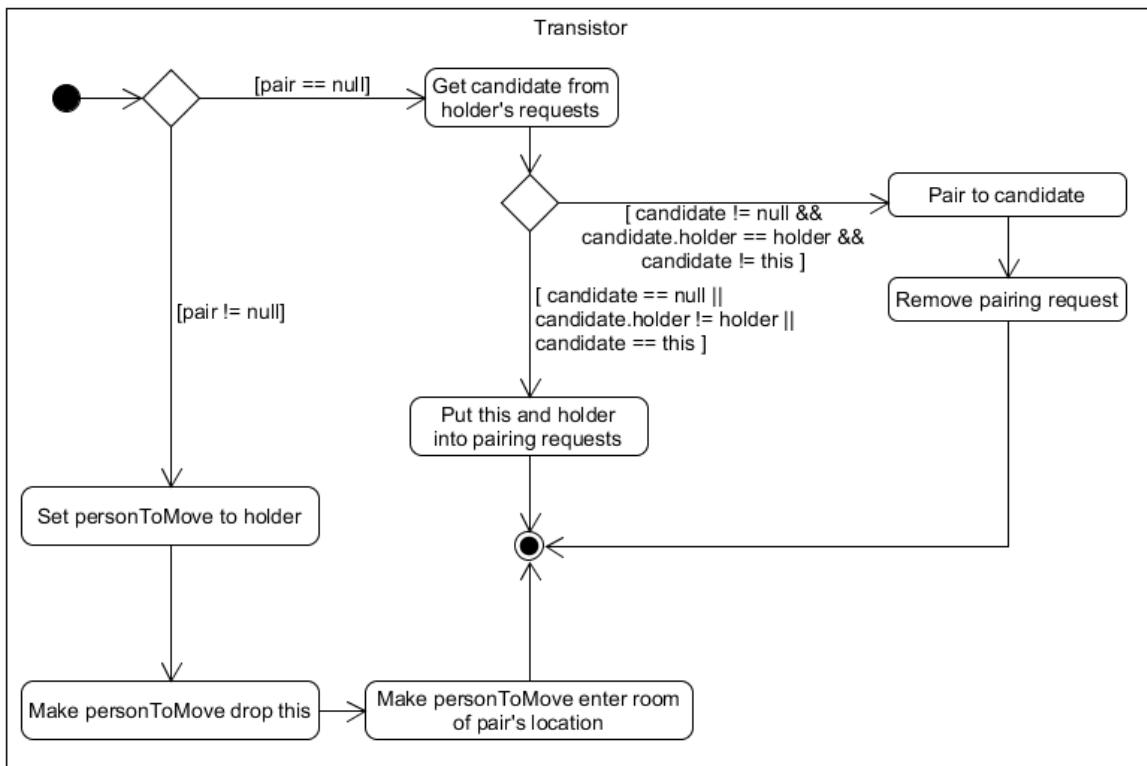
- **+Transistor(Room location, Person holder):** Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az ősének a konstruktörét, ezzel beállítva tagváltozóinak értékét.
- **+void setPair(Transistor transistor):** Tranzisztor egyirányú párosítása. Beállítja a pair-t a paraméterben kapott értékre.
- **+Transistor getPair():** Tranzisztor pájának lekérdezése. A pair-rel tér vissza.
- **+void activate():** Két tranzisztor összekapcsolását illetve azok használatát, a teleportálást is lehetővé tevő metódus. Ha a pair értéke nem nullpointer, tehát már párosítva van a tranzisztor, akkor a birtokosán meghívja a dropItem(this) metódust ezzel kidobatva a magát, és kezeményezi a birtokosának átlépését abba a szobába, ahol a párja van. Ha viszont a pair értéke nullpointer a következők történnek: Megnézi, hogy a pairingRequests-ben a saját birtokosa már helyezett-e el tranzisztorát. Ha nem helyezett el, vagy helyezett, de az vagy már nem az ő kezében van (a másik tranzisztor holdere és a tranzisztor holdere nem ugyanaz) vagy megegyezik a jelenlegi tranzisztorral, akkor beletesszük a jelenlegi tranzisztorról és holder-ét a pairingRequests-be, ezzel frissítve az esetleges eddigi találatot, vagy hozzáadva újként. Ha viszont ezen feltételekből egyik sem volt igaz, akkor a találatunk megfelel a párosítási feltételeknek. Ekkor minden tranzisztornak beállítjuk egymást párként, és kiveszük a birtokos-t és a találatot a pairingRequests tárolóból.

```

ha (pair != null) akkor
    personToMove = holder
    personToMove.dropItem(önmaga)
    personToMove.enterRoom(pair.location)
különben
    candidate = pairingRequests.get(holder)
    ha (candidate == null vagy candidate.holder != holder vagy
        candidate == önmaga) akkor
        pairingRequests.put(holder, önmaga)
    különben
        pair = candidate
        candidate.pair = önmaga
        pairingRequests.remove(holder)
visszatérés

```

12. pszeudo-kód: Transistor activate() metódusa



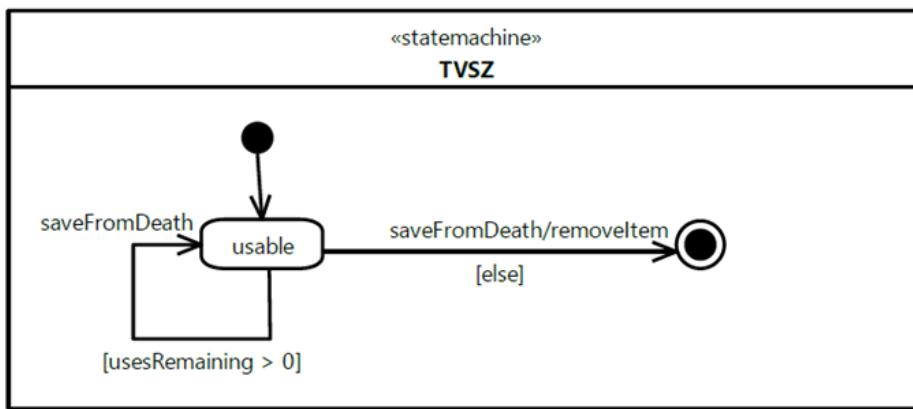
4. aktivitás diagram: Transistor activate() metódusa

- **+void meet(Person person):** Egy személlyel való találkozás kezelése. A metódus üres, mert minden tárgy csak akkor találkozik személlyel, ha földön van, és a tranzisztorak ekkor nincs teendője.
- **+boolean saveFromDeath(Person killer):** Kibukás elleni védelem kezelése. Mivel a tranzisztor nem véd a kibukástól, a metódus logikai hamissal tér vissza minden esetben, és nem történik más.
- **+bool saveFromGas():** Mérgező gáz elleni védelem kezelése. Mivel a tranzisztor nem nyújt védelmet a gáz ellen, a metódus logikai hamissal tér vissza minden esetben, és nem történik más.
- **+void timeElapsed(int time):** Idő telésének kezelése. Mivel a tranzisztor nem tesz semmit az idő műlásával, a metódus üres.

7.1.20 TVSZ

- **Felelősség**

A TVSZ denevérbőrre írt példánya veszély esetén megvédi az oktatótól az őt hordozó hallgatót. Felelőssége, hogy veszély esetén, ha rá kerül a sor automatikusan aktiválódjon. Továbbá, ha többször nem használható kezdeményezi a megsemmisítését birtokosánál.



8. állapotgép: A TVSZ osztály állapotgépe

- **Ősosztályok**

Item

- **Attribútumok**

- **-int usesRemaining:** Ez az egész szám változó tárolja, hogy a tárgy még hány alkalommal tud védelmet biztosítani egy oktatóval szemben.

- **Metódusok**

- **+TVSZ(Room location, Person holder):** Az osztály konstruktora új objektum létrehozásához. A kapott paraméterekkel meghívja az ősének a konstrukturát, ezzel beállítva a megfelelő tagváltozónak értékét. A usesRemaining értékét 3-ra állítja.
- **+void activate():** A TVSZ manuális aktiválásakor nem történik semmi, üres metódus.
- **+void meet(Person person):** Egy személlyel való találkozás kezelése. A metódus üres, mert minden tárgy csak akkor találkozik személlyel, ha földön van, és a TVSZ-nek ekkor nincs teendője.
- **+boolean saveFromDeath(Person killer):** Kibukás elleni védelem kezelése. A TVSZ csökkenti a usesRemaining értékét eggyel. Amennyiben a változó értéke nulla lesz, akkor az objektum a birtokosánál kezdeményezi a tárgy eltávolítását. minden esetben igazzal tér vissza.
- **+bool saveFromGas():** Mérgező gáz elleni védelem kezelése. Mivel a TVSZ nem nyújt védelmet a gáz ellen, a metódus logikai hamissal tér vissza minden esetben, és nem történik más.
- **+void timeElapsed(int time):** Idő telésének kezelése. Mivel a TVSZ nem tesz semmit az idő műlásával, a metódus üres.
- **+int getUsesRemaining():** A maradék használatok számának lekérdezése. A usesRemaining változó értékével tér vissza.
- **+void setUsesRemaining(int usesRemaining):** A maradék használatok számának beállítása. Az objektum átállítja a usesRemaining változójának értékét a paraméterként kapottra.

7.1.21 Interpreter

- **Felelősség**

Egy interpretet valósít meg, amely lehetővé teszi a felhasználó számára, hogy parancsokkal vezérelje a modell elemeit. Fogadja a felhasználói bemenetet, majd értelmezi és feldolgozza a kapott parancsokat. Ezek alapján végrehajtja az objektumok módosításait, amihez kezeli az objektumokat és azok állapotait, tehát létrehozza, módosítja, eltávolítja őket. Valamint visszaadja a felhasználónak az értelmezett és végrehajtott parancsok eredményét.

- **Attribútumok**

- **-HashMap<String,Command> commands:** tárolja a parancsokat és azokhoz tartozó végrehajtó függvényeket (Command interfész implementációkat).
- **-HashMap<String,Object> objects:** az objektumokat tárolja a nevük alapján. Lehetővé teszi az objektumok kezelését és lekérdezését a nevük alapján.

- **Metódusok**

- **+Interpreter():** betölti a parancsokat a commands HashMap-be. A parancsok megegyeznek a prototípus interfész definíciójában meghatározottakkal, működésük az ott szereplő leírásuknak megfelelően történik.
- **+String getName(Object object):** Ez a metódus megkeresi az objektum nevét az objects HashMap-ben, és visszaadja azt.
- **+void printList(String name, Collection<?> list):** Ez a metódus kiírja a konzolra az első attribútumban megadott néven a második attribútumban megadott gyűjteményt, benne az eleminek nevével.
- **+void execute(String line):** Ez a metódus egy parancssor stringet kap bemenatként, majd kiválasztja és végrehajtja a megfelelő parancsot a commands HashMap-ból a parancssor első szava alapján.
- **+void main(String[] args):** elvégzi egy interpreter inicializálását majd a parancsok beolvasását a konzolról és továbbítja az interpreternek végrehajtásra.

7.2 A tesztek részletes tervei, leírásuk a teszt nyelvén

7.2.1 Átlépés oktatóhoz nem védő tárgyakkal 1

- **Leírás**

Két szomszédos szoba, az egyikbe egy hallgató, a másikba egy oktató létrehozása. A hallgatónak adni egy FalseTVSZ-t, egy AirFresher-t, egy Camambert-et, egy Transistor és egy Rag-et. A hallgató átléptetése a másik szobába.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A szobák közti mozgás (enterRoom, movePerson, acceptPerson), a kibuktatás (meet, greet, kill), a tárgyak kibuktatás elleni védelmények (saveFromDeath) és a játék végének (Interpreter - status) tesztelése.

- **Bemenet**

```

create room r1
create room r2
create student s
create teacher t
create falsetvsz ftv
create airfresher a
create camembert c
create transistor tr
create rag r
neighbour r1 r2
addperson s r1
addperson t r2
add ftv s
add a s
add c s
add tr s
add r s
enter r2 s
status

```

- **Elvárt kimenet**

Game over

7.2.2 Átlépés oktatóhoz nem védő tárgyakkal 2

- **Leírás**

Két szomszédos szoba, az egyikbe egy hallgató, a másikba egy oktató létrehozása. A hallgatónak adni egy aktiválatlan BeerGlass-t és egy Mask-ot. A hallgató átléptetése a másik szobába.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A szobák közti mozgás, a kibuktatás és tárgyak kibuktatás elleni védelmények (saveFromDeath) tesztelése.

- **Bemenet**

```

create room r1
create room r2
create student s
create teacher t
create beerglass b
create mask m
neighbour r1 r2
addperson s r1
addperson t r2
add b s
add m s
enter r2 s
status

```

- **Elvárt kimenet**

Game over

7.2.3 Átlépés oktatóhoz TVSZ-szel

- **Leírás**

Két szomszédos szoba, az egyikbe egy hallgató, a másikba egy oktató létrehozása. A hallgatónak adni egy TVSZ-t. A hallgató átléptetése a másik szobába.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A szobák közti mozgás, a kibuktatás (kill), a TVSZ kibuktatás elleni védelemnének (saveFromDeath) és a játék állapotának (Interpreter - status) tesztelése.

- **Bemenet**

```

create room r1
create room r2
create student s
create teacher t
create tvsz tv
neighbour r1 r2
addperson s r1
addperson t r2
add tv s
enter r2 s
status tv
status

```

- **Elvárt kimenet**

```

tv
location: r2
holder: s
usesRemaining: 2
Game in progress

```

7.2.4 Átlépés oktatóhoz BeerGlass-szal

- **Leírás**
Két szomszédos szoba, az egyikbe egy hallgató, a másikba egy oktató létrehozása. A hallgatónak adni egy BeerGlass-t. A BeerGlass aktiválása. A hallgató átléptetése a másik szobába.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
A szobák közti mozgás, a kibuktatás és BeerGlass kibuktatás elleni védelmények (saveFromDeath) tesztelése.
- **Bemenet**

```
create room r1
create room r2
create student s
create teacher t
create beerglass b
neighbour r1 r2
addperson s r1
addperson t r2
add b s
activate b s
enter r2 s
status
```
- **Elvárt kimenet**
Game in progress

7.2.5 Átlépés gázba nem védő tárgyakkal 1

- **Leírás**
Két szomszédos szoba, melyek közül az egyik gázos és a másikba egy hallgató létrehozása. A hallgatónak adni egy TVSZ-t, egy AirFresher-t, egy Camembert-t, egy Transistor-t és egy Rag-et. A hallgató átléptetése a másik szobába.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
A szobák közti mozgás (acceptPerson), elkábulás (stun), gáz elleni védelemnyújtás (saveFromGas), személy állapotának kiírása (Interpreter – status s)
- **Bemenet**

```
create room r1
create room r2
setroom r2 gas true
create student s
create tvsz tv
create airfresher a
create camembert c
create transistor tr
create rag r
neighbour r1 r2
addperson s r1
add tv s
add a s
add c s
add tr s
add r s
```

- **Elvárt kimenet**

```

s
location: r2
itemsInHand:
stunRemaining: 3

```

7.2.6 Átlépés gázba nem védő tárgyakkal 2

- **Leírás**
Két szomszédos szoba, melyek közül az egyik gázos és a másikba egy hallgató létrehozása. A hallgatónak adni egy BeerGlass-t és egy FalseMask-ot. A hallgató átléptetése a másik szobába.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
A szobák közti mozgás és a tárgyak gáz elleni védelmének (saveFromGas) tesztelése.
- **Bemenet**

```

create room r1
create room r2
setroom r2 gas true
create student s
create beerglass b
create falsemask fm
neighbour r1 r2
addperson s r1
add b s
add fm s
enter r2 s
status s

```

- **Elvárt kimenet**
- ```

s
location: r2
itemsInHand:
stunRemaining: 3

```

### 7.2.7 Átlépés gázba Mask-kal

- **Leírás**  
Két szomszédos szoba, melyek közül az egyik gázos és a másikba egy hallgató létrehozása. A hallgatónak adni egy Mask-ot. A hallgató átléptetése a másik szobába.
- **Ellenőrzött funkcionalitás, várható hibahelyek**  
A szobák közti mozgás, elkábulás (stun) és a Mask gáz elleni védelmének (saveFromDeath) tesztelése.
- **Bemenet**

```

create room r1
create room r2
setroom r2 gas true
create student s
create mask m
neighbour r1 r2
addperson s r1

```

```

 add m s
 enter r2 s
 status s
 • Elvárt kimenet
 s
 location: r2
 itemsInHand: m,
 stunRemaining: 0

```

### 7.2.8 Takarító átléptetése

- **Leírás**

Két szomszédos szoba létrehozása, melyekből az egyik gázos és bele egy hallgató, a másikba egy takarító létrehozása. A gázos szoba stickiness-ének beállítása. A takarító átléptetése a szomszédos szobába.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A takarító mozgásának (enterRoom) és a szoba tisztításának (clean) tesztelése.

- **Bemenet**

```

 create room r1
 create room r2
 setroom r2 gas true
 setroom r2 stickiness 3
 create student s
 create cleaner c
 neighbour r1 r2
 neighbour r2 r1
 addperson s r2
 addperson c r1
 enter r2 c
 status r2
 status r1

```

- **Elvárt kimenet**

```

 r2
 neighbours: r1,
 peopleInRoom: c,
 itemsInRoom:
 gas: false
 cursed: false
 capacity: 3
 curseActive: false
 stickiness: 0
 r1
 neighbours: r2,
 peopleInRoom: s,
 itemsInRoom:
 gas: false
 cursed: false
 capacity: 3
 curseActive: false
 stickiness: 1

```

## 7.2.9 Oktató találkozása személyekkel és tárgyakkal

- **Leírás**

Két szomszédos szoba létrehozása. A második szoba kapacitásának beállítása 5-re. Az egyikbe egy oktató és hallgató, a másikba egy hallgató, egy oktató és egy takarító létrehozása. Az első szobában lévő hallgató kezébe egy rongy létrehozása, hozzáadása és aktiválása. A második szobába továbbá minden tárgyból egy példány létrehozása és hozzáadása. Az első szobában lévő hallgató átléptetése a másik szobába, majd az egyedül lévő oktató átléptetése a másik szobába.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A különböző személyek megbuktatásra (kill), találkozására (meet, greet) való reagálásának tesztelése. Személyek és tárgyak találkozásának (meet) tesztelése.

- **Bemenet**

```

create room r1
create room r2
setroom r2 capacity 5
create student s1
create teacher t1
create student s2
create teacher t2
create cleaner c2
create rag rag
create rag r
create beerglass bg
create mask m
create falsemask fm
create sliderule sr
create falsesliderule fsr
create tvsz tv
create falsetvsz ftv
create airfresher a
create transistor t
create camembert c
neighbour r1 r2
addperson s1 r1
addperson t1 r1
addperson s2 r2
addperson t2 r2
addperson c2 r2
add rag s1
activate rag s1
add r r2
add bg r2
add m r2
add fm r2
add sr r2
add fsr r2
add tv r2
add ftv r2
add a r2
add t r2

```

```

add c r2
enter r2 s1
enter r2 t1
status r1
status r2
status t1
status t2
status

```

- **Elvárt kimenet**

```

r1
neighbours: r2,
peopleInRoom:
itemsInRoom:
gas: false
cursed: false
capacity: 3
curseActive: false
stickiness: 0
r2
neighbours:
peopleInRoom: t2, c2, s1, t1,
itemsInRoom: r, bg, m, fm, sr, fsr, tv, ftv, a, t, c,
gas: false
cursed: false
capacity: 5
curseActive: false
stickiness: 2
t1
location: r2
itemsInHand:
stunRemaining: 3
t2
location: r2
itemsInHand:
stunRemaining: 3
Game in progress

```

## 7.2.10 Tárgy felvétele és lerakása

- **Leírás**

Egy szoba létrehozása benne egy hallgatóval és 2 TVSZ-szel. A hallgató kezébe 4 TVSZ létrehozása. A hallgató megpróbálja felvenni először az egyik, majd a másik a földön lévő TVSZ-t, majd lerak egyet a nála lévő TVSZ-ekből.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A Room és a Student tárgykezeléssel kapcsolatos függvényeinek tesztelése (addItem, pickUpItem, dropItem, removeItem).

- **Bemenet**

```

create room r1
create student s
addperson s r1
create tvsz t1
create tvsz t2
create tvsz t3
create tvsz t4
create tvsz t5
create tvsz t6
add t1 r1
add t2 r1
add t3 s
add t4 s
add t5 s
add t6 s
pickup t1 s
pickup t2 s
drop t6 s
status r1
status s

```

- **Elvárt kimenet**

```

r1
neighbours:
peopleInRoom: s,
itemsInRoom: t2, t6,
gas: false
cursed: false
capacity: 3
curseActive: false
stickiness: 0
s
location: r1
itemsInHand: t3, t4, t5, t1,
stunRemaining: 0

```

### 7.2.11 Tárgyak időtelése

- **Leírás**

Egy szoba és benne két hallgató és egy oktató létrehozása. Az egyik hallgató kezébe egy 2 időegység műlva lejáró Rag, a másik kezébe egy 5 időegység műlva lejáró aktivált BeerGlass és egy tartóssággal már nem rendelkező, 6 időegység műlva lejáró aktivált Mask létrehozása. Az idő léptetése először 1-et, majd még egyet, majd 3-mat.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Az időérzékeny objektumok viselkedésének tesztelése. (timeRemaining, timeElapsed)

- **Bemenet**

```

create room r
create student s1
create student s2
create teacher t
create rag rag
create beerglass b
create mask m
addperson s1 r
addperson s2 r
addperson t r
add rag s1
activate rag s1
setinterval timeremaining rag 2
add b s2
activate b s2
add m s2
activate m s2
setremaining m 0
elapsetime 1
status s1
status s2
status t
elapsetime 1
status s1
status t
elapsetime 3
status

```

- **Elvárt kimenet**

```

s1
location: r
itemsInHand: rag,
stunRemaining: 0
s2
location: r
itemsInHand: b, m,
stunRemaining: 0
t
location: r
itemsInHand:
stunRemaining: 3
s1
location: r
itemsInHand:
stunRemaining: 0
t
location: r
itemsInHand:
stunRemaining: 2
Game over

```

### 7.2.12 Szobák módosítása

- **Leírás**

Egy szoba létrehozása 3 tárggyal benne, gázos tulajdonság igazra állítása. 3, az eredeti szobával szomszédos szoba létrehozása. A szoba felosztása, majd egyesítése a felosztás által újonnan létrehozott szobával.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Szoba szétválásának és szobák összeolvadásának tesztelése. (split, requestMerge, merge)

- **Bemenet**

```

create room r1
setroom r1 gas true
create room r2
create room r3
create room r4
create mask m
create tvsz tv
create beerglass b
neighbour r1 r2
neighbour r1 r3
neighbour r1 r4
neighbour r4 r1
add m r1
add tv r1
add b r1
status r1
split r1 r5
status r1
status r5
merge r1 r5 r6
status r6

```

- **Elvárt kimenet**

```

r1
neighbours: r2, r3, r4,
peopleInRoom:
itemsInRoom: m, tv, b,
gas: true
cursed: false
capacity: 3
curseActive: false
stickiness: 0
r1
neighbours: r4, r5,
peopleInRoom:
itemsInRoom: b,
gas: true
cursed: false
capacity: 3
curseActive: false
stickiness: 0
r5

```

```

neighbours: r2, r3, r1,
peopleInRoom:
itemsInRoom: m, tv,
gas: false
cursed: false
capacity: 3
curseActive: false
stickiness: 0
r6
neighbours: r2, r3, r4,
peopleInRoom:
itemsInRoom: m, tv, b,
gas: true
cursed: false
capacity: 3
curseActive: false
stickiness: 0

```

### 7.2.13 Gázhoz kapcsolódó tárgyak aktiválása

- **Leírás**

Egy szoba és benne egy hallgató létrehozása. A hallgatónak a kezébe egy Mask, egy Camembert és egy AirFresher létrehozása. A Camembert, majd az AirFresher aktiválása.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A Camembert és az AirFresher működésének tesztelése. (activate, setGas, removeItem)

- **Bemenet**

```

create room r1
create student s
create mask m
create camembert c
create airfresher a
addperson s r1
add c s
add a s
add m s
activate m s
activate c s
status r1
activate a s
status r1
status s
status m

```

- **Elvárt kimenet**

```

r1
neighbours:
peopleInRoom: s,
itemsInRoom:
gas: true
cursed: false

