Végleges egyesített dokumentáció

6 – ripgyork

Konzulens:
Ádám Zsófia

Csapattagok

Fodor Attila	EUGN1B	afodor998@gmail.com
Fodor Dávid	D02DBR	dfodor999@gmail.com
Földi Balázs	AB8Y3S	fbalu8@gmail.com
Ludányi Barnabás	V5PWP4	ludanyib2003@gmail.com
Mikola Bálint István	TCV0Y9	mikola.balint.istvan@gmail.com (kapcsolattartó)

2024.05.23

Tartalom

2. Követelmény, projekt, funkcionalitás

6 – ripgyork

Konzulens:
Ádám Zsófia

Csapattagok

Fodor Attila	EUGN1B	afodor998@gmail.com
Fodor Dávid	D02DBR	dfodor999@gmail.com
Földi Balázs	AB8Y3S	fbalu8@gmail.com
Ludányi Barnabás	V5PWP4	ludanyib2003@gmail.com
Mikola Bálint István	TCV0Y9	mikola.balint.istvan@gmail.com

2024.02.25

2. Követelmény, projekt, funkcionalitás

2.1 Bevezetés

2.1.1 Cél

A dokumentum célja bemutatni az e félévben tervezett munkálatainkat a projekt feladaton. Bemutatja az előzetesen elképzelt működést és megismerteti az olvasót a játék alapvető elképzelésével, működésével és annak különböző feltételeivel. Elolvasás után már bárki képes lesz szakszerűen használni, illetve játszani a játékunkat.

2.1.2 Szakterület

A szoftver egy játékot fog megvalósítani, amellyel, bárki, bármikor játszhat, akinek hozzáférése van. Ezen felül természetesen a szakmai oldalát nézve, ez egy kötelező projektfeladat, így felhasználási szakterülete még kiterjed a beadandó házifeladatra is, illetve a szakmai tanulmányok előrehaladásának segítésére is.

2.1.3 Definíciók, rövidítések

A dokumentumban rengeteg már az angolból közismert szóval találkozunk majd, melyekre mi is hivatkozunk és használjuk őket, mint akár bármilyen hétköznapi használatban. Rengeteg számítógépes játékkal és azok "szótárával" bővült a magyar nyelv, így nem minden (egyértelmű) definíciót tüntetünk fel. Természetesen vannak rövidítések, definíciók melyeket a fejlesztő csapat saját használatában lesznek fogalmazva, ezek a következők: (igazi - mi általunk használt felbontásban)
Repository – repo
Use case diagram - használati esetek diagramja

2.1.4 Hivatkozások

https://www.iit.bme.hu/dashboard -feladat kiírás és egyéb fontosabb információk
https://gaphor.org/ - use case diagram elkészítése
https://www.messenger.com/ - kommunikációs csatorna, csoport formájában
https://www.microsoft.com/hu-hu/microsoft-teams - másodlagos kommunikációs csatorna
-https://onedrive.live.com- - dokumentációs csatorna (adatvédelmi szempontból nem linket teszünk ide)
https://github.com/ - projekt kódjához használt fejlesztői platform

2.1.5 Összefoglalás

Innentől a dokumentumban, ezen rövid ismertetés után, következnek a lényeges tudnivalók. Ezekből nyerhet a felhasználó majd lényegi és tudásbeli át-, illetve betekintést a szoftver működésébe. Megtudhatja belőle, hogyan is van tervben a szoftver belső felépítése, ki milyen módon, illetve milyen célra tudja majd használni. Ezen felül olvasni lehet a fontosabb követelményekről, illetve a használati esetekről. Szerepelni fog még a szótár is, melybe a saját "nyelvi eszközeinket" fogjuk felsorolni, ismertetni. Majd végül a projekt elkészítésének és folyamatának terve lesz megtalálható a dokumentum végén. Illetve természetesen egy munkanapló is melybe folyamatosan vezetve lesz, minden a projektbe fektetett munkaóra és azok személyenkénti lebontása.

2.2 Áttekintés

2.2.1 Általános áttekintés

A megvalósítandó projekt a Logarléc elnevezésű egyszerű játék, melynek lényege, hogy a tetszőleges számú, élő személyek által vezérelt hallgatók valamelyike, birtokába vegye a logarléc elnevezésű tárgyat, mielőtt a hallgatók számától függő számú, gép által vezérelt oktatók ki nem ejtik az összes hallgatót. A kiejtés feltételei, valamint a lehetséges kimenetelek a „Funkciók” bekezdésben vannak részletesen kifejtve. A játék fontosabb elemei a hallgatók, melyek tárgyakat tudnak felvenni, letenni, használni őket, valamint közlekedni is tudnak a szobák között; az oktatók, akik a játékosok ellen dolgoznak; a szobák, melyek képesek osztódni, összeolvadni, valamint tetszőleges számú más szobákkal összeköttetésben állni; valamint a tárgyak, melyek a játék kimenetelét képesek befolyásolni. A játékosszám megadása után a hallgatók, oktatók és tárgyak elosztása véletlenszerűen történik a szobákba, annyi kikötéssel, hogy a logarléccel egy szobába nem kerülhet hallgató. Az oktatók, tárgyak és szobák kezdeti száma mind a játékosszámtól függ. A játékosok a hallgatókat billentyűlenyomásokkal irányíthatják. A játékmenet felépítése oly módon történik, hogy a hallgatók egymás után hajtják végre lépéseiket, ha minden hallgató lépett, akkor megtörtént egy teljes ciklus (lásd Szótár). A lépési lehetőségek, valamint a játékmenet vizuális megjelenítése szintén a „Funkciók” bekezdésben van részletesen kifejtve.

2.2.2 Funkciók

Projektünkben egy Logarléc névre hallgató körökre osztott stratégia játékot fogunk megvalósítani. A játék célja, hogy a játékosok mágikus szobákon átverekedve magukat és különböző tárgyakat használva megkaparintsák a mágikus logarlécet, miközben próbálják elkerülni a gonosz oktatókat.

A pályát négyzet alakú szobák négyzetrácsa alkotja. Egy szobából véges számú ajtó nyílhat másik szobákba. Mivel a szobák rendszere egy elvarázsolt útvesztőt alkot; a szoba egyik ajtaján kilépve közel sem biztos, hogy a szomszédos szobába fog jutni a játékos. Egy szobában 3 féle ajtó lehet: Olyan, amin csak távozni lehet; Olyan, amin keresztül csak érkezni lehet és olyan, ami 2 irányú. Egy szobából egyértelműen látszik az összes olyan szoba, ahova lépni lehet. Minden szobának van befogadóképessége, ami egy véletlenszerűen generált szám 1 és játékos szám + oktatószám között. Minden játék elején játékosszámtól függő darab szoba generálódik le véletlenszerű szomszédokkal. Létrejöttükkor bizonyos szobák el vannak gázosítva. Ezekbe a szobákba a belépés eszméletvesztéssel és a következő ciklusból való kimaradással jár, kivéve, ha van maszkja a játékosnak (lásd lentebb a ciklusokat és a tárgyakat). A szobák száma a későbbiekben még változhat (lásd lentebb a szobák osztódását és egyesülését.)

A játék ún. ciklusokra van osztva. Egy ciklus addig tart, amíg minden játékos (és oktató, lásd: lentebb) sorra nem kerül, és végre nem hajtja a körét.

Egy játékosnak minden körben van 3 elköltendő akciópontja. A játékos körének akkor és csak akkor van vége, ha mindhárom akciópontját elköltötte. Egy játékos az akciópontjait 3 féle akcióra költheti el: Átlépés egy másik szobába, egy szobában lévő tárgy felvétele, illetve hallgatónál lévő tárgy letétele. (Tárgyakról bővebben: lentebb).

A játékosokra a játék egész ideje alatt oktatók „vadásznak” az alább kifejtett módon. A játék elején játékosszámtól függő darab oktató véletlenszerűen elhelyezésre kerül szobákban. Az oktatók köre mindig a játékosok köre után következik. Az oktatóknak is minden körben 3 akciópontjuk van, melyeket véletlenszerűen költenek el minden ciklusban. Az oktatókat lehet látni akkor is, ha nincs senki azonos szobában velük. Ha egy hallgató egy olyan szobába lép, ahol oktató is tartózkodik, vagy pedig egy oktató lép egy hallgatót tartalmazó szobába, akkor még abban a körben kiszívja a játékos lelkét, ezzel kiejtve az adott játékost az adott játékból.

A hallgatók feladatát különböző tárgyak segítik. Minden tárgy a játék elején jön létre és véletlenszerűen elhelyezésre kerül szobákban. A játék elején létrejövő tárgyak száma arányos a játékosok számával. Kívülről nem látszik, ha egy tárgy egy szobában van. Meglátni és felvenni egy tárgyat csak akkor tudunk, ha belépünk az őt tartalmazó szobába. Alap helyzetben minden tárgy inaktív állapotban van és csak onnantól érvényesül a képessége, hogy egy játékos felvette. Játék közben nem tud új tárgy keletkezni, csak elhasználódni. Bizonyos tárgyaknak vannak tartósságpontjaik, amik nullára csökkenés esetén elveszik az adott tárgy képességét, ebben az esetben a tárgy még a játékosnál marad. Tárgyakat az oktatók is tudnak felvenni, viszont kizárólag „elkobzás” céljából, használni nem tudják ezeket a tárgyakat, illetve a logarlécet még elkobozni sem tudják. Minden játékosnál és oktatónál maximum 5 tárgy lehet. A lentiekben a részletezzük a játékban található tárgyak tulajdonságait:

- Logarléc: Amint aktiválják (felveszik a földről), a játékot megnyerik a játékosok (oktató nem veheti fel.)
- TVSZ denevérbőrre nyomtatott példánya: Ha ez a tárgy egy játékosnál és ez a játékos egy szobába kerül egy oktatóval, akkor az oktató nem ejti ki a játékost, a TVSZ veszít egy tartósságpontot és a játék halad tovább. Aktiváláskor a tárgy három tartósságponttal rendelkezik.
- Szent sörös poharak: Ezeknek a tárgyaknak a képessége megegyezik a TVSZ denevérbőrre nyomtatott példányainak képességével, annyi különbséggel, hogy a szent sörös poharak minden ciklusban veszítenek egy tartósságpontot.
- Nedves táblatörlő: Ha egy játékosnál egy nedves táblatörlő van és egy szobába kerül egy oktatóval, akkor egy ciklus erejéig megbénítja az oktatót. A táblatörlő 3 tartósságponttal kezd és ciklusonként veszít egyet.
- Dobozolt káposztás camembert: Ezt a tárgyat letéve, a tárgy elgázosítja a szobát, ami a játék végéig gázos is marad.
- FFP2 maszk: Hordozója nem ájul el az elgázosított szobában. Három tartósságponttal rendelkezik és minden olyan ciklusban veszít egy pontot, melyben a viselője egy elgázosított szobában tartózkodik valamennyi időre.
- Tranzisztor: Két tranzisztor automatikusan összepárosodik, ha felveszi őket egy játékos. Egy összepárosított tranzisztorpár egyik darabját lehelyezve a játék végéig a másik tranzisztort birtokló játékos visszateleportálhat abba a szobába, ahova a tranzisztort letette.

A játéktéren uralkodó mágia miatt a játék elején véletlen darab ajtó megátkozódik és miután háromszor átmentek rajta, 3 ciklus erejéig eltűnik, majd újra felbukkan. Amíg egy ajtó el van tűnve, addig nem lehet áthaladni rajta. Szintén a fent említett mágia miatt minden ciklus elején a szobák véletlenszerűen osztódhatnak, helyben maradhatnak vagy pedig összeolvadhatnak egy szomszédos szobával. Két szomszédos szoba egyesülésével létrejövő szoba a korábbi két szoba tulajdonságaival és szomszédaival rendelkezik, de a befogadóképessége a nagyobb szoba befogadóképességével lesz azonos. Az újonnan létrejött szoba tartalmazni fogja az összes tárgyat, amit az őt létrehozó szobák tartalmaztak. Az osztódó szoba két olyan szobára válik szét, amelyek egymás szomszédai lesznek, és megosztoznak a korábbi szoba képességein és szomszédain (a korábbi szomszédok, vagy csak az egyik, vagy csak a másik "új" szobának lesznek szomszédai). Osztódáskor a tárgyak, játékosok és oktatók véletlenszerűen kerülnek az egyik, illetve a másik szobába. Szobák egyesülésére az egyetlen feltétel, hogy 3 szobának minimum lennie kell a pályán.

Egy játékon cikluskorlát nincs, addig tart, amíg minden játékos ki nem esik, vagy valaki fel nem veszi a logarlécet.

2.2.3 Felhasználók

A felhasználóktól alapvetően elvárt az olvasási készség, de ezen kívül egyéb megkötés nem szükséges a játék használatához. A játék célközönsége a jellegéből adódóan legfőképp a fiatalabb generációk képviselői, viszont a játék egyszerűsége miatt semelyik korosztálynak nem jelenthet komolyabb kihívást.

2.2.4 Korlátozások

Az elkészített szoftvert a kari hercules weblapról indulva kell feltölteni, valamint elvárás, hogy a kari felhőben (10 20H2 – JDK-Eclipse-WSU sablon) biztosított, az ott megtalálható JDK alatt lefordítható és futtatható legyen.

2.2.5 Feltételezések, kapcsolatok

A kari oldalon megfogalmazott elvárások és útmutatás alapján építettük fel a tervezetet. Mindent a feladat szövegének megfelelően építettünk fel a tervünkben, mellyen dolgozni fogunk. Egyes részekhez a feladat jellege alapján különböző weboldalakat is használatba vettünk a tervezéshez, lásd pl. Gaphor.

2.3 Követelmények

2.3.1 Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
REQ-01	A rendszernek képesnek kell lennie a játékterület és a játékosok kezelésére.	Ellenőrizni kell, hogy a rendszer kezeli-e a játékosok interakcióit.	Alapvető	Csapat	Játékos mozgatása	
REQ-02	A rendszernek lehetővé kell tennie a játékosoknak, hogy tárgyakat vegyenek fel és helyezzenek, vagy dobjanak le.	Ellenőrizni kell, hogy a játékosok fel tudnak-e venni és letenni tárgyakat a játékterületen.	Alapvető	Csapat	Tárgyak Felvétele, Tárgyak Lehelyezése, Tárgyak eldobása	A tárgyak kezelése nélkülözhetetlen a játékmenetben.
REQ-03	A rendszernek tartalmaznia kell mechanizmust az oktatók megjelenítésére és kezelésére.	Ellenőrizni kell, hogy az oktatók megjelennek-e a játékterületen és interakcióba léphetnek-e a játékosokkal.	Alapvető	Csapat	Oktató Találkozása	Az oktatók fontosak a játék kihívásainak biztosításához.
REQ-04	A rendszernek szükséges az elátkozott szobák speciális tulajdonságainak kezelésére.	Ellenőrizni kell, hogy a rendszer megfelelően kezeli-e az elátkozott szobák sajátosságait és hatásait a játékmenetre.	Fontos	Csapat	-	Az elátkozott szobák változatos kihívásokat jelentenek a játékosoknak.
REQ-05	A rendszernek biztosítania kell a tranzisztorok működését és kezelését.	Ellenőrizni kell, hogy a tranzisztorok a játékban megfelelően működnek-e és a játékosok tudják-e őket kezelni.	Fontos	Csapat	Tárgyak felvétele, Tárgyak eldobása, Tárgyak lehelyezése	A tranzisztorok fontos szerepet játszanak a játékban a tárgyak mozgatásában.
REQ-06	A rendszernek lehetőséget kell adnia a játékosoknak a játék megnyerésére.	Ellenőrizni kell, hogy a játékosok képesek-e megnyerni a játékot a Logarléc megtalálásával.	Alapvető	Csapat	Játék Megnyerése	A játék célja a játékosok győzelmének lehetősége.
REQ-07	A rendszernek lehetőséget kell adnia a játékosoknak a játék elvesztésére.	Ellenőrizni kell, hogy a játékosok képesek-e elveszteni a játékot, ha az oktatókkal találkoznak	Alapvető	Csapat	Játék Elvesztése	
REQ-08	A rendszernek képesnek kell lennie a játékosok tárgyainak interakcióira az elátkozott szobákban.	Ellenőrizni kell, hogy a tárgyak megfelelően működnek-e az elátkozott szobákban és azok hatással vannak-e a játékmenetre.	Fontos	Csapat	-	A tárgyak interakciója segíti a játékosokat az elátkozott szobák kihívásainak leküzdésében.
REQ-09	A rendszernek lehetőséget kell adnia a játékosoknak a játék elindítására és befejezésére.	Ellenőrizni kell, hogy a játékosok képesek-e elindítani és befejezni a játékot a főmenüből.	Alapvető	Csapat	Játék Indítása, Kilépés a Játékból	A játékmenet kezelése alapvető funkcionalitás a felhasználók számára.
REQ-10	A rendszernek képesnek kell lennie az oktatók tárgyainak interakciójára	Ellenőrizni kell, hogy a rendszer ezt lehetővé teszi-e.	Fontos	Csapat	-	

2.3.2 Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
REQ-1	Alap számítógépes tartozékok.	Ellenőrizni kell hogy monitor, billentyűzet valamint egér elérhető-e.	nélkülözhetetlen	Játékos kell biztosítsa	Enélkül játszhatatlan a játék

2.3.3 Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
REQ-01	Projekt feladat bemutatásakor futtatni kell a megadott virtuális gépen	A futtatást ellenőrzött labor körülmények közt végezzük	Időben elkészített forráskód és dokumentáció.	Ez a projekt elfogadásának feltétele így a forrás a feladat kiadója.	

2.3.4 Egyéb nem funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
REQ-1	Tesztelhetőség	A program valamint a dokumentáció jól értelmezhető kell legyen.	fontos	Fejlesztők felelőssége	Későbbi munkát nagyban megkönnyíti ha ennek eleget teszünk

2.4 Lényeges use-case-ek

2.4.1 Use-case leírások

Use-case neve	Játékosszám beállítása
Rövid leírás	A főmenüben a felhasználó beállítja a játékhoz résztvevő játékosok számát, ami befolyásolja a játék egészét.
Aktorok	Játékosok: A felhasználók, akik részt vesznek a játékban.
Forgatókönyv	1. A felhasználó elindítja a programot. 2. A program megjeleníti a főmenüt. 3. A felhasználó kiválasztja a szövegdobozt, amelyben beállíthatja a játékosok számát. 4. A felhasználó átírja a szövegdobozban lévő számot a kívánt játékosszámra. 5. Alternatív lefutás: Ha a felhasználó érvénytelen értéket ad meg (pl. negatív szám, nem egész szám stb.), a program figyelmezteti és lehetőséget ad az érvényes érték megadására. 6. A program frissíti a játékosszámot és a játékmenetet a beállított számhoz igazítja.

Use-case neve	Játék indítása
Rövid leírás	A felhasználó elindítja a játékot, hogy részt vehessen a játékmenetben.
Aktorok	Játékosok: A felhasználók, akik részt vesznek a játékban.
Forgatókönyv	1. A felhasználó elindítja a programot. 2. A program betölti a kezdőképernyőt. 3. A felhasználó kiválasztja a "Játék Indítása" lehetőséget a kezdőképernyőn. 4. A program előkészíti a játékmezőt és a játékmenetet. 5. Alternatív lefutás: Ha a játék betöltése során hiba lép fel (pl. hiányzó fájlok, adatbázis hiba stb.), a program megjeleníti a hibaüzenetet. 6. A játék elindul, és megjeleníti a játékteret a felhasználónak. 7. A felhasználó játszik a játékkal.

Use-case neve	Kilépés a játékból
Rövid leírás	A felhasználó bezárja a programot az ablakon található bezárás gombbal
Aktorok	Játékosok: A felhasználók, akik részt vesznek a játékban.
Forgatókönyv	1. A felhasználó játszik a játékkal. 2. Amikor a felhasználó befejezni kívánja a játékot, az ablak jobb felső sarkában található bezárás gombra kattint. 3. A játékprogram érzékeli a bezárás gombra kattintást. 4. Ha a felhasználó véletlenül nyomja meg a bezárás gombot, a program figyelmeztetheti a felhasználót a bezárásról és megkérdezi, hogy biztosan ki akar-e lépni a játékból. 5. A program bezárja a játékot.

Use-case neve	Játékos léptetése
Rövid leírás	A játékos kiválaszt egy mezőt a játékmezőről, majd odalép arra a mezőre a játékmenetben, amennyiben ez lehetséges.
Aktorok	Játékosok: A felhasználók, akik részt vesznek a játékban.
Forgatókönyv	1. A játékos kiválaszt egy mezőt a játékmezőről. 2. A program ellenőrzi, hogy a kiválasztott mező lépésközbeni elérhetőségét. 3. Ha a mező lépésközben elérhető, a játékos odalép arra a mezőre. 4. Alternatív: Ha a mező nem elérhető, akkor a játékos nem mozdul, újra próbálkozhat. 5. Ha a mezőre lépés közben valamilyen akció vagy esemény következik be (pl. találkozás ellenféllel), a program feldolgozza az eseményt és értesíti a játékost az eredményről. 6. A program frissíti a játékteret a játékos új pozíciójával és az esetlegesen változó játékállapottal.

Use-case neve	Tárgy felvétele
Rövid leírás	A játékos felvesz egy tárgyat a játékmezőről és hozzáadja azt a saját eszköztárához.
Aktorok	Játékosok: A felhasználók, akik részt vesznek a játékban.
Forgatókönyv	1. A játékos egy olyan mezőre lép, vagy már áll, amin található legalább egy tárgy. 2. A játékos kiválaszt egy tárgyat a játékmezőn és megkísérli felvenni. 3. A program ellenőrzi, hogy a tárgy felvehető-e 4. Alternatív: Ha a tárgy nem vehető fel, mert nincs elég szabad hely az eszköztárban, akkor a tárgy a földön marad. 5. Ha a tárgy felvétele során valamilyen esemény következik be, a program feldolgozza az eseményt és értesíti a játékost az eredményről. 6. A program hozzáadja a tárgyat a játékos eszköztárjához. 7. A program frissíti a játékteret, hogy a tárgy eltűnjön a játékmezőről (ha szükséges), és megjelenítse a játékos aktuális eszköztárát.

Use-case neve	Tárgy eldobása
Rövid leírás	A játékos eldob egy tárgyat a saját eszköztárából.
Aktorok	Játékosok: A felhasználók, akik részt vesznek a játékban.
Forgatókönyv	1. A játékos kiválasztja a tárgyat az eszköztárából, amelyet el szeretne dobni. 2. A program ellenőrzi, hogy a tárgy eldobható-e (lentebb, a tranzisztor esete) 3. Ha a tárgy eldobható, a program eltávolítja azt a játékos eszköztárjából, és elhelyezi a játékmezőn. 4. A program frissíti a játékteret, hogy a tárgy megjelenjen a játékmezőn és az eszköztár frissült állapotát mutassa.

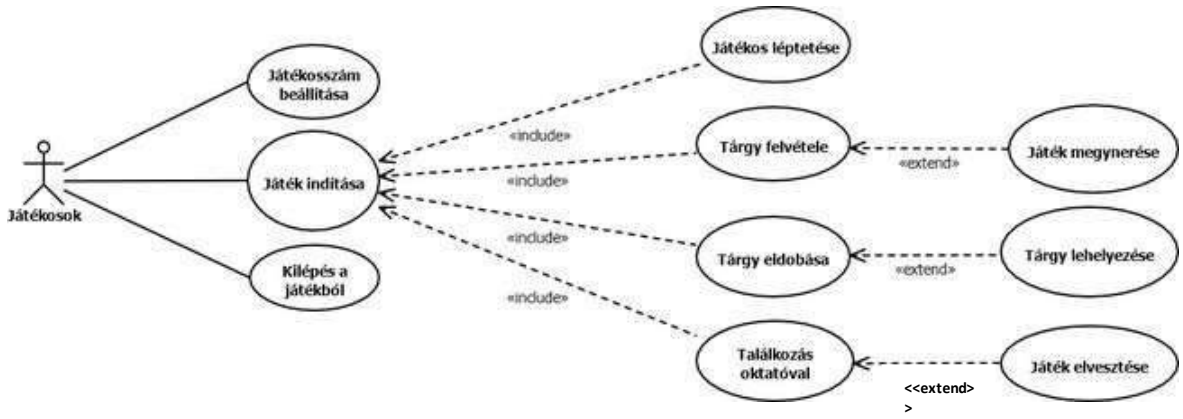
Use-case neve	Tárgy lehelyezése
Rövid leírás	A játékos lehelyez egy tárgyat a saját eszköztárából. Az eldobás egy speciális esete
Aktorok	Játékosok: A felhasználók, akik részt vesznek a játékban.
Forgatókönyv	1. A játékos kiválasztja a „Tranzisztor” nevű tárgyat az eszköztárából, amelyet le szeretne helyezni. 2. A program ellenőrzi, hogy a tárgy lehelyezhető-e (Csak akkor helyezhető le, ha legalább 2 tranzisztor van az eszköztárjában az adott játékosnak, vagy ha az adott szobában még nincs egy aktív tranzisztor se) 3. Ha a tárgy lehelyezhető, a program eltávolítja azt a játékos eszköztárjából, és elhelyezi a játékmezőn. 4. Alternatív: Amennyiben a tranzisztort nem lehet lehelyezni, tehát az eszköztárban csak 1 van, akkor az akció „Tárgy eldobása” akciónak minősül. 5. A program frissíti a játékteret, hogy a tárgy megjelenjen a játékmezőn és az eszköztár frissült állapotát mutassa.

Use-case neve	Játék megnyerése
Rövid leírás	A játékosok együttműködve teljesítik a játék célját és ezzel győzelmet aratnak.
Aktorok	Játékosok: A felhasználók, akik részt vesznek a játékban.
Forgatókönyv	1. A játék folyamán a játékosok együttműködnek és közösen teljesítik a játék célját. 2. A program érzékeli a győzelmet. 3. A program megjeleníti a győzelmi üzenetet a játékosok számára, értesítve őket a nyerésről. 4. A program megjeleníti a kezdőképernyőt adott idő letelte után.

Use-case neve	Játék elvesztése
Rövid leírás	A játékosoknak nem sikerül teljesíteniük a játék célját és ezzel veszítik el a játékot.
Aktorok	Játékosok: A felhasználók, akik részt vesznek a játékban.
Forgatókönyv	1. Amikor a játék során valamilyen oknál fogva a játékosok nem képesek teljesíteni a továbbiakban a játék célját a program érzékeli a vereséget. 2. A program megjeleníti a vesztes üzenetet vagy animációt a játékosok számára, értesítve őket a játék elvesztéséről. 3. A program megjeleníti a kezdőképernyőt adott idő letelte után.

Use-case neve	Találkozás oktatóval
Rövid leírás	A játékosok találkoznak egy vagy több oktatóval egy szobában ezzel interaktálnak az oktatóval.
Aktorok	Játékosok: A felhasználók, akik részt vesznek a játékban.
Forgatókönyv	1. A játékosok egy szobában tartózkodnak. 2. Az oktatók belépnek a szobába, ahol a játékosok tartózkodnak. 3. Amikor az oktatók és a játékosok egy szobában találkoznak, a program érzékeli az ütközést. 4. A program megvizsgálja, van-e bármi ok, ami miatt a hallgatók nem vesztik el a játékot (tárgyak) 5. Ha nincs, akkor a játéknak vége, ha van akkor a tárgynak megfelelő esemény következik be.

2.4.2 Use-case diagram



1. ábra

2.5 Szótár

Akció - tárgy felvétel, tárgy letétel és lépés
Akciópont - ebbe kerülnek az akciók
Ciklus - két játékos egy-egy köre (tehát “A” és “B” játékos is elvégzi a lépéseit)
Eszköztár - A játok egy 5 elemű tárhelye, ahol a felvett tárgyakat tárolja
Gázmaszk - FFP2 maszk
Játék - a szoftver egy játszható program, így játékként hivatkozunk rá
Káposzta - dobozolt káposztás camembert
Kör - egy játékos 3 akciópontjával elvégzett összes tevékenység
Sör - szent söröspohár
Táblatörlő - nedves táblatörlő
TVSZ - TVSZ denevérbőrre nyomatatott példánya
Útvesztő - labirintus (könnyebb kimondani)

2.6 Projekt terv

A Szoftver projekt laboratórium projektcsoportunk, a ripgyork, egy komplex folyamat részeként, gondosan előkészített stratégiával és elkötelezett csapattagok bevonásával vállalkozik a projekt megtervezésére és megvalósítására. A siker érdekében számos lépést, és minden tőlünk telhető erőfeszítést megteszünk a határidők pontos betartása és a projekt kiváló minőségének elérése érdekében.
A csapat tagjai: Fodor Attila, Fodor Dávid, Földi Balázs, Ludányi Barnabás, Mikola Bálint (csapatvezető).
A projektirányításért Mikola Bálint, a csapatvezető felelős. Heti rendszerességű személyes megbeszéléseket tartunk a csapattagokkal, ahol közösen áttekintjük és megbeszéljük az adott heti feladatokat. Ezután egységesen, a csapat összes tagja részvételével osztjuk ki a feladatokat, figyelembe véve az egyes tagok képességeit, tapasztalatát és preferenciáit, valamint a projekt aktuális állapotát. Abban a szélsőséges esetben, ha a csapat nem jutna egyetértésre a feladatok kiosztásában vagy a feladatot érintő döntésekben, előzetes megegyezés alapján a csapatvezető szava dönt. Célunk az, hogy a projektmunka során minden csapattag elégedett legyen a feladatkörével, és motiváltan dolgozhasson a játék elkészítése során. A rugalmas munkamegosztás és a csapatvezető vezetői döntése révén biztosítjuk, hogy mindenki hatékonyan és összehangoltan tudjon dolgozni a lehető legjobb eredmény elérésének érdekében.
Az IIT oldalán meghirdetett határidőket alaposan tanulmányoztuk, megértettük és rögzítettük. Ezek a határidők és/vagy mérföldkövek alapvetően meghatározzák a projekt készületének menetét és lépéseit. A projekt folyamán a személyes megbeszélések mellett rendszeres online konzultációkat is tartunk, hogy megosszuk egymással a haladást. A csapatmunkát támogató eszközök széles skáláját alkalmazzuk. A kommunikáció elsődleges csatornája a Messenger alkalmazás, másodlagosan a Microsoft Teams alkalmazás. Ezen döntések alapját a kommunikáció gyorsaságának, megbízhatóságának biztosítása és az alkalmazások által kínált lehetőségek határozták meg. A forráskód megosztásához egy, a csapatvezető által létrehozott GitHub repository-t használunk, kihasználva, hogy ezen verziókezelő rendszer használatát a csapat minden tagja elsajátította a 3. féléves tanulmányaink során. A dokumentum megosztásához egységesen a Google Drive mellett döntöttünk. Segítségével akár több csapattag is dolgozhat egyszerre ugyanazon a dokumentumon és a megtekinthető verzióelőzmények segítenek bármely csapattagnak megérteni és átlátni a dokumentumban keletkezett változtatásokat.
Az erőforrások tekintetében gondosan tervezünk és gazdálkodunk az idővel és az emberi erőforrásokkal. Figyelembe véve a feladat komplexitását és a rendelkezésre álló időkeretet, célunk a munkafolyamat és a csapatmunka optimalizálása. A választott technikák és keretrendszerek alaposan átgondoltak és megfelelően alkalmazottak, biztosítva a hatékony fejlesztési folyamatot.
Összességében, projektünk sikere érdekében elkötelezetten dolgozunk együtt, minden erőforrást felhasználva és minden lehetőséget kihasználva, hogy a legjobb eredményt érjük el a meghatározott határidőkig.

2.7 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2024.02.21 13:00	1,5 óra	Fodor A. Fodor D. Földi Ludányi Mikola	Értekezlet. Döntés: Fodor D. szerkeszti a dokumentumot és készíti el a 2.1-es részt. Ludányi és Mikola készítik a 2.2-es részt. Földi a use caseket és a diagramot. Fodor A. írja a projekttervet. A szótárat és a naplót mindenki a saját része szerint tölti ki. A 2.3-as rész együttes munkával kell kitölteni. Nyomtatást Fodor A. és Fodor D. végzik. Minden feladatot egyeztetett határidőig (2024.02.25 12:00) el kell végezni.
2024.02.24 14:00	2 óra	Fodor D.	Tevékenység: Fodor D. elkészítette a dokumentum teljes 2.1-es részét és szerkesztette a dokumentumot és a 2.3.3-es részt. OneDrive létrehozása.
2024.02.23 16:00	2 óra	Ludányi	Tevékenység: A 2.2-es rész alpontjai a “Funkciók” rész kivételével, valamint a 2.3.2-es és 2.3.4-es részek.
2024.02.24 17:00	2 óra	Mikola	Tevékenység: A 2.2.2-es rész kidolgozása.
2024.02.24 18:15	2 óra	Fodor A.	Tevékenység: A 2.6-os rész kidolgozása, a dokumentum helyesírási és nyelvtani hibáinak kijavítása, nyomtatás előkészítése
2024.02.22 16:25	1 óra	Földi	Tevékenység: A Use Case leírások kidolgozása
2024.02.24 11:00	1 óra	Földi	Tevékenység: Funkcionális követelmények és Use case diagram kidolgozása

3. Analízis modell (I. változat)

6 – ripgyork

Konzulens:
Ádám Zsófia

Csapattagok

Fodor Attila	EUGN1B	afodor998@gmail.com
Fodor Dávid	D02DBR	dfodor999@gmail.com
Földi Balázs	AB8Y3S	fbalu8@gmail.com
Ludányi Barnabás	V5PWP4	ludanyib2003@gmail.com
Mikola Bálint István	**TCV0Y9**	**mikola.balint.istvan@gmail.com (kapcsolattartó)**

2024.03.03

3. Analízis modell kidolgozása

3.1 Objektum katalógus

3.1.1 Objektum1 -> Game

Az objektum feladata a játékmenet, pontosabban a játék egyes köreinek menedzselése, valamint a játékban résztvevő oktatók, hallgatók és nyilvántartása, tárolása. Ezen felül a véget ért körök számát is tárolnia kell. Az objektum feladata a halott játékosok eltávolítása, valamint annak vizsgálata, hogy van e még életben lévő hallgató. Amennyiben nincs, véget vet a játéknak.

3.1.2 Objektum2 -> Room

Ennek az objektumnak a felelőssége a szobákkal kapcsolatos legtöbb információ tárolása, valamint a szobákhoz tartozó legtöbb metódus végrehajtása vagy kezdeményezése. Tárolnia kell a szoba kapacitását, a benne tartózkodó játékosokat és tárgyakat, valamint azt is, hogy az adott szoba gázos-e vagy sem. A szobákban különböző ajtók is találhatók, melyeken keresztül a játékosok másik szobákba juthatnak.

3.1.3 Objektum3 -> Door

Az ajtó objektum feladata számon tartani magáról, hogy elátkozott-e, mely szobákba nyílik, zárva van-e (elátkozott szoba esetén ajtók adott időre eltűnnek) valamint számon kell tartania a rajta áthaladások számát is.

3.1.4 Objektum4 -> Board

Ennek az objektumnak a felelőssége az oktatók és hallgatók, a szobák, valamint a tárgyak listázása, nyilvántartása. Ezen felül felelőssége a szobák kettéosztódásának, valamint összeolvadásának menedzselése és megvalósítása.

3.1.5 Objektum5 -> Item

Az Item [tárgy] objektum feladata a tárgyak nevének tárolása, valamint a tárgyak kopásának, elhasználódásának mértékét is menedzselnie kell. Ezen kívül felelőssége tudni minden tárgyról, hogy éppen aktív-e vagy sem, valamint, hogy az adott tárgy éppen fel van e véve egy hallgató vagy egy oktató által (csak a felvétel tényét kell tárolni, hogy oktató vagy hallgató vette-e fel, azt nem).

3.1.6 Objektum6 -> Player

Ezen objektum felelőssége a játékos birtokában lévő tárgyak nyilvántartása, valamint minden játékos rendelkezik egy egyedi azonosítóval is. Tudnia kell azt is, hogy az adott játékos épp bénított állapotban van-e, vagy sem. A játékosok mozgásának, valamint tárgyak felvételének és letételének megvalósítása szintén ennek az objektumnak a felelősségkörébe tartozik. Ezen kívül a hallgatóknál azt is tárolni kell, hogy életben vannak-e vagy sem.

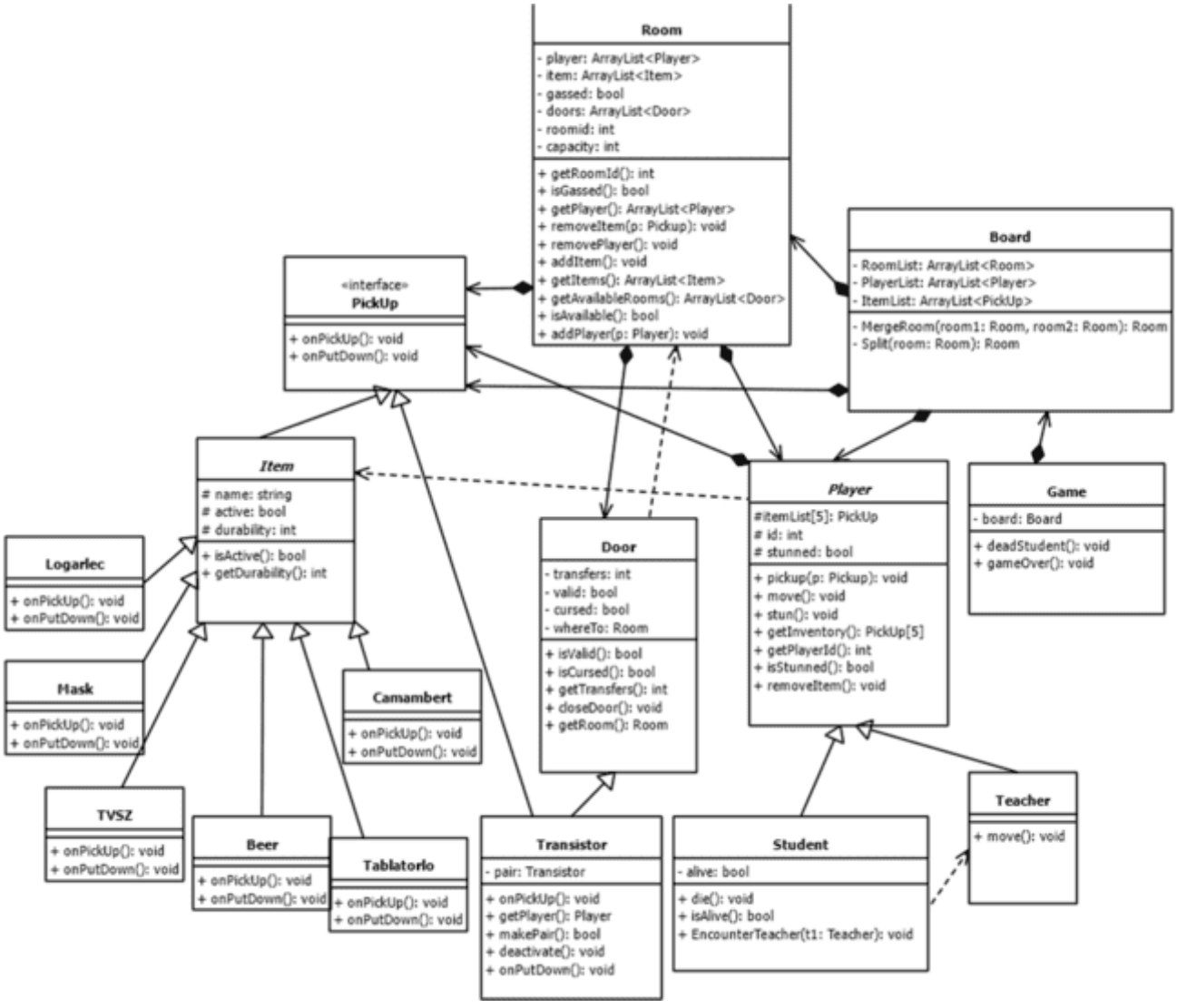
3.1.7 Objektum6 -> Transistor

Ezen objektum egy, a játék során felvehető objektumot reprezentál. A transistorok kezdetben felvehető tárgyak, majd összekapcsolás/összepárosítás után ajtóként viselkednek a két szoba közt, amelybe lehelyezték őket.

3.1.8 Objektum6 -> Logarléc

Ezen objektum a játék legfontosabb elemét reprezentálja. Ez egy, a játékban résztvevő tárgyak egyike. Ez a tárgy mind felett áll, hiszen ennek felvétele egy hallgató által egyben a játék végét is jelzi, hiszen ez a játék célja, hogy ezt megszerezzék, mielőtt az oktatók megtalálnának minden hallgatót.

3.2 Statikus struktúra diagramok



3.3 Osztályok leírása

3.3.1 Board

- **Felelősség**

Ez az osztály felelős a szobák és a játéktér reprezentálásáért. Nyilvántartja a szobákat és azokat összevonhatja és szétszedheti.

- **Asszociációk**
  - **Room**: A szobákkal van kapcsolatban, ezeket tárolja
  - **Player:** Tárolja, milyen játékosok vannak a táblán
  - **PickUp:** Kapcsolatban van az interfésszel, tárolja a felvehető objektumokat

- **Attribútumok**
  - **- RoomList: ArrayList<Room>:** A pályán lévő szobákat tárolja
  - **- PlayerList: ArrayList<Player>:** A pályán lévő játékosokat tárolja
  - **- ItemList: ArrayList<PickUp>:** A pályán lévő felvehető objektumokat tárolja

- **Metódusok**
  - **Void Merge(Room o1, Room o2):** A függvény a paraméterként kapott két szobát vonja össze, a specifikációban leírtaknak megfelelően.
  - **Void Split(Room o1):** A paraméterként kapott szobát kettéosztja a specifikációban leírtak szerint.

3.3.2 Door

- **Felelősség**

Az ajtókat reprezentáló osztály. Egy ajtó nyilvántartja a szobát, ahova rajta keresztül el lehet jutni, ezért az egyik legfontosabb osztály. Lényegében a pálya szobái, mint gráf csúcspontok, közötti irányított élek.

- **Asszociációk**
  - **Room**: A szoba, ahova az ajtón keresztül el lehet jutni
- **Attribútumok:**
  - **- transfers: int** – Megadja, hogy az ajtón eddig hányszor mentek keresztül
  - **- cursed: bool** – Megadja, ha az ajtó átkozott. Ezt a felhasználó úgy fogja majd látni, mintha a szoba lenne az.
  - **- isActive: bool** – elérhető e az ajtó
  - **- whereTo: Room** – Az a szoba, ahova az ajtón keresztül el lehet jutni
  - **Metódusok**
  - **void closeDoor()**: bezár egy ajtót bizonyos időre
  - **bool isValid()**: megnézi, hogy egy ajtó használható-e
  - **Int getTransfers()**: megnézi egy ajtón hányszor mentek át
  - **void useDoor()**: használja az ajtót egy játékos
  - **Room getRoom()**: megadja melyik szobában van az ajtó

3.3.3 Game

- **Felelősség:** A játék (körök, ciklusok, inicializálás, játék vége) menedzselését kezelő osztály.
- **Asszociációk**
  - **Board**: A játék tábláját tárolja, ezzel eléri a rajta lévő objektumokat
- **Attribútumok:**
  - **-board: Board**: A játéktáblát tárolja, ezzel a rajta
- **Metódusok:**
  - **+void deadStudent()**: meghalt játékosok menedzselését végzi, kiveszi őket a játékból.
  - **+ void gameOver()**: Ha olyan esemény történik, ami miatt a játéknak vége, az a metódus kezeli.
- **Megjegyzés:** Ez az osztály egy Controller-ként működik, viszont a feladat jelen szakaszához szükséges metódusait megjelenítettük, a többit egyelőre nem.

3.3.4 Item

- **Felelősség:** A tárgyakat reprezentáló osztályok absztrakt ősosztály, azért absztrakt, mert nem lehet pédányosítani, mert tárgy nincs, csak különböző fajtáik
- **Interface: PickUp:** ezt valósítja meg az Item absztrakt ősosztály
- **Attribútumok:**
  - **# name: string:** A tárgy neve
  - **# durability: int**: A tárgy deaktiválásáig hátralévő ciklusok száma
  - **# active: bool**: A tárgy aktívságát jelzi, ha valami elhasználódik, akkor false az értéke egyébként true, vagy ha nincs felvéve a tárgy akkor is false
- **Metódusok**
  - **+ bool isActive()**: megnézi a metódus, hogy egy tárgy éppen aktív-e vagy nem.
  - **+ int getDurability()**: az elhasználódó tárgyaknak lekérdezi, mennyi "életük" van még hátra

3.3.5 PickUp

- **Felelősség:** A felvehető objektumok függvényeit adó interfész.
- **Metódusok:**
  - **+ void onPickUp()**: Felvételkor elvégzendő tevékenységet/eseményt ez a függvény valósítja meg.
  - **+ void onPutDown()**: Letételkor elvégzendő tevékenységet/eseményt ez a függvény valósítja meg.

3.3.6 Player

- **Felelősség**

A játékosokat (hallgató és oktató) reprezentáló osztályok absztrakt őse.

- **Asszociációk:**
  - **PickUp**: A játékos által tárolt felvehető objektumok kezeléséhez
- **Attribútumok:**
  - **# id: int**: A játékost egyértelműen azonosító szám
  - **# itemList: PickUp[5]**: A játékos 5 elemű eszköztára
  - **# stunned:bool**: kábítva van-e a játékos
  -
- **Metódusok:**

- **+ void move()**: A játékos mozgatását végzi.
- **+ void stun()**: megbénítja a Player-t
- + **pickup(PickUp p)**: Felveszi a paraméterül kapott PickUp objektumot az itemList-jébe
- +**PickUp[5]getInventory()**: lekérdezi a játékosnál milyen tárgyak vannak
- **+ int getPlayerId()**: lekérdezi a játékos egyedi azonosítóját.
- **+bool isStunned()**: lekérdezi, hogy kábítva van-e a játékos
- **+void removeItem()**: eldob egy tárgyat a játékos

3.3.7 Room

- **Felelősség**

A szobákat reprezentáló osztály, tárolja a benne lévő játékosokat (hallgatók, oktatók) és a benne lévő felvehető objektumokat (PickUp).

- **Asszociációk**
  - **PickUp**: Tárolás
  - **Player**: Tárolás
  - **Door**: Tárolás

- **Attribútumok**
  - **- capacity: int:** A szoba befogadóképességét adja, ez a játék elején kap egy alapértéket
  - **- gassed: bool:** Ha az értéke “true”, az azt jelenti, hogy a szoba mérgezett. Ekkor a benne lévő játékosok megkapják a specifikációban leírt effekteket, végrehajtódnak az akciók
  - **- doors: ArrayList<Door>:** Megadja a szobába található ajtókat, ezzel együtt tartja nyilván a szomszédos szobákat.
  - **- items: ArrayList<Item>:** Tárolja a szobába található tárgyakat
  - **- players: ArrayList<Player>** - Tárolja az éppen a szobában tartózkodó hallgatókat és oktatókat
  - **- roomId: int:** A szoba azonosító száma

- **Metódusok**
  - **+int getRoomId()**: visszaadja egy adott szobához tartozó id-t
  - **+bool isGassed():** lekérdezi, hogy az adott szoba gázos e
  - **+ArrayList<Player> getPlayer():** visszaadja egy szobában milyen játékosok tartózkodnak
  - **+void removeItem(PickUp p):** ha valaki felvesz egy tárgyat kiveszi a szobából
  - **+void removePlayer():** ha valaki kimegy egy szobából töröljük a szoba listájából
  - **+void addItem():** ha valaki letesz egy tárgyat az bekerül a szoba listájába
  - **+ArrayList<Item> getItems():** visszaadja, milyen tárgyak vannak a szobában
  - **+ArrayList<Door> getAvailableRooms():** visszaadja, hogy hova lehet jutni a szobából
  - **+bool isAvailable():** lekérdezi, hogy a szoba elérhető-e (nincs-e tele)

- **+void addPlayer(Player p):** ha egy játékos belép egy szobába hozzáadódik a szoba listájához

3.3.8 Student

- **Felelősség**

A hallgatókat reprezentáló osztály. A hallgató képes mindenre, amit az ősosztály kínál neki, ezen felül képes meghalni, lekérdezhető, hogy él-e még és

- **Ősosztály**: Player
- **Attribútumok**
  - **+ alive: bool** : A játékos él-e még vagy nem.
- **Metódusok:**
  - **+ void die()**: Meghal a játékos.
  - **+ bool isAlive()**: lekérdezi életben van-e a játékos
  - **+ void EncounterTeacher(Teacher t1)**: ha találkozik a hallgató egy tanárral akkor vizsgálja, hogy milyen tárgyak vannak nála és mi a következménye a találkozásnak.

3.3.9 Teacher

- **Felelősség:** Az oktatókat reprezentáló osztály
- **Ősosztály**: Player
- **Metódusok:**
  - **+ void move()**: Felülírt move függvény
  - **+ void pickup(PickUp p)**: felülírt függvény, máshogy működik, mint a Student-é, hiszen vannak tárgyak, amit egy tanár nem vehet fel (pl. Logarléc).

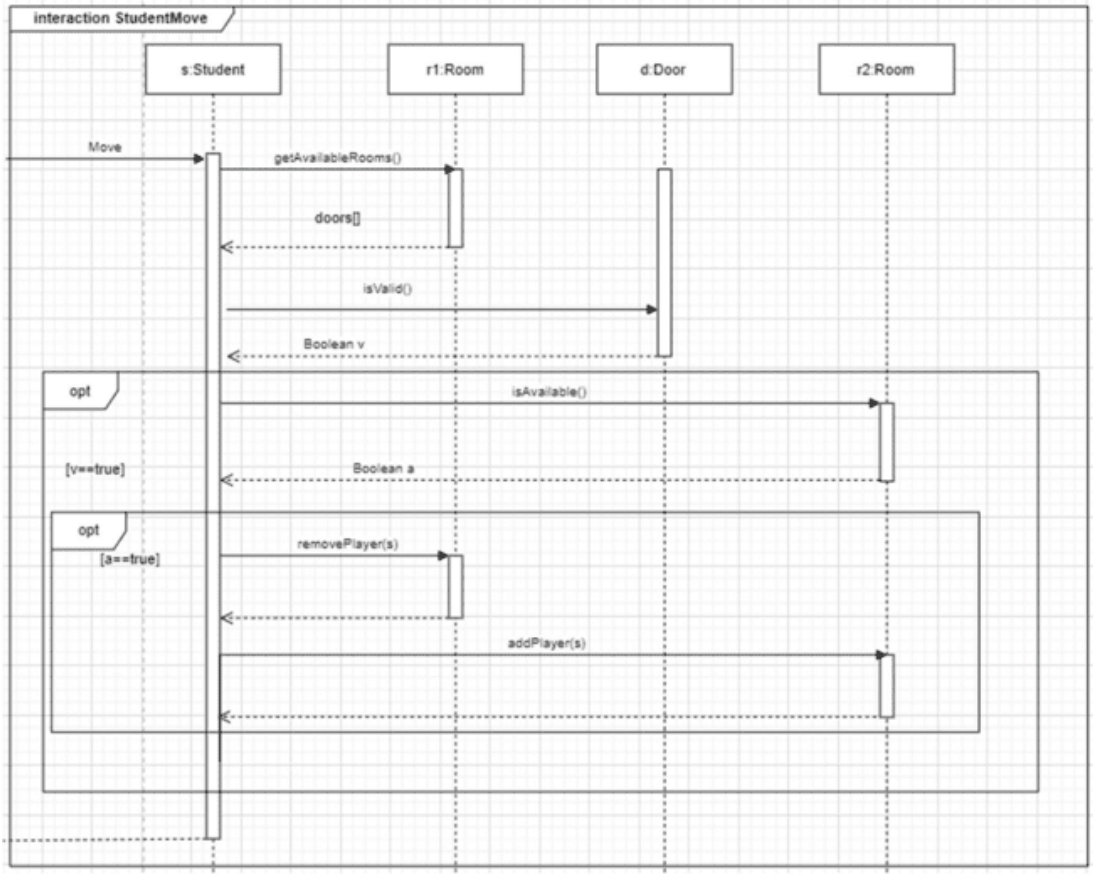
3.3.10 Transistor

- **Felelősség:** A Transistor objektumot reprezentáló osztály. A transistor egyszerre egy felvehető tárgy is, de amint párosítják és leteszik már ajtóként funkcionál.
- **Ősosztály:** Room
- **Interface:** PickUp
- **Attribútumok:**
  - **pair: Transistor** :a párban hozzákapcsolt transistort tárolja
- **Metódusok:**
  - **+ void onPutDown()**: Felüldefiniálja az interface függvényét.
  - **+ void onPickUp()**: Felüldefiniálja az interface függvényét.
  - **+Player getPlayer()**: Lekérdezi, hogy kinél van, vagy ki tette le az adott Transistort
  - **+bool makePair()**: öszekapcsol két Transistort, bool visszatérés, hogy sikeres-e az összekapcsolás
  - **+void deactivate()**: ha használva lettek, kikapcsolja az adott transistorokat

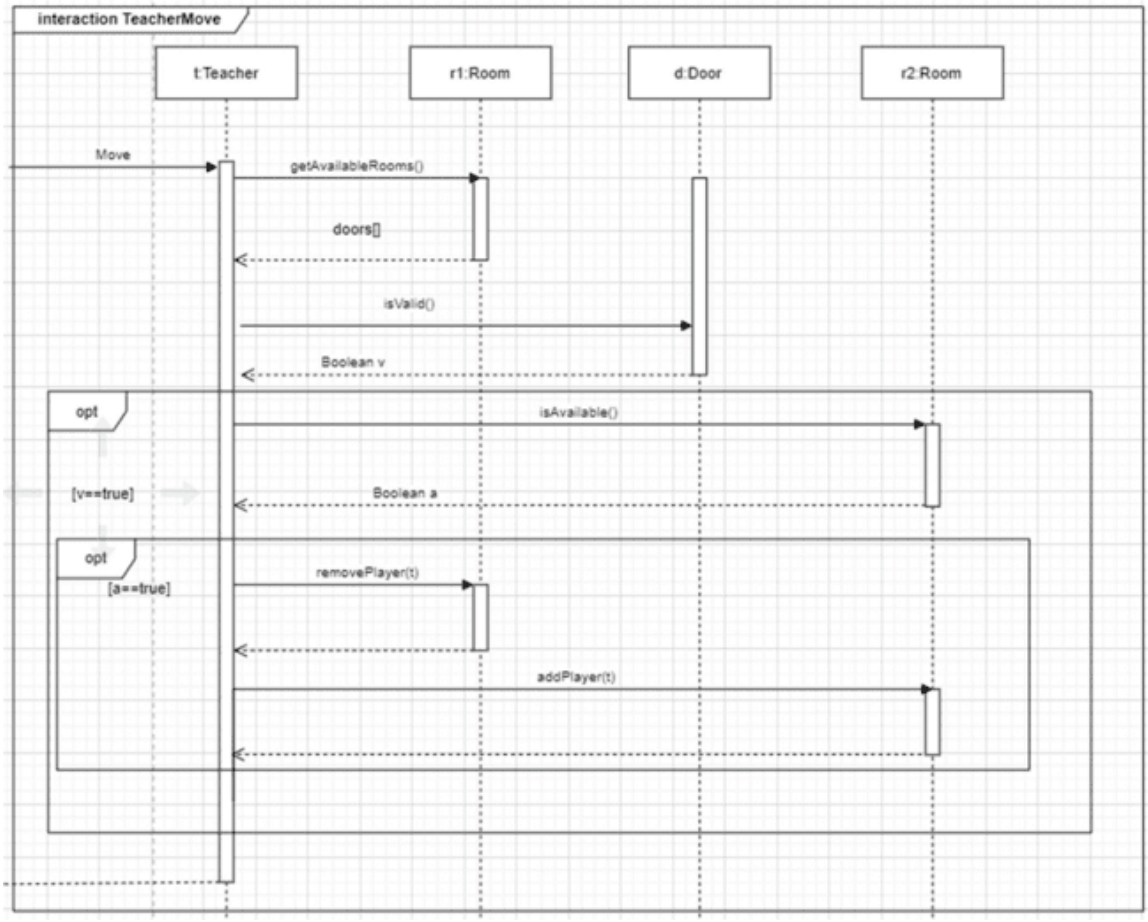
Megjegyzés: A maradék tárgyak csak az onPickUp() és az onPutDown() függvényeik megvalósításában térnek el, ezért itt nem listázzuk külön őket.

