

Szoftver projekt laboratórium

70 – trolleybus

Konzulens:
Rácz Gergely

Tagok:

Bende Zoltán	GS7BDA	bendez95@gmail.com
Szatmáry Péter	C29106	peter.szatmary@windowslive.com
Králik Bálint Bendegúz	O7L11H	xeeonz@gmail.com
Máté László	H3R4OG	mate.laci7@gmail.com
Kapitány Gergely	UBOJ5G	gregoriuscaptain@gmail.com

2018.05.18

2. Követelmény, projekt, funkcionális

2.1. Bevezetés

2.1.1. Cél

A projekt követelményeinek, felépítésének és funkcionálisának megfogalmazása, ezek alapján dönten a fejlesztés folyamatáról és ütemezéséről, emellett megfogalmazni a végleges programmal kapcsolatos felépítésbeli és működésbeli követelményeket. Fejlesztés közben ezeket be kell tartani, lényegben eltérni tőlük nem szabad.

2.1.2. Szakterület

A Szoftver Projekt Laboratórium (szoftprojlab - BMEVIIIA02) megoldását adja. Az elkészítendő szoftver egy szórakoztató célú számítógépes játék. Típusát tekintve logikai, stratégiai, ezért azoknak ajánlott, akik szeretik ezen műfajt.

2.1.3. Definíciók, rövidítések

BME: Budapesti Műszaki és Gazdaságtudományi Egyetem

VIK - Villamosmérnöki és Informatika Kar

IIT - Irányítástechnika és Informatika Tanszék

szoftprojlab: Szoftver Projekt Laboratórium

JDK: Java Development Kit, a Java fejlesztői környezete és APIja.

JRE: Java Runtime Environment, a Java futtatókörnyezete.

Git: Egy verziókezelő rendszer.

GitHub: A Git verziókezelő rendszerre épülő internetes szolgáltatás.

bitbucket: A Git verziókezelő rendszerre épülő internetes szolgáltatás.

IntelliJ: Fejlesztőkörnyezet főként a Java programozási nyelvhez

Eclipse: Fejlesztőkörnyezet főként a Java programozási nyelvhez

NetBeans: Fejlesztőkörnyezet főként a Java programozási nyelvhez

.jar: A Java programozási nyelv által használt fájltípus; egy archívum, amiben egy adott program vagy programmodul futtatásához szükséges fájlok vannak.

Discord: Internetes szóbeli kommunikációra használható program.

Google Drive: Internetes konkurrens szerkesztést és verziókezelést is támogató dokumentumszerkesző program.

verziókezelő: A több verzióval rendelkező adatok kezelését végző szoftver

.doc: A specifikációkat és egyéb leadandókat tartalmazó dokumentum konténer fileformátum.

Win10: Windows 10, Microsoft által fejlesztett, egyik legelterjedtebb operációs rendszer

spawn: A pálya azon része, mezője, ahol az egyes a pályán megjelenő objektumok, mint karakterek vagy dobozok megjelennek a játék indításakor.

draw.io: webes szolgáltatás rajzok készítésére.

2.1.4. Hivatkozások

Szoftver projekt laboratórium - <https://www.iit.bme.hu/targyak/BMEVIIIA02>

2.1.5. Összefoglalás

A dokumentum további részeiben található:

- A szoftver áttekintése nagy vonalakban
- A szoftverhez kötődő követelmények
- Use-case-ek felsorolása
- Szótár, ami a szoftverrel kapcsolatos egyedi kifejezéseket gyűjti össze, és leírja a jelentésüket.
- A projekt kivitelezésének terve
- Projekt Napló

2.2. Áttekintés

Általános áttekintés

A fontosabb alrendszerek:

- Felhasználói felület
- Főmenü
- Játékmotor
- Pályageneráló

A felhasználók folyamatos, közvetlen kapcsolatban állnak a felhasználói felülettel. Ezen keresztül férhetnek hozzá a főmenühöz. A főmenüből indíthatják el a játékot a játékosok. Ekkor a Játékmotor kér egy pályát a Pályageneráló modultól, és elkezdődik a játék. A grafikus felületen keresztül a játékosok folyamatosan nyomon követhetik a játék alakulását.

A program nem igényel hálózati kapcsolatot, a pályát automatikusan generáljuk, bizonyos előre meghatározott korlátozások alkalmazásával így nincsen szükség adatokat tartalmazó fájlokra sem.

2.2.1. Funkciók

Csapatunk egy úgynevezett Szokóban nevezetű játék fejlesztését vállalta el. A szókoban (倉庫番 「そうこばん」, nyugaton sōkoban, „raktáros”) egy olyan fejtőrő játék, ahol a játékosnak egy felülnézetes labirintusban kell dobozokat tologatnia a helyükre. Esetünkben ez a játék úgynevezett "többjátékos" formájában fog szerepelni, ami azt jelenti, hogy a pályán nem egy, hanem több játékos tologatja a dobozokat körökre osztott formában. A körökre bontott játékmenet lehetővé teszi, hogy a játékosok egymást kijátszva, taktikus gondolkodást alkalmazva győzedelmeskedjenek a többi felett, emellett így a billentyűzet által nyújtott fizikai korlátozás is elkerülhető (nehéz lenne négyen egy billentyűzeten játszani).

A pálya mérete egy állandó mérettel van megadva amelynek szegélyét egy körbe futó fal elem sorozat reprezentál, benne a pálya elemek viszont véletlenszerűen, procedurálisan vannak generálva. Pálya elemek alatt a falakat, a mezőket, a célmezőket, a lánckat, alap lyukakat, kapcsolókat, az általuk kinyitott lyukakat, és azoknak kapcsolatait, pozíciót, továbbá a játékosokat értjük. minden pálya elemre szigorú funkcionális megkötések vonatkoznak.

A falak olyan elemek, amelyeken dobozokat áttolni nem lehet, és a játékosok sem képesek átmenni rajtuk, mondhatni ez a pálya elem számít a “legerősebbnek”. Általánosságban

elmondható, hogy ha egy játékos mellett valamilyen irányban fal van, akkor az abba az irányba nem léphet.

A mezők olyan elemek, amelyekre a dobozok tolthatók és amelyekre a játékosok is léphetnek. Ezeknek a kapcsolata alkotja a bejárható pályarészt.

A dobozok olyan elemek, amelyek a játékos által tolhatóak, viszont húzni nem lehet őket. Csak olyan irányban mozoghatnak, amilyen irányban a játékos is mozoghat (fel, le, jobbra, balra). A játékos képes arra is, hogy több lánát eltoljon egyszerre: abban az esetben ha a lánák egymás mellett vannak és akad elég hely az eltolásukra. A lánák emellett képesek a lyukakon leesni és megsemmisülni.

A célmezők olyan elemek, amelyekre ha egy dobozt rátolunk, a doboz eltűnik. Ekkor az a játékos aki a dobozt rátolta kap 1 db pontot.

A játék egyetlen, felhasználók által direktben irányított eleme a játékos. Ebből a játékban minimum 2, de akár maximum 4 is lehet. Képessége hogy tologathatja a dobozokat, aktiválhatja a kapcsolókat, mászkálhat a mezőkön. Mozgását tekintve csak 4 irányban mozoghat az égtájaknak megfelelően. Fel, le, jobbra, balra; az átlós mozgás nem engedélyezett. A játékosok egymással szorosabb kapcsolatba nem tudnak lépni, mivel egy mezőn egyszerre csak egy játékos lehet. Interakciójuk egyetlen formája az egymással szemben elkövetett szándékos emberölés (amit amúgy a törvény büntet), de ez egy játék, szóval ezt elnézzük.

A kapcsoló egy speciális mezőfajta, amelyre ha a játékos rátol egy lánát, megnyit egy lyukat a pálya egy random pontján. Ez akár olyan mező is lehet amelyen láná vagy egy másik játékos áll (sőt akár saját maga is!). Fontos tényező, hogy a kapcsoló mező kinézetre nem különbözik a sima mezőtől (így izgalmassabb a játékmenet, mert a játékosokat meglepetések érik). Ha a lánát letolják a kapcsolóról akkor a lyuk amit megnyitott bezárul.

A lyukat tartalmazó mező abban jellegzetes, hogy ami rákerül az beleesik és úgymond megszűnik létezni (játékos esetén persze, meghal, nullázódik az ideje). Ilyen mező létezik alapból is, de kapcsoló által is keletkezhet.

Ez mind vonatkozik a belső falakra, oszlopokra (egy fal elem), lánakra, kapcsolókra, a kapcsolók által kinyitott lyukra, és magukra a játékosokra is. Fontos kitétel, hogy a pályán nincsenek "generált" csapdák (úgy indul a játékos, hogy minden irányban, vagy legalábbis, északon, délen, keleten és nyugaton falak veszik körül).

A játékmenetet tekintve, minden játékosnak fix külön időt adunk (mint a sakkban), ezzel az idővel kell gázdálkodnia a körök alatt. Akinek az ideje lejárt, az többet nem kap lehetőséget a lépésre, ha meghal az ideje automatikusan nullázódik. A játék akkor ér véget ha mindenkinél az ideje lejárt. Ekkor kiértékelődnek a pontok.

A legtöbb ponttal rendelkező játékos lesz a nyertes. Döntetlenek is kialakulhatnak. A játékosok célja, hogy minél több lánát toljanak a cél mezőkre, és így minél több pontot össze tudjanak gyűjteni.

2.2.2. Felhasználók

A felhasználók többjátékos módban játszanak, körökre osztva, egymást váltva, tehát egyszerre csak egy játékos irányíthatja a karakterét. A játékban nem lesz pályaszerkesztő. A játék menetét nem lehet elmenteni, mindig újból kell kezdeni.

A játék főmenüjében található egy pár mondatos rövid leírás, mely a játékost megismerteti a játék működésével, szabályaival.

2.2.3. Korlátozások

A BME IIT szoftprojlab tárgyának oktatói (a megrendelők) által kiírt specifikáció megköveteli, hogy a program a HSZK gépein fusson. A gépeknek Win10 operációs rendszerrel kell rendelkezniük, amire telepítve van a JDK 1.7-es verziója (vagy bármelyik újabb verzió).

2.2.4. Feltételezések, kapcsolatok

Szoftver projekt labor feladatkiírás - <http://devil.iit.bme.hu/~balage/projlab/feladat.html>

2.3. Követelmények

2.3.1. Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
F01	A raktárépület padlója négyzetekre van osztva	kiértékelés	alapvető	BME-IIT	View Map	NxN-es mátrix képzi az alapját
F02	A padlón állnak a négyzetekkel megegyező alapterületű lágák	kiértékelés	alapvető	BME-IIT	View Map	
F03	A lágák eltolhatók	kiértékelés	alapvető	BME-IIT	Move Crates	észak-dél-kelet-nyugat irányba
F04	Egy mozgatással minden csak egy szomszédos négyzetre kerülhetnek a lágák	kiértékelés	alapvető	BME-IIT	Move Crates	
F05	A raktárban többen dolgoznak, a játékosok őket irányítják.	kiértékelés	alapvető	BME-IIT	Control Character	multiplayer

F06	A cél a ládák előírt helyre tologatása.	kiértékelés	alapvető	BME-IIT	Move Crates	
F07	A raktárnak falai és oszlopai is vannak	kiértékelés	fontos	BME-IIT	View Map	
F08	A falakon és oszlopokon a ládák nem tolhatók át	kiértékelés	alapvető	BME-IIT	Move Crates	
F09	A ládák egymást el tudják tolni	kiértékelés	alapvető	BME-IIT	Move Crates	
F10	Ha egy munkásra ládát tolunk, akkor a munkás automatikusan a szomszédos mezőre tolódik	kiértékelés	alapvető	BME-IIT	Move Crates, Control Character	
F11	Ha nem tud eltolódni (mert fal vagy láda van mellette) a munkás meghal	kiértékelés	alapvető	BME-IIT	Control Character	
F12	A ládák nem nyomhatók össze.	kiértékelés	fontos	BME-IIT	Move Crates	
F13	A padlón egyes helyeken lyukak találhatók	kiértékelés	alapvető	BME-IIT	View Map	
F14	A lyukakra ládát tolva a láda leesik (eltűnik).	kiértékelés	alapvető	BME-IIT	Move Crates	
F15	Ha munkás lép rá a lyukra, meghal	kiértékelés		BME-IIT	Control Character	
F16	Némelyik lyuk csak akkor viselkedik lyukként, ha egy kapcsolómezőn láda áll, egyébként padlónak tűnik	kiértékelés	alapvető	BME-IIT	View Map	

F17	A kapcsolómezőn ládának kell állnia, hogy kinyissa a lyukat,	kiértékelés	alapvető	BME-IIT	Control Switches	
F18	Ha munkás áll a kapcsolómezőre, akkor nem kapcsol	kiértékelés		BME-IIT	Control Switches	
F19	A játék véget ér, ha az összes láda a helyén van, vagy ha már nem lehet többet tolni.	kiértékelés	fontos	BME-IIT	Move Crates	
F20	Az nyer, aki ekkorra a legtöbb ládat a helyére tolta.	kiértékelés	fontos	BME-IIT	Control Character, Move Crates	
F21	A padlón állnak a négyzetekkel megegyező alapterületű emberek	kiértékelés	fontos	csapat	View Map, Control Character	
F22	A padlón állnak a négyzetekkel megegyező alapterületű oszlopok	kiértékelés	fontos	csapat	View map	
F23	A padlón vannak a négyzetekkel megegyező alapterületű lyukak	kiértékelés	fontos	csapat	View Map	
F24	A padlón állnak a négyzetekkel megegyező alapterületű kapcsolók	kiértékelés	fontos	csapat	View Map	
F25	A kapcsolók láthatatlanok, ha nincs rajtuk láda	kiértékelés	opcionális	csapat	View Map	

2.3.2. Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
E01	Git	nincs	alapvető	csapat	Verziókezelő, forráskód
E02	Github account	nincs	alapvető	csapat	Git tárhely, forráskód
E03	JRE	bemutatás	alapvető	megrendelő	1.7++
E04	Eclipse	nincs	opcionális	csapat	Java IDE
E05	IntelliJ IDEA	nincs	opcionális	csapat	Java IDE
E06	NetBeans IDE	nincs	opcionális	csapat	Java IDE
E07	HSZK-ban találhatóakkal azonos vagy jobb PC	bemutatás	alapvető	megrendelő	Sandy Bridge Pentium++, 4GB RAM
E08	Monitor	nincs	alapvető	megrendelő	1280*720++
E09	Egér	nincs	alapvető	megrendelő	
E10	Billentyűzet	nincs	alapvető	megrendelő	
E11	Google Drive	nincs	alapvető	csapat	Dokumentáció
E12	Draw.io	nincs	alapvető	csapat	UML, use-case diagramhoz

2.3.3. Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
A01	Követelmény, projekt, funkcionálitás	beadás	alapvető	csapat	február 19.
A02	Analízis modell kidolgozása 1.	beadás	alapvető	csapat	február 26.
A03	Analízis modell kidolgozása 2.	beadás	alapvető	csapat	március 5.
A04	Szkeleton tervezése	beadás	alapvető	csapat	március 12.
A05	Szkeleton	beadás	alapvető	csapat	március 19.
A06	Prototípus koncepciója	beadás	alapvető	csapat	március 26.
A07	Részletes tervezek	beadás	alapvető	csapat	április 9.
A08	Prototípus	beadás	alapvető	csapat	április 23.
A09	Grafikus felület specifikációja	beadás	alapvető	csapat	május 2.
A10	Grafikus változat	beadás	alapvető	csapat	május 14.
A11	Összefoglalás	bemutatás	alapvető	csapat	május 18.

2.3.4. Egyéb nem funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
NF01	Futtatható .jar állomány	bemutatás	opcionális	csapat	bemutatáskor
NF01	Egyéb tesztelési feladatok	bemutatás	fontos	csapat, BME-IIT	bemutatáskor

2.4. Követelmények

2.4.1. Use-case leírások

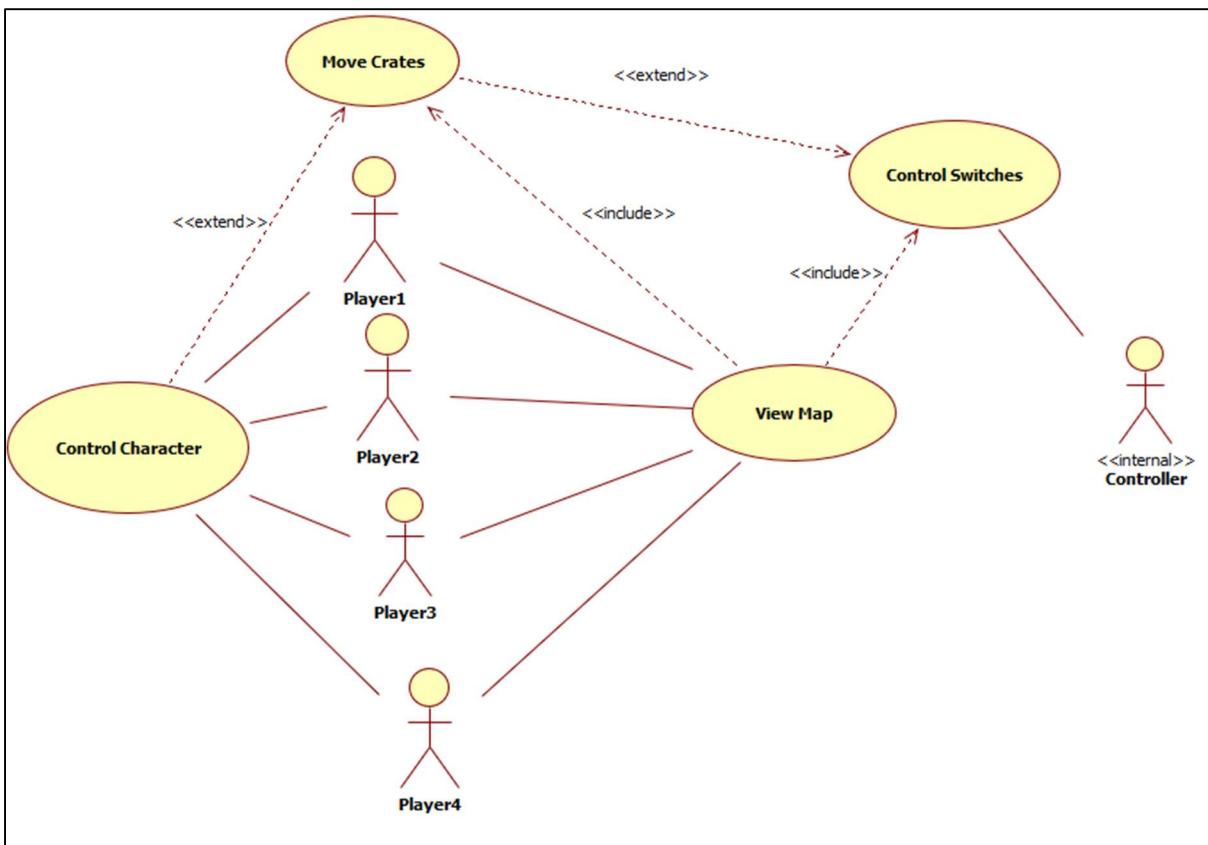
Use-case neve	View Map
Rövid leírás	A játékos megtekinti a pályát.
Aktorok	Player
Forgatókönyv	A rendszer kirajzolja a pálya aktuális állapotát. A játékos megtekinti a pálya aktuális állapotát.

Use-case neve	Move Crates
Rövid leírás	A játékos tologatja a lánkat a pályán.
Aktorok	Player
Forgatókönyv	Ha a játékos nekimegy egy lánaknak, és az tud tovább mozogni, azaz nem ütközik falnak, vagy esik le a lyukba, akkor az a következő mezőre kerül. Lának tolhatják egymást tovább.

Use-case neve	Control Character
Rövid leírás	A játékos irányítja a karakterét a pályán.
Aktorok	Player
Forgatókönyv	Az játékos jobbra, balra, felfele vagy lefele mozoghat.

Use-case neve	Control Switches
Rövid leírás	A lyukakat nyitja és csukja.
Aktorok	Controller
Forgatókönyv	A játékos ha kapcsolómezőre tol egy lánát, annak hatására a program be vagy kikapcsolja a lyukakat a pályán.

2.4.2. Use-case diagram



2.5. Szótár

Character - Akit a játékos (ember) irányít.

Control - Irányítása egy a játékban megjelenő objektumnak, elemnek, mint láda, karakter, és további elemek.

Controller - A gépi oldala az alkalmazásnak, vezérli a gépi eseményeket.

Crate - Alternatív név a dobozra, olyan pályán található objektum, amely a játékosok által mozgatható a mezőkön.

Csapat - A leadandó szoftver projektet elkészítő emberek halmaza.

Csapda - Alternatív név a lyukra. Olyan mező, amelyre ha egy doboz vagy egy játékos lép, akkor a delikvens rendre megsemmisül, vagy meghal.

Doboz - Pályán található objektum, amely a játékosok által mozgatható a mezőkön.

Fal - Olyan mező melyre nem lehet lépni, és nem lehet rátolni, áttolni rajta objektumokat, határolóelem.

Játékos - Ember, aki a karaktereket irányítja.

Kapcsolómező - Olyan mező, amelyre egy dobozt rátolva aktiválódik, és egy lyukat hoz létre.

Karakter (leírásban munkás) - Az a szereplő akit a játékos irányít.

Láda - Alternatív név a dobozra, olyan pályán található objektum, amely a játékosok által mozgatható a mezőkön.

Lyuk - Olyan mező amelyre ha egy doboz vagy egy játékos rálép, megsemmisül, meghal.

Map - Alternatív név a raktárépületre, a pálya, amin a karaktereket irányítani lehet, a játék színtere.

Mező - A pálya, a raktárépület színtere elemi építőeleme. Mezők NxN-es táblázata képzi a pályát.