```

```

capacity: 3
curseActive: false
stickiness: 0
r1
neighbours:
peopleInRoom: s,
itemsInRoom:
gas: false
cursed: false
capacity: 3
curseActive: false
stickiness: 0
s
location: r1
itemsInHand: m,
stunRemaining: 0
m
location: r1
holder: s
timeRemaining: 6
duration: 4

```

### 7.2.14 Transistor használata

- Leírás**  
Három szoba létrehozása, melyek közül a középső szomszédos a két szélsővel. Egy szélsőbe egy hallgató létrehozása kezében két Transistor-ral. A Transistorok aktivállással való párosítása, majd az egyik kidobása. A hallgató átléptetése a középső, majd a másik szélső szobába és a nála lévő Transistor aktiválása.
- Ellenőrzött funkcionalitás, várható hibahelyek**  
A Transistor működésének tesztelése. (activate, dropItem, enterRoom, pairingRequests)
- Bemenet**

```

create room r1
create room r2
create room r3
create student s
create transistor t1
create transistor t2
neighbour r1 r2
neighbour r2 r3
addperson s r1
add t1 s
add t2 s
activate t1 s
activate t2 s
drop t2 s
enter r2 s
enter r3 s
activate t1 s
status s

```

*status t1*

- **Elvárt kimenet**

```
s
location: r1
itemsInHand:
stunRemaining: 0
t1
location: r3
holder: NULL
pair: t2
```

## 7.2.15 Bénított hallgató

- **Leírás**

Két szomszédos szoba és az egyikbe egy hallgató létrehozása. A hallgató bénítása majd átléptetése a másik szobába.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Bénultság tesztelése átlépéskor (enterRoom)

- **Bemenet**

```
create room r1
create room r2
create student s
setstun s 5
neighbour r1 r2
addperson s r1
enter r2 s
status s
```

- **Elvárt kimenet**

```
s
location: r1
itemsInHand:
stunRemaining: 5
```

## 7.2.16 Elátkozott szoba

- **Leírás**

Két szomszédos szoba létrehozása, melyek közül az egyik elátkozott. Az elátkozott szobába két hallgató létrehozása. Ez egyik hallgató átléptetése a másik szobába. Az idő léptetése, hogy aktívvá váljon az átok. A másik hallgató átléptetés a másik szobába. Az idő léptetése, hogy inaktívvá váljon az átok. A másik hallgató átléptetésének megismétlése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Elátkozódás tesztelése. (timeElapsed, curseActive, movePerson)

- **Bemenet**

```
create room r1
create room r2
neighbour r1 r2
setroom r1 cursed true
create student s1
create student s2
addperson s1 r1
```

```

addperson s2 r1
enter r2 s1
elapsetime 5
enter r2 s2
status r1
elapsetime 5
enter r2 s2
status r1

```

- **Elvárt kimenet**

```

r1
neighbours: r2,
peopleInRoom: s2,
itemsInRoom:
gas: false
cursed: true
capacity: 3
curseActive: true
stickiness: 0
r1
neighbours: r2,
peopleInRoom:
itemsInRoom:
gas: false
cursed: true
capacity: 3
curseActive: false
stickiness: 0

```

### 7.2.17 Teli szoba

- **Leírás**

Két szomszédos szoba és az egyikbe egy hallgató létrehozása. A másik szoba kapacitásának 0-ra állítása. A hallgató átléptetése a másik szobába.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Teli szobába lépés tesztelése. (acceptPerson)

- **Bemenet**

```

create room r1
create room r2
setroom r2 capacity 0
neighbour r1 r2
create student s1
addperson s1 r1
enter r2 s1
status r2

```

- **Elvárt kimenet**

```

r2
neighbours:
peopleInRoom:
itemsInRoom:
gas: false
cursed: false

```

*capacity: 0  
curseActive: false  
stickiness: 0*

### 7.2.18 Tanár és logarléc

- **Leírás**  
Egy szoba, benne egy oktató és egy SlideRule létrehozása. Az oktató felveszi a SlideRule-t.
- **Ellenőrzött funkcionalitás, várható hibahelyek**  
Teacher és SlideRule interakciójának tesztelése. (pickedUpSlideRule)
- **Bemenet**

*create room r  
create teacher t  
addperson t r  
create sliderule sr  
add sr r  
pickup sr t  
status sr*

- **Elvárt kimenet**

*sr  
location: r  
holder: NULL*

### 7.2.19 Cleaner logarléc

- **Leírás**  
Egy szoba, benne egy takarító és egy SlideRule létrehozása. A takarító felveszi a SlideRule-t.
- **Ellenőrzött funkcionalitás, várható hibahelyek**  
Cleaner és SlideRule interakciójának tesztelése. (pickedUpSlideRule)
- **Bemenet**

*create room r  
create cleaner c  
addperson c r  
create sliderule sr  
add sr r  
pickup sr c  
status sr*

- **Elvárt kimenet**

*sr  
location: r  
holder: NULL*

### 7.2.20 Cleaner átlép cursed szobába

- **Leírás**  
Két szomszédos szoba létrehozása, melyekből az egyik gázos és elátkozott, és bele egy hallgató, a másikba egy takarító létrehozása. A gázos szoba stickiness-ének beállítása. A takarító átléptetése a szomszédos szobába.
- **Ellenőrzött funkcionalitás, várható hibahelyek**  
A takarító mozgásának tesztelése, takarítás elátkozott szobán (clean)

- **Bemenet**

```

create room r1
create room r2
setroom r2 gas true
setroom r2 cursed true
setroom r2 stickiness 3
elapsetime 5
create student s
create cleaner c
addperson s r2
addperson c r1
neighbour r1 r2
neighbour r2 r1
enter r2 c
status r2

```

- **Elvárt kimenet**

```

r2
neighbours: r1,
peopleInRoom: s, c,
itemsInRoom:
gas: false
cursed: true
capacity: 3
curseActive: true
stickiness: 0

```

### 7.2.21 Sikertelen tranzisztor párosítás

- **Leírás**

Két szomszédos szoba létrehozása. Az egyikben egy hallgató és a kezében két Transistor létrehozása. Egy Transistor aktiválása, majd eldobása. A másik Transistor aktiválása. A hallgató átmozgatása a másik szobába, majd a nála lévő Transistor aktiválása.

- **Ellenőrzött funkcionális, várható hibahelyek**

Transistor-ok párosíthatóságának ellenőrzése (activate, pairingRequests).

- **Bemenet**

```

create room r1
create room r2
create student s
create transistor t1
create transistor t2
neighbour r1 r2
addperson s r1
add t1 s
add t2 s
activate t1 s
drop t1 s
activate t2 s
enter r2 s
activate t2 s
status t2

```

*status s*

- **Elvárt kimenet**

*t2*

*location: r2*

*holder: s*

*pair: NULL*

*s*

*location: r2*

*itemsInHand: t2,*

*stunRemaining: 0*

## 7.2.22 Ragacsos szoba

- **Leírás**

Egy szoba és benne egy hallgató és egy TVSZ létrehozása. A szoba ragacsosságának határérték fölé állítása. A TVSZ hallgató általi felvétele.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ragacsosság hatásának a tárgyfelvételre tesztelése (addItem, pickUpItem).

- **Bemenet**

*create room r*

*setroom r stickiness 6*

*create student s*

*addperson s r*

*create tvsz t*

*add t r*

*pickup t s*

*status s*

- **Elvárt kimenet**

*s*

*location: r*

*itemsInHand:*

*stunRemaining: 0*

## 7.2.23 Sikertelen kettéválás és egyesülés

- **Leírás**

Két szomszédos szoba létrehozása, az egyikbe egy hallgatóval. A hallgatót tartalmazó szoba felosztása, majd a két szoba egyesítése mindenkét paraméterezi sorrenddel.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Szobaegyesülés és szétválás sikertelen lefutásának tesztelése (split, mergeRequest, merge)

- **Bemenet**

*create room r1*

*create room r2*

*neighbour r1 r2*

*neighbour r2 r1*

*create student s*

*addperson s r1*

*split r1 r3*

*merge r1 r2 r4*

*merge r2 r1 r5*

```

status r1
status r2
• Elvárt kimenet
r1
neighbours: r2,
peopleInRoom: s,
itemsInRoom:
gas: false
cursed: false
capacity: 3
curseActive: false
stickiness: 0
r2
neighbours: r1,
peopleInRoom:
itemsInRoom:
gas: false
cursed: false
capacity: 3
curseActive: false
stickiness: 0

```

### 7.2.24 Takarító általi mozgatás

- **Leírás**  
Három szoba létrehozása, a középső kölcsönösen szomszédos a két szélsővel. A középső szobához 2 hallgató adása. Az egyik szélső szobához egy takarító adása. A takarító szobájának kapacitása 1-re állítása. A másik szélső szoba kapacitásának 0-ra állítása. A takarító középső szobába lépése.
- **Ellenőrzött funkcionalitás, várható hibahelyek**  
Takarító szobába lépésének és ezáltal más szereplők mozgatásának tesztelése (clean)
- **Bemenet**

```

create room r1
setroom r1 capacity 1
create room r2
create room r3
setroom r3 capacity 0
neighbour r1 r2
neighbour r2 r1
neighbour r2 r3
neighbour r3 r2
create student s1
create student s2
create cleaner c
addperson s1 r2
addperson s2 r2
addperson c r1
enter r2 c
status r1
status r2
status r3

```

- **Elvárt kimenet**

```

r1
neighbours: r2,
peopleInRoom: s1,
itemsInRoom:
gas: false
cursed: false
capacity: 1
curseActive: false
stickiness: 1
r2
neighbours: r1, r3,
peopleInRoom: s2, c,
itemsInRoom:
gas: false
cursed: false
capacity: 3
curseActive: false
stickiness: 0
r3
neighbours: r2,
peopleInRoom:
itemsInRoom:
gas: false
cursed: false
capacity: 0
curseActive: false
stickiness: 0

```

### 7.2.25 Tárgyak eltűnése és aktivált rongy földön viselkedése

- **Leírás**

Két szomszédos szoba létrehozása. Az egyikhez egy oktató hozzáadása. Az oktató kezébe egy maszk létrehozása. A maszk aktiválása, timeRemaining-jének 1-re és duration-jének 0-ra állítása. Az oktató szobájába egy söröspohár tétele, aktiválása és timeRemaining-jének 1-re állítása. A másik szobába egy rongy rakása, aktiválása és timeRemaining-jének 1-re állítása. Az oktató másik szobába lépése, majd 1 egységnyi idő telésének jelzése. Az oktató eredeti szobába való (sikertelen) visszaléptetése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Oktató és rongy találkozásának (meet), kábult ember mozgásának (stun, enterRoom), valamint tárgyak eltűnésének (timeElapsed, timeRemaining) tesztelése.

- **Bemenet**

```

create room r1
create room r2
neighbour r1 r2
neighbour r2 r1
create teacher t
create mask m
create beerglass b
create rag r
addperson t r1

```

```

add m t
setinterval activated m true
setinterval timeremaining m 1
setremaining m 0
add b r2
setinterval activated b true
setinterval timeremaining b 1
add r r2
setinterval activated r true
setinterval timeremaining r 1
enter r2 t
status r2
status t
elapsetime 1
enter r1 t
status r2
status t

```

- **Elvárt kimenet**

```

r2
neighbours: r1,
peopleInRoom: t,
itemsInRoom: b, r,
gas: false
cursed: false
capacity: 3
curseActive: false
stickiness: 1
t
location: r2
itemsInHand: m,
stunRemaining: 3
r2
neighbours: r1,
peopleInRoom: t,
itemsInRoom:
gas: false
cursed: false
capacity: 3
curseActive: false
stickiness: 1
t
location: r2
itemsInHand:
stunRemaining: 2

```

## 7.2.26 Ragacsosság és sörösüveg

- **Leírás**

Két szomszédos szoba létrehozása. Az egyikbe egy hallgató hozzáadása és egy sör a földre. A sör aktiválása. Egy tranzisztor földre rakása. A másik szobába egy oktató rakása. A hallgató szobájának ragacsosság beállítása éppen a határérték alá. Hallgató seed-jének beállítása. Tranzisztor és sör sikeres felvétele. Oktató szobába lépése. Hallgató megvédi magát, de eleji a tranzisztort (pszeudo-randomitás). Tranzisztor felvétele (sikertelenül).

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Stickiness általi különböző viselkedés, sörösüveg használatának tesztelése  
(saveFromDeath, dropRandomItem, addItem, pickUpItem)

- **Bemenet**

```

create room r1
create room r2
neighbour r1 r2
neighbour r2 r1
create student s
addperson s r1
create beerglass b
setinterval activated b true
add b r1
create transistor t1
add t1 r1
create teacher t
addperson t r2
setroom r1 stickiness 4
setseed s -1
pickup t1 s
pickup b s
status s
enter r1 t
pickup t1 s
status s
status r1

```

- **Elvárt kimenet**

```

s
location: r1
itemsInHand: t1, b,
stunRemaining: 0
s
location: r1
itemsInHand: b,
stunRemaining: 0
r1
neighbours: r2,
peopleInRoom: s, t,
itemsInRoom: t1,
gas: false
cursed: false
capacity: 3
curseActive: false
stickiness: 5

```

### 7.2.27 Hallgató megvédi magát és nyer

- **Leírás**

8 szoba létrehozása megfelelő szomszédokkal. Ebből egyikbe egy tanár, egy másikba egy hallgató kezében egy tvsz-szel, egy harmadikba földön lévő aktiválatlan rongy, egy negyedikbe pedig Logarléc inicializálása. A rongy szobája kölcsönösen szomszédos a másik hárommal.

Oktató átlépése a rongy szobájába. Hallgató átlépése a tanár szobájába, megvédi magát a TVSZ-szel, ennek ellenőrzése. A hallgató átlépése a logarléc szobájába, majd logarléc felvétele. Győzelem tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Hallgató és tanár interakciója (meet, kill), TVSZ használata (saveFromDeath), szobák közti mozgás (enterRoom, movePerson, acceptPerson), Logarléc felvétele (addItem, pickUpItem, pickedUpSlideRule) és játék állapotának (Interpreter – status) tesztelése.

- **Bemenet**

```

create room r1
create room r2
create room r3
create room r4
create room r5
create room r6
create room r7
create room r8
neighbour r1 r2
neighbour r2 r1
neighbour r8 r7
neighbour r7 r8
neighbour r7 r2
neighbour r2 r7
neighbour r2 r6
neighbour r6 r2
neighbour r6 r5

```

*neighbour r5 r6*  
*neighbour r6 r4*  
*neighbour r4 r6*  
*neighbour r5 r2*  
*neighbour r2 r5*  
*neighbour r5 r4*  
*neighbour r4 r5*  
*neighbour r4 r3*  
*neighbour r3 r4*  
*neighbour r2 r3*  
*neighbour r3 r2*  
*create teacher T*  
*create student H*  
*addperson T r1*  
*addperson H r7*  
*create tvsz tv*  
*create rag r*  
*create sliderule sr*  
*add tv H*  
*add r r2*  
*add sr r3*  
*enter r2 T*  
*pickup r T*  
*enter r2 H*  
*status r2*  
*status tv*  
*enter r3 H*  
*pickup sr H*  
*status*

- **Elvárt kimenet**

*r2*  
*neighbours: r1, r7, r6, r5, r3,*  
*peopleInRoom: T, H,*  
*itemsInRoom:*  
*gas: false*  
*cursed: false*  
*capacity: 3*  
*curseActive: false*  
*stickiness: 2*  
*tv*  
*location: r2*  
*holder: H*  
*usesRemaining: 2*  
*Win*

## 7.2.28 Játék állapotai és hamis Logarléc

- **Leírás**

Két szoba inicializálása, az egyikben egy hallgatóval, a másikban egy oktatóval. A hallgató kezébe egy aktivált, 1 időegység múlva lejáró sörösüveg. A tanár szobájában egy hamis Logarléc. A tanár felveszi a hamis Logarlécet (sikertelenül). A hallgató átlépése a másik szobába. A hallgató felveszi a hamis logarlécet. Egységnyi idő telése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Hamis logarléc működése (pickedUpSlideRule) és játékállapot (Interpreter - status).

- **Bemenet**

```

create room r1
create room r2
neighbour r1 r2
create student s
addperson s r1
create beerglass b
add b s
setinterval activated b true
setinterval timeremaining b 1
create teacher t
addperson t r2
create falsesliderule fsr
add fsr r2
pickup fsr t
enter r2 s
status r2
pickup fsr s
status
elapsetime 1
status

```

- **Elvárt kimenet**

```

r2
neighbours:
peopleInRoom: t, s,
itemsInRoom: fsr, b,
gas: false
cursed: false
capacity: 3
curseActive: false
stickiness: 1
Game in progress
Game over

```

### 7.3 A tesztelést támogató programok tervei

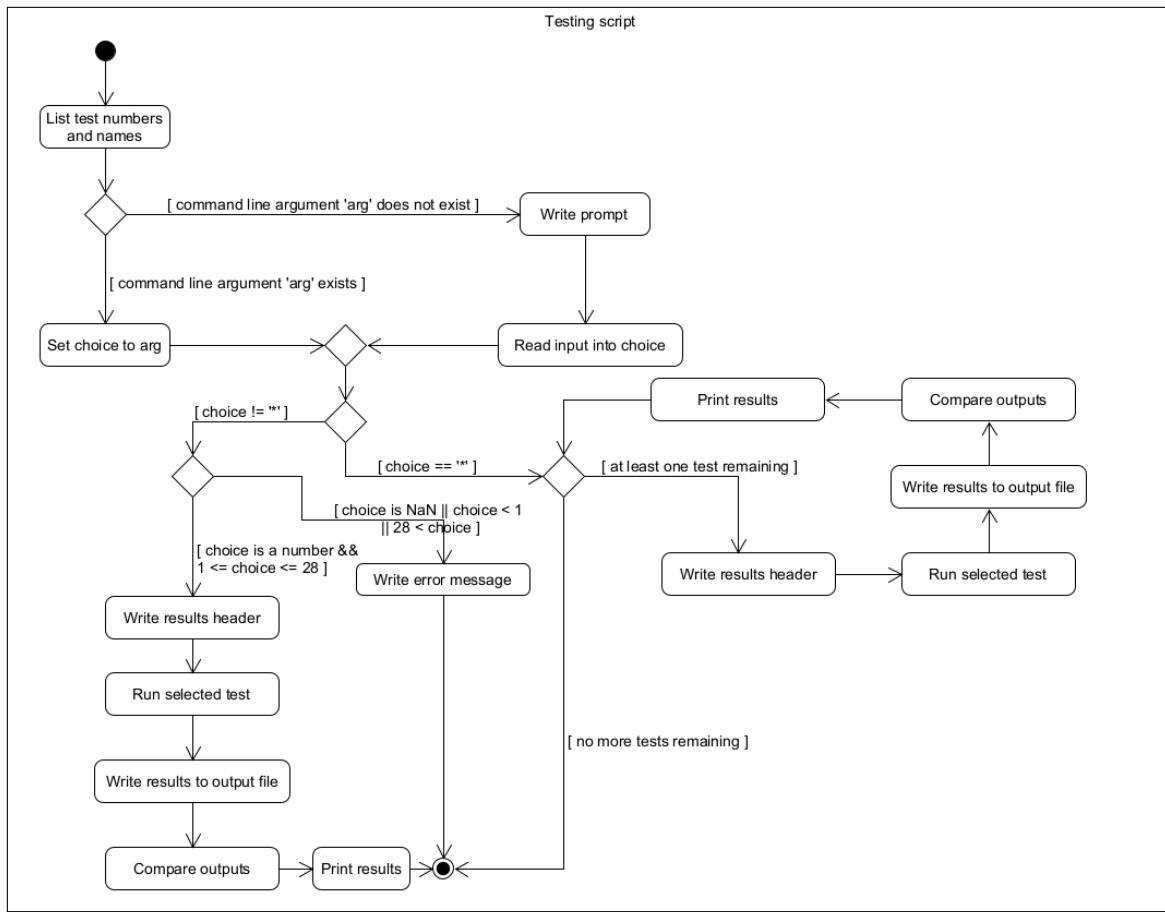
- A tesztelő program: ***teszt.bat***:  
A tesztelést egy Windows parancssor alatt futtatható batch script fájl fogja segíteni. Ez a tesztek sorszámmal és címmel való felsorolása után megkérdezi a felhasználótól, melyik tesztet (vagy esetleg az összeset) kívánja futtatni, majd futtatja a tesztet a kért teszthez tartozó bemeneti fájl megadásával, majd a kimenetet egy fájlba irányítja. Ezután ellenőrzi az FC parancs segítségével az átirányított kimeneti fájl egyenlőségét a teszthez tartozó elvárt kimeneti fájlhoz képest. Egy teszt akkor sikeres, ha az FC kimenete az, hogy a két fájl között nincs különbség. Alternatív lehetőséggént, a teszteset számát az első parancssori kapcsolóval is meg lehet adni.
- Futtatás: Parancssorból, „teszt [teszteset száma | '\*' ]”.
- Bemenetek:
  - Egy szám (a teszteset száma), vagy '\*' az összes teszteset kiválasztásához. Ha parancssori kapcsolóval adtuk meg a teszteset számát, akkor további kiválasztásra nincs szükség.
- Kimenetek:
  - Az *FC.EXE* program kimenete, a megfelelő tesztesetek be- és kimeneteire vonatkozva. minden teszteset kimenete a '\*\*\*\* N. TESZTESET EREDMÉNYE \*\*\*\*' fejléccel kezdődik, ahol N az éppen futó teszteset számát jelöli.
  - Az *FC.EXE* lehetséges kimenetei és ezek értelmezése a tesztelés értelmében:
    - *"FC: no differences encountered"*: A teszt sikeresnek értékelhető, hiszen a kapott és az elvárt kimenetek pontosan megegyeznek.
    - *Bármi más*: A teszt sikertelen, hiszen a kapott és az elvárt kimenet fájlok tartalmában van különbség. Ekkor ez a különbség is kiírásra kerül.
- Működése:

```

eljárás teszt:
futtasd: "type tests.txt"
választás = 0
ha <meg van adva parancssori kapcsoló> akkor
 írd ki: "Melyik tesztet kell futtatni? (szám vagy '*' az összeshez): "
 olvasd be: választás
egyébként
 választás = kapcsolók[1]
ha <választás = '*'> akkor
 ciklus i = 1-től 28-ig
 írd ki: "**** %i. TESZTESET EREDMÉNYE ****"
 ahol %i helyére helyettesítsd be i-t
 futtasd: "java -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -
 Dsun.stderr.encoding=UTF-8 -cp ./out/ Interpreter <%i_input.txt
 >%i_output.txt && FC %i_output.txt %i_expected.txt"
 ahol %i_ helyére helyettesítsd be i-t
különben
 ha <választás az egy szám> és <1 <= választás <= 28> akkor
 írd ki: "**** %választás. TESZTESET EREDMÉNYE ****"
 ahol %választás helyére helyettesítsd be választás-t
 futtasd: "java proto.jar <%választás_input.txt >%választás_output.txt
 && FC %választás_output.txt %választás_expected.txt"
 ahol %választás_ helyére helyettesítsd be választás-t
különben
 írd ki: "Hiba! Ilyen teszteset nem létezik!"
eljárás vége