3.4 Szekvencia diagramok

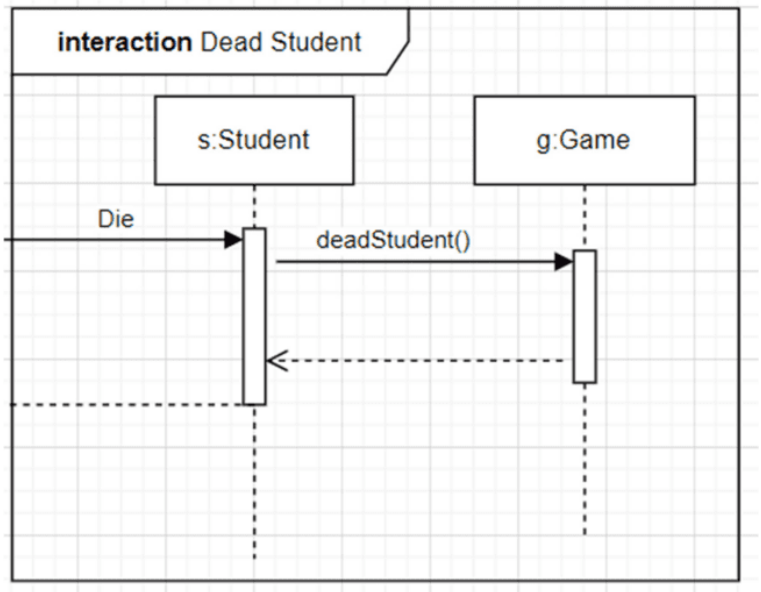
3.4.1.1 Student Move



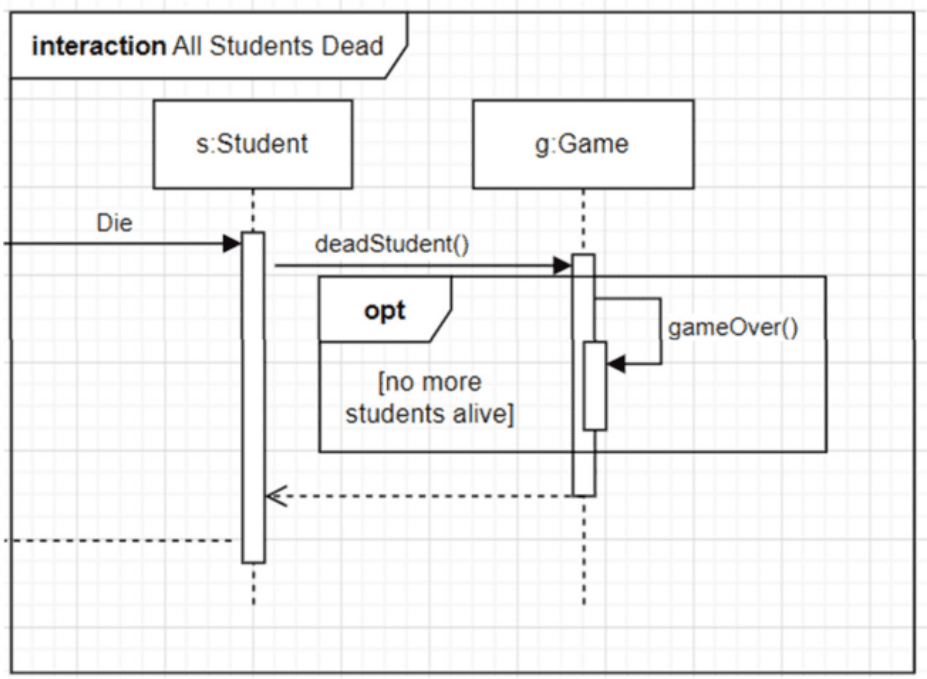
3.4.1.2 Teacher Move



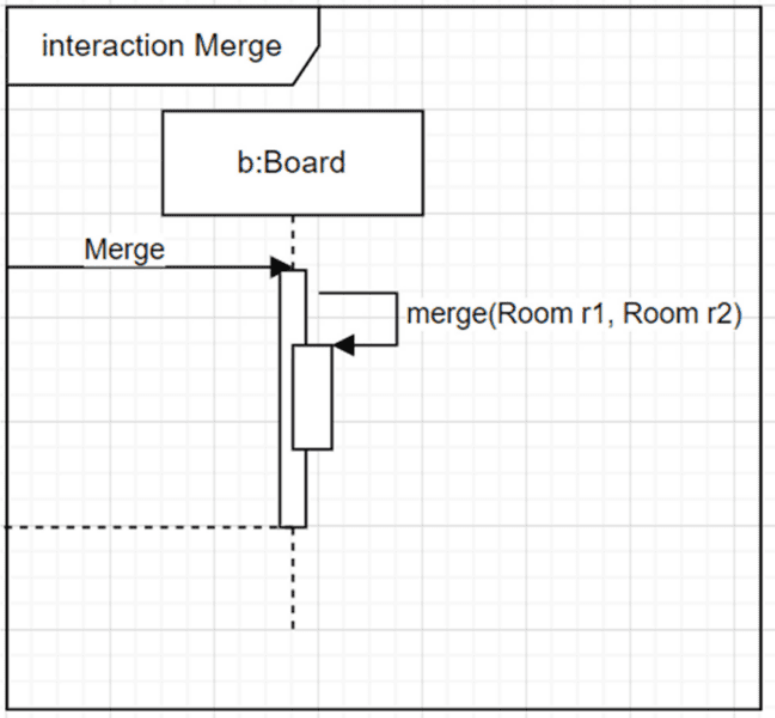
3.4.1.3 Dead Student



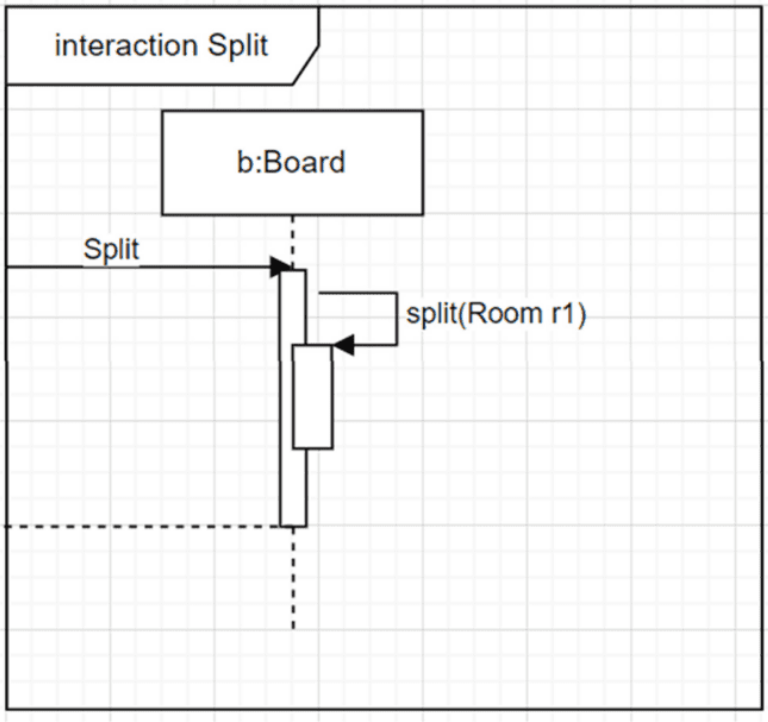
3.4.1.4 All Students Dead



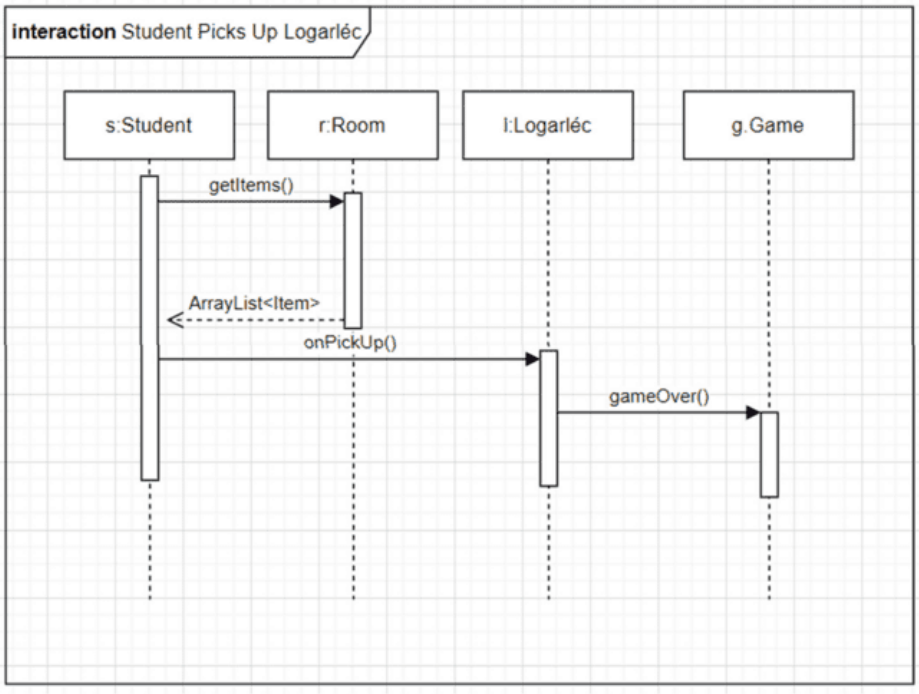
3.4.1.5 Room Merge



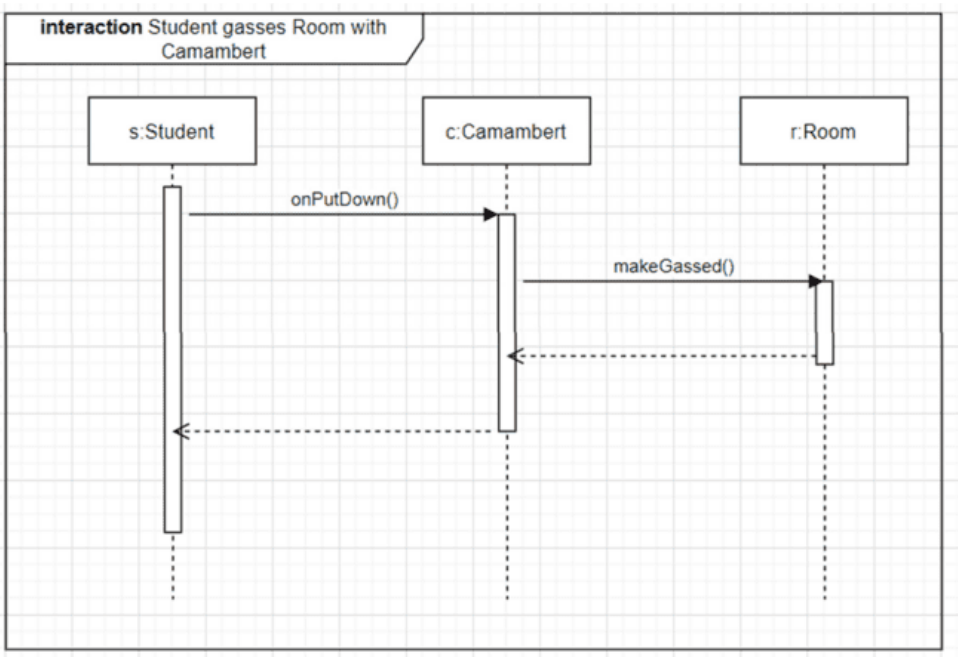
3.4.1.6 Room Split



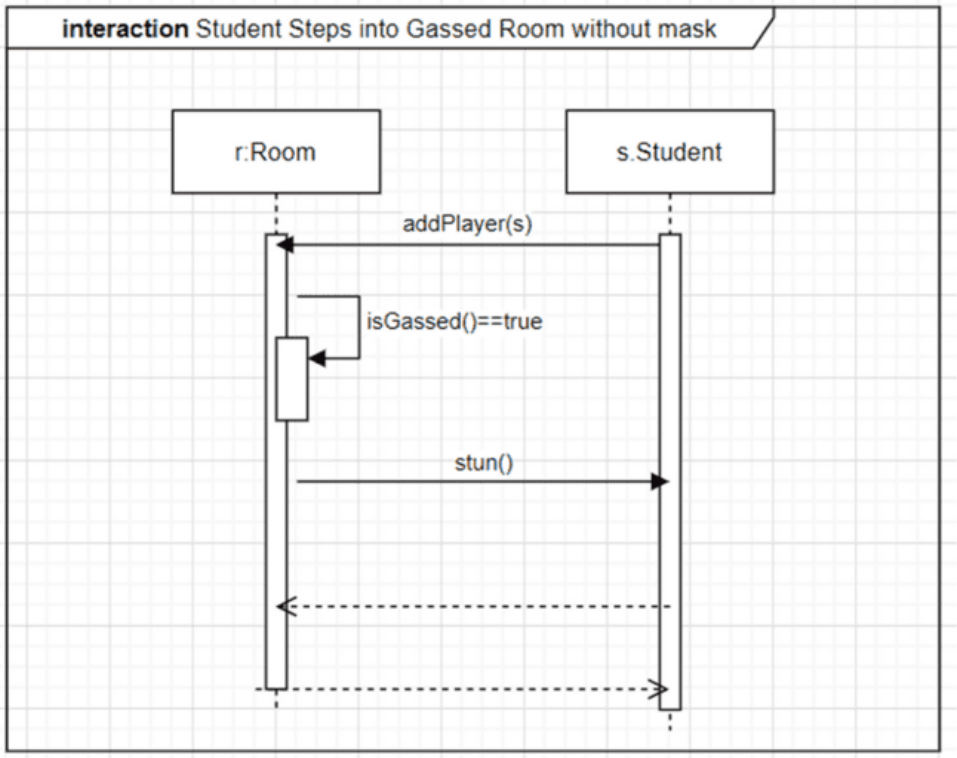
3.4.1.7 Student Picks Up Logarléc



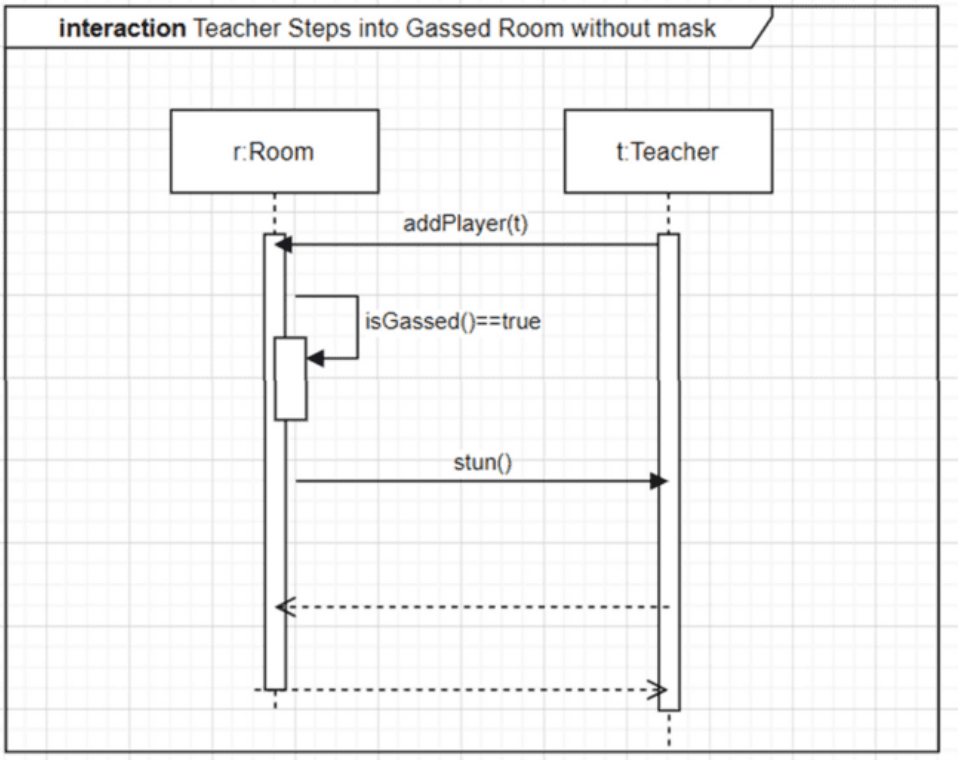
3.4.1.8 Camambert Use



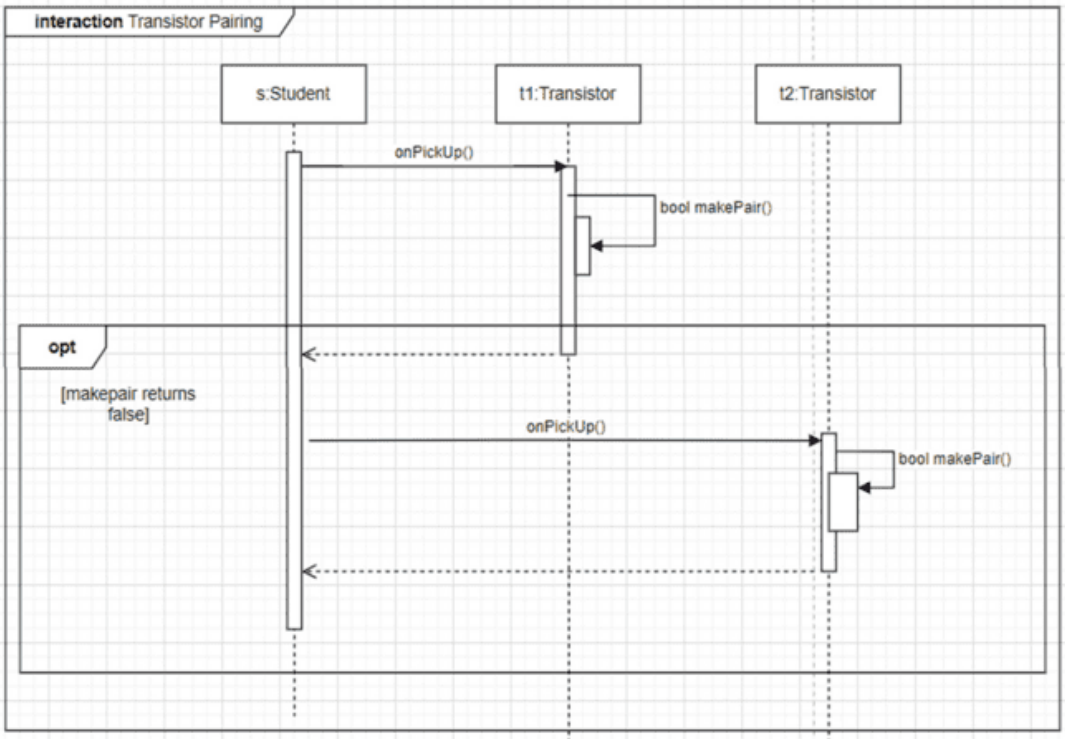
3.4.1.9 Student in Gassed Room without Mask



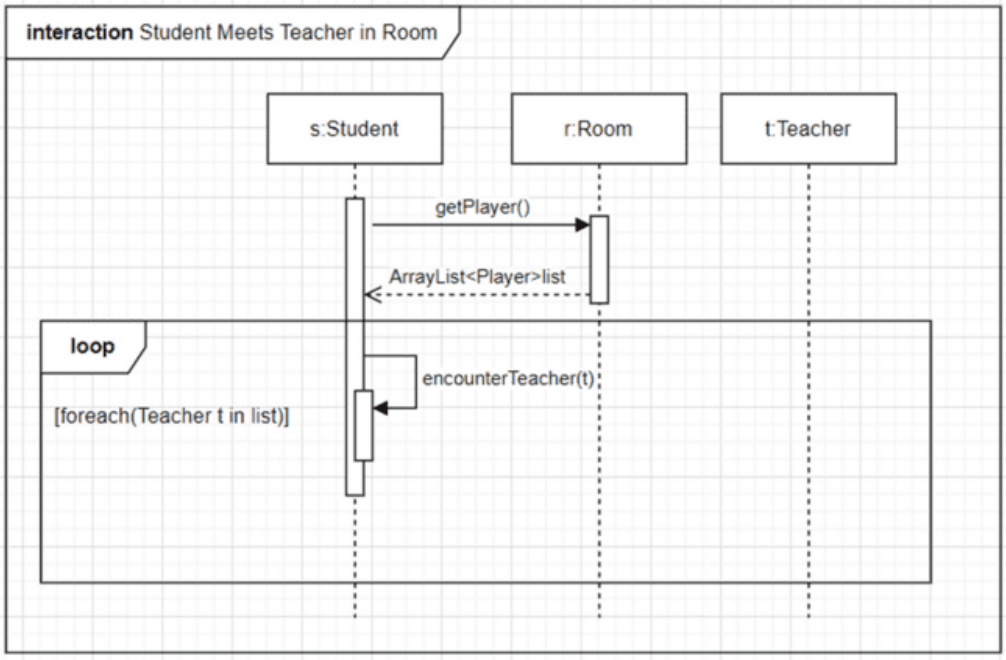
3.4.1.10 Teacher in Gassed Room Without Mask



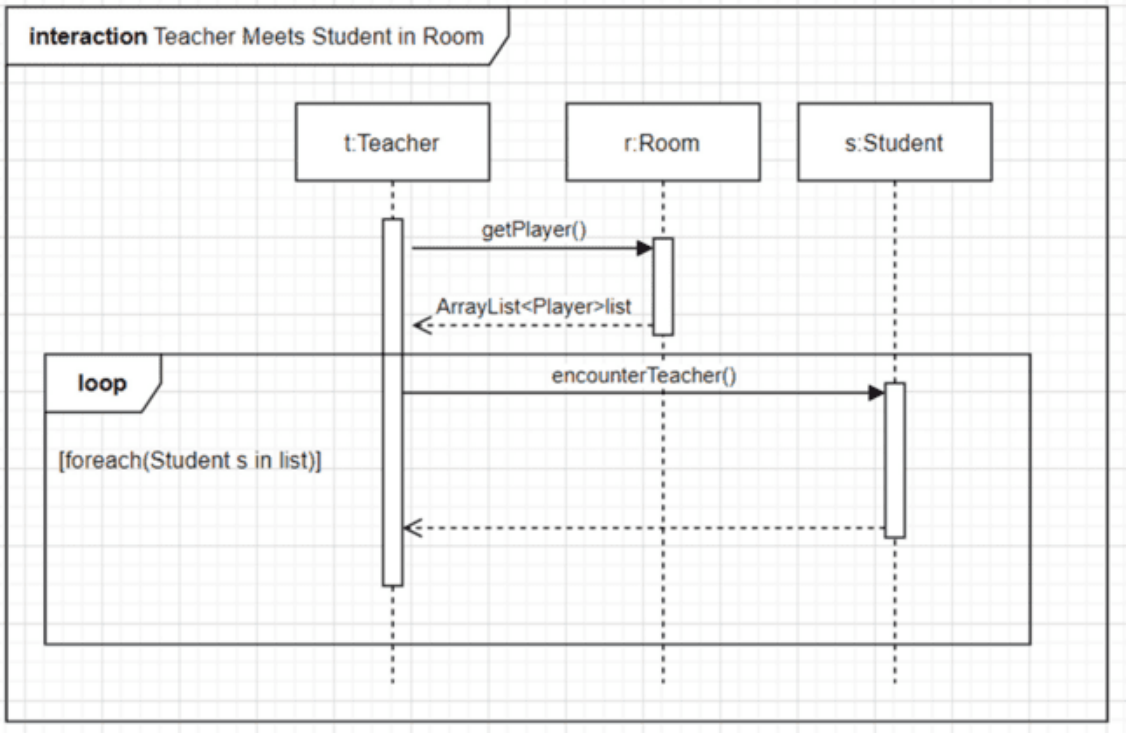
3.4.1.11 Transistor Pairing



3.4.1.12 Student Meets Teacher in Room

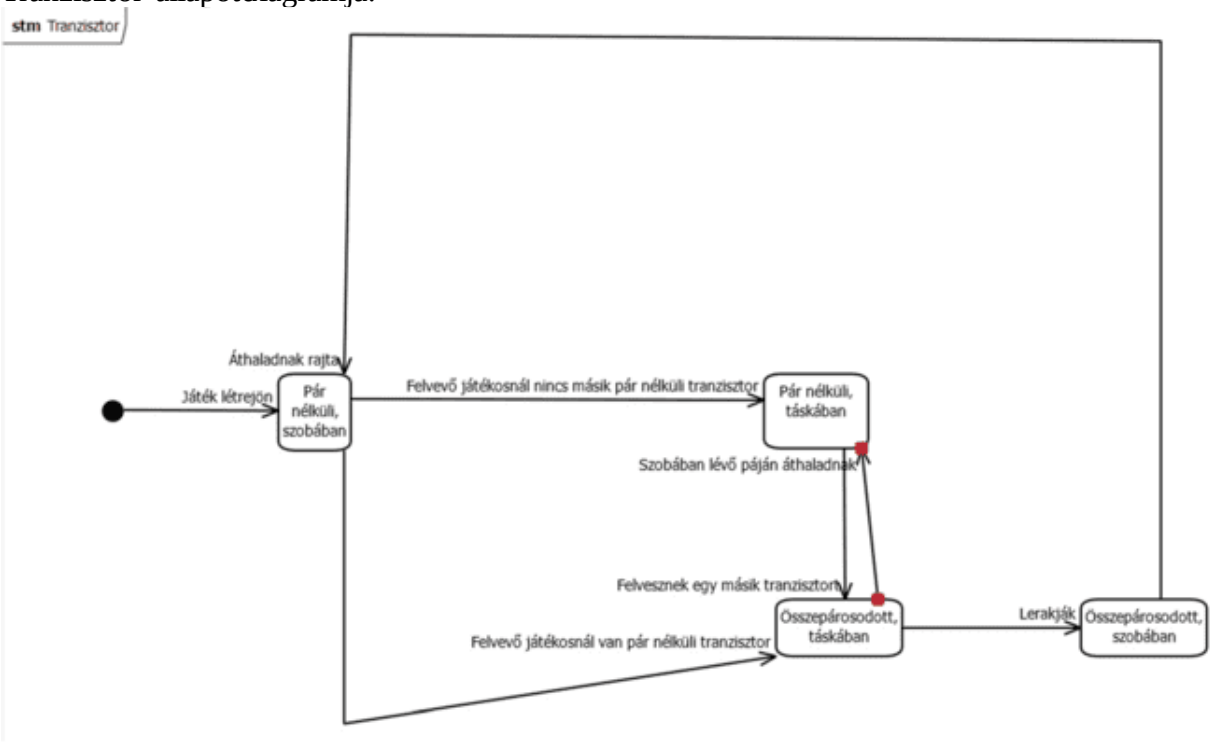


3.4.1.13 Teacher Meets Student

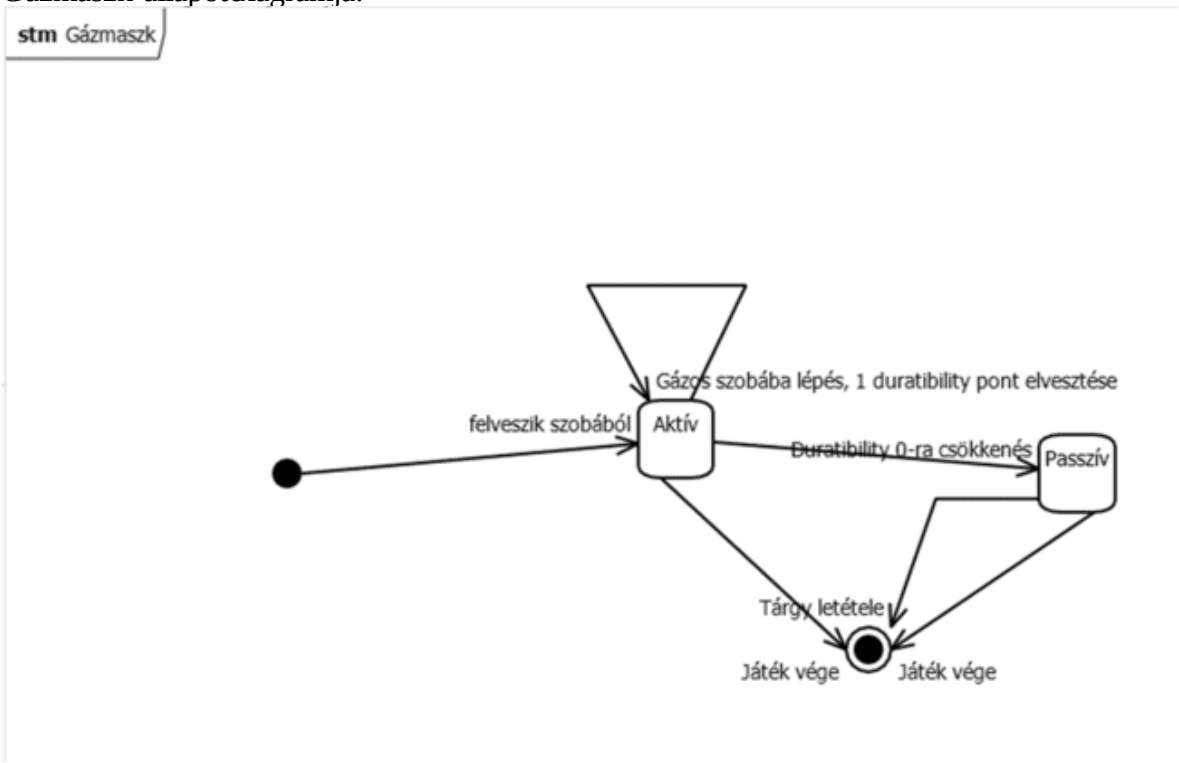


3.5 State-chartok

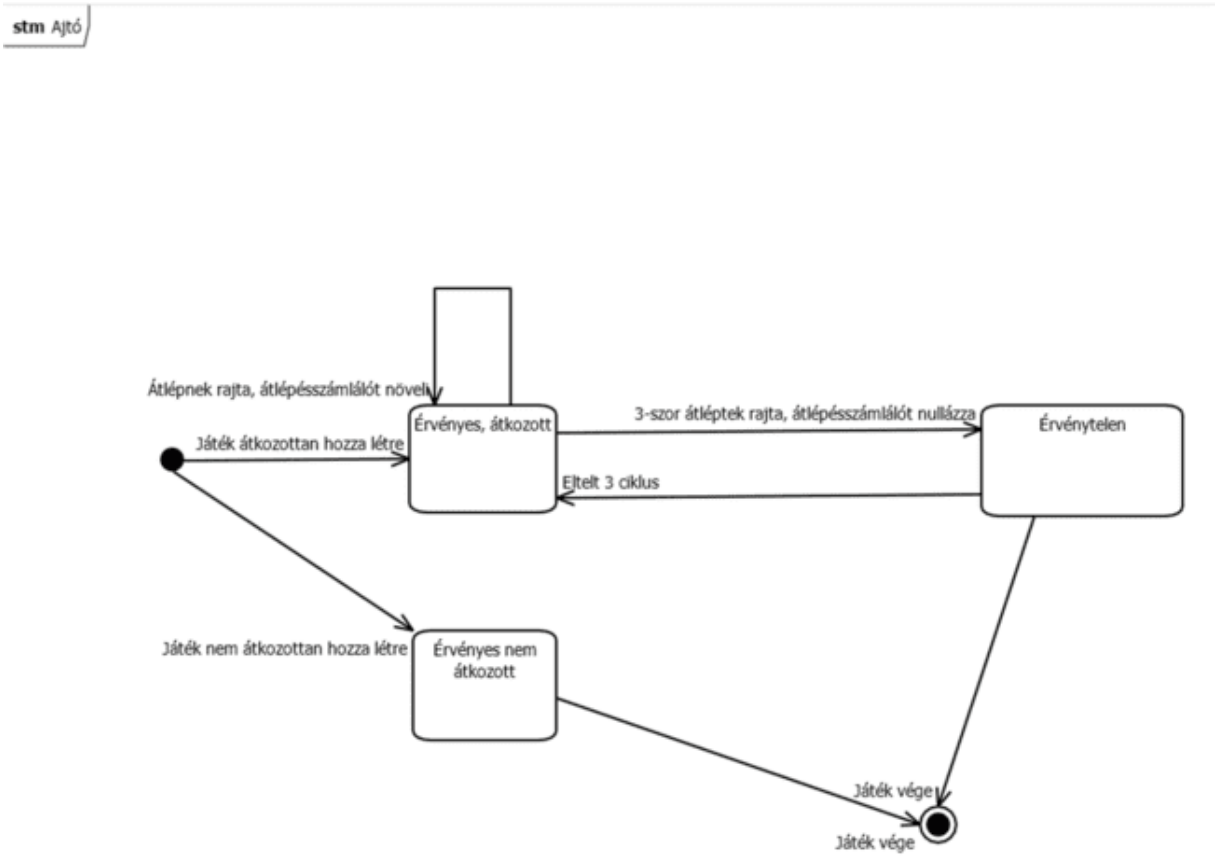
Tranzisztor állapotdiagramja:



Gázmaszk állapotdiagramja:

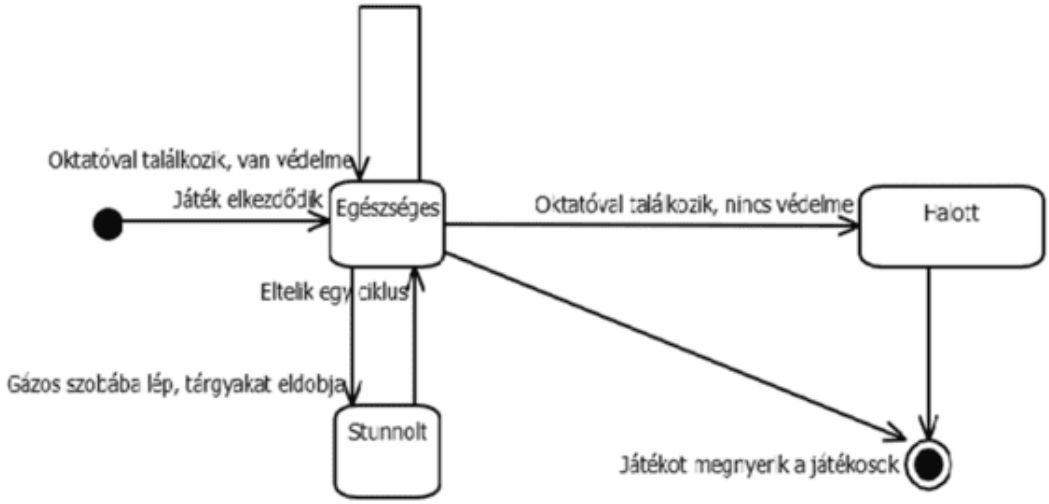


Ajtó állapotdiagramja:

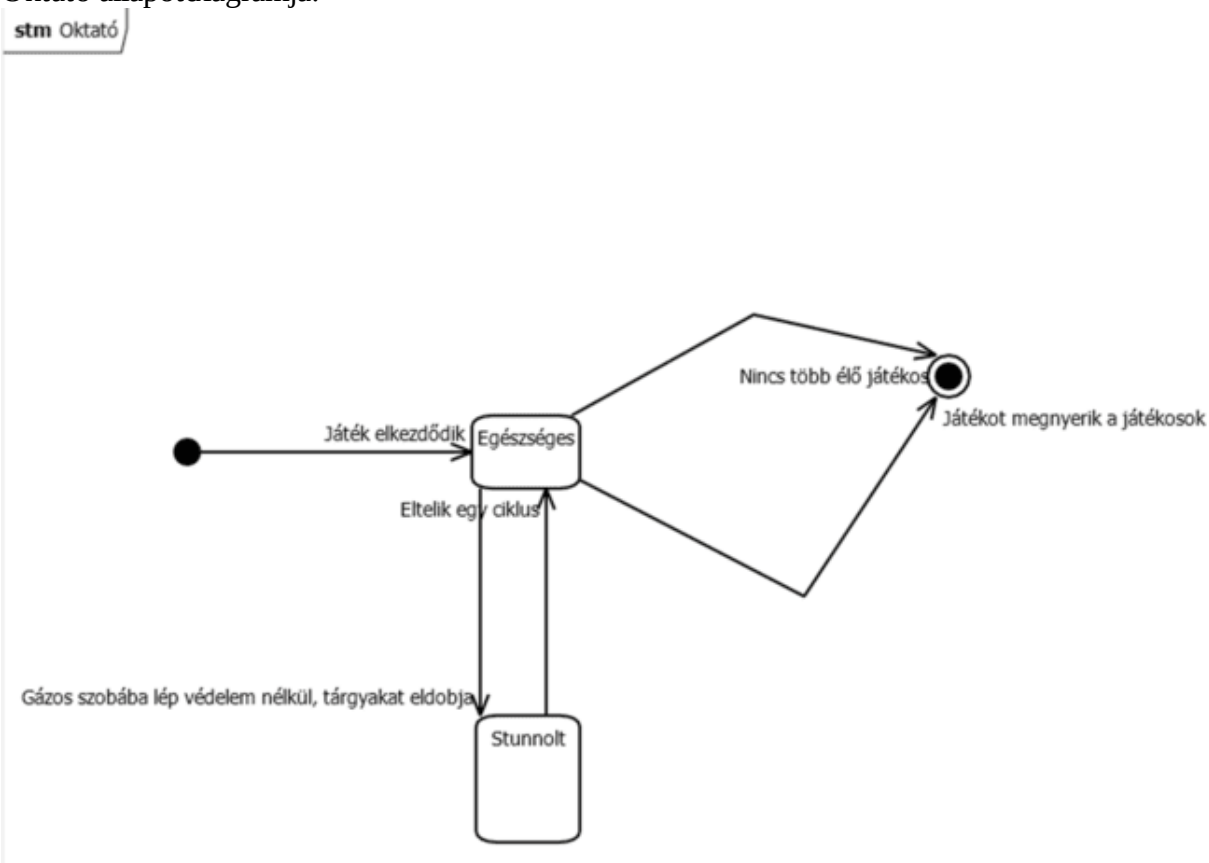


Szoba állapotdiagramja:

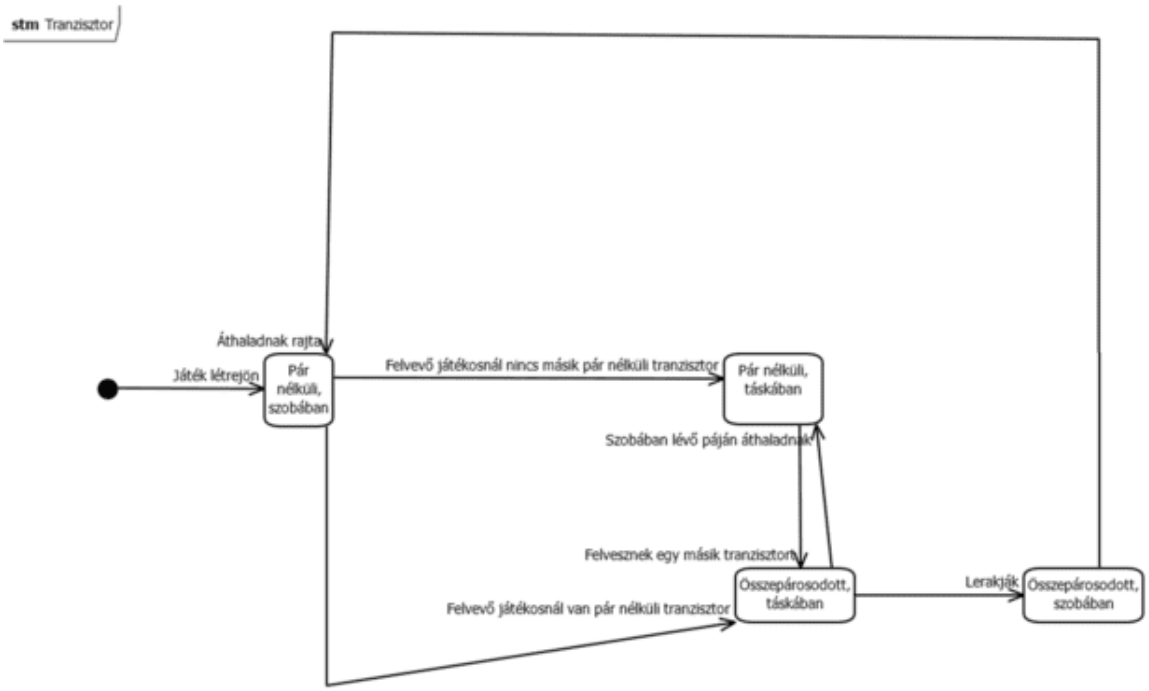
Hallgató állapotdiagramja:



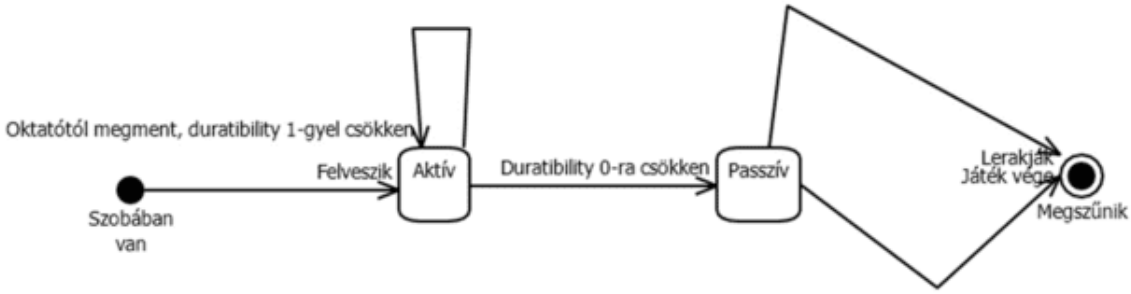
Oktató állapotdiagramja:



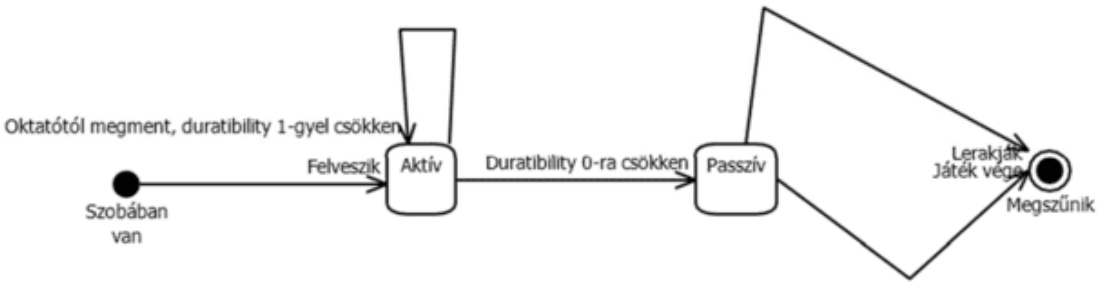
Tranzisztor állapotdiagramja:



TVSZ állapotdiagramja:



Söröspohár és táblatörlő állapotdiagramja (Mivel állapotaikat tekintve a 2 tárgy azonos, egy állapotdiagramon valósítottam meg őket.):



3.6 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2024.02.28. 19:00	1,5 óra	Fodor A. Fodor D. Földi Ludányi Mikola	Értekezlet: A csapat megbeszélte a feladattal kapcsolatos teendőket, objektumokat. Döntés: Fodor D. elkészíti az osztálydiagrammot, Földi az osztályok leírását, Mikola a State-chart-okat, Ludányi az objektum katalógust, Fodor A. a szekvenciadiagrammokat
2023. 03.01. 7:00	3 óra	Fodor D.	3.2-es Osztálydiagram elkészítése az előzetesen megbeszéltek alapján
2023.03.01 10:00	2,5 óra	Földi	3.3 (Osztályleírások) vázlatos leírása
2023.03.01 14:00	2 óra	Fodor A. Fodor D.	3.2-es Osztálydiagram összeállítása, tervezése, szerkesztése. 3.4-es Szekvencia diagrammok tervezése, elkészítése, szerkesztése. Földi, Fodor D. munkájának ellenőrzése.
2023.03.01 23:00	2,5 óra	Ludányi	3.1-es rész elkészítése
2023.03.02 10:00	1,5 óra	Fodor A. Fodor D.	Szekvenciadiagrammokkal kapcsolatos konzultáció, diagrammok készítése.
2023.03.02 13:00	3 óra	Fodor A.	Szekvenciadiagrammok befejezése.
2023.03.02 16:40	1,5 óra	Földi Fodor A. Fodor D. Ludányi Mikola	Kisebb csapatmegbeszélés a munka közben felmerülő kérdések tisztázására
2023.03.03 05:00	1 óra	Fodor D.	Csapatmegbeszélés alapján az Osztálydiagram szerkesztése, átalakítása.
2023.03.03 9:00	1,5 óra	Fodor A.	Megbeszélés alapján szekvenciadiagrammok átdolgozása, ötletek összefoglalása.
2023.03.03 17:00	2,5 óra	Mikola	Állapotdiagramok megalkotása és

			dokumentumba való feltöltése.
2023.03.03 18:00	3 óra	Fodor A. Fodor D.	Szekvenciadiagrammok és osztálydiagramm feltöltése. Dokumentumbeli hibák javítása, 3.3-as, 3.1-es rész befejezése.

4. Analízis modell (II. változat)

6 – ripgyork

Konzulens:
Ádám Zsófia

Csapattagok

Fodor Attila	EUGN1B	afodor998@gmail.com
Fodor Dávid	D02DBR	dfodor999@gmail.com
Földi Balázs	AB8Y3S	fbalu8@gmail.com
Ludányi Barnabás	V5PWP4	ludanyib2003@gmail.com
Mikola Bálint István	**TCV0Y9**	**mikola.balint.istvan@gmail.com (kapcsolattartó)**

2024.03.11

4. Analízis modell kidolgozása

4.1 Objektum *katalógus*

4.1.1 Beer

Ennek az osztálynak tárolnia kell a sör tartósságát, valamint, hogy aktív állapotban van-e. Felelőssége elfogadni, ha felvették, valamint, ha használják, ezen kívül a saját tartósságának állapotát is csökkentenie kell a megfelelő időpontokban.

4.1.2 Board

Ennek az objektumnak a felelőssége a pálya felépítése, valamint a játék közbeni menedzselése. Jeleznie kell, ha vége a játéknak, valamint képesnek kell lennie a szobákat hozzáadni, eltávolítani. Ezen felül felelőssége a szobakettéosztódásra valamint az szobaegyesülésre kiválasztott szoba/szobák-nak való jelzés, hogy hajtsák végre az adott műveletet.

4.1.3 Camembert

Ennek az osztálynak a felelőssége, hogy elfogadja a camembert letételét, hatása a letételtől kezdve érvényes.

4.1.4 CursedRoom

Ennek az objektumnak a feladata az átkozott szobák kezelése. Felelőssége tudni, hogy mely ajtók nyílnak ide, valamint, hogy innen hova nyílnak ajtók

4.1.5 CycleUsage

Az objektum feladata a játékmenet, pontosabban a játék egyes köreinek menedzselése, valamint azon osztályok működésének támogatása, melyeket a körök száma befolyásol. Tárolnia kell azt is, hogy hányadik körnél tartunk jelenleg.

4.1.6 Door

Az ajtó objektum feladata számontartani, hogy hányszor haladnak át rajta, hogy érvényes-e éppen egyik, vagy másik irányba, esetleg teljesen le van e zárva, valamint, hogy honnan és hogy hova nyílik. Ezen kívül az ajtó feladata a játékosokat egyik szobából a másikba áttenni.

4.1.7 Logarlec

Ezen objektum a játék legfontosabb elemét reprezentálja. Ez egy, a játékban résztvevő tárgyak egyike. Ez a tárgy mind felett áll, hiszen ennek felvétele egy hallgató által egyben a játék végét is jelzi, hiszen ez a játék célja, hogy ezt megszerezzék, mielőtt az oktatók megtalálnának minden hallgatót.

4.1.8 Mask

Ennek az osztálynak tárolnia kell a maszk tartósságát. Felelőssége elfogadni, ha felvették, innentől kezdve lehet használni a gázos szoba elleni védekezésben. Ezen kívül a saját tartósságának állapotát is csökkentenie kell a megfelelő időpontokban.

4.1.9 Room

Ennek az objektumnak a felelőssége a szobákkal kapcsolatos legtöbb információ tárolása, valamint a szobákhoz tartozó legtöbb metódus végrehajtása vagy kezdeményezése. Tárolnia kell a szoba kapacitását, a szoba egyedi azonosítóját, valamint azt is, hogy az adott szoba gázos-e vagy sem. A szobákban különböző ajtók is találhatók, melyeken keresztül a játékosok másik szobákba juthatnak. A játékosok a szobák segítségével vehetnek fel és dobhatnak el tárgyakat, valamint az egyes játékosok érkezését, távozását és esetleges halálát is tudnia kell kezelni, támogatni. Az ajtó hozzáadást is meg kell lehessen valósítani, valamint a szoba képes kell legyen az összeolvadás és a kettéosztódás megvalósítására. Jeleznie kell ezen felül, ha egy hallgató felvette a logarlécet és ezáltal megnyerték a játékot, valamint az ajtók nyitott vagy csukott állapotáról is tudnia kell.

4.1.10 Student

Ezen objektum felelőssége a játékos birtokában lévő tárgyak nyilvántartása, valamint minden játékos rendelkezik egy egyedi azonosítóval is. A játékosok mozgásának, valamint tárgyak felvételének és letételének megvalósítása szintén ennek az objektumnak a felelősségkörébe tartozik. Tudnia kell azt is, hogy az adott játékos életben van-e.

4.1.11 Tablatorlo

Ennek az osztálynak tárolnia kell a táblatörlő tartósságát. Felelőssége elfogadni, ha letették, innentől kezdve él a hatása, ezen kívül a saját tartósságának állapotát is csökkentenie kell a megfelelő időpontokban.

4.1.12 Teacher

Ezen objektum felelőssége a játékos birtokában lévő tárgyak nyilvántartása, valamint minden játékos rendelkezik egy egyedi azonosítóval is. A játékosok mozgásának, valamint tárgyak felvételének és letételének megvalósítása szintén ennek az objektumnak a felelősségkörébe tartozik. Tudnia kell azt is, hogy az adott játékos épp bénított állapotban van-e, vagy sem, valamint, ha hallgatóval egy szobába kerül, a megölésük is ezen objektum feladata.

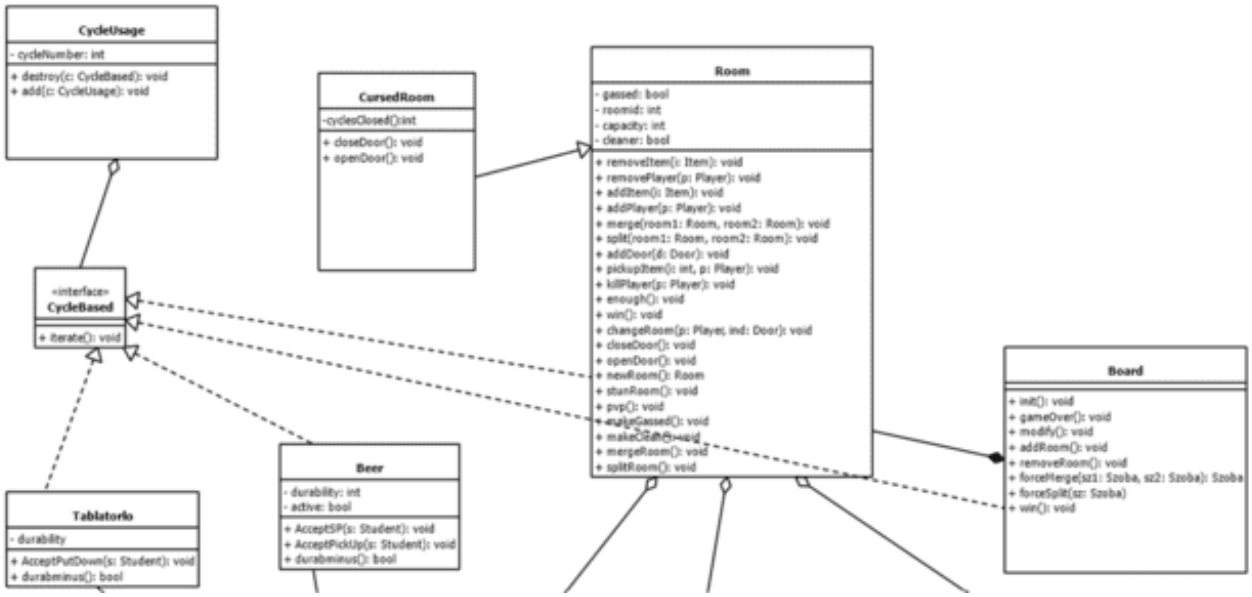
4.1.13 Transistor

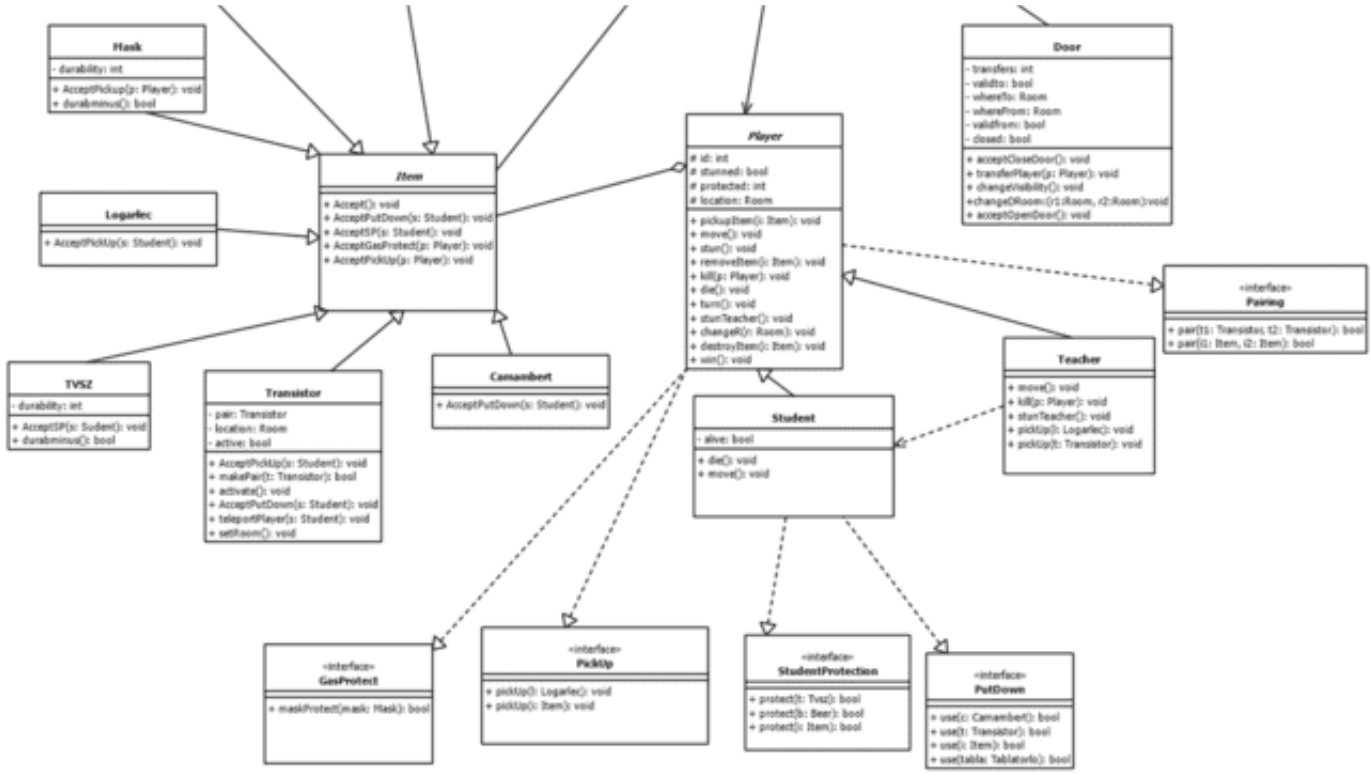
Ezen objektum egy, a játék során felvehető objektumot reprezentál. A transistorok kezdetben felvehető tárgyak, majd összekapcsolás/összepárosítás után ajtóként viselkednek a két szoba közt, amelybe lehelyezték őket. Tudnia kell magáról mely szobákban tették le egyik, valamint másik párt, valamint ismernie kell a párját is. Tudnia kell, hogy aktív állapotban van-e, ezen kívül a párosodását is magának kell kezelnie, és a játékos teleportálását is neki kell megvalósítani.

4.1.14 TVSZ

Ennek az osztálynak tárolnia kell a TVSZ tartósságát. Felelőssége a felvételét és használatát elfogadni, ezen kívül a saját tartósságának állapotát is csökkentenie kell a megfelelő időpontokban.

4.2 Statikus struktúra diagramok


CycleUsage
- cycleNumber: int
+ destroy(c: CycleBased): void
+ add(c: CycleUsage): void
CursedRoom
-cyclesClosed():int
+ closeDoor(): void
+ openDoor(): void
Room
- gassed: bool
- roomid: int
- capacity: int
- cleaner: bool
+ removeItem(i: Item): void
+ removePlayer(p: Player): void
+ addItem(i: Item): void
+ addPlayer(p: Player): void
+ merge(room1: Room, room2: Room): void
+ split(room1: Room, room2: Room): void
+ addDoor(d: Door): void
+ pickupItem(i: int, p: Player): void
+ killPlayer(p: Player): void
+ enough(): void
+ win(): void
+ changeRoom(p: Player, ind: Door): void
+ closeDoor(): void
+ openDoor(): void
+ newRoom(): Room
+ stunRoom(): void
+ pvp(): void
+ makeGassed(): void
+ mergeRoom(): void
+ splitRoom(): void
«interface»
CycleBased
+ iterate(): void
Board
+ init(): void
+ gameOver(): void
+ modify(): void
+ addRoom(): void
+ removeRoom(): void
+ forceMerge(sz1: Szoba, sz2: Szoba): Szoba
+ forceSplit(sz: Szoba)
+ win(): void
Tablatorlo
- durability
+ AcceptPutDown(s: Student): void
+ durabminus(): bool
Beer
- durability: int
- active: bool
+ AcceptSP(s: Student): void
+ AcceptPickUp(s: Student): void
+ durabminus(): bool



4.3 Osztályok leírása

4.3.1 Beer

- **Felelősség**

A sört reprezentáló osztály

- **Ősosztály**
  - Item
- **Interfészek**
  - CycleBased
- **Attribútumok**
  - -durability: int - Megszünéséig hátralévő körök száma
  - -active: bool
- **Metódusok**
  - +acceptPickUp(s: Student): void - Játékoshoz kerül
  - +acceptSP(s: Student): void - A Student-től érkező requestre válasz
  - +durabMinus(): bool - Csökkenti a durability-t eggyel, ha nem sikerül hamis

4.3.2 Board

- **Felelősség**

Ez az osztály felelős a szobák és a játéktér reprezentálásáért. Nyilvántartja a szobákat és azokat összevonhatja és szétszedheti. Felépíti a pályát, módosítja és karbantartja.