Oszlop - 1x1 méretű mozdíthatatlan és áthatolhatatlan fal szakasz.

Padló - Alternatív név az üres mezőre, ahova a játékos léphet és dobozt is tolhat rá, továbbá tartalmazhat rejtett állapotban csapdát (lyukat), vagy kapcsolómezőt.

Pálya - Alternatív név a raktárépületre, a pálya, amin a karaktereket irányítani lehet, a játék színtere.

Player - Alternatív név a játékosra. A szereplő, aki a karaktert irányítja.

Raktárépület - A játék színtere, maga a pálya, amin játszani lehet. Fix méretű, kívülről áthatolhatatlan falakkal van határolva.

Switch - Alternatív név a kapcsolómezőre. Olyan mező, amelyre egy dobozt rátolva aktiválódik, és egy lyukat hoz létre.

Térkép - Alternatív név a raktárépületre, a pálya, amin a karaktereket irányítani lehet, a játék színtere.

2.6. Projekt terv

2.6.1. Csapat

A csapatunk 5 főből áll, név szerint:

- Bende Zoltán
- Szatmáry Péter
- Máté László
- Králik Bálint
- Kapitány Gergő

A csapat közösen dolgozik a feladatot, nincsenek különböző felelősségek kiosztva az egyes csapattagoknak, az aktuális munkához igazítjuk, hogy kinek mi a feladata.

2.6.2. Kommunikáció

Verziókezelés: A forráskód megosztására a Git verziókezelő szoftvert használjuk, ami a dokumentációt illeti, a Google cég webes szolgáltatását, a Google Drive-ot használjuk, ily módon egyszerre többen is tudjuk szerkeszteni a dokumentációt.

Facebook: Személyes találkozások, vagy online munkafolyamatok időpontjainak megbeszélésére használjuk elsősorban.

Megbeszélések: Ha szükségesnek találjuk, akkor egy héten egyszer, vagy akár többször is összeülnünk és megvitatjuk a szoftverrel kapcsolatban előjött főbb kérdéseket, gondolatokat.

Discord: Interneten keresztül, mikrofon segítségével hatékonyan meg tudjuk beszélni a szoftverfejlesztéssel kapcsolatos főbb kérdéseket, illetve az üzenőfalon meg tudjuk osztani a fontosabb fájlokat a programon keresztül.

2.6.3. Programok

Verziókezelés: A Git verziókezelő programot használjuk.

Dokumentáció: A dokumentációhoz a Google Drive-ot, és Microsoft Word, illetve Adobe PDF dokumentformátumokat használunk.

Fejlesztőkörnyezet: A szoftvert egyéni preferencia függvényében Eclipse-szel, NetBeans-szel, illetve IntelliJ IDEA fejlesztőkörnyezet segítségével fejlesztjük.

UML Diagram: A szoftverhez tartozó uml diagrammokat a draw.io webes rajzolóprogram segítségével hozzuk létre.

2.6.4. Mérföldkövek, határidők

Határidő	Termék
febr. 19.	02. Követelmény, projekt, funkcionalitás
febr. 26.	03. Analízis modell kidolgozása 1.
márc. 5.	04. Analízis modell kidolgozása 2.
márc. 12.	05. Szkeleton tervezése
márc. 19.	06. Szkeleton
márc. 26.	07. Prototípus koncepciója
ápr. 9.	08. Részletes tervezések
ápr. 23.	10. Prototípus
máj. 2.	11. Grafikus felület specifikációja
máj. 14.	13. Grafikus változat
máj. 18.	14. Összefoglalás

A feladat 3 fő lépésből áll, amik a következők:

A **szkeleton** változat célja annak bizonyítása, hogy az objektum és dinamikus modellek a definiált feladat egy modelljét alkotják. A szkeleton egy program, amelyben már valamennyi, a végső rendszerben is szereplő business objektum szerepel. Az objektumoknak csak az interfésze definiált. Valamennyi metódus az indulás pillanatában az ernyőre szöveges változatban kiirja a saját nevét, majd meghívja azon metódusokat, amelyeket a szolgáltatás végrehajtása érdekében meg kell hívnia. Amennyiben a metódusból valamely feltétel fennállása esetén hívunk meg más metódusokat, akkor a feltételre vonatkozó kérdést interaktívan az ernyőn fel kell tenni és a kapott válasz alapján kell a továbbiakban eljární.

A szkeletonnak alkalmasnak kell lenni arra, hogy a különböző forgatókönyvek és szekvencia diagramok ellenőrizhetők legyenek. Csak karakteres ernalókezelés fogadható el, mert ez biztosítja a rendszer egyszerűségét.

A **prototípus** program célja annak demonstrálása, hogy a program elkészült, helyesen működik, valamennyi feladatát teljesíti. A prototípus változat egy elkészült program kivéve a kifejlett grafikus interfészét. A változat tervezési szempontból elkészült, az ütemezés, az aktív objektumok kezelése megoldott. A business objektumok - a megjelenítésre vonatkozó részeket kivéve - valamennyi metódusa a végeges algoritmusokat tartalmazza. A megjelenítés és működtetés egy alfanumerikus ernalón követhető, ugyanakkor a megjelenítés fájlban is logolható, ezzel megteremtve a rendszer tesztelésének lehetőségét. Különös figyelmet kell fordítani az interfész logikájára, felépítésére, valamint arra, hogy az mennyiben tükrözi és teszi láthatóvá a program működését, a beavatkozások hatásait.

A **grafikus** változat csak egy grafikus felhasználói felületben különbözik a prototípustól, a játék logikája nem változik.

2.7. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.02.17. 12:10	2 óra	Bende Szatmáry Máté Králik Kapitány	Feladat áttekintő megbeszélése, részfeladatok kiosztása a csapattagok számára
2018.02.17 14:30	3 óra	Bende	2.6 Projekt terv, 2.3.2 Erőforrásokkal kapcsolatos követelmények
2018.02.17 14:30	3 óra	Szatmáry	2.2 Áttekintés 2.4.1 Use-case leírások 2.5 Szótár
2018.02.17 14:30	3 óra	Máté	2.3 Követelmények 2.4.2 Use-case diagram
2018.02.17 14:30	3 óra	Králik	2.2.2 Funkciók 2.3 Követelmények
2018.02.17 14:30	1 óra	Kapitány	2.1 Bevezetés
2018.02.18 14:30	2 óra	Bende Szatmáry Máté Králik	Követelmény, projekt, funkcionális beadandó - utolsó simítások
2018.02.18 17:00	30 perc	Bende Szatmáry Máté Králik Kapitány	2.7 Napló elkészítése értekezletként, korrektúrázás

3. Analízis modell kidolgozása

3.1. Objektum katalógus

3.1.1. Box

A pályán található dobozokért felelős osztály. A játékosok tudják őket egy karakter vagy egy másik doboz nekitolásával mozgatni a Direction által meghatározott irányokba. Ezeket kell a játékosnak megfelelő helyre tolnia.

3.1.2. Character

A játékosokért felelős osztály, a pályán található játékosokhoz tartozó objektum. A karakterek azok a játékelemek, amiket a játékos közvetlenül tud irányítani.

3.1.3. Direction

Felsorolás, a pályán megkülönböztetett irányokat jelöli, amely irányokba a karakterek vagy dobozok mozognak.

3.1.4. Field

A Map osztályban található mezők. Azok a négyzetek, amik a pályát alkotják. Ezek a mezők lehetnek Goal, NormalField, Trap vagy Switch típusúak.

3.1.5. Game

A játék indításáért és befejezéséért felelős osztály. Tartalmazza a játékhoz kialakított pályát.

3.1.6. Goal

Cél mező, ha látótolunk rá, pontot kap az a játékos, amelyik rátolta, a rátolt látta pedig eltűnik a pályáról.

3.1.7. Map

A pálya, amin a játék zajlik, mezők (Field) nxn méretű táblájából áll, amin megjelenhetnek falak, dobozok, csapdák, kapcsolók, karakterek, és üres mezők (NormalField).

3.1.8. NormalField

Üres mező, aminek semmi különleges szerepe nincs, azaz nem történik semmilyen extra esemény, ha rákerül egy karakter vagy egy doboz.

3.1.9. Switch

Kapcsoló mező, ez hozzá működésbe a hozzá tartozó csapdát ha látótolunk rá. Játékostól nem aktiválódik.

3.1.10. Thing

Gyűjtőfogalom a pályán található tárgyakra, mint karakter, fal, vagy doboz.

3.1.11. Timer

Az egyes játékosok külön számlálóval rendelkeznek, ami a hátralevő (lépés előtti gondolkozásra felhasználható) idejüket tartja nyilván. A Timer osztály egy példánya ezeket az időket csökkenti másodpercenként az éppen aktuális játékosnak. Ha a játékos ideje elfogy, akkor többet már nem léphet, viszont a játékot ettől függetlenül még megnyerheti.

3.1.12. Trap

Csapda mező, ha az éppen aktuális Trap-hoz tartozó Switch osztály egy példánya bekapsolja, akkor megsemmisíti a rajta lévő Box vagy Character elemet.

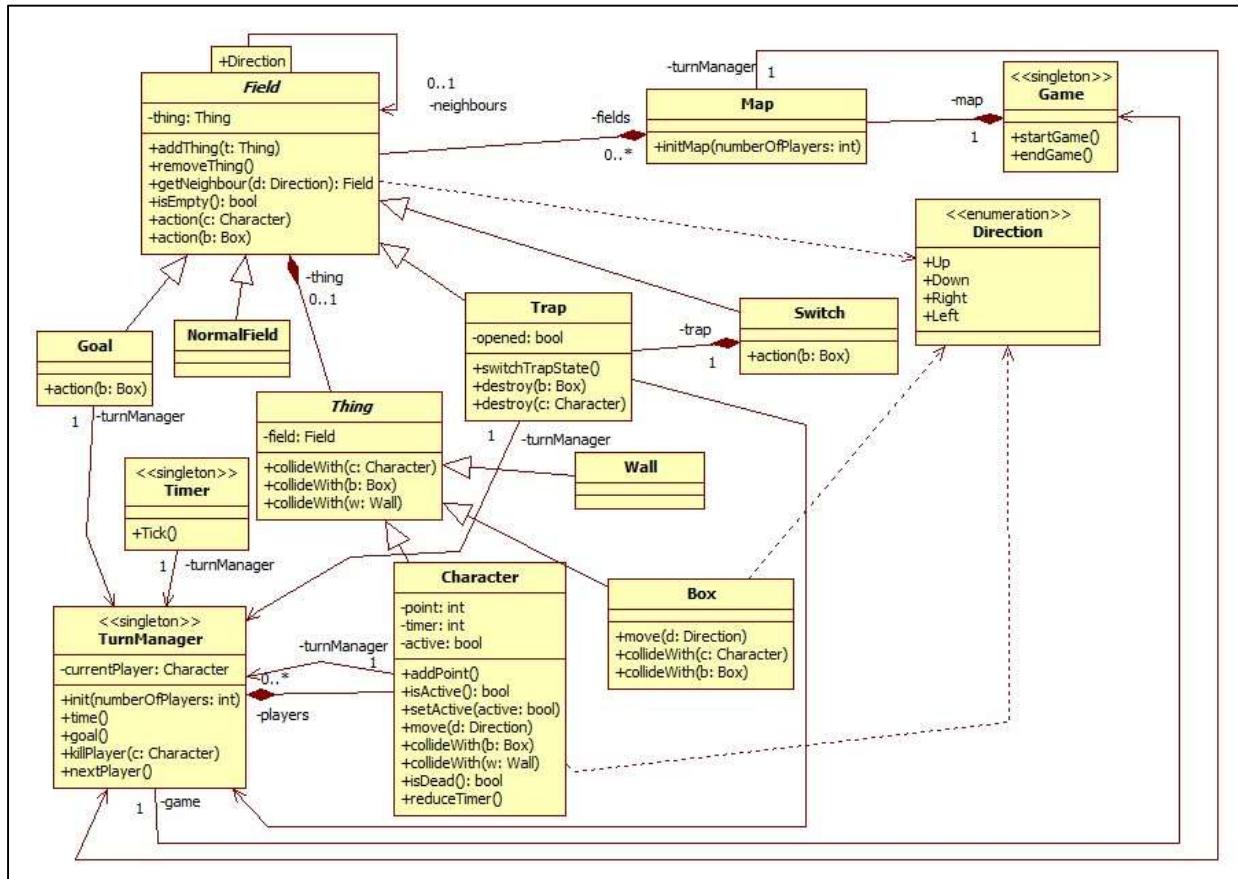
3.1.13. TurnManager

Megkapja a játékosok számát, majd menedzseli a játékmenetet, azaz, hogy melyik játékosnak a köre következik, ki az aktuális játékos, annak mennyi ideje van hátra, továbbá pontokat oszt ki a feladatok végrehajtásával.

3.1.14. Wall

A játékban található falakat modellező osztály. Abban különbözik a pályán lévő többi elemtől (Thing), hogy semmilyen módon nem elmozdítható. A fal irányába nem léphet a játékos, ha azt teszi, az is lépésnek számít, de a karakter nem fog mozdulni, így effektíve kimerít a körből az adott játékos.

3.2. Statikus struktúra diagramok



3.3. Osztályok leírása

3.3.1. Box

- **Felelősség**

A pályán a Map initMap függvénye által létrehozott dobozokat jelképző objektumok. Ezeket tologathatja a játékosok szabadon a pályán. Ha egy játékos egy dobozt egy célmezőre tol, akkor pontot kap érte, a doboz pedig eltűnik. A dobozok le is eshetnek a lyukakba, de más módon nem tűnhetnek el a pályáról, nem nyomhatják össze egymást, ugyanakkor egymást amennyiben a pályán van szabad hely - arrébb tolhatják a következő mezőre, ugyanúgy ahogy karaktert is tovább tolthatnak, falhoz nyomva (üres következő mező hiányában) meg is ölhetik azt. Ez bármennyi sorban lévő doboz esetén rekurzívan is érvényes.

- **Ősosztályok**

Thing

- **Metódusok**

void move(Direction d): Eltolja a dobozt egy másik mezőre.

void collideWith(b: Box): Ha másik dobozzal ütközik, tovább tudja azt tolni.

void collideWith(c: Character): Miközben őt tolják (mint dobozt), és neki megy egy másik Characternek, azt el tudja tolni.

3.3.2. Character

- **Felelősség**

A játékos által irányított karakter. 2 és 4 darab között lehet a játékosok száma. minden játékosnak minden körben pontosan egy darab gomblenyomásra van lehetősége, ha a játékos mozdíthatatlan pályaelem, mint fal, oszlop, vagy ezek mellett lévő doboz felé mozgatná a karaktert, a rendszer azt is lépésnek veszi, tehát a játékos az adott körben a lépését elhasználta, ennek ellenére nem mozgott a pályán.

- **Ősosztályok**

Thing

- **Attribútumok**

int point: A játékos aktuális pontszáma.

int timer: A játékos még hátralévő ideje.

bool active: Jelzi, hogy a játékos éppen aktív-e.

- **Metódusok**

void addPoint(): Hozzáad egy pontot a játékos pontszámához.

bool isActive(): Visszatér egy igaz/hamis értékkel, hogy a játékos éppen aktív-e.

void setActive(bool activate): Aktiválja vagy deaktiválja a játékosot, függően attól, hogy ő van e soron.

void collideWith(b: Box): Doboznak próbál menni, megpróbálja eltolni.

void collideWith(w: Wall): Ha falnak ütközik, megnézi, hogy ő az aktív játékos-e, ha nem, akkor meghal.

bool isDead(): Visszatér egy igaz/hamis értékkel, hogy meghalt-e már valamilyen okból a karakter.

void reduceTimer(): Megadott időegységgel csökkenti a játékos idejét.

3.3.3. Direction

- **Felelősség**

A pályára vonatkozó közlekedési irányok enumerációját testesíti meg.

- **Attribútumok**

Up: Fel.

Down: Le.

Left: Balra.

Right: Jobbra.

3.3.4. Field

- **Felelősség**

A Map osztályban található mezőknek az absztrakt alaposztálya. Összefoglaló osztály, ezek tömbjét tárolja a Map osztály egy adattagjában.

- **Attribútumok**

Thing thing: Az a Thing “dolog”, ami az adott pillanatban ezen a mezőn rajta van. Lehet ez doboz, fal, vagy játékos.

- **Metódusok**

addThing(t: Thing): Valami vagy valaki hozzáadása a mezőhöz (“ráhelyezzük”).

removeThing(): Akkor hívódik meg, ha egy Character ellép, vagy egy Box eltolódik az adott Field mezőről.

Field getNeighbour(d: Direction): A paraméterként kapott irányba található szomszédját adja vissza.

bool isEmpty(): Visszaadja, hogy üres-e a mező (van-e rajta valami/valaki).

action(c: Character): A mezőn lévő karakter esetén végrehajtandó művelet, a leszármazottakban felülírjuk a megfelelő működéssel.

action(b: Box): A mezőn lévő doboz esetén végrehajtandó művelet, a leszármazottakban felülírjuk a megfelelő működéssel.

3.3.5. Game

- **Felelősség**

A pálya nyilvántartásáért és a játékmenet legfőbb funkcióiért (elindítása vagy lezárása) felelős. Meghívja további osztályok inicializáló függvényeit.

- **Attribútumok**

Map map: Számoltartja a játékhoz tartozó pályát.

- **Metódusok**

void startGame(): A teljes játék elindításáért felelős. Bekéri a felhasználótól a játékosok számát, majd meghívja a Map osztály initMap függvényét.

void endGame(): A játék lezárásáért felelős. Akkor fut le, amikor minden játékosnak elfogy a saját ideje. Ilyenkor összegzi az eredményeket, kiírja a győztest, vagy győzteseket és visszalép a játék főmenüjébe.

3.3.6. Goal

- **Felelősség**

A pályának azon, megkülönböztetett színnel jelzett mezői, ahova a játékosoknak dobozokat kell tologatniuk, hogy pontokat kapjanak. A játék célja, hogy adott játékos minél több célmezőre, minél több doboz toljon.

- **Ősosztályok**

Field

- **Attribútumok**

TurnManager turnManager: referencia a TurnManagerre, hogy tudja jelezni neki, ha pontot kell adni az aktuális játékosnak.

- **Metódusok**

void action(b: Box): Ha dobozt tolunk rá, akkor jön csak működésbe, az ősosztály ezen metódusát felüldefiniálva oldottuk meg. Ez ad pontot az aktuális játékosnak.

3.3.7. Map

- **Felelősség**

A pálya, amin a játék zajlik. Mezők nxn méretű mátrixából áll, eltárolja azokat. A pályát ő maga generálja le megadott szabályoknak és kikötéseknek megfelelően.

- **Attribútumok**

Field[][] fields: A pálya mátrixának elemeit tartalmazó tömb. A Field absztrakt osztály típusának mezőit tartalmazza, azaz Goal, NormalField, Trap, és Switch objektumok kerülhetnek bele - ezek a pálya alapelemei.

- **Metódusok**

void initMap(numberOfPlayers int): A pályát létrehozó függvény, paraméterül kapja a Game singleton-tól az elindítani kívánt játékban résztvevő játékosok számát, amiket aztán tovább is ad a TurnManager osztálynak, ami azt az init() függvényében használ fel.

3.3.8. NormalField

- **Felelősség**

A pályán található sima üres mezőkért felelős osztály. A játékosok és a dobozok ezeken tudnak szabadon közlekedni, az erre való rálépéskor nem történik semmilyen speciális esemény.

- **Ősosztályok**

Field

3.3.9. Switch

- **Felelősség**

A pályán megtalálható láthatatlan (úgy néz ki mint egy sima mező) kapcsoló, amelyhez pontosan egy csapda objektum tartozik.. Akkor aktiválódik, ha egy dobozt tolnak rá, játékosként rálépve nem történik semmi.

- **Ősosztályok**

Field

- **Attribútumok**

Trap trap: A kapcsolóhoz tartozó csapda.

- **Metódusok**

action(Box b): Ha dobozt kerül a mezőre, akkor aktiválja a hozzá tartozó csapdát.

3.3.10. Thing

- **Felelősség**

Ez az osztály felelős a mezőkön megtalálható “dolgokért”. Ilyen lehet például fal (Wall), doboz (Box) és játékos (Character). Esetleges ütközések esetén meghatározza, hogy milyen cselekmény fog végbe menni.

- **Attribútumok**

Field field: Az a mező amelyen a dolog található.

- **Metódusok**

void collideWith(Character c): Ha egy adott dolog egy játékossal ütközik, kezeli azt.

Kimenetele függ, hogy milyen dolog ütközik a játékossal.

void collideWith(Box b): Ha egy adott dolog egy dobozzal ütközik, kezeli azt. Kimenetele függ, hogy milyen dolog ütközik a dobozzal.

void collideWith(Wall w): Ha egy adott dolog egy fallal ütközik, kezeli azt. Kimenetele függ, hogy milyen dolog ütközik a fallal.

3.3.11. Timer

- **Felelősség**

A számítógép belső óráját kérdezi le, és értesíti a TurnManager osztályt arról, hogy eltelt egy másodperc.

- **Attribútumok**

TurnManager turnManager: Egy referencia a TurnManager singleton osztályra, amelyen keresztül eléri annak a time() függvényét, és inkrementálja a játékosok megfelelő adattagjait.

- **Metódusok**

void Tick(): Végtelen ciklusban fut, lekérdezi a számítógép idejét, és ha eltelik egy megadott időegység, akkor meghívja a TurnManager time() függvényét, hogy csökkentse az aktuális játékos idejét.

3.3.12. Trap

- **Felelősség**

A pályán megtalálható csapda más néven "lyuk" objektumokért felelős osztály. Ha egy játékos vagy doboz belelép az megsemmisül/meghal. minden switchhez pontosan egy csapda tartozik.

- **Ősosztályok**

Field

- **Attribútumok**

bool opened: Jelzi a csapda jelenlegi állapotát (aktivált, deaktivált).