```

13. pszeudo-kód: a tesztelő szkript



5. aktivitás diagram: A tesztelő szkript

## 7.4 Napló

| Kezdet              | Időtartam | Résztvevők | Leírás                                                  |
|---------------------|-----------|------------|---------------------------------------------------------|
| 2024. 04. 08. 14:00 | 2 óra     | Tömöri     | Room osztály tervezé                                    |
| 2024. 04. 08. 20:00 | 5 óra     | Görömbey   | Első 10 teszt részletes tervezé                         |
| 2024. 04. 08. 21:00 | 3 óra     | Tömöri     | Person osztály és leszármazottainak tervezé             |
| 2024. 04. 08. 21:00 | 3 óra     | Sági       | IntervalItem-ek, TVSZ, Tranzisztor és SlideRule tervezé |
| 2024. 04. 08. 22:00 | 1.5 óra   | Vizhányó   | AirFresher és Camembert tervezé                         |
| 2024. 04. 09. 10:00 | 2 óra     | Görömbey   | Parancsvezérlő létrehozásának tervezése                 |
| 2024. 04. 09. 10:00 | 3 óra     | Vizhányó   | 10 terv inputja                                         |
| 2024. 04. 09. 11:00 | 2 óra     | Tömöri     | 6 terv inputja                                          |

|                     |         |                                                 |                                                                                                                       |
|---------------------|---------|-------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| 2024. 04. 10. 10:00 | 2 óra   | Tömöri<br>Görömbey<br>Sági<br>Vízhányó<br>Héjja | Konzultáció                                                                                                           |
| 2024. 04. 10. 13:00 | 4 óra   | Vízhányó                                        | Tesztbemenetek javítása, status-ok írása a megfelelő összehasonlításhoz.                                              |
| 2024. 04. 10. 13:00 | 7 óra   | Tömöri                                          | Osztályleírások pontosítása, metódusok pontos leírása, formázás.                                                      |
| 2024. 04. 11. 10:00 | 2.5 óra | Vízhányó                                        | Elvárt teszt kimenetek megírása, tesztek ellenőrzése, dokumentálása                                                   |
| 2024. 04. 13. 10:00 | 2 óra   | Héjja                                           | Tesztelest támogató program specifikálása                                                                             |
| 2024. 04. 13. 14:00 | 2 óra   | Sági                                            | Pszeudokódok és aktivitás diagramok készítése                                                                         |
| 2024. 04. 13. 15:00 | 1 óra   | Görömbey                                        | Interpreter formázás                                                                                                  |
| 2024. 04. 13. 17:00 | 0.5 óra | Vízhányó                                        | Elvárt kimenetek formázása                                                                                            |
| 2024. 04. 13. 17:00 | 1 óra   | Sági                                            | Pszeudokódok javítása                                                                                                 |
| 2024. 04. 13. 17:00 | 5.5 óra | Tömöri                                          | Dokumentum formázása, ábrák és pszeudokódot átírása. input fájlok formázása, állapotdiagramok elkészítésében segítés. |
| 2024. 04. 13. 19:30 | 4 óra   | Héjja                                           | Tranzisztor/tesztelő program állapotdiagramok létrehozása, tesztelő program javítása                                  |

## 8. Prototípus beadása

### 8.1 Fordítási és futtatási útmutató

#### 8.1.1 Fájllista

| Fájl neve           | Méret (byte) | Keletkezés ideje    | Tartalom                                |
|---------------------|--------------|---------------------|-----------------------------------------|
| AirFresher.java     | 2092         | 2024. 04. 22. 20:00 | AirFresher osztály implementációja      |
| BeerGlass.java      | 2872         | 2024. 04. 22. 20:00 | BeerGlass osztály implementációja       |
| Camembert.java      | 2031         | 2024. 04. 22. 20:00 | Camembert osztály implementációja       |
| Cleaner.java        | 2442         | 2024. 04. 22. 20:00 | Cleaner osztály implementációja         |
| FalseMask.java      | 1169         | 2024. 04. 22. 20:00 | FalseMask osztály implementációja       |
| FalseSlideRule.java | 656          | 2024. 04. 22. 20:00 | FalseSlideRule osztály implementációja  |
| FalseTVSZ.java      | 996          | 2024. 04. 22. 20:00 | FalseTVSZ osztály implementációja       |
| IntervalItem.java   | 1859         | 2024. 04. 22. 20:00 | IntervalItem osztály implementációja    |
| Item.java           | 2849         | 2024. 04. 22. 20:00 | Item osztály implementációja            |
| ItemHandler.java    | 588          | 2024. 04. 22. 20:00 | ItemHandler interface implementációja   |
| Mask.java           | 3754         | 2024. 04. 22. 20:00 | Mask osztály implementációja            |
| Person.java         | 7020         | 2024. 04. 22. 20:00 | Person osztály implementációja          |
| Rag.java            | 2689         | 2024. 04. 22. 20:00 | Rag osztály implementációja             |
| Room.java           | 16232        | 2024. 04. 22. 20:00 | Room osztály implementációja            |
| SlideRule.java      | 2344         | 2024. 04. 22. 20:00 | SlideRule osztály implementációja       |
| Student.java        | 2327         | 2024. 04. 22. 20:00 | Student osztály implementációja         |
| Teacher.java        | 1886         | 2024. 04. 22. 20:00 | Teacher osztály implementációja         |
| TimeSensitive.java  | 381          | 2024. 04. 22. 20:00 | TimeSensitive interface implementációja |
| Transistor.java     | 4012         | 2024. 04. 22. 20:00 | Transistor osztály implementációja      |
| TVSZ.java           | 2670         | 2024. 04. 22. 20:00 | TVSZ osztály implementációja            |
| Interpreter.java    | 39293        | 2024. 04. 22. 20:00 | Interpreter osztály implementációja     |
| teszt.bat           | 1042         | 2024. 04. 22. 20:00 | Tesztellenőrző szkript kódja            |
| tests.txt           | 1060         | 2024. 04. 22. 20:00 | Tesztek sorszáma és címe                |
| 1input.txt          | 284          | 2024. 04. 22. 20:00 | 1.teszt bemeneti parancsai              |
| 2input.txt          | 190          | 2024. 04. 22. 20:00 | 2.teszt bemeneti parancsai              |
| 3input.txt          | 174          | 2024. 04. 22. 20:00 | 3.teszt bemeneti parancsai              |
| 4input.txt          | 180          | 2024. 04. 22. 20:00 | 4.teszt bemeneti parancsai              |
| 5input.txt          | 266          | 2024. 04. 22. 20:00 | 5.teszt bemeneti parancsai              |
| 6input.txt          | 186          | 2024. 04. 22. 20:00 | 6.teszt bemeneti parancsai              |
| 7input.txt          | 150          | 2024. 04. 22. 20:00 | 7.teszt bemeneti parancsai              |
| 8input.txt          | 214          | 2024. 04. 22. 20:00 | 8.teszt bemeneti parancsai              |
| 9input.txt          | 712          | 2024. 04. 22. 20:00 | 9.teszt bemeneti parancsai              |
| 10input.txt         | 266          | 2024. 04. 22. 20:00 | 10.teszt bemeneti parancsai             |
| 11input.txt         | 402          | 2024. 04. 22. 20:00 | 11.teszt bemeneti parancsai             |
| 12input.txt         | 308          | 2024. 04. 22. 20:00 | 12.teszt bemeneti parancsai             |
| 13input.txt         | 217          | 2024. 04. 22. 20:00 | 13.teszt bemeneti parancsai             |
| 14input.txt         | 281          | 2024. 04. 22. 20:00 | 14.teszt bemeneti parancsai             |
| 15input.txt         | 118          | 2024. 04. 22. 20:00 | 15.teszt bemeneti parancsai             |

|                |     |                     |                             |
|----------------|-----|---------------------|-----------------------------|
| 16input.txt    | 234 | 2024. 04. 22. 20:00 | 16.teszt bemeneti parancsai |
| 17input.txt    | 132 | 2024. 04. 22. 20:00 | 17.teszt bemeneti parancsai |
| 18input.txt    | 103 | 2024. 04. 22. 20:00 | 18.teszt bemeneti parancsai |
| 19input.txt    | 103 | 2024. 04. 22. 20:00 | 19.teszt bemeneti parancsai |
| 20input.txt    | 241 | 2024. 04. 22. 20:00 | 20.teszt bemeneti parancsai |
| 21input.txt    | 236 | 2024. 04. 22. 20:00 | 21.teszt bemeneti parancsai |
| 22input.txt    | 118 | 2024. 04. 22. 20:00 | 22.teszt bemeneti parancsai |
| 23input.txt    | 167 | 2024. 04. 22. 20:00 | 23.teszt bemeneti parancsai |
| 24input.txt    | 313 | 2024. 04. 22. 20:00 | 24.teszt bemeneti parancsai |
| 25input.txt    | 459 | 2024. 04. 22. 20:00 | 25.teszt bemeneti parancsai |
| 26input.txt    | 347 | 2024. 04. 22. 20:00 | 26.teszt bemeneti parancsai |
| 27input.txt    | 709 | 2024. 04. 22. 20:00 | 27.teszt bemeneti parancsai |
| 28input.txt    | 327 | 2024. 04. 22. 20:00 | 28.teszt bemeneti parancsai |
| 1expected.txt  | 11  | 2024. 04. 22. 20:00 | 1.teszt elvárt kimenete     |
| 2expected.txt  | 11  | 2024. 04. 22. 20:00 | 2.teszt elvárt kimenete     |
| 3expected.txt  | 65  | 2024. 04. 22. 20:00 | 3.teszt elvárt kimenete     |
| 4expected.txt  | 18  | 2024. 04. 22. 20:00 | 4.teszt elvárt kimenete     |
| 5expected.txt  | 50  | 2024. 04. 22. 20:00 | 5.teszt elvárt kimenete     |
| 6expected.txt  | 50  | 2024. 04. 22. 20:00 | 6.teszt elvárt kimenete     |
| 7expected.txt  | 53  | 2024. 04. 22. 20:00 | 7.teszt elvárt kimenete     |
| 8expected.txt  | 262 | 2024. 04. 22. 20:00 | 8.teszt elvárt kimenete     |
| 9expected.txt  | 429 | 2024. 04. 22. 20:00 | 9.teszt elvárt kimenete     |
| 10expected.txt | 201 | 2024. 04. 22. 20:00 | 10.teszt elvárt kimenete    |
| 11expected.txt | 270 | 2024. 04. 22. 20:00 | 11.teszt elvárt kimenete    |
| 12expected.txt | 567 | 2024. 04. 22. 20:00 | 12.teszt elvárt kimenete    |
| 13expected.txt | 365 | 2024. 04. 22. 20:00 | 13.teszt elvárt kimenete    |
| 14expected.txt | 92  | 2024. 04. 22. 20:00 | 14.teszt elvárt kimenete    |
| 15expected.txt | 50  | 2024. 04. 22. 20:00 | 15.teszt elvárt kimenete    |
| 16expected.txt | 257 | 2024. 04. 22. 20:00 | 16.teszt elvárt kimenete    |
| 17expected.txt | 124 | 2024. 04. 22. 20:00 | 17.teszt elvárt kimenete    |
| 18expected.txt | 31  | 2024. 04. 22. 20:00 | 18.teszt elvárt kimenete    |
| 19expected.txt | 31  | 2024. 04. 22. 20:00 | 19.teszt elvárt kimenete    |
| 20expected.txt | 132 | 2024. 04. 22. 20:00 | 20.teszt elvárt kimenete    |
| 21expected.txt | 95  | 2024. 04. 22. 20:00 | 21.teszt elvárt kimenete    |
| 22expected.txt | 49  | 2024. 04. 22. 20:00 | 22.teszt elvárt kimenete    |
| 23expected.txt | 259 | 2024. 04. 22. 20:00 | 23.teszt elvárt kimenete    |
| 24expected.txt | 399 | 2024. 04. 22. 20:00 | 24.teszt elvárt kimenete    |
| 25expected.txt | 371 | 2024. 04. 22. 20:00 | 25.teszt elvárt kimenete    |
| 26expected.txt | 248 | 2024. 04. 22. 20:00 | 26.teszt elvárt kimenete    |
| 27expected.txt | 202 | 2024. 04. 22. 20:00 | 27.teszt elvárt kimenete    |
| 28expected.txt | 167 | 2024. 04. 22. 20:00 | 28.teszt elvárt kimenete    |

### 8.1.2 Fordítás

A szoftver x86-os architektúrán, Windows operációs rendszer alatt fordítható. Pontosabban a forrásprogramnak a kari felhőben (<https://niif.cloud.bme.hu/>) vagy (<https://fured.cloud.bme.hu/>), a „Windows 10 20H2 - JDK-Eclipse-WSU” sablonnal meghatározott környezetben fordítható és futtatható. Ehhez belépési segédlet itt található: <https://www.mit.bme.hu/node/11462>. Egy ilyen típusú virtuális gépre csatlakozás után másolja fel (pl.: vágólapon keresztül) a rendelkezésre álló *proto.zip*-ból kicsomagolt *proto* mappát.

Megjegyzés: Ha a kicsomagolás során egy új, a zip-pel megegyező mappába teszi a fájlokat, akkor kettő proto mappa lesz (az egyikben lesz a másik, ami eredetileg a zip-ben volt). Ebben az esetben, ha *proto* mappáról beszélünk, a struktúrában lejjebb lévő mappát értjük.

A kód fordítását a *proto* mappában megnyitott cmd shellben teheti meg a következő paranccsal:

```
javac -encoding utf8 -d ./out/ ./src/main/java/model/*.java
./src/main/java/Interpreter.java
```

Ezzel előáll a bináris futtatható kód, futtathatja a programot, vagy az azt használó tesztelő szkriptet.

### 8.1.3 Futtatás

A szoftver x86-os architektúrán, Windows operációs rendszer alatt futtatható. Pontosabban a forrásprogramnak a kari felhőben (<https://niif.cloud.bme.hu/>) vagy (<https://fured.cloud.bme.hu/>), a „Windows 10 20H2 - JDK-Eclipse-WSU” sablonnal meghatározott környezetben fordítható és futtatható. Ehhez belépési segédlet itt található: <https://www.mit.bme.hu/node/11462>. Amennyiben egy ilyen gépen elvégezte az előző pontban (10.1.2 Fordítás) megadott lépésekét és fordította a programfájlokat, a program futtatható a *proto* mappában megnyitott cmd shellben a következő paranccsal:

```
java -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -
Dsun.stderr.encoding=UTF-8 -cp ./out/ Interpreter
```

Így a futó program a standard bemenetén várja a parancsokat. Amennyiben fájlt szeretnénk megadni be- vagy kimenetként, arra is van lehetőség a standard be és kimenetek átirányításával. Például: tesztelésnél egy *input.txt* fájl használata bemenetként és egy *output.txt* fájl létrehozása kimenetként:

```
java -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8
-Dsun.stderr.encoding=UTF-8 -cp ./out/ Interpreter <
/path/to/input.txt > /path/to/output.txt
```

A programból „exit” parancs megadásával, vagy EOF jellet lehet kilépni.

A tesztek futtatása a tesztelő programmal (*teszt.bat*):

Amennyiben a megadott gépen elvégezte az előző pontban (10.1.2 Fordítás) megadott lépéseket és fordította a programfájlokat, a tesztelő program futtatható a *proto* mappában megnyitott cmd shellben a következő paranccsal:

```
.\testing\teszt.bat [teszt száma]
```

Amennyiben nem adott meg előre tesztesetet, a megjelenő cím és sorszám listából választható ki a futtatni kívánt teszt. Egy teszt sorszámnak megadásával választható egy teszt. Továbbá „\*\*” karakterrel választható az összes teszt futtatása. Kimenetként a választott (egy vagy összes) teszt eredményét láthatjuk. Az „FC: no differences encountered” kimenet esetén a teszt helyesen futott le, egyébként nem. A teszt(ek) futtatása után a program kilép.

Amennyiben futtatni szeretne még további teszteket, futtassa újra a tesztelőprogramot!

## **8.2 Tesztek jegyzőkönyvei**

### **8.2.1 Átlépés oktatóhoz nem védő tárgyakkal 1**

|                 |                     |
|-----------------|---------------------|
| Tesztelő neve   | Vizhányó Miklós     |
| Teszt időpontja | 2024. 04. 20. 17:33 |

### **8.2.2 Átlépés oktatóhoz nem védő tárgyakkal 2**

|                 |                     |
|-----------------|---------------------|
| Tesztelő neve   | Vizhányó Miklós     |
| Teszt időpontja | 2024. 04. 20. 17:33 |

### **8.2.3 Átlépés oktatóhoz TVSZ-szel**

|                 |                     |
|-----------------|---------------------|
| Tesztelő neve   | Vizhányó Miklós     |
| Teszt időpontja | 2024. 04. 20. 17:33 |

### **8.2.4 Átlépés oktatóhoz BeerGlass-szal**

|                 |                     |
|-----------------|---------------------|
| Tesztelő neve   | Vizhányó Miklós     |
| Teszt időpontja | 2024. 04. 20. 17:33 |

### **8.2.5 Átlépés gázba nem védő tárgyakkal 1**

|                 |                     |
|-----------------|---------------------|
| Tesztelő neve   | Vizhányó Miklós     |
| Teszt időpontja | 2024. 04. 20. 17:33 |

### **8.2.6 Átlépés gázba nem védő tárgyakkal 2**

|                 |                     |
|-----------------|---------------------|
| Tesztelő neve   | Vizhányó Miklós     |
| Teszt időpontja | 2024. 04. 20. 17:33 |

### **8.2.7 Átlépés gázba Mask-kal**

|                 |                     |
|-----------------|---------------------|
| Tesztelő neve   | Vizhányó Miklós     |
| Teszt időpontja | 2024. 04. 20. 17:33 |

### **8.2.8 Takarító átléptetése**

|                 |                     |
|-----------------|---------------------|
| Tesztelő neve   | Vizhányó Miklós     |
| Teszt időpontja | 2024. 04. 20. 17:33 |

### **8.2.9 Oktató találkozása személyekkel és tárgyakkal**

|                          |                                                                                                                              |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <b>Tesztelő neve</b>     | Vizhányó Miklós                                                                                                              |
| <b>Teszt időpontja</b>   | 2024. 04. 20. 17:33                                                                                                          |
| <b>Teszt eredménye</b>   | Sikertelen teszt, nem megegyező expected és output fájl                                                                      |
| <b>Lehetséges hibaok</b> | Hibás bemeneti vagy elvárt fájl. Rag osztály meet metódusa vagy Teacher osztály slip metódusa.                               |
| <b>Változtatások</b>     | Az expected.txt fájlban a peopleInRoom mögött lemaradt egy szóköz, a Game in progress után pedig volt egy felesleges szóköz. |

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 20:03 |

### **8.2.10 Tárgy felvétele és lerakása**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

### **8.2.11 Tárgyak időtelése**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

### **8.2.12 Szobák módosítása**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

### **8.2.13 Gázhoz kapcsolódó tárgyak aktiválása**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

### **8.2.14 Transistor használata**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

### **8.2.15 Bénított hallgató**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

### **8.2.16 Elátkozott szoba**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

### **8.2.17 Teli szoba**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

### **8.2.18 Tanár és logarléc**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

**8.2.19      Cleaner és logarléc**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

**8.2.20      Cleaner átlép cursed szobába**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

**8.2.21      Sikertelen tranzisztor párosítás**

|                          |                                                                                                                                                               |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Tesztelő neve</b>     | Vizhányó Miklós                                                                                                                                               |
| <b>Teszt időpontja</b>   | 2024. 04. 20. 17:33                                                                                                                                           |
| <b>Teszt eredménye</b>   | Sikertelen teszt, nem megegyező expected és output fájl                                                                                                       |
| <b>Lehetséges hibaok</b> | Hibás bemeneti vagy elvárt fájl. Transistor osztály activate metódusa.                                                                                        |
| <b>Változtatások</b>     | A tranzisztor activate metódusában a candidate kiválasztásakor lemaradt az a feltétel, hogy a candidate nem lehet önmaga, ezért volt a pair: t2 NULL helyett. |

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 20:03 |

**8.2.22      Ragacsos szoba**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

**8.2.23      Sikertelen kettéválás és egyesülés**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

**8.2.24      Takarító általi mozgatás**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

**8.2.25      Tárgyak eltűnése és aktivált rongy földön viselkedése**

|                          |                                                                                                                            |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------|
| <b>Tesztelő neve</b>     | Vizhányó Miklós                                                                                                            |
| <b>Teszt időpontja</b>   | 2024. 04. 20. 17:33                                                                                                        |
| <b>Teszt eredménye</b>   | Sikertelen teszt, nem megegyező expected és output fájl.<br>Error: Rooms are not neighbours hibakimenet az output fájlból. |
| <b>Lehetséges hibaok</b> | Hibás bemeneti vagy elvárt fájl vagy Interpreter-ben vagy enter metódusban lehet még hiba.                                 |
| <b>Változtatások</b>     | Input fájl kijavítása, szomszédság megadása a másik irányba is.                                                            |

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 20:03 |

**8.2.26 Ragacsosság és sörösüveg**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

**8.2.27 Hallgató megvédi magát és nyer**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

**8.2.28 Játék állapotai és hamis Logarléc**

|                        |                     |
|------------------------|---------------------|
| <b>Tesztelő neve</b>   | Vizhányó Miklós     |
| <b>Teszt időpontja</b> | 2024. 04. 20. 17:33 |

**8.3 Értékelés**

| <b>Tag neve</b>        | <b>Tag neptun</b> | <b>Munka százalékban</b> | <b>Aláírás</b> |
|------------------------|-------------------|--------------------------|----------------|
| Görömbey László        | I3B5JU            | 20                       |                |
| Héjja Márton           | RECW35            | 20                       |                |
| Sági Péter Pál         | KQF28D            | 20                       |                |
| Tömöri Péter András    | I4RZ0O            | 20                       |                |
| Vizhányó Miklós Ferenc | NVY1AG            | 20                       |                |

**8.4 Napló**

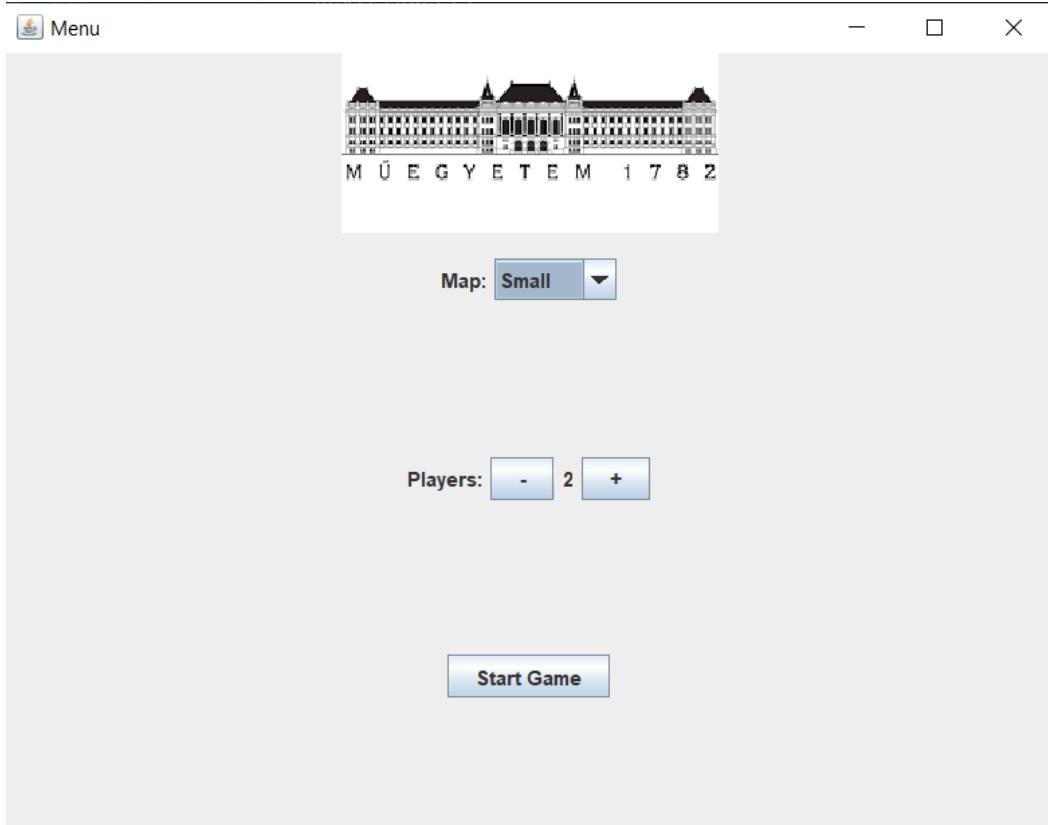
| <b>Kezdet</b>       | <b>Időtartam</b> | <b>Résztvevők</b>                               | <b>Leírás</b>                                                           |
|---------------------|------------------|-------------------------------------------------|-------------------------------------------------------------------------|
| 2024. 04. 17. 10:15 | 1.5 óra          | Görömbey<br>Héjja<br>Sági<br>Tömöri<br>Vizhányó | Konzultáció                                                             |
| 2024. 04. 17. 12:00 | 2 óra            | Görömbey<br>Héjja<br>Sági<br>Tömöri<br>Vizhányó | Feladatok felosztása, határidők egyeztetése, megbeszélés                |
| 2024. 04. 17. 18:00 | 3 óra            | Görömbey                                        | Interpreter vázának megírása a parancsok nélkül                         |
| 2024. 04. 17. 18:00 | 4 óra            | Tömöri                                          | Room, Person és Cleaner osztályok megírása                              |
| 2024. 04. 17. 18:00 | 4 óra            | Vizhányó                                        | TimeSensitive és ItemHandler interfések, Item és IntervalItem megírása  |
| 2024. 04. 17. 18:00 | 4 óra            | Sági                                            | SlideRule, Camembert, AirFresher, TVSZ és Transistor osztályok megírása |

|                     |         |          |                                                                                                               |
|---------------------|---------|----------|---------------------------------------------------------------------------------------------------------------|
| 2024. 04. 18. 16:00 | 3 óra   | Héjja    | Batch fájl megírása                                                                                           |
| 2024. 04. 18. 17:00 | 4 óra   | Görömbey | Interpreter parancsok hozzáadása, végrehajtásuk megírása a status kivételével                                 |
| 2024. 04. 19. 19:30 | 3 óra   | Sági     | Maradék Item-leszármazott osztályok implementálása                                                            |
| 2024. 04. 19 19:30  | 2 óra   | Tömöri   | Student és Teacher osztályok implementálása                                                                   |
| 2024. 04. 19. 22:00 | 1.5 óra | Héjja    | Batch fájl kiegészítése tesztek felsorolásával, tests.txt megírása. Interpreter futtási parancsának javítása. |
| 2024. 04. 20. 11:00 | 2.5 óra | Görömbey | Status parancs implementálása, interpreter működésének ellenőrzése, hibák javítása                            |
| 2024. 04. 20. 14:00 | 1 óra   | Vizhányó | Fordítási és futtatási instrukciók megírása, teszteléshez mappázás előkészítése, javítása a szkriptben        |
| 2024. 04. 20. 17:30 | 3 óra   | Vizhányó | Tesztek futtatása szkripttel. Hibahelyek megkeresése és kijavítása.                                           |
| 2024. 04. 21. 17:00 | 2 óra   | Görömbey | Interpreterhez JavaDoc írása                                                                                  |
| 2024. 04. 21. 17:00 | 2 óra   | Sági     | Új osztályokhoz (Cleaner, AirFresher, FalseXYZ) JavaDoc írása                                                 |
| 2024. 04. 21. 17:00 | 2 óra   | Tömöri   | Meglévő osztályok kommentjeinek frissítése                                                                    |
| 2024. 04. 22. 16:00 | 1.5 óra | Tömöri   | Fájllista megírása                                                                                            |
| 2024. 04. 22. 18:00 | 0.5 óra | Héjja    | Futtatás tesztelése virtuális környezetben                                                                    |