- **Asszociációk**
  - Room: A szobákkal van kapcsolatban, ezeket tárolja
- **Interfészek**
  - CycleBased
- **Metódusok**
  - +Init(): void - Inicializálja pályát
  - +GameOver(): void - Befejezi a játékmenetet
  - +Modify(): void - Módosítja a játékteret
  - +addRoom(r: Room): void - Új szobát ad a játékhoz
  - +removeRoom(r: Room) - Törli a megadott szobát
  - +forceMerge(r1: Room, r2: Room): void – Jelzi a nagyobb kapacitású szobának szobának, hogy mergelni kéne
  - +forceSplit(r: Room): void – Jelzi az ‘r’ szobának, hogy split függvényt végre kéne hajtani
  - +win(): void – jelzi, ha a játék nyereséggel ért véget

4.3.3 Camambert

- **Felelősség**

A camambert sajtot reprezentáló osztály

- **Ősosztály**
  - Item
- **Metódusok**
  - +acceptPutDown(s: Student): void - Válasz a kérésre, a tárgy letevődik

4.3.4 CursedRoom

- **Felelősség**

Az osztály az átkozott szobák reprezentálásért felel.

- **Ősosztály**
  - Room
- **Metódusok**
  - +closeDoor(): void - Bezár minden ajtót
  - +openDoor(): void – Kinyitja őket

4.3.5 CycleBased Interfész

- **Felelősség**

Az interfész a körönként változó objektumokért felelős, ezeken a körönkénti események jelzéséért (kör léptetése) felel

- **Metódusok**
  - +iterate(): void - léptet

4.3.6 CycleUsage

- **Felelősség**
  - Az osztály felelős azon osztályok helyes működéséért, amiket valamilyen módon befolyásol a ciklusok, vagy körök múlása
- **Asszociációk**
  - Cyclebased: Ezen interfészt megvalósító objektumokat tárol
- **Attribútumok**
  - -cycleNumber: int – Éppen hanyadik ciklus száma
- **Metódusok**
  - +destroy(c: cycleBased): void - Megszünteti az objektumot
  - +add(c: cycleUsage): void - interfész tárolás miatt kell hozzáadás

4.3.7 Door

- **Felelősség**
  - Az ajtókat reprezentáló osztály. Egy ajtó nyilvántartja a szobákat, ahová rajta keresztül el lehet jutni.
- **Asszociációk**
  - Room: Azon szobák, melyek között az ajtó vezet
- **Attribútumok:**
  - - transfers: int – Megadja, hogy az ajtón eddig hányszor mentek keresztül
  - -roomA: Room –Egyik szoba
  - -roomB: Room - Másik szoba
  - -validA: bool - Járható-e B-ből A irányba az ajtó
  - -validB: bool - Analóg
- **Metódusok**
  - +acceptCloseDoor(): void - bezárja magát
  - +transferPlayer(p: Player): void - Átjuttat egy játékost a másik szobába
  - +changeVisibility(): void – A láthatóságot kapcsolja
  - +changeDRoom(r1: Room, r2: Room): void - átteszi a játékost egy másik szobába, önmagán keresztül
  - +acceptOpenDoor():void - ajtó kinyitását elfogadó függvény

4.3.8 GasProtect Interfész

- **Felelősség**

A gáz elleni automatikus védelemben játszik szerepet

- **Metódusok**
  - +maskProtect(m: Mask): void - Aktiválja a maszkot

4.3.9 Item

- **Felelősség:**

A tárgyakat reprezentáló osztályok absztrakt ősosztálya.

- **Asszociációk**
  - Room
  - Player
- **Metódusok**
  - +accept(): void - ősosztály függvény, történések elfogadására
  - +acceptPutDown(s: Student): void – A Student-től érkező requestre válasz
  - +acceptSP(s: Student): void - A Student-től érkező requestre válasz
  - +acceptGasProtect(p: Player): void - A Player-től érkező requestre válasz
  - +acceptPickUp(p: Player): void - A Player-től érkező requestre válasz

4.3.10 Logarlec

- **Felelősség**

A Logarlécet reprezentáló osztály

- **Ősosztály**
  - Item
- **Metódusok**
  - +acceptPickUp(s: Student): void - Játékoshoz kerül

4.3.11 Mask

- **Felelősség**

Az FFP2 maszkot reprezentáló osztály

- **Ősosztály**
  - Item
- **Attribútumok**
  - -durability: int - Megszünéséig hátralévő körök száma
- **Metódusok**
  - +acceptPickUp(s: Student): void - Játékoshoz kerül
  - +durabMinus(): bool - Csökkenti a durability-t eggyel, ha nem sikerül hamissal tér vissza

4.3.12 Pairing

- **Felelősség**

Itemek párosításáért felel. Metódusai igazzal térnek vissza, ha ez sikeres, hamissal, ha nem.

- **Metódusok**
  - +pairTrans(t1: Transistor, t2: Transistor): bool - párosít két paraméterként kapott transistort
  - +pairItem(i1: Item, i2: Item): bool - két paraméterként kapott Itemet párosít

4.3.13 PickUp Interfész

- **Felelősség**

Tárgyak felvételében játszik szerepet, jelez a megfelelő osztály felé az állapotváltozásról

- **Metódusok**
  - +pickUp(l: Logarlec): void - Logarléc felvétele, speciális esemény
  - +pickUp(i: Item): void - tárgy felvétele

4.3.14 Player

- **Felelősség**

A játékosokat (hallgató és oktató) reprezentáló osztályok absztrakt őse.

- **Asszociációk:**
  - Item: A játékos által tárolt felvehető objektumok kezeléséhez
- Interfészek
  - PickUp
  - GasProtect
  - Pairing
- **Attribútumok:**
  - # id: int - A játékost egyértelműen azonosító szám
  - # protected: bool - Védett-e a játékos
  - # stunned: bool - kábítva van-e a játékos
  - # location: Room – Az a szoba ahol a játékos van
- **Metódusok:**
  - +pickUp(i: Item): void – i felvétele
  - +move(): void - Játékos léptetése
  - +stun(): void - Játékos kábítása
  - +removeItem(i: Item): void – i tárgy eltávolítása a játékos eszköztárjából
  - +kill(p: Player): void – p játékos megölése
  - +turn(): void – a játékos körének megvalósítására használjuk
  - +stunTeacher(): void - Oktató kábítása
  - +changeRoom(r: Room): void – Szoba váltása
  - +destroyItem(i: Item): void – i tárgy törlése
  - +win(): void - játék megnyerését jelző függvény

4.3.15 PutDown Interfész

- **Felelősség**

Tárgyak lehelyezésében, aktiválásában játszik szerepet. Amennyiben sikeres a tárgy használata a függvényei igazzal térnek vissza.

- **Metódusok**
  - +use(c: Camambert): bool -
  - +use(t: Transistor): bool -
  - +use(t: Tablatorlo): bool -
  - +use(i: Item): bool -

4.3.16 Room

- **Felelősség**

A szobákat reprezentáló osztály, tárolja a benne lévő játékosokat (hallgatók, oktatók) és a benne lévő felvehető objektumokat.

- **Asszociációk**
  - Item: Tárolás
  - Player: Tárolás
  - Door: Tárolás
- **Interfészek**
  - CycleBased
- **Attribútumok**
  - - capacity: int - A szoba befogadóképességét adja, ez a játék elején kap egy alapértéket
  - - gassed: bool - Ha az értéke “true”, az azt jelenti, hogy a szoba mérgezett. Ekkor a benne lévő játékosok megkapják a specifikációban leírt effekteket, végrehajtódnak az akciók
  - - roomId: int - A szoba azonosító száma
  - -cleaner: bool – Van-e a szobában táblatörlő
- **Metódusok**
  - +removeItem(i: Item): void - Törli a szobából az adott tárgyat
  - +removePlayer(p: Player): void - Törli a szobából az adott játékost
  - +addItem(i: Item): void - Analóg
  - +addPlayer(p: Player): void - Analóg
  - +mergeRoom(r1: Room, r2: Room): void - Két szoba összeolvasztását végzi
  - +splitRoom(r1: Room, r2: Room): void – Saját maga szétválasztását végzi, tulajdonságait megkapja
  - +addDoor(d: Door): void - Ajtó hozzáadása a szobához
  - +removeDoor(d: Door): void - Ajtó eltávolítása
  - +pickUpItem(i: int, p: Player): void – Az i-edik indexű tárgyat megkapja p játékos
  - +killPlayer(p: Player): void – A játékost megöli, további körökben nem létezik
  - +enough(): bool – Van-e elég hely a szobában, hogy átkerülhessen a játékos
  - +win(): void – Amennyiben a logarléc felvételre került, akkor ezzel jelzi a Board felé, hogy nyertek a hallgatók
  - +changeRoom(p: Player, d: Door): void - Átküldi a p játékost a d ajtó másik felére
  - +closeDoor(d: Door): void - Bezárja a d ajtót
  - +openDoor(d: Door): void – Kinyitja a d ajtót
  - +newRoom(): Room - Új szoba inicializálása, ha egy szoba szétoszlik
  - +stunRoom(): void – Stunol mindenkit
  - +pvp(): void – Mindenki megpróbál mindenkit “megölni”, ezzel ellenőrizve, ki van a szobában és ellenőrizve kinél milyen tárgyat kell gyengíteni
  - +makeGassed(): void - elgázosodik a szoba
  - +makeClean():void - táblatörlős lett a szoba

4.3.17 Student

- **Felelősség**

A hallgatókat reprezentáló osztály.

- **Ősosztály**:
  - Player
- Interfészek
  - StudentProtection
  - PutDown
- **Attribútumok**
- **Metódusok:**
  - +die(): void - Meghal a játékos.
  - +move(): void - Felülírt move függvény
  - +putDown(i: Item): void - tárgyak letételéért felelős függvény

4.3.18 StudentProtection Interfész

- **Felelősség**

A hallagató automatikus védelmében játszik szerepet. HA szükséges, akkor értesíti a megfelelő objektumot. Amennyiben sikeresek, igazzal térnek vissza a függvényei.

- **Metódusok**
  - +protect(t: TVSZ): bool -
  - +protect(b: Beer): bool -
  - +protect(i: Item): bool -

4.3.19 Tablatorlo

- **Felelősség**

A táblatörlőt reprezentáló osztály

- **Ősosztály**
  - Item
- **Interfészek**
  - CycleBased
- **Attribútumok**
  - -durability: int - Megszünéséig hátralévő körök száma
- **Metódusok**
  - +acceptPutDown(s: Student): void - Válasz a kérésre, a tárgy letevődik
  - +durabMinus(): bool - Csökkenti a durability-t eggyel, ha nem sikerül hamis

4.3.20 Teacher

- **Felelősség:**

Az oktatókat reprezentáló osztály

- **Ősosztály**:
  - Player
- **Metódusok:**
  - + move(): void - Felülírt move függvény
  - +kill(p: Player): void – p játékos megölése
  - +stunTeacher(): void - Felülírt stunTeacher fv
  - +pickUp(l: Logarlec): void – Jelzi, hogy megpróbálja felvenni a Logarlécet, de nem engedi neki

4.3.21 Transistor

- **Felelősség:**

A Transistor objektumot reprezentáló osztály. A transistor egyszerre egy felvehető tárgy is, de amint párosítják és leteszik már ajtóként funkcionál.

- **Ősosztály:**
  - Item
- **Attribútumok:**
  - -pair: Transistor - a párban hozzákapcsolt transistort tárolja
  - -location: Room – A helyét tárolja, melyik szobában van
  - -active: bool - Aktív-e
- **Metódusok**
  - +acceptPickUp(s: Student): void - Játékoshoz kerül
  - +acceptPutDown(s: Student): void - Válasz a kérésre, a tárgy letevődik
  - +makePair(t: Transistor): bool - Összepárosít két tranzisztort
  - +activate(): void - Aktiválja a tranzisztort
  - +teleportPlayer(s: Student): void - Játékos mozgatása
  - +setRoom(r: Room) - Szoba beállítása

4.3.22 TVSZ

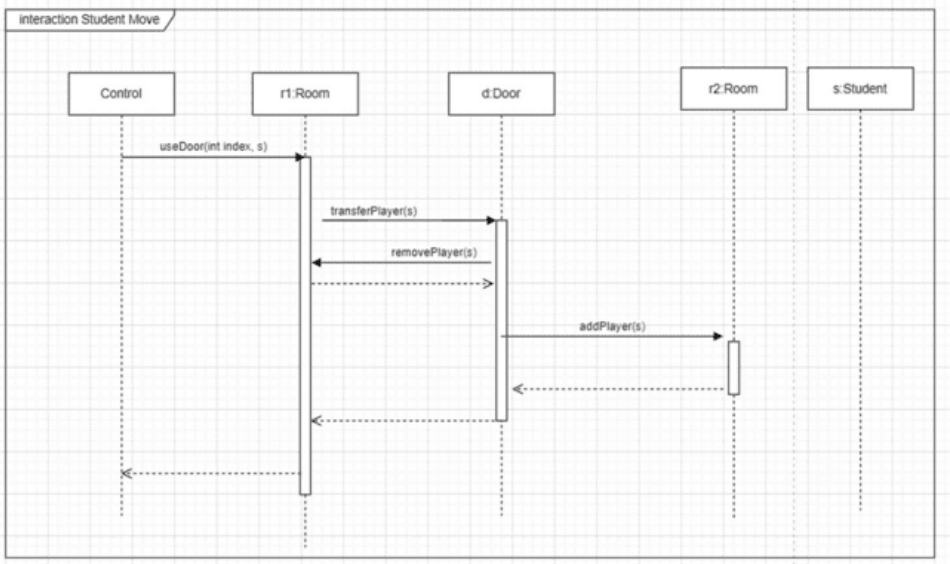
- **Felelősség**

A TVSZ-t reprezentáló osztály

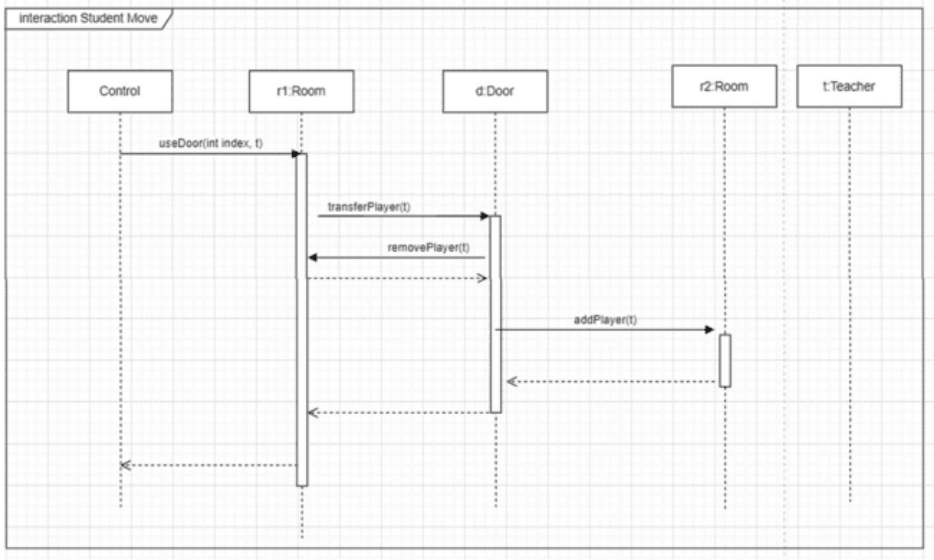
- **Ősosztály**
  - Item
- **Attribútumok**
  - -durability: int – Megszűnéséig hátralévő körök száma
- **Metódusok**
  - +acceptSP(s: Student): void – s hallgatót megvédi
  - +durabMinus(): bool - Csökkenti a durability-t eggyel, ha nem sikerül hamissal tér vissza

4.4 Szekvencia diagramok

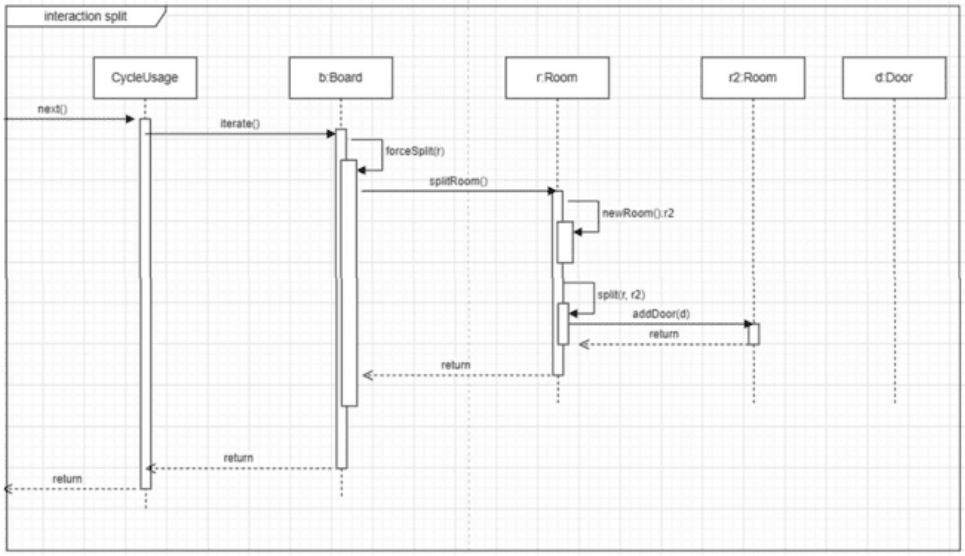
Student Move



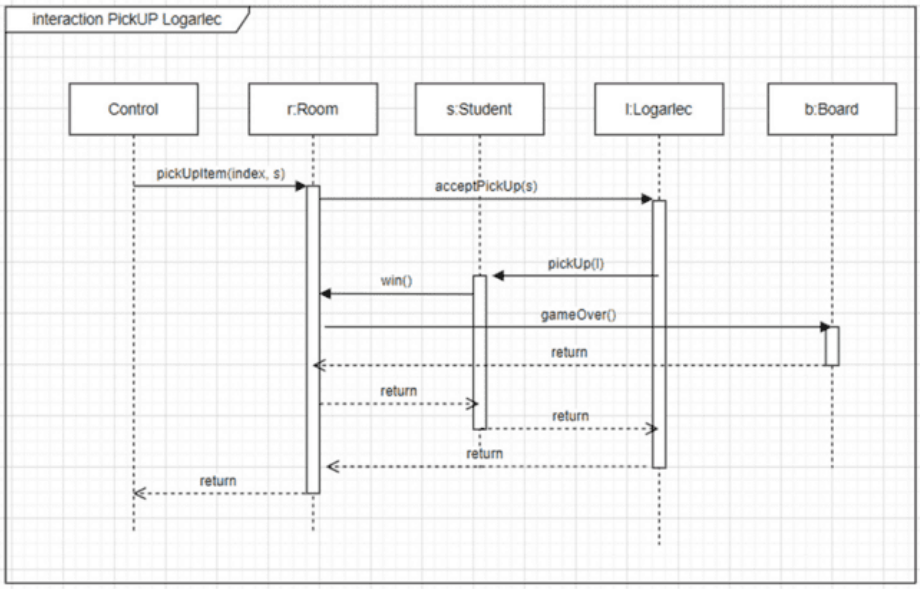
Teacher Move



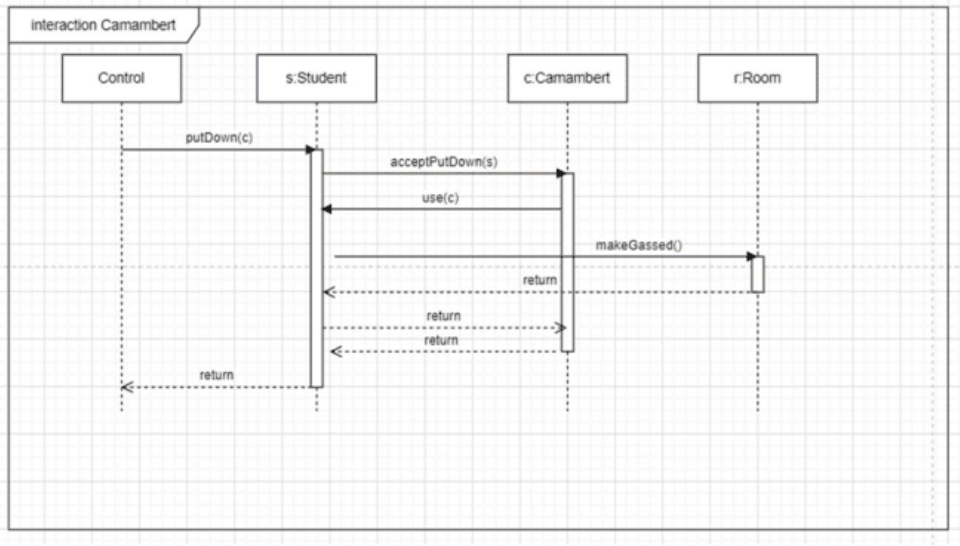
Room Split



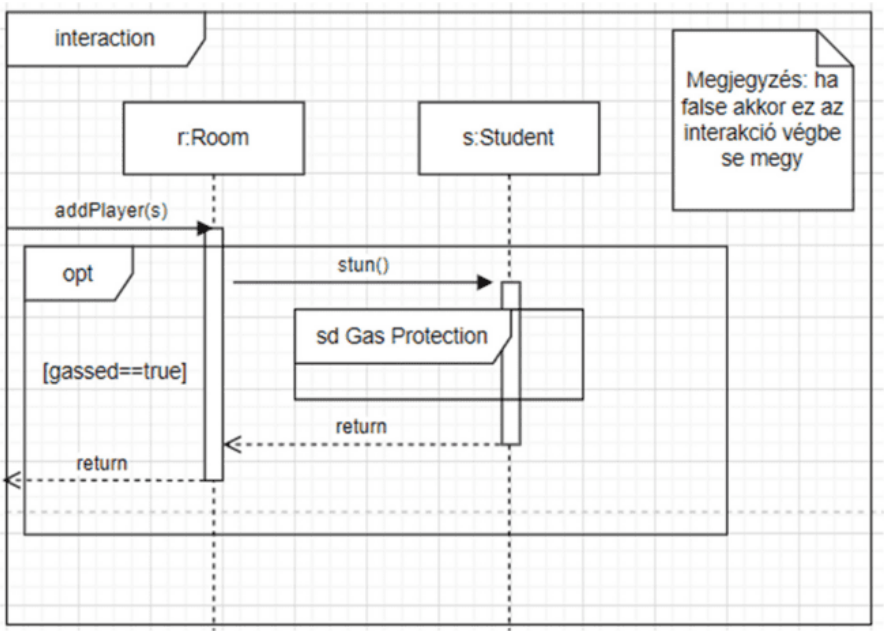
Student Picks Up Logarléc



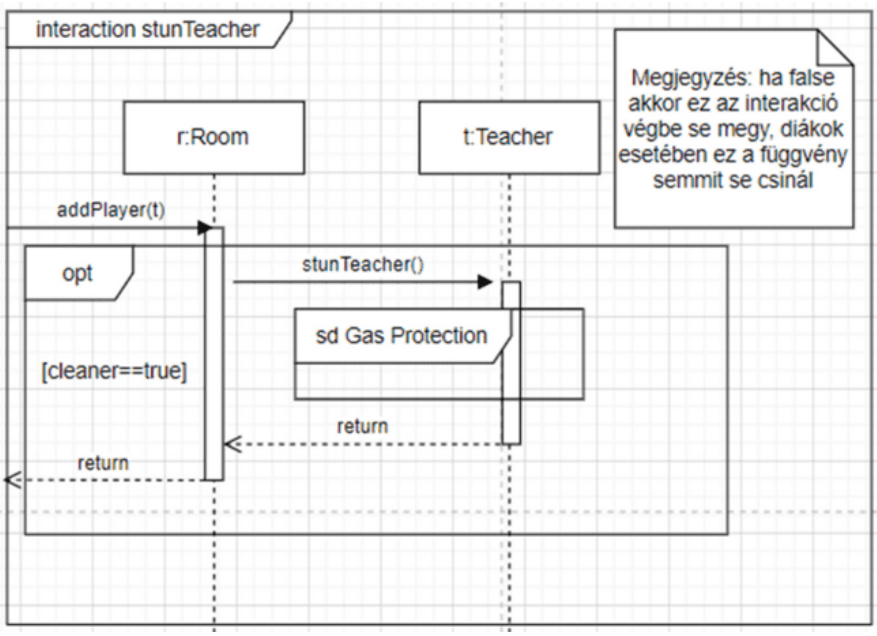
Camambert Use



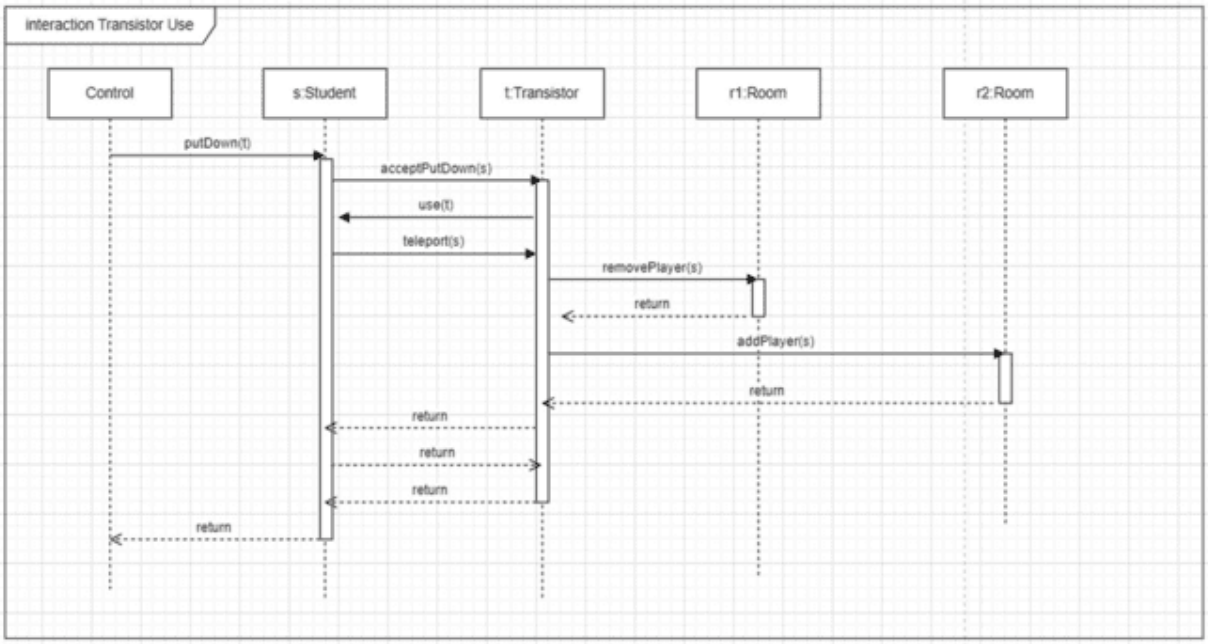
Student in Gassed Room without Mask



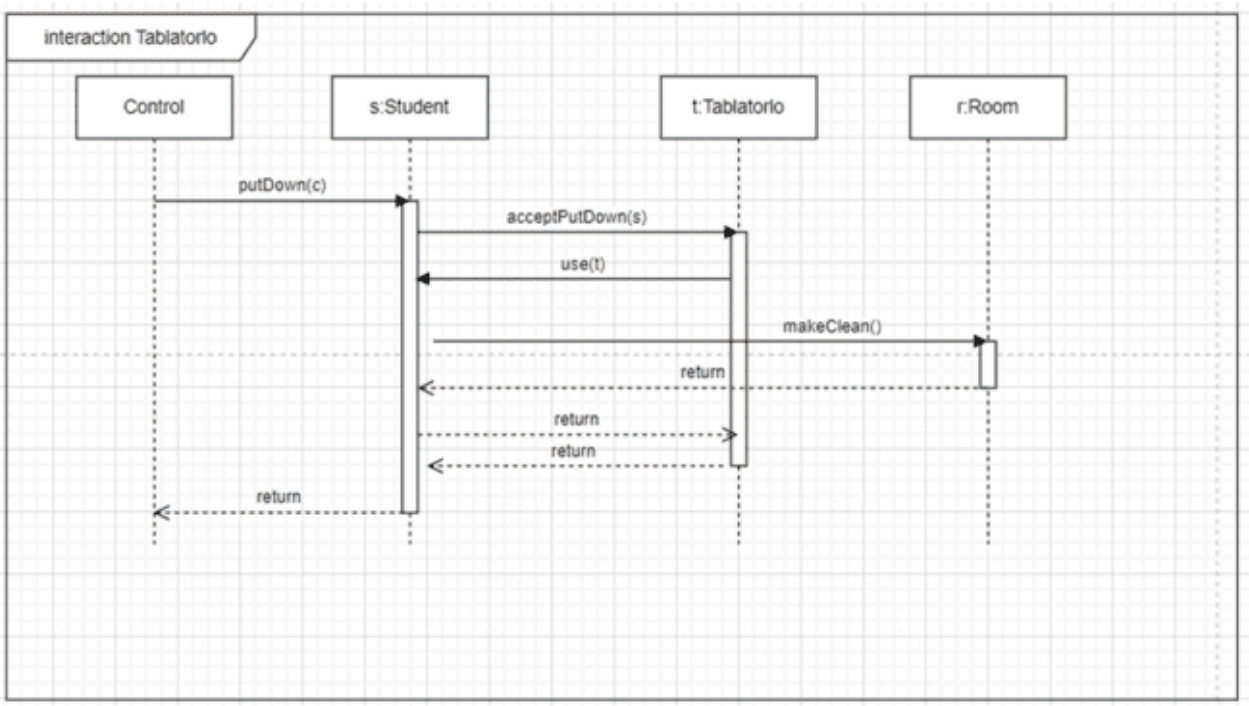
Teacher in clean room



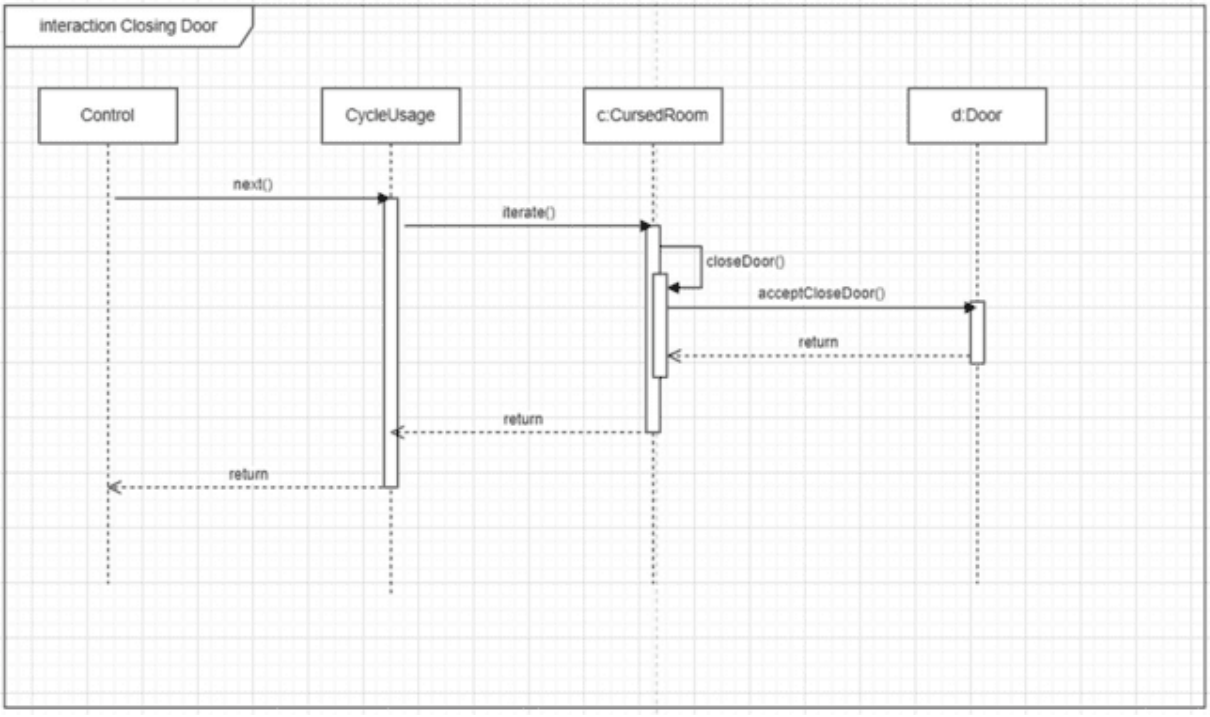
Transistor use



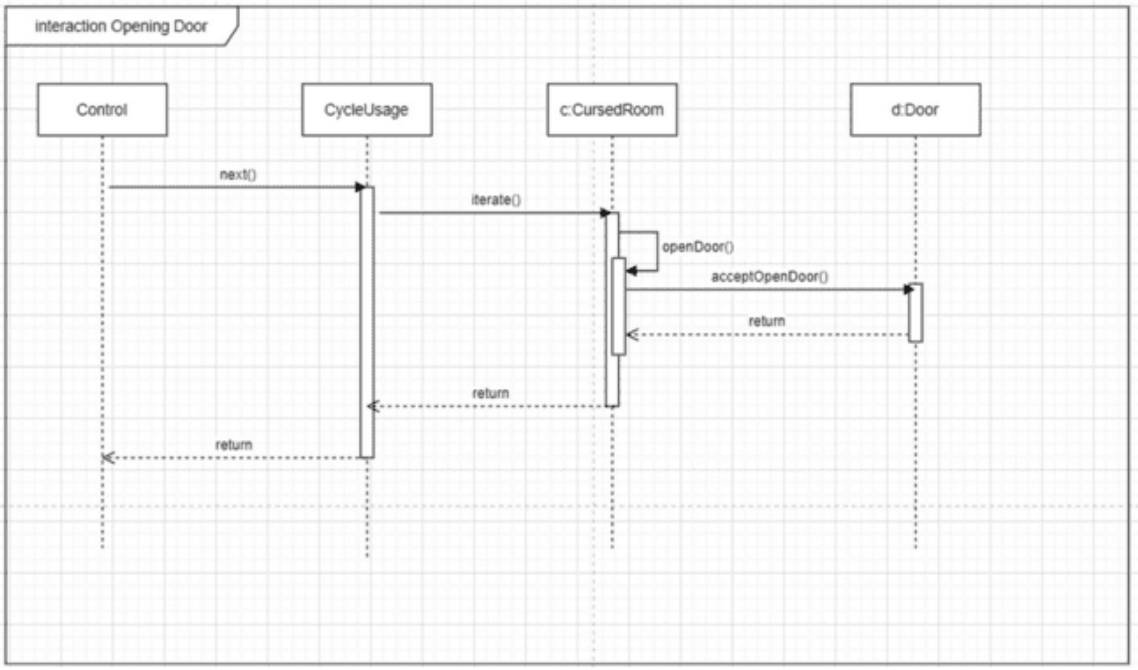
Tablatorlo Use



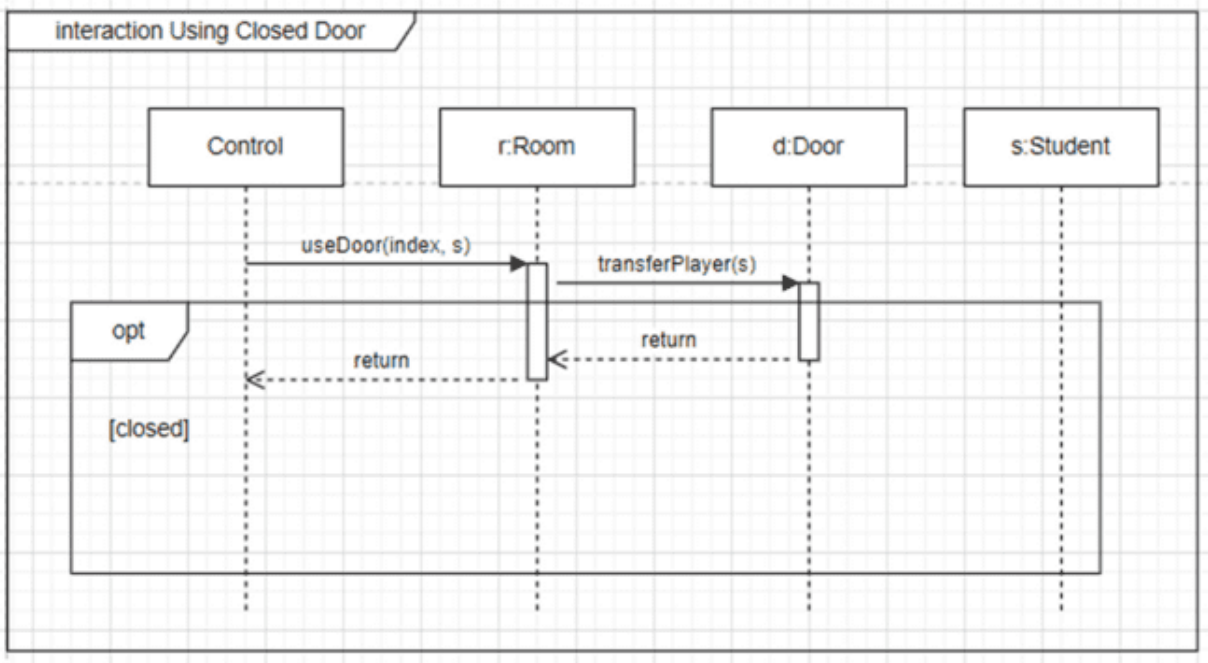
Door closing



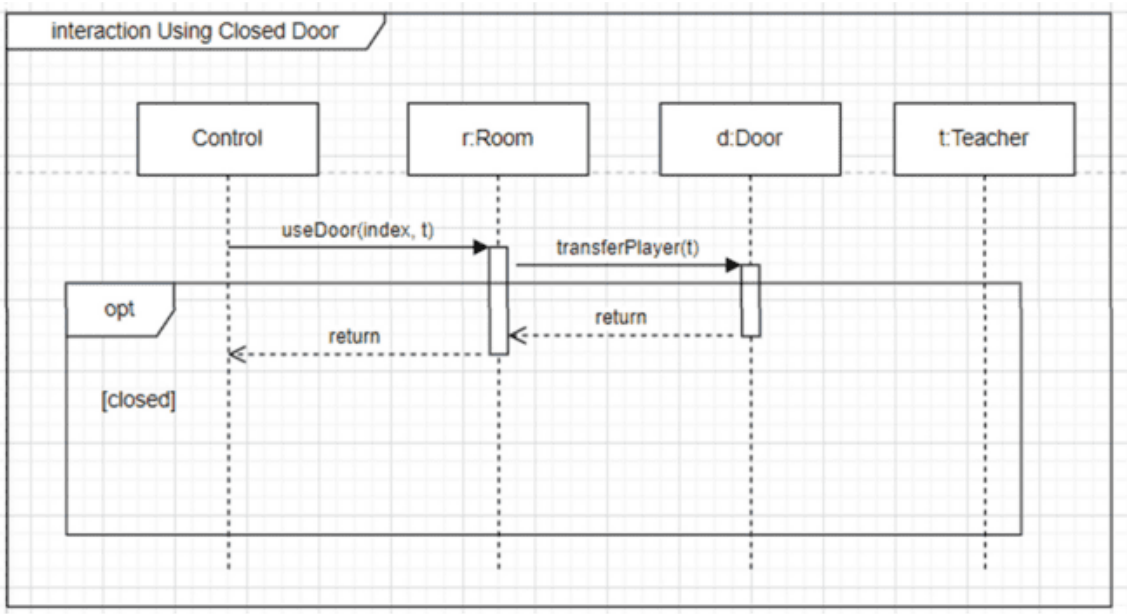
Door opening



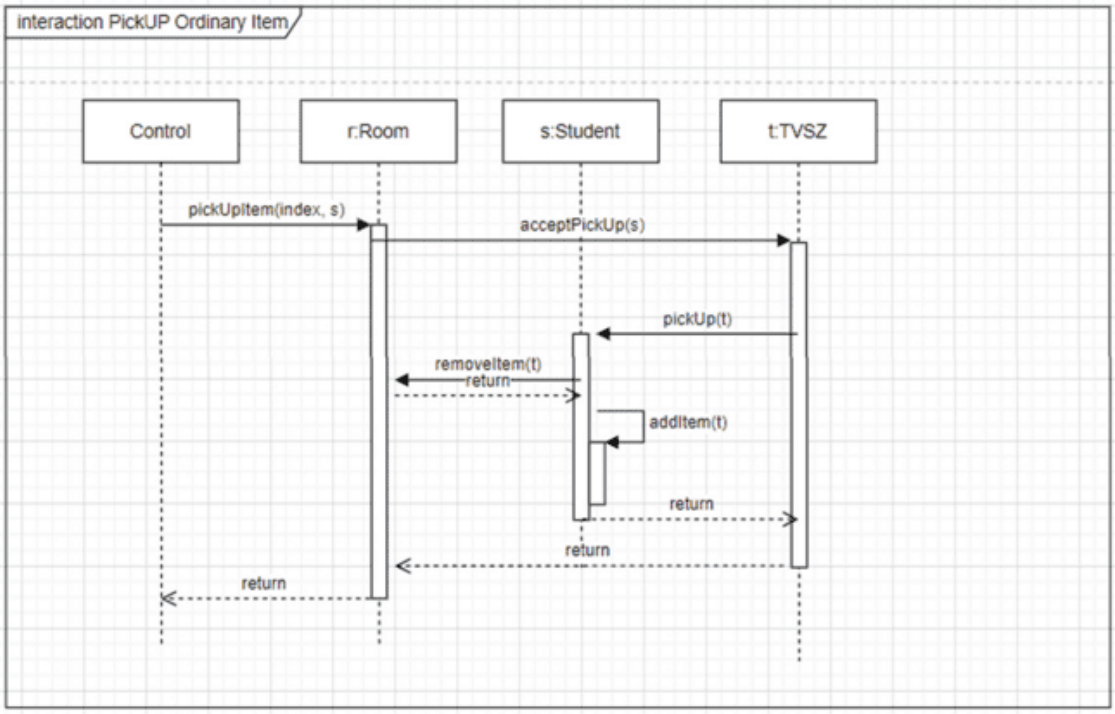
Using closed door



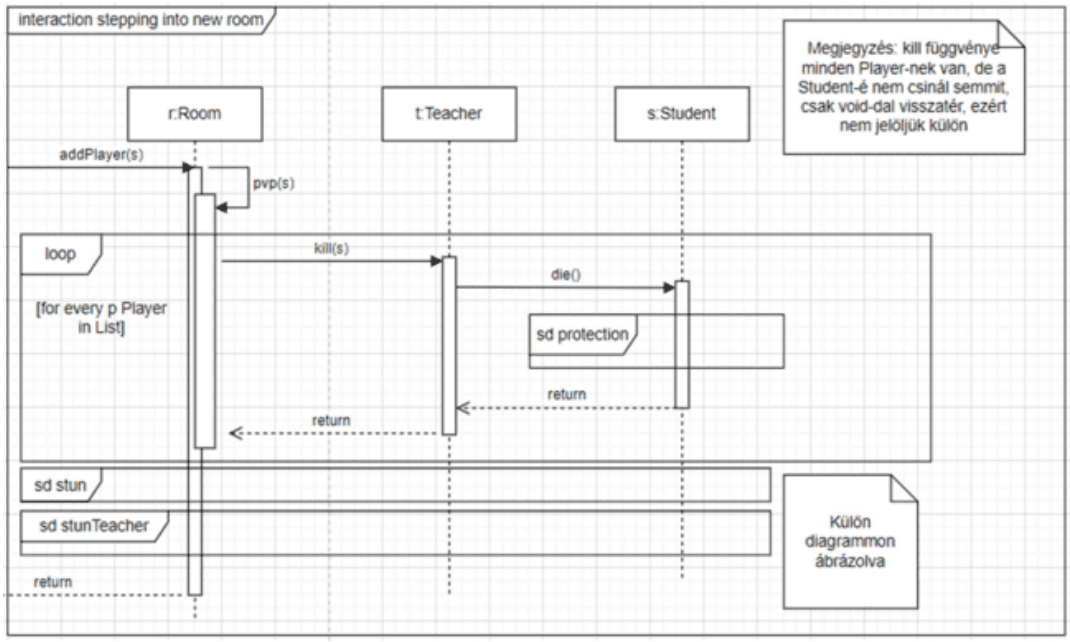
Teacher using closed door


interaction Using Closed Door
Control
r:Room
d:Door
t:Teacher
useDoor(index, t)
transferPlayer(t)
opt
return
return
[closed]

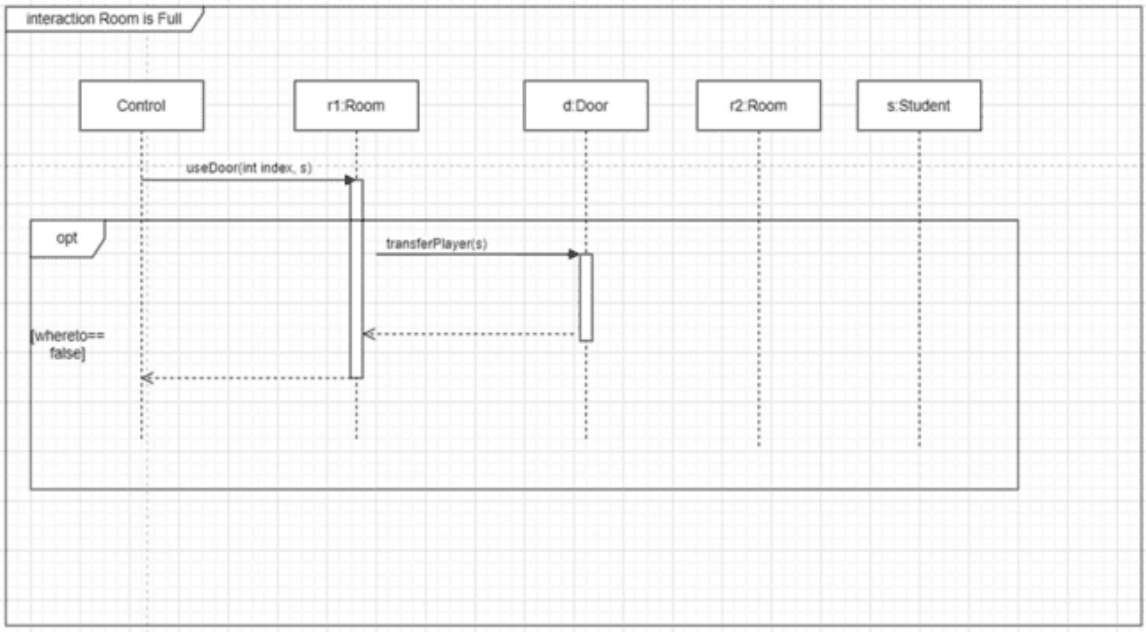
PickUp Item


interaction PickUP Ordinary Item
Control
r:Room
s:Student
t:TVSZ
pickUpItem(index, s)
acceptPickUp(s)
pickUp(t)
removeItem(t)
return
addItem(t)
return
return
return

Stepping into new room

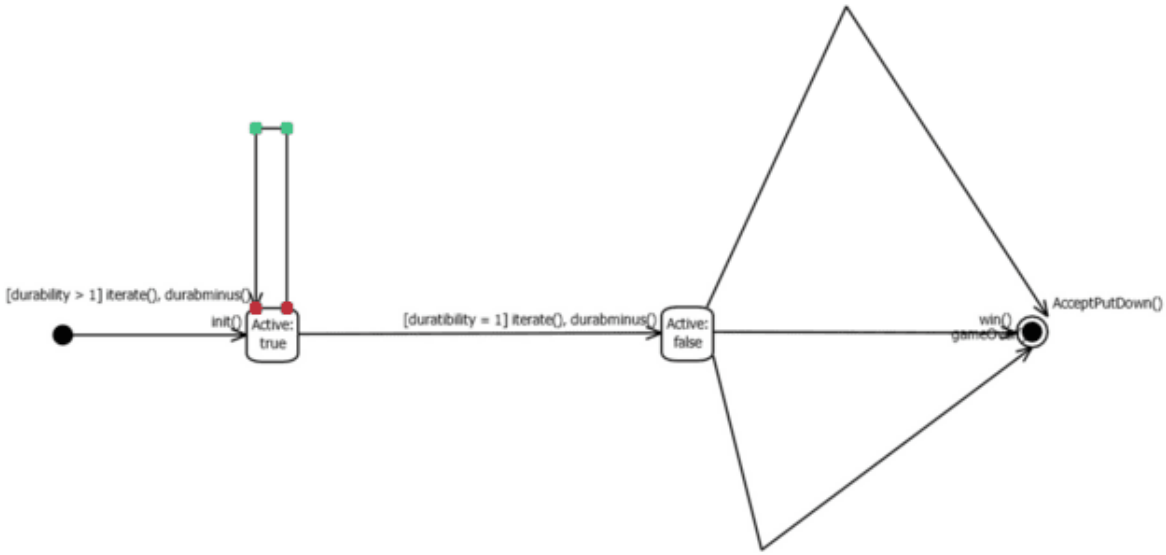


Room is full

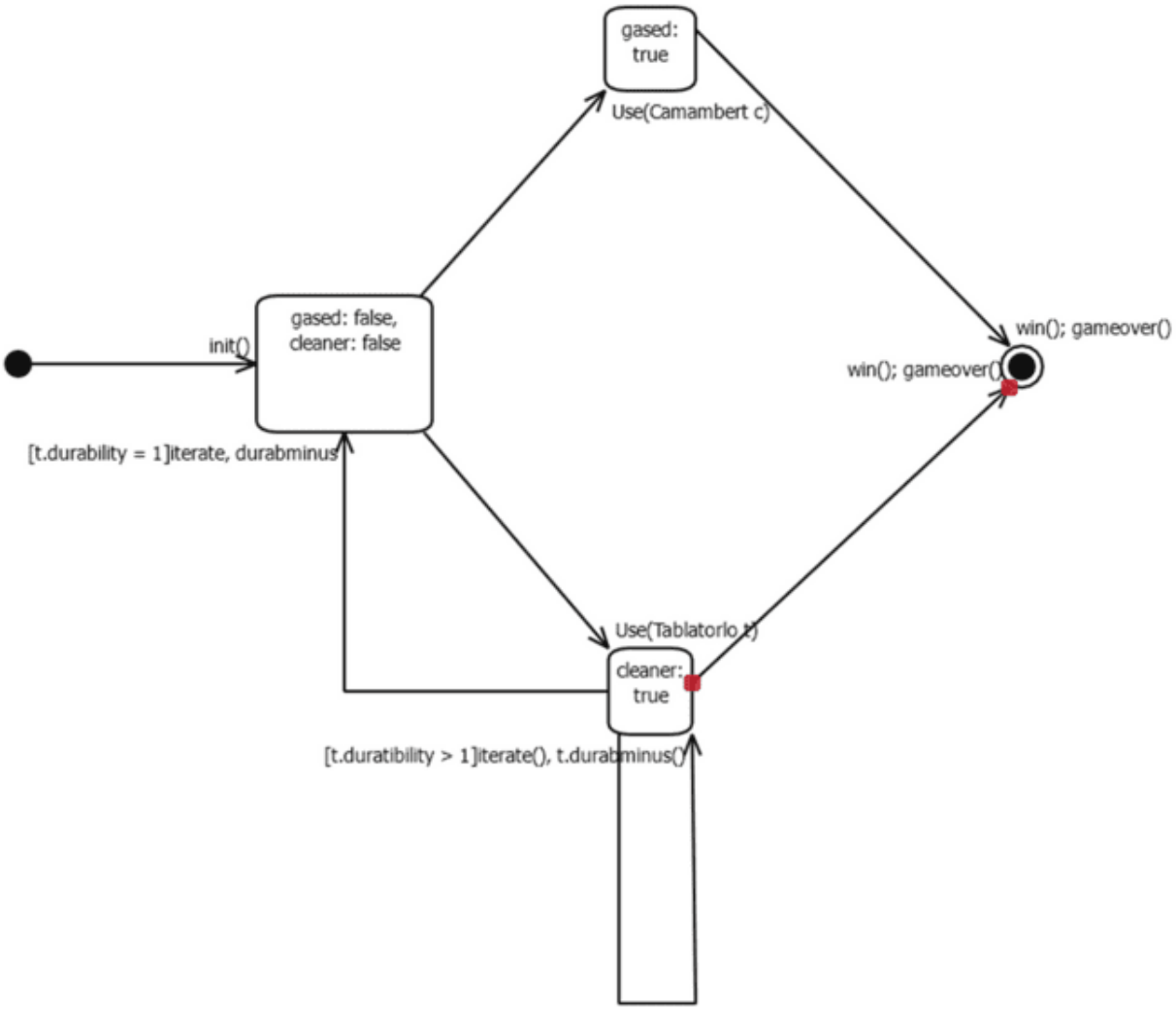


4.5 State-chartok

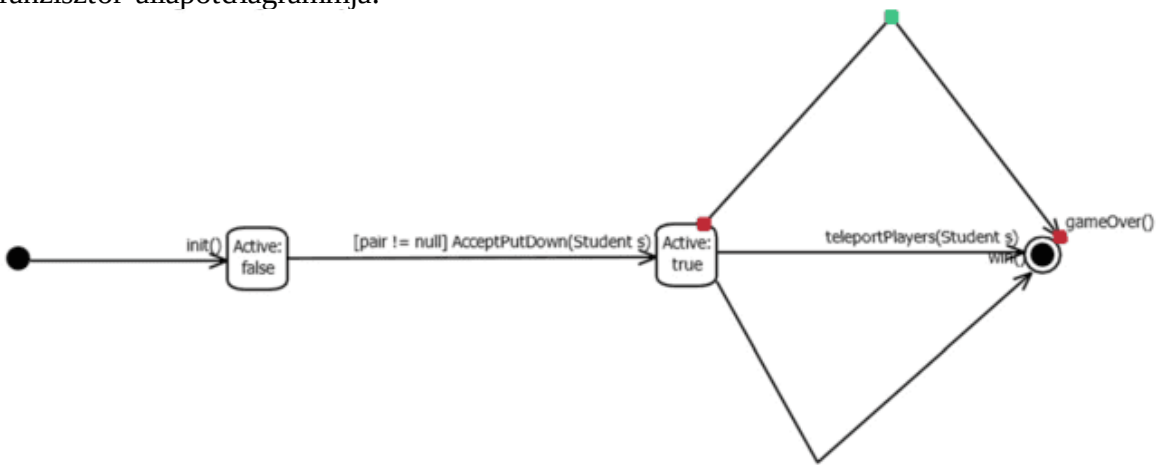
Sör állapotdiagramja:



Szoba állapotdiagramja (durabminus metódus ellenőriz és ha 0-ra csökken a durability, akkor visszaállítja a szobát):



Tranzisztor állapotdiagrammja:



4.6 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2023.03.07 9:00	2,5 óra	Fodor A. Fodor D. Földi Ludányi Mikola	Értekezlet: A csapat megbeszélte a feladattal kapcsolatos teendőket, objektumokat. Újragondoltuk az osztályok felelősségeit, kapcsolatait. Módosítottuk az osztálydiagrammot, kérdéseket fogalmaztunk meg.
2023.03.08. 8:00	2 óra	Fodor A. Fodor D.	Szekvenciadiagrammok javításának elkezdése, kérdések megfogalmazása a feladattal kapcsolatban, elvi hibák javítása
2023.03.09 11:00	3 óra	Fodor A. Fodor D.	Osztálydiaramm átdolgozása az eddig megbeszéltek alapján, új ötletek implementálása, pl. Interfészek
2023.03.10 9:00	3 óra	Fodor D.	Osztálydiagram újabb változtatása, szerkesztése, előzetes hibák kijavítása
2023.03.10 12:00	1,5 óra	Fodor A. Fodor D. Földi Ludányi Mikola	Értekezlet: Csapatmegbeszélés arról, hogy, hogyan álljunk neki a szekvenciadiagrammok átalakításának
2023.03.10 19:00	1,5 óra	Fodor A. Fodor D. Földi Ludányi Mikola	Értekezlet: Szekvenciadiagrammok tervezése, Osztálydiagramm javítása, feladatok ellenőrzése
2023.03.10 20:30	4 óra	Fodor A.	Szekvenciadiagrammok végleges tervezése, megvalósítása, előzőek szerkesztése. Mások munkájának ellenőrzése, felosztás ellenőrzése, osztálydiagramban hibák feljegyzése

2023.03.10 20:30	3 óra	Ludányi	4.1-es rész elkészítése, 4.3-as átellenőrzése
2023.03.10 21:00	3 óra	Mikola	Állapotdiagramok újragondolása, új állapotdiagramok létrehozása, régiek eltávolítása, többiek munkájának átnézése.
2023.03.10 22:00	2 óra	Fodor A. Fodor D.	Megbeszélés alapján osztálydiagram alakítása majdnem végleges állapotába
2023.03.11 7:30	4 óra	Fodor A. Fodor D.	Mások munkájának ellenőrzése, kijavítása, saját feladatok megvalósítása, Szekvenciadiagrammok átdolgozása, osztálydiagramm kijavítása, befejezése, dokumentum nyomtatása, összeszerkesztése végleges állapotba
2023.03.11 8:30	2,5 óra	Ludányi	Módosított osztálydiagram alapján kisebb változtatások, dokumentum formázása, szekvenciadiagrammok csoportos megvitatása