- **Metódusok**

void switchTrapState(): Aktiválja, deaktiválja magát.

destroy(Box b): Megsemmisíti a rákerülő Box objektumot, ha a csapda aktiválva van.

destroy(Character c): Megsemmisíti a rákerülő Character objektumot, ha a csapda aktiválva van.

3.3.13. TurnManager

- **Felelősség**

Megkapja a játékosok számát, majd menedzseli a játékmenetet, azaz, hogy melyik játékosnak a köre következik, ki és ki az aktuális játékos, aki az adott körben léphet.

A Timer fix időközönként (mp) inkrementálja a TurnManager által tárolt, és éppen aktív karakterek a hátralévő idejét a time() függvénye segítségével.

- **Attribútumok**

Character players: A játékban résztvevő karaktereket tárolja.

Character currentPlayer: Az éppen aktuális karakterre mutató referencia.

- **Metódusok**

void init(int numberOfPlayers): Kezdéskor létrehozza a kollekciót a játékosokra.

void time(): Amikor a Timer jelez neki, levon az aktuális játékos idejéből.

void goal(): Amikor a Goal jelez neki, pontot ad az aktuális játékosnak.

void killPlayer(c: Character): Ha egy játékosnak elfogy az ideje, vagy csapdába esik, meghal, ez a függvény valósítja ezt meg.

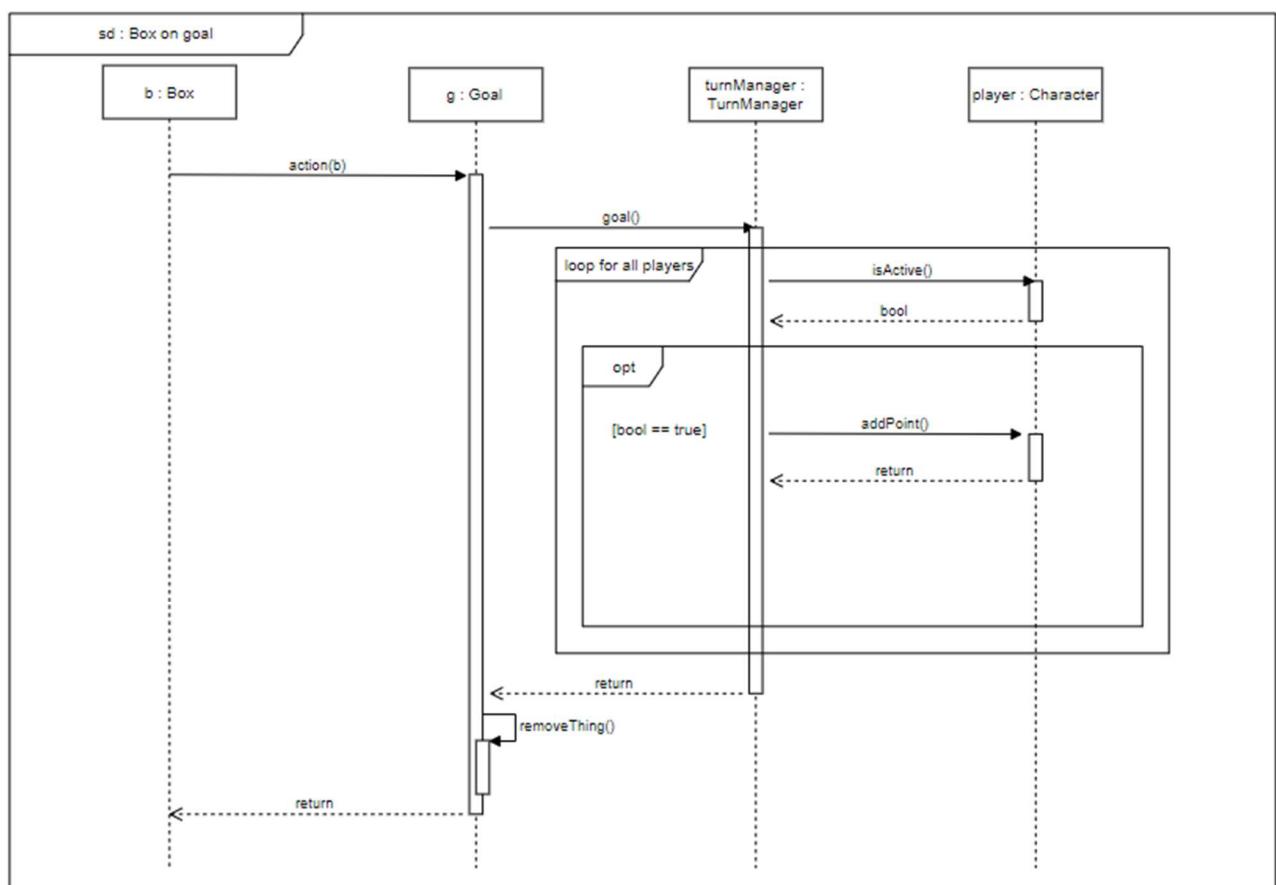
3.3.14. Wall

- **Felelősség**

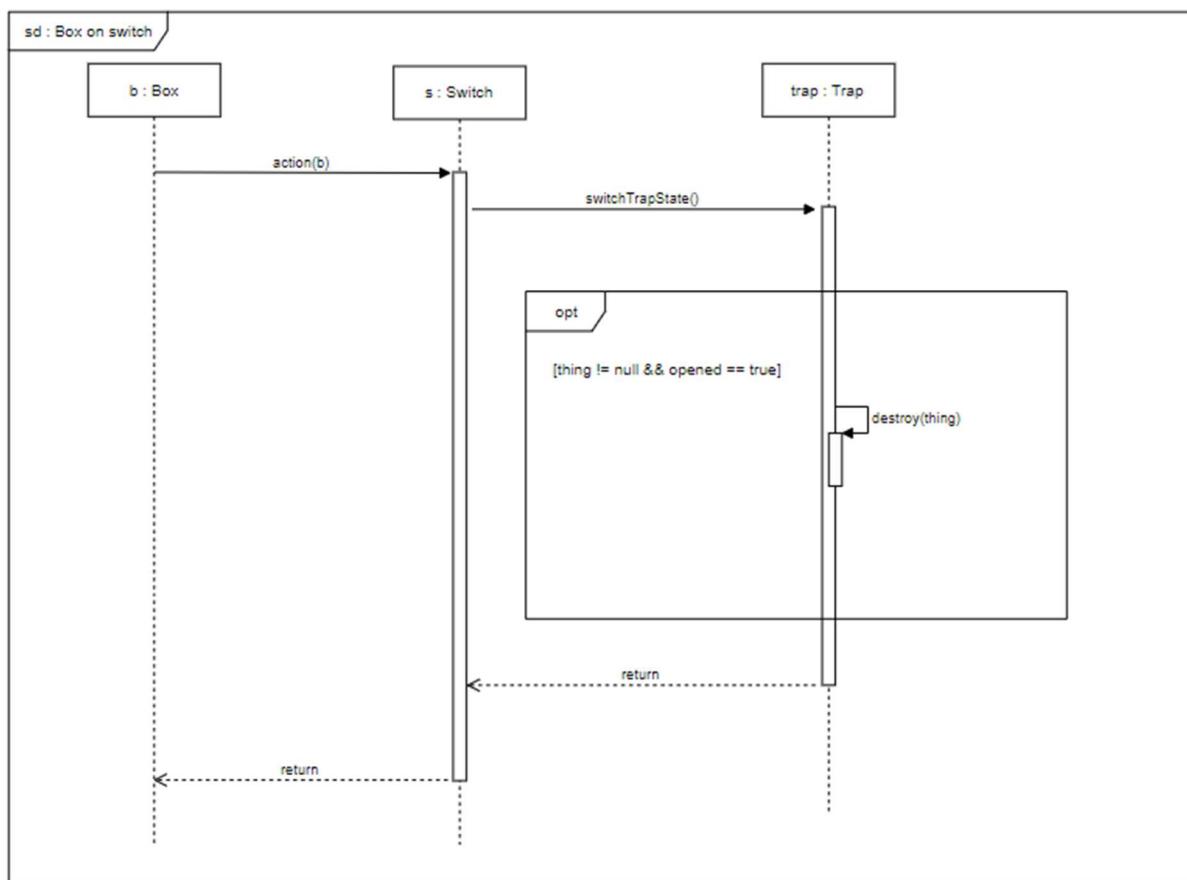
A pályán megtalálható mozdíthatatlan, és áthatolhatatlan fal objektumokért felelős. Ha egy falszakasz nem több, egymással összefüggő komponensből áll (1×1 mező méretű), akkor oszlop, máskülönben fal. Ha magától felnak menne a játékos által irányított karakter, akkor beleütközik, nem történik semmi. Hasonlóan a dobozzal is, viszont amennyiben egy karakter, vagy egy doboz tol egy másik karaktert felnak, az meghal.

3.4. Szekvencia diagramok

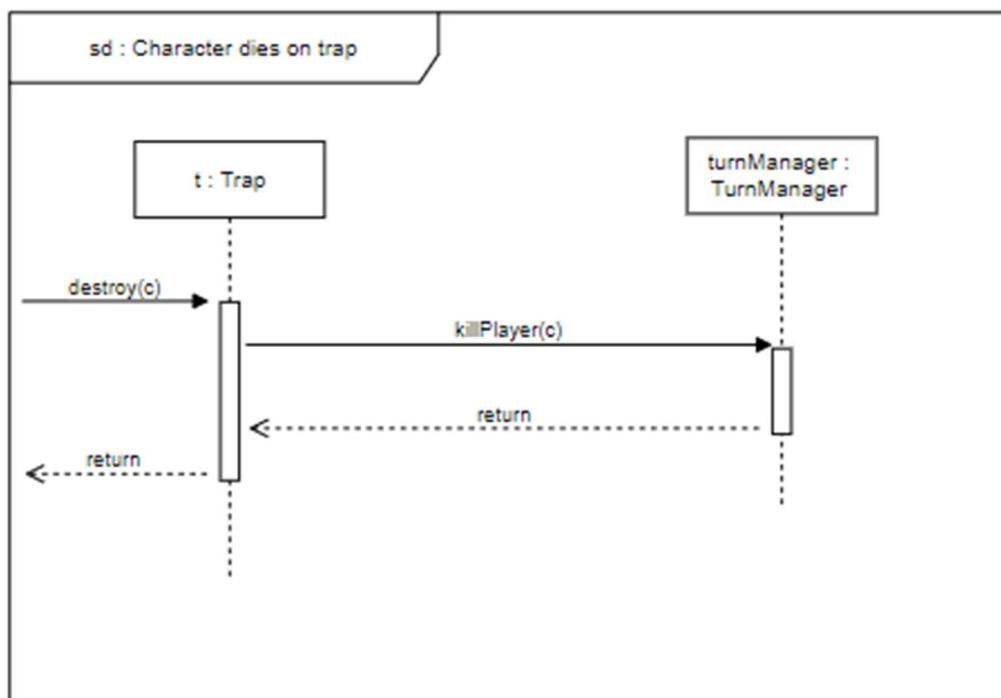
3.4.1. Box on Goal



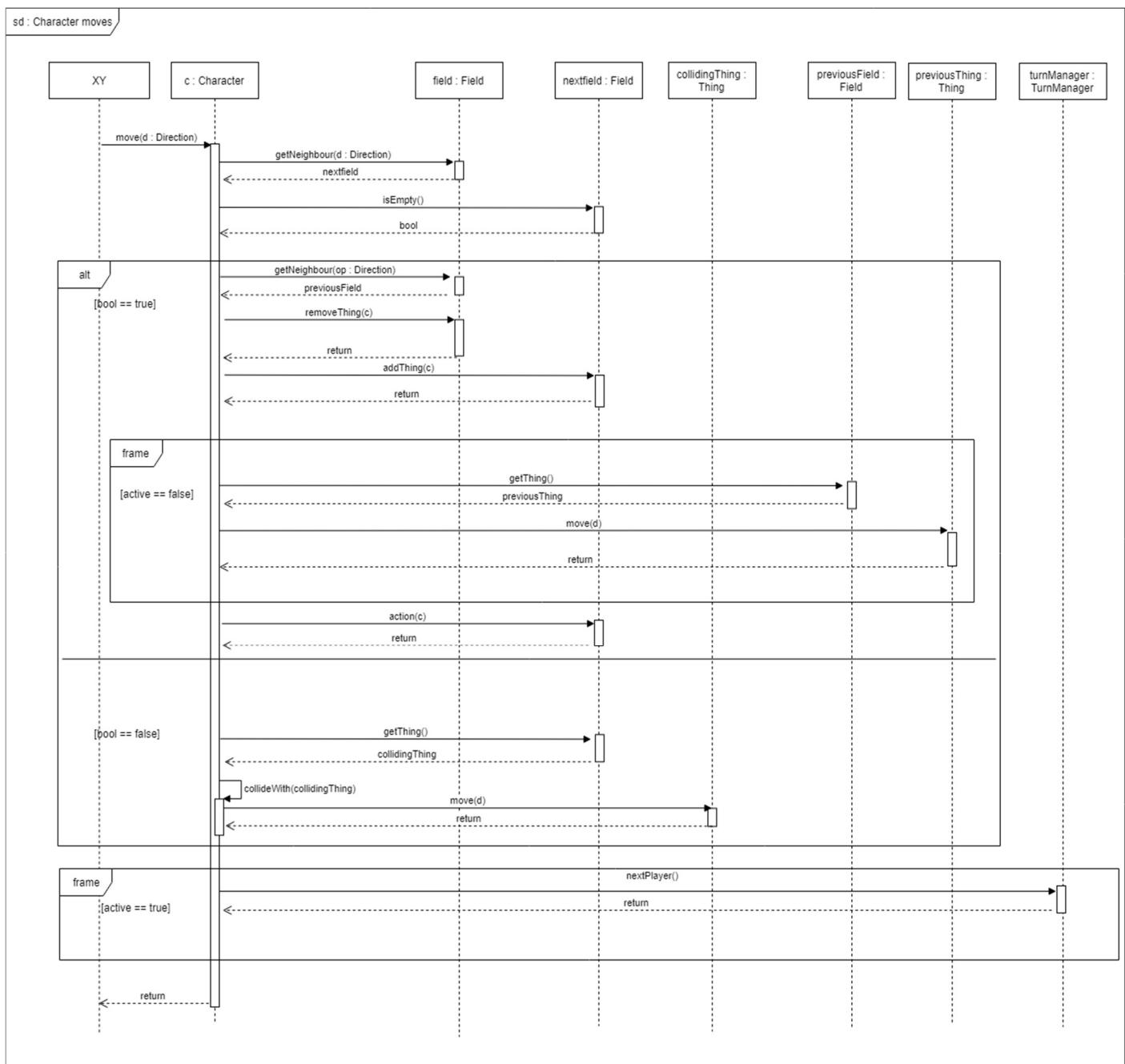
3.4.2. Box on Switch



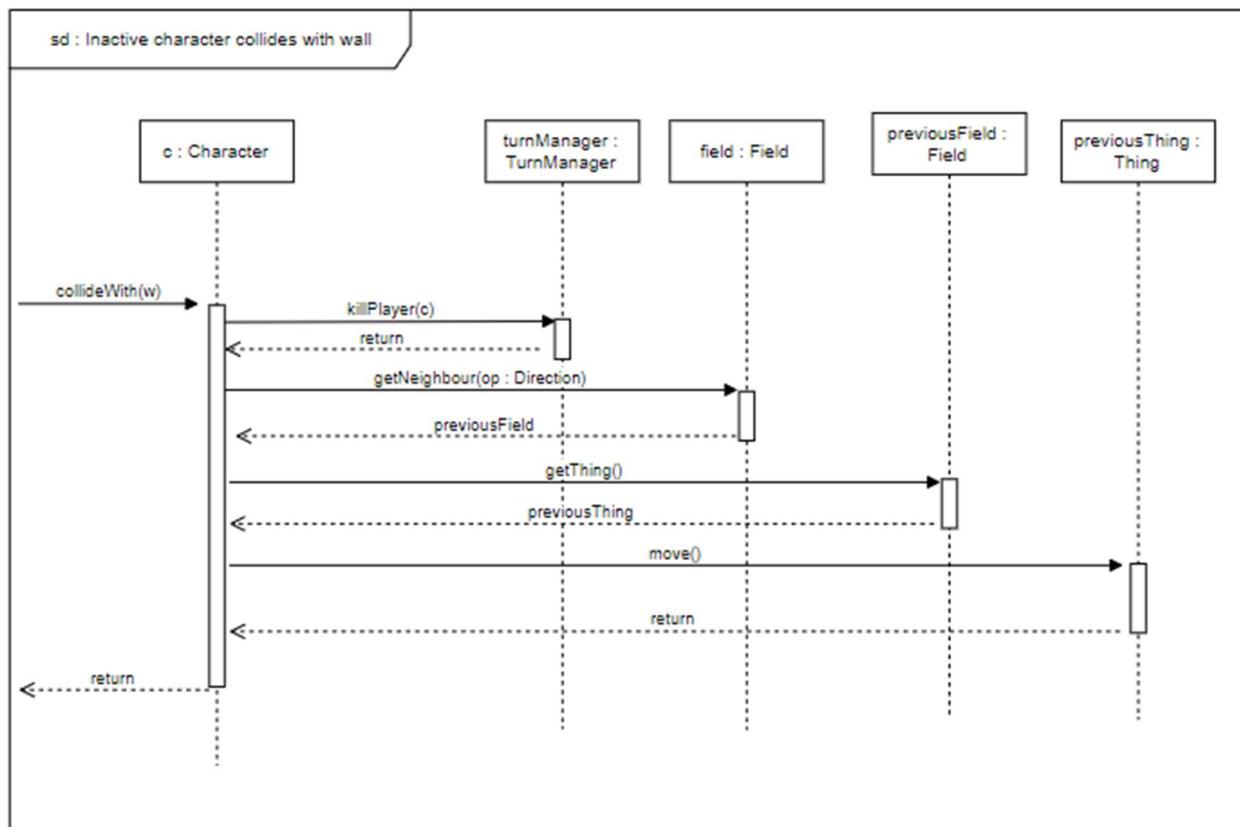
3.4.3. Character dies on trap



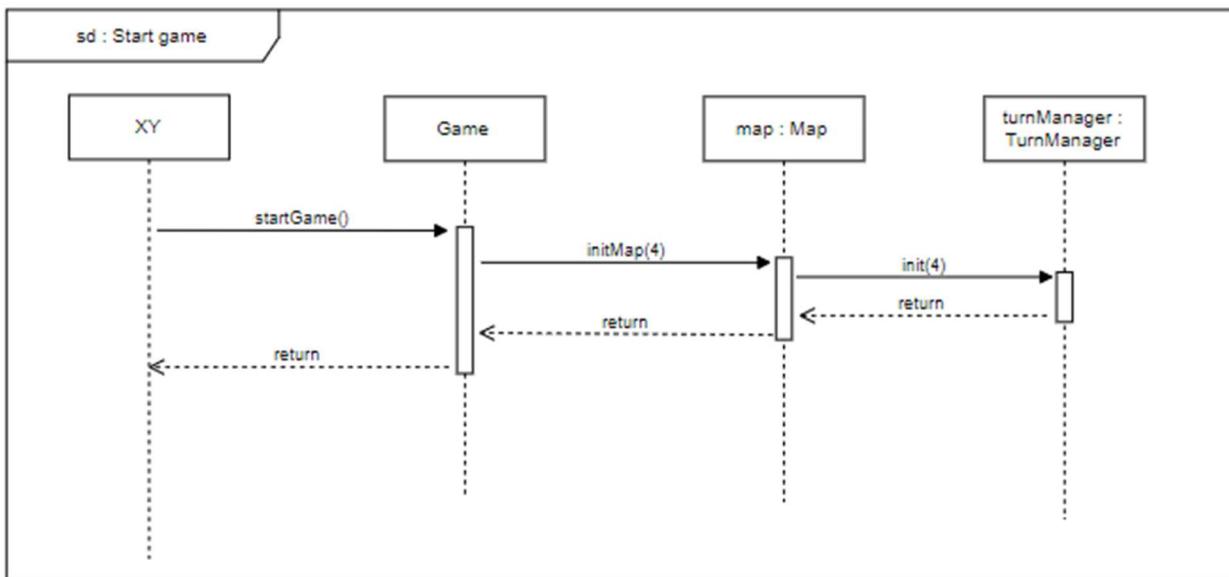
3.4.4. Character moves



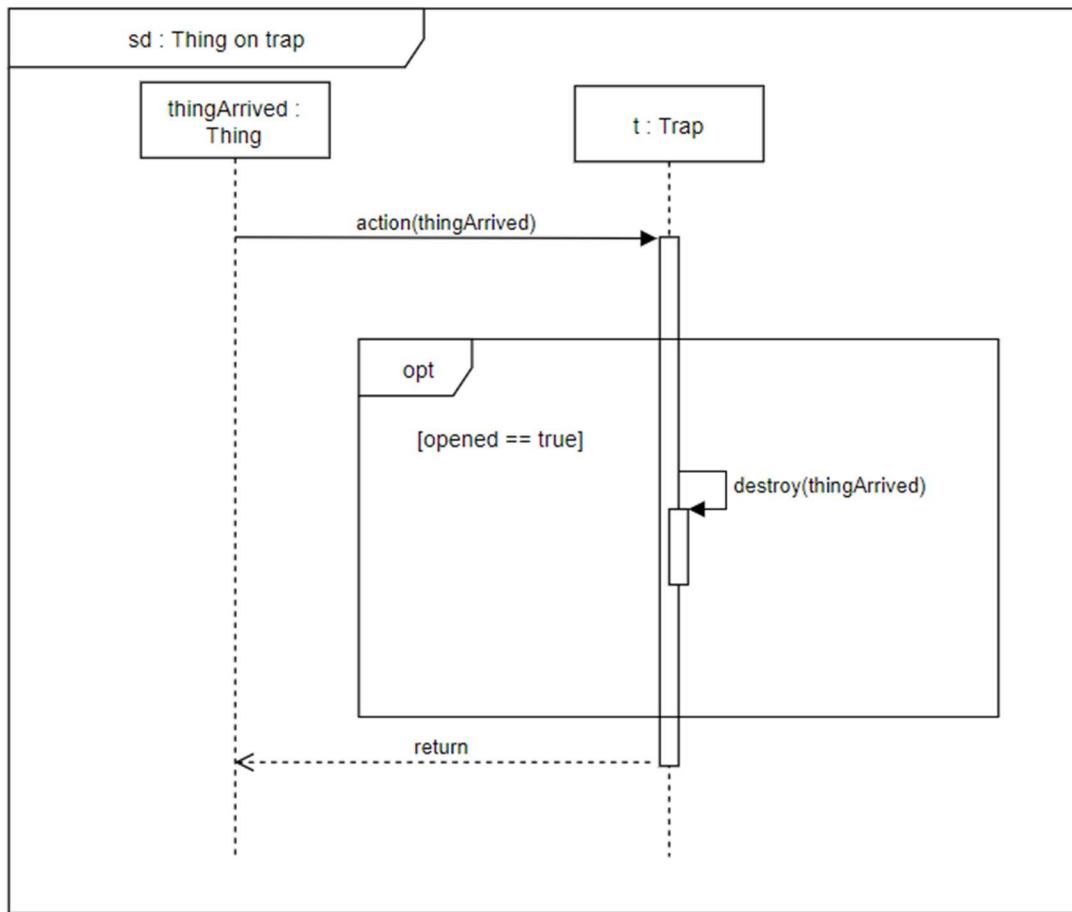
3.4.5. Inactive character collides with wall