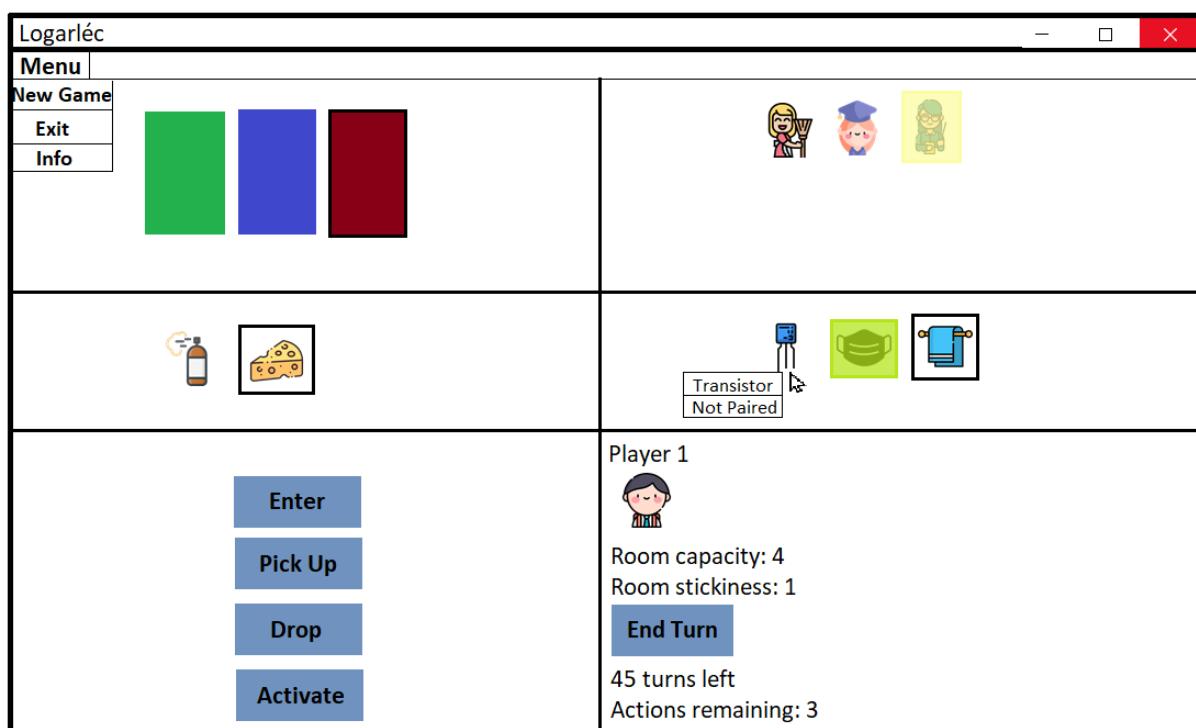
## 9. Grafikus felület specifikációja

### 9.1 A grafikus interfész

#### 9.1.1 Menü nézete



#### 9.1.2 Játék nézete



### 9.1.3 Leírás

A játék elindításakor egy menüvel találkozik a felhasználó. Itt ki tudja választani, hogy milyen pályán szeretne játszani: kicsi, közepes, nagy vagy véletlenszerű. Ezután a játékosszám meghatározására van lehetőség, melynek értéke 1 és 4 közé eső szám. A játék indítására a Start Game gomb megnyomásával van lehetőség. A játék alatt a fenti Menu-nél lehetőség van a játék újraindítására, a játékból való kilépésre, vagy a leírás megtekintésére.

A játék elindulásakor az első játékos nézete kerül megjelenítésre. minden játékosnak meghatározott számú akció végrehajtására van lehetőség egy adott körben (Actions remaining). Ha nem szeretne vagy nem tud már mást csinálni, de még maradt akciója, akkor az End Turn gomb megnyomásával tovább lépteti a játékot a következő játékoshoz. Ha egy adott körben az utolsó játékos is befejezte az akciót, a tanárok és takarítók is lépnek, valamint egységnyi idő telése, szobák egyesülése/osztódása is megtörténik. Ezután a soron következő játékos fog következni. Ha egy játékos meghal, nem fog megjelenni többé. Ha az utolsó játékos is meghal megjelenik egy felugró ablak „BME” felirattal, majd a Menü jelenik meg.

Egy játékos ablakának háttere a szobájának színével egyezik meg, míg a bal felső panelen találhatók a szobából kifelé vezető ajtók, melynek színe a célszobával egyezik meg. Ezekre lehet kattintani így kiválasztva egyet. Ezzel elérhető lesz az Enter gomb, mellyel átlép a kiválasztott szobába. Ez az akció 2 akciópontba kerül. A jobb felső panelen látható minden, a játékossal egy szobában tartózkodó szereplő. Ha egy szereplő ikonja sárgán van kijelölve az azt jelenti, hogy elkábult.

A bal középső panelen a játékos szobájában levő tárgyak találhatók. Egy tárgyra kattintva elérhetővé válik a tárgy felvétele Pick Up gomb megnyomásával (1 akciópont). A tárgy csak akkor vehető fel tényleg, ha a szoba ragacsossága (Room stickiness) nem érte el az 5-t. A jobb középső panelen a játékosnál lévő tárgyak láthatóak. Ezen egy ikonra kattintással elérhetővé válik a Drop és Activate gombok megnyomásával a tárgy eldobása és aktiválása (1-1 akciópont). Ha egy tárgy aktiválva van, akkor zöld színnel van kiemelve. Ha egy tárgy ikonjára húzza az egeret, egy kis ablakban megjelenik a tárgy neve és státusza.

A bal alsó panelen találhatóak a már említett gombok, míg a jobb alsó panelen az eddig említett állapotok mellett a szoba elátkozottságáról, gázosságáról, a játékos kábultságáról és a hátralevő körökről láthat információkat a felhasználó.

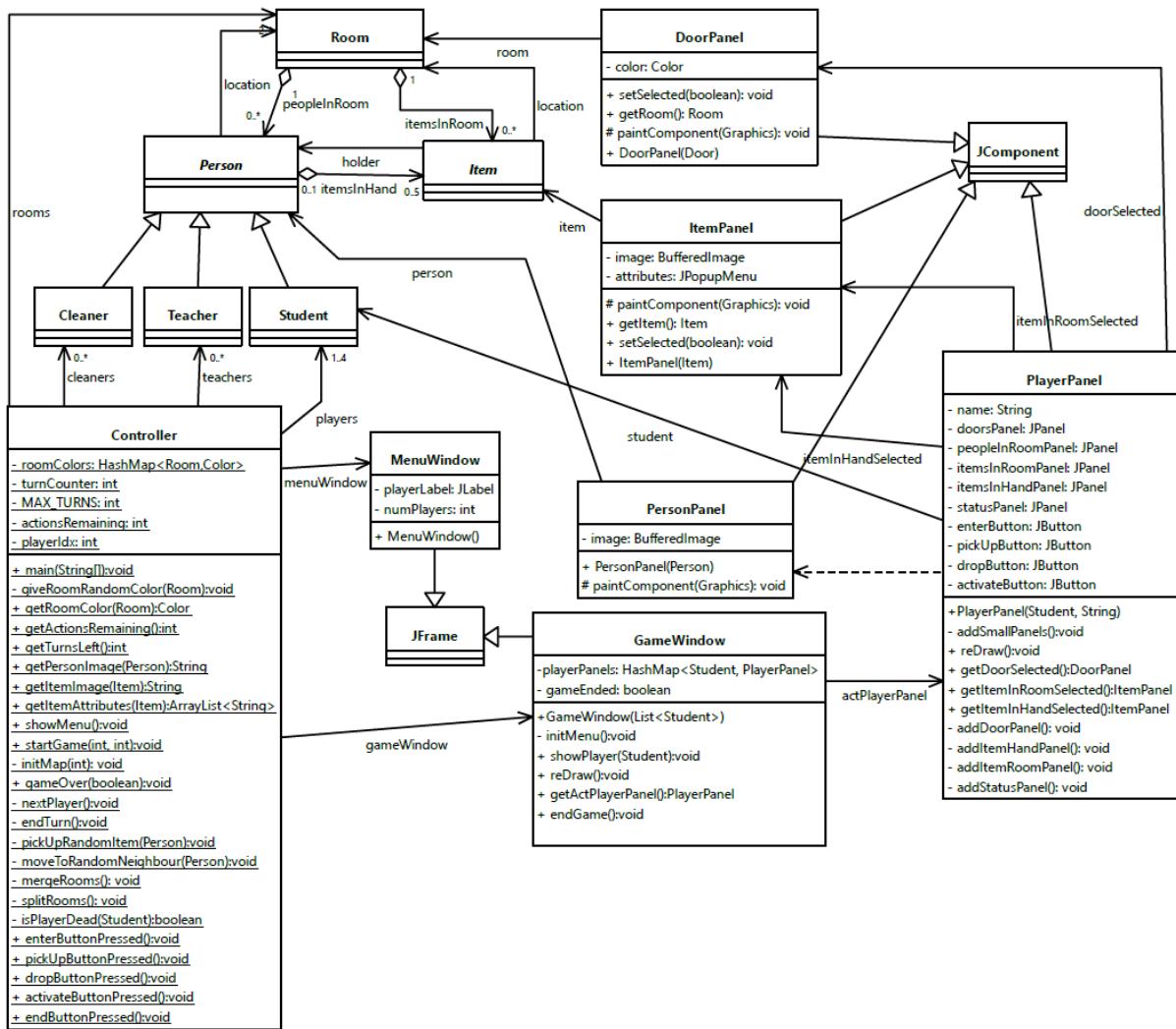
A játék sikerrel ér véget, ha a játékosok valamelyike megtalálja és felveszi a valódi Logarlécet. Ám óvatosan, vannak megtévesztő, hamis tárgyak is, melyekről csak használatkor derül ki mit tudnak valójában. Ha elfogynak a rendelkezésre álló körök, vagy minden játékos kibukik az egyetemről, a játék sikertelenül ér véget, de ne csüggédjen kolléga, rególya bármikor lehet!

## 9.2 A grafikus rendszer architektúrája

### 9.2.1 A felület működési elve

A program grafikus felülettel rendelkező verziójának elkészítéséhez az **MVC** architektúrát vesszük alapul. Ebből kifolyólag a modellt nem kell módosítani a grafikus felület működéséhez. A játék állapotváltozása gombnyomásokra történik. Ezen gombok tevékenysége a **Controller** felelőssége, a modell-en a változást ő idézi elő a rendelkezésre álló metódusokkal. Mivel változás ezen gombok hatására történhet, **pull** alapú grafikus frissítést használunk: minden gombnyomáskor a Controller lekérdezi a modell aktuális állapotát és annak megfelelően jeleníti meg a szükséges panelt. A grafikai megjelenítésért felelős osztályok ismerik a modell-beli objektumuk megfelelőjét, hogy annak logikai állapota alapján jelenjenek meg, de a megjelenítés módját maguk tartalmazzák.

## 9.2.2 A felület osztály-struktúrája



## 9.3 A grafikus objektumok felsorolása

### 9.3.1 Controller

- Felelősség**

A program állapotának számon tartása és vezérlése, a modell és megjelenítés összekapcsolása, a program ablakainak vezérlése és az események kezelése.

- Attribútumok**

- actionsRemaining**: int; soron lévő játékos hátralévő lépéseinek számát tárolja
- cleaners**: List<Cleaner>; a játékban résztvevő takarítókat tárolja
- gameWindow**: GameWindow; a játékkablakot tárolja
- MAX\_TURNS**: int; a játékosok maximális fordulóinak számát tartalmazó konstans.
- menuWindow**: MenuWindow; a menüablakot tárolja
- playerIdx**: int; A soron következő játékos számát tárolja
- players**: List<Student>; a játékosokat reprezentáló hallgatókat tárolja
- roomColors**: HashMap<Room,Color>; a szobák színeit tárolja
- rooms**: List<Room>; a játék szobait tárolja
- teachers**: List<Teacher>; a játékban résztvevő tanárokat tárolja

- **-turnCounter:** int; a fordulók számát tárolja
- **Metódusok**
- **+void main(String[] args):** a menüablakot láthatóra állítja
- **+void activateButtonPressed():** Aktiválja a tárgyat. Kezeli a halált, játék végét, újra rajzol.
- **+void dropButtonPressed():** A tárgy eldobását kezeli.
- **+void endButtonPressed():** nextPlayer-t hív
- **-void endTurn():** Véget vet az adott fordulónak. Növeli a turnCountert és ellenőrzi, hogy kisebb-e még a maximális fordulószámnál. Lépteti a tanárokat és takarítókat. Mulasztja az időt. Ha egy játékos meghalt, eltávolítja. Ha nincs több játékos véget vet a játéknak. Egyesíti, szétválasztja a szobákat.
- **+void enterButtonPressed():** Megpróbálja a játékost beléptetni a választott szobába. Kezeli a halált, újra rajzol.
- **+void gameOver(boolean win):** Véget vet a játéknak. Kiíratja az eredményt, cseréli a játékkalakot menüablakra.
- **+int getActionsRemaining():** visszaadja az aktuális játékos hátralevő lépéseinnek számát.
- **+ArrayList<String> getItemAttributes(Item item):** visszaadja a tárgy attribútumainak listáját szöveges formátumban
- **+String getItemImage(Item item):** az item objektum típusa alapján visszaadja a képének az elérési útját
- **-ArrayList<Room> getMergingRooms():** megpróbál keresni két szobát amelyek megfelelők az egyesítésre. Amennyiben talál visszatér velük, ha nem akkor üres listát ad vissza.
- **+String getPersonImage(Person person):** visszaadja a person objektum típusa szerint a képének az elérési útját
- **+Color getRoomColor(Room room):** visszaadja a paraméterként kapott szoba színét
- **+int getTurnsLeft():** a hátralevő fordulókat adja vissza.
- **-void giveRoomRandomColor(Room room):** generál egy véletlenszerű színt és ezt a színt párosítja a szobához
- **-void initMap(int mapSize):** Térkép létrehozása a megfelelő mapSize alapján. Lehet Small, Medium, Large vagy Random. Nem csak a szobákért és szomszédságukért felel, hanem abba a tárgyak és tanárok, illetve takarítók elhelyezéséért (és az utóbbiak listához adásáért is).
- **-void isPlayerDead(Student player):** Ellenőrzi, hogy él-e még a játékos.
- **-void mergeRooms():** Megpróbál keresni megfelelő szobákat véletlenül majd egyesíteni őket.
- **-void moveToRandomNeighbour(Person person):** átmozgatja egy véletlenszerűen választott szomszédos szobába a személyt.
- **-void nextPlayer():** A következő játékos nézetét jeleníti meg, a hátralevő lépések számát alaphelyzetbe állítja. Ha minden játékos lépett már a fordulóban, akkor meghívja az endTurn függvényt.
- **+void pickUpButtonPressed():** Megpróbálja felvetetni a kiválasztott tárgyat a játékossal. Kezeli a győzelmet.
- **-void pickUpRandomItem(Person person):** felvetet egy tárgyat a szobájából véletlenszerűen a megadott személlyel.
- **+void showMenu():** láthatóvá teszi a menüt ablakát.
- **-void splitRooms():** Megpróbál véletlen szobákat szétválasztani.

- **+void startGame(int mapSize, int playerNumber):** Menüablak eltüntetése, játék inicializálása. PlayerNumber darab játékos létrehozása és elhelyezése a térképen. Játékkép létrehozása.

### 9.3.2 DoorPanel

- **Felelősség**

Az adott szobába vezető ajtó grafikus megjelenítéséért felel.

- **Ősosztályok**

JComponent

- **Attribútumok**

- **-color:** Color; az ajtó színét tárolja
- **-room:** Room; a szobát tárolja, amibe az ajtó vezet

- **Metódusok**

- **+Room getRoom():** visszaadja a room változó értékét
- **#void paintComponent(Graphics g):** betölti az ajtó képet és kitölti a color színnel
- **+void setSelected(boolean selected):** fehér keretet ad az ajtónak, amennyiben ki van választva, egyébként leveszi a keretet

### 9.3.3 GameWindow

- **Felelősség**

Az aktuális játék kirajzolásáért és a menüért felel.

- **Ősosztályok**

JFrame

- **Attribútumok**

- **+actPlayerPanel:** PlayerPanel; az aktuális játékos nézetét jelöli
- **+gameEnded:** boolean; vége van-e a játéknak
- **-playerPanels:** HashMap<Student,PlayerPanel>; a játékosok nézeteit tárolja

- **Metódusok**

- **+GameWindow(List<Student> players):** létrehoz egy GameWindow-ot, amiben a paraméterben kapott hallgatók vezérelhetők.
- **-void initMenu():** inicializálja a menüsávot, a menüt és a menü elemeit: kilépés, új játék, információs ablak.
- **+void reDraw():** újra rajzolja az aktuális játékos nézetét
- **+void showPlayer(Student player):** kicseréli az aktuális játékost a paraméterben kapotttra és újrareajzolatja a nézetét.

### 9.3.4 ItemPanel

- **Felelősség**

Tárgy megjelenítéséért felel.

- **Ősosztályok**

JComponent

- **Attribútumok**

- **-attributes:** JPopupMenu; a tárgy attribútumait tartalmazó felugró menüt tárolja
- **-image:** BufferedImage; a tárgy képét tárolja
- **-item:** Item; a kirajzolt tárgyat tárolja

- **Metódusok**

- **+ItemPanel(Item item):** létrehoz egy ItemPanel-t a paraméterben kapott Item megjelenítésére.
- **+Item getItem():** visszaadja az item-et
- **#void paintComponent(Graphics g):** A tárgy képének kirajzolását végzi. Ha az egy IntervalItem és aktiválva van, akkor zöldre színezi.
- **+void setSelected(boolean selected):** Fekete keretet ad a tárgynak, ha aktiválva van, egyébként ezt leveszi.

### 9.3.5 MenuWindow

- **Felelősség**

Menü ablak megjelenítése

- **Ősosztályok**

JFrame

- **Attribútumok**

- **-numPlayers:** int; játékosok beállított számának tárolása
- **-playerLabel:** JLabel; játékosok beállított számának kijelzhető formában való tárolása

- **Metódusok**

- **+MenuWindow():** létrehozza a menü ablakot és beállítja annak tartalmát: pálya kiválasztás és annak működése, játékosszám kiválasztás és annak működése, játék indítása gomb.

### 9.3.6 PersonPanel

- **Felelősség**

Egy Person példány kirajzolása.

- **Ősosztályok**

JComponent

- **Attribútumok**

- **-image:** BufferedImage; a személyhez megjelenítendő képet tárolja
- **-person:** Person; a megjeleníteni szánt Person objektumot tárolja

- **Metódusok**

- **+PersonPanel(Person person):** létrehoz egy PersonPanel-t a paraméterben kapott személy megjelenítésére.
- **#void paintComponent(Graphics g):** A személy képét rajzolja ki. Ha a person stunRemaining ideje nagyobb, mint 0, akkor sárgára színezi.

### 9.3.7 PlayerPanel

- **Felelősség**

Egy-egy játékos nézetének tárolása, irányítása. Grafikus elemek összefogása, újra rajzolása.

- **Ősosztályok**

JComponent

- **Attribútumok**

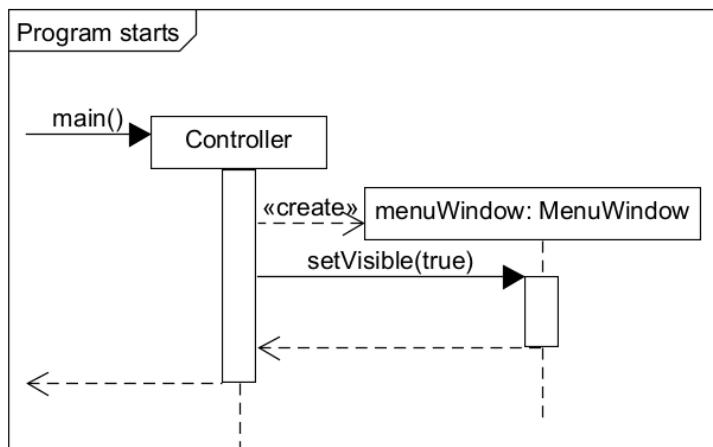
- **-activateButton:** JButton; a „Activate” gomb, amivel a kiválasztott tárgy aktiválható
- **+doorSelected:** DoorPanel; a kiválasztott ajtó panel-ét tárolja
- **-doorsPanel:** JPanel; az ajtókat tartalmazó panel
- **-dropButton:** JButton; az „Drop” gomb, amivel a kiválasztott tárgy dobható el
- **-enterButton:** JButton; az „Enter” gomb, amivel a kiválasztott szobába lehet belelépni
- **+itemInHandSelected:** ItemPanel; a kézből kiválasztott tárgy panel-ét tárolja
- **+itemInRoomSelected:** ItemPanel; a földről kiválasztott tárgy panel-ét tárolja
- **-itemsInHandPanel:** JPanel; a játékos kezében levő tárgyakat tartalmazó panel
- **-itemsInRoomPanel:** JPanel; a játékos szobájában, a földön levő tárgyakat tartalmazó panel
- **-name:** String; a játékos neve
- **-peopleInRoomPanel:** JPanel; a játékossal egy szobában tartózkodókat megjelenítő panel
- **-pickUpButton:** JButton; a „Pick Up” gomb, amivel a kiválasztott tárgy vehető fel
- **-statusPanel:** JPanel; a játékos és a szoba státuszát megjelenítő objektumokat és az „End Game” gombot fogja össze
- **-student:** Student; a játékost képviselő Student példánya

- **Metódusok**
- **+PlayerPanel(Student student, String name)**: Létrehoz egy PlayerPanel-t a paraméterben kapott hallgatónak, inicializálja a paneleket és beállítja a gombokat
- **-void addDoorPanel()**: Hozzáadja a szomszédos szobákat jelképező DoorPanel-eket a doorsPanel-hez és kijelölhetővé teszi őket.
- **-void addItemHandPanel()**: Hozzáadja a játékos kezében lévő tárgyakat jelképező ItemPanel-eket az itemsInHandPanel-hez és kijelölhetővé teszi őket.
- **-void addItemRoomPanel()**: Hozzáadja a szobában lévő tárgyakat jelképező ItemPanel-eket az itemsInRoomPanel-hez és kijelölhetővé teszi őket.
- **-void addSmallPanels()**: Inicializálja a kis paneleket a megfelelő add... metódusok hívásával, továbbá hozzáadja a bent tartózkodó embereket a paneljükhöz.
- **-void addStatusPanel()**: Feltölti információval a statusPanel-t.
- **+void reDraw()**: Eltávolítja a kis panelek tartalmát, és null-ra állítja a kiválasztott tárgyakat, ajtót, tiltja a gombokat. Ezután inicializálja újra a kis paneleket, lekéri a Controller-től a szoba színét és beállítja azt a panelek hátterének.