5. Szkeleton tervezés

6 – ripgyork

Konzulens:
Ádám Zsófia

Csapattagok

Fodor Attila	EUGN1B	afodor998@gmail.com
Fodor Dávid	D02DBR	dfodor999@gmail.com
Földi Balázs	AB8Y3S	fbalu8@gmail.com
Ludányi Barnabás	V5PWP4	ludanyib2003@gmail.com
Mikola Bálint István	TCV0Y9	mikola.balint.istvan@gmail.com (kapcsolattartó)

2024.03.18

5.

ripgyork

6. 5. Szkeleton tervezése

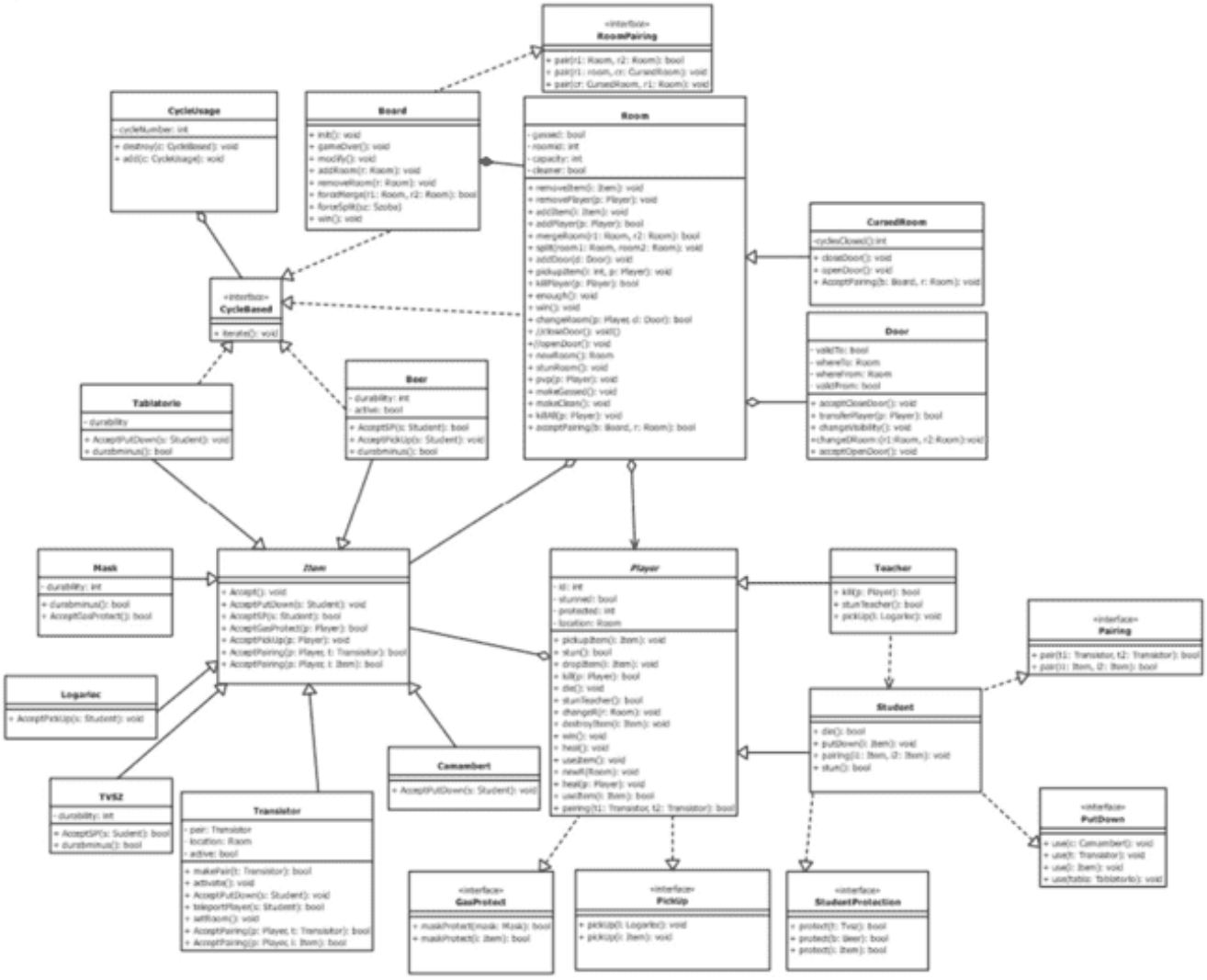
5.0 Analízis modell változásai

- **Board**
  - A forceMerge() függvény visszetérési érteke bool lett
  - Mostantól megvalósítja az új, RoomPairing interfészt is
- **Door**
  - Transfers attribútumot és a changeVisibility() metódust töröltük
  - A transferPlayer() metódus visszatérési értéke mostantól bool
- **GasProtect Interfész**
  - A maskProtect() metódusok visszatérési értéke bool
- **Item**
  - Az acceptGasProtect() és az acceptSP() metódusok visszatérési értéke bool
  - Új függvény
    - +acceptPairing(p: Player, t: Transistor): bool – Tranzisztorok párosítását valósítja meg, ha ez sikerült, igazzal tér vissza
  -
- **Player**
  - Az attribútumainak láthatósága protected helyett private lett
  - A move() és a turn() függvények eltávolításra került mind az ősosztályban, mind a leszármazottakban
  - A stun(), stunTeacher() és kill() függvények visszatérési értéke logikai lett
  - Új függvények:
    - +heal(p: Player): void – Stun megszüntetése
    - +useItem(i: Item): bool- Tárgy használatát valósítja meg, ha sikeres, igaz
    - +pairing(t1: Transistor, t2: Transistor): bool – Tranzisztorok párosítására szolgál, ha sikerült, igazzal tér vissza
- **Room**
  - Az addPlayer(), mergeRoom(), killPlayer(), és changeRoom() metódusok továbbiakban logikai értékkel térnek vissza
  - A closeDoor() és openDoor() függvényeket töröltük, csak a leszármazott, CursedRoom valósítja meg.
  - Új függvények:
    - +acceptPairing(b: Board, r: Room): bool - Párba állítja egy másik szobával, ezzel megoldva a merge által okozott problémákat. Ha sikerült, akkor igazzal tér vissza.
    - +killAll(p: Player):
  - A pvp() metódus mostantól paraméterként egy p: Player objektumot vár
- **Student**
  - A die() és stun() függvények mostantól logikai értékkel térnek vissza
  - Új függvény:
    - +pairing(t1: Transistor, t2: Transistor): bool – Felülírt függvény az ősosztályból
- **Teacher**
  - A kill() és stunTeacher() metódusainak visszatérési értéke logikaira változott
- **Transistor**
  - A teleportPlayer() függvénye logikai értékkel tér vissza

ripgyork

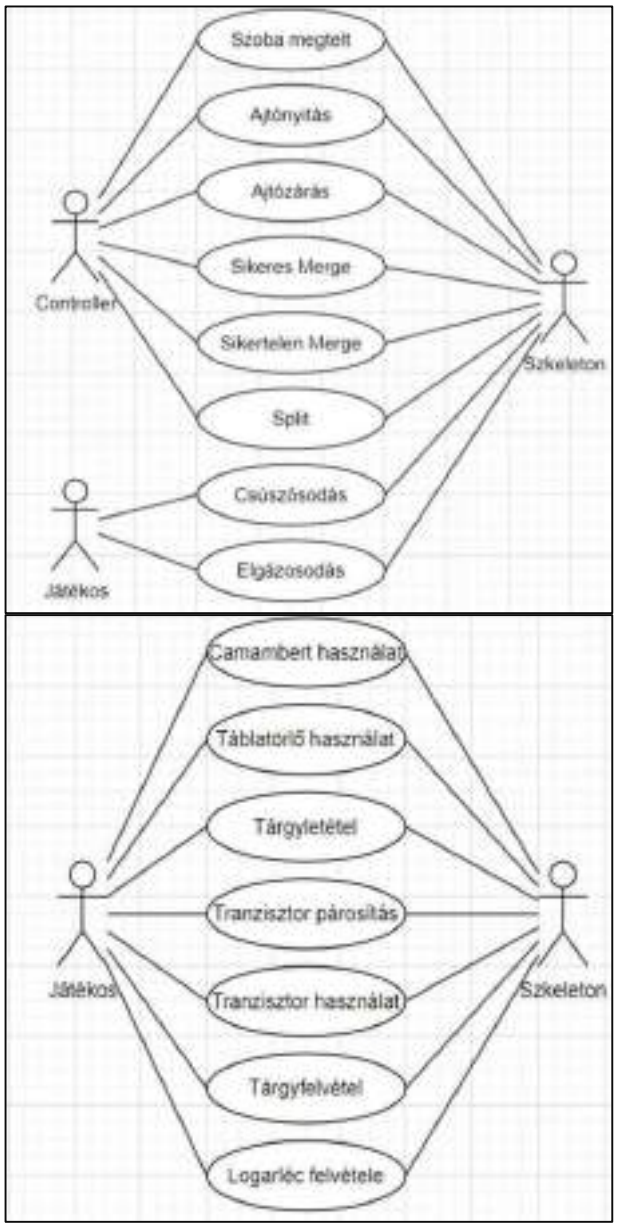
- **ÚJ INTERFÉSZ**: **RoomPairing**
  - **Felelősség**
    Az interfész felel a merge funkció helyes működéséért, az abban felmerülő problémák kiküszöböléséért. Ha sikerült a merge, igazzal tér vissza. Az utolsó két metódus nem engedi őket mergelni.
  - **Metódusok**
    - +pair(r1: Room, r2: Room): bool
    - +pair(cr: CursedRoom, r: Room): void
    - +pair(r: Room, cr: CursedRoom): void

Új osztálydiagramm

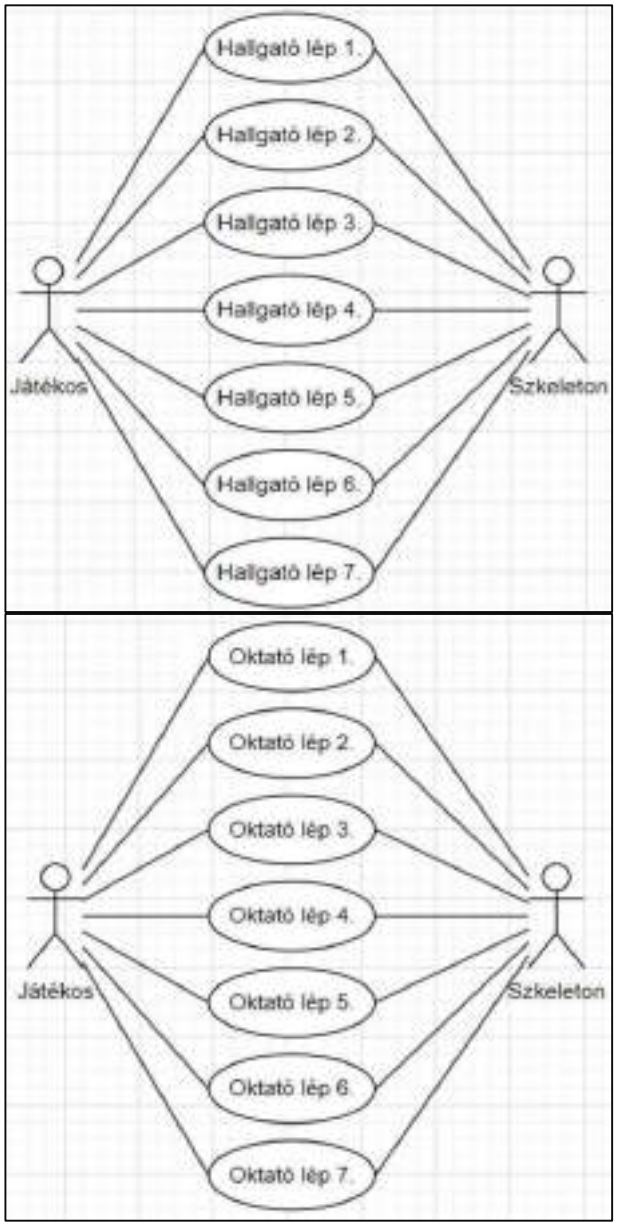


6.1 5.1 A szkeleton modell valóságos use-case-ei

5.1.1 Use-case diagram



ripgyork


Hallgató lép 1.
Hallgató lép 2.
Hallgató lép 3.
Hallgató lép 4.
Hallgató lép 5.
Hallgató lép 6.
Hallgató lép 7.
Játékos
Szkeleton
Oktató lép 1.
Oktató lép 2.
Oktató lép 3.
Oktató lép 4.
Oktató lép 5.
Oktató lép 6.
Oktató lép 7.
Játékos
Szkeleton

5.1.2 Use-case leírások

Use-case neve	Sikeres merge
Rövid leírás	Merge kérés érkezik 2 szobára, melyek közt van/nincs gázos, a játékosok száma a két szobában együttvéve pedig kevesebb, mint a nagyobb kapacitású szoba kapacitása
Aktorok	Szkeleton, Controller
Forgatókönyv	A merge sikeresen végrehajtódik, a kisebb kapacitású szobából minden átkerül a nagyobb kapacitásúba, valamint a kisebb szoba ajtajait is a nagyobb szoba kapja, végül a kis szoba törlődik

Use-case neve	Sikertelen merge
Rövid leírás	Merge kérés érkezik 2 szobára, melyek közt van/nincs gázos, a játékosok száma a két szobában együttvéve pedig több, mint a nagyobb kapacitású szoba kapacitása
Aktorok	Szkeleton, Controller
Forgatókönyv	Nem történik semmilyen strukturális változás

Use-case neve	Ajtó zárás
Rövid leírás	Egy ajtó bezáródik, mivel a szoba, ahova lehetne jutni megtelt, vagy átkozott volt és megadott számú kör már eltelt
Aktorok	Szkeleton, Controller
Forgatókönyv	Az ajtó closed attribútuma true-ra állítódik, és innentól kezdve a játékosok nem fogják tudni használni

Use-case neve	Ajtó nyitás
Rövid leírás	Egy ajtó kinyílik, mivel a szoba, ahova lehetne jutni eddig telve volt és most szabadult, vagy átkozott volt és megadott számú kör már eltelt
Aktorok	Szkeleton, Controller
Forgatókönyv	Az ajtó closed attribútuma false-ra állítódik, és innentől kezdve a játékosok már fogják tudni használni

Use-case neve	Szoba megtelt
Rövid leírás	Egy szoba kapacitását elérte a benne tartózkodók száma, így a szoba minden bele irányuló ajtót bezár.
Aktorok	Szkeleton, Controller
Forgatókönyv	Minden ajtó closed attribútumatrue-ra állítódik, és innentől kezdve a játékosok nem fogják tudni használni

Use-case neve	Elgázosodás
Rövid leírás	Egy hallgató lehelyezi a camembert, a szoba ezzel gázossá válik.
Aktorok	Játékos , Szkeleton
Forgatókönyv	A szoba gázosságát jelző paraméter true-ra állítódik, valamint a stun metódus minden bent tartózkodóra meghívódik. Amennyiben a szoba eredetileg is gázos volt, nincs változás.

Use-case neve	Csúszóssá válás
Rövid leírás	Egy hallgató lehelyezi a táblatörlőt, mire a szoba csúszóssá válik.
Aktorok	Szkeleton, Játékos
Forgatókönyv	A szoba csúszósságát jelző paraméter true-ra állítódik, valamint stunTeacher() metódus minden bent tartózkodóra meghívódik (csak tanárokra van hatással). Amennyiben a szoba eredetileg is csúszós volt, nincs változás.

Use-case neve	Split
Rövid leírás	A kiválasztott szoba kettéosztódik.
Aktorok	Szkeleton, Controller
Forgatókönyv	Létrejön egy új szoba, amely kizárólag a split metódust végrehajtó eredeti szobával van összeköttetésben, valamint minden egyéb attribútuma (kapacitás, gázos-e, csúszós-e) megegyezik az eredeti szobával. Az eredeti szobában tartózkodó játékosok és tárgyak az eredeti szobában maradnak.

Use-case neve	Hallgató lép 1.
Rövid leírás	A hallgató olyan szobába lép, ahol nem fér el.
Aktorok	Játékos, Szkeleton
Forgatókönyv	A szkeleton létrehoz 2 szobát, egy ajtót, ami a két szobát köti össze és egy hallgatót. A szobák közül az egyik kapacitása nulla, a szkeleton a másikhoz rendeli a halgatót. A szkeleton meghívja a hallgató changeR() függvényét, amivel elkezdődik a folyamat.

Use-case neve	Hallgató lép 2.
Rövid leírás	A hallgató egy gázos szobába lép és van maszkja
Aktorok	Játékos, Szkeleton
Forgatókönyv	A szkeleton létrehoz 2 szobát, egy ajtót, ami a két szobát köti össze, egy maszkot és egy hallgatót, akihez hozzárendeli a maszkot. Az egyik szoba gázos, mindkettő kapacitása legalább egy. A szkeleton a nem gázos szobához rendeli a játékost, majd meghívja a changeR() függvényét, amivel elindul a folyamat.

Use-case neve	Hallgató lép 3.
Rövid leírás	A hallgató egy gázos szobába lép, nincs maszkja
Aktorok	Játékos, Szkeleton
Forgatókönyv	A szkeleton létrehoz 2 szobát, egy ajtót, ami a két szobát köti össze és egy hallgatót. Az egyik szoba gázos, mindkettő kapacitása legalább egy. A szkeleton a nem gázos szobához rendeli a hallgatót, majd meghívja a changeR() függvényét, amivel elindul a folyamat.

Use-case neve	Hallgató lép 4.
Rövid leírás	Egy védett hallgató olyan szobába lép, ahol van egy oktató
Aktorok	Játékos, Szkeleton
Forgatókönyv	A szkeleton létrehoz 2 szobát, egy ajtót, ami a két szobát köti össze, egy TVSZ-t, egy oktatót és egy hallgatót, akihez hozzárendeli a TVSZ-t. Az egyik szobához hozzárendeli az oktatót, a másik szobához rendeli a hallgatót, majd meghívja a changeR() függvényét, amivel elindul a folyamat.

Use-case neve	Hallgató lép 5.
Rövid leírás	Egy védtelen hallgató olyan szobába lép, ahol van egy oktató
Aktorok	Játékos, Szkeleton
Forgatókönyv	A szkeleton létrehoz 2 szobát, egy ajtót, ami a két szobát köti össze, egy oktatót és egy hallgatót. Az egyik szobához hozzárendeli az oktatót, a másik szobához rendeli a hallgatót, majd meghívja a changeR() függvényét, amivel elindul a folyamat.

Use-case neve	Hallgató lép 6.
Rövid leírás	A hallgató olyan szobába lép, ahol *x* oktató van. A hallgatónak egy olyan tárgya van, ami kevesebb, mint *x* alkalommal menti meg életét.
Aktorok	Játékos, Szkeleton
Forgatókönyv	A szkeleton létrehoz 2 szobát, egy ajtót, ami a két szobát köti össze, egy TVSZ-t, ami 1 alkalommal véd, 2 oktatót és egy hallgatót, akihez hozzárendeli a TVSZ-t. Az egyik szobához hozzárendeli az oktatókat, a másik szobához rendeli a hallgatót, majd meghívja a changeR() függvényét, amivel elindul a folyamat.

Use-case neve	Hallgató lép 7.
Rövid leírás	A hallgató olyan szobába lép, ahol *x* oktató van. A hallgatónak egy olyan tárgya van, ami még *x* alkalommal menti meg életét.
Aktorok	Játékos, Szkeleton
Forgatókönyv	A szkeleton létrehoz 2 szobát, egy ajtót, ami a két szobát köti össze, egy TVSZ-t, ami 2 alkalommal véd, 2 oktatót és egy hallgatót, akihez hozzárendeli a TVSZ-t. Az egyik szobához hozzárendeli az oktatókat, a másik szobához rendeli a hallgatót, majd meghívja a changeR() függvényét, amivel elindul a folyamat.

Use-case neve	Oktató lép 1.
Rövid leírás	Egy oktató egy üres szobába lép, ahol elfér
Aktorok	Skeleton, Oktató
Forgatókönyv	Létrejön 2 üres szoba. Az egyik szobában létrejön egy oktató. Oktató átlép egyik szobából egy másikba.

Use-case neve	Oktató lép 2.
Rövid leírás	Egy oktató egy olyan szobába lép, ahol van hallgató védelem nélkül
Aktorok	Skeleton, Oktató
Forgatókönyv	Létrejön 2 szoba. Az egyik szobában létrejön egy hallgató a másikban egy oktató. Az oktató átlép a másik szobába és megöli a hallgatót.

Use-case neve	Oktató lép 3.
Rövid leírás	Egy oktató egy olyan szobába lép, ahol van hallgató védelemmel (pl.: TVSZ)
Aktorok	Skeleton, Oktató
Forgatókönyv	Létrejön 2 szoba. Az egyik szobában létrejön egy hallgató a másikban egy oktató. Az oktató átlép a másik szobába és megpróbálja megölni a hallgatót. A hallgató védekezik és életben marad

Use-case neve	Oktató lép 4.
Rövid leírás	Egy oktató egy olyan szobába lép, ami üres és el van gázosítva. Az oktatónak nincs maszkja
Aktorok	Skeleton, Oktató
Forgatókönyv	Létrejön 2 szoba: egy sima és egy elgázosított. A sima szobában létrejön egy oktató. Az oktató átlép az elgázosított szobába, elkábul es eldobja az összes tárgyát.

Use-case neve	Oktató lép 5.
Rövid leírás	Egy oktató egy olyan szobába lép, ami üres és el van gázosítva. Az oktatónak van maszkja
Aktorok	Skeleton, Oktató
Forgatókönyv	Létrejön 2 szoba: egy sima és egy elgázosított. A sima szobában létrejön egy oktató. Az oktató átlép az elgázosított szobába, de a maszkja megvédi.

Use-case neve	Oktató lép 6.
Rövid leírás	Oktató egy olyan szobába lép, ahol van nedves táblatörlő
Aktorok	Skeleton, Oktató
Forgatókönyv	Létrejön 2 szoba: egy sima és egy olyan, ahol le van téve egy nedves táblatörlő. A sima szobában létrejön egy oktató. Az oktató átlép a táblatörlős szobába, elkábul és eldobja az összes nála lévő tárgyat.

Use-case neve	Oktató lép 7.
Rövid leírás	Egy oktató egy olyan szobába lép, ahol nem fér el
Aktorok	Skeleton, Oktató
Forgatókönyv	Létrejön 2 üres szoba. Az egyik szobában létrejön egy oktató. A másik szoba kapacitása 0. Oktató megpróbál átlépni a másik szobába, de nem sikerül neki.

ripgyork

Use-case neve	Diák használ Camambert
Rövid leírás	Egy diák letesz egy Camambert
Aktorok	Skeleton, Diák
Forgatókönyv	Létrejön egy diák. Leteszi a Camambertet, ami nála van, a szoba gázos lesz és a diák stunolódik.

Use-case neve	Tablatorlo használat, Teacher stun
Rövid leírás	Egy diák letesz egy Tablatorlot egy olyan szobában, ahol van tanár.
Aktorok	Skeleton, Oktató, Diák
Forgatókönyv	Létrejön egy szoba, melyben van egy diák és egy tanár. (Feltételezzük, hogy a diák védve lett, így bent maradhat a tanárral a szobában), majd leteszi a nála lévő Tablatorlot, a tanár stunolódik.

Use-case neve	Tárgy letétel
Rövid leírás	Diák letesz egy tárgyat, ami nála van, de nem tud használódni, így csak letevődik
Aktorok	Skeleton, Diák
Forgatókönyv	Létrejön egy szoba, benne van egy diák, letesz egy tárgyat, ami nem használódik a letételkor (pl. Beer), mivel nincs bent tanár, így a tárgy csak letevődik és nem történik semmi

Use-case neve	Transistor párosítás
Rövid leírás	Két különböző, nem aktív tranzisztort párosít, mikor már nála vannak
Aktorok	Skeleton, Diák
Forgatókönyv	Feltételezzük, hogy egy diáknál már van két tranzisztor, őket párosítja magánál.

Use-case neve	Transitor használat
Rövid leírás	Két különböző szobában letett és aktív tranzisztort használ a diák
Aktorok	Skeleton, Diák
Forgatókönyv	Létrejön két szoba, egyikben egy diák. Feltételezük, hogy az egyik szobában le van téve egy Tranzisztor. A másik szobában van a diák és nála a Transistor pár. Leteszi és ezzel átkerül a másik Transistor szobájába

Use-case neve	Tárgyfelvétel
Rövid leírás	Diák felvesz egy tárgyat a szobából
Aktorok	Skeleton, Diák
Forgatókönyv	Létrejön egy szoba, benne egy diák és egy tárgy (teszt esetben Tvsz, de bármi más nem Logarléc is lehetne). A diák felveszi a tárgyat, ami elfér nála

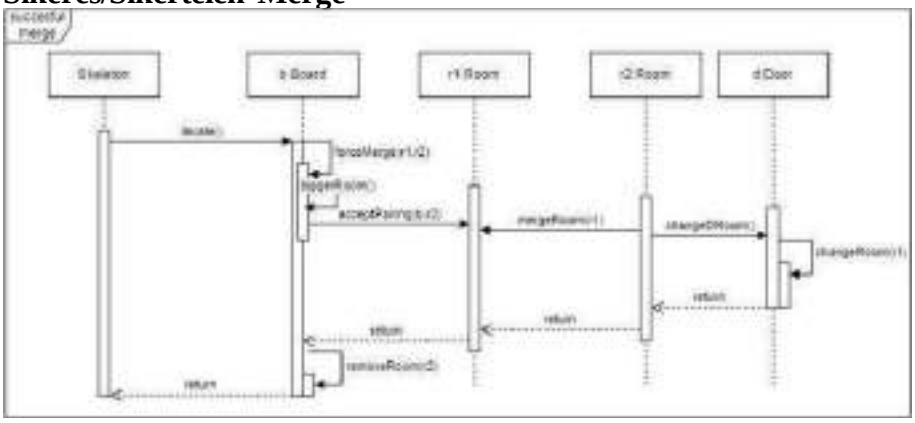
Use-case neve	Logarléc felvétel
Rövid leírás	Diák felveszi a Logarlécet a szobából
Aktorok	Skeleton, Diák
Forgatókönyv	Létrejön egy szoba, benne egy diák és egy Logarléc. A diák felveszi a Logarlécet, ami elfér nála majd jelez a szobának, hogy felvette és ezzel megnyerte a játékot

5.2 A szkeleton kezelői felületének terve, dialógusok

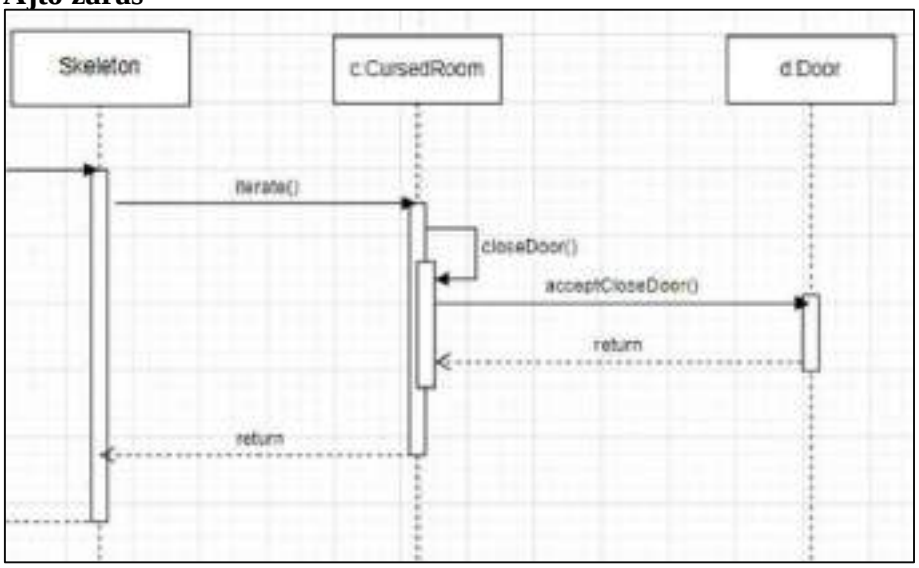
A szkeleton kezelői felületünk az indításkor 5 különböző menüpontot fog tartalmazni abc sorrendben növekvő betűkkel: Hallgató mozgása, Oktató mozgása, Tárgyfelvétel, Tárgyhasználat és Szobák változása. Egy menüpontot a hozzá tartozó betű, majd egy enter leütésével lehet megnyitni. A menüpontok fogják tartalmazni a különböző use-case-eket szintén betűkkel ellátva. Egy use-case kiválasztása után (egy use case-t ugyanúgy lehet kiválasztani, ahogy egy menüpontot.) lefut az adott use-case. Minden use-case lefutása előtt viszont inicializálásra kerülnek a használati esetben résztvevő objektumok. A lefutás során minden metódus csak annyit tesz, hogy kiírja a nevét, a típusát és a lefutása végén a visszatérési értékét (voidnál egy üres sort) a kimenetre, illetve, ha szükséges meghívja az utána következő metódust. Ha a use-case közben olyan ponthoz érkezik a program, ahol fontos, hogy egy objektumnak milyen állapota van, feltevődik egy eldöntendő kérdés a felhasználónak (pl.: ez a szoba gázos?), amit a felhasználó egy 'y' vagy 'n' karakter és az enter leütésével válaszolhat meg. Minden teszteset lefutása után a program visszakerül kezdőállapotába, ahol a fentebb említett 5 menüpontból lehet majd választani. A keretprogramot és az objektumok inicializálását is a szkeleton objektum fogja intézni.

5.3 Szekvencia diagramok a belső működésre

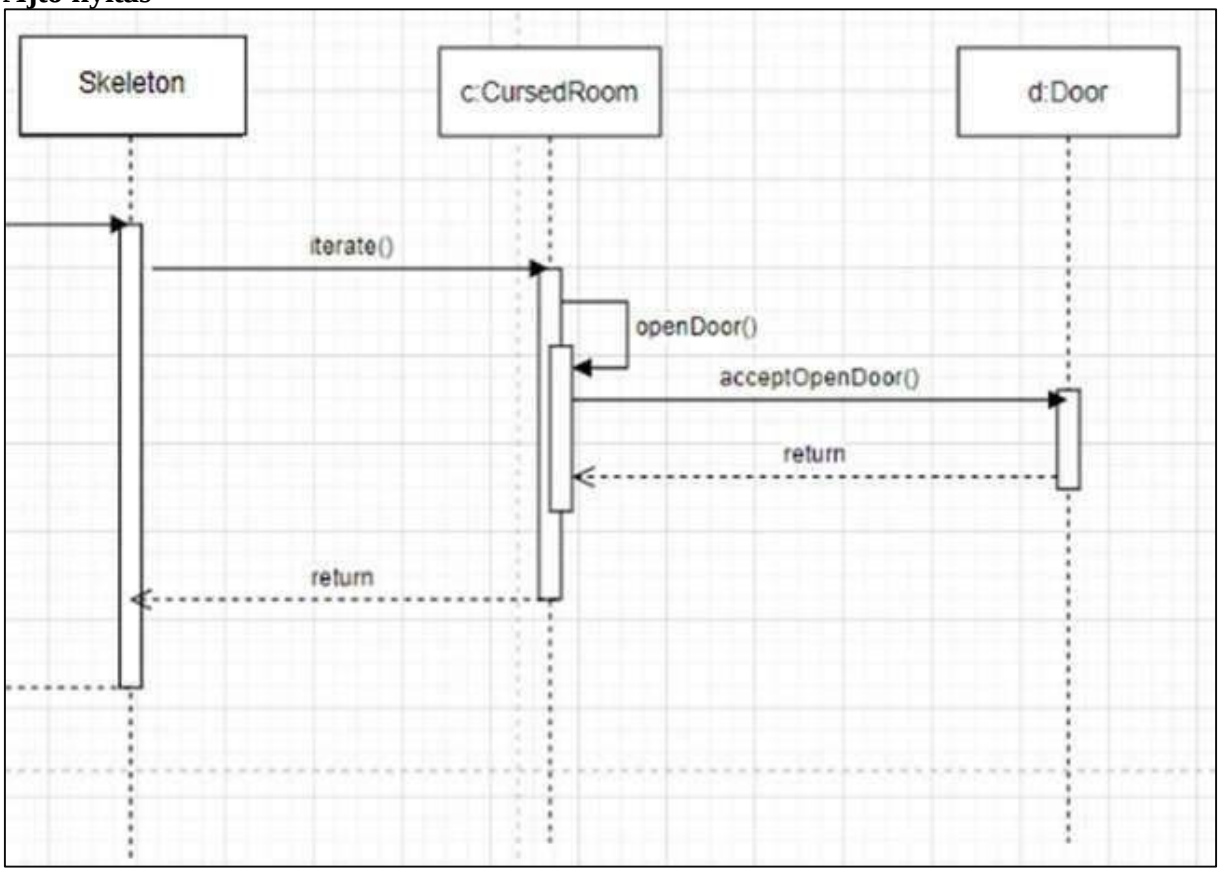
Sikeres/Sikertelen Merge



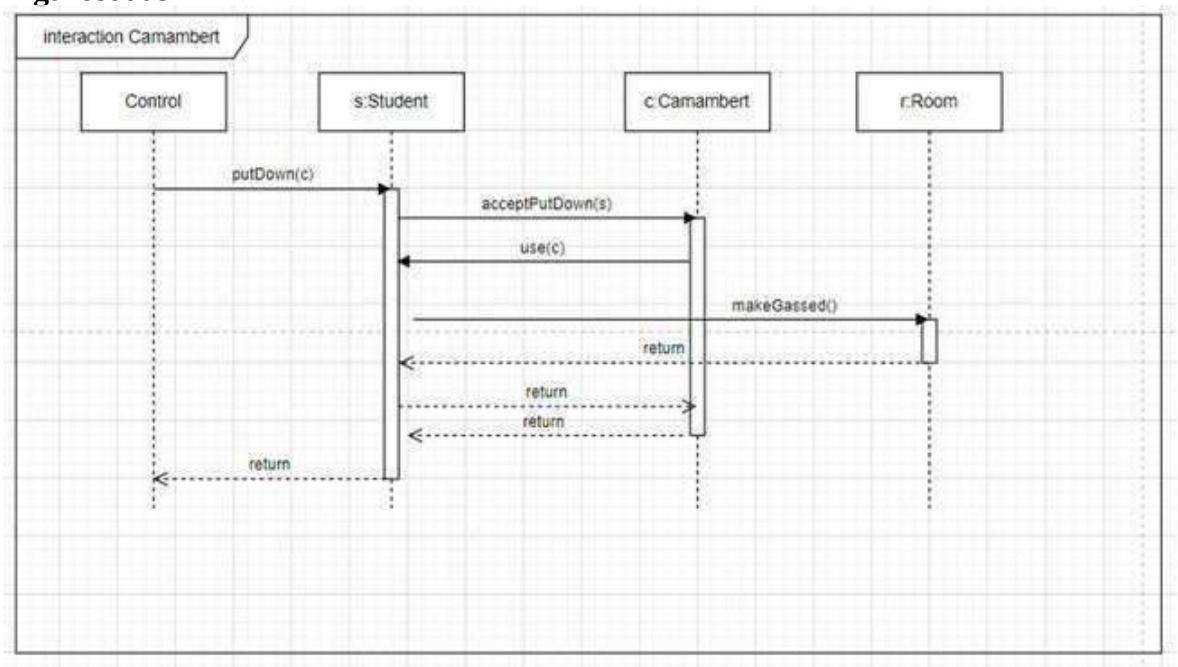
Ajtó zárás



Ajtó nyitás

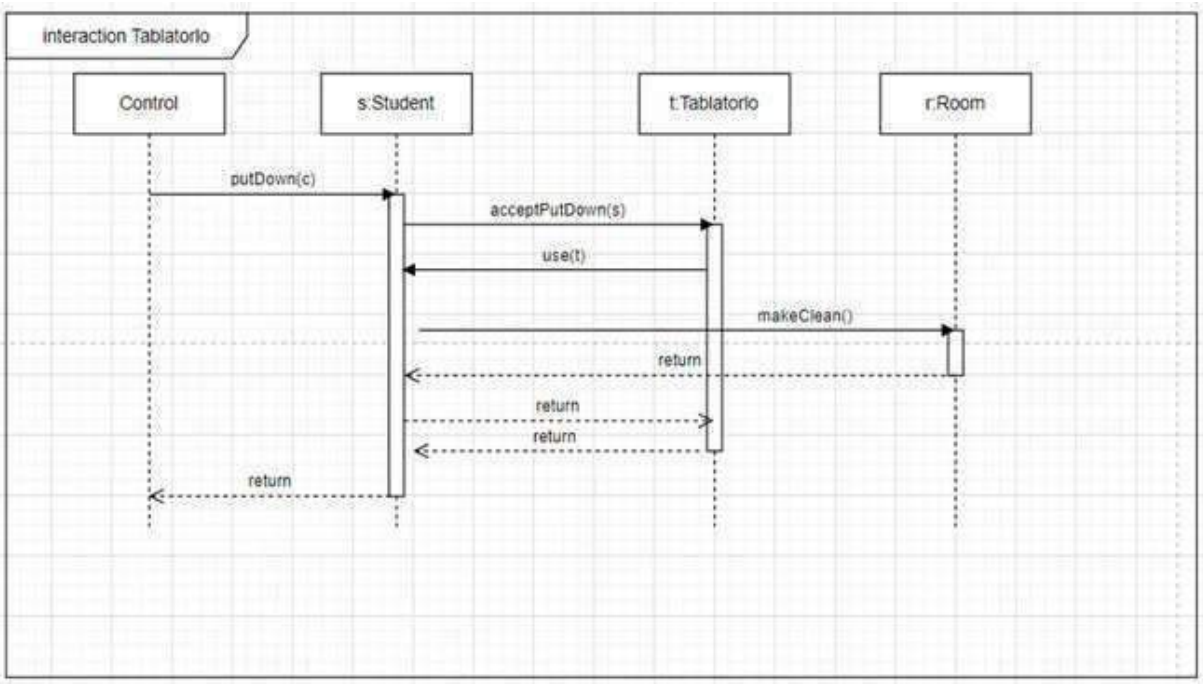


Elgázosodás

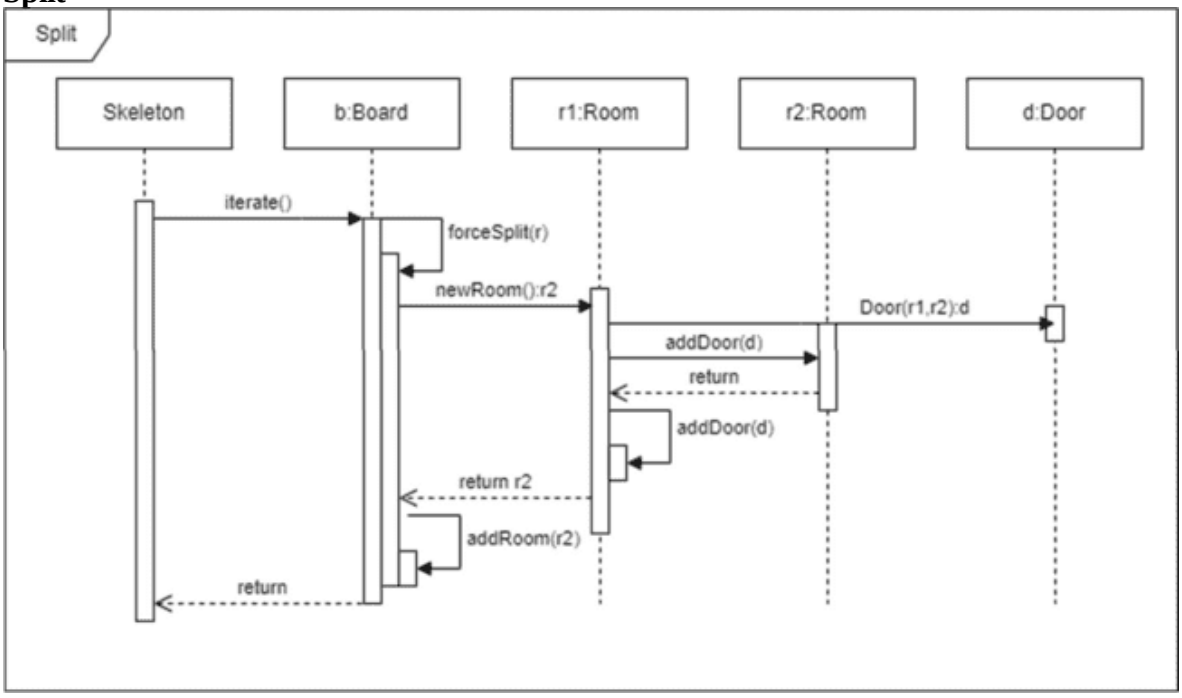


ripgyork

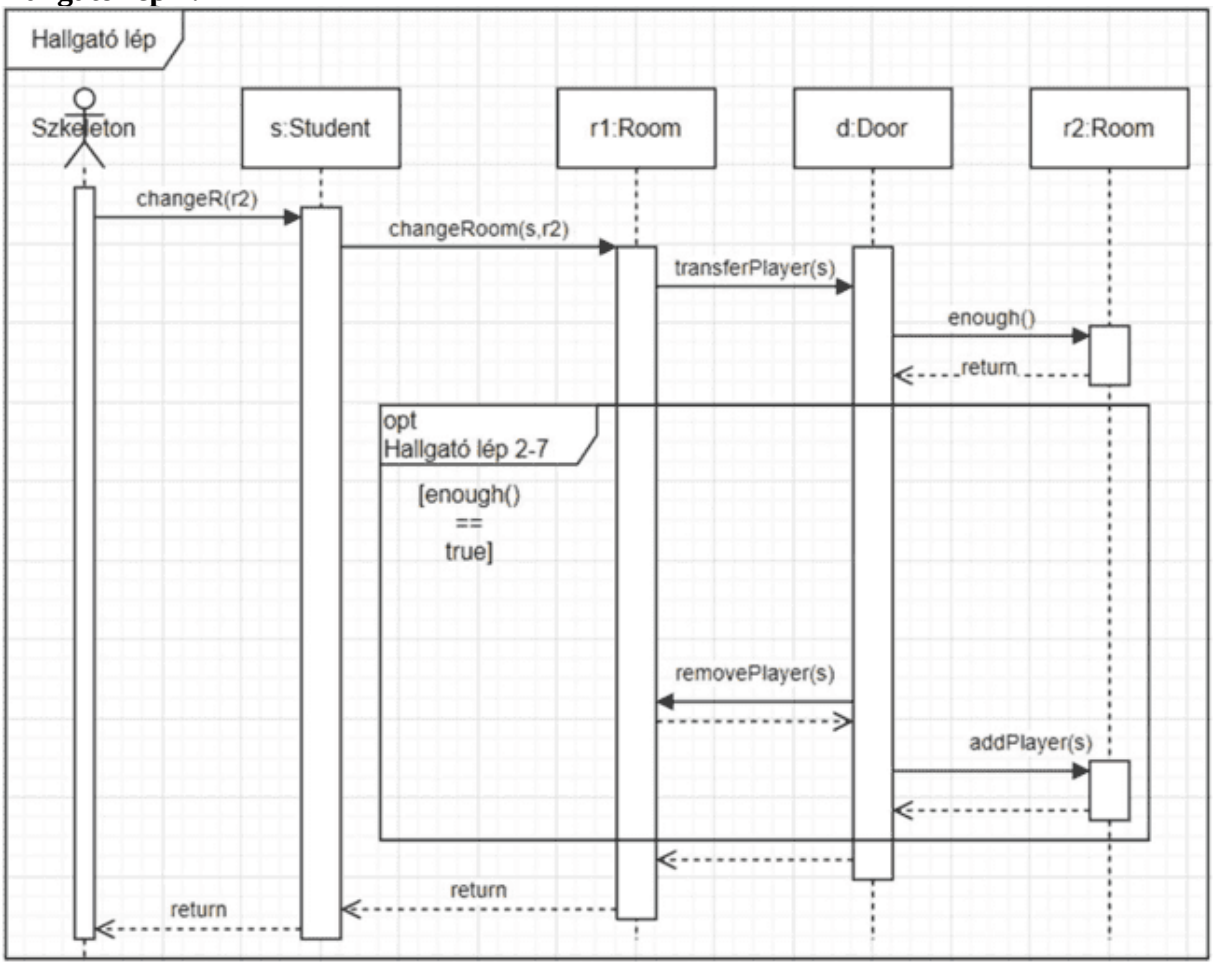
Csúszóssá válás



Split

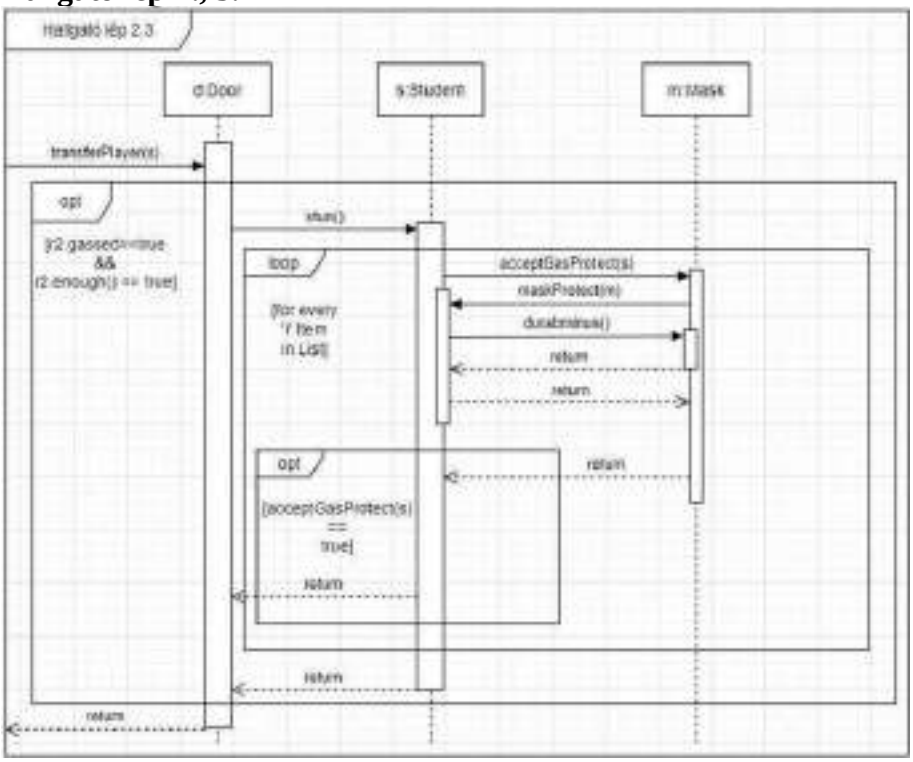


Hallgató lép 1.

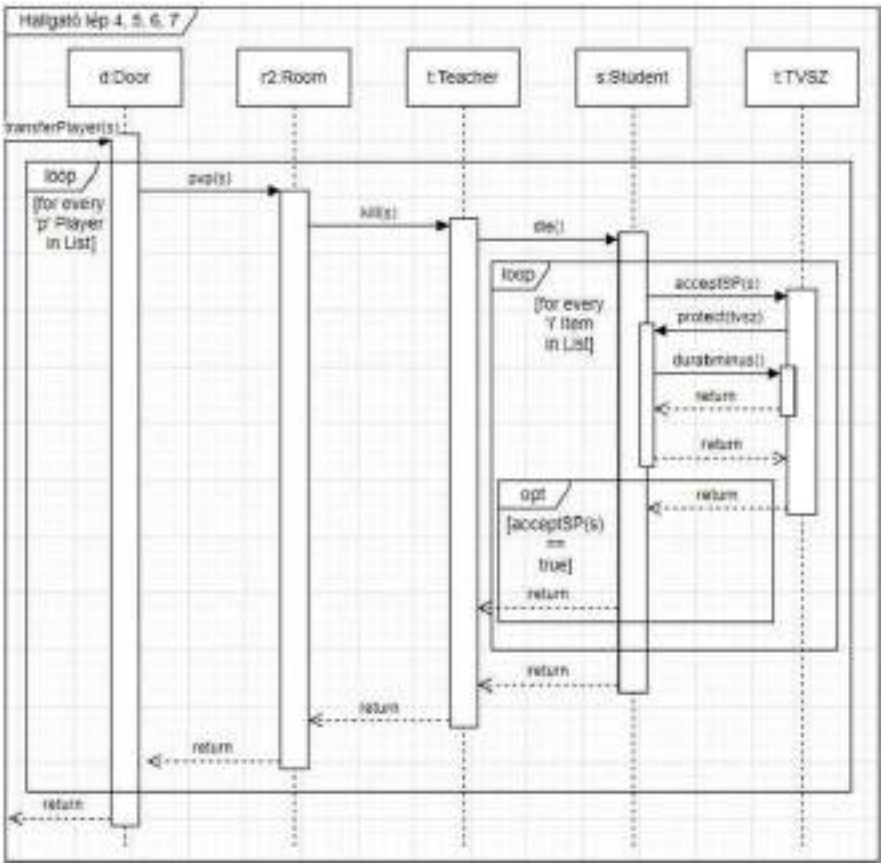


A fenti (Hallgató lép) szekvenciadiagram
a további, Hallgató lép 2-7., diagramok alapja,
ahol az enough() igaz értékkel tér vissza.

Hallgató lép 2., 3.

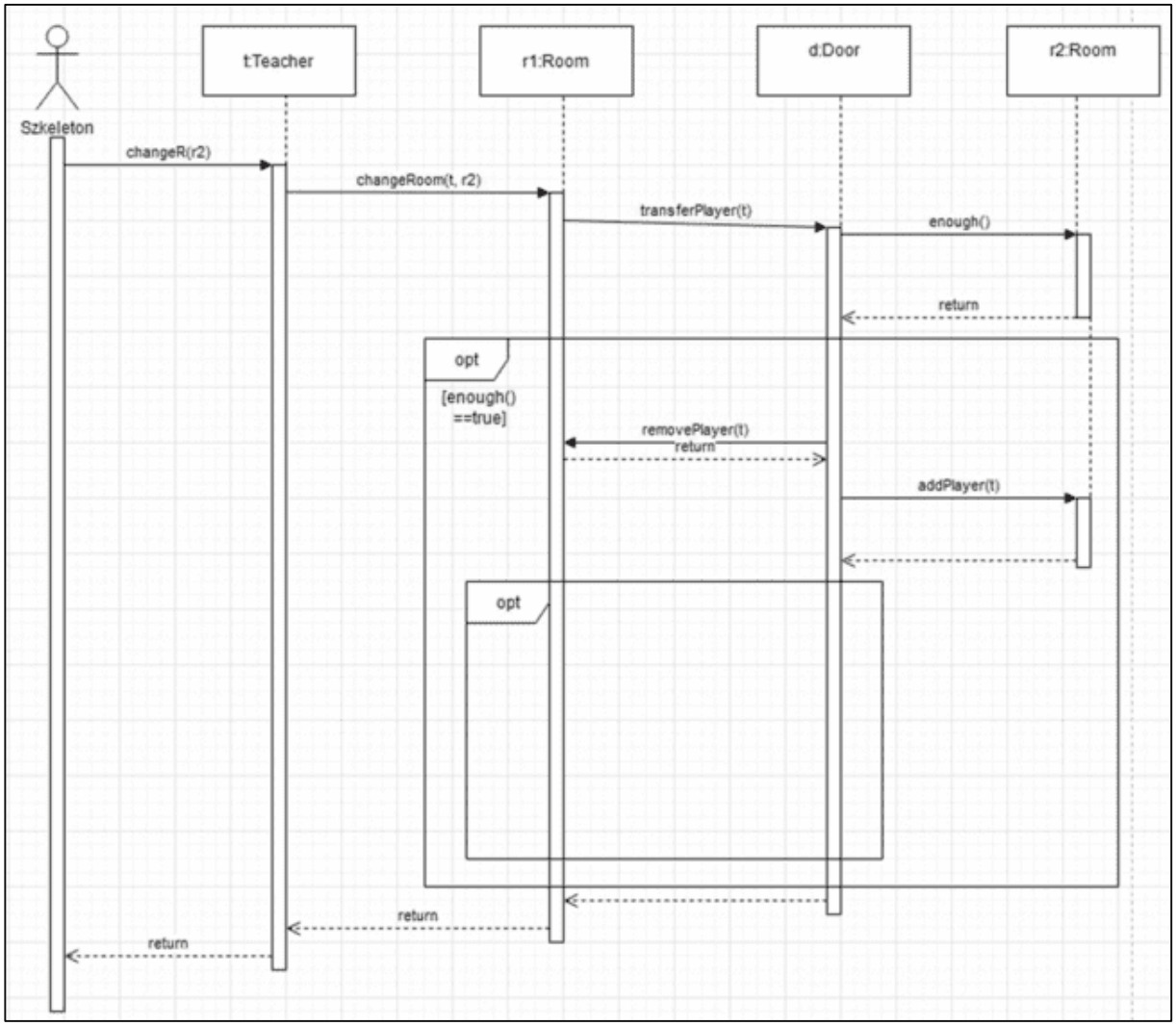


Hallgató lép 4., 5., 6., 7.



ripgyork

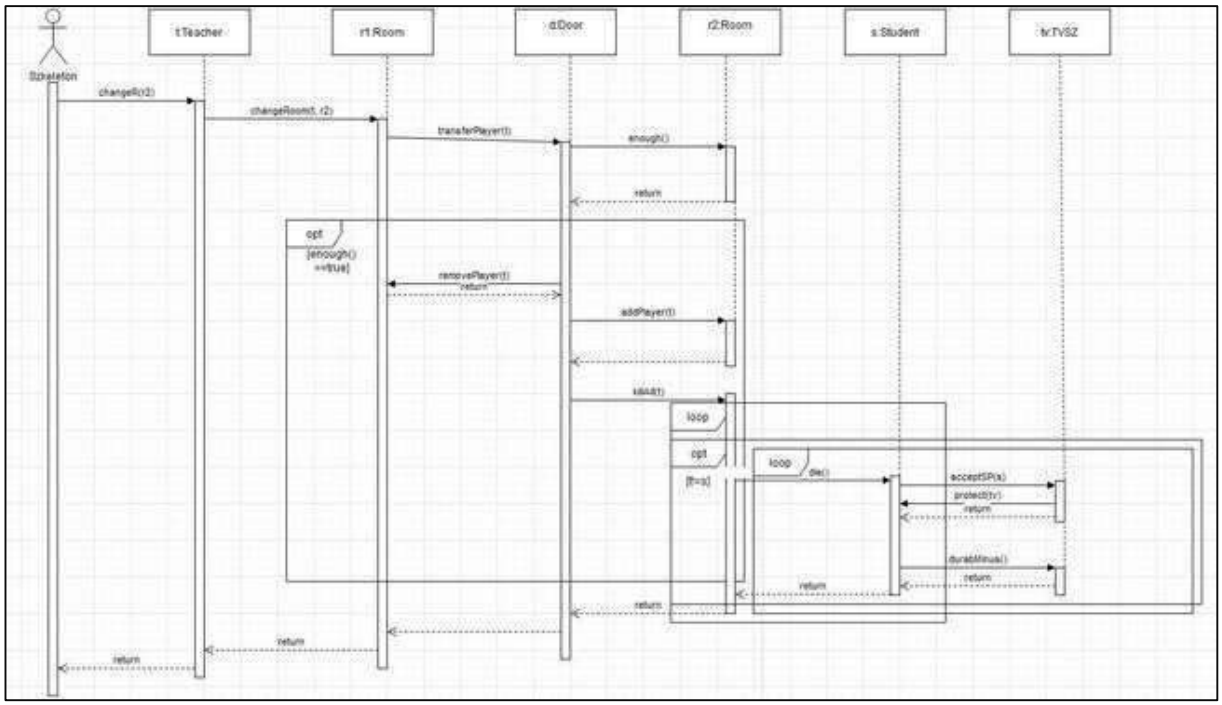
Oktató lép 1., 7.



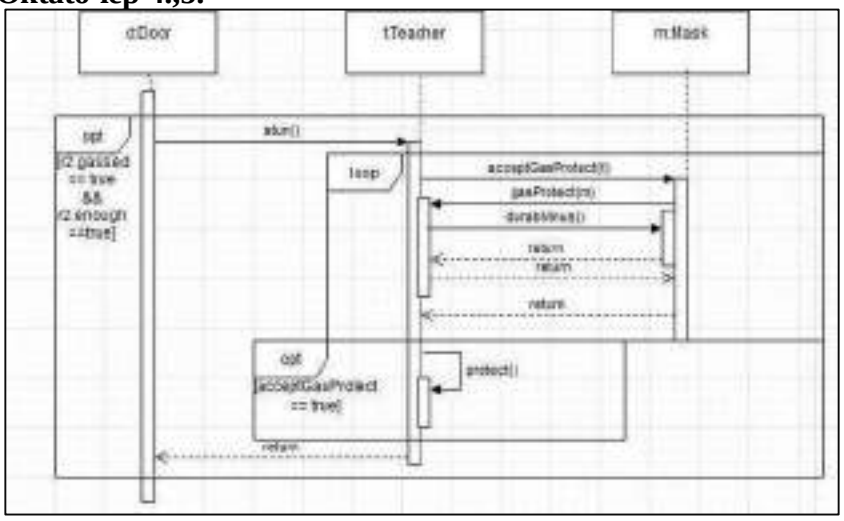
(A következő diagramok egy része ebből indul ki és feltételezi, hogy az oktató sikeresen be tudott lépni a szobába. A fenti diagramban véletlenül benne maradt egy opt ablak, jelentése nincsen.)

ripgyork

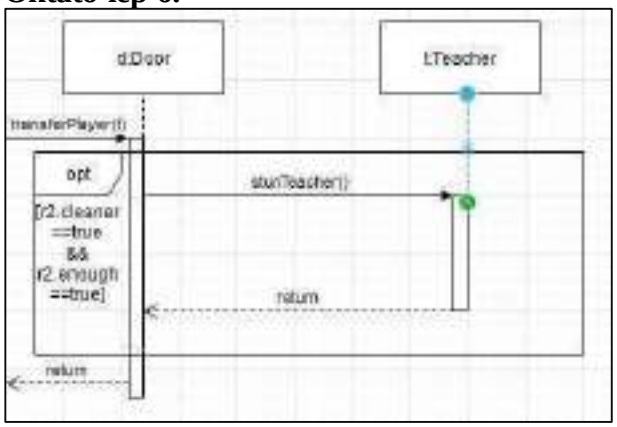
Oktató lép 2.,3.