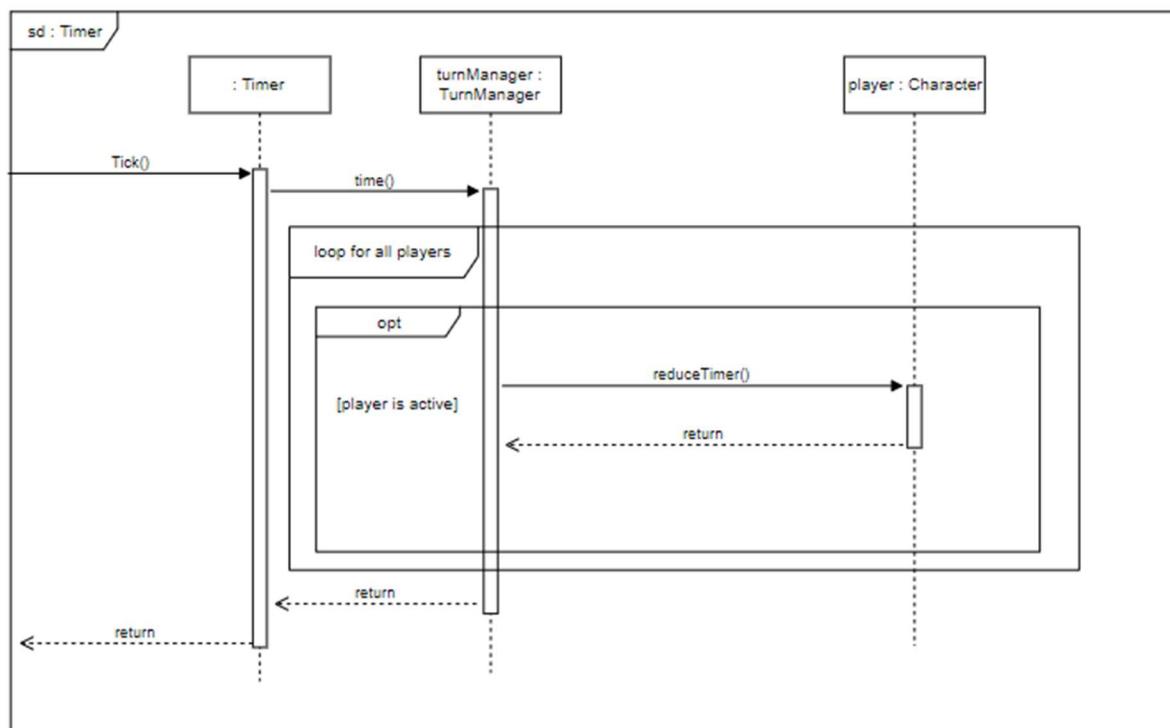
3.4.6. Start game



3.4.7. Thing on trap

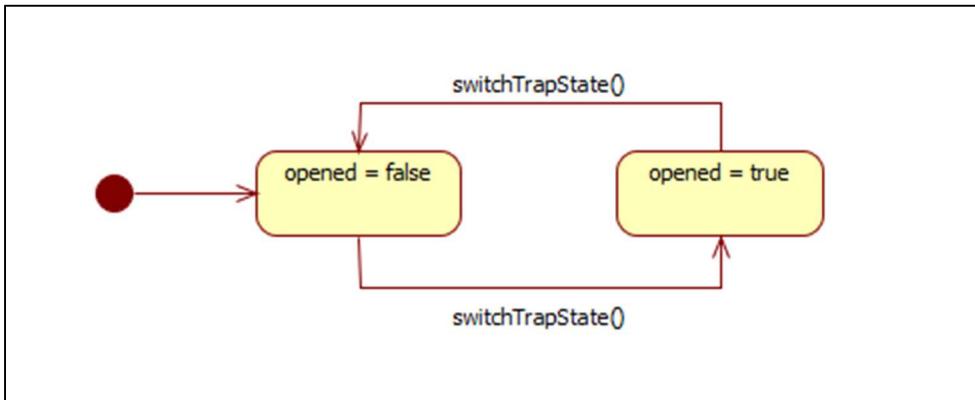


3.4.8. Timer



3.5. State-chartok

3.5.1. Trap state changes



3.6. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.02.22. 15:00	3 óra	Szatmáry Máté Bende	Értekezlet: UML osztálydiagram tervezésének kezdete
2018.02.24. 21:00	1 óra	Szatmáry Máté Bende Kapitány Králik	Értekezlet: UML osztálydiagram tervezésének folytatása
2018.02.24. 22:00	3 óra	Szatmáry Máté Bende Králik	Értekezlet: UML osztálydiagram tervezésének folytatása II.
2018.02.25 11:00	6 óra	Szatmáry	Objektum leírások és osztályok leírása, 3.3.1, 3.3.2 és 3.3.3 elkészítése
2018.02.25 11:00	6 óra	Máté	Osztálydiagram elkészítése, 3.3.4 és 3.3.5 elkészítése
2018.02.25 11:00	6 óra	Králik	Objektum katalógus, Osztályok leírása 3.3.6, 3.3.7 és 3.3.8 elkészítése
2018.02.25 11:00	6 óra	Kapitány	Szekvencia diagramok elkezdése 3.3.9, 3.3.10 és 3.3.11 elkészítése
2018.02.25 11:00	6 óra	Bende	State chart elkészítése, 3.3.12, 3.3.13 és 3.3.14 elkészítése
2018.02.25 17:00	0.5 óra	Szatmáry Bende Máté Králik Kapitány	Értekezlet: napló elkészítése.

4. Analízis modell kidolgozása

4.1. Objektum katalógus

4.1.1. Box

A pályán található dobozokért felelős osztály. A játékosok tudják őket egy karakter vagy egy másik doboz nekitolásával mozgatni a Direction által meghatározott irányokba. Ezeket kell a játékosnak megfelelő helyre tolnia.

4.1.2. Character

A játékosokért felelős osztály, a pályán található játékosokhoz tartozó objektum. A karakterek azok a játékelemek, amiket a játékos közvetlenül tud irányítani.

4.1.3. Direction

Felsorolás, a pályán megkülönböztetett irányokat jelöli, amely irányokba a karakterek vagy dobozok mozoghatnak.

4.1.4. Field

A Map osztályban található mezők. Azok a négyzetek, amik a pályát alkotják. Ilyenek a Goal, NormalField, Trap vagy Switch mezők.

4.1.5. Game

A játék indításáért és befejezéséért felelős osztály. Tartalmazza a játékhoz kialakított pályát.

4.1.6. Goal

Cél mező, ha látad tolunk rá, pontot kap az a játékos, amelyik rátolta, a rátolt láda pedig eltűnik a pályáról.

4.1.7. Hole

Lyuk mező, amire ha rákerül, akkor megsemmisíti a rajta lévő Box vagy Character elemet. Annyiban különbözik a Trap, azaz csapda mezőtől, hogy a lyuk az konstans lyukként viselkedik, és nem kapcsoló hatására vált át üres, járható mezőből lyukká, csapdává.

4.1.8. Map

A pálya, amin a játék zajlik, mezők (Field) nxn méretű táblájából áll, amin megjelenhetnek falak, dobozok, csapdák, kapcsolók, karakterek, és üres mezők (NormalField).

4.1.9. NormalField

Üres mező, aminek semmi különleges szerepe nincs, azaz nem történik semmilyen extra esemény, ha rákerül egy karakter vagy egy doboz.

4.1.10. Switch

Kapcsoló mező, ez hozza működésbe a hozzá tartozó csapdát ha látad tolunk rá. Játékostól nem aktiválódik.

4.1.11. Thing

Gyűjtőfogalom a pályán található tárgyakra, mint karakter, fal, vagy doboz.

4.1.12. Timer

Az egyes játékosok külön számlálóval rendelkeznek, ami a hátralevő (lépés előtti gondolkozásra felhasználható) idejüket tartja nyilván. A Timer osztály egy példánya ezeket az időket csökkenti másodpercenként az éppen aktuális játékosnak. Ha a játékos ideje elfogy, akkor többet már nem léphet, viszont a játékot ettől függetlenül még megnyerheti.

4.1.13. Trap

Csapda mező, ha az éppen aktuális Trap-hoz tartozó Switch osztály egy példánya bekapcsolja, akkor megsemmisíti a rajta lévő Box vagy Character elemet.

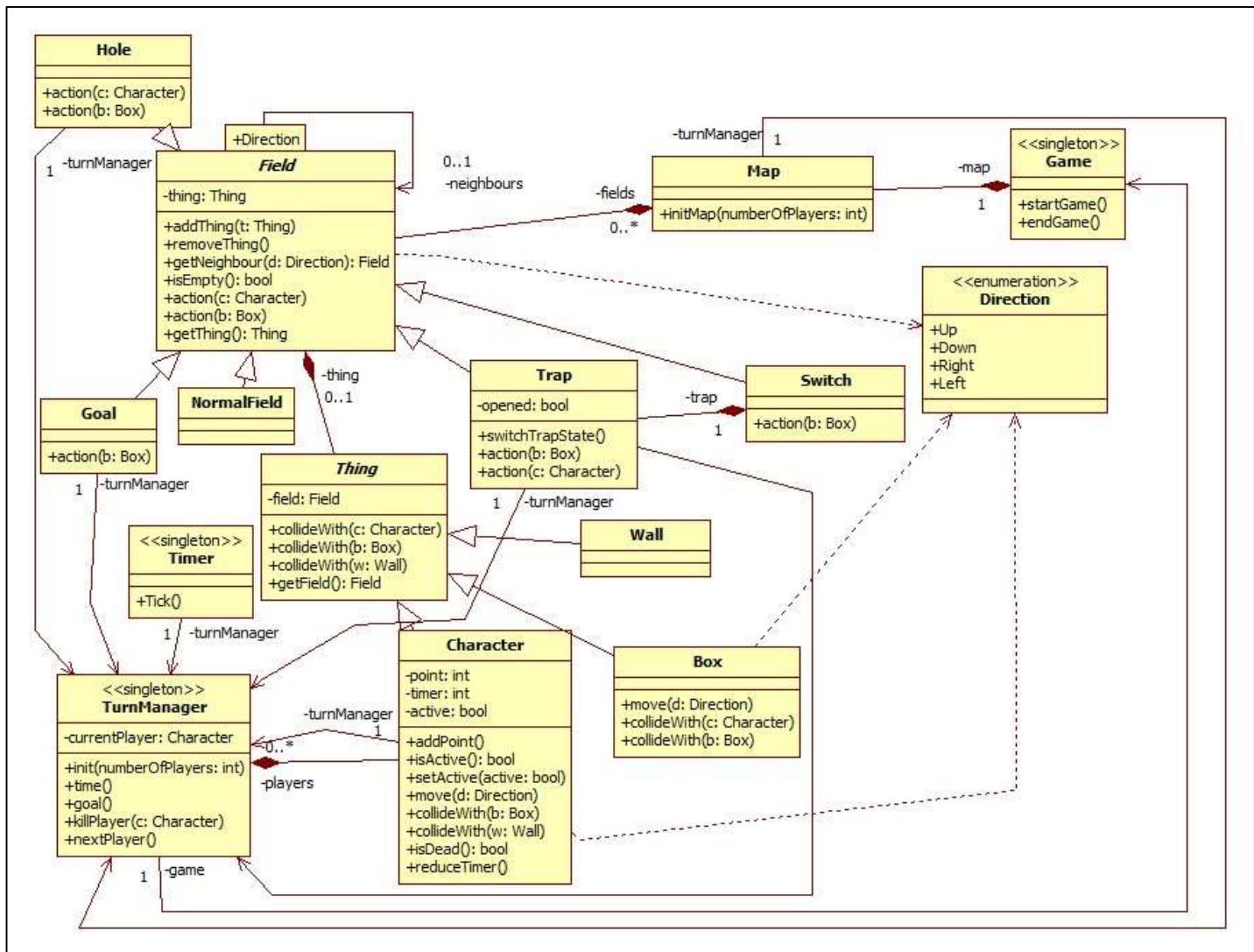
4.1.14. TurnManager

Megkapja a játékosok számát, majd menedzseli a játékmenetet, azaz, hogy melyik játékosnak a köre következik, ki az aktuális játékos, annak mennyi ideje van hátra, továbbá pontokat oszt ki a feladatok végrehajtásával.

4.1.15. Wall

A játékban található falakat modellező osztály. Abban különbözik a pályán lévő többi elemtől (Thing), hogy semmilyen módon nem elmozdítható. A fal irányába nem léphet a játékos, ha azt teszi, az is lépésnek számít, de a karakter nem fog mozdulni, így effektíve kimarad a körből az adott játékos.

4.2. Statikus struktúra diagramok



Az osztályokhoz és a nevesített asszociációvégekhez implicit getter és setter függvények tartoznak, amelyeket a diagram az olvashatóság kedvéért nem minden esetben jelöl.

4.3. Osztályok leírása

4.3.1. Box

- Felelősség**

A pályán a Map initMap függvénye által létrehozott dobozokat jelképző objektumok. Ezeket tologathatja a játékosok szabadon a pályán. Ha egy játékos egy dobozt egy célmezőre tol, akkor pontot kap érte, a doboz pedig eltűnik. A dobozok le is eshetnek a lyukakba, de más módon nem tűnhetnek el a pályáról, nem nyomhatják össze egymást, ugyanakkor egymást amennyiben a pályán van szabad hely - arrébb tolhatják a következő mezőre, ugyanúgy ahogy karaktert is tovább tolhatnak, falhoz nyomva (üres következő mező hiányában) meg is ölhetik azt. Ez bármennyi sorban lévő doboz esetén rekurzívan is érvényes.

- Ősosztályok**

Thing

- Metódusok**

void move(Direction d): Eltolja a dobozt egy másik mezőre.

void collideWith(b: Box): Ha másik dobozzal ütközik, tovább tudja azt tolni.

void collideWith(c: Character): Miközben őt tolják (mint dobozt), és neki megy egy másik Characternek, azt el tudja tolni.

4.3.2. Character

- Felelősség**

A játékos által irányított karakter. minden játékosnak minden körben pontosan egy darab gomblenyomásra van lehetősége, ha a játékos mozdíthatatlan pályaelem, mint fal, oszlop, vagy ezek mellett lévő doboz felé mozgatná a karaktert, a rendszer azt is lépésnek veszi, tehát a játékos az adott körben a lépését elhasználta, ennek ellenére nem mozgott a pályán.

- Ősosztályok**

Thing

- Attribútumok**

int point: A játékos aktuális pontszáma.

int timer: A játékos még hátralévő ideje.

bool active: Jelzi, hogy a játékos éppen aktív-e.

- Metódusok**

void addPoint(): Hozzáad egy pontot a játékos pontszámához.

bool isActive(): Visszatér egy igaz/hamis értékkel, hogy a játékos éppen aktív-e.

void setActive(bool activate): Aktiválja vagy deaktiválja a játékost, függően attól, hogy ő van e soron.

void collideWith(b: Box): Doboznak próbál menni, megpróbálja eltolni.

void collideWith(w: Wall): Ha falnak ütközik, megnézi, hogy ő az aktív játékos-e, ha nem, akkor meghal.

bool isDead(): Visszatér egy igaz/hamis értékkel, hogy meghalt-e már valamilyen okból a karakter.

void reduceTimer(): Megadott időegységgel csökkenti a játékos idejét.

4.3.3. Direction

- **Felelősség**

A pályára vonatkozó közlekedési irányok enumerációját testesíti meg.

- **Attribútumok**

Up: Fel.

Down: Le.

Left: Balra.

Right: Jobbra.

4.3.4. Field

- **Felelősség**

A Map osztályban található mezőknek az absztrakt alaposztálya. Összefoglaló osztály, ezek tömbjét tárolja a Map osztály egy adattagjában.

- **Attribútumok**

Thing thing: Az a Thing “dolog”, ami az adott pillanatban ezen a mezőn rajta van. Lehet ez doboz, fal, vagy játékos.

- **Metódusok**

addThing(t: Thing): Valami vagy valaki hozzáadása a mezőhöz (“ráhelyezzük”).

removeThing(): Akkor hívódik meg, ha egy Character ellép, vagy egy Box eltolódik az adott Field mezőről.

Field getNeighbour(d: Direction): A paraméterként kapott irányba található szomszédját adja vissza.

bool isEmpty(): Visszaadja, hogy üres-e a mező (van-e rajta valami/valaki).

action(c: Character): A mezőn lévő karakter esetén végrehajtandó művelet, a leszármazottakban felülírjuk a megfelelő működéssel.

action(b: Box): A mezőn lévő doboz esetén végrehajtandó művelet, a leszármazottakban felülírjuk a megfelelő működéssel.

4.3.5. Game

- **Felelősség**

A pálya nyilvántartásáért és a játékmenet legfőbb funkcióiért (elindítása vagy lezárása) felelős. Meghívja további osztályok inicializáló függvényeit.

- **Attribútumok**

Map map: Számoltartja a játékhoz tartozó pályát.

- **Metódusok**

void startGame(): A teljes játék elindításáért felelős. Bekéri a felhasználótól a játékosok számát, majd meghívja a Map osztály initMap függvényét.

void endGame(): A játék lezárásáért felelős. Akkor fut le, amikor minden játékosnak elfogy a saját ideje. Ilyenkor összegzi az eredményeket, kiírja a győztest, vagy győzteseket és visszalép a játék főmenüjébe.

4.3.6. Goal

- **Felelősség**

A pályának azon, megkülönböztetett színnel jelzett mezői, ahova a játékosoknak dobozokat kell tologatniuk, hogy pontokat kapjanak. A játék célja, hogy adott játékos minél több célmezőre, minél több dobozt toljon.

- **Ősosztályok**

Field

- **Attribútumok**

TurnManager turnManager: referencia a TurnManagerre, hogy tudja jelezni neki, ha pontot kell adni az aktuális játékosnak.

- **Metódusok**

void action(b: Box): Ha dobozt tolunk rá, akkor jön csak működésbe, az ősosztály ezen metódusát felüldefiniálva oldottuk meg. Ez ad pontot az aktuális játékosnak.

4.3.7. Hole

- **Felelősség**

Olyan Field mező, amire ha rákerül Character vagy Box objektum, akkor azt megsemmisül/meghal. Abban különbözik a Trap osztály példányaitól, hogy még azok a Switch-el való aktiválásig NormalField-ként viselkednek (nem jelentenek veszélyt a rájuk tolta Box vagy Character-re), addig a Hole az konstans lyukként viselkedik, bármelyik időpillanatban megsemmisít, bármilyen rá kerülő objektumot.

- **Ősosztályok**

Field

- **Metódusok**

action(Character c): Megsemmisíti a rákerülő Character objektumot.

action(Box b): Megsemmisíti a rákerülő Box objektumot.

4.3.8. Map

- **Felelősség**

A pálya, amin a játék zajlik. Mezők nxn méretű mátrixából áll, eltárolja azokat. A pályát ō maga generálja le megadott szabályoknak és kikötéseknek megfelelően.

- **Attribútumok**

Field[][] fields: A pálya mátrixának elemeit tartalmazó tömb. A Field absztrakt osztály típusának mezőit tartalmazza, azaz Goal, NormalField, Trap, és Switch objektumok kerülhetnek bele - ezek a pálya alapelemei. A játékmenetet a GetNeighbour() alapján lesz számolva, ez a tömb a kirajzoláshoz van segítségül.

- **Metódusok**

void initMap(numberOfPlayers int): A pályát létrehozó függvény, paraméterül kapja a Game singleton-tól az elindítani kívánt játékban résztvevő játékosok számát, amiket aztán tovább is ad a TurnManager osztálynak, ami azt az init() függvényében használ fel.

4.3.9. NormalField

- **Felelősség**

A pályán található sima üres mezőkért felelős osztály. A játékosok és a dobozok ezeken tudnak szabadon közlekedni, az erre való rálépéskor nem történik semmilyen speciális esemény.