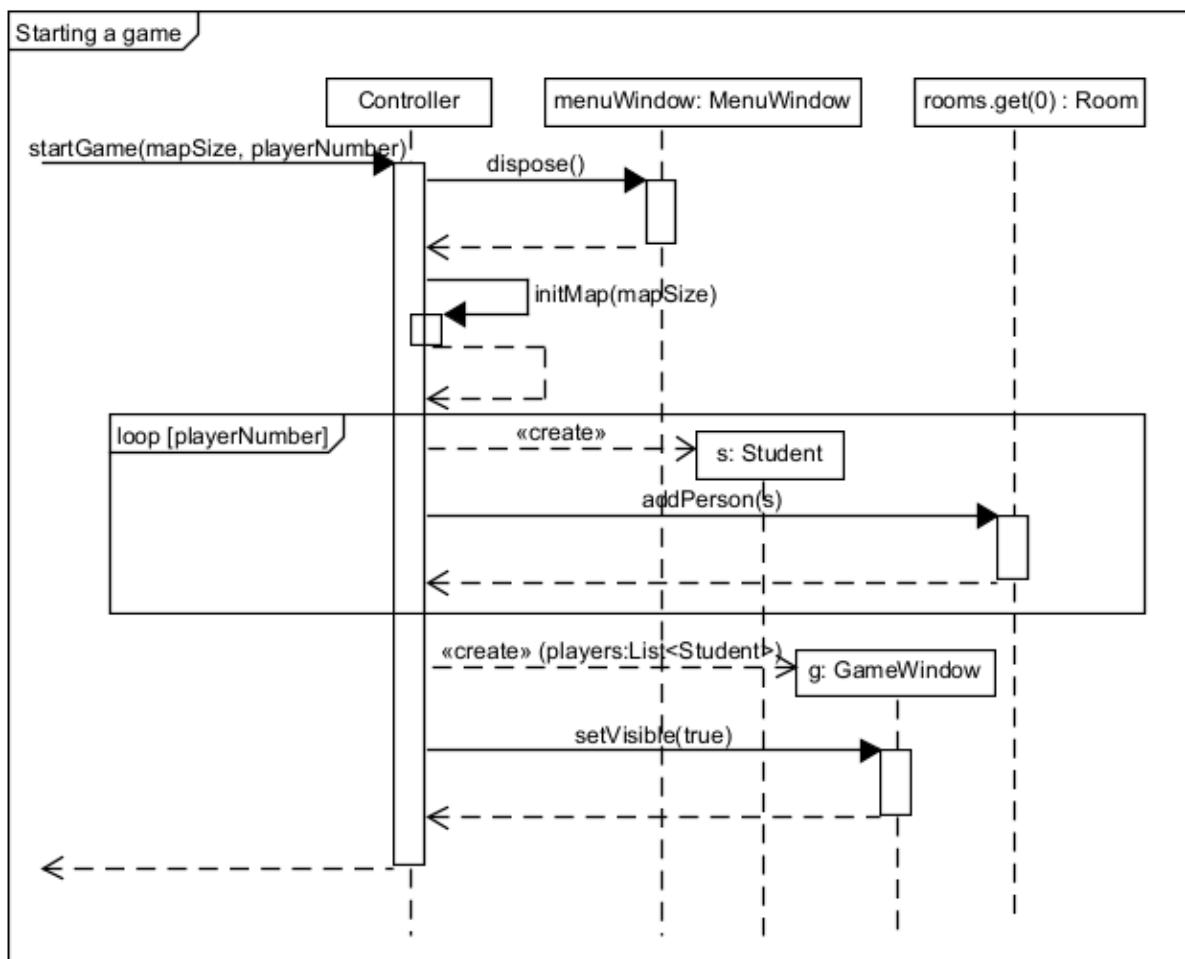
## 9.4 Kapcsolat az alkalmazói rendszerrel