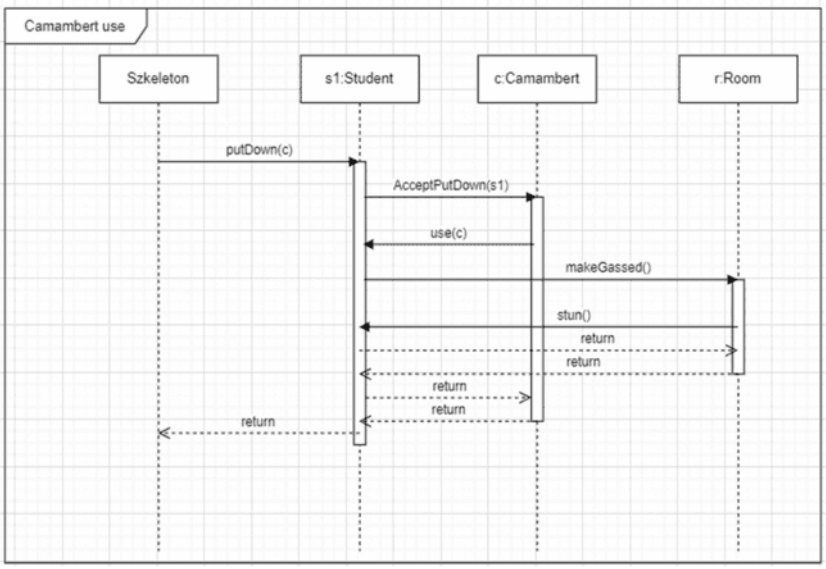
Oktató lép 4.,5.



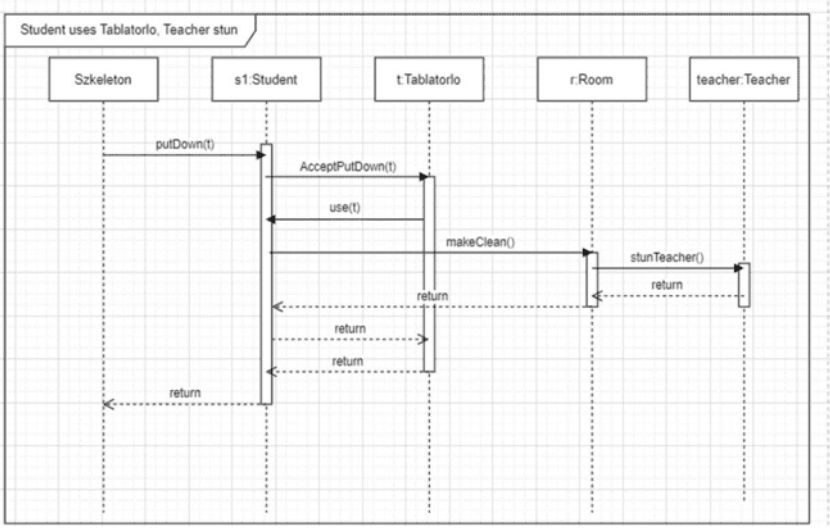
Oktató lép 6.



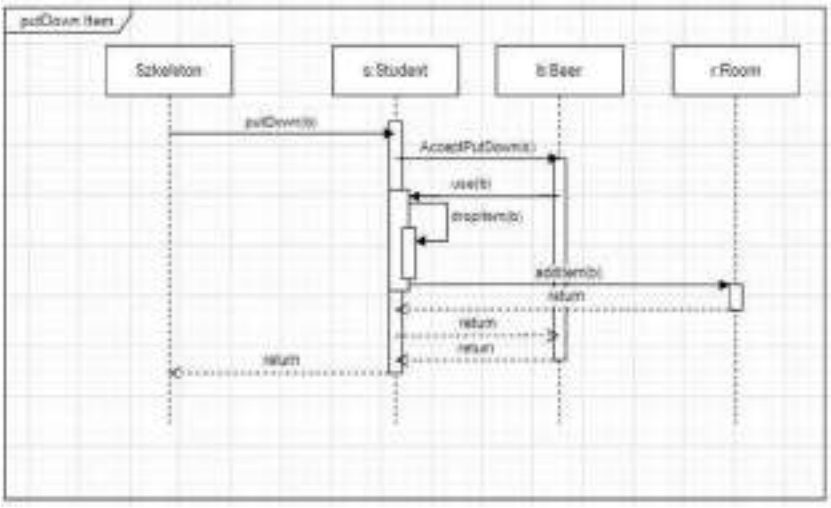
Diák használ Camambert



Diák használ Tablatorlo, Tanár stun

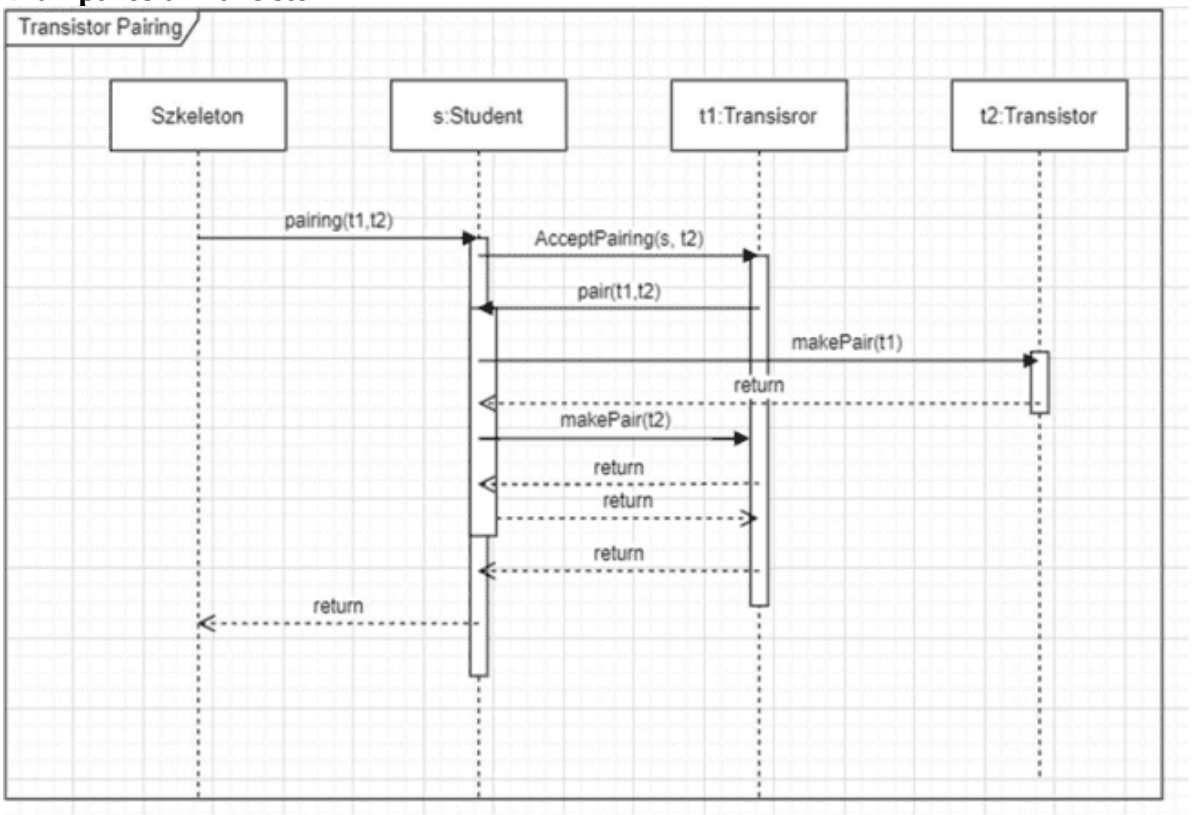


Diák letesz tárgy

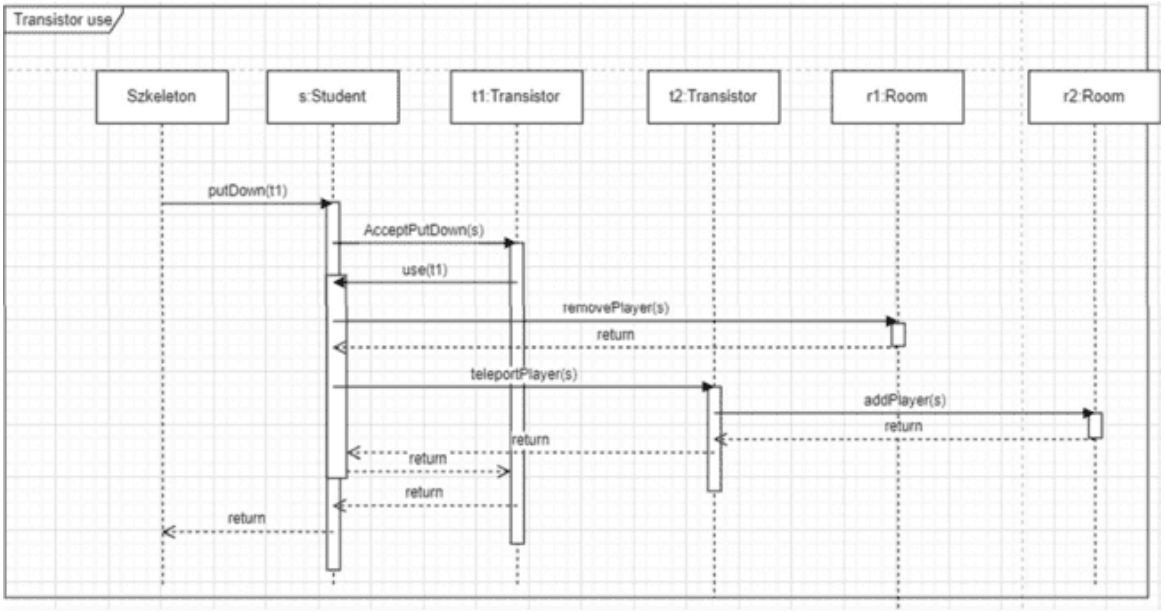


ripgyork

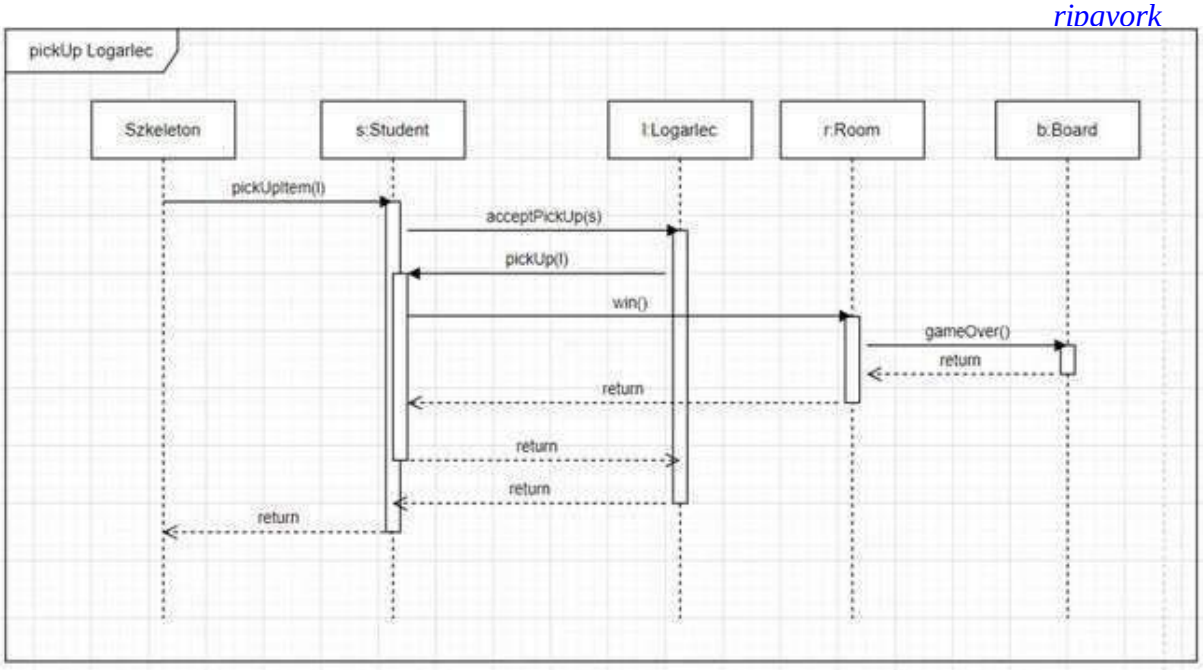
Diák párosít Transistor



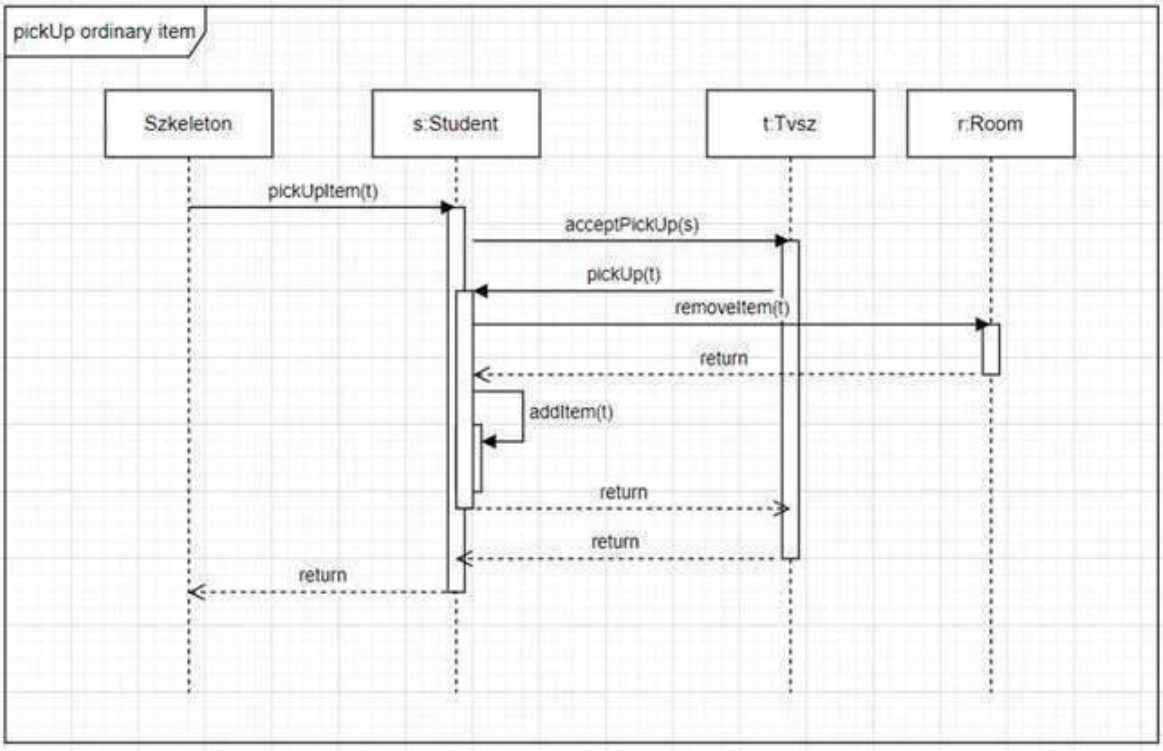
Diák használ Transistor



ripavork



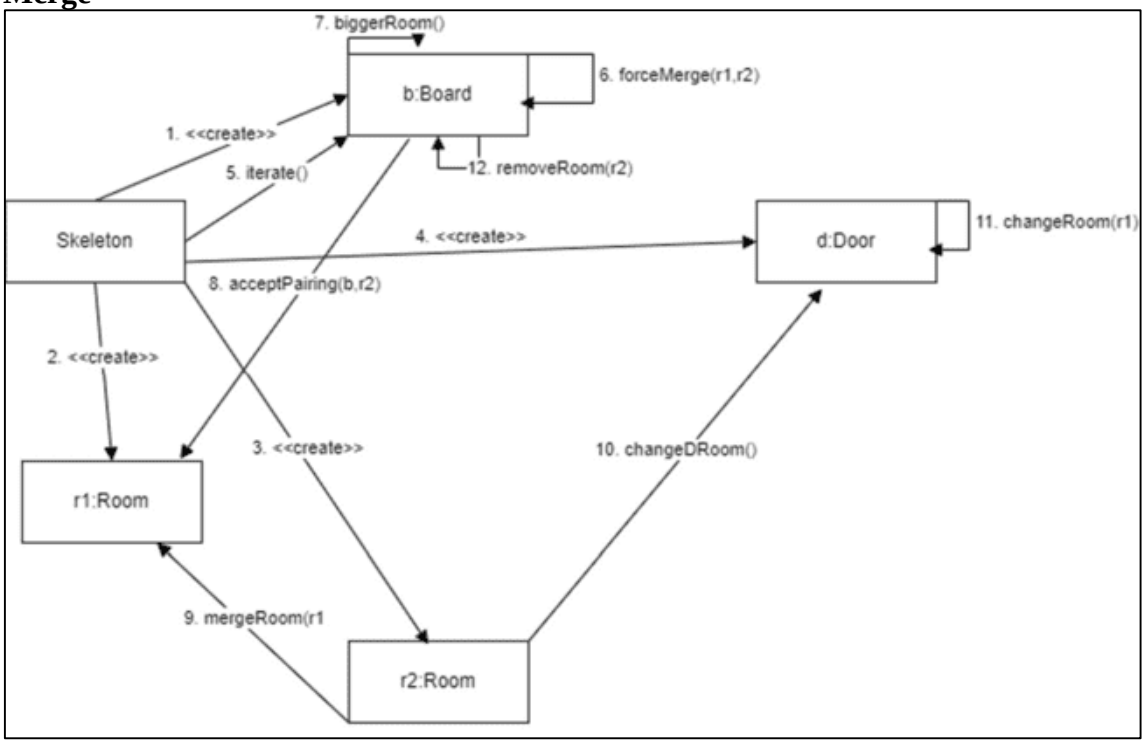
Sima (nem Logarléc) tárgy felvétele



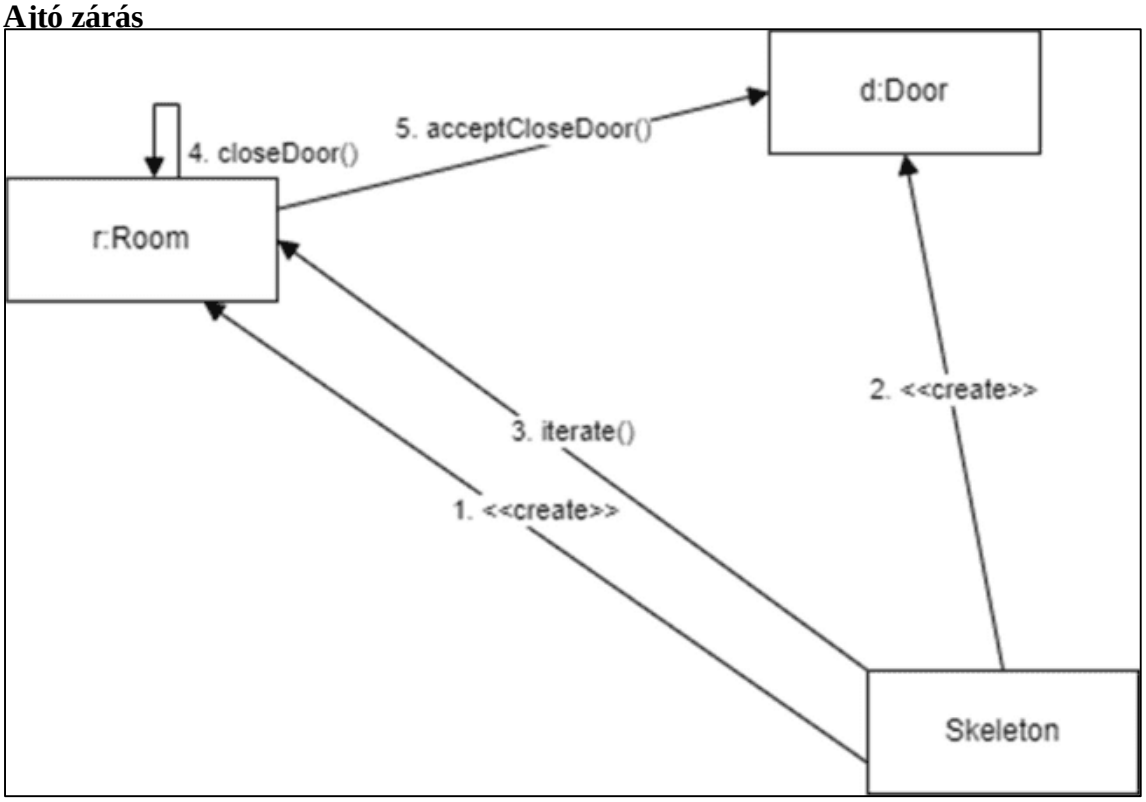
5.4 Kommunikációs diagramok

Az olvashatóság érdekében egyes diagrammokon nem jelöltünk minden visszatérési értéket.

Merge

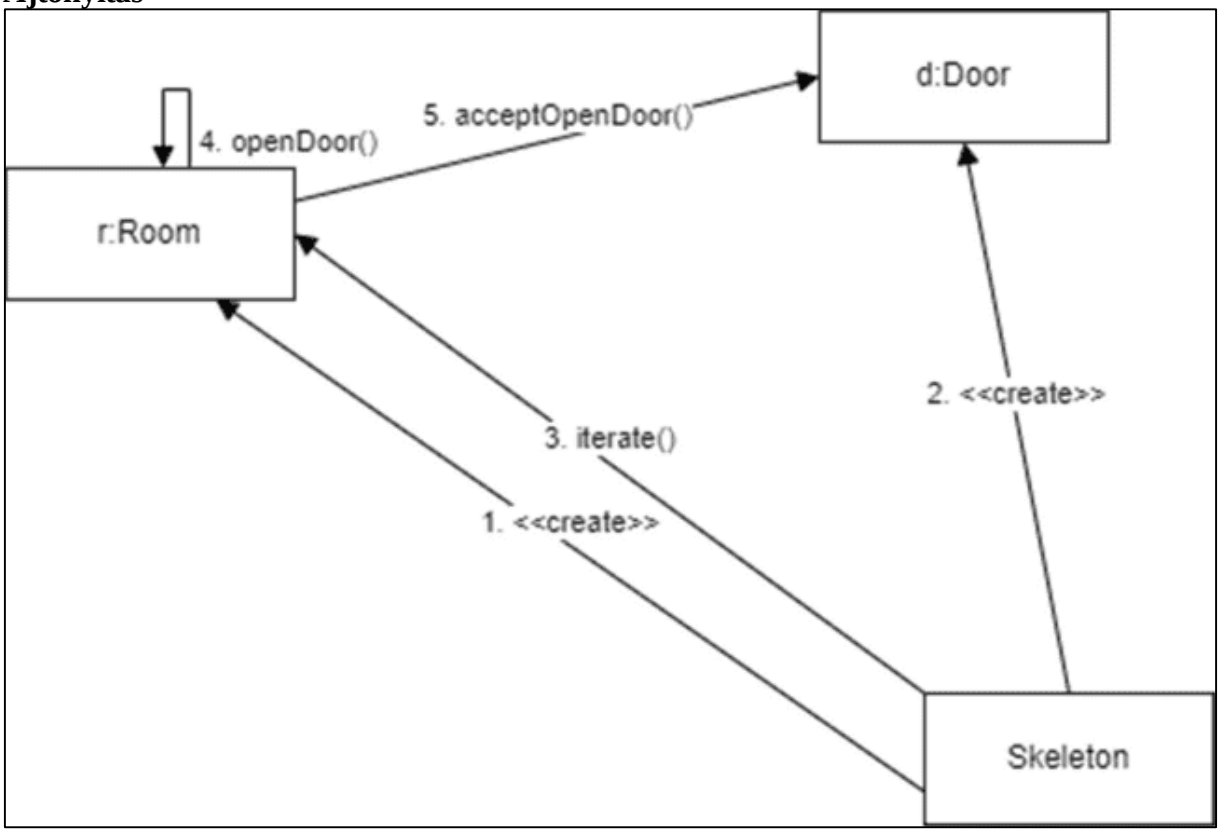


Ajtó zárás

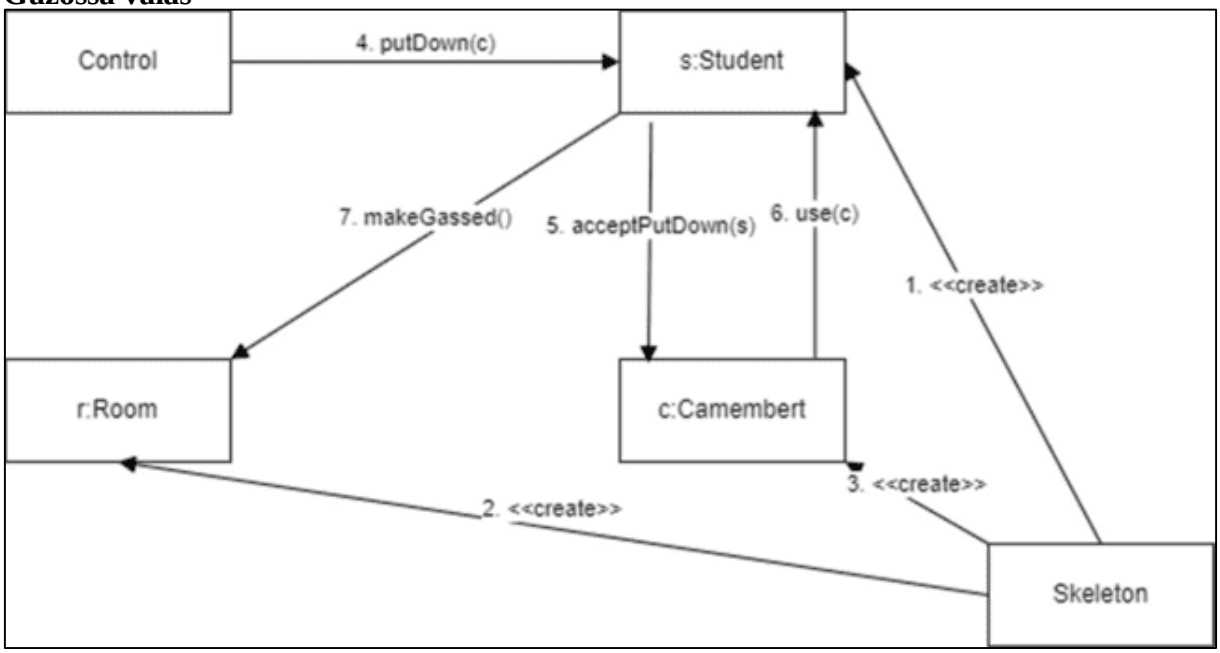


ripgyork

Ajtónyitás

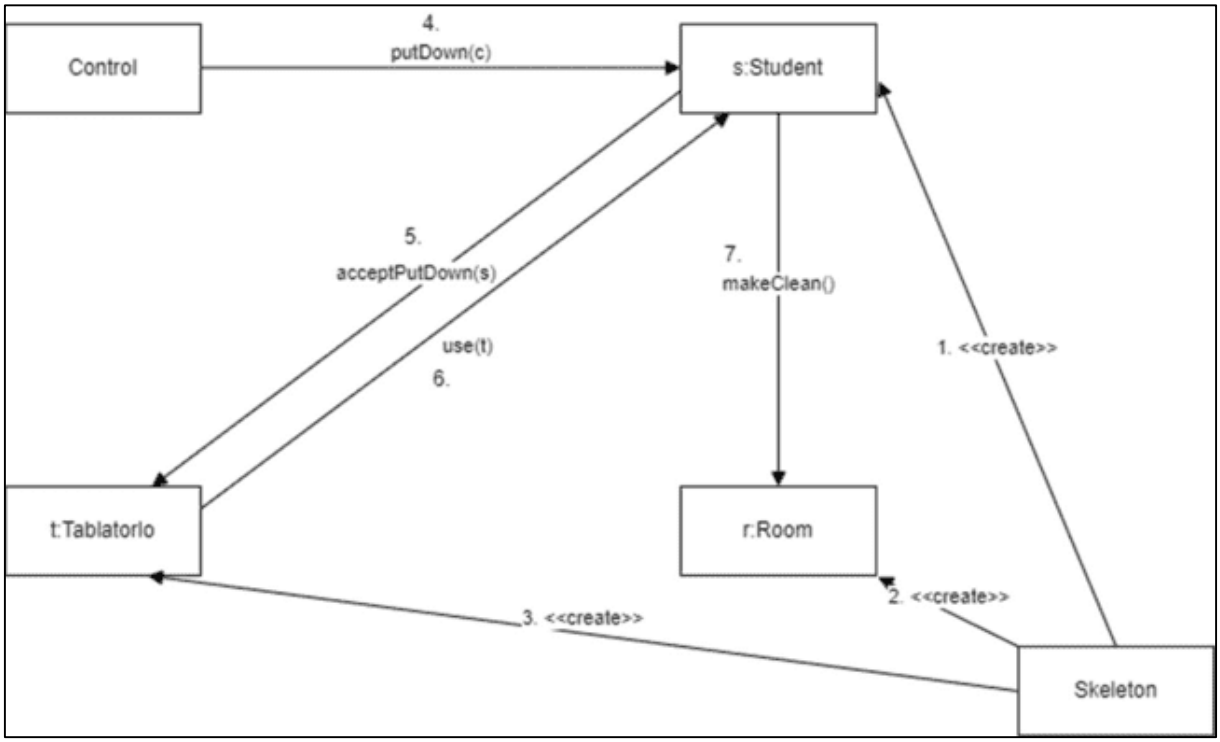


Gázossá válás

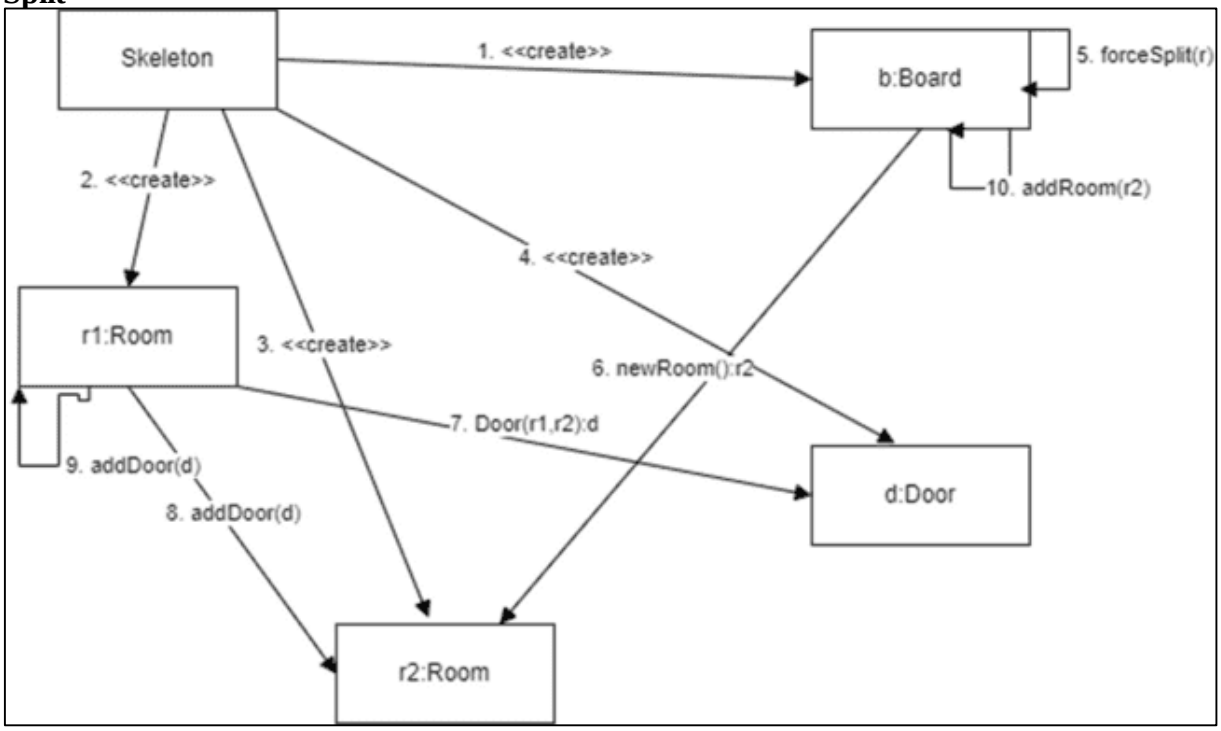


ripgyork

Csúszóssá válás

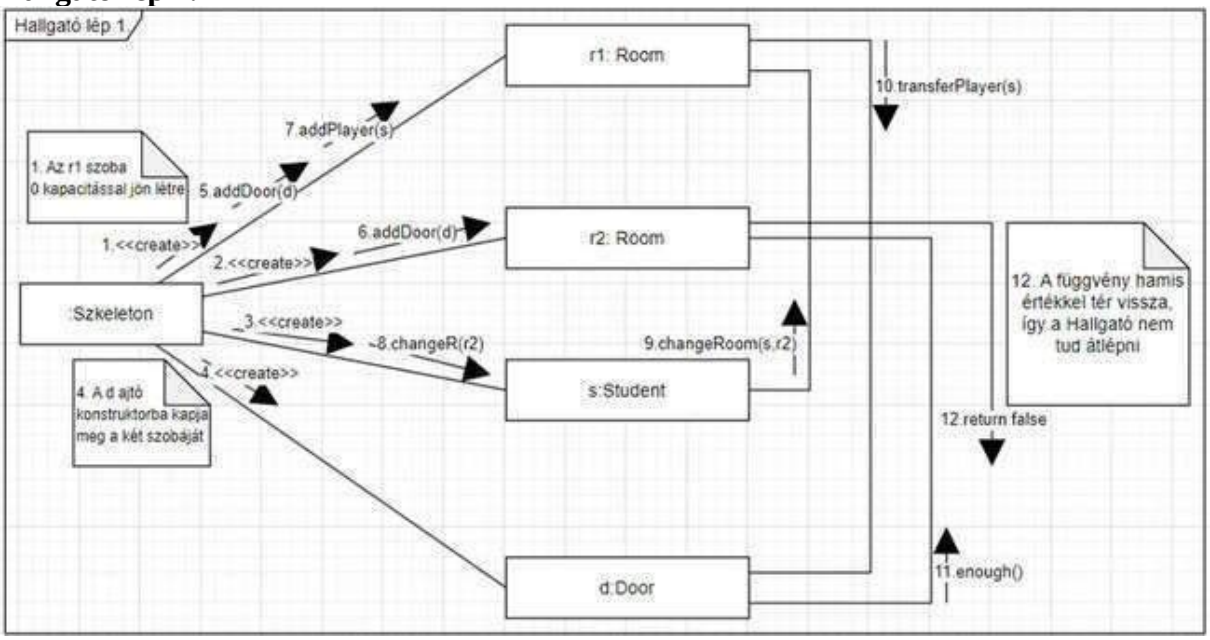


Split

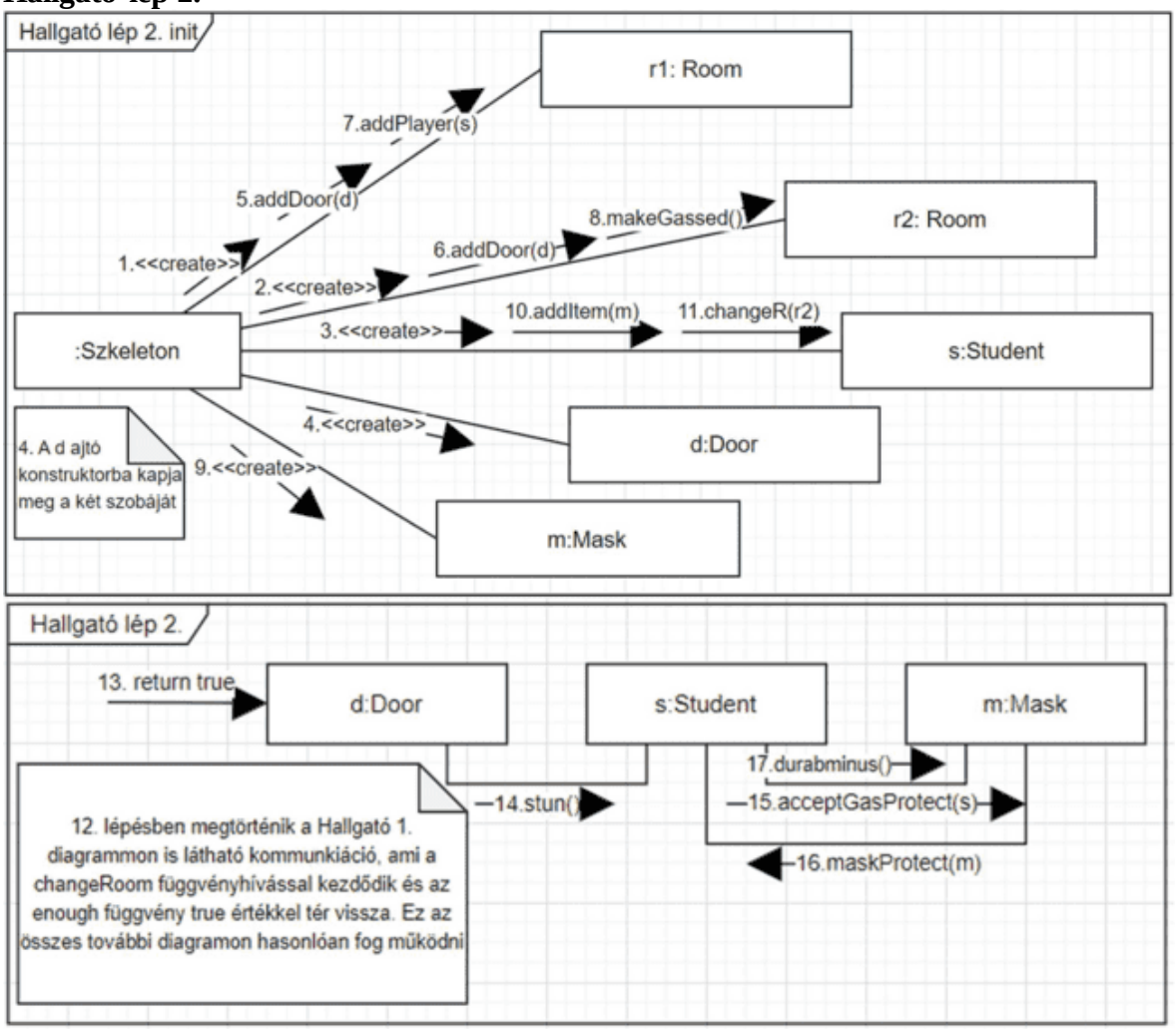


ripgyork

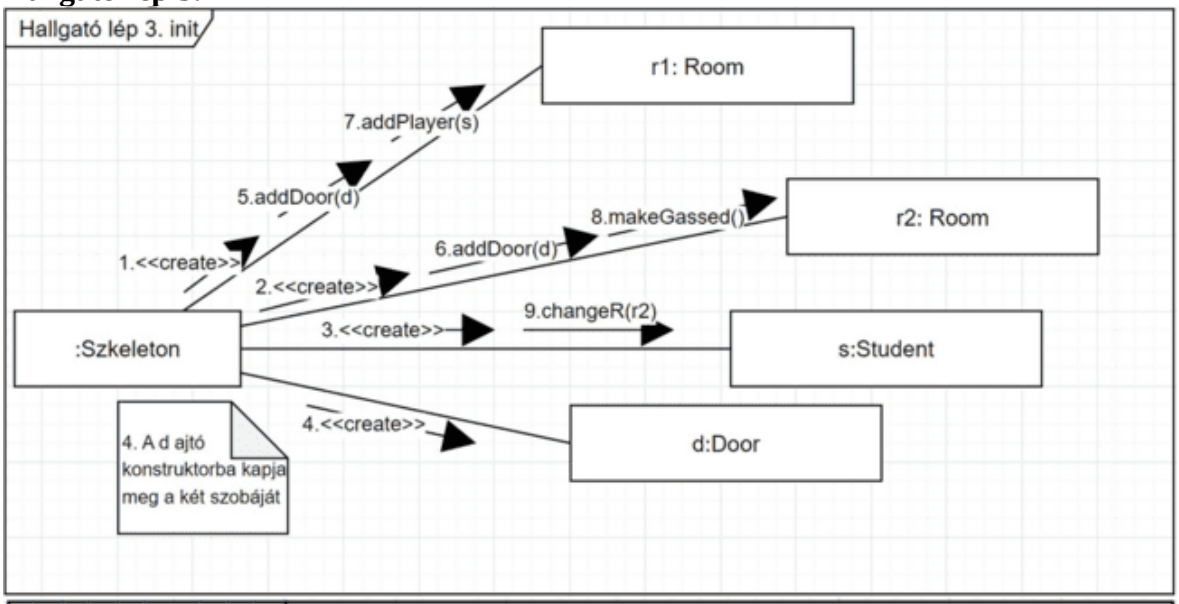
Hallgató lép 1.

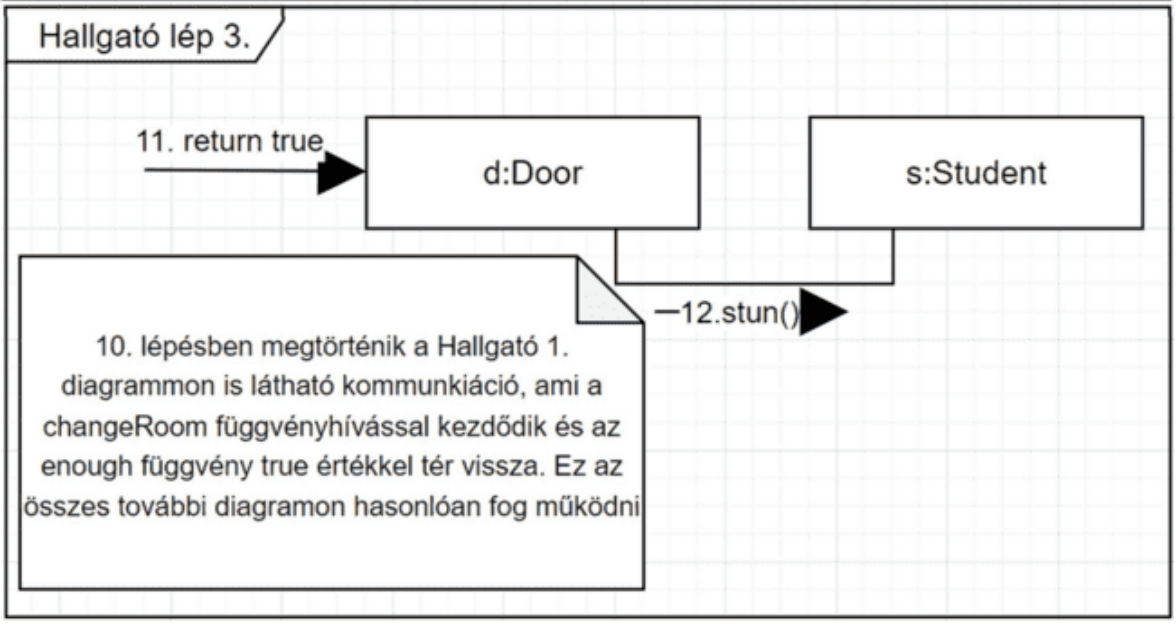


Hallgató lép 2.

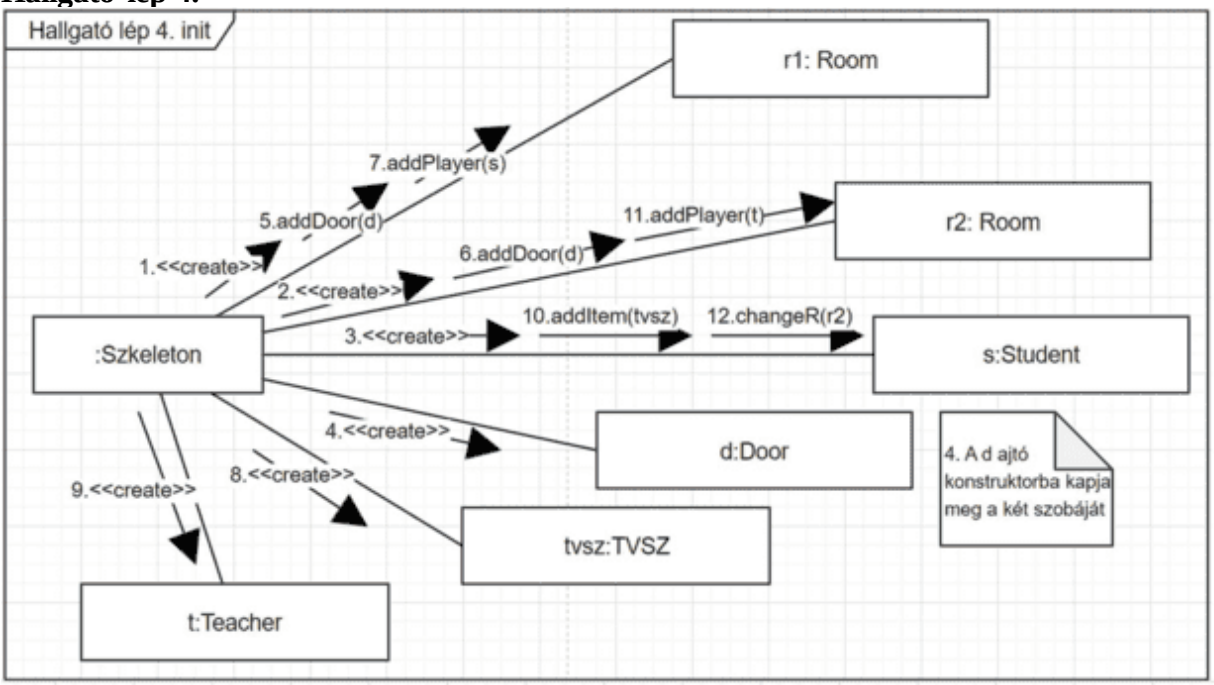


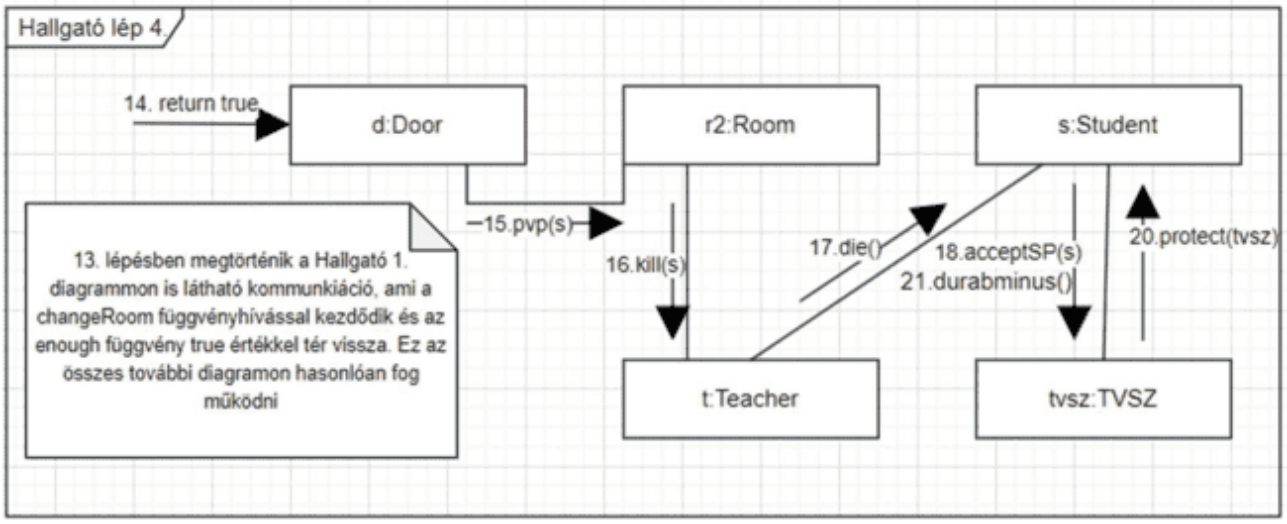
Hallgató lép 3.



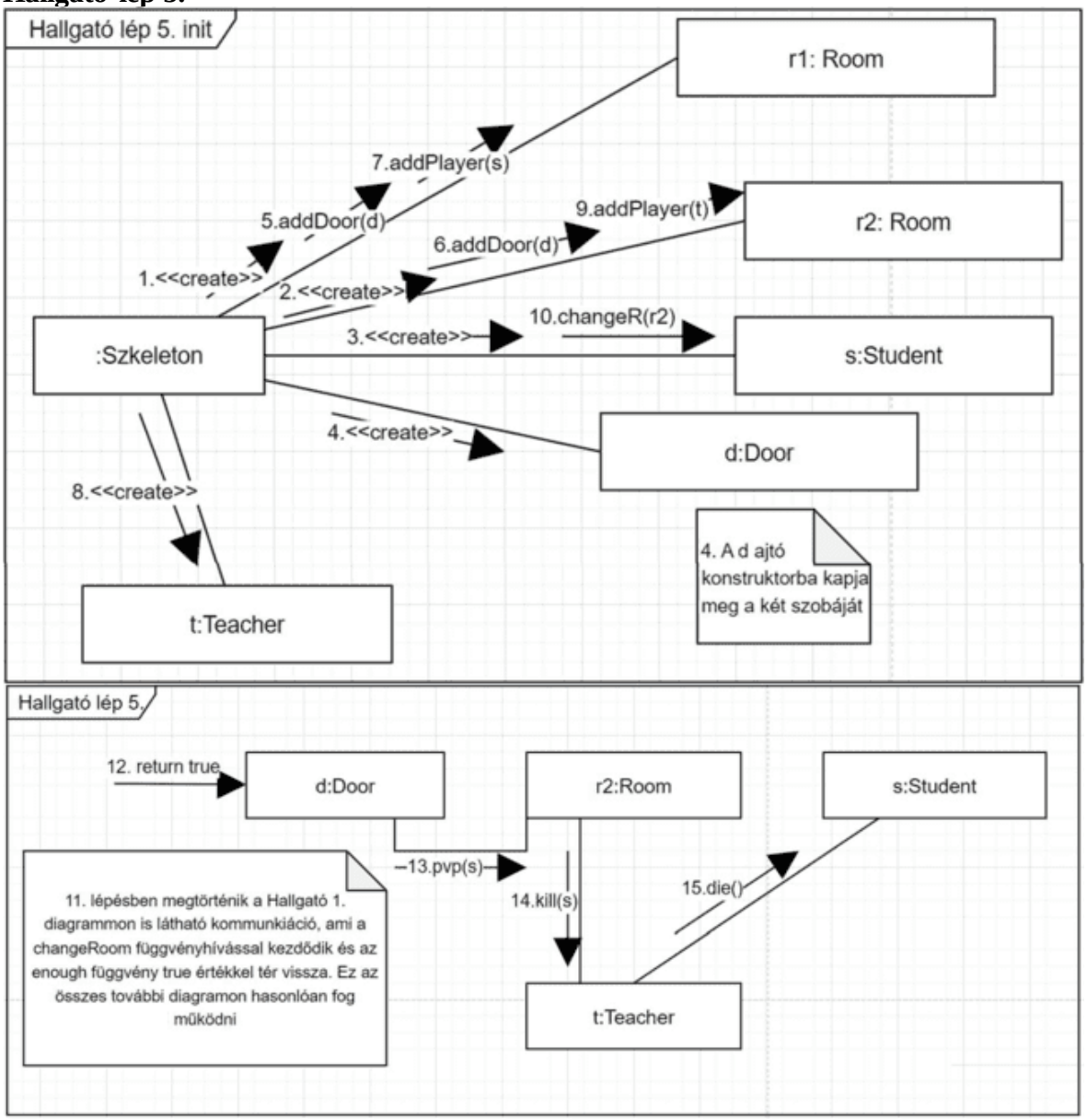


Hallgató lép 4.



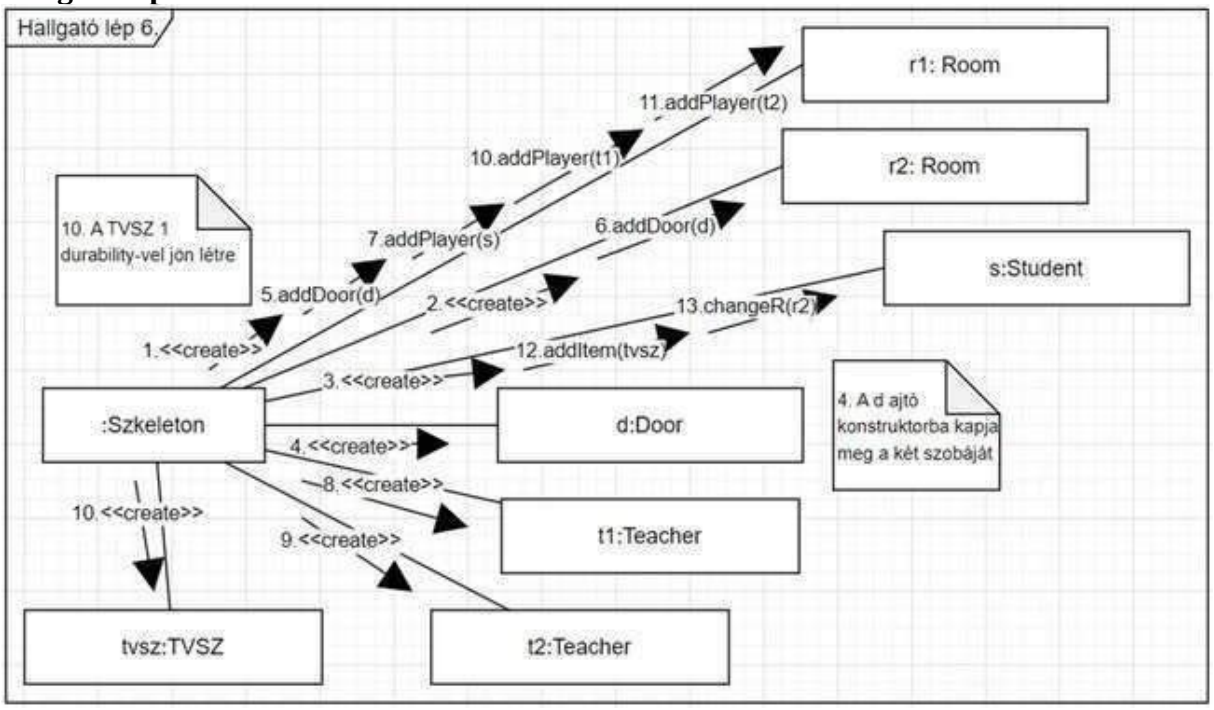


Hallgató lép 5.



ripgyork

Hallgató lép 6.



A további kommunikációk az eddigi diagrammok kombinációja. Ehhez ezért külön ábra nem tartozik.

14. Lépésben folytatódik a **Hallgató lép 1**. diagrammon látható **9.changeRoom()** függvényhívással. Az **enough()** visszatér igazzal.

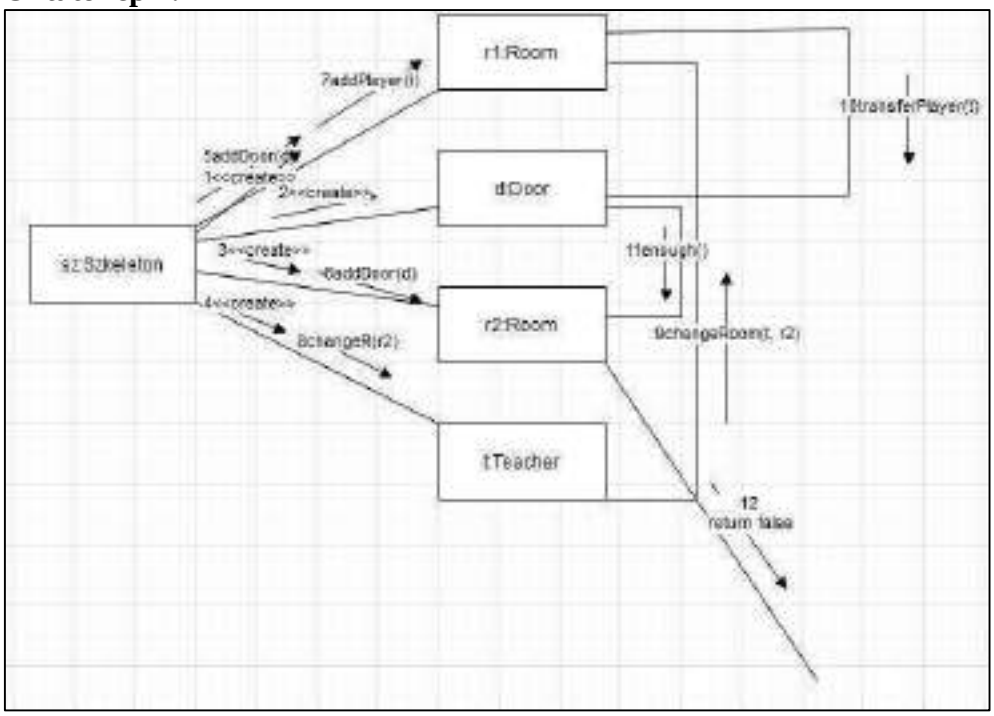
A **Hallgató lép 4.** ábra **14.return true** kommunikációval folytatódik. Mivel 2 oktató volt és a hallgató csak 1-et tudott védeni ezért a t1 oktató **kill()** függvénye hamissal tér vissza.