- **Ősosztályok**

Field

4.3.10. Switch

- **Felelősség**

A pályán megtalálható láthatatlan (úgy néz ki mint egy sima mező) kapcsoló, amelyhez pontosan egy csapda objektum tartozik.. Akkor aktiválódik, ha egy dobozt tolnak rá, játékosként rálépve nem történik semmi.

- **Ősosztályok**

Field

- **Attribútumok**

Trap trap: A kapcsolóhoz tartozó csapda.

- **Metódusok**

action(Box b): Ha dobozt kerül a mezőre, akkor aktiválja a hozzá tartozó csapdát.

4.3.11. Thing

- **Felelősség**

Ez az osztály felelős a mezőkön megtalálható “dolgokért”. Ilyen lehet például fal (Wall), doboz (Box) és játékos (Character). Esetleges ütközések esetén meghatározza, hogy milyen cselekmény fog végbe menni.

- **Attribútumok**

Field field: Az a mező amelyen a dolog található.

- **Metódusok**

void collideWith(Character c): Ha egy adott dolog egy játékossal ütközik, kezeli azt.

Kimenetele függ, hogy milyen dolog ütközik a játékossal.

void collideWith(Box b): Ha egy adott dolog egy dobozzal ütközik, kezeli azt. Kimenetele függ, hogy milyen dolog ütközik a dobozzal.

void collideWith(Wall w): Ha egy adott dolog egy fallal ütközik, kezeli azt. Kimenetele függ, hogy milyen dolog ütközik a fallal.

4.3.12. Timer

- **Felelősség**

A számítógép belső óráját kérdezi le, és értesíti a TurnManager osztályt arról, hogy eltelt egy másodperc.

- **Attribútumok**

TurnManager turnManager: Egy referencia a TurnManager singleton osztályra, amelyen keresztül eléri annak a time() függvényét, és inkrementálja a játékosok megfelelő adattagjait.

- **Metódusok**

void Tick(): Végtelen ciklusban fut, lekérdezi a számítógép idejét, és ha eltelik egy megadott időegység, akkor meghívja a TurnManager time() függvényét, hogy csökkentse az aktuális játékos idejét.

4.3.13. Trap

- **Felelősség**

A pályán megtalálható csapda más néven “lyuk” objektumokért felelős osztály. Ha egy játékos vagy doboz belelép az megsemmisül/meghal. minden switchhez pontosan egy csapda tartozik.

- **Ősosztályok**

Field

- **Attribútumok**

bool opened: Jelzi a csapda jelenlegi állapotát (aktivált, deaktivált).

- **Metódusok**

void switchTrapState(): Aktiválja, deaktiválja magát.

action(Box b): Megsemmisíti a rákerülő Box objektumot, ha a csapda aktiválva van.

action(Character c): Megsemmisíti a rákerülő Character objektumot, ha a csapda aktiválva van.

4.3.14. TurnManager

- **Felelősség**

Megkapja a játékosok számát, majd menedzseli a játékmenetet, azaz, hogy melyik játékosnak a köre következik, ki és ki az aktuális játékos, aki az adott körben léphet.

A Timer fix időközönként (mp) inkrementálja a TurnManager által tárolt, és éppen aktív karakternek a hátralévő idejét a time() függvénye segítségével.

- **Attribútumok**

Character players: A játékban résztvevő karaktereket tárolja.

Character currentPlayer: Az éppen aktuális karakterre mutató referencia.

- **Metódusok**

void init(int numberOfPlayers): Kezdéskor létrehozza a kollekciót a játékosokra.

void time(): Amikor a Timer jelez neki, levon az aktuális játékos idejéből.

void goal(): Amikor a Goal jelez neki, pontot ad az aktuális játékosnak.

void killPlayer(c: Character): Ha egy játékosnak elfogy az ideje, vagy csapdába esik, meghal, ez a függvény valósítja ezt meg.

void nextPlayer(): Visszaadja a következő játékos referenciáját.

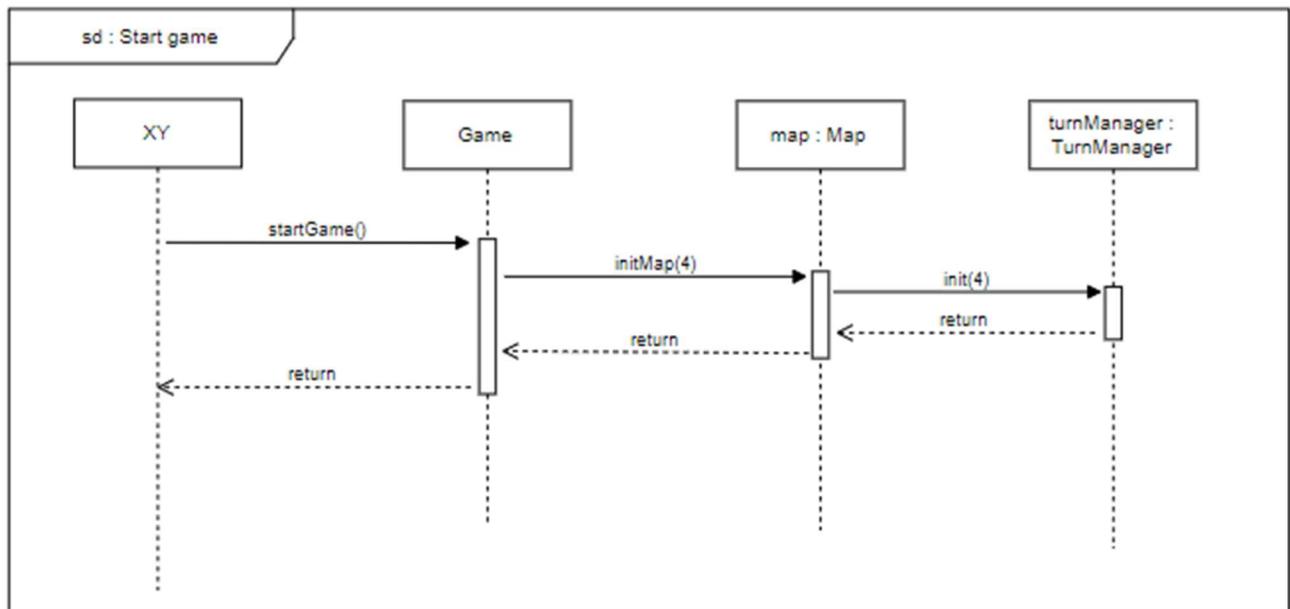
4.3.15. Wall

- **Felelősség**

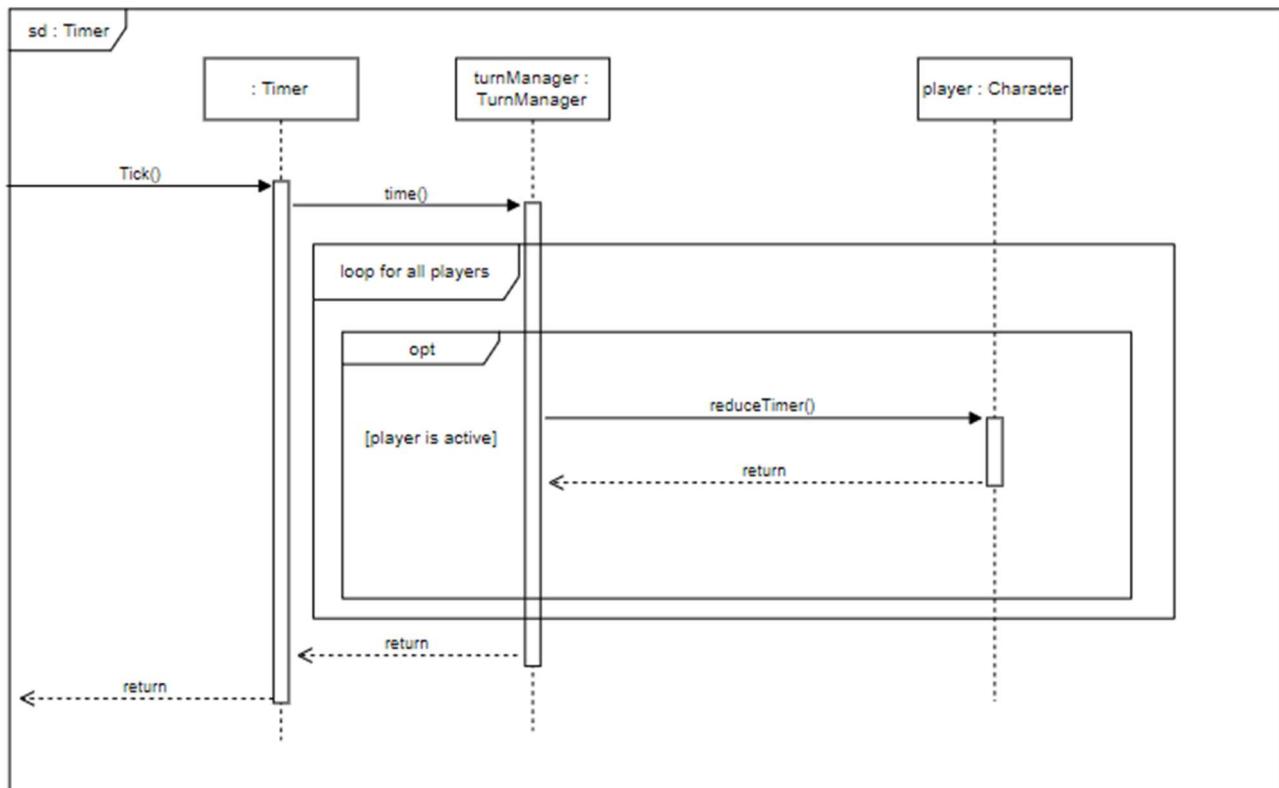
A pályán megtalálható mozdíthatatlan, és áthatolhatatlan fal objektumokért felelős. Ha egy falszakasz nem több, egymással összefüggő komponensből áll (1x1 mező méretű), akkor oszlop, máskülönben fal. Ha magától felnak menne a játékos által irányított karakter, akkor beleütközik, nem történik semmi. Hasonlóan a dobozzal is, viszont amennyiben egy karakter, avagy egy doboz tol egy másik karaktert felnak, az meghal.