### 9.4.1 Controller

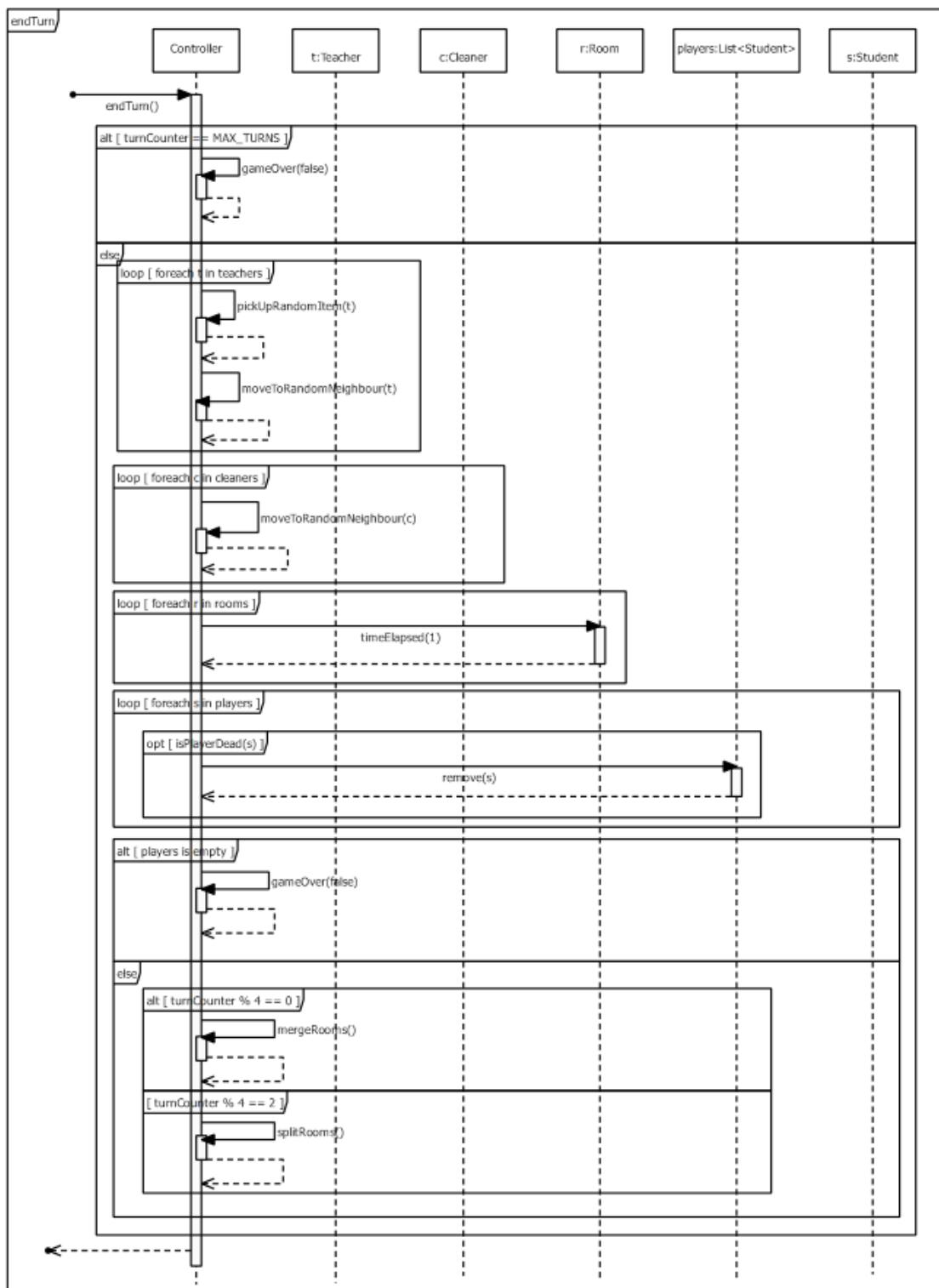
#### 9.4.1.1 Main



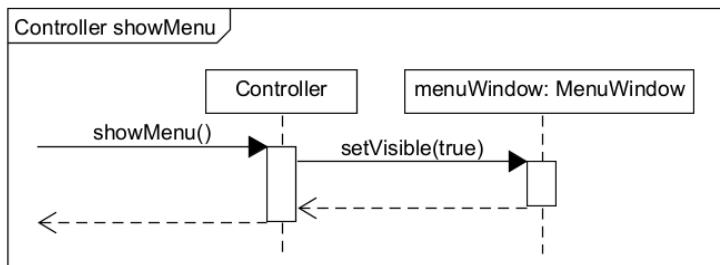
### 9.4.1.2 A játék indítása



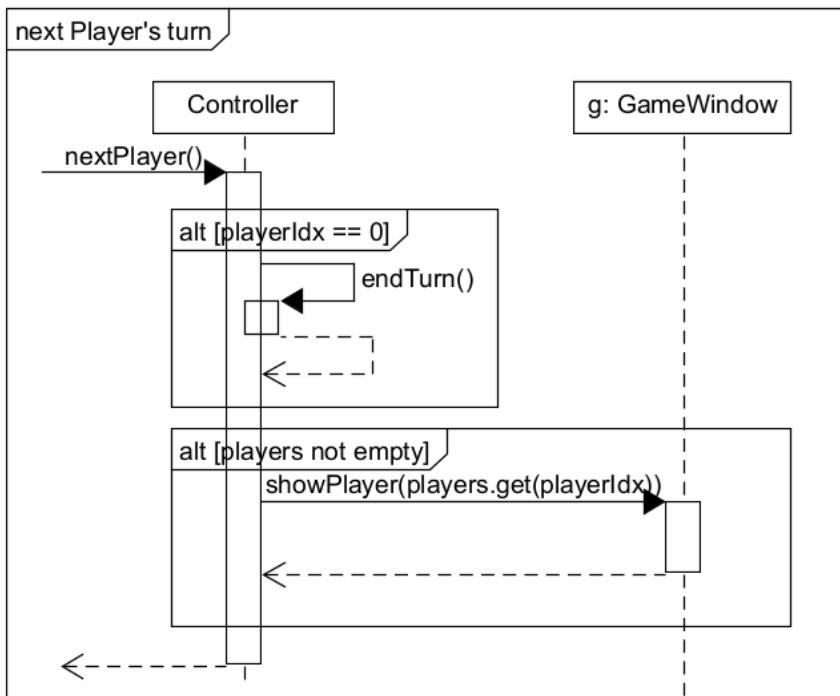
### 9.4.1.3 Kör vége



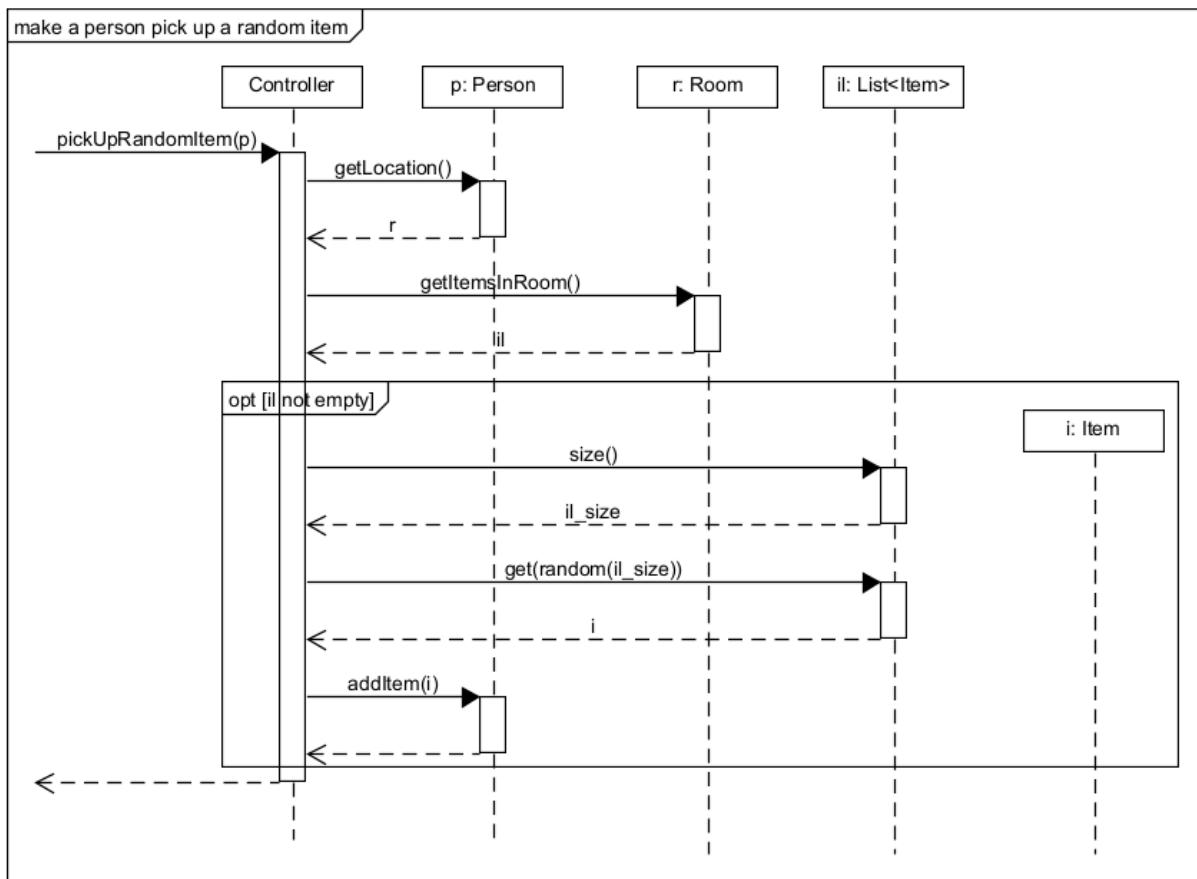
#### 9.4.1.4 A menü megjelenítése



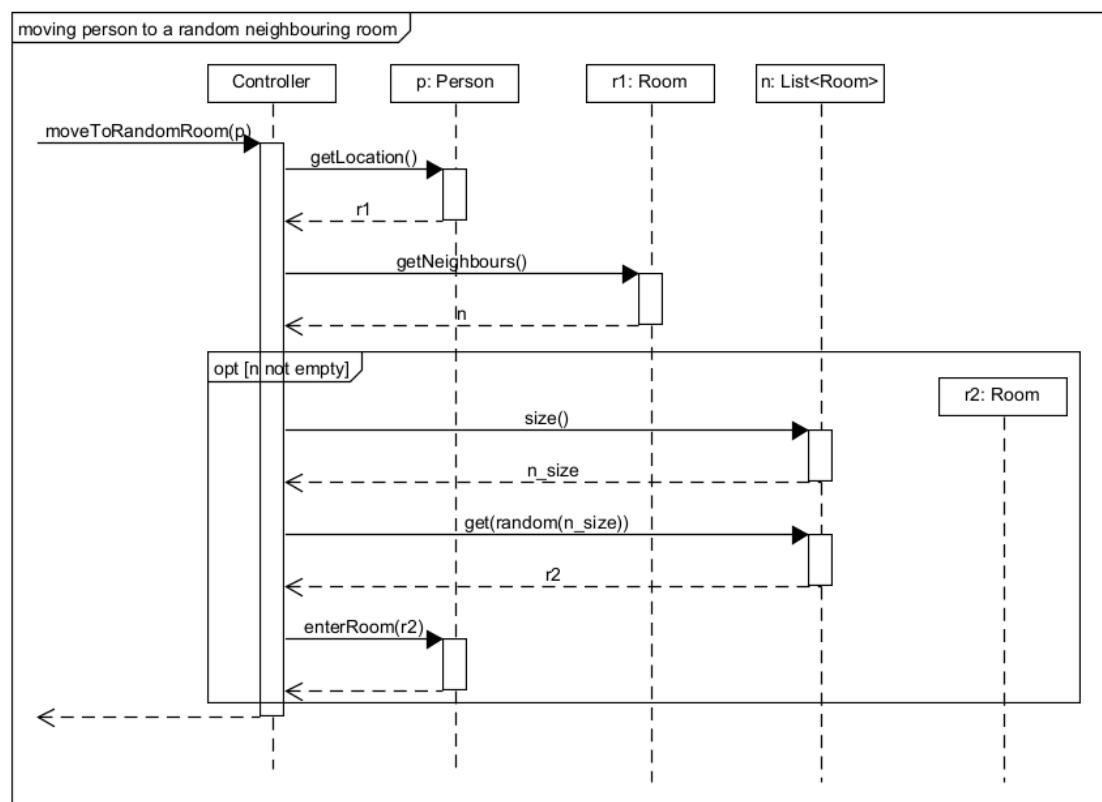
#### 9.4.1.5 Következő játékos



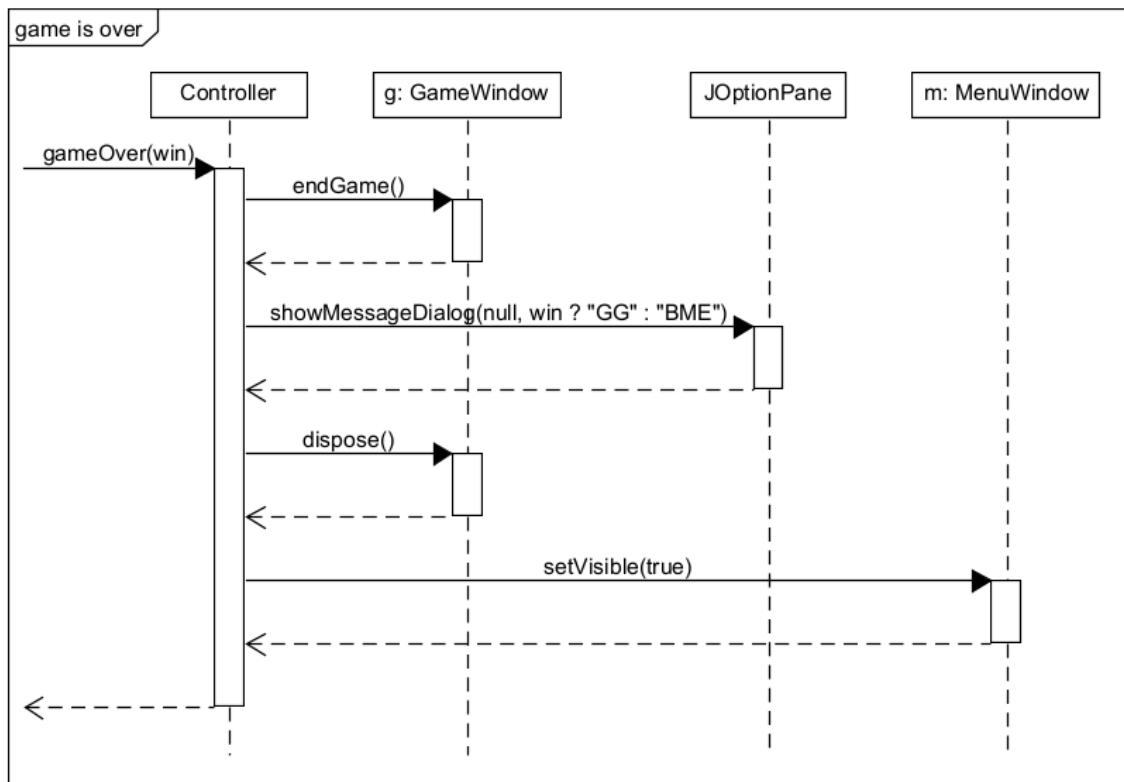
### 9.4.1.6 Véletlenszerű tárgyfelvétel



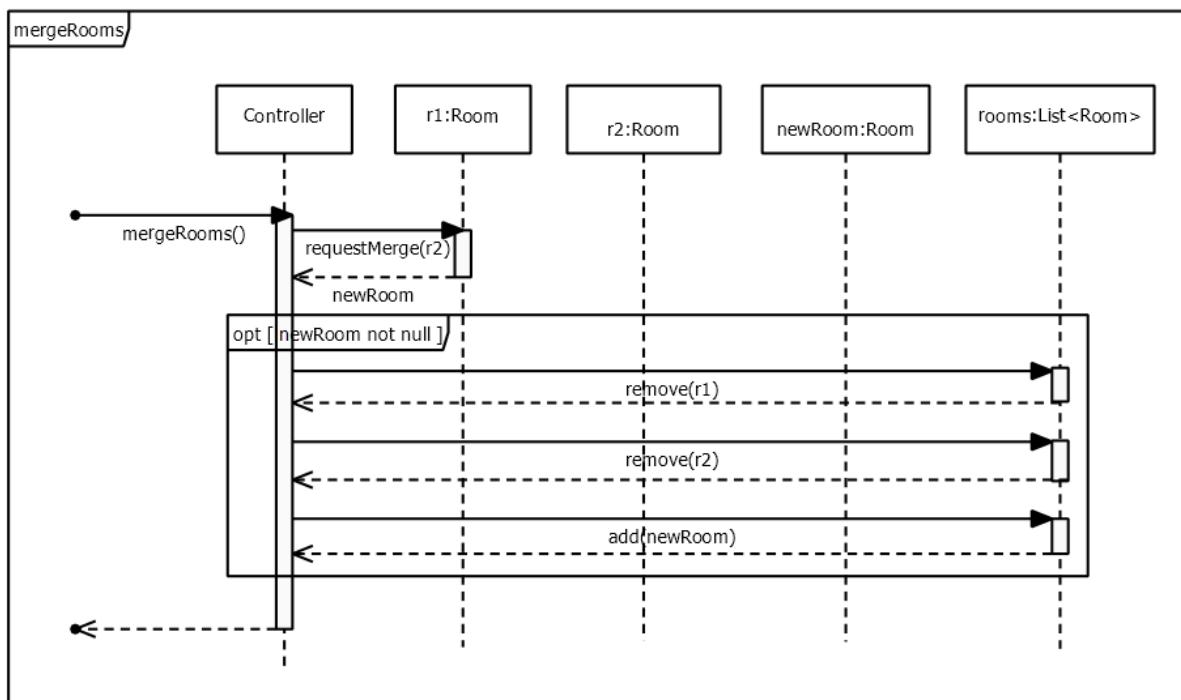
### 9.4.1.7 Véletlenszerű mozgás



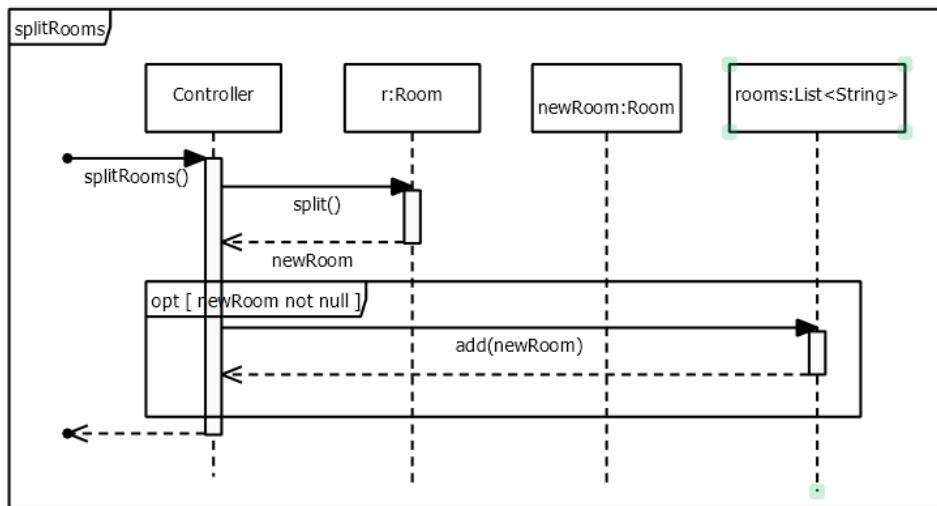
### 9.4.1.8 A játék vége



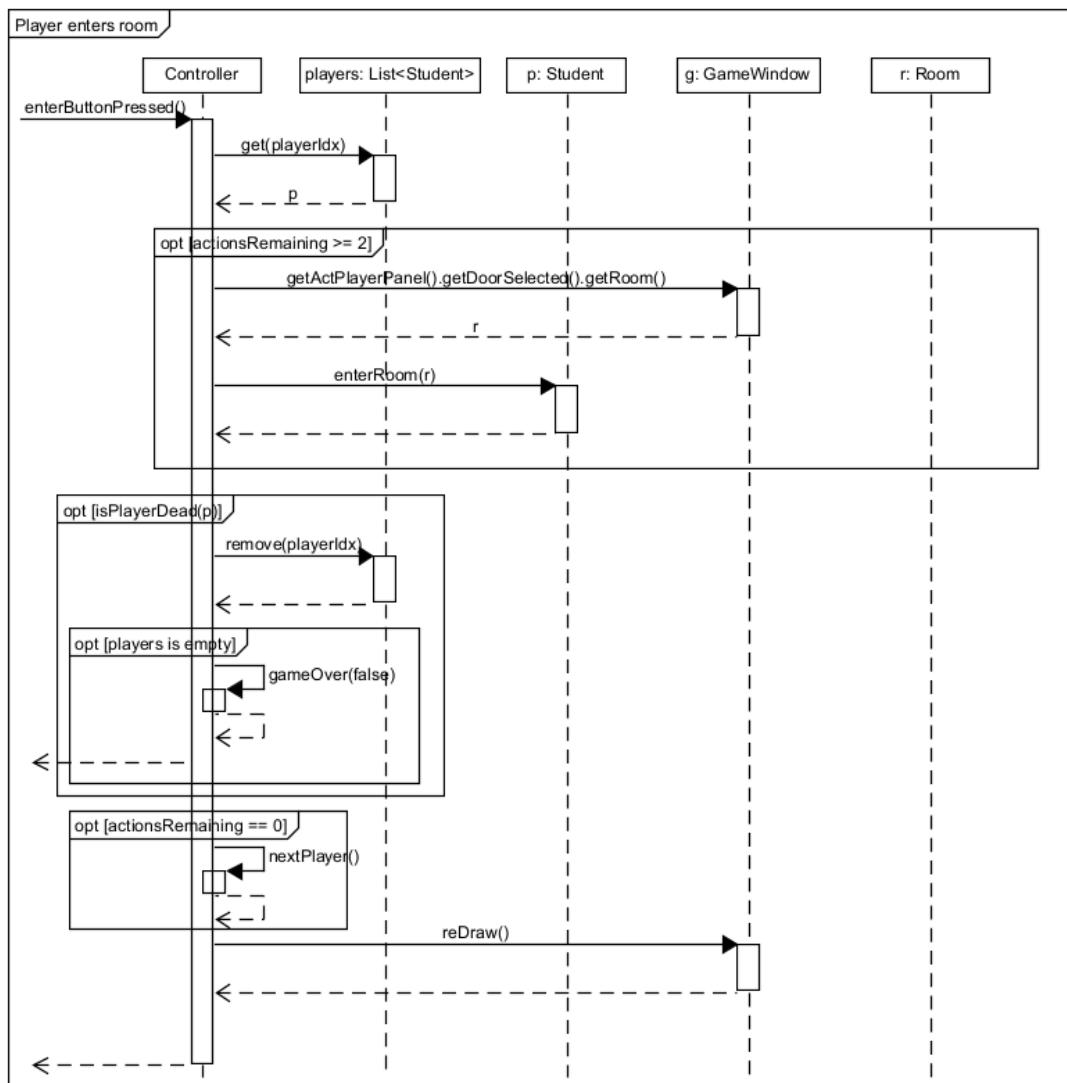
### 9.4.1.9 Szobaegyesítés



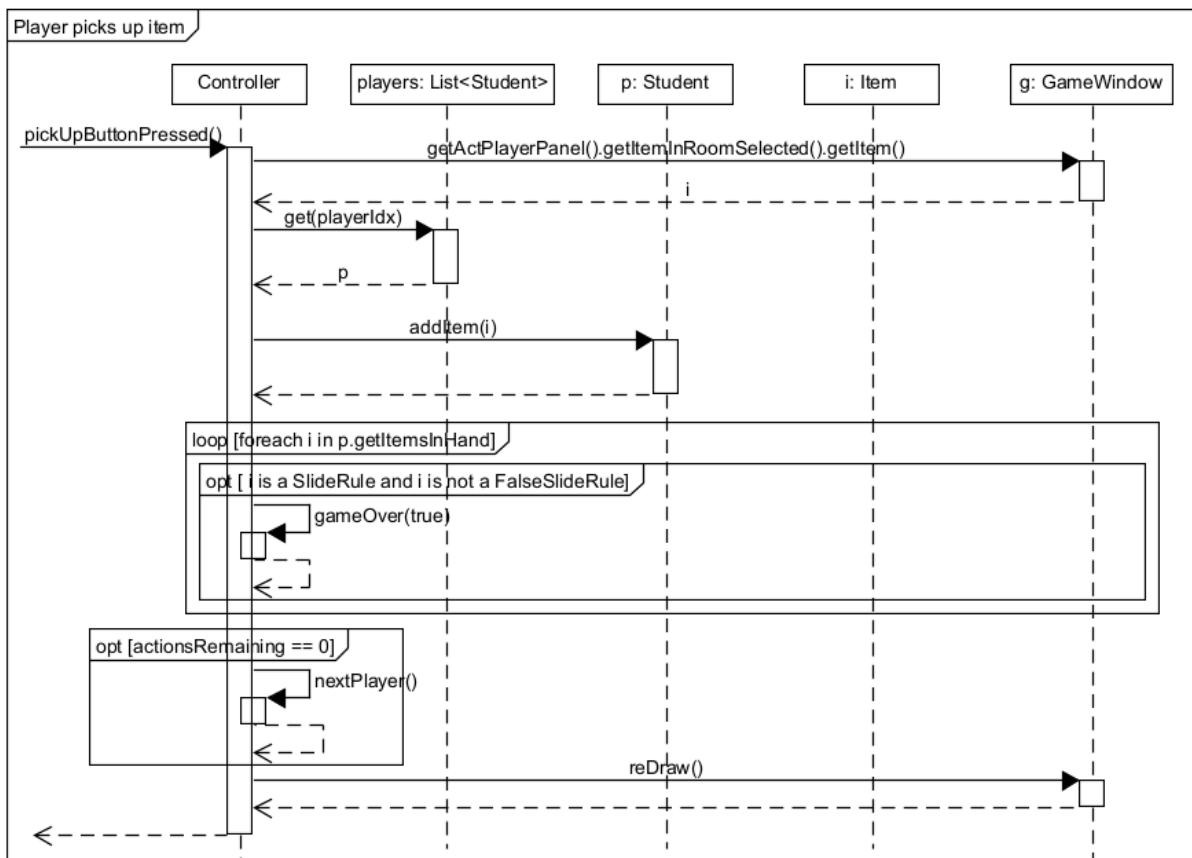
### 9.4.1.10 Szobafelosztás



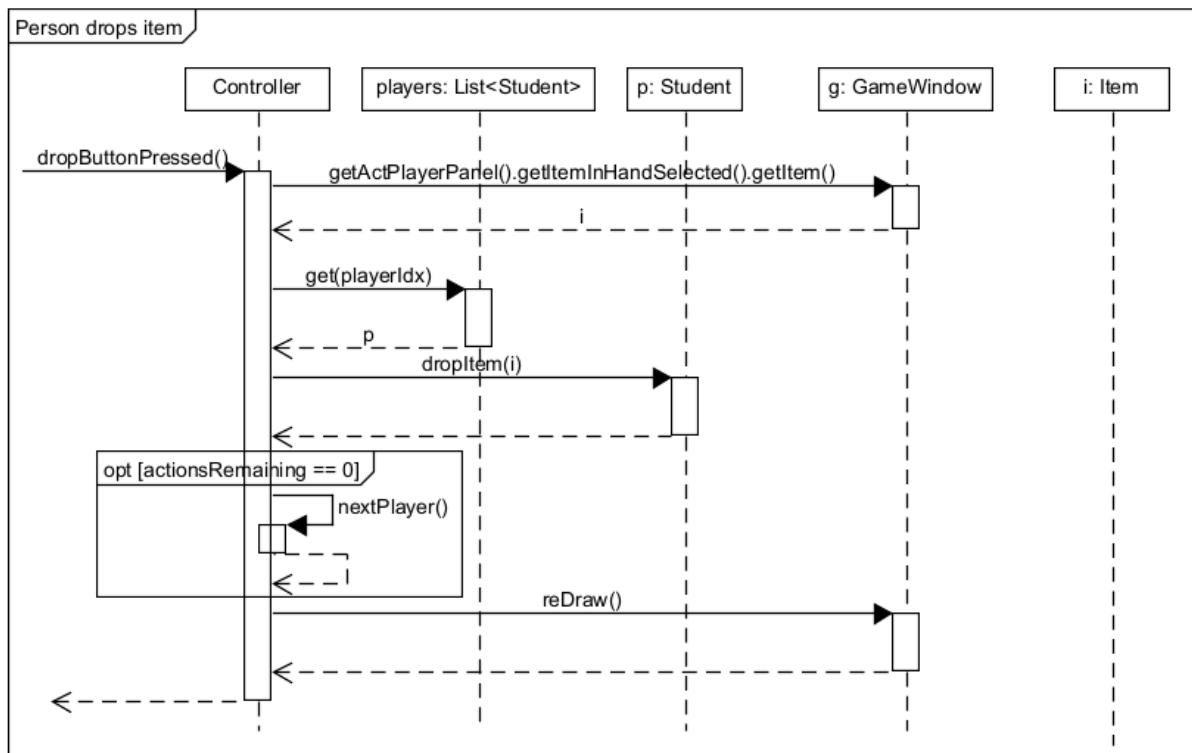
### 9.4.1.11 Az 'Enter' gomb megnyomása



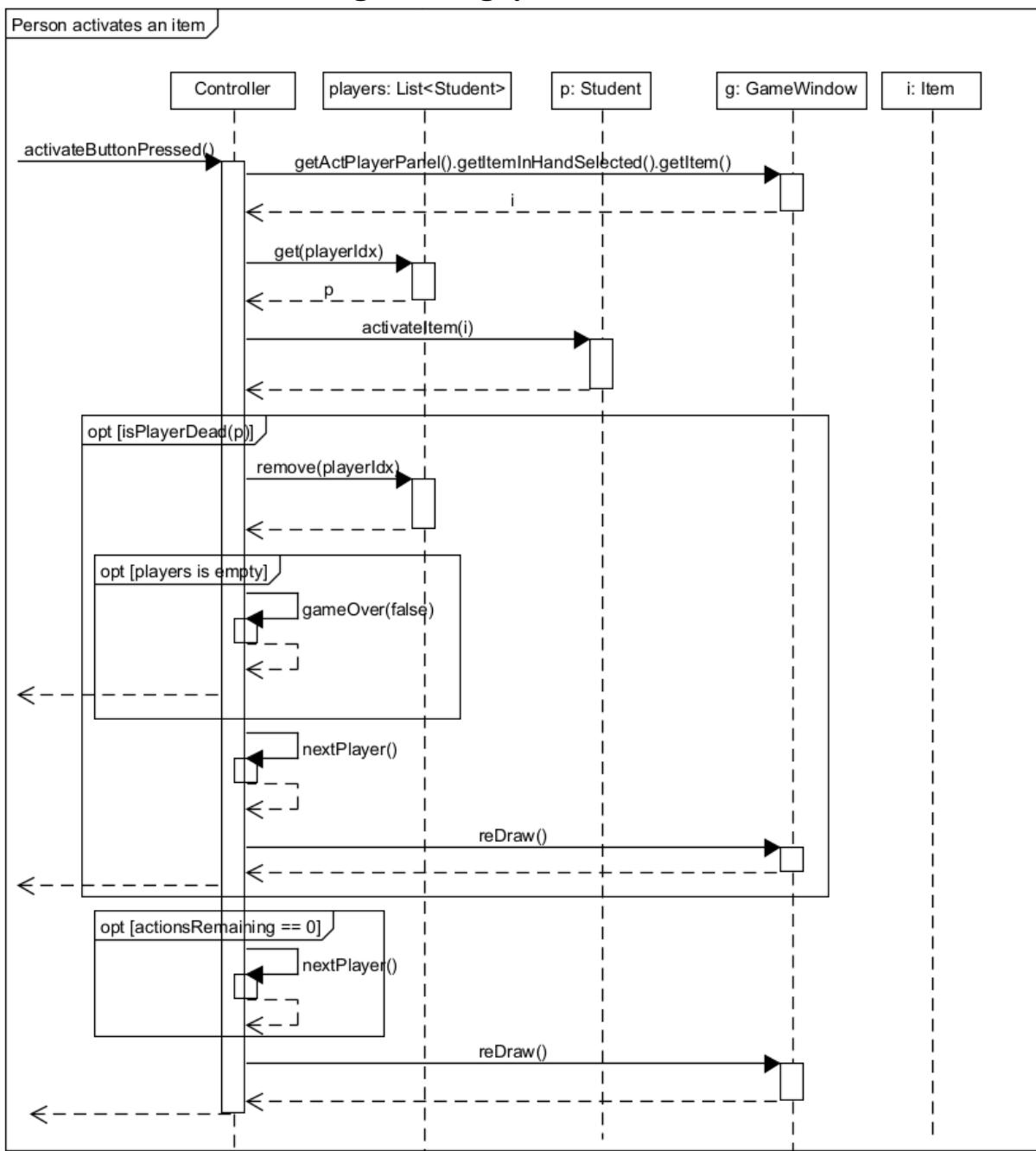
### 9.4.1.12 A 'Pick Up' gomb megnyomása