Így a t2 oktatóra is lefut ez a folyamat. Ebben az esetben a **Hallgató lép 5.** diagram **14.kill(s)** függvényhívás következik be, amit a **t1 helyett t2** végez. A kommunikáció vége megegyezik a **Hallgató lép 5.** diagrammal.

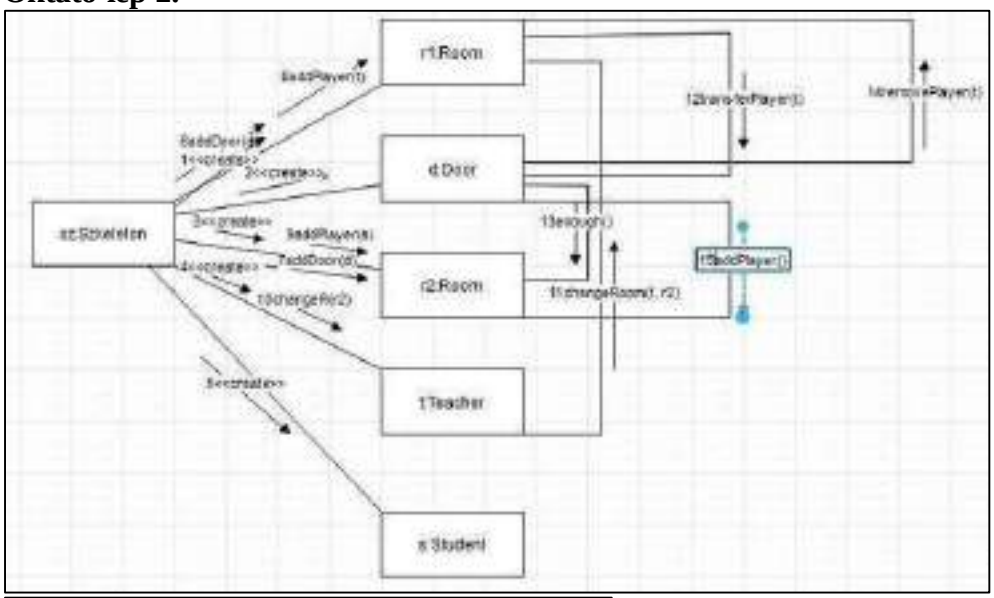
A **Hallgató lép 7**. használati eset inicializálása megegyezik a **Hallgató lép 6**. ábrán láthatóval, annyi különbséggel, hogy a **TVSZ 2 durabilitás**sal jön létre. A folyamat megegyezik a **Hallgató lép 6** ábrával, annyi különbséggel, hogy amikor a **t2** oktató is meghívja a **kill()** függvényt, akkor a **Hallgató lép 4**. ábra szerint fognak zajlani a kommunikációk, ez a vége is.

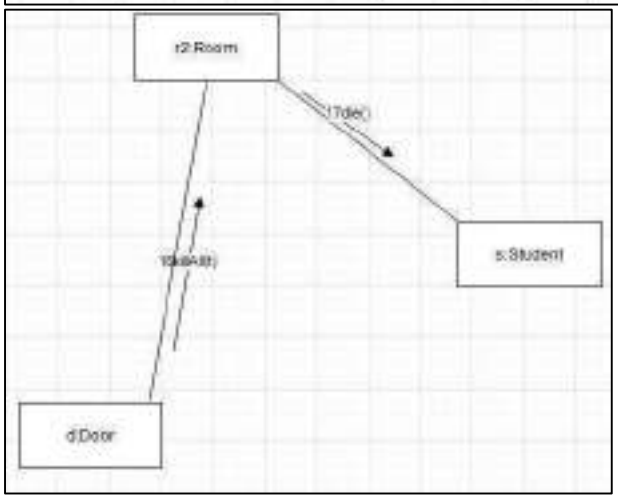
ripgyork

Oktató lép 1.

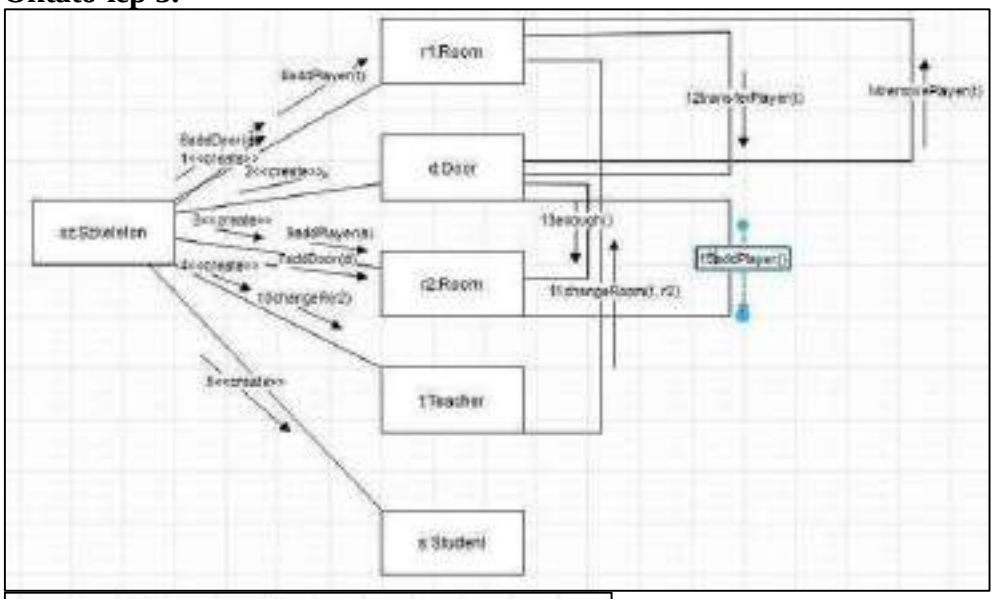


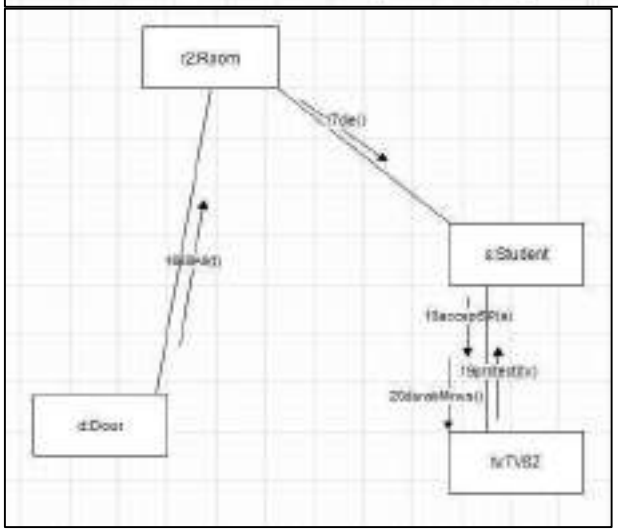
Oktató lép 2.



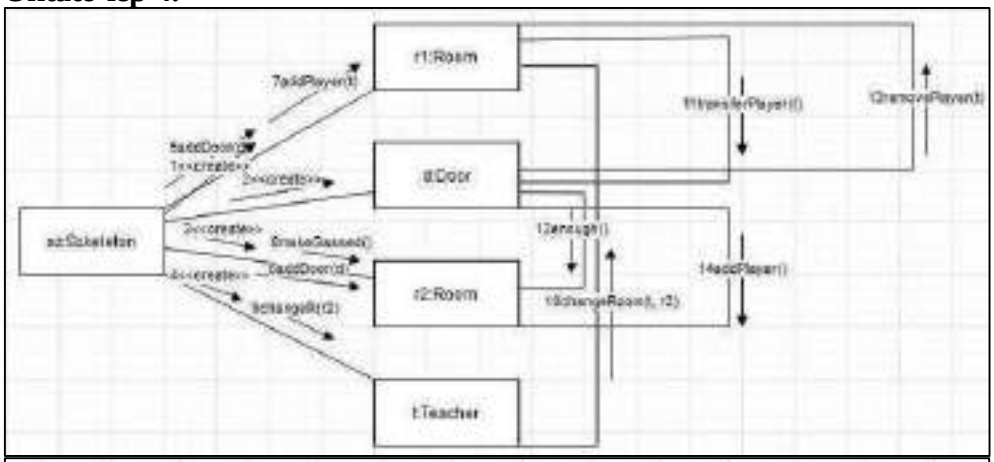


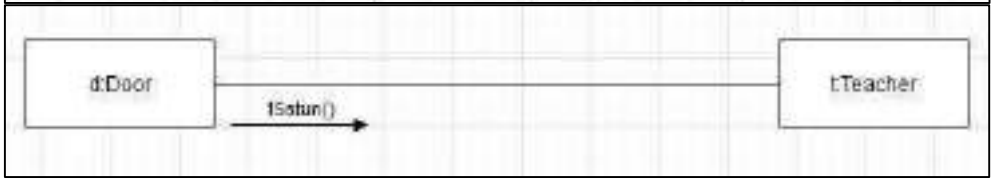
Oktató lép 3.



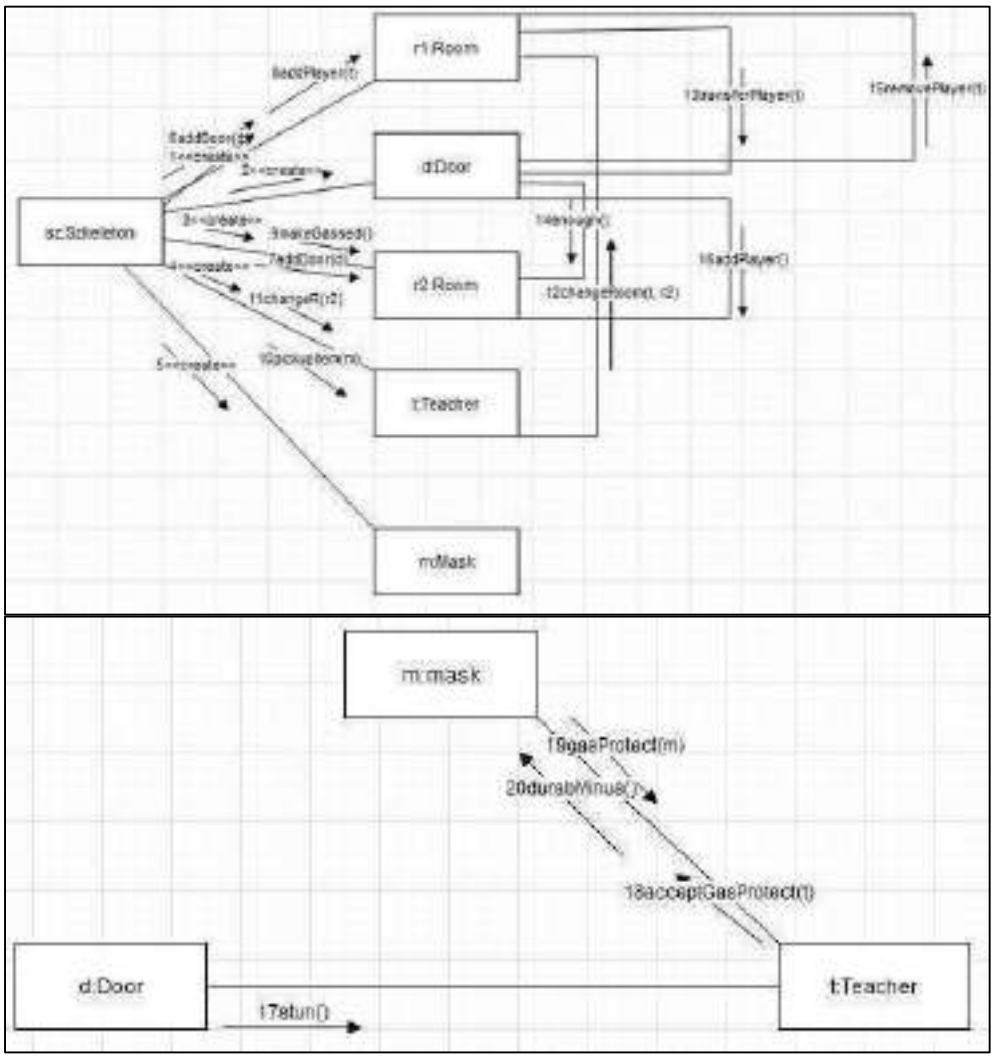


Oktató lép 4.

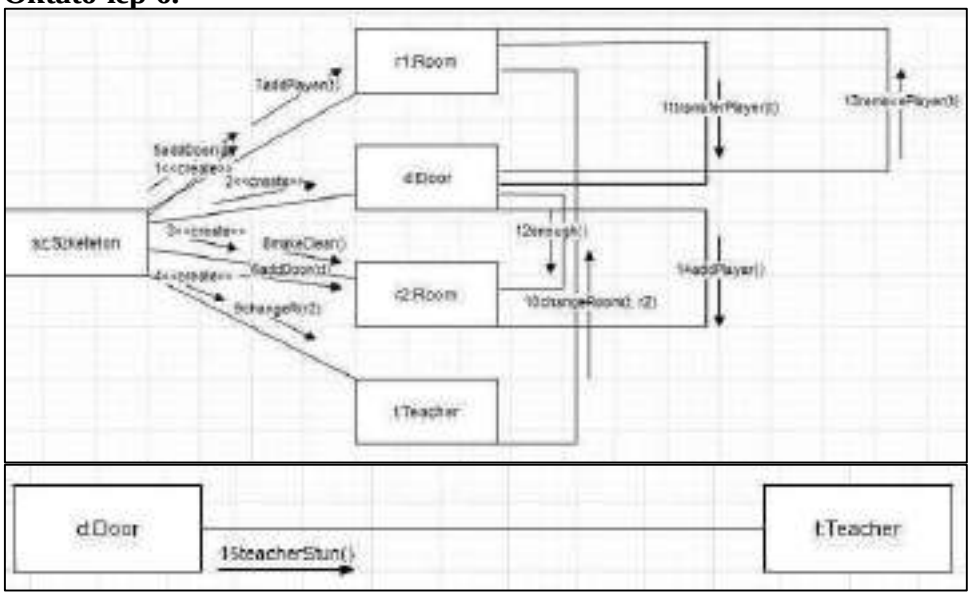




ripgyork

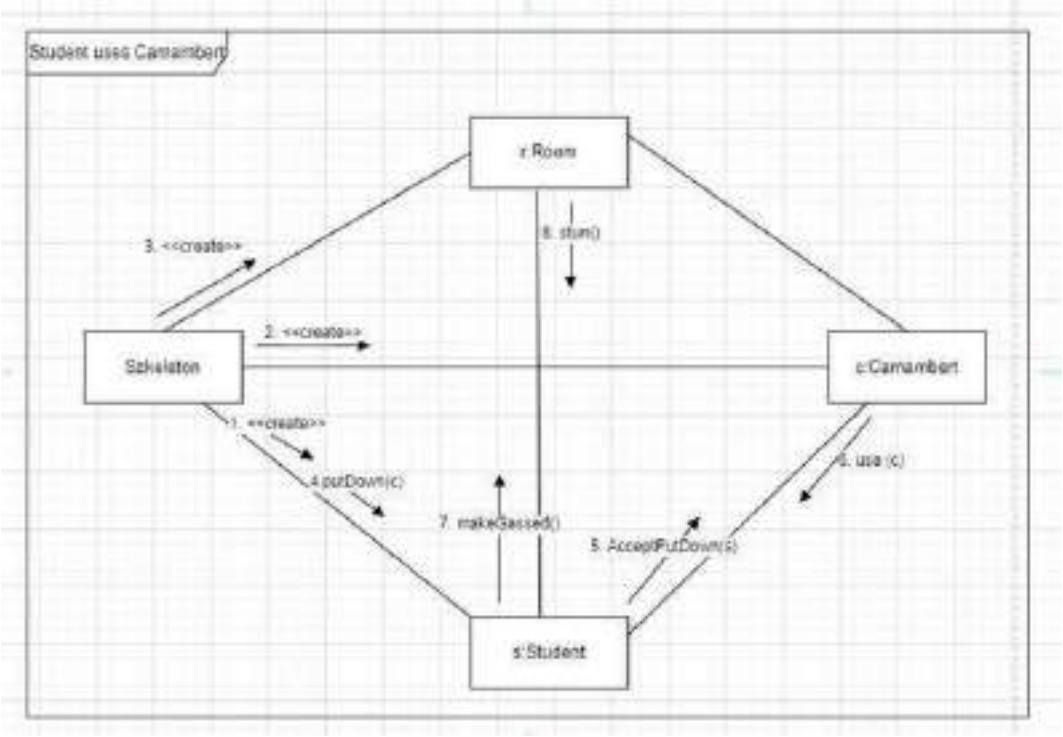


Oktató lép 6.

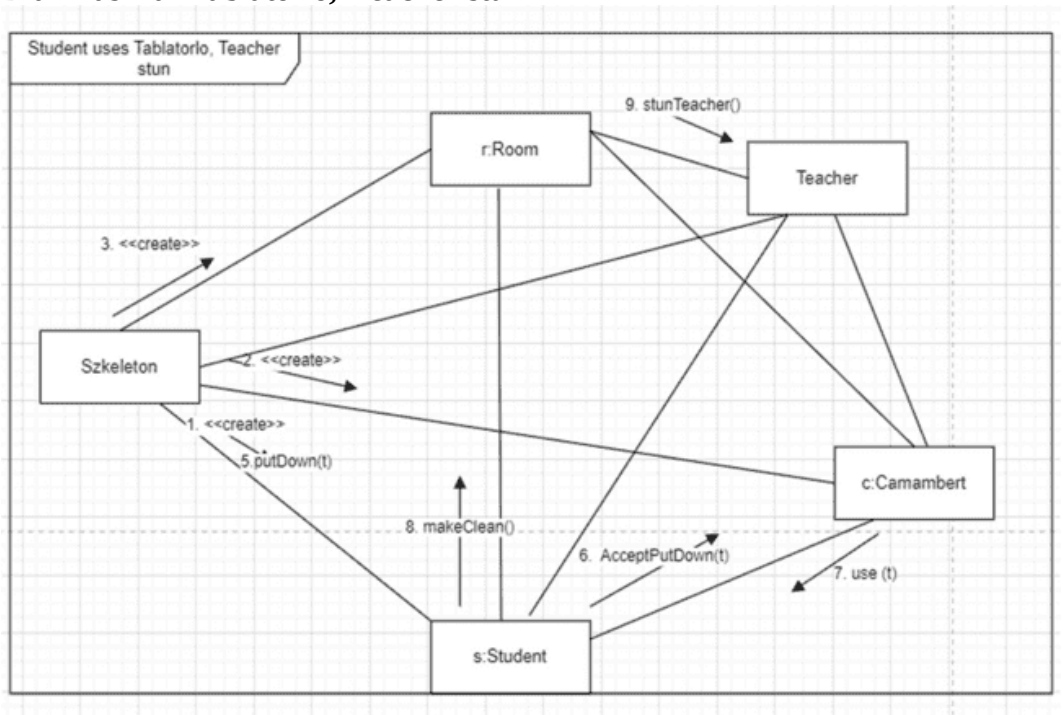


ripgyork

Diák használ Camambert



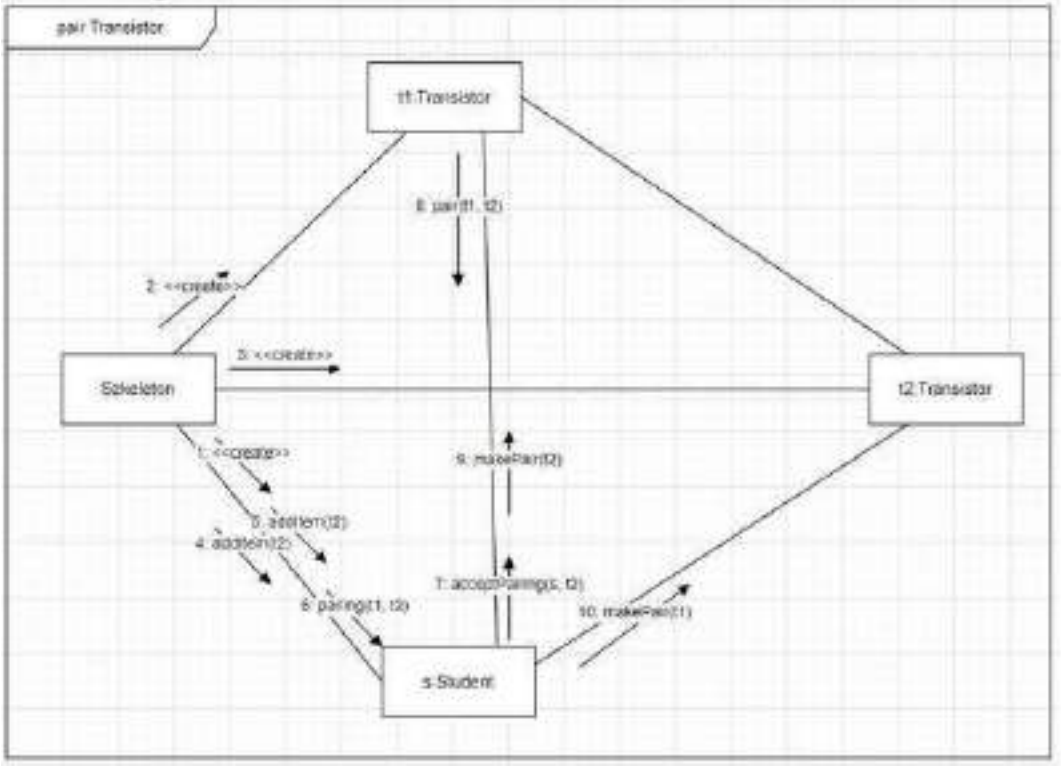
Diák használ Tablatorlo, Teacher stun



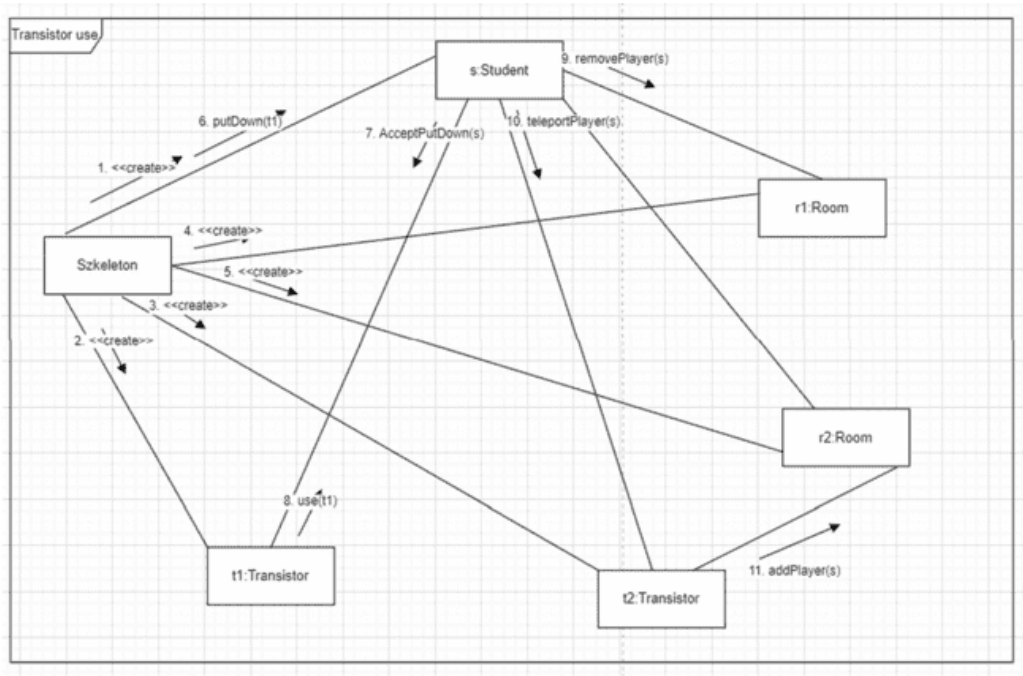
ripgyork

ripgyork

Transistor párosít

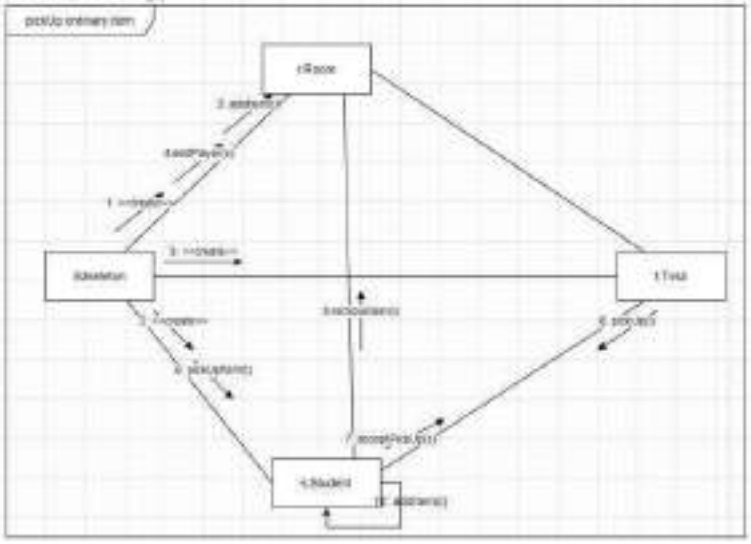


Transistor használat

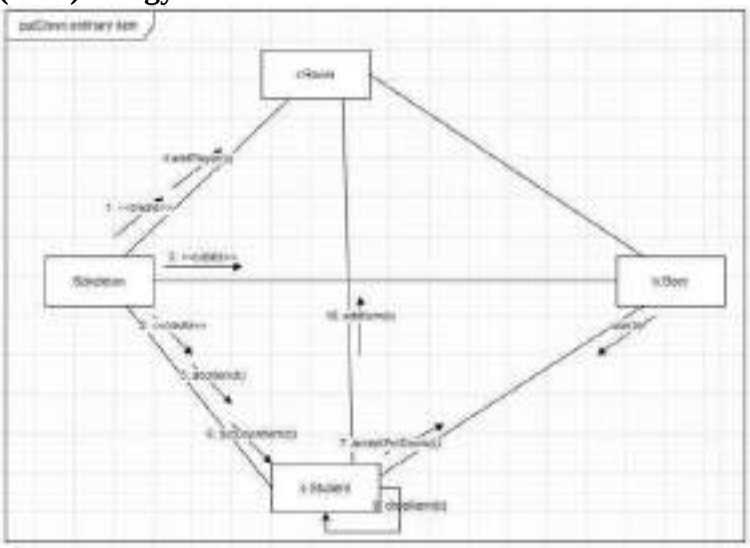


ripgyork

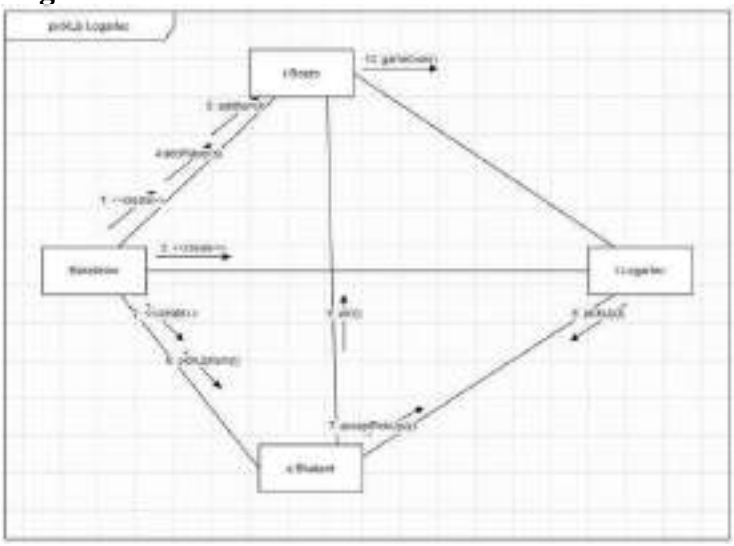
(sima) Tárgyfelvétel



(sima) Tárgy letesz



Logarlec felvétel



5.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2024.03.15. 9:00	3 óra	Fodor A Fodor D Földi Ludányi Mikola	Értekezlet. Döntés a feladat elkészítéséről, nekiállásról és a folytatásról.
2024.03.15. 14:00	3,5 óra	Fodor A. Fodor D.	Előzetes osztálydiagramm javítása, további hibás pontok keresése, felmérése
2024.03.15. 17:00	3 óra	Ludányi	Use case-ek felvétele a dokumentumba, szekvencia diagramok egy része
2024.03.15 18:00	1,5 óra	Földi	Use case-ek kidolgozása
2024.03.16 16:00	2,5 óra	Földi	Valamennyi use case és a hozzá tartozó diagrammok kidolgozása
2024.03.16 17:00	1 óra	Földi Ludányi Mikola Fodor A. Fodor D.	Kisebb csapatmegbeszélés, kérdéses részel megvitatása, megoldás keresése
2024.03.16	2 óra	Mikola	5.2-es rész átgondolása és megírása. Oktatók mozgásának use-case-einek összegyűjtése, leírása.
2024.03.16 21:00	3 óra	Földi	Use-case diagrammok javítása, pontosítása, újragondolása
2024.03.16 22:00	2 óra	Ludányi	Szekvencia és kommunikációs diagrammok
2024.03.17 9:00	5 óra	Földi	Use case diagrammok javítása, kommunikációs diagrammok tervezése
2024.03.17 12:00	2 óra	Mikola	Szekvenciadiagrammok egy részének kidolgozása
2024.03.17 15:00	2 óra	Mikola	Szekvenciadiagramok befejezése, kommunikációs diagrammok elkezdése
2024.03.17. 16:00	1,5 óra	Földi	Szekvencia és kommunikációs diagrammok véglegesítése
2024.03.17 17:00	2 óra	Fodor A.	Szekvenciadiagrammmok elkezdése, kommunikációs diagrammok elkezdése

2024.03.17 17:00	2 óra	Fodor D.	Szekvenciadiagrammmok elkezdése, kommunikációs diagrammok elkezdése
2024.03.17 18:00	1,5 óra	Mikola	Kommunikációs diagrammok befejezése, szekvencia diagramok javítása
2024.03.17 18:00	2 óra	Ludányi	Szekvenciadiagrammok javítása
2024.03.17. 19:00	3 óra	Fodor A. Fodor D.	Közös munka a diagrammokon, ötletelés
2024.03.17 22:00	1 óra	Mikola Fodor A. Fodor D. Földi Ludányi	Értekezlet: Szobák állapotváltozásával, szobák függvényeivel kapcsolatos kérdések megbeszélés
2024.03.17 23:00	3 óra	Ludányi	Szekvenciadiagrammok véglegesítése a megbeszéltek alapján, kommunikációs diagrammok befejezése
2024.03.18 6:00	1,5 óra	Fodor A. Fodor D.	Saját feladatok befejezése, dokumentum szerkesztése, helyesírási hibák javítása
2024.03.18 7:00	4 óra	Földi	A 4. és 5. részek konzisztens állapotba hozása. Use case diagrammok elkészítése
2024.03.18 8:30	3 óra	Ludányi	Szekvenciadiagram és kommunikációs diagrammok javítása, 1 use case diagram

6. Szkeleton beadás

6 – ripgyork

Konzulens:
Ádám Zsófia

Csapattagok

Fodor Attila	EUGN1B	afodor998@gmail.com
Fodor Dávid	D02DBR	dfodor999@gmail.com
Földi Balázs	AB8Y3S	fbalu8@gmail.com
Ludányi Barnabás	V5PWP4	ludanyib2003@gmail.com
Mikola Bálint István	TCV0Y9	mikola.balint.istvan@gmail.com (kapcsolattartó)

2024.03.24

6. Szkeleton beadás

6.1 Fordítási és futtatási útmutató

6.1.1 Fájllista

Fájl neve	Méret [byte]	Keletkezés ideje	Tartalom
App.java	228	2024.03.24.	Teszt indítása
Beer.java	2000		Sör osztály
Board.java	2380		Játéktábla osztály
Camambert.java	1010		Camambert osztály
CursedRoom.java	1300		Átkozott szobák osztálya
CycleBased.java	267		CycleBased Interfész
CycleUsage.java	1240		CycleUsage osztály
Door.java	1980		Ajtó osztály
GasProtect.java	422		GasProtect interfész
Item.java	2510		Tárgyak ősosztálya
Logarlec.java	908		Logarléc tárgy osztálya
Mask.java	1540		FPP2 maszk tárgy osztálya
Pairing.java	277		Pairing interfész
PickUp.java	455		PickUp interfész
Player.java	8900		Játékos ősosztály
PutDown.java	462		PutDown interfész
Room.java	6530		Szoba osztály
RoomPairing.java	220		RoomPairing interfész
Student.java	7520		Hallgató osztály
StudentProtection.java	415		StudentProtection interfész
Tablatorlo.java	1360		Táblatörlő tárgy osztálya
Teacher.java	1590		Oktató osztály
Transistor.java	4200		Tranzisztor tárgy osztálya
Tvsz.java	1570		TVSZ tárgy osztálya

6.1.2 Fordítás

A fordításhoz a Logarlec\src\logarlecTheGame mappából kell az alábbi parancsot terminálból kiadni:
javac .\App.java .\Skeleton*.java .\Skeleton\test_felvetel*.java .\Skeleton\test_hallgato*.java .\Skeleton\test_oktato*.java .\Skeleton\test_targyletesz*.java .\Model\Item*.java .\Model\Interfaces*.java .\Model*.java

6.1.3 Futtatás

A kód futtatása fordítás után érhető el. Ehhez a Logarlec\src\ mappából kell a következő parancsot terminálból kiadni:
java -cp . logarlecTheGame.App

6.2 Értékelés

Tag neve	Tag neptun	Munka százalékban
Fodor Attila	EUGN1B	20
Fodor Dávid	D02DBR	20
Földi Balázs	AB8Y3S	20
Ludányi Barnabás	V5PWP4	20
Mikola Bálint István	TCV0Y9	20

6.3 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2024.03.23 18:00	1,5 óra	Fodor A. Fodor D. Földi Ludányi Mikola	Értekezlet: Felosztottuk a kódot 5 részre, a továbbiakban mindenki a saját részein dolgozik
2024.03.24 7:00	12 óra	Fodor A.	Értekezlet alapján megbeszélt kódrészek implementálása.
2024.03.24 7:00	12 óra	Fodor D.	Értekezlet alapján megbeszélt kódrészek implementálása.
2024.03.24 8:00	12 óra	Földi	Értekezlet alapján megbeszélt kódrészek implementálása.
2024.03.24 9:00	12 óra	Mikola	Értekezlet alapján megbeszélt kódrészek implementálása.
2024.03.24 9:00	12 óra	Ludányi	Értekezlet alapján megbeszélt kódrészek implementálása.

7. Prototípus koncepciója

6 – ripgyork

Konzulens:
Ádám Zsófia

Csapattagok

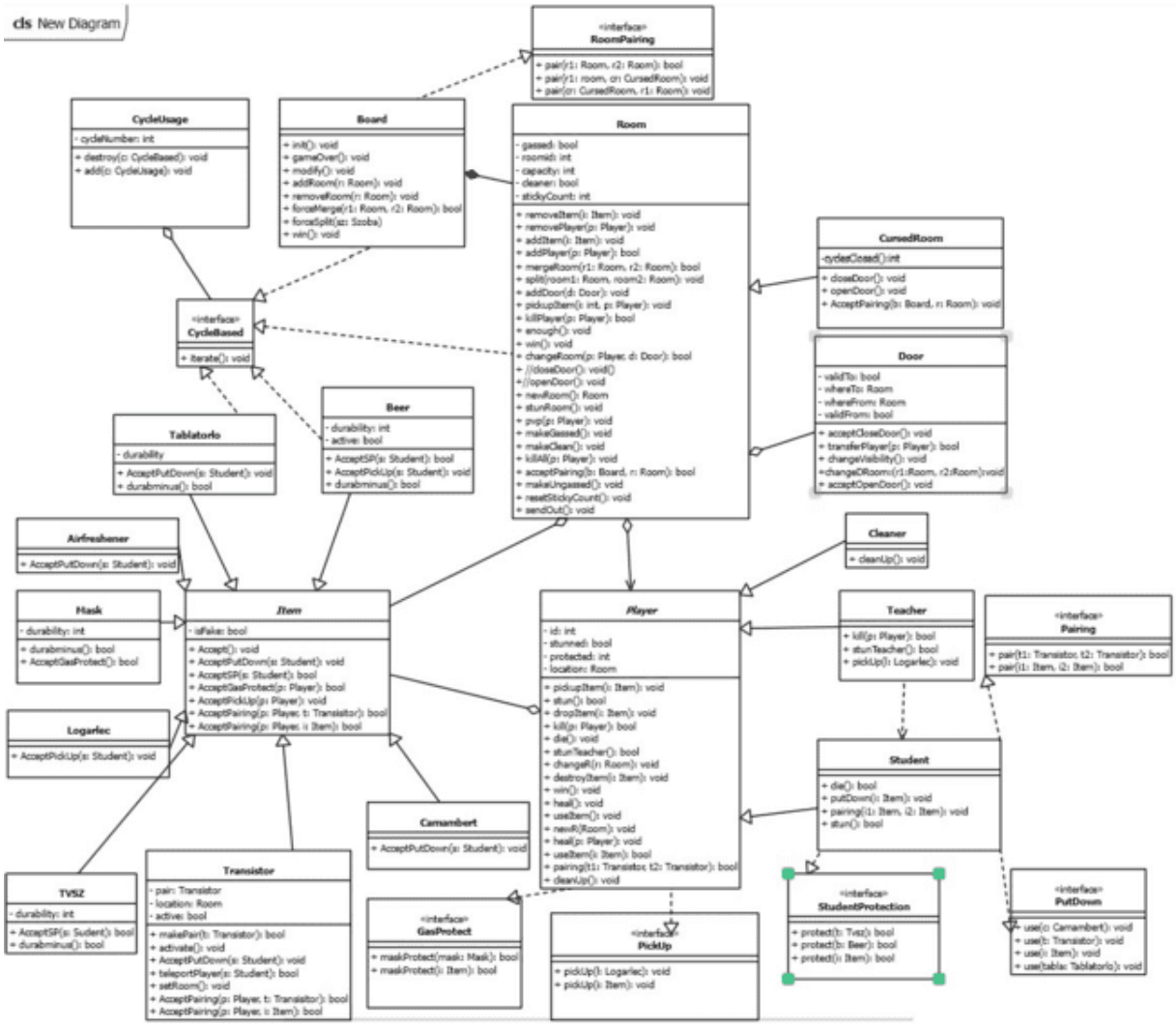
Fodor Attila	EUGN1B	afodor998@gmail.com
Fodor Dávid	D02DBR	dfodor999@gmail.com
Földi Balázs	AB8Y3S	fbalu8@gmail.com
Ludányi Barnabás	V5PWP4	ludanyib2003@gmail.com
Mikola Bálint István	**TCV0Y9**	**mikola.balint.istvan@gmail.com (kapcsolattartó)**

2024.04.08

7. Prototípus koncepciója

7.0 Változás hatása a modellre

Módosult osztálydiagram



Új vagy megváltozó metódusok

Accept(), AcceptPutDown(), AcceptPickUp(), AcceptSP(), AcceptGasProtect() és AcceptPairing: Csak akkor hajtódnak végre ezek a metódusok, ha a tárgy isFake attribútuma hamis.

pickupItem(): (Többek között) Csak akkor engedi a paraméterül kapott hallgatónak felvenni a tárgyat, ha a szoba stickyCount-ja kisebb 3-nál.

makeUngassed(): Room új metódusa, szoba gassed attribútumát false-ra állítja.

resetStickyCount(): Room új metódusa, szoba stickyCountját 0-ra állítja.

sendOut(): Room új metódusa, végigmegy a szoba összes ajtaján és átrakja rajta keresztül a nem stunnolt játékosokat, amíg vannak játékosok a szobában.

changeR(): Mielőtt bármit is csinálna leellenőrzi, hogy a játékos stunnolt-e.

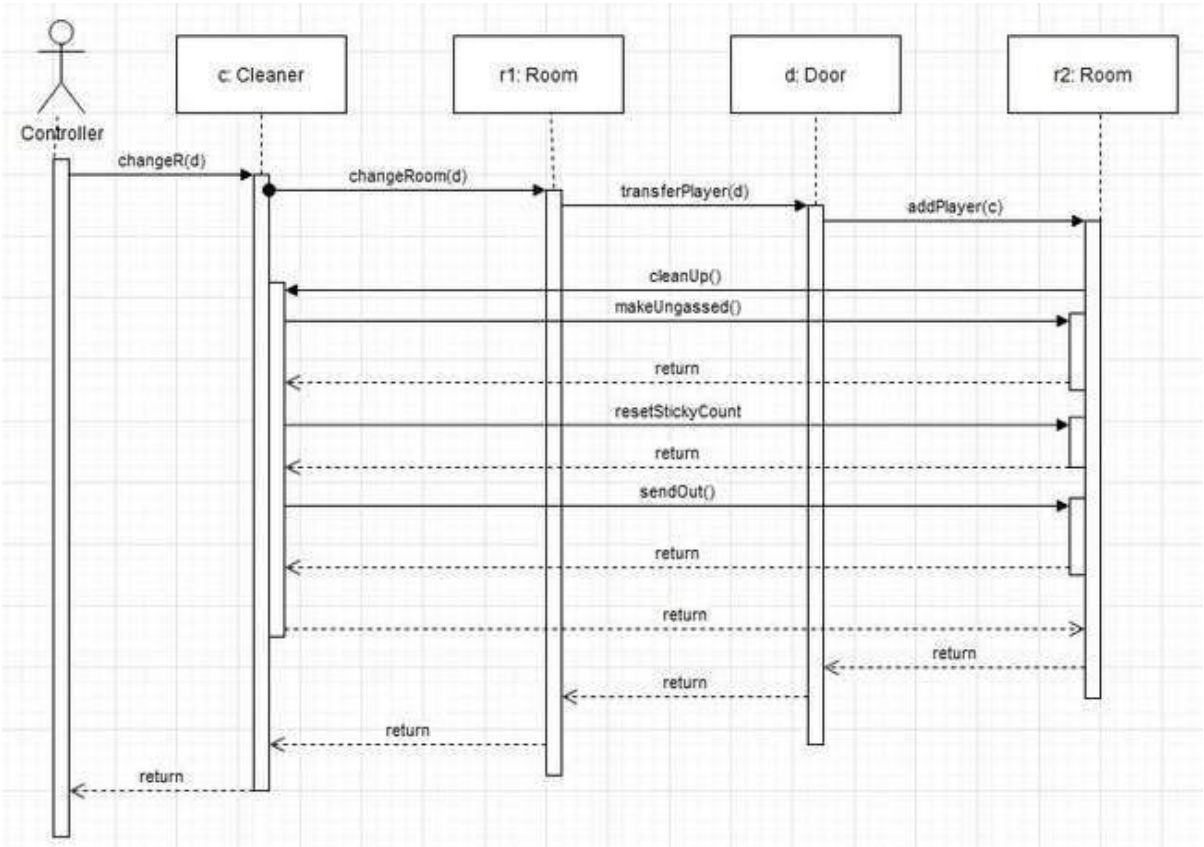
cleanUp(): Player új absztrakt metódusa. Csak a takarító valósítja meg. Meghívja az őt tartalmazó szoba resetStickyCount, sendOut() és makeUngassed() metódusát.

protect(Beer b): Student metódusa megváltozik. Ha megvédi a studentet, akkor véletlenszerűen a student egy tárgyára meghívja a DropItem() metódust.

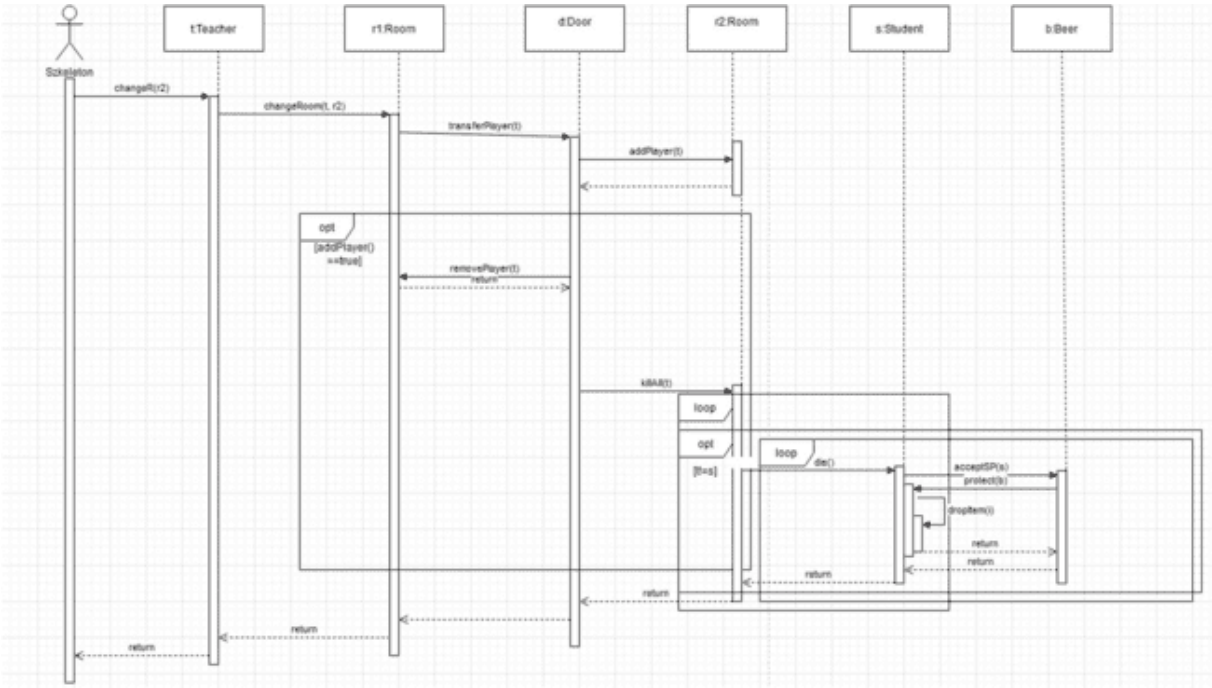
use(Airfreshener a): Stundent (putDown interfész) új metódusa, eldobja a légfrissítőt, majd a legfrissítőt tartalmazó szobának meghívja a makeUngassed metódusát.

Szekvencia-diagramok

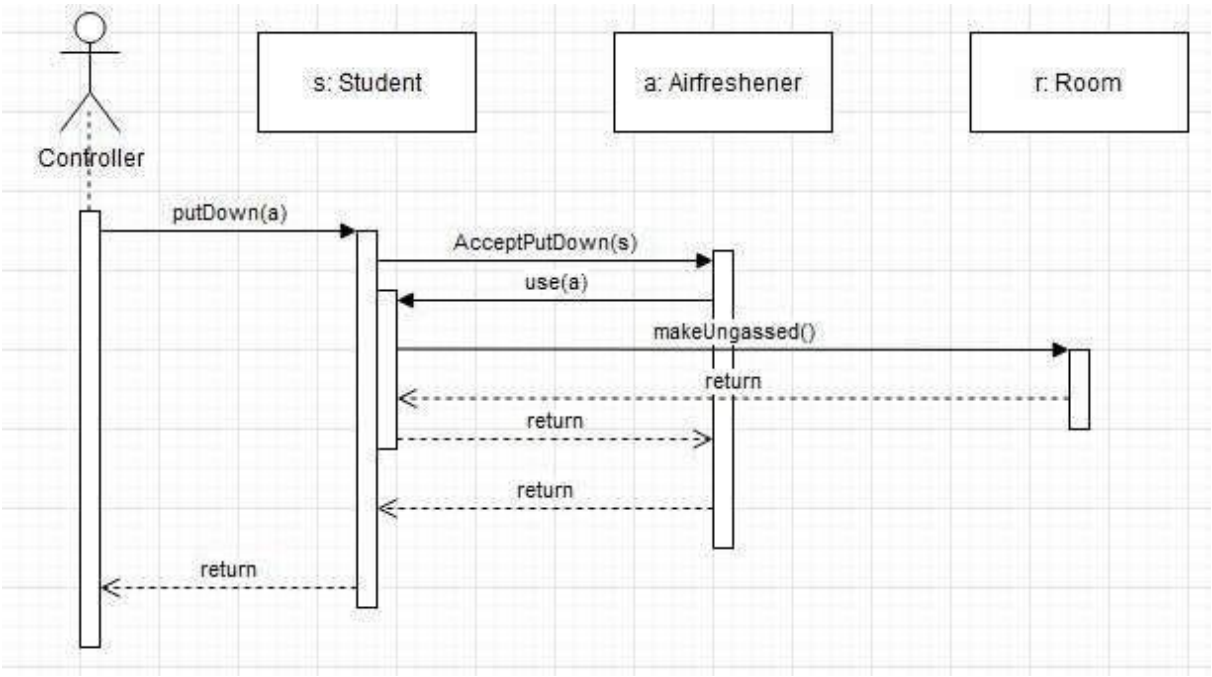
Szekvencia diagram egy takarító lépésére üres szobába (annak az ellenőrzése, hogy eflér-e, az addPlayer-ben történik, ha nem fér el, nem hívódik a cleanUp metódus):



SzekvenciaDiagram sörre:



Szekvenciadiagram légfrissítőre:



7.1 Prototípus interface-definíciója

7.1.1 Az interfész általános leírása

Az interface úgy néz ki, hogy a program konkrét, előre definiált parancsokat vár el, mellyel előidézi az adott eseményeket. Kiválasztható, hogy parancssorból vagy fájlból várja el őket. Minden parancsnak egy kimenete lesz melyet a kimeneti nyelv definiál, ezeket a konzolon jeleníti meg a program, de egy erre a célra kialakított paranccsal fájlba is lehet menteni a játék menetét és állását.

7.1.2 Bemeneti nyelv

Megjegyzés: a board-ok betöltése valójában parancsok beolvasása, ezekkel építve a pályát, ezért ennek nyelvezetét külön nem definiáljuk, a parancsokban látható

loadBoard

Leírás: Betölti a táblát
Opciók: A betöltendő pálya neve (boards mappából)

pickUp

Leírás: Tárgyfelvétel
Opciók: A tárgyak neve (pl. Logarlec, Beer) és aki felveszi neve (Player1)

putDown

Leírás: Tárgy letétele
Opciók: A játékosnál lévő tárgy neve (pl. Beer) és aki leteszi neve (Player1)

useDoor

Leírás: Ajtó használat, szobába lépés
Opciók: Az ajtó neve, amit használni akarunk, paraméterek: <aki használja> <melyik ajtó>

randomOff

Leírás: Véletlenszerűség kikapcsolása
Opciók: -

randomOn

Leírás: Véletlenszerűség bekapcsolása
Opciók: -

teacherRound

Leírás: tanár elvégzi a random műveletét (lépés vagy felvétel)
Opciók: -
Megjegyzés: Csak akkor adható ki, ha a randomOn parancs érvényben van

janitorRound

Leírás: takarító elvégzi a random műveletét (csak lépés és egyben takarítás)
Opciók: -
Megjegyzés: Csak akkor adható ki, ha a randomOn parancs érvényben van

merge

Leírás: Mergelés
Opciók: Room1, Room2 paraméterekkel (szobák amit mergelni akarunk)
Megjegyzés: Csak akkor adható ki, ha a randomOff parancs érvényben van

split

Leírás: Splitelés
Opciók: Room1 paraméterrel (szoba amit splitelni akarunk)
Megjegyzés: Csak akkor adható ki, ha a randomOff parancs érvényben van

iterateAll

Leírás: minden CycleBased iterate függvénye
Opciók: -
Megjegyzés: Csak akkor adható ki, ha a randomOn parancs hatása érvényben van

beerIterate

Leírás: sör iterálása
Opciók: sör neve
Megjegyzés: Csak akkor adható ki, ha a randomOff parancs érvényben van

maskIterate

Leírás: maszk iterálása
Opciók: maszk neve
Megjegyzés: Csak akkor adható ki, ha a randomOff parancs érvényben van

pair

Leírás: Párosítás
Opciók: Párosítást kezdeményező játékos, két párosítandó tárgy a paramétere (Item1, Item2)

listPlayers

Leírás: Összes játékos kilistázása
Opciók:

listRoom

Leírás: Játékosok, tárgyak kilistázása egy szobában
Opciók: Egy szobát kap (Room1)

listRooms

Leírás: Szobák kilistázása
Opciók: -

listItem

Leírás: Tárgyak kilistázása a pályán, ami nem játékosoknál van
Opciók: -

listPlayerItem

Leírás: Kilistázza egy játékosnál a tárgyakat
Opciók: Egy játékost kap (Player1)

save

Leírás: Játékállás elmentése
Opciók: -

gasRoom

Leírás: Szoba gázosítása
Opciók: Egy szobát kap, amelyik gázos lesz (Room1)

cleanRoom

Leírás: Szoba táblatörlőssé tétele
Opciók: Egy szobát kap, amelyik táblatörlős lesz (Room1)

takaritottRoom
Leírás: Szoba takarítottá tétele
Opciók: Egy szobát kap, amelyik takarított lesz (Room1)

stickyRoom
Leírás: Szoba ragacsossá tétele
Opciók: Egy szobát kap, amelyik takarított lesz (Room1)

closeDoor
Leírás: Ajtók bezárása
Opciók: Egy szobát kap (Room1), aminek az ajtajait bezárja

openDoor
Leírás: Ajtók kinyitása
Opciók: Egy szobát kap (Room1), aminek az ajtajait kinyitja

addDoor
Leírás: Ajtók hozzáadás
Opciók: Két szobát kap (Room1, Room2) és két boolt az irányok miatt, név

addRoom
Leírás: Szoba hozzáadása
Opciók: Típusa, neve

addPlayerToRoom
Leírás: Játékos szobához adása (új)
Opciók: Játékos típusa, neve, szoba, ahova kerül

addItemToRoom
Leírás: Tárgy szobához adása (új)
Opciók: Tárgy típusa, neve, szoba, ahova kerül

7.1.3 Kimeneti nyelv

loadBoard
Eredmény: Sikeres vagy sikertelen
Megjelenítés: “Pálya betöltés sikeres”/”Pálya betöltés sikertelen”

pickUp
Eredmény: tárgy felvétele
Megjelenítés: “<tárgy neve>: <felvevő játékos neve> <sikeresség>

putDown
Eredmény: letétel/használat helye

Megjelenítés: “<tárgy neve>: <Szoba neve>, <használó játékos neve >, “”putDown””

useDoor
Eredmény: ajtó használat sikeressége
Megjelenítés: “<ajtó neve>: <játékos neve>, sikeresség

randomOff/randomOn
Eredmény: random működés be-/kikapcsolása
Megjelenítés: “random [on/off]”

teacherRound
Eredmény: tanár által random végzett művelet
Megjelenítés: kiírás

janitorRound
Eredmény: takarító által végzett művelet
Megjelenítés: kiírás

merge
Eredmény: merge sikeressége
Megjelenítés: “<szoba1> <szoba2> sikeres/sikertelen merge”

split
Eredmény: split eredménye
Megjelenítés: “<szoba1> split, uj szoba: <szoba2>

iterateAll
Eredmény: körfüggő tárgyak iterációja
Megjelenítés: kiírás

beerIterate
Eredmény: körfüggő sör iterációja
Megjelenítés: “<sör neve> iterate”

maskIterate
Eredmény: körfüggő maszk iterációja
Megjelenítés: “<maszk neve> iterate”

pair
Eredmény: sikeres/sikertelen
Megjelenítés: “Sikeres/sikertelen párosítás: <tárgy1 neve>:<tárgy2 neve>

listPlayers
Eredmény: játékosok listája
Megjelenítés: “<Játékos neve> <Szoba>

listRooms
Eredmény: szobák listája
Megjelenítés: “<szoba neve>

listRoom
Eredmény: adott szobába található elemek
Megjelenítés: “<Szoba neve>: <tárgy 1> <tárgy 2>.../n
<játékos1 neve> <játékos2 neve>

listPlayerItem
Eredmény: adott játékosnál lévő tárgyak
Megjelenítés: “<játékos neve>: <tárgy 1> <tárgy 2>...

listItem
Eredmény: tárgyak kilistázása a pályán
Megjelenítés: “<Szoba1 neve>: <tárgy 1> <tárgy 2>.../n
<Szoba2 neve>: <tárgy 1> <tárgy 2>.../n
...