4.4. Szekvencia diagramok

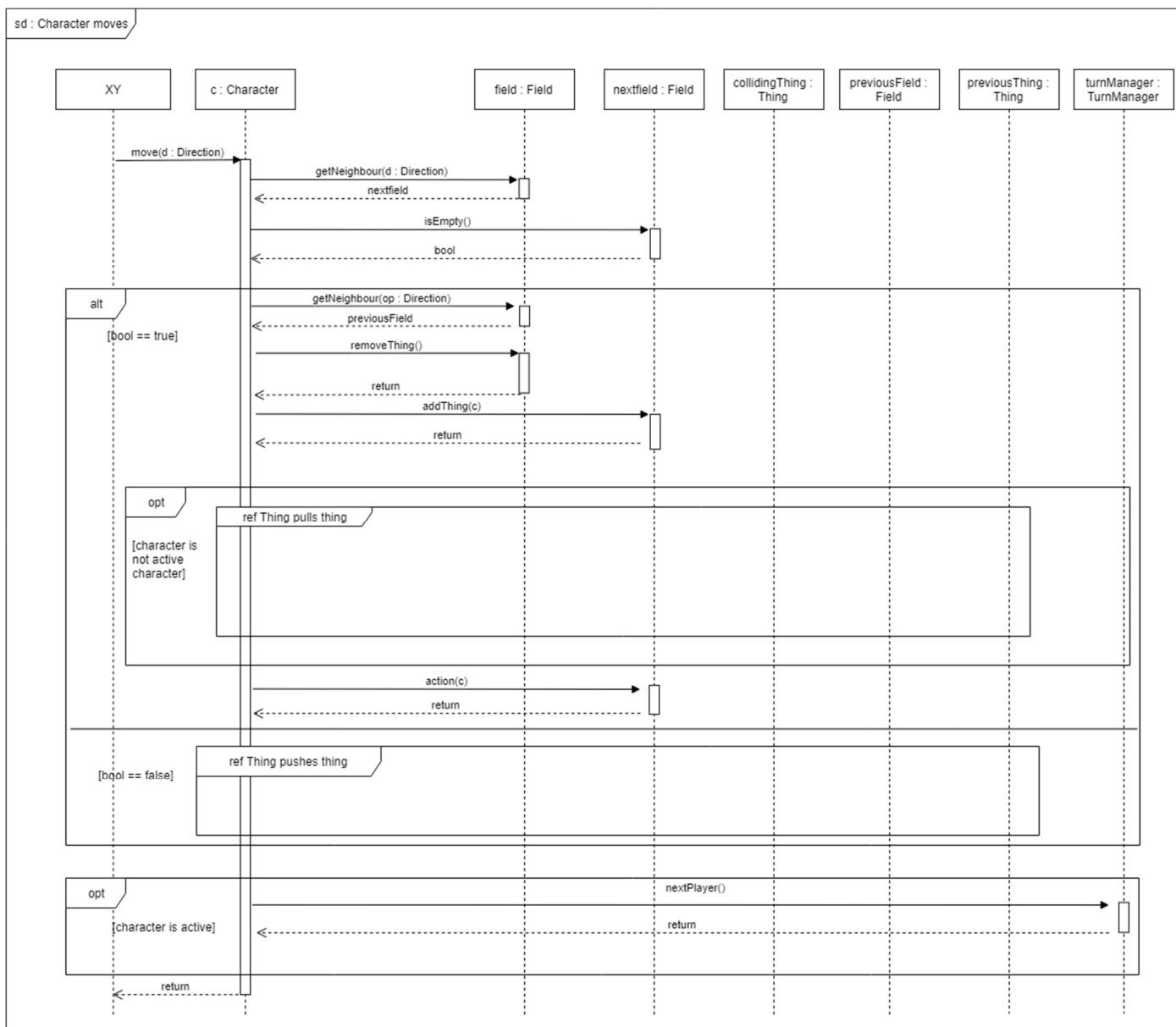
4.4.1. Start game



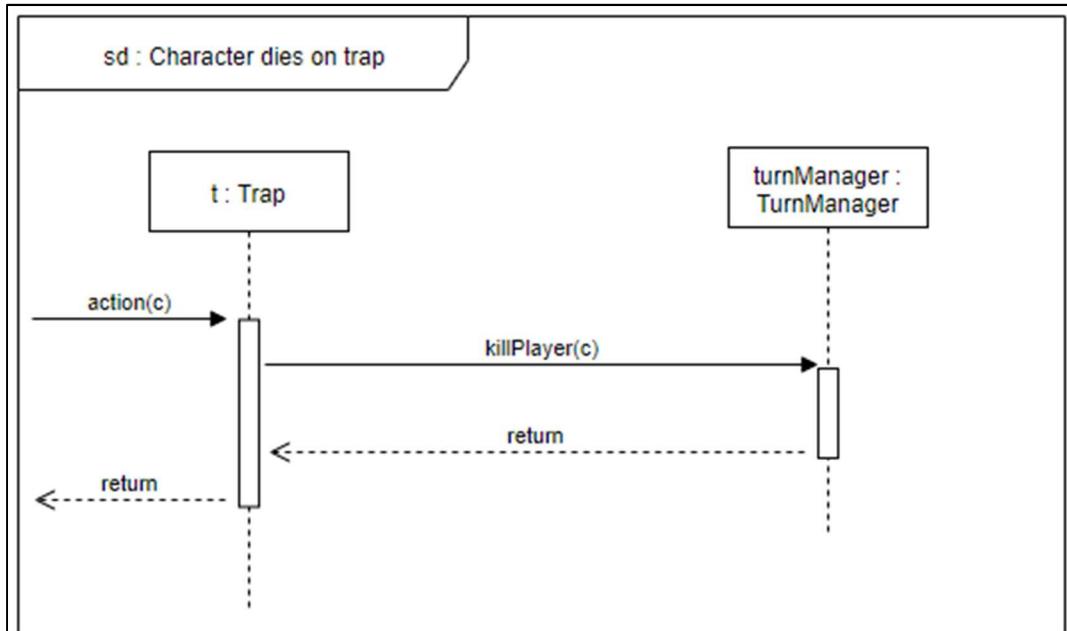
4.4.2. Timer



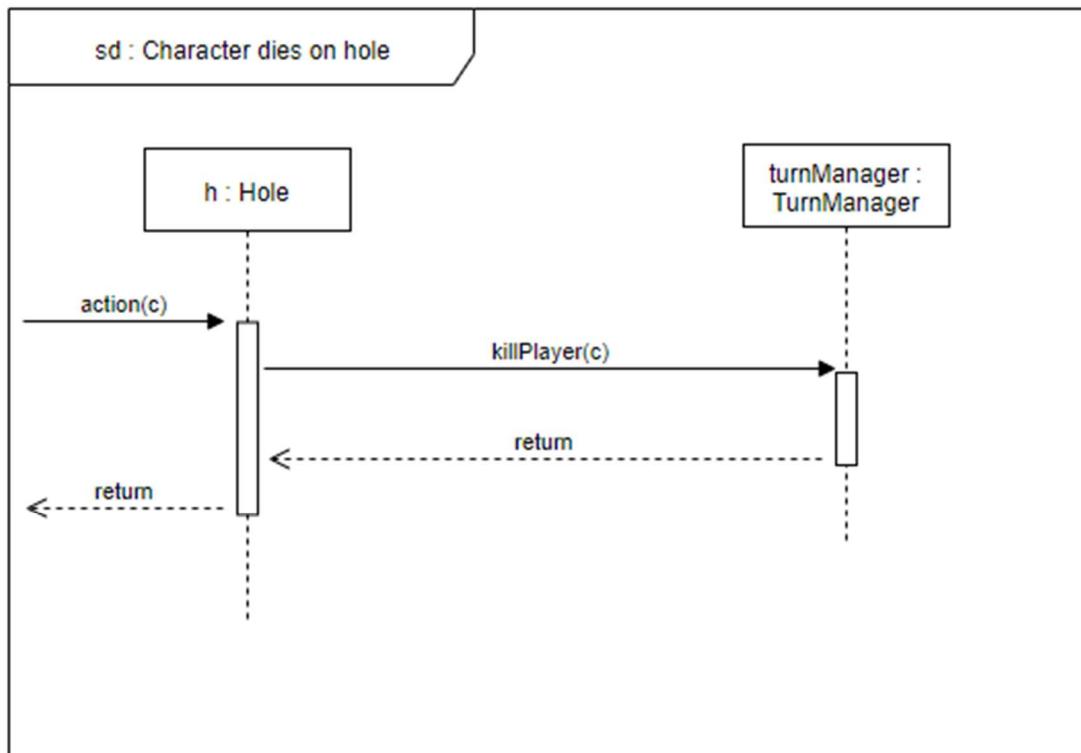
4.4.3. Character moves



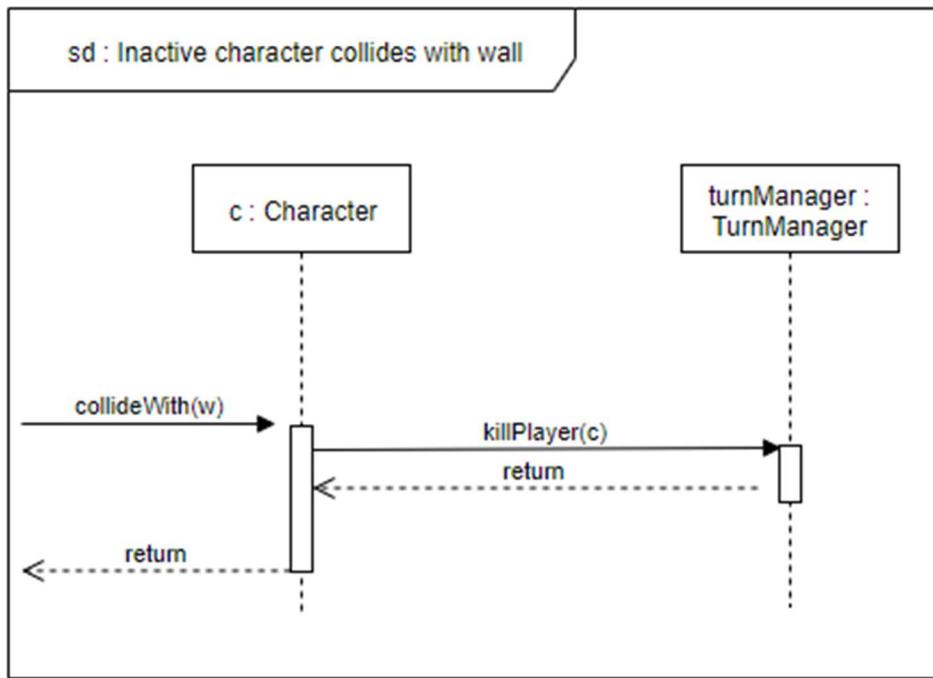
4.4.4. Character dies on trap



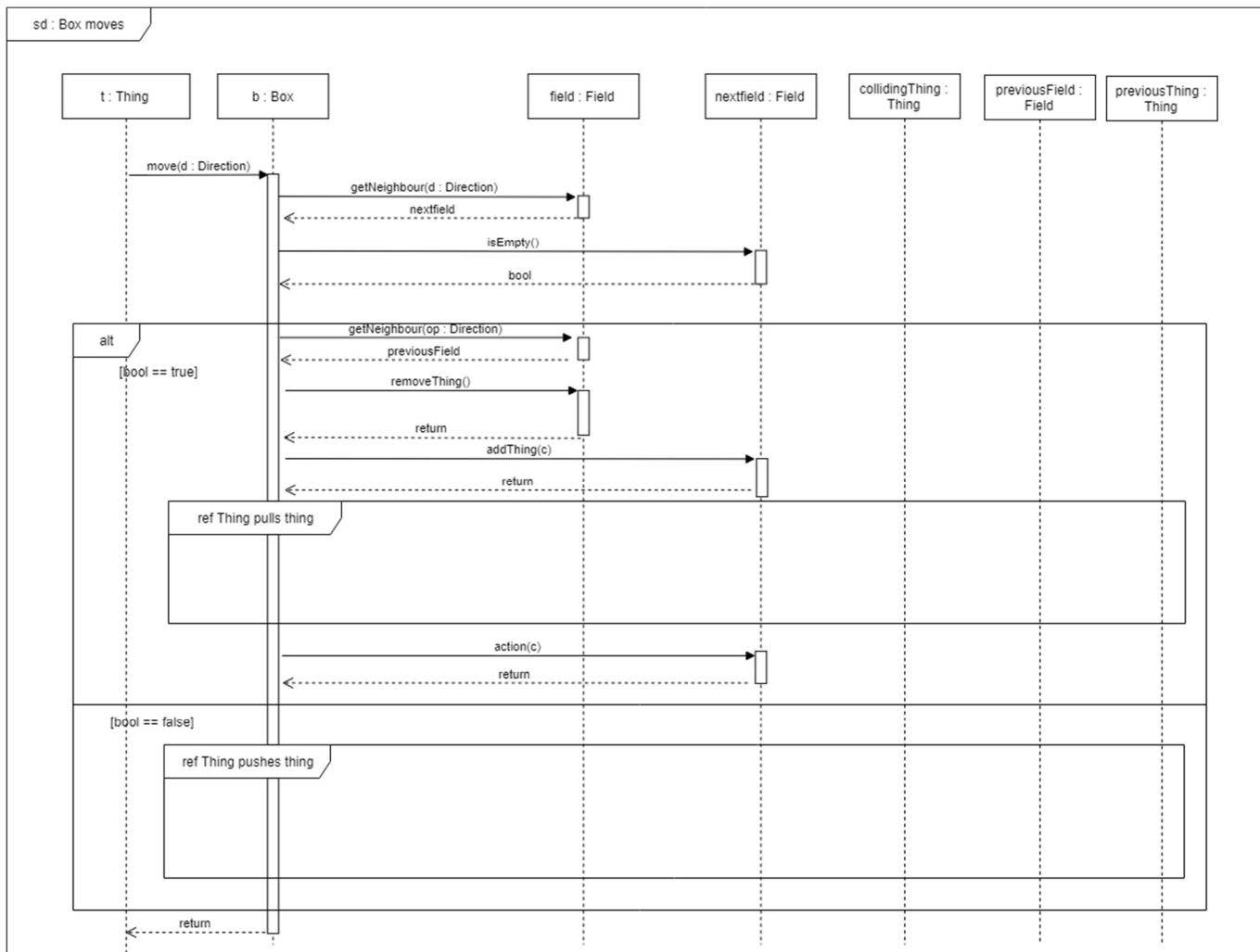
4.4.5. Character dies on hole



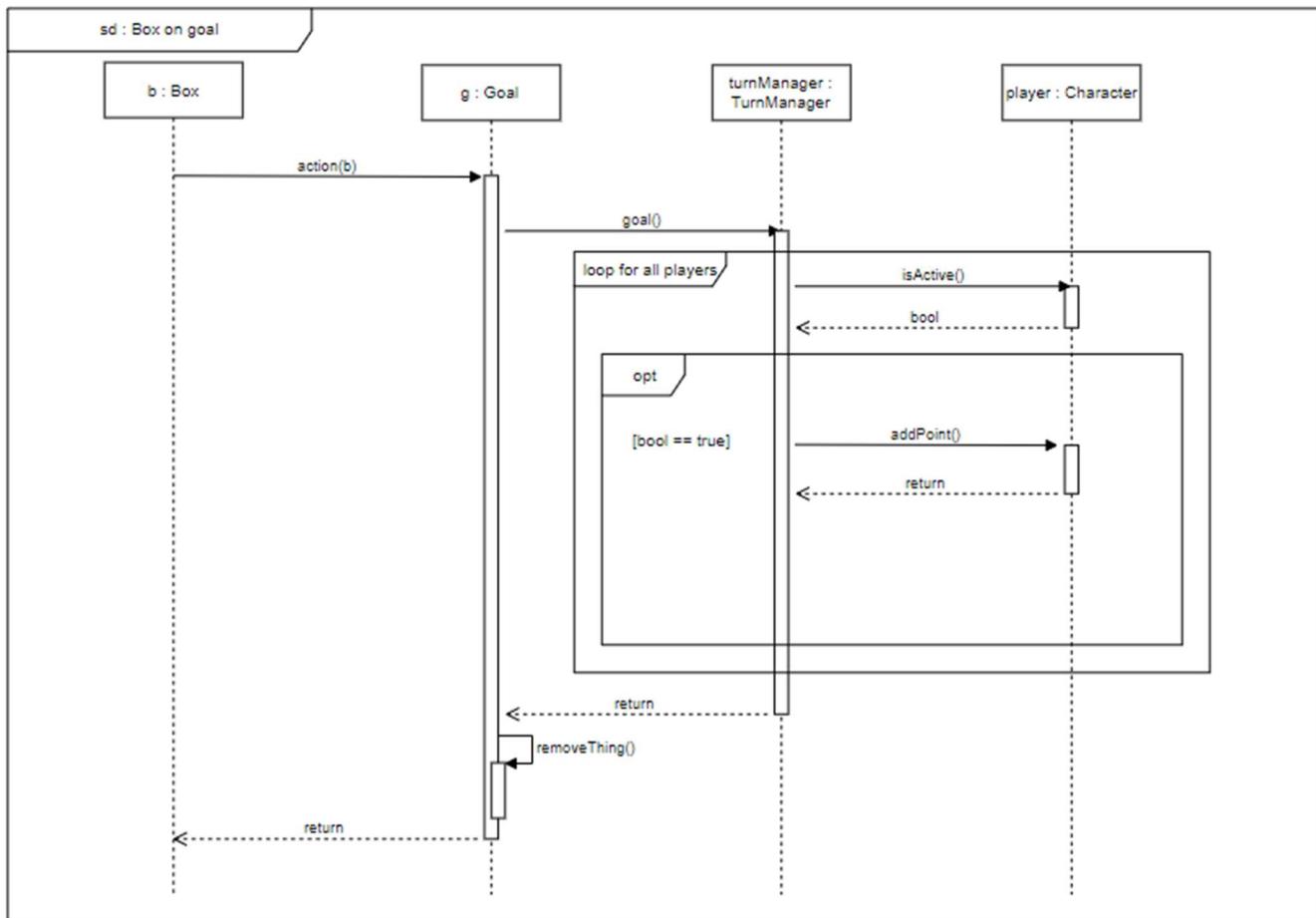
4.4.6. Inactive character collides with wall