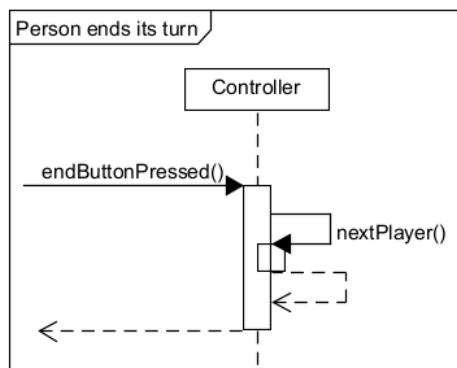
### 9.4.1.13 A 'Drop' gomb megnyomása



### 9.4.1.14 Az 'Activate' gomb megnyomása

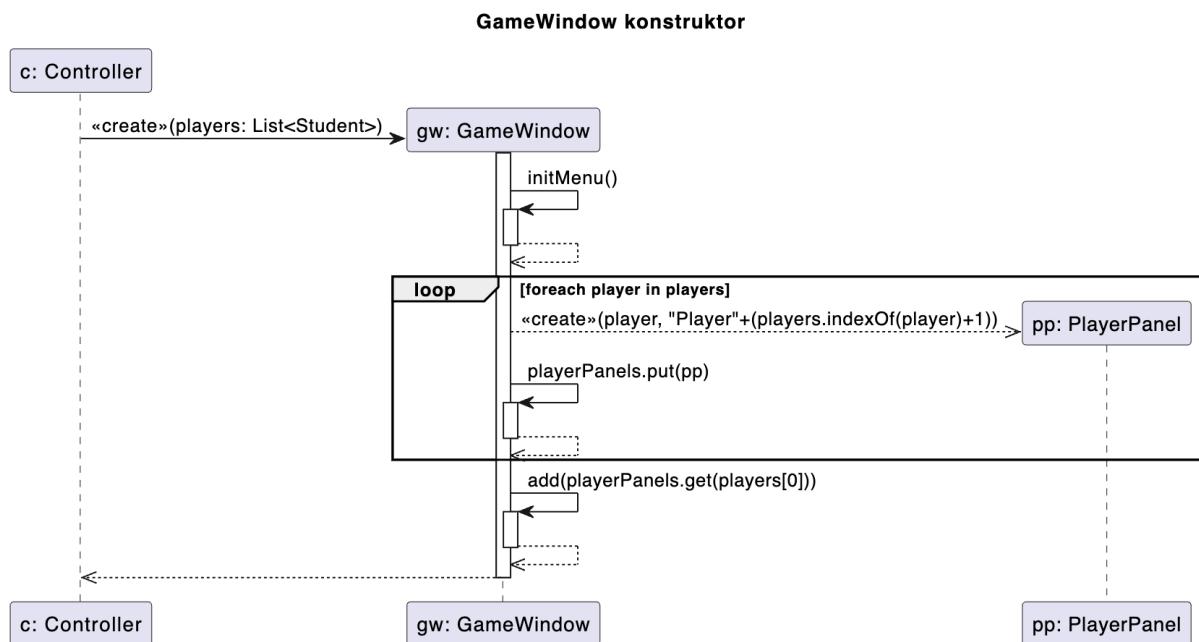


### 9.4.1.15 Az 'End Turn' gomb megnyomása

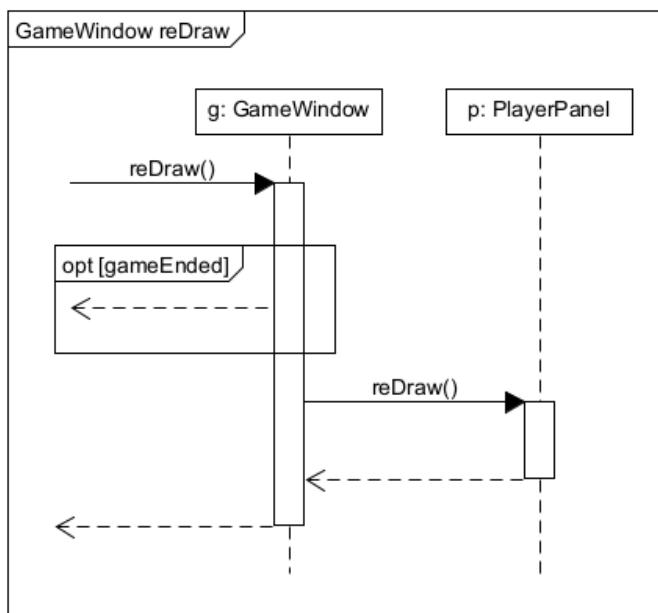


## 9.4.2 GameWindow

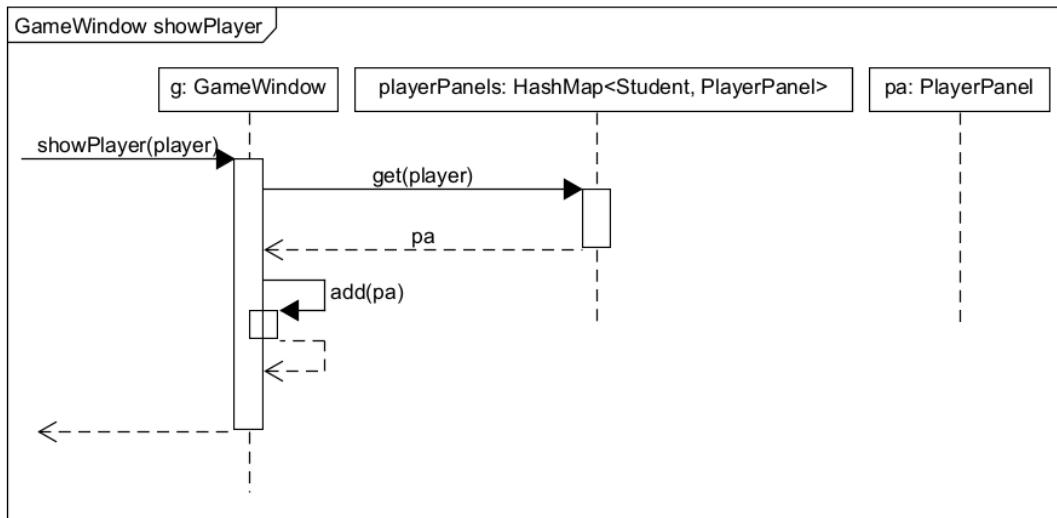
### 9.4.2.1 Konstruktor



### 9.4.2.2 Rajzolás

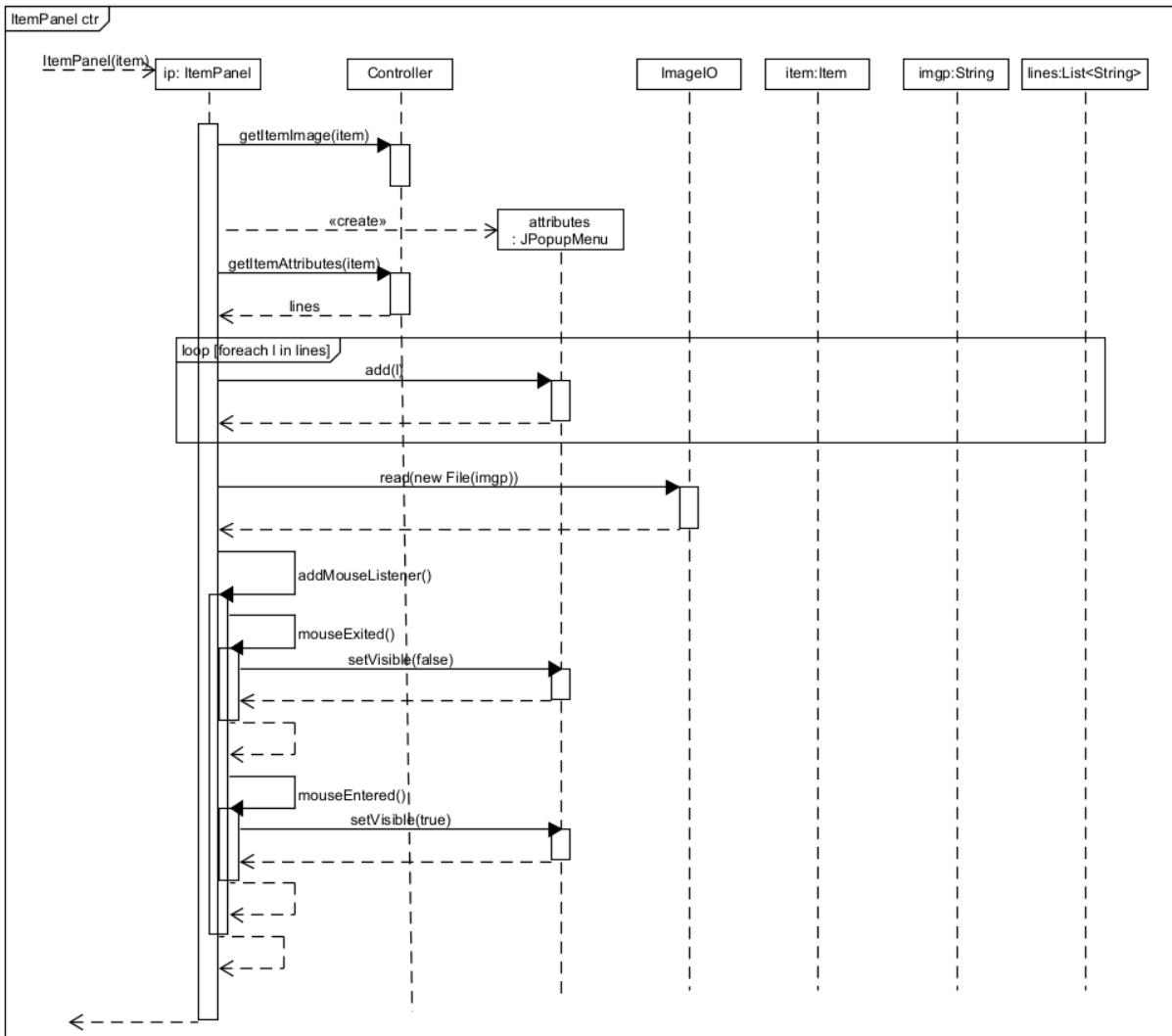


### 9.4.2.3 Játékos megjelenítése



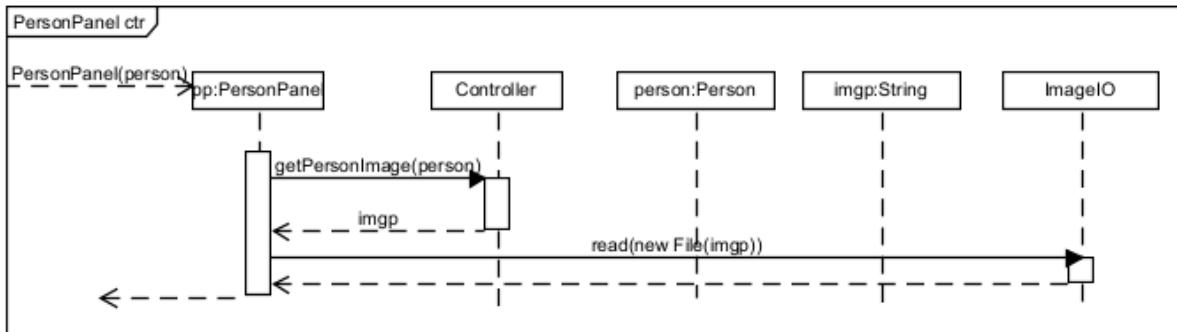
### 9.4.3 ItemPanel

#### 9.4.3.1 Konstruktör



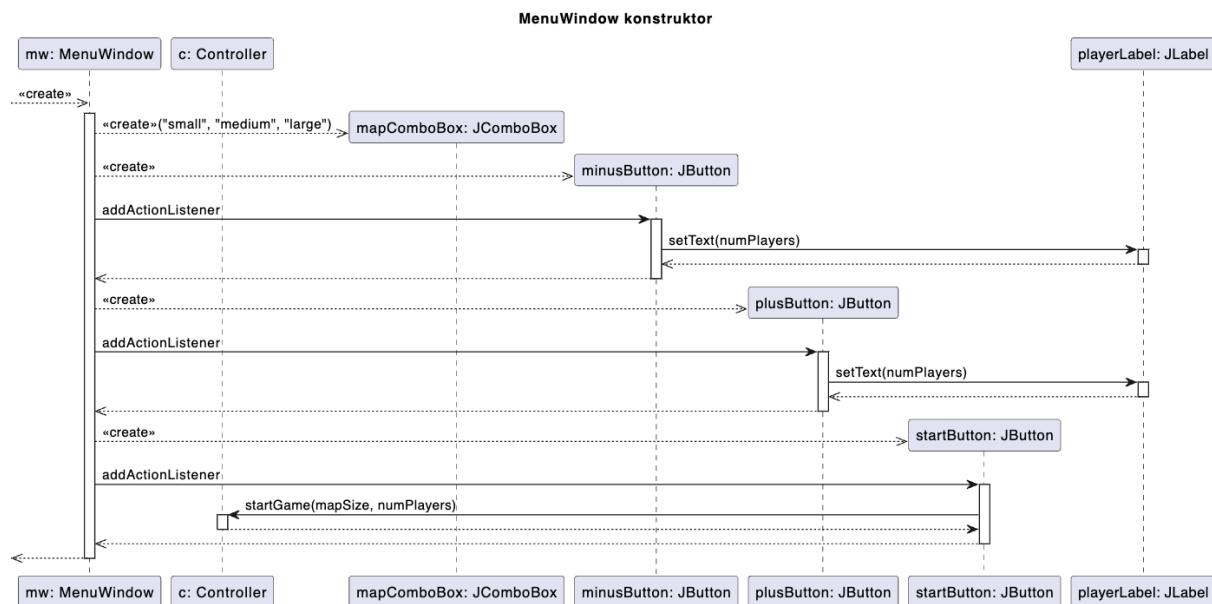
## 9.4.4 PersonPanel

### 9.4.4.1 Konstruktor



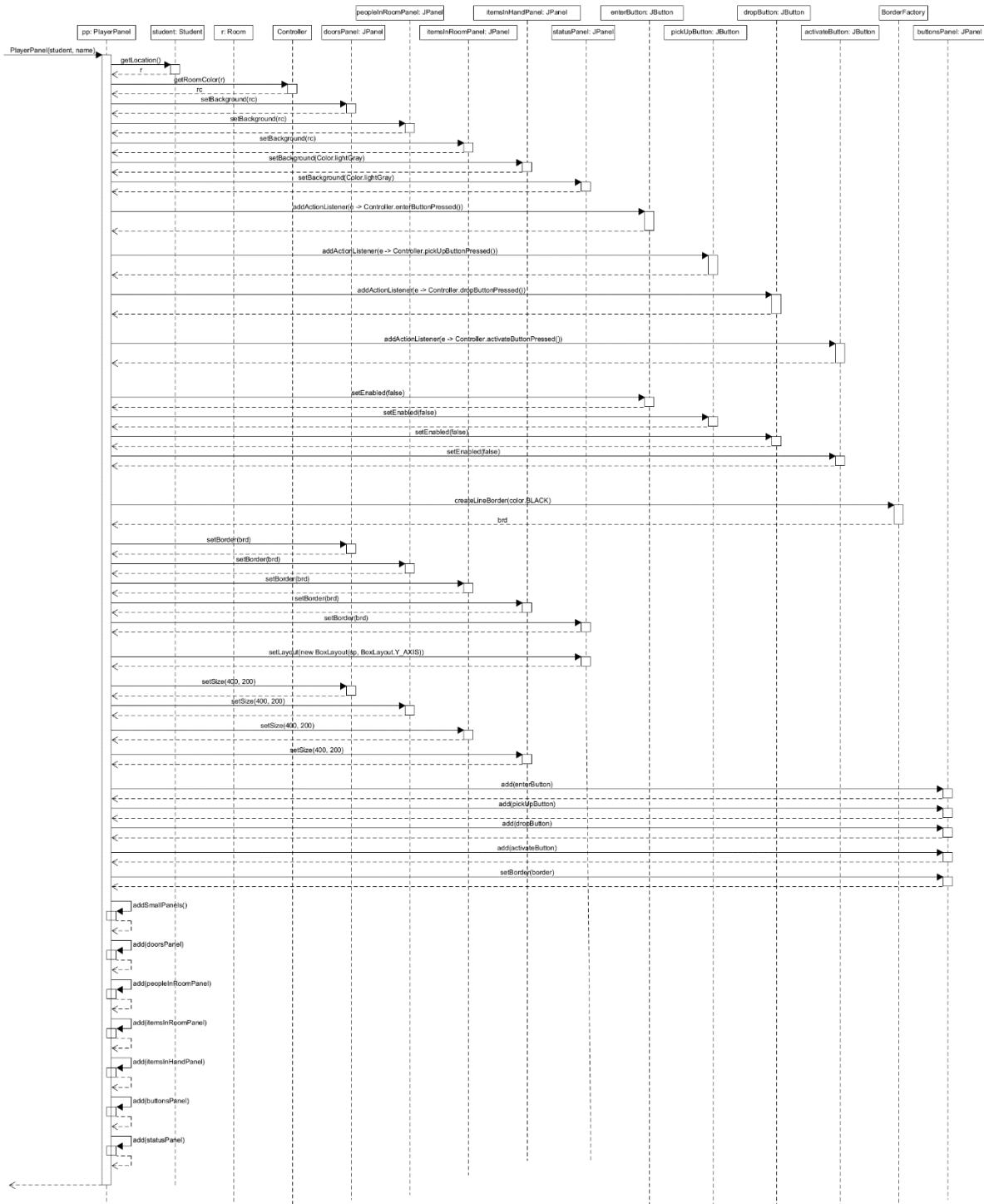
## 9.4.5 MenuWindow

### 9.4.5.1 Konstruktor

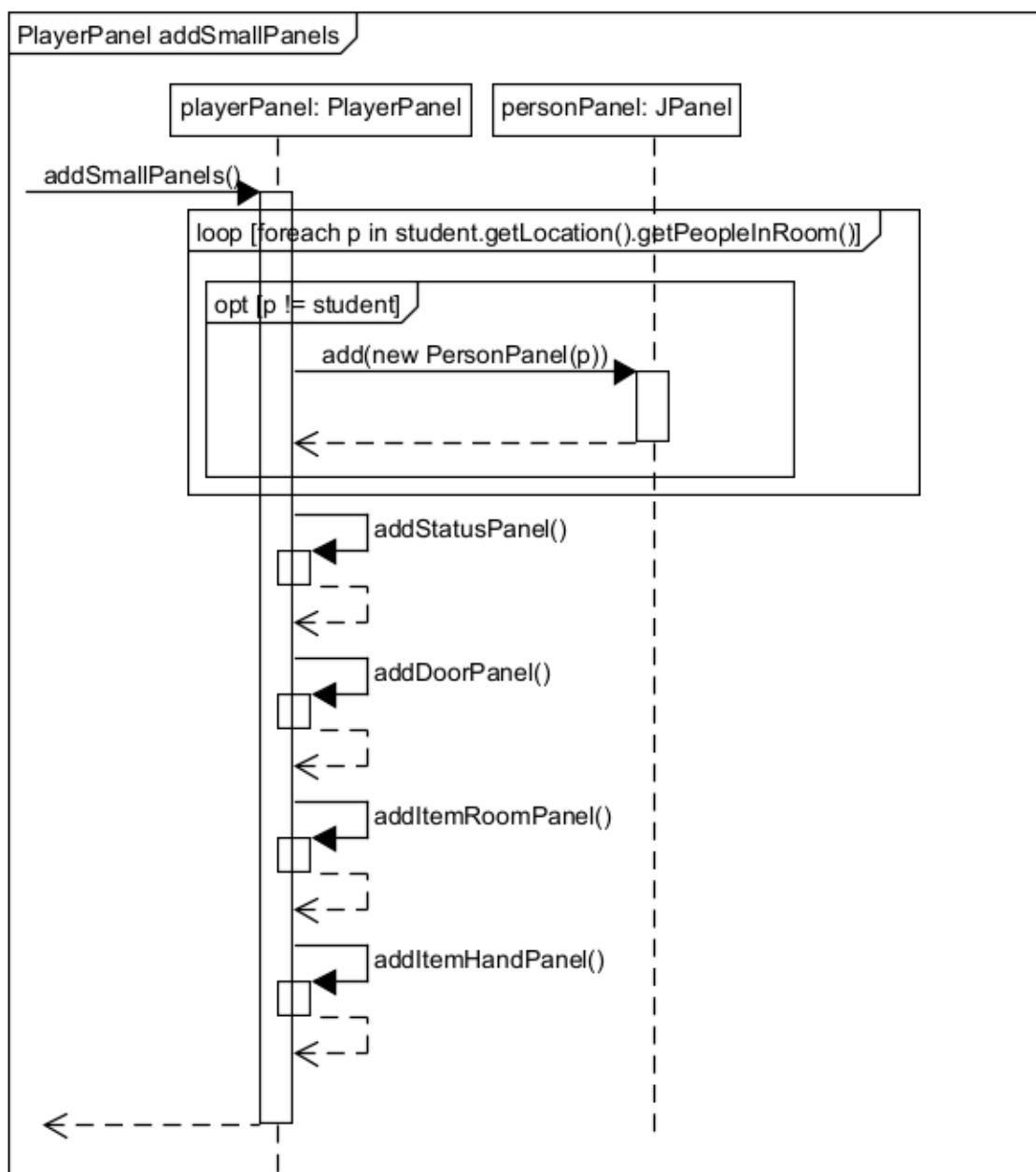


## 9.4.6 PlayerPanel

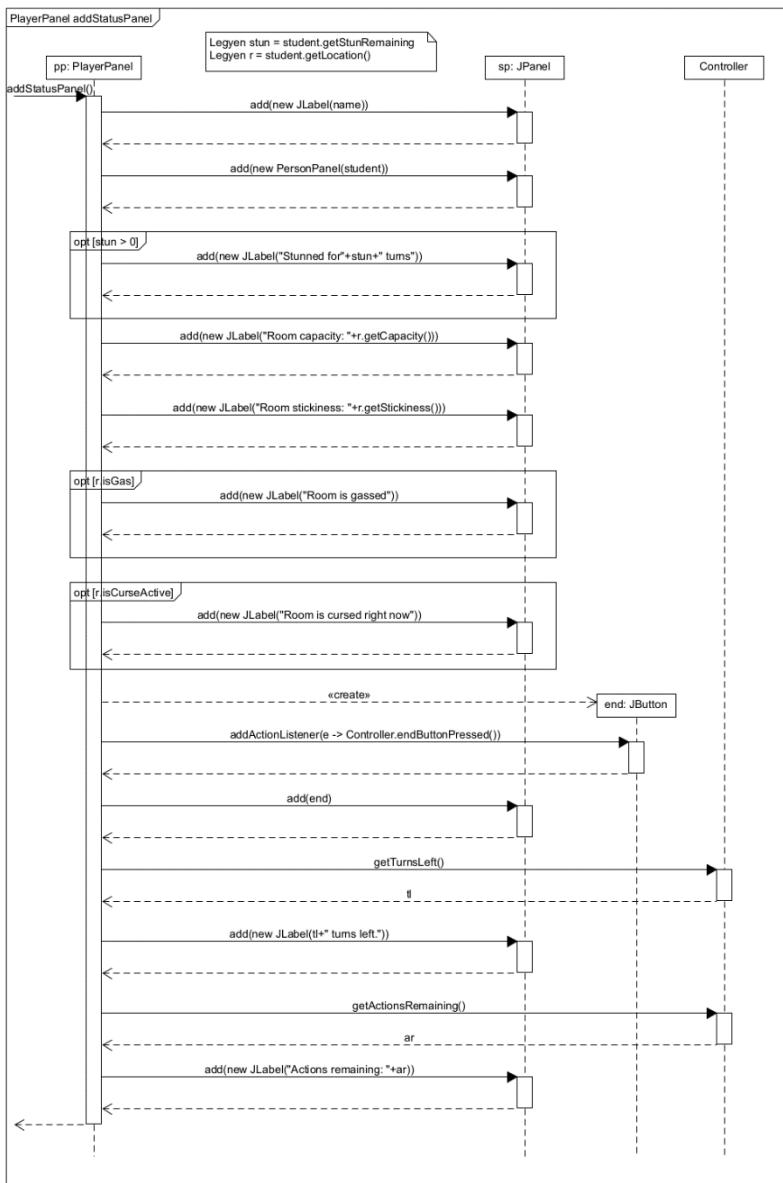
### 9.4.6.1 Konstruktor



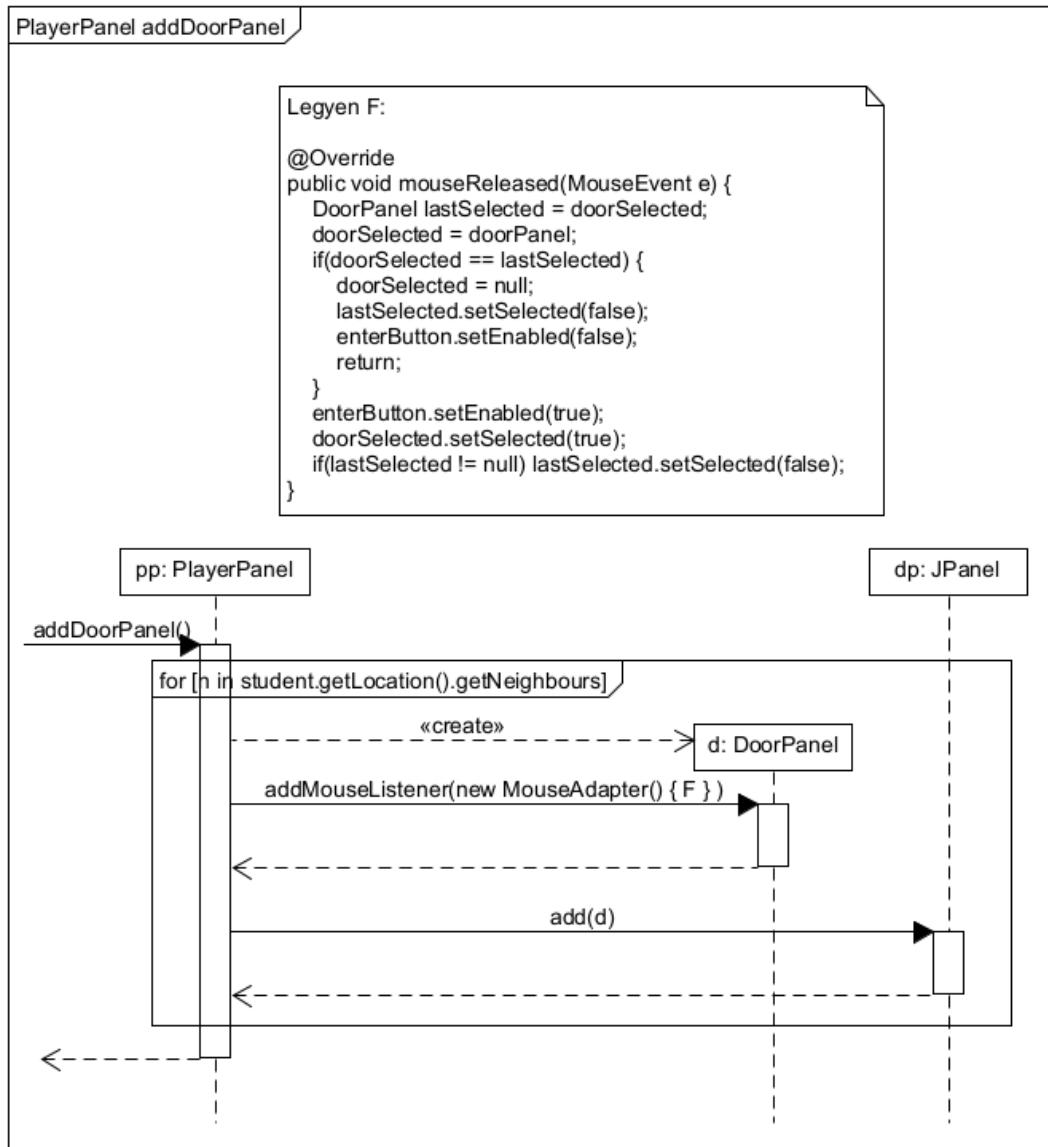
### 9.4.6.2 Kisebb panelek beállítása



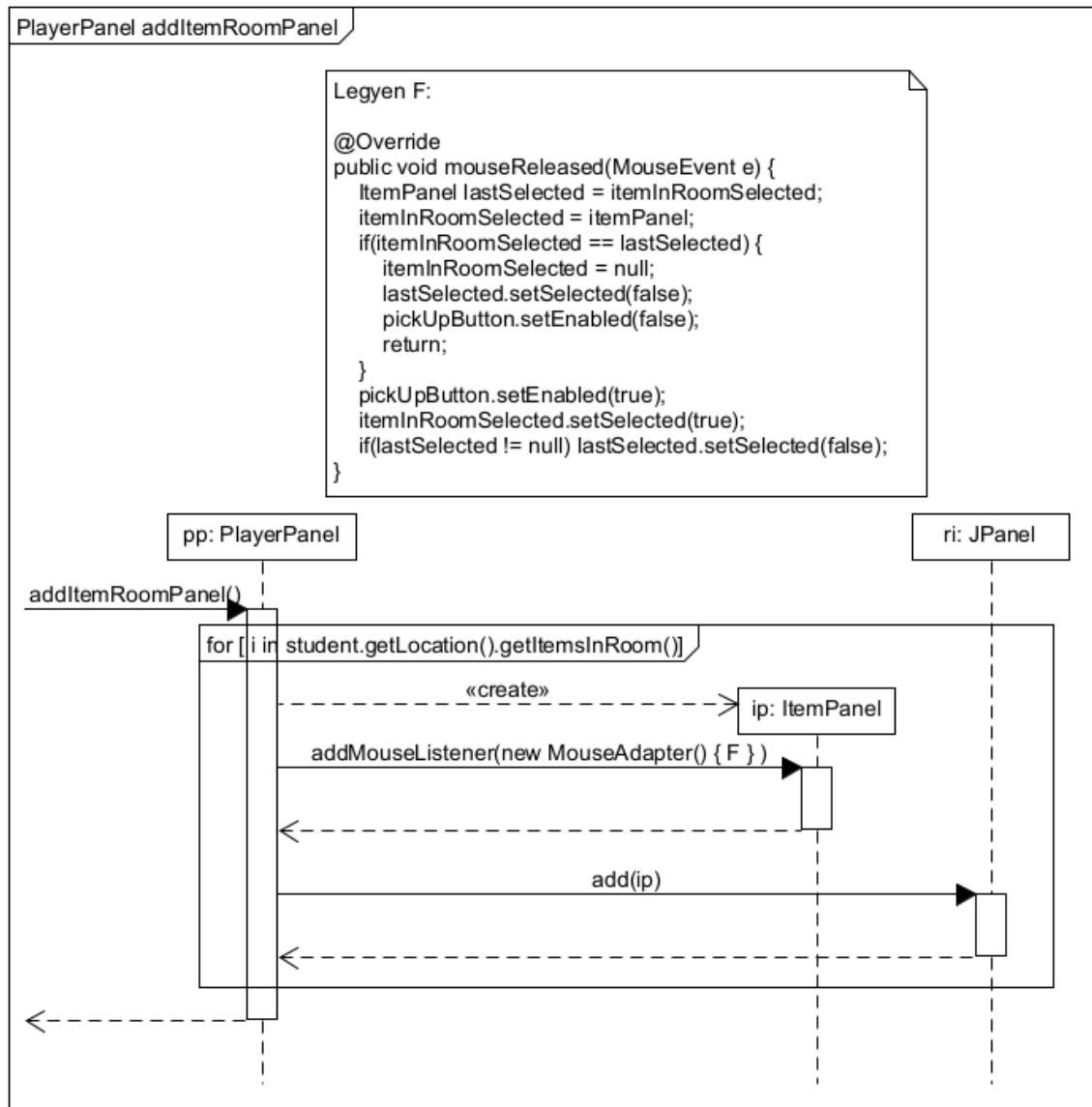
### 9.4.6.3 Státsz panel hozzáadása



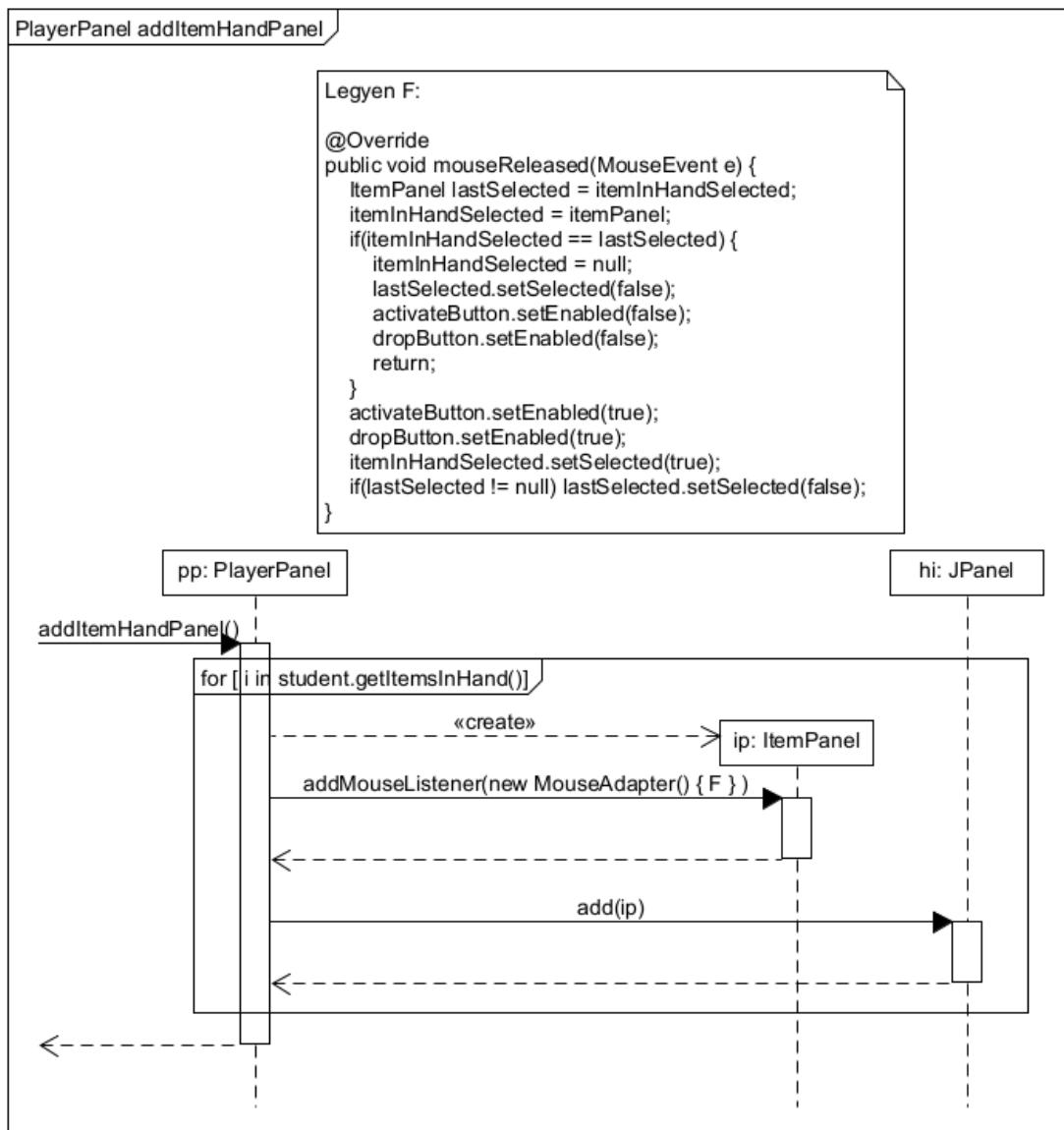
#### 9.4.6.4 Ajtó panel hozzáadása



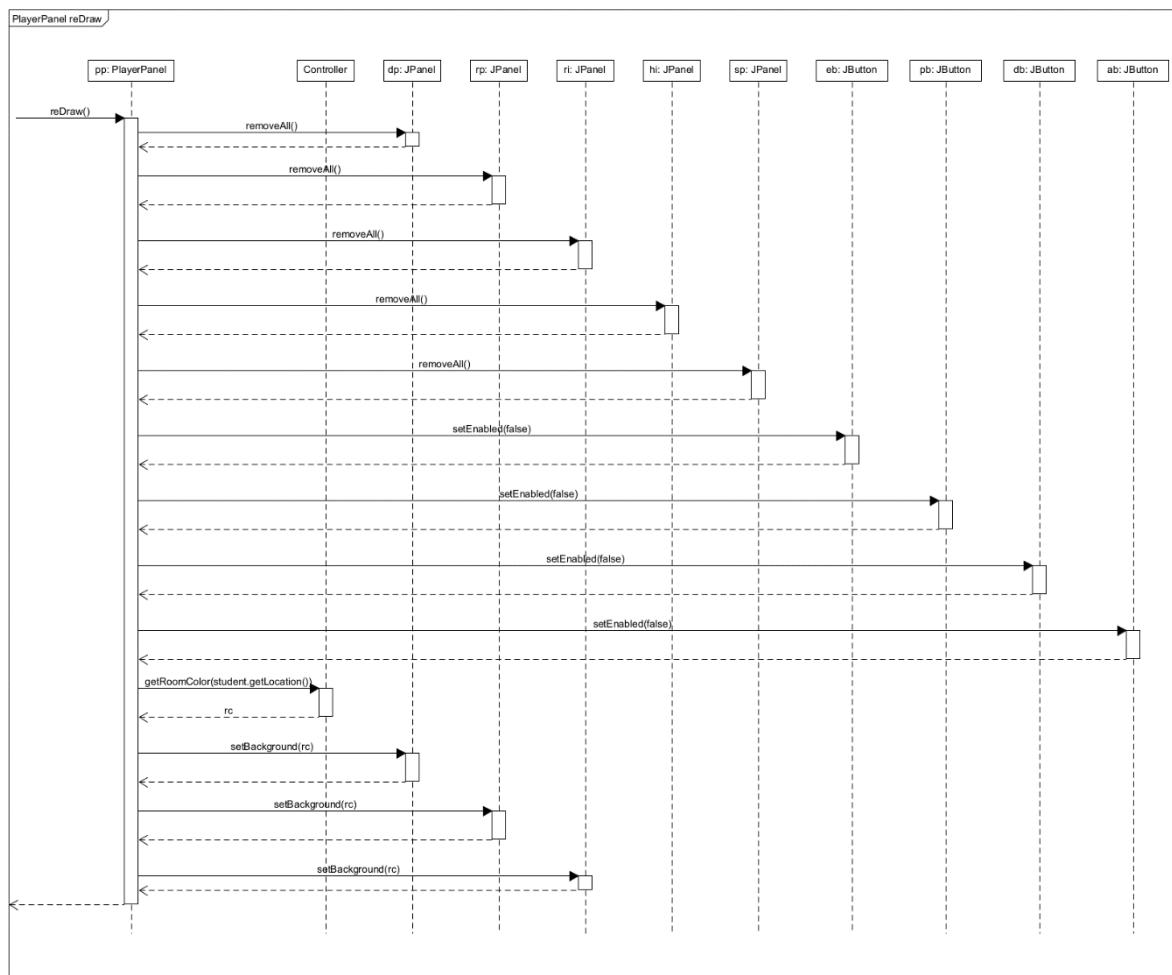
### 9.4.6.5 Szoba tárgy panel hozzáadása



### 9.4.6.6 Kézi tárgy panel hozzáadása

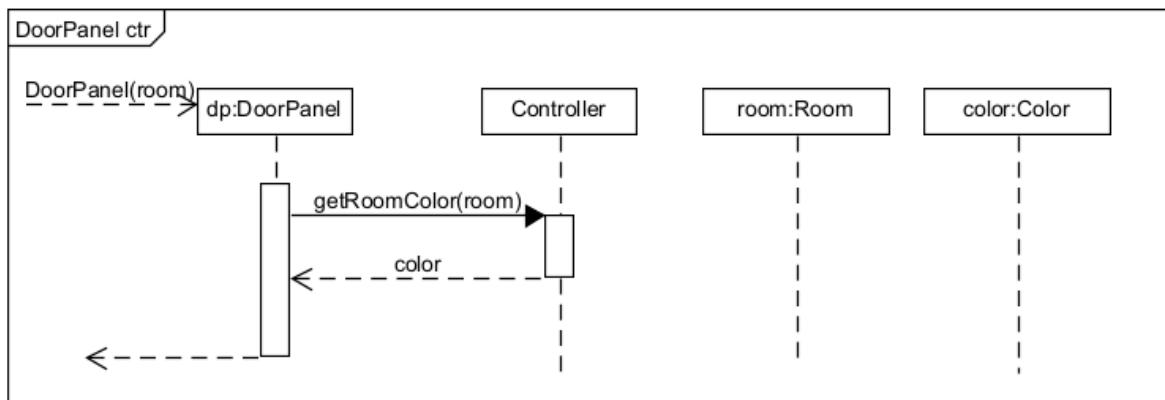


### 9.4.6.7 Rajzolás



### 9.4.7 DoorPanel

#### 9.4.7.1 Konstruktor



## 9.5 Napló

| Kezdet              | Időtartam | Résztvevők                             | Leírás                                                                                          |
|---------------------|-----------|----------------------------------------|-------------------------------------------------------------------------------------------------|
| 2024. 04. 30. 15:00 | 3 óra     | Görömbey<br>Sági<br>Vizhányó<br>Tömöri | Grafikus terv elképzélések egyeztetése, konzultálás                                             |
| 2024. 05. 01. 14:00 | 2 óra     | Tömöri                                 | Nézetek elkészítése                                                                             |
| 2024. 05. 01. 14:00 | 3 óra     | Görömbey                               | Osztálydiagram váz                                                                              |
| 2024. 05. 01. 17:00 | 4 óra     | Vizhányó                               | Osztályleírások váz                                                                             |
| 2024. 05. 01. 17:00 | 3 óra     | Sági                                   | Játék inicializáláshoz szükséges metódusok tervezése                                            |
| 2024. 05. 02. 12:00 | 2 óra     | Görömbey                               | Controller tervezése                                                                            |
| 2024. 05. 02. 12:00 | 2 óra     | Vizhányó                               | GameWindow és PlayerPanel tervezése metódusokkal                                                |
| 2024. 05. 02. 12:00 | 2 óra     | Sági                                   | Item Door és Person Panelek megtervezése                                                        |
| 2024. 05. 02. 12:00 | 2 óra     | Tömöri                                 | MenuWindow megtervezése, Controller kiegészítése                                                |
| 2024. 05. 02. 16:00 | 2 óra     | Tömöri                                 | Leírás elkészítése, dokumentum formázása                                                        |
| 2024. 05. 02. 16:00 | 2 óra     | Vizhányó<br>Sági                       | Metódusok felelősségének csökentése, privát metódusok bevezetése                                |
| 2024. 05. 02. 20:00 | 3 óra     | Görömbey                               | Osztálydiagram elkészítése                                                                      |
| 2024. 05. 05. 00:00 | 1,5 óra   | Héjja                                  | Szekvencia-diagramok kiválasztása, első két diagram elkészítése                                 |
| 2024. 05. 06. 10:00 | 1,5 óra   | Héjja                                  | Szekvencia-diagramok, 5 diagram elkészítése                                                     |
| 2024. 05. 05. 14:00 | 1,5 óra   | Héjja                                  | Szekvencia-diagramok kiválasztása, 10 diagram elkészítése                                       |
| 2024. 05. 05. 18:00 | 1,5 óra   | Héjja                                  | Szekvencia-diagramok kiválasztása, 10 diagram elkészítése                                       |
| 2024. 05. 05. 20:00 | 6 óra     | Sági                                   | ItemPanel, DoorPanel és PersonPanel szekvenciáinak újraírása, Controller refaktorálása          |
| 2024. 05. 05. 20:00 | 6 óra     | Tömöri                                 | Controller szekvenciák javítása, PlayerPanel szekvenciák refaktorálása. Osztálydiagram javítása |
| 2024. 05. 05. 23:00 | 2 óra     | Vizhányó                               | GameWindow szekvenciáinak javítása                                                              |

## 10. Grafikus változat beadása

### 10.1 Fordítási és futtatási útmutató

#### 10.1.1 Fájllista

| Fájl neve           | Méret | Keletkezés ideje    | Tartalom                                                      |
|---------------------|-------|---------------------|---------------------------------------------------------------|
| airfresher.png      | 2974  | 2024. 05. 15. 22:18 | A légrissítő képe.                                            |
| beerglass.png       | 1822  | 2024. 05. 15. 22:18 | A söröspohár képe.                                            |
| bme.png             | 4928  | 2024. 05. 15. 22:18 | A BME logója.                                                 |
| camembert.png       | 3849  | 2024. 05. 15. 22:18 | A Camembert képe.                                             |
| cleaner.png         | 4505  | 2024. 05. 15. 22:18 | A takarító képe.                                              |
| door.png            | 4868  | 2024. 05. 15. 22:18 | Az ajtó képe.                                                 |
| info.txt            | 3583  | 2024. 05. 15. 22:18 | A játék leírása és a hozzá tartozó útmutató.                  |
| mask.png            | 1086  | 2024. 05. 15. 22:18 | A maszk képe.                                                 |
| player1.png         | 3615  | 2024. 05. 15. 22:18 | Az első játékos képe.                                         |
| player2.png         | 3773  | 2024. 05. 15. 22:18 | A második játékos képe.                                       |
| player3.png         | 3583  | 2024. 05. 15. 22:18 | A harmadik játékos képe.                                      |
| player4.png         | 3530  | 2024. 05. 15. 22:18 | A negyedik játékos képe.                                      |
| rag.png             | 1817  | 2024. 05. 15. 22:18 | A rongy képe.                                                 |
| sliderule.png       | 1412  | 2024. 05. 15. 22:18 | A logarléc képe.                                              |
| teacher.png         | 3826  | 2024. 05. 15. 22:18 | A tanár képe.                                                 |
| transistor.png      | 1245  | 2024. 05. 15. 22:18 | A tranzisztor képe.                                           |
| tvsz.png            | 3353  | 2024. 05. 15. 22:18 | A TVSZ képe.                                                  |
| Controller.java     | 30947 | 2024. 05. 15. 22:18 | A játék vezérlésért felelős Controller osztály megvalósítása. |
| AirFresher.java     | 2081  | 2024. 05. 15. 22:18 | A légrissítő játékbeli megvalósítása.                         |
| BeerGlass.java      | 2837  | 2024. 05. 15. 22:18 | A sörösvég játékbeli megvalósítása.                           |
| Camembert.java      | 2022  | 2024. 05. 15. 22:18 | A Camembert játékbeli megvalósítása.                          |
| Cleaner.java        | 2432  | 2024. 05. 15. 22:18 | A takarító játékbeli megvalósítása.                           |
| FalseMask.java      | 1149  | 2024. 05. 15. 22:18 | A hamis maszk játékbeli megvalósítása.                        |
| FalseSlideRule.java | 639   | 2024. 05. 15. 22:18 | A hamis logarléc játékbeli megvalósítása.                     |
| FalseTVSZ.java      | 976   | 2024. 05. 15. 22:18 | A hamis TVSZ játékbeli megvalósítása.                         |
| IntervalItem.java   | 1835  | 2024. 05. 15. 22:18 | Az időzített tárgyakért felelős interfész megvalósítása.      |
| Item.java           | 2851  | 2024. 05. 15. 22:18 | A tárgyak absztrakt játékbeli megvalósítása.                  |
| ItemHandler.java    | 590   | 2024. 05. 15. 22:18 | A tárgyakat hordozó objektumok játékbeli megvalósítása.       |

|                    |       |                     |                                                                          |
|--------------------|-------|---------------------|--------------------------------------------------------------------------|
| Mask.java          | 3712  | 2024. 05. 15. 22:18 | A maszk játékbeli megvalósítása.                                         |
| Person.java        | 7027  | 2024. 05. 15. 22:18 | A személyek absztrakt játékbeli megvalósítása.                           |
| Rag.java           | 2650  | 2024. 05. 15. 22:18 | A rongy játékbeli megvalósítása.                                         |
| Room.java          | 16269 | 2024. 05. 15. 22:18 | A szoba játékbeli megvalósítása.                                         |
| SlideRule.java     | 2309  | 2024. 05. 15. 22:18 | A logarléc játékbeli megvalósítása.                                      |
| Student.java       | 2301  | 2024. 05. 15. 22:18 | A tanuló játékbeli megvalósítása.                                        |
| Teacher.java       | 1867  | 2024. 05. 15. 22:18 | A tanár játékbeli megvalósítása.                                         |
| TimeSensitive.java | 377   | 2024. 05. 15. 22:18 | Az időtelésre reagáló objektumokért felelős interfész megvalósítása.     |