7.2 Összes részletes use-case

Use-case neve	Iterálás
Rövid leírás	A Cyclebased dolgokat lépteti
Aktorok	Játékos
Forgatókönyv	A cycle hatására bizonyos objektumok különböző lépéseket hajthatnak végre, vagy tárgyak esetében hatásuk vagy tartósságuk csökkenhet

Use-case neve	Merge
Rövid leírás	Merge kérés érkezik 2 szobára, melyek közt van/nincs gázos, a játékosok száma a két szobában együttvéve pedig kevesebb, mint a nagyobb kapacitású szoba kapacitása
Aktorok	Játékos
Forgatókönyv	A merge sikeresen végrehajtódik, a kisebb kapacitású szobából minden átkerül a nagyobb kapacitásúba, valamint a kisebb szoba ajtajait is a nagyobb szoba kapja, végül a kis szoba törlődik

Use-case neve	Elgázosodás
Rövid leírás	Egy hallgató lehelyezi a camembert, a szoba ezzel gázossá válik.
Aktorok	Játékos
Forgatókönyv	A szoba gázosságát jelző paraméter true-ra állítódik, valamint a stun metódus minden bent tartózkodóra meghívódik. Amennyiben a szoba eredetileg is gázos volt, nincs változás.

Use-case neve	Csúszóssá válás
Rövid leírás	Egy hallgató lehelyezi a táblatörlőt, mire a szoba csúszóssá válik.
Aktorok	Játékos

Forgatókönyv	A szoba csúszósságát jelző paraméter true-ra állítódik, valamint stunTeacher() metódus minden bent tartózkodóra meghívódik (csak tanárokra van hatással). Amennyiben a szoba eredetileg is csúszós volt, nincs változás.

Use-case neve	Ragacsossá válás
Rövid leírás	Egy idő után ha nem takarítják és elég játékos járt benn, a szoba ragacsossá válik
Aktorok	Játékos
Forgatókönyv	A szoba ragacsosságot jelző paraméter true-ra állítódik.

Use-case neve	Tisztává válás
Rövid leírás	Takarítás után a szoba tisztává válik.
Aktorok	Játékos
Forgatókönyv	A szoba gázosságot jelző paraméter false-ra állítódik.

Use-case neve	Split
Rövid leírás	A kiválasztott szoba kettéosztódik.
Aktorok	Játékos
Forgatókönyv	Létrejön egy új szoba, amely kizárólag a split metódust végrehajtó eredeti szobával van összeköttetésben, valamint minden egyéb attribútuma (kapacitás, gázos-e, csúszós-e) megegyezik az eredeti szobával. Az eredeti szobában tartózkodó játékosok és tárgyak az eredeti szobában maradnak.

Use-case neve	Hallgató lép
Rövid leírás	A hallgató átlép (ha tud) egy szobából egy másikba
Aktorok	Játékos
Forgatókönyv	Az átlépés vagy sikeres és a hallgató átkerül az új szobába, vagy nem sikeres, mert például a szoba tele van, ilyenkor marad a helyén.

Use-case neve	Oktató lép
Rövid leírás	Az oktató átlép (ha tud) egy szobából egy másikba
Aktorok	Játékos
Forgatókönyv	Az átlépés vagy sikeres és az oktató átkerül az új szobába, vagy nem sikeres, mert például a szoba tele van, ilyenkor marad a helyén.

Use-case neve	Takarító lép
Rövid leírás	A takarító átlép (ha tud) egy szobából egy másikba
Aktorok	Játékos
Forgatókönyv	Az átlépés vagy sikeres és a takarító átkerül az új szobába, vagy nem sikeres, mert például a szoba tele van, ilyenkor marad a helyén.

Use-case neve	Tárgy letétel
Rövid leírás	Diák letesz egy tárgyat, ami nála van, bizonyos tárgyaknál ennek utána hatása van, de bizonyos tárgyak egyszerűen csak kikerülnek az inventoryból.
Aktorok	Játékos
Forgatókönyv	Tárgytól függ, lehet gázosítás, csúszóssá tétel stb, vagy nem történik semi.

Use-case neve	Transistor párosítás
Rövid leírás	Két különböző, nem aktív tranzisztort párosít, mikor már nála vannak
Aktorok	Játékos
Forgatókönyv	Feltételezzük, hogy egy diáknál már van két tranzisztor, őket párosítja magánál.

Use-case neve	Transitor használat
Rövid leírás	Két különböző szobában letett és aktív tranzisztort használ a diák
Aktorok	Játékos
Forgatókönyv	Létrejön két szoba, egyikben egy diák. Feltételezük, hogy az egyik szobában le van téve egy Tranzisztor. A másik szobában van a diák és nála a Transistor pár. Leteszi és ezzel átkerül a másik Transistor szobájába

Use-case neve	Tárgyfelvétel
Rövid leírás	Diák felvesz egy tárgyat a szobából
Aktorok	Játékos
Forgatókönyv	A diák felveszi a tárgyat, ami elfér nála. Ennek következménye lehet akár a játék megnyerése is, ha a tárgy a logarléc.

Use-case neve	Tárgy hozzáadása
Rövid leírás	Hozzáadjuk a tárgyat a szobához vagy egy játékoshoz
Aktorok	Játékos
Forgatókönyv	A szobában vagy a diak inventoryában megjelenik a tárgy.

Use-case neve	Szoba hozzáadása
Rövid leírás	Hozzáadjuk a szobát a pályához.
Aktorok	Játékos
Forgatókönyv	A szoba megjelenik a pályán.

Use-case neve	Játékos hozzáadása
Rövid leírás	Hozzáadjuk a Játékost valamelyik szobához.
Aktorok	Játékos
Forgatókönyv	A játékos megjelenik az adott szobában.

Use-case neve	Ajtó hozzáadása

Rövid leírás	Hozzáadjuk az ajtót valamelyik két szobához, valamint hogy egy vagy 2 irányú.
Aktorok	Játékos
Forgatókönyv	Az ajtók megjelennek az adott szobákban.

7.3 Tesztelési terv

Teszt-eset neve	Pálya betöltése
Rövid leírás/ Forgatókönyv	A rendszer betölti a pályát. Beállít minden paramétert. Betölti a játékosokat.
Teszt célja	A pálya helyes betöltésének ell.

Teszt-eset neve	Tárgy felvétele
Rövid leírás/ Forgatókönyv	A szobában van egy tárgy, ami nem a logarléc, illetve egy játékos. A játékos felveszi a tárgyat. Megvizsgáljuk, hogy a tágy a játékos eszköztárában megtalálható-e.
Teszt célja	Tárgyak felvétele.

Teszt-eset neve	Diák felvesz Logarlecet
Rövid leírás/ Forgatókönyv	A szobában van egy diák és egy logarléc. A diák felveszi a logarlecet, ezzel a játék véget ér.
Teszt célja	Játék megnyerése

Teszt-eset neve	Tárgy letétele
Rövid leírás/ Forgatókönyv	Diák letesz egy tárgyat, ami nála van, de nem tud használódni, így csak letevődik
Teszt célja	Tárgyak letétele

Teszt-eset neve	Camambert használata
Rövid leírás/ Forgatókönyv	A játékos leteszi a tárgyat, amitől a szoba gázossá válik
Teszt célja	Camambert funkcionalitása, szoba elgázosítása

Teszt-eset neve	Táblatörlő használata
Rövid leírás/ Forgatókönyv	A játékos leteszi a tárgyat, amitől a szoba csúszós lesz
Teszt célja	Táblatörlő funkcionalitása, szoba csúszósítása

Teszt-eset neve	Légfrissítő használata
Rövid leírás/ Forgatókönyv	A játékos leteszi a tárgyat, amitől a szoba tisztává válik, nem lesz gázos továbbiakban
Teszt célja	Légfrissítő funkcionalitása, a szoba megtisztítása

Teszt-eset neve	Sikeres merge
Rövid leírás/ Forgatókönyv	Merge kérés érkezik 2 szobára, melyek közt van/nincs gázos, a játékosok száma a két szobában együttvéve pedig kisebb vagy egyenlő, mint a nagyobb kapacitású szoba kapacitása. A merge sikeresen végrehajtódik, a kisebb kapacitású szobából minden átkerül a nagyobb kapacitásúba, valamint a kisebb szoba ajtajait is a nagyobb szoba kapja, végül a kis szoba törlődik.
Teszt célja	Merge tesztelése

Teszt-eset neve	Sikertelen merge
Rövid leírás/ Forgatókönyv	Merge kérés érkezik 2 szobára, melyek közt van/nincs gázos, a játékosok száma a két szobában együttvéve pedig több, mint a nagyobb kapacitású szoba kapacitása. Nem történik semmilyen strukturális változás.
Teszt célja	Merge tesztelése

Teszt-eset neve	Split
Rövid leírás/ Forgatókönyv	Létrejön egy új szoba, amely kizárólag a split metódust végrehajtó eredeti szobával van összeköttetésben, valamint minden egyéb attribútuma (kapacitás, gázos-e, csúszós-e) megegyezik az eredeti szobával. Az eredeti szobában tartózkodó játékosok és tárgyak az eredeti szobában maradnak.
Teszt célja	Split tesztelése

Teszt-eset neve	Játékos lép teli szobába
Rövid leírás/ Forgatókönyv	A hallgató olyan szobába lép, ahol nem fér el. Két szoba van, amik közül az egyik, amelyikben nincs senki, annak a kapacitása 0. A játékos ide próbál meg belépni
Teszt célja	Hallgató lépése

Teszt-eset neve	Játékoslép gázos szobába
Rövid leírás/ Forgatókönyv	A hallgató egy gázos szobába lép, nincs maszkja. A gáz minden hatása érvényesül rá.
Teszt célja	Hallgató lépése

Teszt-eset neve	Oktató vs. Hallgató
Rövid leírás/ Forgatókönyv	Egy védtelen hallgató olyan szobába lép, ahol van egy oktató. Az oktató megöli a játékost.
Teszt célja	Hallgató és Oktató találkozása

Teszt-eset neve	Oktató nedves táblatörlős szobába lép
Rövid leírás/ Forgatókönyv	Oktató egy olyan szobába lép, ahol van aktív nedves táblatörlő. Az oktató elkábul és eldobja az összes nála lévő tárgyat.
Teszt célja	Nedves táblatörlő és oktató

Teszt-eset neve	Transistor használata
Rövid leírás/ Forgatókönyv	Feltételezzük, hogy egy diáknál már van két tranzisztor, amik párosítva vannak Egyik szobába lerakja az egyiket, másikba a másikat. Mikor leteszi a másodikat, átkerül a másik Transistor szobájába.
Teszt célja	Transistor helyes működése

Teszt-eset neve	Takarító lép gázos szobába, ahol van kábult játékos.
Rövid leírás/ Forgatókönyv	A takker belép egy olyan szobába, ahol 2 játékos van, az egyik kábult. A takker kitessékeli a nem bénult játékost, majd kiszellőztet, ezzel megszűnik a szoba gázossága és a szoba nem lesz többé ragacsos, vagyis lecsökkenti az alapra az értékét.
Teszt célja	Takarító helyes viselkedése

7.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása

Megadhatunk parancssorban vagy fájlban különböző teszteseteket, majd az eredményt is, vagy olvashatjuk a szabványos kimeneten, vagy fájlba is írhatjuk ki. Ezek felhasználásával a felhasználó tudja összesíteni a kapott és az elvárt eredményeket.
Alapértelmezett Windows parancssori parancsokkal összehasonlítható lesz a két eredmény (pl. diff/FC)

Napló

Kezdet	**Időtartam**	**Résztvevők**	**Leírás**
2024.04.05. 14:30	45 perc	Fodor A. Fodor D. Földi Ludányi Mikola	Értekezlet. Felosztottuk a feladat fejezeteit.
2024.04.07. 14:00	1 óra	Fodor A. Fodor D. Földi Ludányi	Kérdések megbeszélése, egyes fejezetek összeegyeztetése
2024.04.07. 15:00	3 óra	Fodor A. Fodor D.	7.2-es és 7.5-ös részek kidolgozása, mások munkájának átnézése
2024.04.07 15:30	1,5 óra	Földi	Tesztelési terv
2024.04.07 16:00	2,5 óra	Ludányi	Use-casek

2024.04.07 18:00	2,5 óra	Mikola	Változtatások hatása a modellre rész megírása, diagrammok szerkesztése/javítása

8. Részletes tervek

6 – ripgyork

Konzulens:
Ádám Zsófia

Csapattagok

Fodor Attila	EUGN1B	afodor998@gmail.com
Fodor Dávid	D02DBR	dfodor999@gmail.com
Földi Balázs	AB8Y3S	fbalu8@gmail.com
Ludányi Barnabás	V5PWP4	ludanyib2003@gmail.com
Mikola Bálint István	TCV0Y9	mikola.balint.istvan@gmail.com (kapcsolattartó)

2024.04.14

8. Részletes tervek

8.1 Osztályok és metódusok tervei.

8.1.1 Airfreshener

- **Felelősség**

A légfrissítőt reprezentáló osztály, mely az Item osztályból származik, hisz ő maga is egy tárgy, szoba tisztításáért (gáz eltüntetése) felel

- **Ősosztály**
  - Item
- **Metódusok**
  - +acceptPutDown(s: Student): void – A tárgy letételére szolgál, mely meghívja a use(Airfreshener a) függvényét a játékosnak (this paraméterrel) ami majd kitisztítja a gázos szobát

8.1.2 Beer

- **Felelősség**

A sört reprezentáló osztály, mely leszármazottja az Item osztálynak, hiszen ő maga is egy tárgy

- **Ősosztály**
  - Item
- **Interfészek**
  - CycleBased
- **Attribútumok**
  - -durability: int - Megszünéséig hátralévő körök száma
  - -isFake: bool: Ez a tulajdonság szolgál arra, hogy maga a tárgy, vagyis a sör igazi-e, vagy hamis, szóval haszontalan. Ha true, akkor hamis, ha false, akkor igazi a tárgy.
  - -active: bool: Azt jelzi, hogy használatban van-e éppen a Sör
- **Metódusok**
  - +acceptPickUp(s: Student): void - Játékoshoz kerül, vagyis elfogadja a felvételt, így bekerülhet a játékos inventoryjába. Meghívja a játékos pickUp függvényét.
  - +acceptSP(s: Student): bool- A hallgató védelmére szolgál, ha sikerült megvédeni, igazzal tér vissza. Ezzel együtt a játékos el is ejt egy random tárgyat.
  - +durabMinus(): bool - Csökkenti a durability-t eggyel, ha nem sikerül (már nem lehet kisebb) hamissal tér vissza.

8.1.3 Board

Felelősség

Ez az osztály felelős a szobák és a játéktér reprezentálásáért. Nyilvántartja a szobákat és azokat összevonhatja és szétszedheti. Felépíti a pályát, módosítja és karbantartja.

Attribútumok

- RoomList: ArrayList<Room>

Interfészek

- CycleBased
- RoomPairing

Metódusok

- +Init(): void - Inicializálja pályát

- +GameOver(): void - Befejezi a játékmenetet, ha jelet kap, hogy valami miatt vége van
- +addRoom(r: Room): void - Új szobát ad a játéktérhez
- +removeRoom(r: Room): void - Törli a megadott szobát a játéktérből
- +forceMerge(r1: Room, r2: Room): bool – Egybeolvasztja r1 és r2 szobákat, ha sikerült, igazzal tér vissza, meghívja a szobák Merge függvényét
- +forceSplit(r: Room): void – Jelzi az 'r' szobának, hogy osztódni kell és meghívja a newR függvényét
- +win(): void – jelzi, ha a játék nyereséggel ért véget, mehhívja a GameOver() függvényét

8.1.4 Camambert

- **Felelősség**

A camambert sajtot reprezentáló osztály, mely az Item osztályból származik, hisz ő maga is egy tárgy, szoba gázosításért felel

- **Ősosztály**
  - Item
- **Metódusok**
  - +acceptPutDown(s: Student): void – A tárgy letételére szolgál, mely meghívja a use(Camambert c) függvényét a játékosnak (this paraméterrel) és az majd elgázosítja a szobát

8.1.5 CursedRoom

- **Felelősség**

Az osztály az átkozott szobák reprezentálásért felel.

- **Interfészek**

  CycleBased: mert a szobák alakulása is függ az adott körtől (iterációtól)

- **Ősosztály**
  - Room
- **Metódusok**
  - +closeDoor(): void - Bezár minden ajtót a szobában (minden ajtóra meghívja az acceptCloseDoor függvényt) amikor meghívódik
  - +openDoor(): void – Kinyit minden ajtót a szobában (minden ajtóra meghívja az acceptOpenDoor függvényt) amikor meghívódik
  - +AcceptPairing(Board b, Room r1): void – nem engedi a párosítást

8.1.6 CycleBased Interfész

- **Felelősség**

Az interfész a körönként változó objektumokért felelős, ezeken a körönkénti események jelzéséért (kör léptetése) felel

- **Metódusok**
  - +iterate(): void - lépteti a köröket, melyet minden őt megvalósító osztály felüldefiniál

8.1.7 CycleUsage

- **Felelősség**
  - Az osztály felelős azon osztályok helyes működéséért, amiket valamilyen módon befolyásol a ciklusok, vagy körök múlása
- **Attribútumok**
  - -cycleNumber: int – Éppen hanyadik ciklus száma, ezzel számolódik
  - -cycleList: ArrayList<Cyclebased>: CycleBased interfészt megvalósító objektumokat tárol
- **Metódusok**
  - +add(c: cycleUsage): void - interfész tárolás miatt kell hozzáadás, hozzáadja a listához

8.1.8 Door

- **Felelősség**
  - Az ajtókat reprezentáló osztály. Egy ajtó nyilvántartja a szobákat, ahová rajta keresztül el lehet jutni.
- **Attribútumok:**
  - -whereTo: Room –Egyik szoba, ahol van az ajtó
  - -whereFrom: Room - Másik szoba, ahol van az ajtó
  - -validTo: bool - Járható-e From-ból To irányba az ajtó, ha true akkor igen, ha false akkor nem
  - -validFrom: bool - Járható-e To-ból Rrom irányba az ajtó, ha true akkor igen, ha false akkor nem

- **Metódusok**
  - +acceptCloseDoor(): void - elfogadja a bezárást, és bezárja vagyis átállítja a bool attribútumait false-ra
  - +transferPlayer(p: Player): bool - Átjuttat egy játékost a másik szobába önmagán keresztül, ha nincs zárva és abba az irányba lehet menni, akkor meghívja a cél szoba addPlayer függvényét, ha az igazzal tér vissza meghívja a saját removePlayer függvényét és a kilAll és pvp függvényeket p paraméterrel
    - Pszeudokód az algoritmushoz:

```
Ha nem closed és whereTo nem false
    Ha whereTo addplayer sikeres
        WhereFrom removePlayer
        WhereTo  killAll
        WhereTo pvp
        Visszatér igaz
    Egyébként visszatér hamis
```

  - +changeDRoom(r1: Room, r2: Room): void – Az ajtó r1 oldali szobáját lecseréli r2-re
  - +acceptOpenDoor():void – elfogadja az ajtó kinyitását és átállítja a bool attribútumait true-ra

8.1.9 GasProtect Interfész

- **Felelősség**

A gáz elleni automatikus védelemben játszik szerepet

- **Metódusok**
  - +maskProtect(m: Mask): bool – ha még aktív a tárgy (tehát sikeres a védelem) akkor csökkenti a tárgy durability-jét és igazzal tér vissza
  - +maskProtect(i: Item): bool – hamissal tér vissza, hiszen csak a maszk képes védelemre

8.1.10 Item

- **Felelősség:**

A tárgyakat reprezentáló osztályok absztrakt ősosztálya.

- **Attribútumok**
  - isFake: bool – a tárgyak tulajdonsága, hogy igazi-e vagy nem, true, ha nem igazi és false, ha igazi
- **Metódusok**
  - +acceptPutDown(s: Student): void – A tárgy letételére szolgál, ezt minden leszármazott megvalósítja, akinek szüksége van rá
  - +acceptSP(s: Student): bool - A hallgató védelmére szolgál, ha sikerül, akkor igazzal tér vissza, ezt minden leszármazott megvalósítja, akinek szüksége van rá
  - +acceptGasProtect(p: Player): bool - A játékosok gáz elenni védelmére szolgál, ha sikeres a védés, igazzal tér vissza, ezt minden leszármazott megvalósítja, akinek szüksége van rá
  - +acceptPickUp(p: Player): void - A tárgy felvételére szolgál, ezt minden leszármazott megvalósítja, akinek szüksége van rá
  - +acceptPairing(p: Player, t: Transistor): bool – Tranzisztorok párosítását valósítja meg, ha ez sikerült, igazzal tér vissza, ezt minden leszármazott megvalósítja, akinek szüksége van rá
  - +acceptPairing(p: Player,i: Item): bool – Tárgyak párosítását valósítja meg, de ez nem lehetséges, így csak jelez, hogy nem lehetséges, ezt minden leszármazott megvalósítja, akinek szüksége van rá

8.1.11 Janitor

- **Felelősség:**

A takarítókat reprezentáló osztály, akik Player absztrakt ős leszármazottjai, hisz ők is játékosok

- **Ősosztály**:
  - Player
- **Metódusok:**
  - +kill(p: Player): bool – hamissal tér vissza hiszen nem tud ölni
  - +stunTeacher(): bool- hamissal tér vissza, mert rá nem hat a táblatörlő
  - +pickUp(Item i): void – csak visszatér, mert nem tud felvenni tárgyat
  - +cleanRoom(): void - meghívja a szobája makeUngassed függvényét, és a sendOut() függvényét
  - + die(): void – csak visszatér, mert nem tud meghalni
  - + maskProtect(Mask m): bool – hamissal tér vissza hiszen őt nem védheti
  - + maskProtect(Item i): bool – hamissal tér vissza
  - +pickUpItem(i: Item): void– csak visszatér, mert nem tud felvenni tárgyat
  - +pickUp(Logarlec i): void - csak visszatér, mert nem tud felvenni tárgyat
  - +pickUp(Beer i): void - csak visszatér, mert nem tud felvenni tárgyat

- +stun(): bool – hamissal tér vissza, mert rá nem hat a gáz

8.1.12 Logarlec

- **Felelősség**

A Logarlécet reprezentáló osztály. Játék közben felvételével nyernek a hallgatók, oktató vagy takarító nem veheti fel.

- **Ősosztály**
  - Item
- **Attribútumok**
  - -isFake: bool – Ez a tulajdonság szolgál arra, hogy maga a tárgy, vagyis a logarléc igazi-e, vagy hamis, szóval haszontalan. Ha true, akkor hamis, ha false, akkor igazi a tárgy
- **Metódusok**
  - +acceptPickUp(s: Student): void - Tárgy felvétele, meghívja a paraméterül kapott diák pickUp(Logarlec l) függvényét (this paraméterrel), mellyel meg is nyeri a játékot, ha igazi, ha nem, nem történik semmi

8.1.13 Mask

- **Felelősség**

Az FFP2 maszkot reprezentáló osztály, gázos szoba ellen védi meg a hallgatókat vagy a tanárokat

- **Ősosztály**
  - Item
- **Attribútumok**
  - -durability: int - még használható körök száma, csökken minden használattal és körrel is
  - -isFake: bool – Ez a tulajdonság szolgál arra, hogy maga a tárgy, vagyis a maszk igazi-e, vagy hamis, szóval haszontalan. Ha true, akkor hamis, ha false, akkor igazi a tárgy.
- **Metódusok**
  - +acceptPickUp(s: Student): void - Tárgy felvétele
  - +durabMinus(): bool - Csökkenti a durability-t eggyel, ha nem sikerül (elfogy) hamissal tér vissza és innentől kezdve fake tárgy lesz

8.1.14 Pairing

- **Felelősség**

Itemek párosításáért felel. Metódusai igazzal térnek vissza, ha ez sikeres, hamissal, ha nem.

- **Metódusok**
  - +pairing(t1: Transistor, t2: Transistor): bool - párosít két paraméterként kapott transistort, true ha sikeres, false ha hamis
  - +pairing(i1: Item, i2: Item): bool - két paraméterként kapott Itemet párosít

8.1.15 PickUp Interfész

- **Felelősség**

Tárgyak felvételében játszik szerepet, jelez a megfelelő osztály felé az állapotváltozásról

- **Metódusok**
  - +pickUp(l: Logarlec): void - Logarléc felvétele, speciális esemény, felvételével a diákok nyernek, más nem is veheti fel

- +pickUp(b: Beer): void - Sör felvétele, ezzel aktiválódik a tárgy, innentől kezdve iterálható
- +pickUp(i: Item): void – Sima tárgy felvétele

8.1.16 Player

- **Felelősség**

A játékosokat (hallgató, oktató, takarító) reprezentáló osztályok absztrakt őse.

- **Interfészek**
  - PickUp: tárgyak felvételére
  - GasProtect: gáz elleni védelemre
  - Pairing: tranzisztor párosításra
- **Attribútumok:**
  - - id: int - A játékost egyértelműen azonosító szám
  - - stunned: bool - Kábítva van-e a játékos, ha igen akkor igaz értékű. Vonatkozik mind a gáz miatti, mind a táblatörlő miatti kábulásra. Ha igaz, akkor nem tud se mozogni, se tárgyat felvenni
  - - location: Room – Az a szoba, ahol a játékos van
  - - itemList: Arraylist<Item> - játékosnál lévő tárgyak listája

- **Metódusok:**
  - +pickUpItem(i: Item): void– Tárgy felvételére szolgál,akkor tud felvenni tárgyat, ha kevesebb mint 5 tárgy van nála. Meghívja a tárgy acceptPickUp metódusát
  - +pickUp(Logarlec i): void - interfész függvénye, Logarléc felvételét végzi, leszármazott osztályok felüldefiniálják nekik megfelelően
  - +pickUp(Beer i): void - interfész függvénye, Logarléc felvételét végzi, leszármazott osztályok felüldefiniálják nekik megfelelően, felvételkor aktiválódik, innentől iterálható
  - +stun(): bool - Játékos kábítása, ha sikerült igazzal tér vissza és a stunned attribútumát is beállítja true-ra majd a játékos összes tárgyát eldobja, leszármazottak nekik megfelelően felüldefiniálják
  - +removeItem(i: Item): void – i tárgy eltávolítása a játékos itemList-jéből
  - +dropItem(Item i): void – elejti egy tárgyát, ami a location-je itemList-jébe kerül bele (addItem függvénnyel)
  - +kill(p: Player): bool – p játékos megölése, leszármazottak nekik megfelelően felüldefiniálják, igazzal tér vissza, ha sikerült
  - +stunTeacher(): bool - Oktató kábítása, ha sikerül igazzal tér vissza, leszármazottak nekik megfelelően felüldefiniálják
  - +changeR(d:Door): bool– Szoba váltás kezdeményezése a paraméterül kapott ajtón át, meghívja a location-je changeRoom függvényét, igazzal tér vissza, ha sikeres volt
  - +win(): void - játék megnyerését jelző függvény, leszármazottak nekik megfelelően felüldefiniálják
  - +heal(p: Player): void – Stun megszüntetése (bool attribútum false-ra állítása)
  - +setRoom(Room r): void – a location beállítását végzi inicializálásnál és szoba váltásnál
  - + maskProtect(Mask m): bool - interfész függvénye. Ha képes még csökkenteni a durability-n (a durabminus függvénnyel) akkor igazzal tér vissza
  - + maskProtect(Item i): bool - interfész függvénye. Mindig hamissal tér vissza, mert ilyen típusú védelemre csak a maszk képes

- +cleanRoom(): void – adott szoba (location) kitakarítására szolgál, leszármazottak nekik megfelelően felüldefiniálják

8.1.17 PutDown Interfész

- **Felelősség**

Tárgyak lehelyezésében, aktiválásában játszik szerepet. Amennyiben sikeres a tárgy használata a függvényei igazzal térnek vissza.

- **Metódusok**
  - +use(c: Camambert): bool - Amikor meghívódik elgázosítja a szobát vagyis átállítja a szoba gassed értékét true-re
  - +use(t: Transistor): bool - transistor használatot végez és kikapcsolja a transistorokat
  - +use(tabla: Tablatorlo): bool - használatkor megvédi a diákot és cleanerre teszi a szobát, átállítja az értékét true-ra.
  - +use(a: Airfreshener): bool - Használatkor kiszedi a gázt a szobából, vagyis a Room gassed paramétere false lesz és meg is semmisül

8.1.18 Room

- **Felelősség**

  A szobákat reprezentáló osztály, tárolja a benne lévő játékosokat (hallgatók, oktatók) és a benne lévő felvehető objektumokat.

- **Attribútumok**
  - - capacity: int - A szoba befogadóképességét adja, ez a játék elején kap egy alapértéket
  - - gassed: bool - Ha az értéke “true”, az azt jelenti, hogy a szoba mérgezett.
  - - roomId: int - A szoba azonosító száma
  - -cleaner: bool – azt jelzi, hogy a szobában használtak-e táblatörlőt
  - -janitorcleaned: bool – azt jelzi, hogy takarítva van-e
  - -stickyCount: int - a takarítás óta belépő játékosokat számolja, minden belépés
  - -doorList: Arraylist<Door> - ajtók listája
  - -playerList: Arraylist<Player> - játékosok listája
  - -itemList: Arraylist<Item> - tárgyak listája

- **Metódusok**
  - +removeItem(i: Item): void - Törli az itemList-jéből a paraméterül kapott tárgyat
  - +removePlayer(p: Player): void - Törli a playerList-jéből a paraméterül kapott tárgyat
  - +addItem(i: Item): void - hozzáadja az itemList-jéhez a megkapott tárgyat
  - +addPlayer(p: Player): bool - Analóg, ha sikerült (tehát elfér a játékos), akkor hozzáadja és attól függően, hogy a szoba épp gázos/cleaner-es megpróbálja a megfelelő módon stunolni az adott játékost, majd igazzal tér vissza. Inicializálásnál nem probléma, hiszen kezdetben nincs se gázos, se táblatörlős szoba.

    - Pszeudokód az algoritmushoz:
      ha playerList < kapacitás
        Hozzáad játékos

Ha gázos
Stun játékos
Ha Táblatörlős
StunTeacher játékos
Visszatér: igaz
Egyébként visszatér hamis

- +mergeRoom(r1: Room, r2: Room): bool - Két szoba összeolvasztását végzi, ha sikerült, igazzal tér vissza
- +addDoor(d: Door): void - Ajtó hozzáadása a szoba doorList-jéhez
- +removeDoor(d: Door): void - Ajtó eltávolítása a listából
- +killPlayer(p: Player): bool – A játékost megöli, további körökben nem létezik, inkább jelzés értékű a controller felé
- +win(): void – Amennyiben a logarléc felvételre került, akkor ezzel jelzi a Board felé, hogy nyertek a hallgatók
- +changeRoom(p: Player, d: Door): bool - Átküldi az ajtó másik felére a játékost. Megnézi, hogy az ajtó transferPlayer függvénye igazzal tér vissza akkor ő is
- +newRoom(): Room - Új szoba inicializálása, ha egy szoba szétoszlik, az új szoba referenciájával tér vissza, minden attribútumát a sajátja szerint állítja és létrehoz egy ajtót is a két szoba közé, amit hozzá is ad mind a kettőhöz
- +pvp(p: Player): void – a szobában lévő játékosok által megpróbálja megölni az újonnan belépőt.Végigmegy a playerListjén és minden játékos kill függvényének átadja a belépő játékost
- +makeGassed(): void - elgázosodik a szoba (gassed attribútum = true-ra állítás) és minden szobában lévő játékosnak meghívja a stun függvényét
- +makeUnGassed(): void – kitisztul a gáztól a szoba (gassed attribútum = false-ra állítás)
- +makeClean(): void - tábltörlős lesz a szoba (cleaner attribútum = true-ra állítás) és minden szobában lévő játékosnak meghívja a stunTeacher függvényét
- +makeJanitorCleaned(): void - takarító által takarított szobára hívódik, ez átállítja a janitorCleaned attribútumot true-ra, ezután kezdődik a stickyCount számolás
- +killAll(p: Player): a szobába belépő játékos megpróbál megölni mindent bent lévőt, a transferPlayer hívja. Végigmegy a playerList-jén és a belépő játékos kill() függvényének átadja a játékosokat egymás után
- +acceptPairing(b: Board, r: Room): bool - Párba állítja egy másik szobával, ezzel megoldva a merge által okozott problémákat. Ha sikerült, akkor igazzal tér vissza.
- +resetStickyCount(): void - lenullázza a számlálót (újabb takarításnál hívódik)
- +sendOut(): void - takarító takarításkor hívja, végigmegy a bent lévő játékososkon, és az ajtókon, és megpróbálja átléptetni az ajtón, ha valamelyik ajtón sikerül átmenni akkor ugyanezt a következő játékosokkal is megcsinálja
    - Pszeudokód az algoritmushoz:

      for minden játékos in playerList
      For minden ajtó in doorList
      Ha ChangeR(ajtó, játékos) igaz
      Break

8.1.19 RoomPairing Interfész

- **Felelősség**

Az interfész felel a merge funkció helyes működéséért, az abban felmerülő problémák kiküszöböléséért. Ha sikerült a merge, igazzal tér vissza. Az utolsó két metódus nem engedi őket mergelni.

Metódusok

- +pair(r1: Room, r2: Room): bool
- +pair(cr: CursedRoom, r: Room): void
- +pair(r: Room, cr: CursedRoom): void

8.1.20 Student

- **Felelősség**

A hallgatókat reprezentáló osztály.

Ősosztály:

- Player

Interfészek

- StudentProtection: a diák védelmére szolgáló interfész
- PutDown: tárgyak letételét/használatát végző interfész

Attribútumok: csak az örökölt attribútumok

Metódusok:

- +die(): bool- végigmegy az itemList-jén, mindegyik tárgyra meghívja az acceptSP függvényt és ha bármelyik nála lévő tárgyra ez igazzal tér vissza akkor nem hal meg és visszatér hamissal. Ha egyik se tudja megvédeni, akkor igazzal tér vissza, mert meghalt
    - Pszeudokód az algoritmushoz:

```
For minden item in itemList
        Ha item acceptSP igazzal tér vissza
                Visszatér hamis
Visszatér igaz
```

- +stun(): bool - Felülírt függvény. végigmegy az itemList-jén, mindegyik tárgyra meghívja az acceptGasProtect függvényt és ha bármelyik nála lévő tárgy ez igazzal tér vissza akkor nem bénul és visszatér hamissal. Ha egyik se tudja megvédeni, akkor igazzal tér vissza és bénul
    - Pszeudokód az algoritmushoz:

```
For minden item in itemList
        Ha item acceptGasProtect igazzal tér vissza
                Visszatér hamis
For minden item in itemList
dropItem item
Visszatér igaz
```

- +putDown(i: Item): void - tárgyak letételéért felelős függvény. Önmaga itemList-jéből kiveszi a tárgyat és meghívja a location-je addItem függvényét

- +pairing(t1: Transistor, t2: Transistor): bool – Felülírt függvény az ősosztályból
- +protect(t: TVSZ): bool - Meghívja a Tvsz durabminus függvényét és ha az igazzal tér vissza (tehát meg tudja védeni), akkor igazzal tér vissza, amúgy hamissal
- +protect(b: Beer): bool - Meghívódik, igazzal tér vissza, hiszen a sör mindig meg tudja védeni a játékost és ezzel együtt el is ejt miatta a játékos egy random tárgyat, igazzal tér vissza, ha sikerül
- +protect(i: Item): bool - Bármilyen megvédés miatt kell, mindig hamissal tér vissza
- +use(c: Camambert): bool - Amikor meghívódik elgázosítja a szobát vagyis átállítja a szoba gassed értékét true-re
- +use(t: Transistor): bool - transistor használatot végez és kikapcsolja a transistorokat
- +use(tabla: Tablatorlo): bool - használatkor cleanerre teszi a szobát (átállítja az értékét true-ra)
- +use(a: Airfreshener): bool - Használatkor kiszedi a gázt a szobából, vagyis a Room gassed paramétere false lesz és meg is semmisül
- +pairTrans(t1: Transistor, t2: Transistor): bool - párosít két paraméterként kapott transistort, true ha sikeres, false ha hamis
- +pairItem(i1: Item, i2: Item): bool - két paraméterként kapott Itemet párosít
- +pickUp(l: Logarlec): void - Logarléc felvétele, speciális esemény, meghívódik a win függvénye
- +cleanRoom(): void – csak visszatér, mert nem tud takarítani

8.1.21 StudentProtection Interfész

- **Felelősség**

A hallagató automatikus védelmében játszik szerepet. Értesíti a megfelelő objektumot. Amennyiben sikeresek, igazzal térnek vissza a függvényei.

- **Metódusok**
  - +protect(t: TVSZ): bool - Meghívódik, ha meg kell védeni a játékost és meg is védi, igazzal tér vissza, ha sikerül
  - +protect(b: Beer): bool - Meghívódik, ha megvédi a játékost és ezzel együtt el is ejt miatta a játékos egy random tárgyat, igazzal tér vissza, ha sikerül
  - +protect(i: Item): bool - Bármilyen megvédés miatt kell

8.1.22 Tablatorlo

- **Felelősség**

A táblatörlőt reprezentáló osztály, az Item ősosztály leszármazottja, hisz ő maga is egy tárgy

- **Ősosztály**
  - Item
- **Interfészek**
  - CycleBased
- **Attribútumok**
  - -durability: int - Megszüneséig hátralévő körök száma
- **Metódusok**
  - +acceptPutDown(s: Student): void - Válasz a kérésre, a tárgy letevődik és meghívódik a use függvénye, mellyel cleaner attribútumát a szobának igazra állítja
  - +durabMinus(): bool - Csökkenti a durability-t eggyel, ha nem sikerül hamis

8.1.23 Teacher

- **Felelősség:**

Az oktatókat reprezentáló osztály, akik Player absztrakt ős leszármazottjai, hisz ők is játékosok

- **Ősosztály**:
  - Player
- **Metódusok:**
  - +kill(p: Player): bool – p játékos megölése, meghívja a p játékos die függvényét
  - +stunTeacher(): bool- Felülírt stunTeacher függvény, mely igazzal tér vissza
  - +pickUp(l: Logarlec): void – Megpróbálja felvenni a Logarlécet, de nem engedi neki
  - +die (): bool – nem tud meghalni, így csak false-val tér vissza
  - +cleanRoom(): void – csak visszatér, mert nem tud takarítani
  - +stun(): bool - Felülírt függvény. végigmegy az itemList-jén, mindegyik tárgyra meghívja az acceptGasProtect függvényt és ha bármelyik nála lévő tárgy ez igazzal tér vissza akkor nem bénul és visszatér hamissal. Ha egyik se tudja megvédeni, akkor igazzal tér vissza és bénul
    - Pszeudokód az algoritmushoz:

```
For minden item in itemList
        Ha item acceptGasProtect igazzal tér vissza
                Visszatér hamis
For minden item in itemList
dropItem item
        Visszatér igaz
```

8.1.24 Transistor

- **Felelősség:**

A Transistor objektumot reprezentáló osztály. A transistor egyszerre egy felvehető tárgy is, de amint párosítják és leteszik már ajtóként funkcionál.

- **Ősosztály:**
  - Item
- **Attribútumok:**
  - -pair: Transistor - a párban hozzákapcsolt transistort tárolja
  - -location: Room – A helyét tárolja, melyik szobában van
  - -active: bool - Aktív-e, vagyis lehet e használni, vagy még “aktiválni kell”
- **Metódusok**
  - +acceptPickUp(s: Student): void- Játékoshoz kerül, elfogadja a felvételt és meghívódik a pickUp függvénye a játékosnak
  - +acceptPutDown(s: Student): void - meghívja a student use függvényét a tárgy paraméterrel amiben letevődik/használódik a tárgy
  - +makePair(t: Transistor): bool – ha a párja null és a kapott transistornak sincs párja vagy a kapottnak a párja ő maga akkor beállítja párjának a kapottat és visszatér igazzal
  - +activate(): void - Aktiválja a tranzisztort és beállítja az active attribútumot true-ra
  - +teleportPlayer(s: Student): bool , ha sikerült igaz- Játékos mozgatása
    - Pszeudokód az algoritmushoz:

```
Ha a pair arrivingPlayer igazzal tér vissza
                Location removePlayer
```

Visszatér igaz
Visszatér hamis

- +setRoom(r: Room): void - Szoba beállítása, ahova kerül(location beállítása)
- +acceptPairing(p: Player, t: Transistor): bool – Tranzisztorok párosítását valósítja meg, ha ez sikerült, igazzal tér vissza,
- +acceptPairing(p: Player,i: Item): bool – Tárgyak párosítását valósítja meg, de ez nem lehetséges, így csak jelez, hogy nem lehetséges
- +arrivingPlayer(s: Student): bool - Meghívja a location-je addPlayer függvényét s paraméterrel és ha az igazzal tér vissza akkor ő is, he nem akkor hamissal

8.1.25 TVSZ

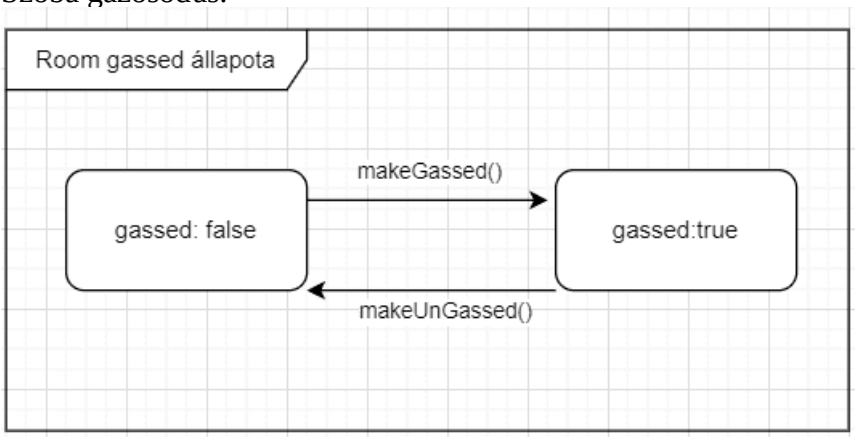
- **Felelősség**

A TVSZ-t reprezentáló osztály, mely az Item ősből származik, hiszen ő maga is egy tárgy

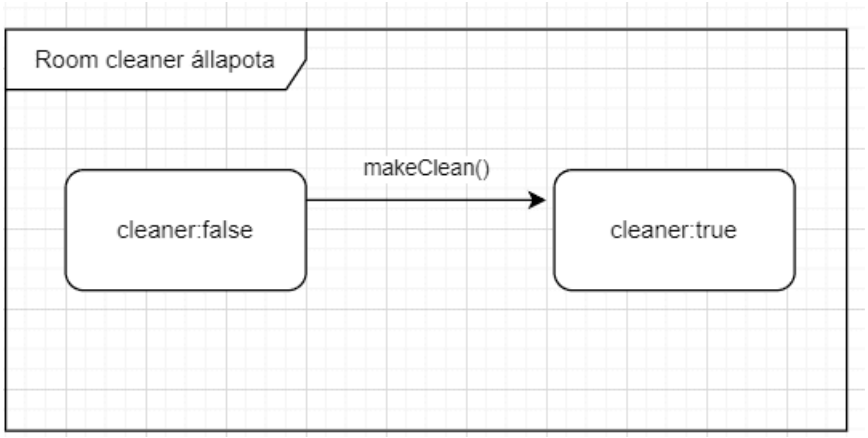
- **Ősosztály**
  - Item
- **Attribútumok**
  - -durability: int – Megszünéséig hátralévő körök száma
  - isFake: bool - Ez a tulajdonság szolgál arra, hogy maga a tárgy, vagyis a TVSZ igazi-e, vagy hamis, szóval haszontalan. Ha true, akkor hamis, ha false, akkor igazi a tárgy.
- **Metódusok**
  - +acceptSP(s: Student): bool– A hallgatót megvédi, ha sikerül igaz, meghívja a játékos protect függvényét.
  - +durabMinus(): bool - Csökkenti a durability-t eggyel, ha nem sikerül hamissal tér vissza

State chart diagrammok

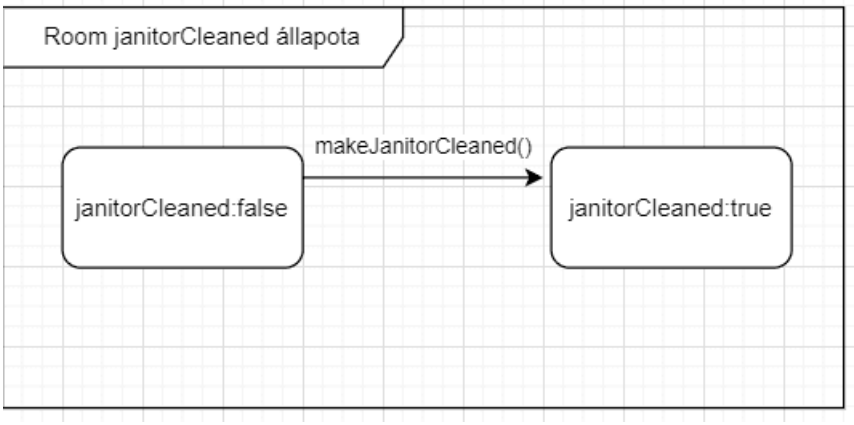
Szoba gázosodás:



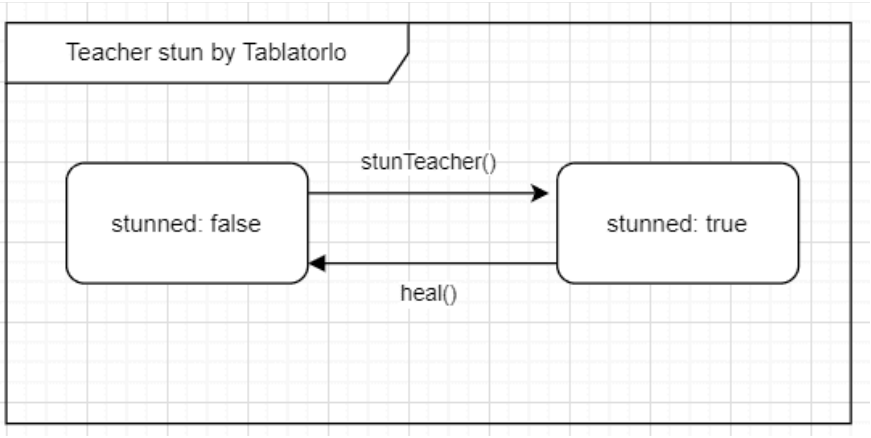
Szoba cleanerré tétele:



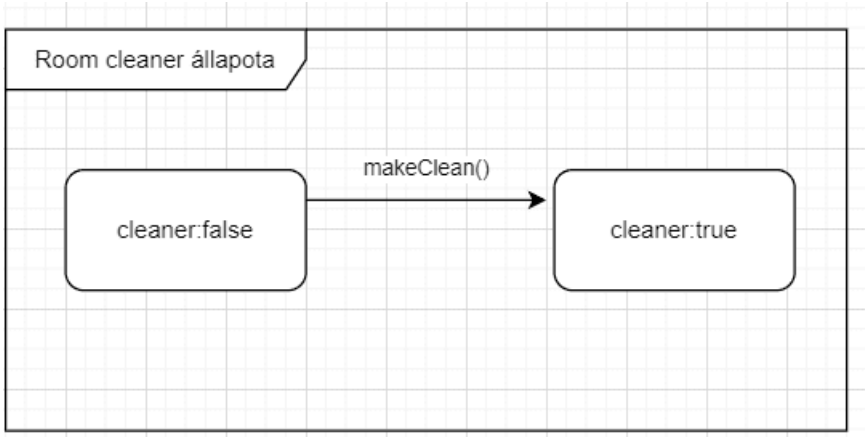
Szoba janitorCleaned-é tétele:



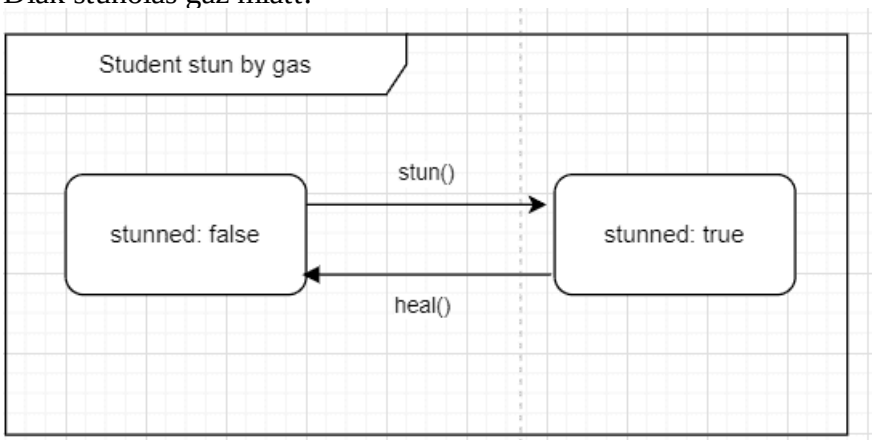
Tanár stunolás Tablatorlo miatt:



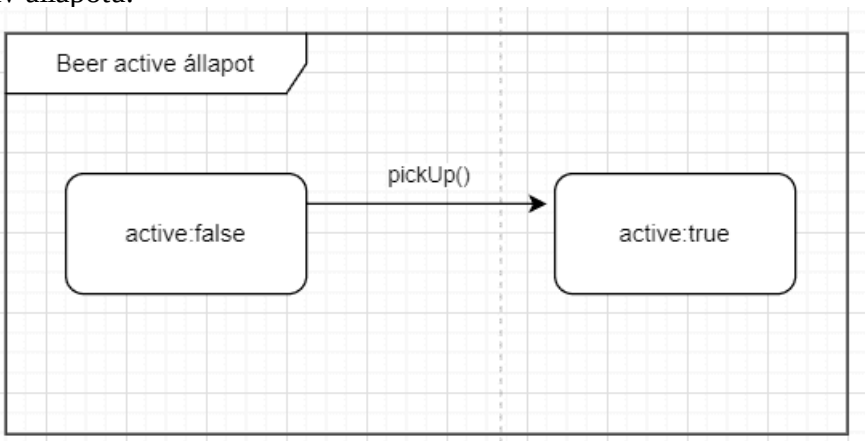
Tanár stunolás gáz miatt:



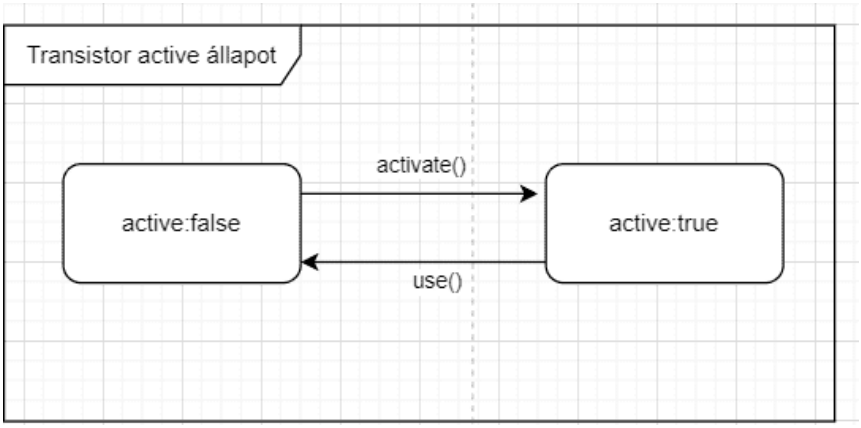
Diák stunolás gáz miatt:



Sör aktív állapota:



Transistor akív állapota:


Transistor active állapot
activate()
active:false
active:true
use()

A tesztek részletes tervei, leírásuk a teszt nyelvén

Bemeneti nyelv változásai:

- Megváltozott parancsok:
  - **addRoom** <tipus> <nev> <kapacitas>
    Leírás: Szoba hozzáadása
    Opciók: Típusa (room/cursedroom), neve, kapacitása
- Új parancsok:
  - **listPlayerAttribs** <*jatekos*>
    Leírás: Játékos igaz értékű állapotainak listázása (új)
    Opciók: Játékos
  - **listRoomAttribs** <*szoba*>
    Leírás: Szoba igaz értékű állapotainak listázása (új)
    Opciók: Szoba
  - **addPlayerToRoom** <*tipus*> <*nev*> <*szoba*>
    Leírás: Játékos szobához adása (új)
    Opciók: Játékos típusa (student, teacher, janitor), neve, szoba
  - **addItemToRoom** <*tipus*> <*hamis-e*> <*nev*> <*durability*> <*szoba*>
    Leírás: Tárgy szobához adása (új)
    Opciók: Tárgy típusa, eredetisége, neve, durability (ha nincs az adott tárgynak akkor –1), szoba, ahova kerül
  - **addItemToPlayer** <*tipus*> <*nev*> <*durability*> <*jatekos*>
    Leírás: Tárgy játékoshoz adása (új)
    Opciók: Tárgy típusa, neve, durability (ha nincs az adott tárgynak akkor –1), játékos, akihez kerül

Kimeneti nyelv változásai:

- Megváltozott eredmények:
  - **pickUp**
    Eredmény: tárgy felvétele
    Megjelenítés: “<felvevő játékos neve>: <tárgy neve> - <OK/FAIL> <nyert> (csak, ha logarlécet vesz fel, ami nem hamis)
  - **putDown**
    Eredmény: letétel/használat helye
    Megjelenítés: “<játékos neve>: <Szoba neve> <tárgy neve> putDown”
  - **useDoor**
    Eredmény: ajtó használat sikeressége
    Megjelenítés: “<ajtó neve>: <játékos neve> - <OK/FAIL>
- Kimeneti nyelv új elemei
  - **listPlayerAttribs** <*jatekos*>
    Eredmény: Játékos igaz értékű állapotainak listázása (új)
    Megjelenítés: “<jatekos neve>: <igazattribútum1> <igazattribútum2>...
  - **listRoomAttribs** <*szoba*>
    Eredmény: Szoba igaz értékű állapotainak listázása (új)
    Megjelenítés: “<szoba neve>: <igazattribútum1> <igazattribútum2>...

Teszteset1: Játékos lép egyirányú ajtón, oda-vissza

- **Leírás**

2 szoba van, 1 kapacitással. Legyenek ezek r1 és r2. A két szobát egy egyirányú ajtó (d12) köt össze, oly módon, hogy r1-ből lehet lépni r2-be, de fordítva már nem. Az r1 szobában egy Hallgató (s) tartózkodik. ‘s’ Hallgató átlép az r1 szobába a d12 ajtón keresztül, majd megpróbál r1 szobába visszalépni ugyan azon ajtón keresztül. Ez nem sikerülhet.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A hallgató megfelelő módon mozog szobák között, az ajtótól függően.

- **Bemenet**

addRoom room 1 1
addRoom room r2 1
addDoor d12 r1 r2 false true
addPlayerToRoom student s r1
useDoor d12 s
listRoom r1
listRoom r2
useDoor d12 s
listRoom r1
listRoom r2

- **Elvárt kimenet**
  (add* függvények nem rendelkeznek kimenettel)

d12: s – OK
r1:
r2: s
d12: s – FAIL
r1:
r2: s

Teszteset2: Gázos szoba, maszk használat

- **Leírás**

Legyen 2 szoba, r1 és r2, ahol r2 szoba gázos. Legyen egy őket összekötő ajtó (d12), illetve két hallgató s1 és s2, ahol ‘s1’-nek nincs semmi tárgya, ‘s2’-nek pedig van egy maszkja (m). ‘s1’ és ‘s2’ az r1 szobában tartózkodnak. Először s1 hallgató átlép az r2 szobába, a d12 ajtón keresztül, majd s2 hallgató is. ‘s1’ elkábul, ezért eldobja a tárgyait, ‘s2’-t pedig megvédi a maszk.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Gázos szobák megfelelő működés, maszk használata

- **Bemenet**

addRoom room r1 1
addRoom room r2 1
addDoor d12 r1 r2 true true
addPlayerToRoom student s1 r1
addPlayerToRoom student s2 r1
addItemToPlayer beer b1 1 s1
addItemToPlayer mask m 1 s2
useDoor d12 s1
listRoom r2
listPlayerItem s1
listPlayerAttribs s1
useDoor d12 s2
listRoom r2
listPlayerItem s2
listPlayerAttribs s2

- **Elvárt kimenet**

d12: s1 – OK
r2: b1 s1
s1:
s1: stunned
d12: s2 – OK
r2: b1 s1 s2
s2:
s2:

Teszteset3: Gázosság

- **Leírás**

Legyen egy r szoba, benne egy s hallgató és neki 3 tárgya a, c, és m, melyek rendre légfrissítő, camambert és maszk. Első lépésként s lerakja a sajtot, amitől a szoba gázos lesz, ő viszont nem kábul el, mert van maszkja. Második lépésben s lerakja a légfrissítőt, amitól a szoba állapota megváltozik, már nem gázos többé.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Camamber és légfrissítő használata.

- **Bemenet**

addRoom room r 1
addPlayerToRoom student s r
addItemToPlayer airfreshener a s
addItemToPlayer camambert c s
addItemToPlayer mask m s
putDown c s
listRoomAttribs r
listPlayerAttribs s
putDown a s
listRoomAttribs r

- **Elvárt kimenet**

r: gassed
s:
r:

Teszteset4: Inventory megtelt, tárgyfelvétel

- **Leírás**

Legyen egy r szoba, benne egy b6 sör és egy s hallgató, akinél van 5 sör, b1-b5. A játékos megpróbálja felvenni a b6 sört, viszont ez eredménytelen lesz, mivel betelt az eszköztára. Ezt követően eldobja a b1 sört majd felveszi a b6 sört. Ez már sikeres lesz.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A játékos eszköztárának és a tárgyak felvételének működése.