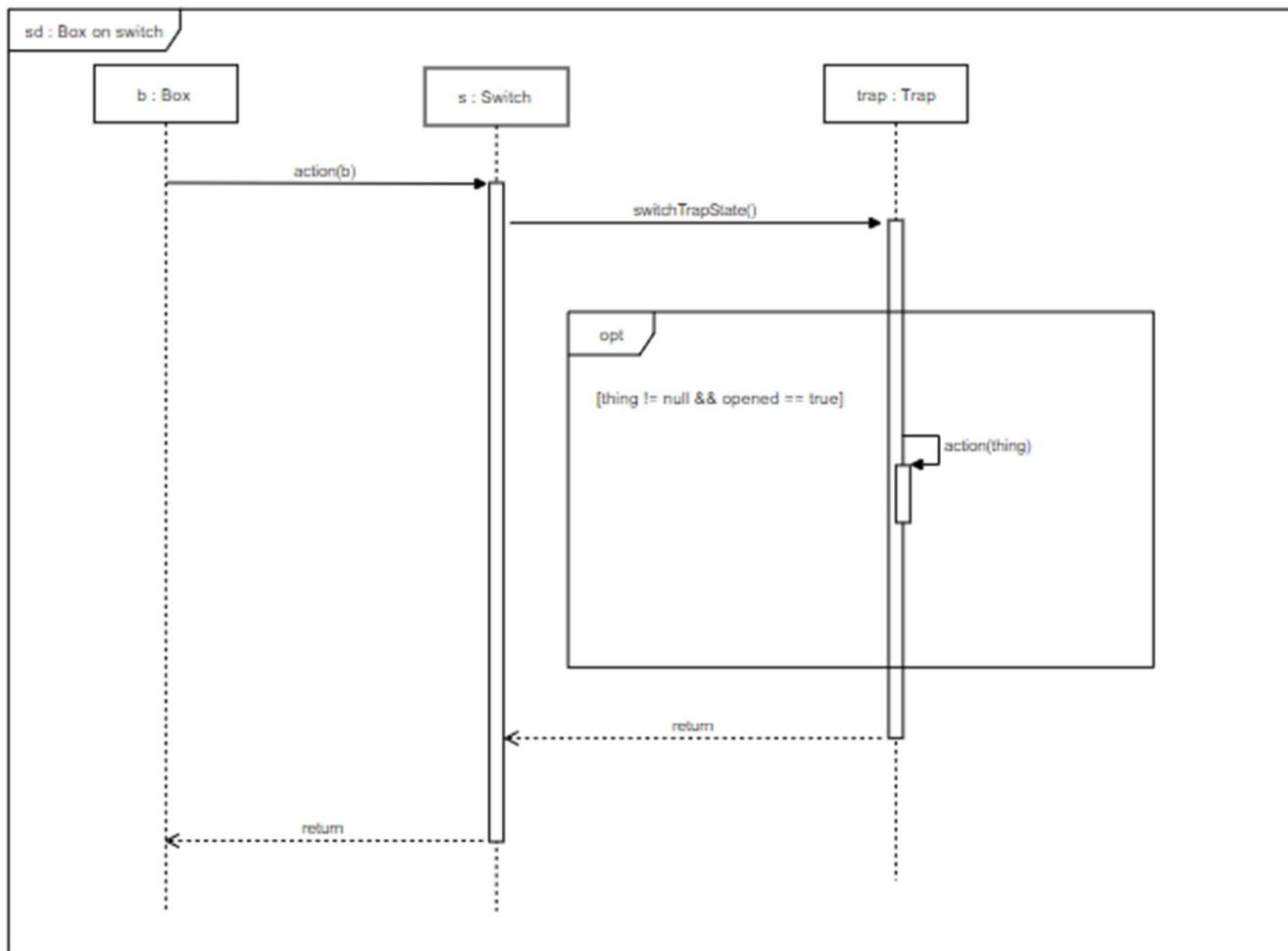
4.4.7. Box moves



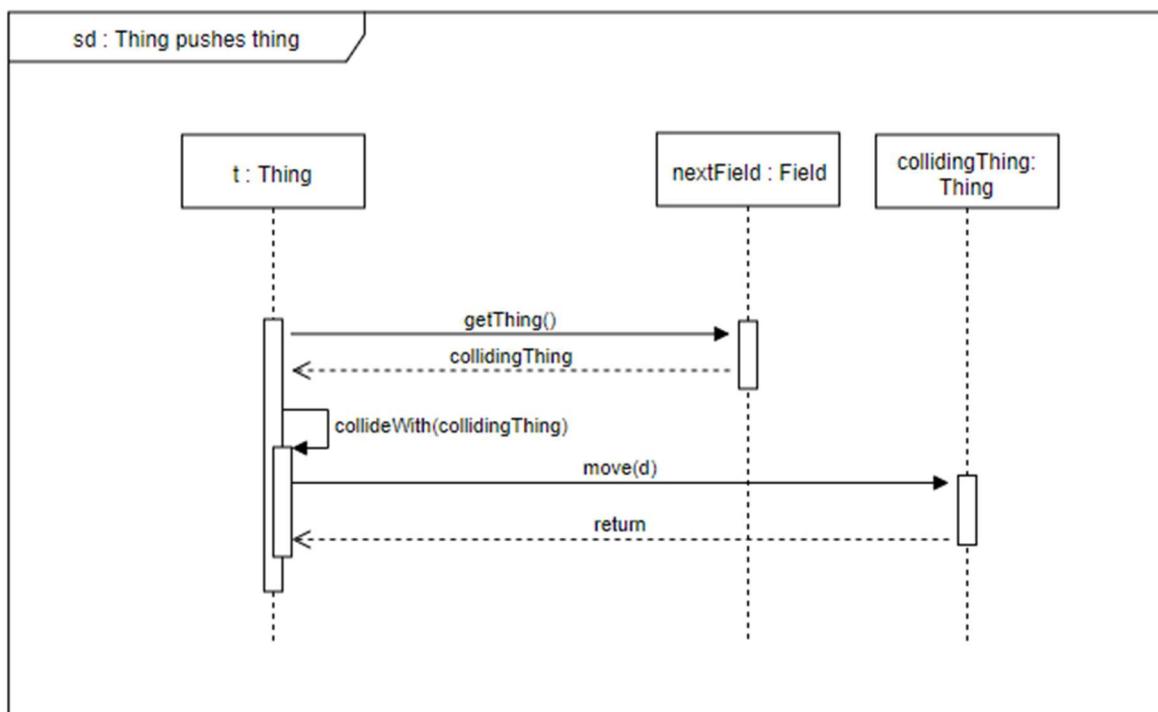
4.4.8. Box on goal



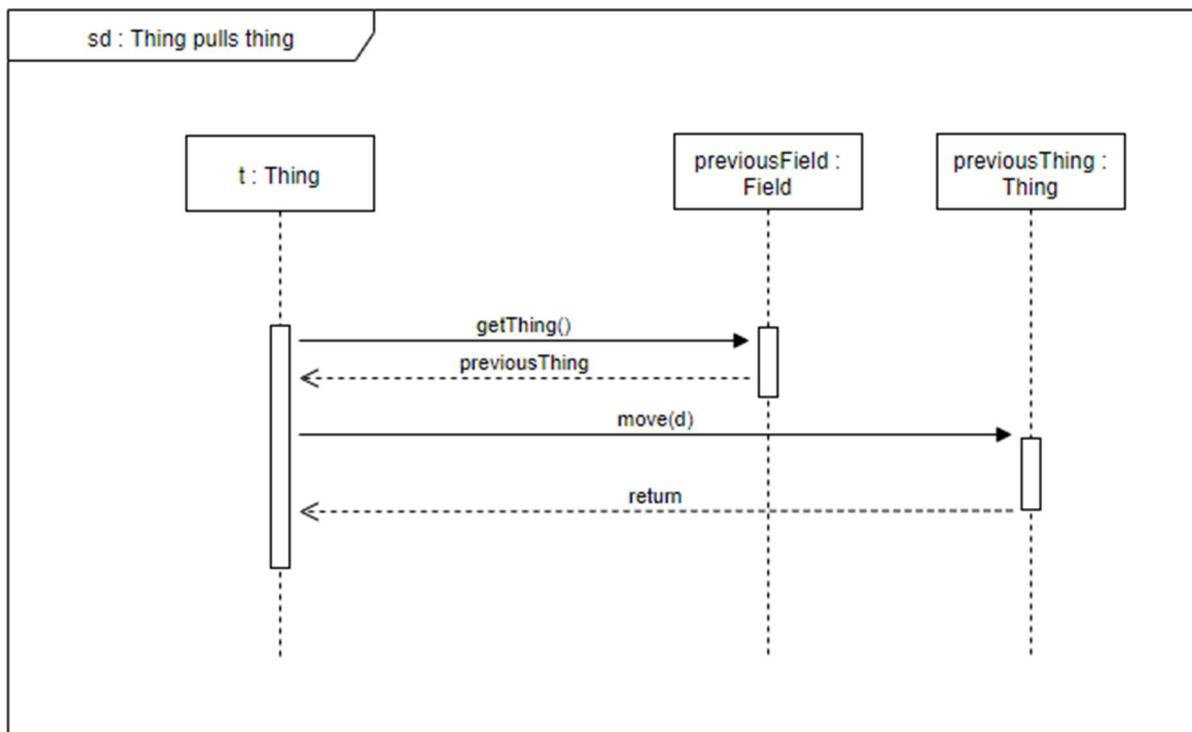
4.4.9. Box on switch



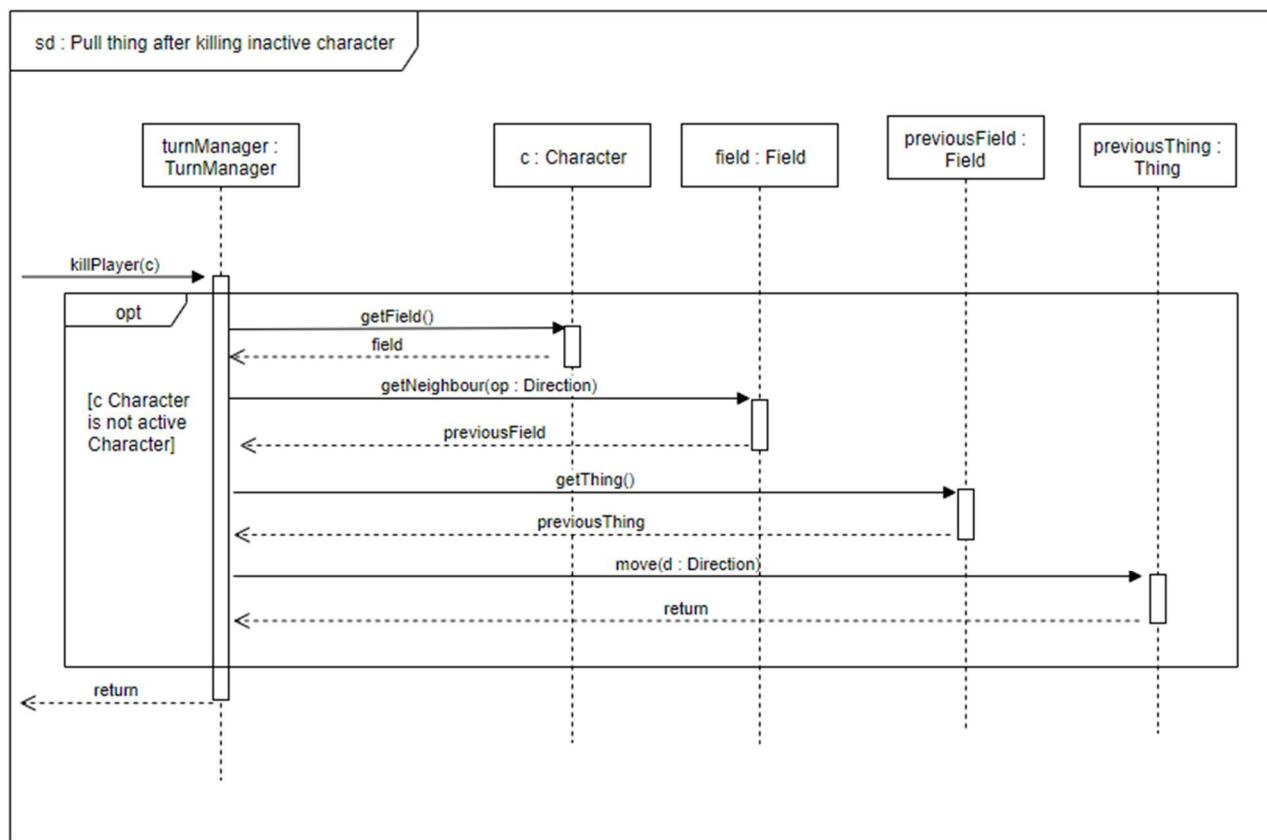
4.4.10. Thing pushes thing



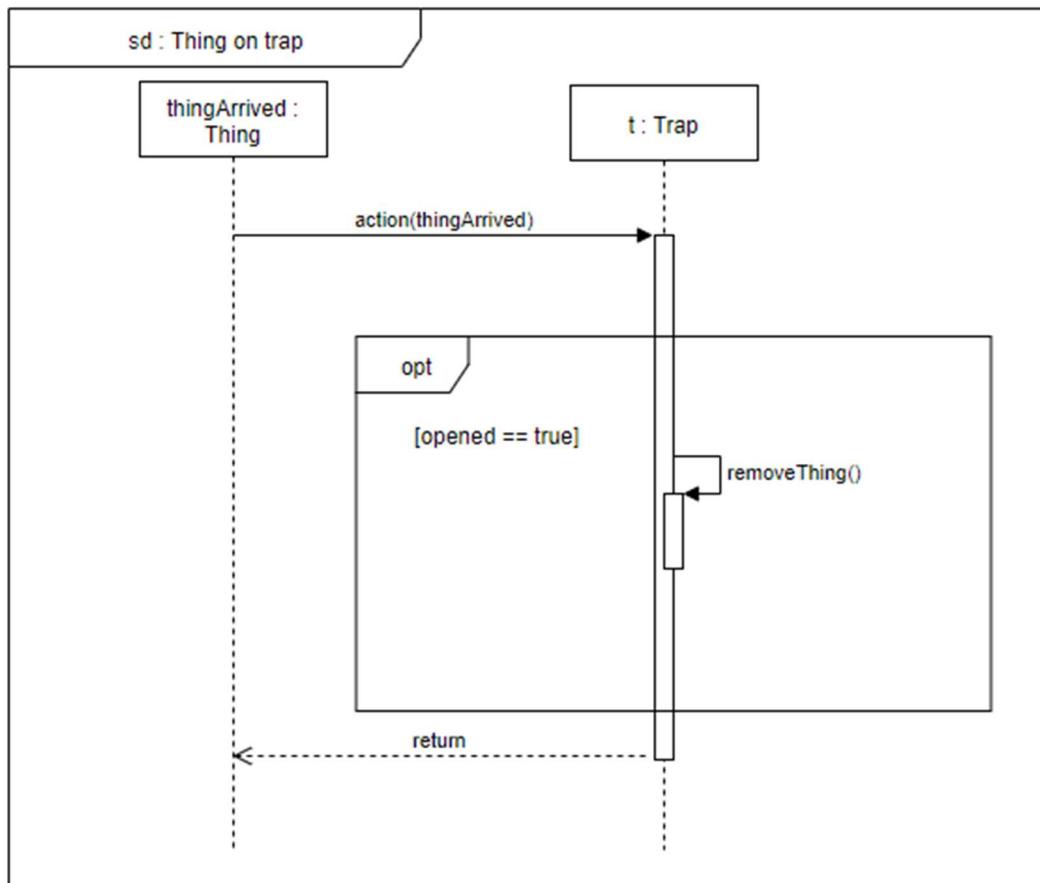
4.4.11. Thing pulls thing



4.4.12. Pull thing after killing inactive character

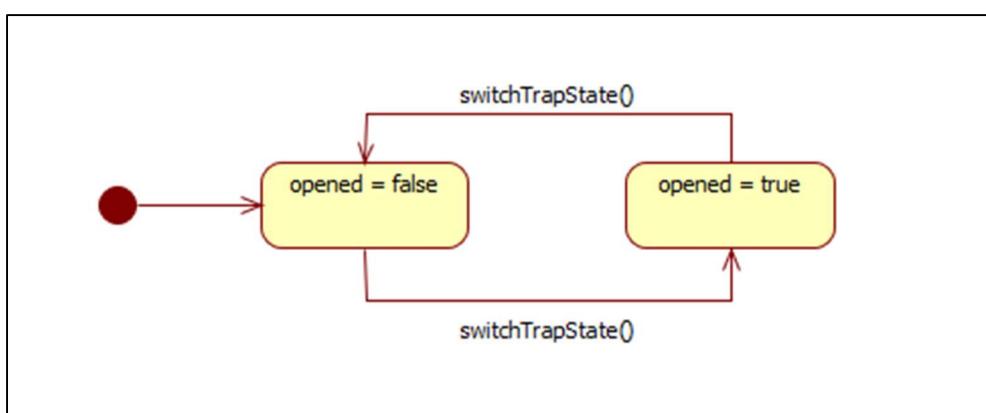


4.4.13. Thing on trap



4.5. State-chartok

4.5.1. Trap state changes



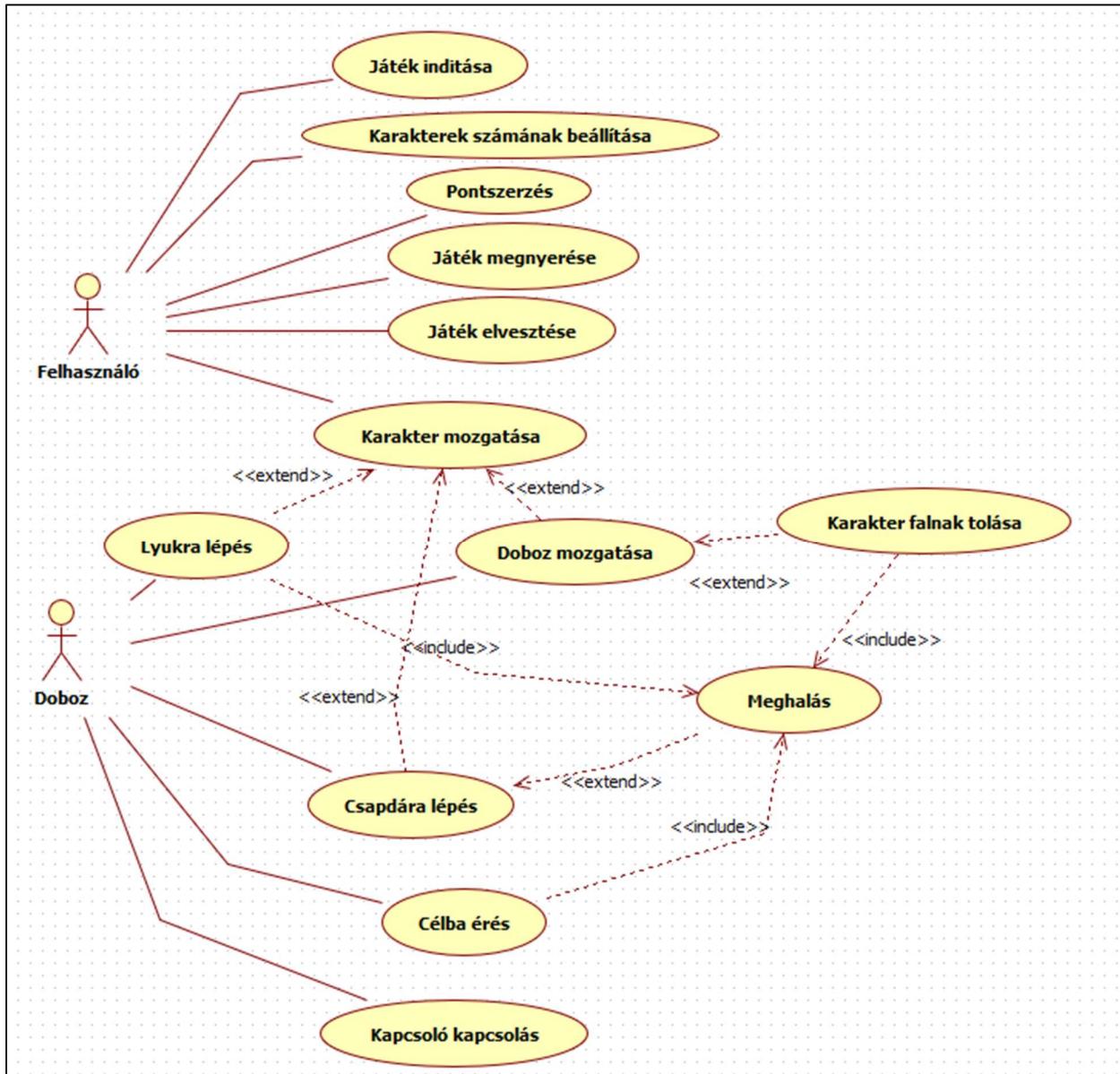
4.6. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.02.22. 15:00	3 óra	Szatmáry Máté Bende	Értekezlet: UML osztálydiagram tervezésének kezdete
2018.02.24. 21:00	1 óra	Szatmáry Máté Bende Kapitány Králik	Értekezlet: UML osztálydiagram tervezésének folytatása
2018.02.24. 22:00	3 óra	Szatmáry Máté Bende Králik	Értekezlet: UML osztálydiagram tervezésének folytatása II.
2018.02.25 11:00	6 óra	Szatmáry	Objektum leírások és osztályok leírása, 3.3.1, 3.3.2 és 3.3.3 elkészítése
2018.02.25 11:00	6 óra	Máté	Osztálydiagram elkészítése, 3.3.4 és 3.3.5 elkészítése
2018.02.25 11:00	6 óra	Králik	Objektum katalógus, Osztályok leírása 3.3.6, 3.3.7 és 3.3.8 elkészítése
2018.02.25 11:00	6 óra	Kapitány	Szekvencia diagramok elkezdése 3.3.9, 3.3.10 és 3.3.11 elkészítése
2018.02.25 11:00	6 óra	Bende	State chart elkészítése, 3.3.12, 3.3.13 és 3.3.14 elkészítése
2018.02.25 17:00	0.5 óra	Szatmáry Bende Máté Králik Kapitány	Értekezlet: napló elkészítése.
2018.03.04 13:00	1.5 óra	Szatmáry Bende Máté Králik Kapitány	Értekezlet: előző verzió hibáinak javítása, osztálydiagram, objektumkatalógus és osztályleírások frissítése.
2018.03.04 23:00	1.5 óra	Kapitány	Szekvenciadiagramok javítása és kiegészítése

5. Szkeleton tervezése

5.1. A szkeleton modell valóságos use-case-ei

5.1.1. Use-case diagram



5.1.2. Use-case leírások

Use-case neve	Játék indítása
Rövid leírás	A felhasználó elindítja a játékot.
Aktorok	Felhasználó
Forgatókönyv	A felhasználónak lehetősége van kiválasztani elindít-e egy új játékot.

Use-case neve	Karakterek számának beállítása
Rövid leírás	A felhasználó kiválasztja hogy hány karakter játszik
Aktorok	Felhasználó
Forgatókönyv	A felhasználó beállíthatja, hogy hány játékossal együtt szeretné játszani a játékot.

Use-case neve	Játék megnyerése
Rövid leírás	A legtöbb pontot elért felhasználó nyer.
Aktorok	Felhasználó
Forgatókönyv	A játék végén megszámlálódnak a felhasználók pontjai és a legtöbb pontot elért játékos nyer.

Use-case neve	Játék elvesztése
Rövid leírás	A legtöbb pontot elért felhasználó nyer, minden más felhasználó veszít.
Aktorok	Felhasználó
Forgatókönyv	Ha az adott felhasználónak nem a legtöbb pontja van, akkor veszít.

Use-case neve	Karakter mozgatása
Rövid leírás	A felhasználó mozgatja az adott játékost.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó a billentyűzet nyilak segítségével a megadott irányba mozgatja a karaktert.

Use-case neve	Doboz mozgatása
Rövid leírás	Doboz eltolása egy irányba.
Aktorok	Felhasználó, Doboz
Forgatókönyv	A játékos eltolja dobozat, vagy a játékos által tolta doboz eltolja az előtte lévő dobozat.

Use-case neve	Lyukra lépés
Rövid leírás	Megsemmisíti a rá került karaktert vagy dobozat.
Aktorok	Felhasználó vagy Doboz
Forgatókönyv	A lyukra lépő karakter meghal. A lyukba tolta doboz megsemmisül.

Use-case neve	Csapdára lépés
Rövid leírás	Az aktív csapda megöli/megsemmisíti a rákerült dolgot, a nem aktív nem csinál semmit.
Aktorok	Felhasználó vagy Doboz
Forgatókönyv	Egy karakter vagy egy doboz egy csapdára lett mozgatva, ami ha nyitott, az adott karakter vagy doboz megsemmisül, ellenkező esetben nem történik semmi.

Use-case neve	Karakter falnak tolása
Rövid leírás	Egy karakter a falnak ütközik.
Aktorok	Felhasználó, Doboz
Forgatókönyv	A doboz nekinyomja a falnak a karaktert, a karakter meghal, és a karakter helyére kerül a doboz.

Use-case neve	Kapcsoló kapcsolás
Rövid leírás	Doboz kapcsolóra mozgatásakor/elhagyásakor átkapcsolja azt.
Aktorok	Doboz
Forgatókönyv	A doboz úgy lett mozgatva hogy az vagy rálépett és bekapcsolta, vagy lelépett és kikapcsolta a kapcsolót.

Use-case neve	Meghalás
Rövid leírás	Karakter halála, vagy doboz megsemmisülése.
Aktorok	Felhasználó vagy Doboz
Forgatókönyv	Lyukba vagy aktív csapdára lépve meghal a karakter, míg lyukra, csapdára vagy célba érve megsemmisülése a doboz.

Use-case neve	Célba érés
Rövid leírás	Doboz célba ér, majd meghal.
Aktorok	Doboz
Forgatókönyv	Célba ér a doboz, a játékos pontot kap, a doboz megsemmisül.

Use-case neve	Pontszerzés
Rövid leírás	A felhasználó pontot kap.
Aktorok	Felhasználó
Forgatókönyv	Ha a felhasználó célba tolta egy dobozt, akkor azért kap egy pontot.

5.2. A szkeleton kezelői felületének terve, dialógusok

Parancssoros, menüvezérelt módon. Az egyes funkciók eléréséhez a tesztelőnek a megfelelő számokat vagy betűket kell megadnia (majd enter nyomni), a megfelelő sorrendben, amely alább látható. Ezáltal elindul egy tesztelési szekvencia, amely végén a tesztelő automatikusan visszatér a főmenübe, ahonnan újabb szekvenciát indíthat el.

(0) Játék indítása megadott játékos számmal

Hány játékos vesz részt?

(1/2/3/4...)

(1) Karakter mozgatása

A karakter dobozt tol / lyukra lép / csapdára lép / üres mezőre lép / falnak megy?

(1) Karakter dobozt tol?

(2) Karakter lyukra lép?

(3) Karakter csapdára lép?

(1) Aktív a csapda

(2) Inaktív a csapda

(4) Karakter üres mezőre lép?

(5) Karakter falnak megy?

(2) Doboz mozgatása.

Doboz falnak megy / dobozt tol / játékost tol / lyukra lép / csapdára lép / kapcsolóra lép / célra lép?

(1) Doboz falnak megy?

(2) Doboz dobozt tol?

(3) Doboz lyukra lép?

(4) Doboz csapdára lép?

(1) Aktív a csapda

(2) Inaktív a csapda

(5) Doboz kapcsolóra lép?

(1) Aktív a csapda

(2) Inaktív a csapda

(1) Doboz van a csapdán

(2) Karakter van a csapdán

(3) Üres a csapda

(6) Doboz célra lép?

(7) Doboz karaktert tol?

(8) Doboz üres mezőre lép?

(3) Thing lehelyezése

(1) Karakter lehelyezése érvényes mezőre?

(1) Igen

(2) Nem

(2) Dobozok lehelyezése érvényes mezőre

(1) Igen

(2) Nem

(4) Karakter pontot szerez

Pontot szerző karakter száma

(1/2/3/4...)

Kaphat-e az adott játékos pontot? (halott-e?)

(1) Igen

(2) Nem

(7) Nyerés, vesztés

Nyerő játékos száma (mindenki más vesztett)

(1/2/3/4...)

(8) Programból való kilépés?

----KIMENET TERV----

<főmenü tartalma kiírva>

----->Választott parancs: 2 • Doboz mozgatása

>>Választott parancs: 5 • Doboz kapcsolóra lép

>>>Választott parancs: 2 • Inaktív a kapcsoló

>>>>Választott parancs: 1 • Doboz van a csapdán

----->[class:Switch].action(Box b1)

→[class:Trap].switchTrapState()

→[class:Trap].isEmpty()

<←[class:Trap].isEmpty()

→[class:Trap].action(Box b2)

→[class:Trap]:removeThing()

<←[class:Trap]:removeThing()

<←[class:Trap].action(Box b2)

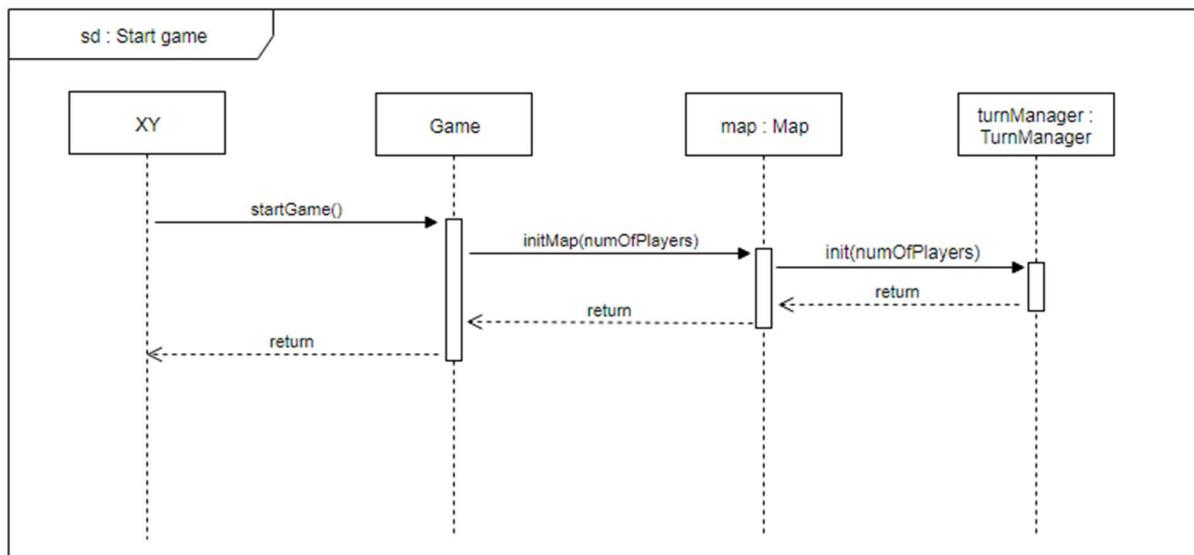
<←[class:Switch].switchTrapState()

<←[class:Switch].action(Box b1)

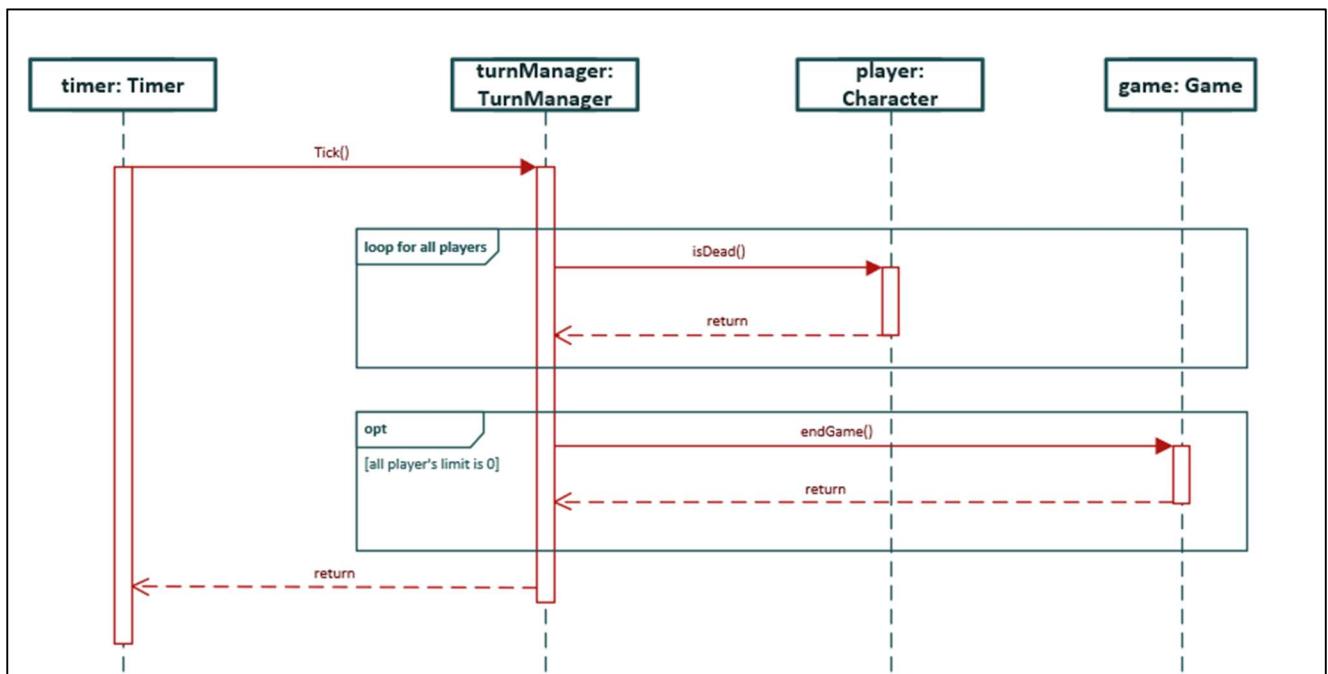
<főmenü tartalma kiírva>

5.3. Szekvenciadiagramok a belső működésre

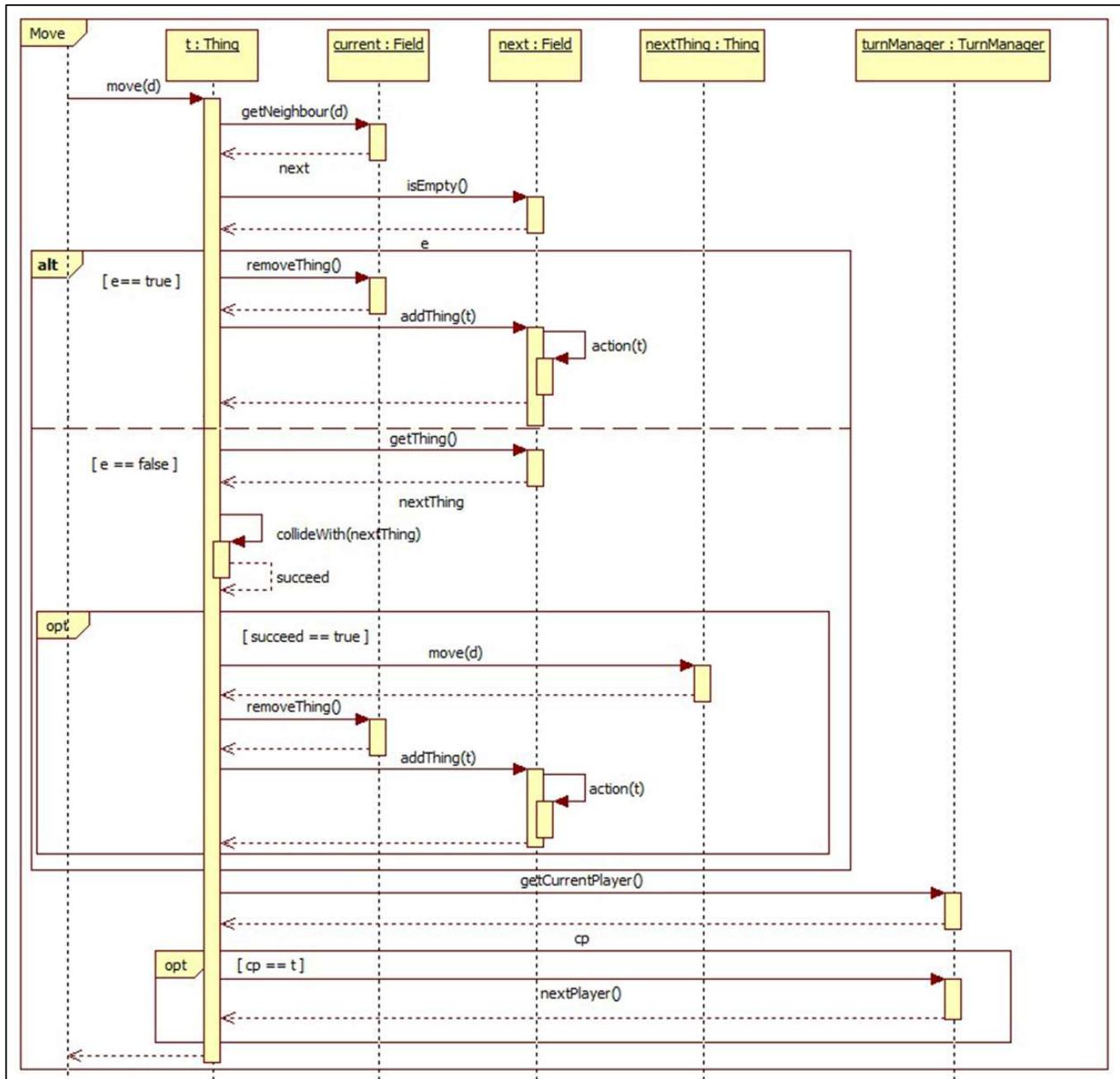
5.3.1. Játék indítása / karakterek számának beállítása