| Transistor.java    | 3999  | 2024. 05. 15. 22:18 | A tranzisztor játékbeli megvalósítása.                                   |
| TVSZ.java          | 2642  | 2024. 05. 15. 22:18 | A TVSZ játékbeli megvalósítása.                                          |
| DoorPanel.java     | 2026  | 2024. 05. 15. 22:18 | Az ajtók megjelenítéséért felelős panel megvalósítása.                   |
| GameWindow.java    | 4728  | 2024. 05. 15. 22:18 | A játékablak megvalósítása.                                              |
| ItemPanel.java     | 3386  | 2024. 05. 15. 22:18 | A tárgyak megjelenítéséért felelős panel megvalósítása.                  |
| MenuWindow.java    | 3788  | 2024. 05. 15. 22:18 | A menüablak megvalósítása.                                               |
| PersonPanel.java   | 1579  | 2024. 05. 15. 22:18 | A személyek megjelenítéséért felelős panel megvalósítása.                |
| PlayerPanel.java   | 12342 | 2024. 05. 15. 22:18 | A játékosok információinak megjelenítéséért felelős panel megvalósítása. |

### 10.1.2 Fordítás és telepítés

A szoftver x86-os architektúrán, Windows operációs rendszer alatt fordítható. Pontosabban a forrásprogramnak a kari felhőben (<https://niif.cloud.bme.hu/>) vagy <https://fured.cloud.bme.hu/>), a „Windows 10 20H2 - JDK-Eclipse-WSU” sablonnal meghatározott környezetben fordítható és futtatható. Ehhez belépési segédlet itt található: <https://www.mit.bme.hu/node/11462>. Egy ilyen típusú virtuális gépre csatlakozás után másolja fel (pl.: vágólapon keresztül) a rendelkezésre álló *grafikus.zip*-ból kicsomagolt *grafikus* mappát.

Megjegyzés: Ha a kicsomagolás során egy új, a zip-pel megegyező mappába teszi a fájlokat, akkor kettő grafikus mappa lesz (az egyikben lesz a másik, ami eredetileg a zip-ben volt). Ebben az esetben, ha *grafikus* mappáról beszélünk, a struktúrában lejjebb lévő mappát értjük.

A kód fordítását a *grafikus* mappában megnyitott cmd shellben teheti meg a következő paranccsal:

```
javac -encoding utf8 -d ./out/ ./src/main/java/model/*.java
./src/main/java/ui/*.java ./src/main/java/controller/*.java
```

Ezután futtathatja a programot.

### 10.1.3 Futtatás

A szoftver x86-os architektúrán, Windows operációs rendszer alatt futtatható. Pontosabban a forrásprogramnak a kari felhőben (<https://niif.cloud.bme.hu/>) vagy (<https://fured.cloud.bme.hu/>), a „Windows 10 20H2 - JDK-Eclipse-WSU” sablonnal meghatározott környezetben fordítható és futtatható. Ehhez belépési segédlet itt található: <https://www.mit.bme.hu/node/11462>. Amennyiben egy ilyen gépen elvégezte az előző pontban (13.1.2 Fordítás) megadott lépésekkel és fordította a programfájlokat, a program futtatható a *grafikus* mappában megnyitott cmd shellben a következő paranccsal:

```
java -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -
Dsun.stderr.encoding=UTF-8 -cp ./out/ controller/Controller
```

Így megjelenik a Menu ablak, melyben be lehet állítani a pálya méretét és a játékosok számát, majd a Start Game gombbal lehet elindítani a játékot.

A programból a Menu menü Exit opcionálisan lehet kilépni.

## 10.2 Értékelés

| Tag neve               | Tag neptun | Munka százalékban |
|------------------------|------------|-------------------|
| Görömbey László        | I3B5JU     |                   |
| Héjja Márton           | RECW35     |                   |
| Sági Péter Pál         | KQF28D     |                   |
| Tömöri Péter András    | I4RZOO     |                   |
| Vizhányó Miklós Ferenc | NVY1AG     |                   |

## 10.3 Napló

| Kezdet              | Időtartam | Résztvevők                     | Leírás                                                                               |
|---------------------|-----------|--------------------------------|--------------------------------------------------------------------------------------|
| 2024. 05. 08. 10:15 | 1.5 óra   | Görömbey<br>Tömöri<br>Vizhányó | Konzultáció és prototípus tesztelése                                                 |
| 2024. 05. 08. 13:00 | 2 óra     | Tömöri                         | MenuWindow elkészítése, info.txt megírása                                            |
| 2024. 05. 08. 13:00 | 3 óra     | Görömbey                       | Controller és a játékvezérléssel kapcsolatos metódusainak elkészítése                |
| 2024. 05. 08. 13:00 | 2 óra     | Sági                           | ItemPanel osztály elkészítése                                                        |
| 2024. 05. 08. 18:00 | 2 óra     | Sági                           | Controller képpel kapcsolatos metódusainak elkészítése, resources mappa elkészítése. |
| 2024. 05. 08. 18:00 | 2 óra     | Tömöri                         | PlayerPanel elkészítése, kijelölhetőség beállítása                                   |
| 2024. 05. 08. 18:00 | 2 óra     | Görömbey                       | GameWindow elkészítése<br>PlayerPanel kiegészítése                                   |
| 2024. 05. 08. 20:00 | 1 óra     | Vizhányó                       | PersonPanel elkészítése                                                              |
| 2024. 05. 13. 16:30 | 1 óra     | Sági                           | DoorPanel elkészítése                                                                |
| 2024. 05. 13. 16:30 | 1.5 óra   | Tömöri                         | Controller kiegészítése split és merge megvalósításával.<br>Small map elkészítése    |
| 2024. 05. 15. 11:00 | 2.5 óra   | Tömöri                         | Controller kiegészítése medium, large és random pályák létrehozásával.               |
| 2024. 05. 15. 14:00 | 2 óra     | Görömbey<br>Sági               | Játék tesztelése, hibák javítása a Controllerben (endTurn, nextPlayer)               |
| 2024. 05. 15. 16:00 | 2.5 óra   | Vizhányó                       | Fordítási és futtatási útmutató megírása, kód kommentezése                           |
| 2024. 05. 15. 19:30 | 3 óra     | Héjja                          | Kommentek szerkesztése, kibővítése                                                   |
| 2024. 05. 21. 10:45 | 1 óra     | Héjja                          | Kommentek ellenőrzése, fájllista kitöltése, kisebb szerkesztések                     |

## 11. Összefoglalás

### 11.1 A projektre fordított összes munkaidő

| Tag neve               | Munkaidő (óra) |
|------------------------|----------------|
| Görömbey László        | 96             |
| Héjja Márton           | 74.5           |
| Sági Péter Pál         | 102.5          |
| Tömöri Péter András    | 134.5          |
| Vizhányó Miklós Ferenc | 108            |
| <b>Összesen</b>        | <b>515.5</b>   |

- A feltöltött programok forrássorainak száma**

| Fázis             | KódSOROK száma |
|-------------------|----------------|
| Szkeleton         | 3174           |
| Prototípus        | 2765           |
| Grafikus változat | 4650           |
| <b>Összesen</b>   | <b>10589</b>   |

### 11.2 Projekt összegzés

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

Konkrét tudás:

- szekvenciadiagramok megfelelő használata.
- előre tervezés
- bővíthetőség, parancsértelmezés, tesztelés

Általában: feladatak megosztása, platformok egyeztetése. Tűrésküszöb megismerése.

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

Legkönnyebb: kód elkészítése a meglévő tervek alapján.

Legnehezebb: felelősségvállalás és időben minőségi munkavégzés, csapattársak motivációjának fenntartása.

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

Ha a feladat nehézségét nézzük akkor nem feltétlenül. A kezdeti specifikációk, tervezés időigényesebb, mint a konkrét program elkészítése. A feladat súlyát tekintve igen, mert a futó program mégiscsak elengedhetetlen.

#### 11.2.4 Ha nem, akkor hol okozott ez nehézséget?

Az analízis modell 1. beadása és a 2. beadása közötti különbség erős volt, valamint a grafikus terv és elkészítés között valamennyi.

### **11.2.5 Milyen változtatási javaslatuk van?**

Az analízis modell pontozása lehangoló. Nem rendelkeztünk a megfelelő alappal ahhoz, hogy az elvárásoknak megfelelő dokumentumunk legyen, valamint az elvárások is túl magasan voltak ahhoz, képest, hogy van lehetőség újabb beadásra egy héttel később. A „gyakori hibák” dokumentum, amit utólag kaptunk sokkal hasznosabb lett volna még az első analízis modell előtt, hogy a fő támponok és szintaktika ne okozzon gondot. Valamint szerintünk nem minden csapatnál ideális rögtön leponozni, ha hibák vannak (még ha komoly is, de nem javíthatatlan). Ha egy csapat megfelelő időt fektet bele és próbálkozik ideálisabb szerintem figyelmeztetni a hibáakra és ezt csak pár pont levonással érzékelhetni, ugyanakkor figyelmeztetni a csapatot, hogy ha a következő beadásnál még lesznek súlyos hibák akkor sokkal azok komolyabban lesznek büntetve.

Az IMSc pontozáson is lehetne valamennyit javítani, mert csak azért elveszteni 2-3 pontot nem jó érzés, hogy a szótárból egy szó kimerül, a szkeleton outputjánál nem határoztuk meg a listák kiíratási sorrendjét, vagy a grafikus megjelenítésen nincs rajta minden létező tárgy, csak minden második. Jó, hogy erre felhívták a figyelmünket, mert tényleg érdemes a dokumentáció során minden alaposan részletezni, de az imsc pontnak nem ettől kéne függnie, inkább egy plusz tárgy vagy játékfunkció, amit a csapatnak meg kell valósítania, vagy valamilyen további diagram a dokumentáció valamely pontján ami alapból nem lenne kötelező.

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

Bármilyen stratégiai-logikai társasjátékokra alapuló projekt jó lehet.

### **11.2.7 Egyéb kritika és javaslat**

A változtatási javaslatokon kívül nincs más javaslatunk.

A tárgy hasznos és fontos, jó kipróbálni magunkat csapatban, komolyabb dokumentációval foglalkozni. Gyakorlásnak kiváló, talán a tanulás része nem jelentős, de lehet ennek a tárgynak ez nem is célja. A játék komolysága is megfelelő volt, valamint a ráfordított órákból is látszódik, hogy a tárgy elvégzése a kreditértékel arányos időt igényel (csak legfeljebb hozzá vagyunk szokva ahhoz, hogy a kreditérték és a ráfordított óra nem ilyen arányban szokott lenni).