- **Bemenet**

AddRoom room r 1
AddItemToRoom beer b6 1 r
AddPlayerToRoom student s r
AddItemToPlayer beer b1 1 s
AddItemToPlayer beer b2 1 s
AddItemToPlayer beer b3 1 s
AddItemToPlayer beer b4 1 s
AddItemToPlayer beer b5 1 s
PickUp b6 s
PutDown b1 s
PickUp b6 s

- **Elvárt kimenet**

s: b6 – FAIL
s: r b1 putDown
s: b6 – OK

Teszteset5: TVSZ teszt

- **Leírás**

Legyen két szoba, r1 és r2 és az ezeket összekötő d12 ajtó. Az r1 szobában legyen két hallgató s1 és s2, utóbbinál pedig egy 1 durabilitású, tvsz nevű TVSZ. Az r2 szobában legyen egy t oktató. ‘t’ oktató átlép az r1 szobába. Ekkor s1 hallgató meghal, eltűnik a szobából, s2 hallgatót pedig megmenti a tvsz, viszont az megsemmisül.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Tvsz helyes működés és az oktatók helyes viselkedése diákkal való találkozáskor

- **Bemenet**

addRoom room r1
addRoom room r2
addDoor d12 r1 r2 true true
addPlayerToRoom student s1 r1
addPlayerToRoom student s2 r1
addPlayerToRoom teacher t r2
addItemToPlayer tvsz tvsz s2
useDoor d12 t
listRoom r1
listPlayerItem s2

- **Elvárt kimenet**

d12: t – OK
r1: s2 t
s2:

Teszteset6: Sör használata

- **Leírás**

  2 szoba van: r1 és r2. Ezeket összeköti egy ajtó: d12. r1-ben egy hallgató (s) van egy sörrel(b). r2-ben egy oktató (t) van. Az oktató az ajtón keresztül átlép a másik szobába és megpróbálja megölni a hallgatót, akit megvéd a söre, viszont a sör képessége miatt el is dobja azt.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

  Sör tényleg megvéd és tényleg eldobat egy tárgyat a hallgatóval használat után. Várható hiba: rossz sorrendben listázza ki a szobában lévő embereket és tárgyakat.

- **Bemenet**

  addRoom room r1 2
  addRoom room r2 1
  addDoor d12 r1 r2 true true
  addPlayerToRoom student s r1
  addItemToPlayer: Beer, b, 3, s
  addPlayerToRoom teacher t r2
  listRoom r1
  listRoom r2
  listPlayerItem s
  useDoor d12 t
  listRoom r1
  listRoom r2
  listPlayerItem s

- **Elvárt kimenet**

r1: s
r2: s
s: b
d12: t –OK
r1: s t b
r2:
s:

Teszteset7: táblatörlő

- **Leírás**
  2 szoba van: r1 és r2. r1-ben van egy hallgató, akinél van egy táblatörlő. Hallgató leteszi a táblatörlőt. r2-ben van egy oktató, aki átlép a másik szobába és stunnolódik a táblatörlő miatt.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  Táblatörlő működése
- **Bemenet**
  addRoom Room r1 2
  addRoom Room r2 2
  addDoor d12 r1 r2 true true
  addPlayerToRoom student s r1
  addItemToPlayer Tablatorlo ta 3 s
  addPlayerToRoom teacher t r2
  listRoomAttribs r1
  listPlayerAttribs t
  putDown ta s
  listRoomAttribs
  useDoor d12 t
  listPlayerAttribs t
- **Elvárt kimenet**
  r1:
  t:
  s: r1 ta putDown
  r1: cleaner
  d12: t –OK
  t: stunned

Teszteset8: Hamis és igazi logarléc

- **Leírás**

Egy szoba van: r1. Szobában van 2 logarléc, egy hamis és egy igazi: L1, L2. Szobában van még egy hallgató, aki felveszi először a hamis, majd az igazi logarlécet. A hamis felvétele után nem történik semmi, az igazi felvétele után megnyerik a játékot.

- **Ellenőrzött funkcionalitás, várható hibahelyek**
  Hamis és igazi tárgyak működése, játék megnyerése
- **Bemenet**
  addRoom r1 1
  addPlayerToRoom student s r1
  addItemToRoom Logarlec false L1 1 r1
  addItemToRoom Logarlec true L2 1 r1
  listRoom r1
  pickUp L1 s
  listRoom r1

listPlayerItem s1
pickUp L2 s

- **Elvárt kimenet**
  r1: s L1 L2
  s: L1 –OK
  r1: s L2
  s: L1
  s: L2 –OK nyert

Teszteset9: Teli szobába lépés

- **Leírás**
  2 szoba van: r1, r2. Mindkét szobának a kapacitása 1 és mindkét szobában egy hallgató van: s1, s2. A 2 szobát összeköti egy ajtó: d12. r1-ben lévő hallgató megpróbál átlépni r2-be, de nem tud, mert r2 tele van.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  Teli szobába lépés sikertelensége
- **Bemenet**
  addRoom Room r1 1
  addRoom Room r2 1
  addDoor d12 r1 r2 true true
  addPlayerToRoom student s1 r1
  addPlayerToRoom student s2 r2
  listRoom r1
  listRoom r2
  useDoor d12 r1
  listRoom r1
  listRoom r2
- **Elvárt kimenet**
  r1: s1
  r2: s2
  d12: s –FAIL
  r1: s1
  r2: s2

Teszteset10: Átkozott szoba

- **Leírás**
  Van 2 szoba: r1 és r2. r1.ben van egy hallgató(s), r2 átkozott. Hallgató átlép r2-be. r2 ajtaja bezárul. Hallgató megpróbál visszalépni, de nem tud.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  Átkozott szobák működése
- **Bemenet**
  addRoom CursedRoom r1 1
  addRoom Room r2
  addPlayerToRoom student s r2
  addDoor d12 r1 r2 true true
  listRoom r1
  listRoom r2
  useDoor d12 s
  listRoom r1
  listRoom r2
  closeDoor r1

useDoor d12 s
listRoom r1
listRoom r2

- **Elvárt kimenet**

r1:
r2: s
d12: s –OK
r1: s
r2:
d12: s –FAIL
r1: s
r2:

Teszteset11: Tranzisztor használat

- **Leírás**

A teszthez létrehozandó egy hallgató, 2 szoba, valamint 2 tranzisztor. A hallgató a tranzisztorokat felveszi, párosítja, egyet letesz, majd átlép a másik szobába. Ott leteszi a másik tranzisztort

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A tranzisztor helyes működését szeretnénk tesztelni.

- **Bemenet**

addRoom room r1
addRoom room r2
addDoor d12 r1 r2 true true
addPlayerToRoom student s1 r1
addItemToPlayer transistor t1 s1
addItemToPlayer transistor t2 s1
pair s1 t1 t2
putDown s1 r1 t1
useDoor d12 s1
putDown s1 r2 t2
listRoom r1
listRoom r2

- **Elvárt kimenet**

Sikeres párosítás t1 t1
S1: r1 t1 putDown
d12: s1 –OK
s1: r2 t2 putDown
r1:s1
r2:

Teszteset12: Split

- **Leírás**

A teszthez létrehozunk egy szobát, benne egy hallgatót és egy tárgyat, ezután kiadjuk a split parancsot, aminek hatására egy új szoba jön majd létre az eredetivel ajtóval összekötve.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A split helyes működését szeretnénk tesztelni.

- **Bemenet**

addRoom room r1

addPlayerToRoom student s1 r1
addItemToRoom transistor true t1 r1
split r1
listRoom r1
listAllRoom

- **Elvárt kimenet**

r1 split, uj szoba r2
r1:s1 t1
r1 r2

Teszteset13: Iterálás

- **Leírás**

A teszthez létrehozunk 3 szobát, az egyikben egy hallgatót és egy sört, 1 durability-vel. A hallgató szobájától különböző szobákban létrehozunk egy-egy oktatót.Ezután a hallgatóval belépünk az egyikhez, iterálunk, majd átlépünk a másikhoz.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Az iterálás helyes működését szeretnénk tesztelni.

- **Bemenet**

addRoom room r1
addRoom room r2
addRoom room r3
addDoor d12 r1 r2 true true
addDoor d23 r2 r3 true true
addPlayerToRoom student s1 r1
addPlayerToRoom teacher t1 r2
addPlayerToRoom teacher t2 r3
AddItemToPlayer beer b1 1 s1
useDoor d12 s1
beerIterate
useDoor d23 s1
listRoom r1
listRoom r2
listRoom r3

- **Elvárt kimenet**

d12: s1 –OK
b1 iterate
d23 s1 –OK
r1:
r2: t1
r3: t2

Teszteset14: Takarító gázos szobát takarít

- **Leírás**

A teszthez létrehozunk 2 szobát, benne az egyikben egy takarítót. A két szobát ajtóval összekötjük, és amelyikben nincs takarító, az gázos.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A takarító helyes működését szeretnénk tesztelni.

- **Bemenet**

addRoom room r1
addRoom room r2
addDoor d12 r1 r2 true true
gasRoom r2
addPlayerToRoom janitor j1 r1
useDoor d12 j1
listRoomAttribs r2

- **Elvárt kimenet**

d12: s1 –OK
r2:

Teszteset15: Takarító gázos szobát takarít és onnan kitessékel

- **Leírás**

A teszthez létrehozunk 2 szobát, benne az egyikben egy takarítót és két diákot. A két szobát ajtóval összekötjük, és amelyikben nincs takarító és diákok, az gázos. Az egyik diákhoz adunk egy maszkot. Mindhárman átlépnek a gázos szobába, a takarító kitessékeli a maszkos diákot, vissza az eredeti szobába.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A takarító helyes működését szeretnénk tesztelni.

- **Bemenet**

addRoom room r1
addRoom room r2
addDoor d12 r1 r2 true true
gasRoom r2
addPlayerToRoom janitor j1 r1
addPlayerToRoom janitor s1 r1
addPlayerToRoom janitor s2 r1
addItemToPlayer mask mask 1 s1
useDoor d12 s1
useDoor d12 s2
useDoor d12 j1
listRoom r1
listRoom r2
listRoomAttribs r2

- **Elvárt kimenet**

d12: s1 –OK
d12: s2 –OK
d12: j1 –OK
r1: s1
r2: s2 j1
r2:

Teszteset16:Merge

- **Leírás**

Van 2 szoba r1 és r2. r1-ben egy hallgató (s), míg r2-ben egy sör van (b). A 2 szoba mergelődik.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Merge ellenőrzése

- **Bemenet**

addRoom Room r1 2
addRoom Room r2 1
AddPlayerToRoom student s r1
AddItemToRoom Beer true b 3 r2
listRoom r1
listRoom r2
Merge r1 r2
listRoom r1

- **Elvárt kimenet**

r1: s
r2: b
r1 r2 sikeres merge
r1: s b

Napló

Kezdet	Időtartam	Résztvevők	Leírás
2024.04.10. 14:15	4 óra	Fodor A. Fodor D.	8.1-es rész megírása, inkonzisztens esetek keresése, eddigi hibák javítása
2024.04.12	3 óra	Földi	Teszteset 1-5 kidolgozása
2024.04.14 7:30	3 óra	Ludányi	11-15 tesztek kidolgozása
2024.04.14 11:00	3 óra	Mikola	Teszteset 6-10 kidolgozása
2024.04.14. 14:15	1,5 óra	Fodor A. Fodor D.	Pszeudokódok megírása, state-chartok szerkesztése
2024.04.14 20:00	1 óra	Mikola	Teszteset 16 kidolgozása, fedlap hozzáadás, apró szerkesztések

10. Prototípus beadása

6 – ripgyork

Konzulens:
Ádám Zsófia

Csapattagok

Fodor Attila	EUGN1B	afodor998@gmail.com
Fodor Dávid	D02DBR	dfodor999@gmail.com
Földi Balázs	AB8Y3S	fbalu8@gmail.com
Ludányi Barnabás	V5PWP4	ludanyib2003@gmail.com
Mikola Bálint István	TCV0Y9	mikola.balint.istvan@gmail.com (kapcsolattartó)

2024.03.03

10. Prototípus beadása

10.1 Fordítási és futtatási útmutató

10.1.1 Fájllista

Fájl neve	Méret [KB]	Keletkezés ideje	Tartalom
Airfreshener.java	0.4	2024.04.20	Airfreshener osztály
App.java	3.17	2024.03.24.	Főprogram
Beer.java	1.3	2024.03.24	Sör osztály
Board.java	6.25	2024.03.24	Játéktábla osztály
Camambert.java	0.578	2024.03.24	Camambert osztály
CommandHandler.java	19	2024.04.20	Parancsolvasásért felel
CursedRoom.java	1.37	2024.03.24	Átkozott szobák osztálya
CycleBased.java	0.267	2024.03.24	CycleBased Interfész
CycleUsage.java	0.817	2024.03.24	CycleUsage osztály
Door.java	2.06	2024.03.24	Ajtó osztály
GasProtect.java	0.422	2024.03.24	GasProtect interfész
Item.java	2.37	2024.03.24	Tárgyak ősosztálya
Janitor.java	4.31	2024.04.20	Takarító osztálya
Logarlec.java	0.630	2024.03.24	Logarléc tárgy osztálya
Mask.java	0.975	2024.03.24	FPP2 maszk tárgy osztálya
Pairing.java	0.277	2024.03.24	Pairing interfész
PickUp.java	0.455	2024.03.24	PickUp interfész
Player.java	6.37	2024.03.24	Játékos ősosztály
PutDown.java	0.462	2024.03.24	PutDown interfész
Room.java	6.98	2024.03.24	Szoba osztály
RoomPairing.java	0.220	2024.03.24	RoomPairing interfész
Student.java	5.75	2024.03.24	Hallgató osztály
StudentProtection.java	0.415	2024.03.24	StudentProtection interfész
Tablatorlo.java	0.909	2024.03.24	Táblatörlő tárgy osztálya
Teacher.java	1.67	2024.03.24	Oktató osztály
Transistor.java	3.15	2024.03.24	Tranzisztor tárgy osztálya
Tvsz.java	0.992	2024.03.24	TVSZ tárgy osztálya
Test1.txt – Test16.txt	3.57	2024.04.20	Tesztbemenetek
Assert1.txt – Assert16.txt	0.829	2024.04.20	Elvárt kimenetek

10.1.2 Fordítás

LogarGame mappából:
javac .\Logarlec\src\logarlecTheGame*.java .\Logarlec\src\logarlecTheGame\Model*.java
.\Logarlec\src\logarlecTheGame\Model\Interfaces*.java
.\Logarlec\src\logarlecTheGame\Model\Item*.java

10.1.3 Futtatás

A kód futtatása fordítás után érhető el. Ehhez a Logarlec\src\ mappából kell a következő parancsot terminálból kiadni:
java -cp . logarlecTheGame.App

10.2 Tesztek jegyzőkönyvei

10.2.1 Teszteset1

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Fodor Attila
Teszt időpontja	2024.04.27

10.2.2 Teszteset2

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Fodor Attila
Teszt időpontja	2024.04.27

10.2.3 Teszteset3

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Fodor Attila
Teszt időpontja	2024.04.27

10.2.4 Teszteset4

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Fodor Dávid
Teszt időpontja	2024.04.27

10.2.5 Teszteset5

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Fodor Dávid
Teszt időpontja	2024.04.27

10.2.6 Teszteset6

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Fodor Dávid
Teszt időpontja	2024.04.27

10.2.7 Teszteset7

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Földi Balázs
Teszt időpontja	2024.04.27

10.2.8 Teszteset8

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Földi Balázs
Teszt időpontja	2024.04.27

10.2.9 Teszteset9

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Földi Balázs
Teszt időpontja	2024.04.27

10.2.10 Teszteset10

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Ludányi Barnabás
Teszt időpontja	2024.04.27

10.2.11 Teszteset11

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Ludányi Barnabás
Teszt időpontja	2024.04.27

10.2.12 Teszteset12

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Ludányi Barnabás
Teszt időpontja	2024.04.27

10.2.13 Teszteset13

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Mikola Bálint
Teszt időpontja	2024.04.27

10.2.14 Teszteset14

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Mikola Bálint
Teszt időpontja	2024.04.27

10.2.15 Teszteset15

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Mikola Bálint
Teszt időpontja	2024.04.27

10.2.16 Teszteset16

[Az alábbi táblázatot az utolsó, sikeres tesztfuttatáshoz kell kitölteni]

Tesztelő neve	Mikola Bálint
Teszt időpontja	2024.04.27

10.3 Értékelés

[A projekt kezdete óta az értékelésig eltelt időben tagokra bontva, százalékban.]

Tag neve	Tag neptun	Munka százalékban
Fodor Attila	EUGN1B	20
Fodor Dávid	D02DBR	20
Földi Balázs	AB8Y3S	20
Ludányi Barnabás	V5PWP4	20
Mikola Bálint István	TCV0Y9	20

10.4 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2024.04.19	2,5 óra	Fodor A Fodor D Földi Ludányi Mikola	Értekezlet. Feladatok felosztása
2024.04.24	8 óra	Fodor A	Kiadott feladat elvégzése
2024.04.26	8 óra	Fodor A	Kiadott feladat elvégzése
2024.04.24	8 óra	Fodor D	Kiadott feladat elvégzése
2024.04.26	8 óra	Fodor D	Kiadott feladat elvégzése
2024.04.24	8 óra	Földi	Kiadott feladat elvégzése
2024.04.26	8 óra	Földi	Kiadott feladat elvégzése
2024.04.24	8 óra	Ludányi	Kiadott feladat elvégzése
2024.04.26	8 óra	Ludányi	Kiadott feladat elvégzése
2024.04.24	8 óra	Mikola	Kiadott feladat elvégzése
2024.04.26	8 óra	Mikola	Kiadott feladat elvégzése

11. Grafikus felület specifikációja

6 – ripgyork

Konzulens:
Ádám Zsófia

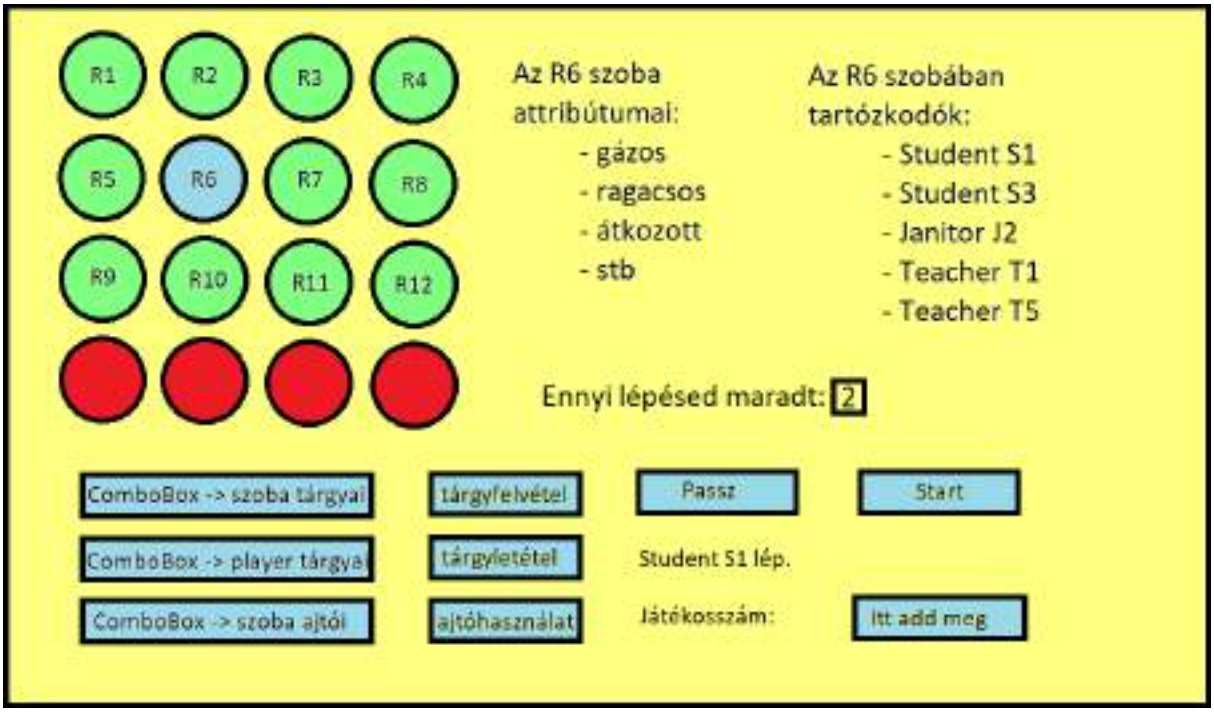
Csapattagok

Fodor Attila	EUGN1B	afodor998@gmail.com
Fodor Dávid	D02DBR	dfodor999@gmail.com
Földi Balázs	AB8Y3S	fbalu8@gmail.com
Ludányi Barnabás	V5PWP4	ludanyib2003@gmail.com
Mikola Bálint István	TCV0Y9	mikola.balint.istvan@gmail.com (kapcsolattartó)

2024.05.05

11. Grafikus felület specifikációja

11.1 A grafikus interfész



Ahogy a képen is látható, a szobákat a bal felső sarokban található 16 darab kör reprezentálja. Ebből alapesetben 12 van használatban, de a szobák száma a merge és split műveletek következtében 1 és 16 közt bárhol lehet. A használatban lévő szobák színe zöld, és bennük a szoba neve látható. A nem használt körök pirosak. Az épp lépő játékos helyzetét a kék színű szoba jelzi. A szobák alatt 3 combobox és 4 gomb található. A felső sor comboboxában a szoba tárgyai közül lehet választani, a mellette lévő gombbal pedig a kiválasztott tárgyat felvenni. A középső sorban a játékos saját tárgyai közül tud választani, majd a gomb lenyomásával a kiválasztott tárgyat letenni. Az alsó sorban az ajtók közül tud választani, hogy melyiken akar áthaladni, a gomblenyomásra pedig meg is teszi ezt. A képernyő jobb oldalán 2 felsorolás található. A bal oldali az aktuális szoba attribútumait sorolja fel, míg a jobboldali a szobában található játékosok listáját adja meg. Ez alatt található egy sor, amely az épp lépő játékos számára rendelkezésre álló lépések számát adja meg. A jobb alsó részben 2 gomb található. A „Passz” gombbal a játékos tovább tudja adni a lépést a következő játékosnak, ez olyankor hasznos, ha például nem tud, vagy nem akar lépni semmit. A „Start” gombbal a játék a legelején elindítható, játék közbeni lenyomásával a játék resetelődik. A Start és Pass gomb alatt egy rövid szöveg tájékoztat arról, hogy melyik játékos köre van. Ez alatt található a játékosszám mező, ahol játékot elkezdő játékosok számát lehet megadni. Ez a mező a start lenyomása után az aktuálisan játékban lévő játékosok számát mutatja.

11.2 A grafikus rendszer architektúrája

11.2.1 A felület működési elve

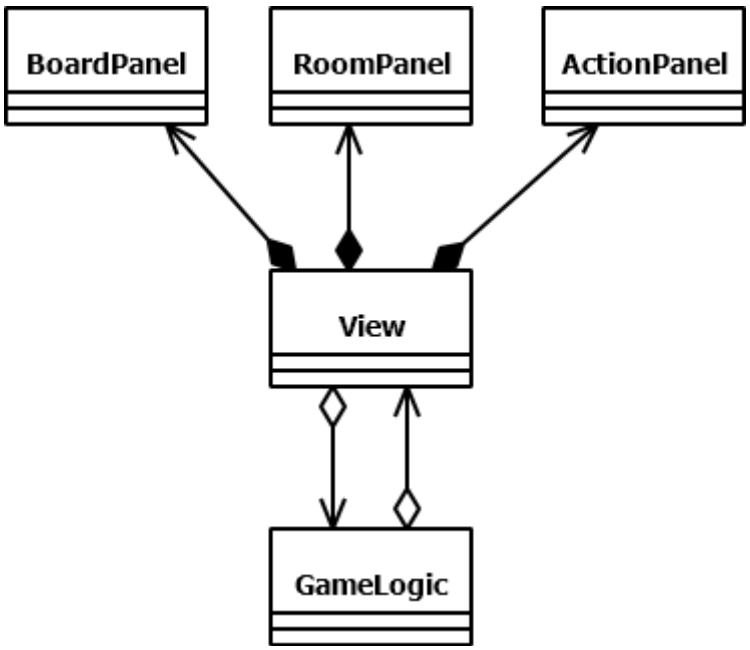
A grafikus felület vázát a View osztály fogja megvalósítani. Ennek az osztálynak lesz egy GameLogic attribútuma, ami a játékosok, tanárok és diákok léptetésével fog foglalkozni. A View osztályon belül lesz egy JFrame, ami tartalmazni fog 3 darab JPanelt:

- BoardPanel: Ezen a panelen lesz rajta a pálya képe. Ismernie kell az összes szobát és azt is, hogy ezek közül melyik az a szoba, amiben az éppen soron lévő játékos tartózkodik.
- RoomPanel: Ezen a panelen lesznek kilistázva, annak a szobának a tulajdonságai, amiben az éppen soron lévő játékos tartózkodik (gázos-e, csúszos-e, kik vannak benne, stb.) Ismernie kell az éppen aktuális szobát.

- ActionPanel: Ez a panel 3 comboboxot és 4 gombot fog tartalmazni. A 3 combobox rendre az aktuális szoba ajtajait, a játékos tárgyait és a szoba tárgyait fogja tartalmazni. A gombokkal lehet átmenni a comboboxban kiválasztott ajtón vagy felvenni a kiválasztott tárgyat, vagy letenni egy tárgyat. A negyedik gomb segítségével le lehet mondani az aktuális játékos maradék akciópontjainak elköltéséről. Az előbbiken kívül ezen a panelen lesz megjelenítve egy akciópont számláló, ami azt mutatja, hogy a játékosnak mennyi akciópontja van még (0-ra csökkenésekor a következő játékos jön.) Ismernie kell az éppen aktuális játékost és a szobát, amiben tartózkodik.

A harmadik panelen elhelyezkedő gombok és comboboxok segítségével lehet majd vezérelni a játékot. Egy gomb megnyomásának hatására megtörténik az akció (tárgy- felvétel, letétel, ajtón átmenés) és jelez a View-nak, hogy frissíteni kell a képet. A View ennek hatására a GameLogic segítségével frissíti a grafikus felületet a módosult adatokkal. Következő játékos körének kezdetén a GameLogic értesíti a View-t, hogy frissüljön (sok esetben itt érdemi frissülés nem lesz, erre az oktatók és takarítók lépései miatt lesz szükség.) A GameLogic-nak is lesz egy View attribútuma, aminek tud szólni, ha új játékos köre kezdődik és frissítésre van szükség.

Struktúradiagram:

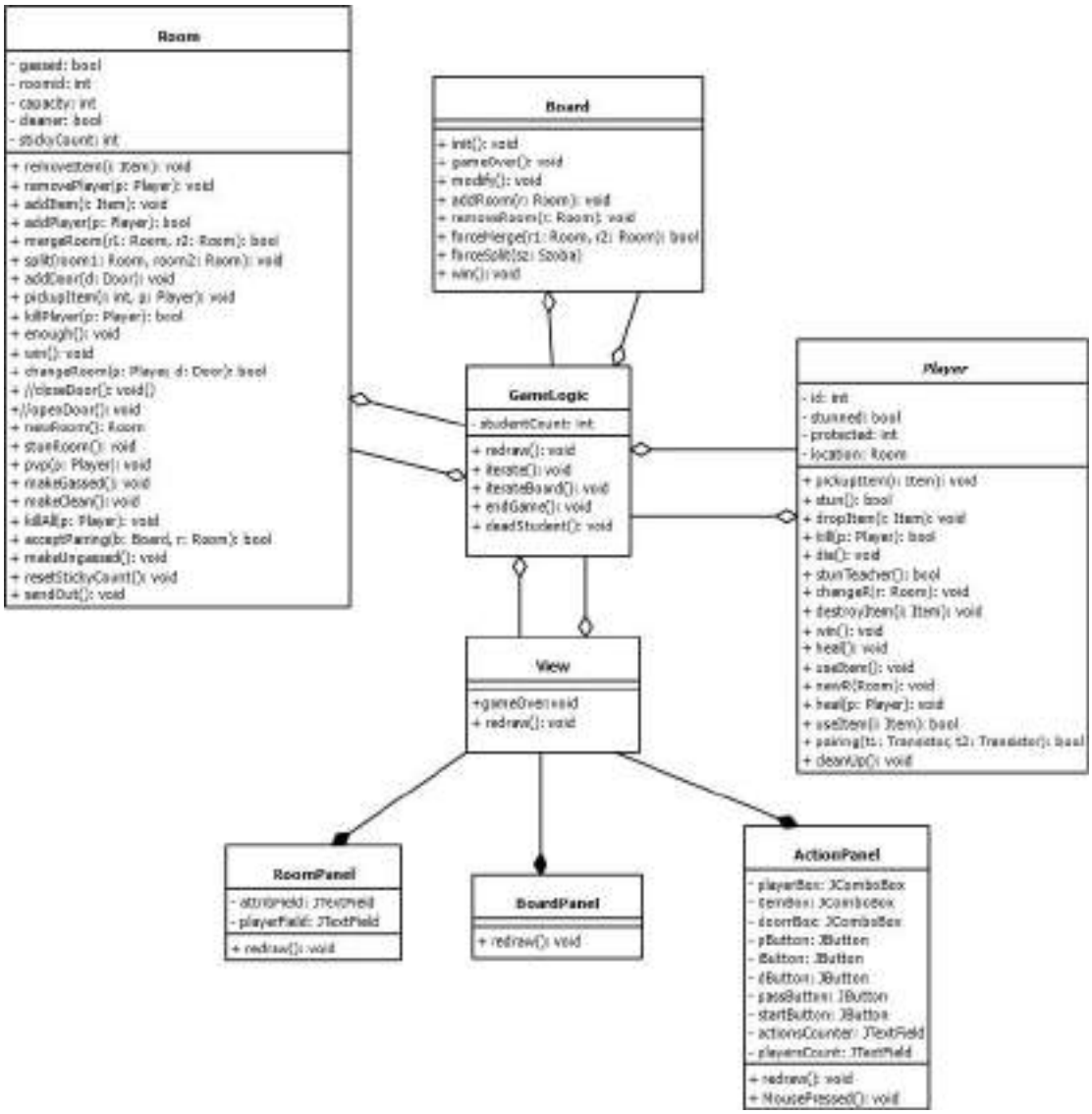


(Tudom, hogy a rombuszos vonal másik végén nem kéne nyílnak lennie, de a Gaphorban csak ilyen rombuszos vonalat találtam.)

Diagramok bővebben: lejjebb

11.2.2 A felület osztály-struktúrája

A grafikus megvalósítása pull alapú lesz. Minden játékos köre elején frissül a View, az éppen aktuális játékos és szoba alapján (ezeket a GameLogic-ból fogja megtudni). Ezen kívül minden gombnyomás után frissülni fog a View, hogy az éppen aktuális állapotot jelenítse meg.



11.3 A grafikus objektumok felsorolása

11.3.1 GameLogic

- **Felelősség**

Az osztály felelős minden játéklogika megvalósításáért. Pl az iterálás. Ehhez számontartja a játékosokat, a szobákat és a táblát. Emellett közvetett módon felel a játék grafikus megjelenítéséért a View osztályon keresztül.

- **Kapcsolatok**
  - Room
  - Board
  - Player

- **Attribútumok**
  - StudentCount: int – a játékosok száma az adott játékban

- **Metódusok**
  - +redraw(): void - értesíti a View osztályt, ha változás történt. Ennek hatására a View osztály frissíti a grafikus felületet.
  - +iterate(): void – CycleBased objektumok iterálása
  - +iterateBoard(): void – A pálya iterálása
  - +endGame(): void – A játék végét jelzi
  - +deadStudent(): void - Hallgató halálát jelzi

11.3.2 View

- **Felelősség**
  - Az osztály felel a grafikus felület megjelítéséért.

- **Metódusok**
  - +gameOver(): void – Jelzi a játék végét
  - +redraw(): void – A grafikus felület újra felrajzolása

11.3.3 RoomPanel

- **Felelősség**

A grafikus felület jobb felső sarkában szereplő Szoba attribútumainak és objektumainak a listájának grafikus megjelenítéséért felel.

- **Attribútumok**
  - -attribField: JTextField – A szoba attribútumainak a listája
  - -playerField: JTextField – A szoba játékosainak listája

- **Metódusok**
  - +redraw(): void – A felület újrarajzolása

11.3.4 BoardPanel

- **Felelősség**

A grafikus felület bal felső sarkában szereplő pálya grafikus megjelenítéséért felel.

- **Attribútumok**

- **Metódusok**
  - +redraw(): void – A felület újrarajzolása

11.3.5 ActionPanel

- **Felelősség**

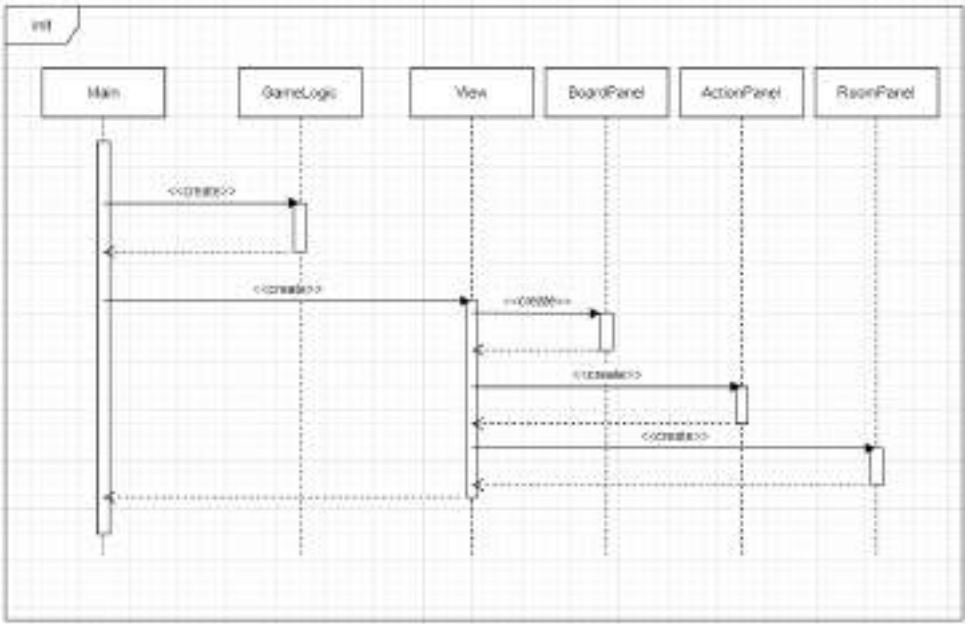
A grafikus felület alsó részén szereplő felület grafikus megjelenítéséért felel.

- **Attribútumok**
  - -playerBox: JComboBox - Játékos tárgyai
  - -itemBox: JComboBox – Szoba tárgyai
  - -doorBox: JComboBox – Szoba ajtajai
  - -pButton: JButton - Tárgyletétel
  - -iButton: JButton - Tárgyfelvétel
  - -dButton: JButton - Ajtóhasználat
  - -passButton: JButton - Kör passzolása
  - -startButton: JButton - Játék indítása
  - -actionsCounter: JTextField - Hátralévő akciók száma
  - -playersCount: JTextField - Játékosok számát adja meg, illetve lehet megadni kezdés előtt

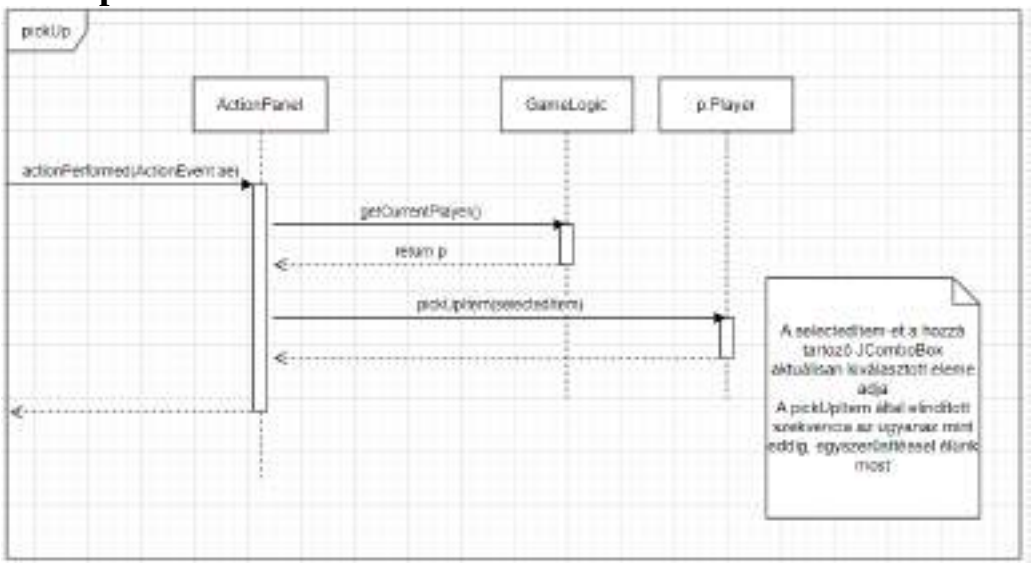
- **Metódusok**
  - +redraw(): void – A felület újrarajzolása
  - +ActionPerformed(): void – A gombnyomások kezelését végzi

11.4 Kapcsolat az alkalmazói rendszerrel

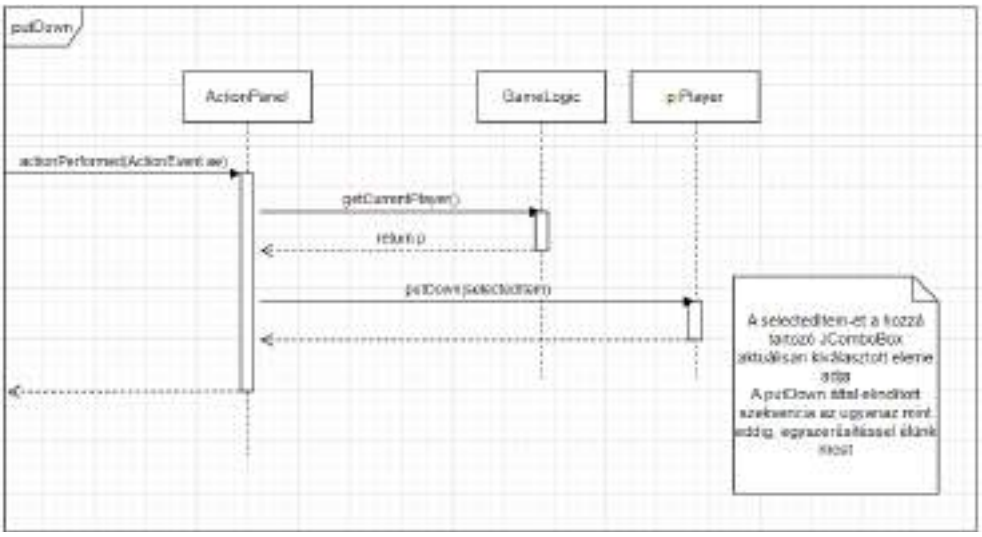
Init



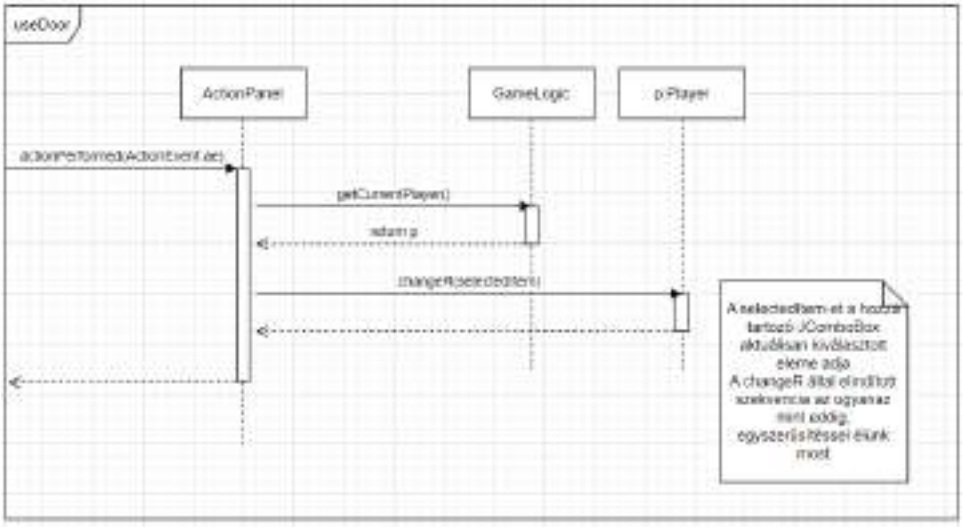
PickUp



PutDown



UseDoor



11.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2024.05.02 12:00	2,5 óra	Fodor A Fodor D Földi Ludányi Mikola	Értekezlet.
2024.05.03. 18:00	2 óra	Ludányi	11.1-es rész elkészítése
2024.05.03 20:00	2 óra	Mikola	11.2 és 11.3-as rész elkészítése
2024.05.04 11:00	1 óra	Mikola	Hibajavítások
2024.05.04 13:00	2 óra	Fodor D.	Osztálydiagramm és hozzá tartozó javítások
2024.05.04 15:00	2 óra	Fodor A.	Szekvenciadiagrammok

13. Grafikus változat beadása

6 – ripgyork

Konzulens:
Ádám Zsófia

Csapattagok

Fodor Attila	EUGN1B	afodor998@gmail.com
Fodor Dávid	D02DBR	dfodor999@gmail.com
Földi Balázs	AB8Y3S	fbalu8@gmail.com
Ludányi Barnabás	V5PWP4	ludanyib2003@gmail.com
Mikola Bálint István	TCV0Y9	mikola.balint.istvan@gmail.com (kapcsolattartó)

2024.05.19

13. Grafikus változat beadása

13.1 Fordítási és futtatási útmutató

13.1.1 Fájllista

Fájl neve	Méret [KB]	Keletkezés ideje	Tartalom
ActionPanel.java	5.65	2024.05.10	Grafikus felület akciópanelje
Airfreshener.java	0.4	2024.04.20	Airfreshener osztály
App.java	3.17	2024.03.24.	Főprogram
Beer.java	1.3	2024.03.24	Sör osztály
Board.java	6.25	2024.03.24	Játéktábla osztály
BoardPanel.java	2.25	2024.05.10	Szobák megjelenítése táblán
Camambert.java	0.578	2024.03.24	Camambert osztály
CommandHandler.java	19	2024.04.20	Parancsolvasásért felel
CursedRoom.java	1.37	2024.03.24	Átkozott szobák osztálya
CycleBased.java	0.267	2024.03.24	CycleBased Interfész
Door.java	2.06	2024.03.24	Ajtó osztály
GameLogic.java	6.66	2024.05.10	Játék view és modell osztályait köti összi
GameMenu.java	7.03	2024.05.10	Főmenü
GasProtect.java	0.422	2024.03.24	GasProtect interfész
Item.java	2.37	2024.03.24	Tárgyak ősosztálya
Janitor.java	4.31	2024.04.20	Takarító osztálya
Logarlec.java	0.630	2024.03.24	Logarléc tárgy osztálya
LostImage.java	1.52	2024.05.10	Vereség esetén játék vége
Mask.java	0.975	2024.03.24	FPP2 maszk tárgy osztálya
Pairing.java	0.277	2024.03.24	Pairing interfész
PickUp.java	0.455	2024.03.24	PickUp interfész
Player.java	6.37	2024.03.24	Játékos ősosztály
PutDown.java	0.462	2024.03.24	PutDown interfész
Room.java	6.98	2024.03.24	Szoba osztály
RoomPairing.java	0.220	2024.03.24	RoomPairing interfész
RoomPanel.java	3.98	2024.05.10	Szoba attribútumai és benne tartozkodók grafikus megj.
Student.java	5.75	2024.03.24	Hallgató osztály
StudentProtection.java	0.415	2024.03.24	StudentProtection interfész
Tablatorlo.java	0.909	2024.03.24	Táblatörlő tárgy osztálya
Teacher.java	1.67	2024.03.24	Oktató osztály
Transistor.java	3.15	2024.03.24	Tranzisztor tárgy osztálya
Tvsz.java	0.992	2024.03.24	TVSZ tárgy osztálya
Test1.txt – Test16.txt	3.57	2024.04.20	Tesztbemenetek
Assert1.txt – Assert16.txt	0.829	2024.04.20	Elvárt kimenetek
View.java	2.97	2024.05.10	Megjelenítés
WinImage.java	1.36	2024.05.10	Nyerés megj.

13.1.2 Fordítás és telepítés

Projlab mappából:
javac .\Logarlec\src\logarlecTheGame*.java .\Logarlec\src\logarlecTheGame\Model*.java .\Logarlec\src\logarlecTheGame\Model\Interfaces*.java .\Logarlec\src\logarlecTheGame\Model\Item*.java .\Logarlec\src\logarlecTheGame\Controller*.java .\Logarlec\src\logarlecTheGame\View*.java

13.1.3 Futtatás

A kód futtatása fordítás után érhető el. Ehhez a Logarlec\src\ mappából kell a következő parancsot terminálból kiadni:
java -cp . logarlecTheGame.App -play

13.2 Értékelés

Tag neve	Tag neptun	Munka százalékban
Fodor Attila	EUGN1B	20
Fodor Dávid	D02DBR	20
Földi Balázs	AB8Y3S	20
Ludányi Barnabás	V5PWP4	20
Mikola Bálint	TCV0Y9	20

13.3 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2024.05.09 17:00	1.5 óra	Fodor A. Fodor D. Földi Ludányi Mikola	Értekezlet. Feladatok kiosztása
2024.05.11	10 óra	Fodor A.	Kiadott feladat elvégzése
2024.05.11	10 óra	Fodor D.	Kiadott feladat elvégzése
2024.05.11	10 óra	Mikola	Kiadott feladat elvégzése
2024.05.12	10 óra	Ludányi	Kiadott feladat elvégzése
2024.05.12	10 óra	Földi	Kiadott feladat elvégzése

Összefoglalás

6 – ripgyork

Konzulens:
Ádám Zsófia

Csapattagok

Fodor Attila	EUGN1B	afodor998@gmail.com
Fodor Dávid	D02DBR	dfodor999@gmail.com
Földi Balázs	AB8Y3S	fbalu8@gmail.com
Ludányi Barnabás	V5PWP4	ludanyib2003@gmail.com
Mikola Bálint István	TCV0Y9	mikola.balint.istvan@gmail.com (kapcsolattartó)

2024.05.22

14. Összefoglalás

14.1 A projektre fordított összes munkaidő

Tag neve	Munkaidő (óra)
Fodor Attila	130
Fodor Dávid	130
Földi Balázs	130
Ludányi Barnabás	130
Mikola Bálint	130
Összesen	**650**

- ***A feltöltött programok forrássorainak száma***

Fázis	Kódsorok száma
Szkeleton	1700
Prototípus	1500
Grafikus változat	1300
Összesen	**4500**

14.2 • Projekt összegzés

14.2.1 Mit tanultak a projektből konkrétan és általában?

A konkrét tudásgyarapodás leginkább a Java nyelv és a Github használat terén figyelhető meg, míg általános érvényű tanulságként a csapatmunkát tudnánk felhozni, ami sokat fejlődött. Határidőre dolgozás képessége is nagy mértékben előkerült és hétről hétre fejlődött a csapat tagjaiban.

14.2.2 Mi volt a legnehezebb és a legkönnyebb?

A legnehezebb a Github bizonyos anomáliáit kiküszöbölni, és a kódolós részeknél a feladatok felosztása, párhuzamosítása volt. Esetlegesen nagyobb hibák gyors kiküszöbölése a következő beadásig. A legkönnyebb a kezdeti tervekben a feladatok felosztása, a megvalósítás logikájának kitalálása volt.

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

A legtöbb helyen igen, de voltak részek ahol kevés pont volt, és rengeteg óra befektetést igényelt. Részenként kevés tapasztalattal álltunk, előzetes hiányosságok miatt, így a pontszámhoz képest túl sok munka kellett, hogy a részt megfelelő minőséggel teljesítsük

14.2.4 Ha nem, akkor hol okozott ez nehézséget?

Az analízis modell elkészítésénél volt, hogy a kevés pontszám miatt kissé későn kezdtük meg a munkálatokat, ezért a befejezés hétfő hajnalig tartott. Grafikus felület terveinél merült még fel a probléma.

14.2.5 Milyen változtatási javaslatuk van?

A konzultációkon a feladatok megbeszélése néhol tévútra csalt, hogy pontosan mit is várnak tőlünk, és ez sok ponthullajtást eredményezett. Szekvenciadiagrammokhoz előzetesen útmutató kiadása szükséges lenne, hisz ilyen mélységekbe nem mentünk bele tanulmányaink során eddig.

14.2.6 Milyen feladatot ajánlanának a projektre?

Valami ehhez hasonló teljesen megfelelő véleményünk szerint, de akár egy kezdetleges aknakereső vagy passziánsz típusú játék (kisebb csavarokkal benne) is jó feladat lenne.

14.2.7 Egyéb kritika és javaslat

Nincsen, ha a tárgyat egy lelkes csapat, megfelelő mennyiségű idő ráfordításával, konzisztensen csinálja hétről hétre, akkor teljesíthető.

15. Végleges változtatások

15.1 Specifikáció[1]

Projektünkben egy Logarléc névre hallgató körökre osztott stratégia játékot fogunk megvalósítani. A játék célja, hogy a játékosok mágikus szobákon átverekedve magukat és különböző tárgyakat használva megkaparintsák a mágikus logarlécet, miközben próbálják elkerülni a gonosz oktatókat.

A pályát (négyzet) **kör** alakú szobák négyzetrácsa alkotja. **A pálya determinisztikus, előre, általunk kidolgozott**. Egy szobából (véges számú) **bármennyi** ajtó nyílhat másik szobákba. Mivel a szobák rendszere egy elvarázsolt útvesztőt alkot; a szoba egyik ajtaján kilépve közel sem biztos, hogy a szomszédos szobába fog jutni a játékos. Egy szobában 3 féle ajtó lehet: Olyan, amin csak távozni lehet; Olyan, amin keresztül csak érkezni lehet és olyan, ami 2 irányú. Egy szobából (egyértelműen) **nem** látszik az összes olyan szoba, ahova lépni lehet. Minden szobának van befogadóképessége, ami egy (véletlenszerűen[2i] generált szám 1 és játékos szám + oktatószám között) **konstans szám**. Minden játék elején (játékosszámtól függő) **12** darab szoba generálódik le (véletlenszerű) **az előre megadott** szomszédokkal. Létrejöttükkor bizonyos szobák el vannak gázosítva. Ezekbe a szobákba a belépés eszméletvesztéssel és a következő ciklusból való kimaradással jár, kivéve, ha van maszkja a játékosnak (lásd lentebb a ciklusokat és a tárgyakat). A szobák száma a későbbiekben még változhat (lásd lentebb a szobák osztódását és egyesülését.)

A játék ún. ciklusokra van osztva. Egy ciklus addig tart, amíg minden játékos (és oktató, lásd: lentebb) sorra nem kerül, és végre nem hajtja a körét.

Egy játékosnak minden körben van 3 elköltendő akciópontja. A játékos körének akkor és csak akkor van vége, ha mindhárom akciópontját elköltötte. Egy játékos az akciópontjait 4 féle akcióra költheti el: Átlépés egy másik szobába, egy szobában lévő tárgy felvétele, hallgatónál lévő tárgy letétele, **valamint tranzisztorok párosítására**. (Tárgyakról bővebben: lentebb).
A játékosokra a játék egész ideje alatt oktatók „vadásznak” az alább kifejtett módon. A játék elején (játékosszámtól függő) **előre megadott** darab oktató (véletlenszerűen elhelyezésre kerül szobákban) **egy adott szobában kerülnek elhelyezésre**. Az oktatók köre mindig a játékosok köre után következik. Az oktatóknak is minden körben 3 akciópontjuk van, melyeket véletlenszerűen költenek el minden ciklusban. Az oktatókat (lehet látni akkor is, ha nincs senki azonos szobában velük) **nem lehet látni, csak ha azonos szobában tartózkodnak vele**. Ha egy hallgató egy olyan szobába lép, ahol oktató is tartózkodik, vagy pedig egy oktató lép egy hallgatót tartalmazó szobába, akkor még abban a körben kiszívja a játékos lelkét, ezzel kiejtve az adott játékost az adott játékból. **Egy másik játékos a Takarító. takarító. A szobák befogadóképessége rá is érvényes. Ha belép egy szobába, minden mozogni képes embert kitessékel onnan. Ha gázos szobába lép, kiszellőztet, megszüntetve a szoba gázosságát. A szobák a takarítást követően adott számú látogató után ragacsossá válnak: a bennük lévő és bennük letett tárgyakat nem lehet felvenni.**

[1] A javítások vastag betűtípussal vannak kiemelve, mellette zárójelben az eredeti koncepció olvasható
[2] Végül felhagytunk a random generált objektumokkal, mivel több átgondolást követően felfedeztük, hogy ezek sok anomáliát szülhetnek.

A hallgatók feladatát különböző tárgyak segítik. Minden tárgy a játék elején jön létre és (véletlenszerűen elhelyezésre kerül szobákban) **előre megadott szobákban kerülnek elhelyezésre**. (A játék elején létrejövő tárgyak száma arányos a játékosok számával.) Kívülről nem látszik, ha egy tárgy egy szobában van. Meglátni és felvenni egy tárgyat csak akkor tudunk, ha belépünk az őt tartalmazó szobába. Alap helyzetben minden tárgy inaktív állapotban van és csak onnantól érvényesül a képessége, hogy egy játékos felvette. Játék közben nem tud új tárgy keletkezni, csak elhasználódni. Bizonyos tárgyaknak vannak tartósságpontjaik, amik nullára csökkenés esetén elveszik az adott tárgy képességét. (, ebben az esetben a tárgy még a játékosnál marad.) Tárgyakat az oktatók is tudnak felvenni, viszont kizárólag „elkobzás” céljából, használni nem tudják ezeket a tárgyakat, illetve a logarlécet még elkobozni sem tudják. Minden játékosnál és oktatónál maximum 5 tárgy lehet. **Egyes tárgyaknak (tvsz, maszk, logarléc) létezik "hamis" változata, amelyiknek nincs az eredeti tárgyra jellemző jó tulajdonsága. Például a hamis logarléc felvételével nem lehet nyerni.** A lentiekben a részletezzük a játékban található tárgyak tulajdonságait:

- Logarléc: Amint aktiválják (felveszik a földről), a játékot megnyerik a játékosok (oktató nem veheti fel.)
- TVSZ denevérbőrre nyomtatott példánya: Ha ez a tárgy egy játékosnál és ez a játékos egy szobába kerül egy oktatóval, akkor az oktató nem ejti ki a játékost, a TVSZ veszít egy tartósságpontot és a játék halad tovább. Aktiváláskor a tárgy három tartósságponttal rendelkezik.
- Szent sörös poharak: Ezeknek a tárgyaknak a képessége megegyezik a TVSZ denevérbőrre nyomtatott példányainak képességével, annyi különbséggel, hogy a szent sörös poharak minden ciklusban veszítenek egy tartósságpontot. **A még aktív söröskorsót használva a hallgatók elejtik az egyik náluk levő tárgyat.**
- Nedves táblatörlő: Ha egy játékosnál egy nedves táblatörlő van és egy szobába kerül egy oktatóval, akkor egy ciklus erejéig megbénítja az oktatót. A táblatörlő 3 tartósságponttal kezd és ciklusonként veszít egyet.
- Dobozolt káposztás camembert: Ezt a tárgyat letéve, a tárgy elgázosítja a szobát, ami a játék végéig gázos is marad.
- FFP2 maszk: Hordozója nem ájul el az elgázosított szobában. Három tartósságponttal rendelkezik és minden olyan ciklusban veszít egy pontot, melyben a viselője egy elgázosított szobában tartózkodik valamennyi időre.
- Tranzisztor: Két tranzisztor automatikusan összepárosodik, ha felveszi őket egy játékos. Egy összepárosított tranzisztorpár egyik darabját lehelyezve a játék végéig a másik tranzisztort birtokló játékos visszateleportálhat abba a szobába, ahova a tranzisztort letette.
- **Légfrissítő: Egyszerhasználatos tárgy. Gázos szobában lerakva semlegesíti a gázhatást.**

A játéktéren uralkodó mágia miatt a játék elején (véletlen) **adott** darab ajtó megátkozódik (és miután háromszor átmentek rajta, 3 ciklus erejéig eltűnik, majd újra felbukkan). **Ezen szobákban az ajtók véletlenszerű időközönként bezáródhatnak, vagy kinyílhatnak.** Amíg egy ajtó el van tűnve, addig nem lehet áthaladni rajta. Szintén a fent említett mágia miatt minden ciklus elején a szobák véletlenszerűen osztódhatnak, helyben maradhatnak vagy pedig összeolvadhatnak egy szomszédos szobával. Két szomszédos szoba egyesülésével létrejövő szoba a korábbi két szoba tulajdonságaival és szomszédaival rendelkezik, de a befogadóképessége a nagyobb szoba befogadóképességével lesz azonos. Az újonnan létrejött szoba tartalmazni fogja az összes tárgyat, amit az őt létrehozó szobák tartalmaztak. Az osztódó szoba két olyan szobára válik szét, amelyek egymás szomszédai lesznek, és megosztoznak a korábbi szoba képességein és szomszédain (a korábbi szomszédok, vagy csak az egyik, vagy csak a másik "új" szobának lesznek szomszédai). Osztódáskor a tárgyak, játékosok és oktatók (véletlenszerűen kerülnek az egyik, illetve a másik szobába) **az eredeti szobában maradnak**. **16 szobánál több nem lehet**. Szobák egyesülésére az egyetlen feltétel, hogy 3 szobának minimum lennie kell a pályán.

Egy játékon cikluskorlát nincs, addig tart, amíg minden játékos ki nem esik, vagy valaki fel nem veszi a logarlécet.

---

i