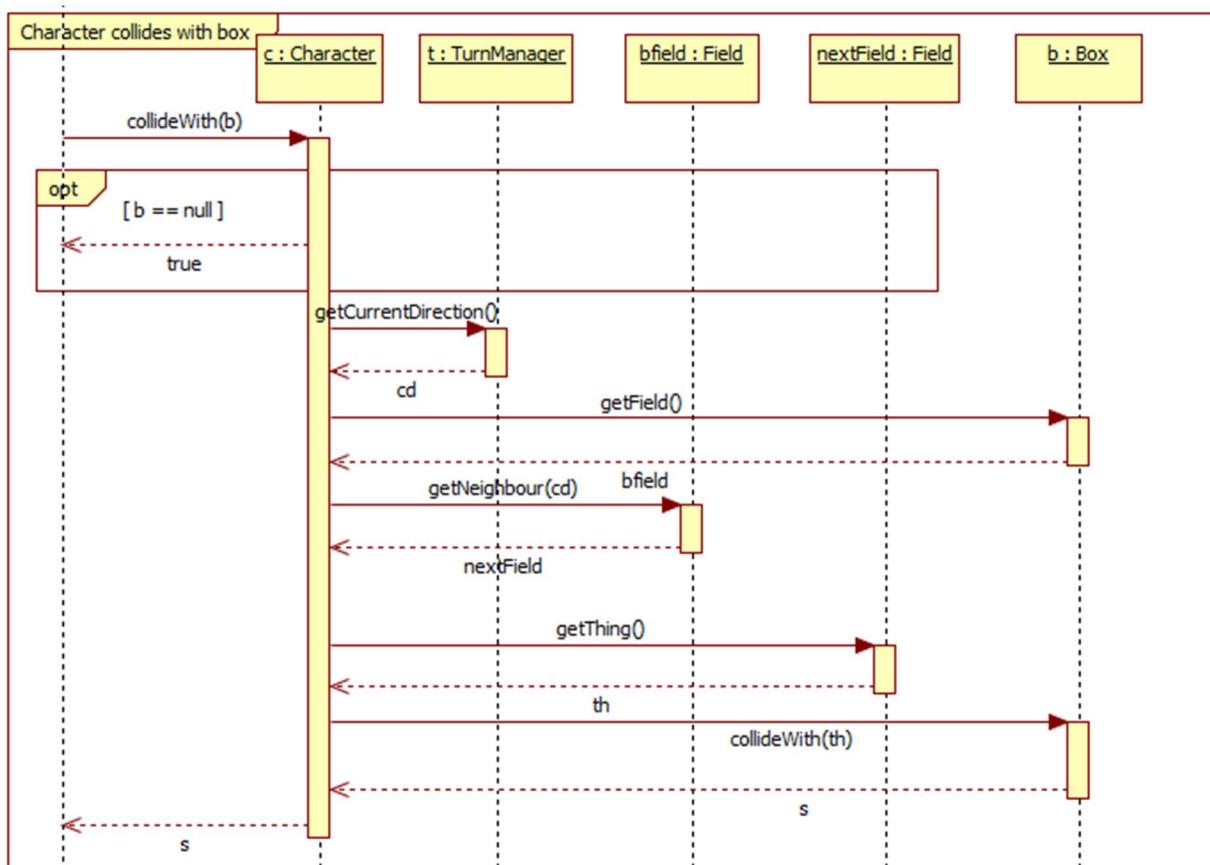
5.3.2. Játék vége (megnyerése / elvesztése)



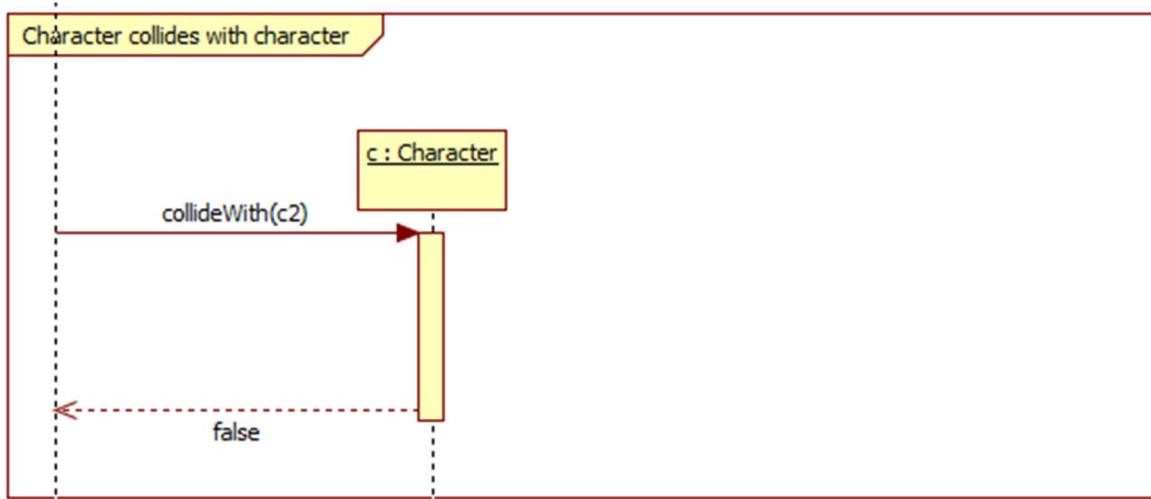
5.3.3. Karakter mozgása



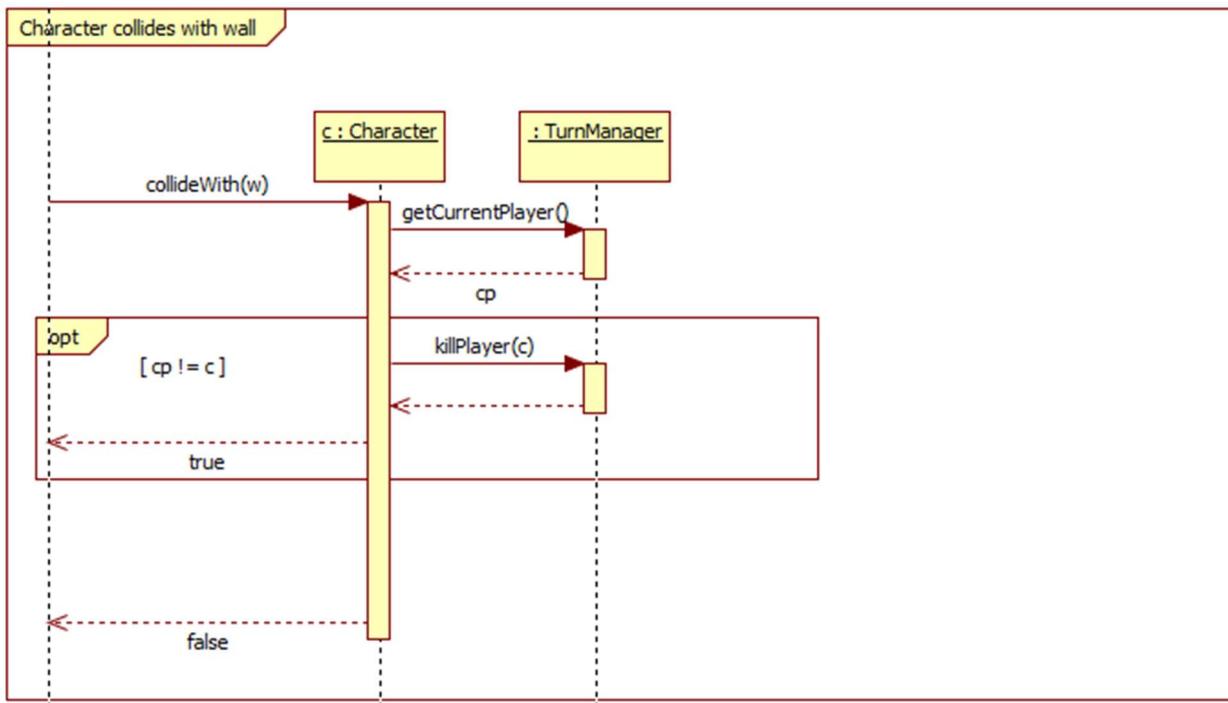
5.3.4. Karakter dobozzal ütközik



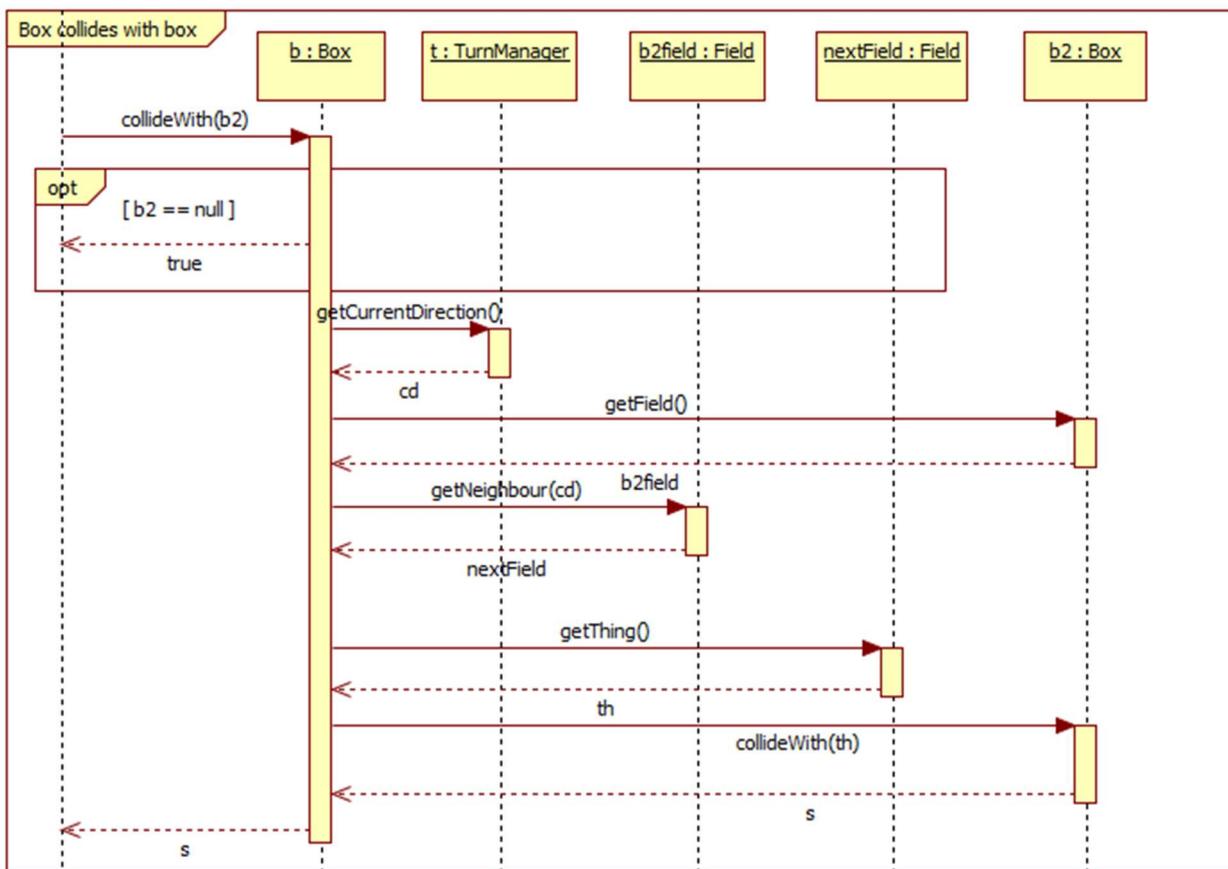
5.3.5. Karakter karakterrel ütközik



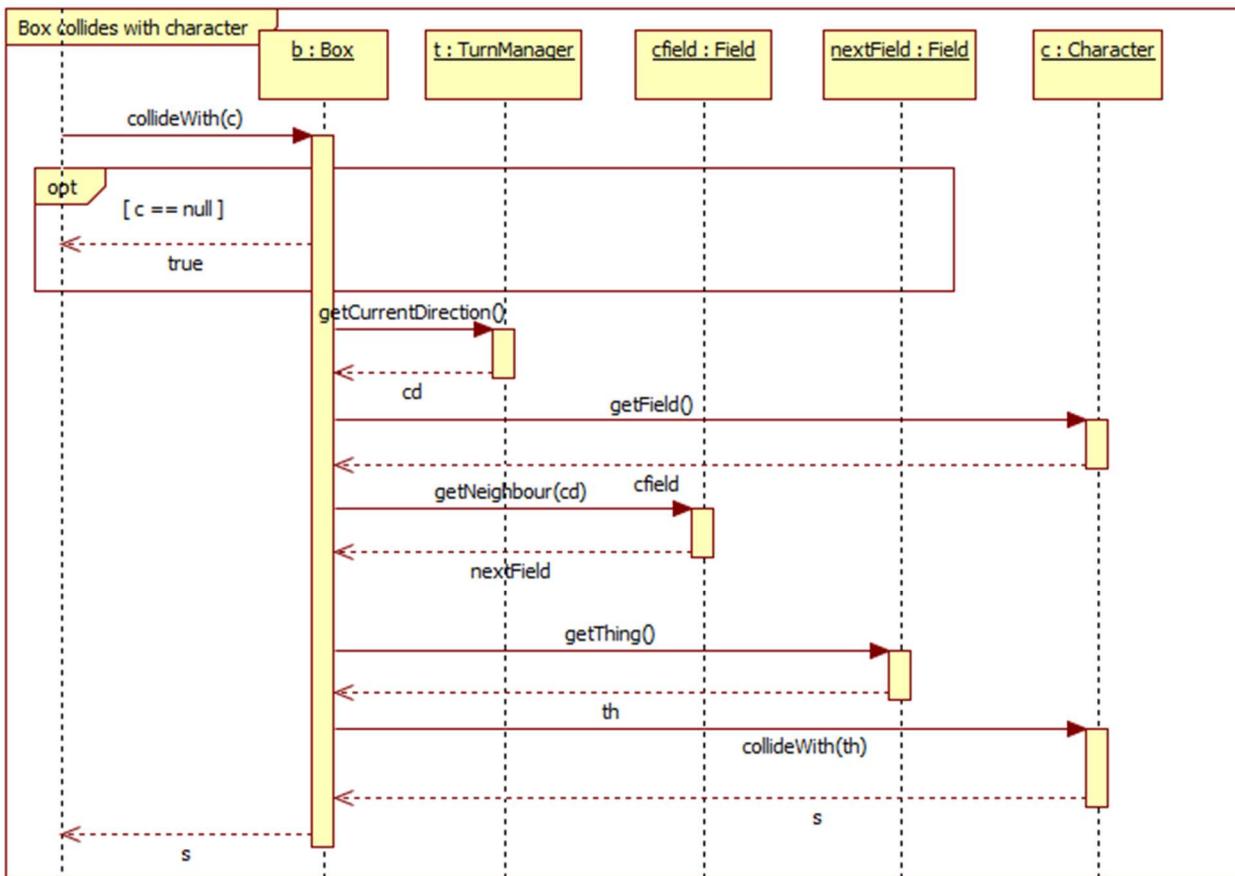
5.3.6. Karakter falnak ütközik



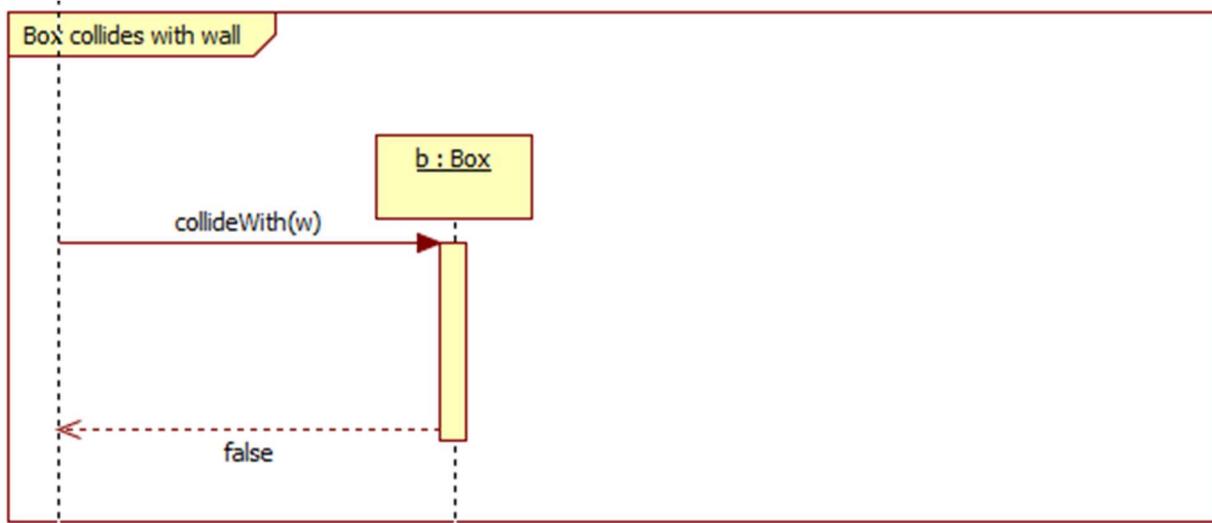
5.3.7. Doboz dobozzal ütközik



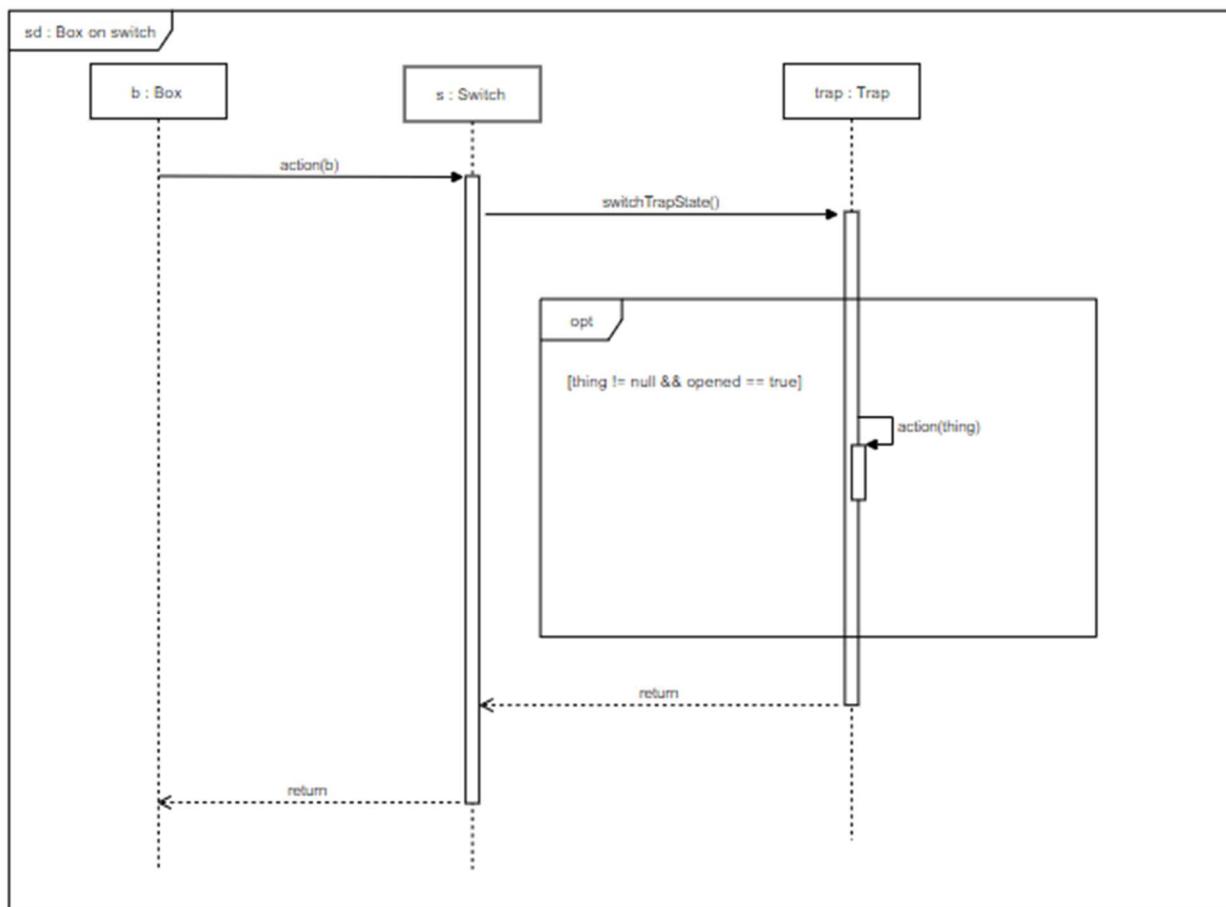
5.3.8. Doboz karakterrel ütközik



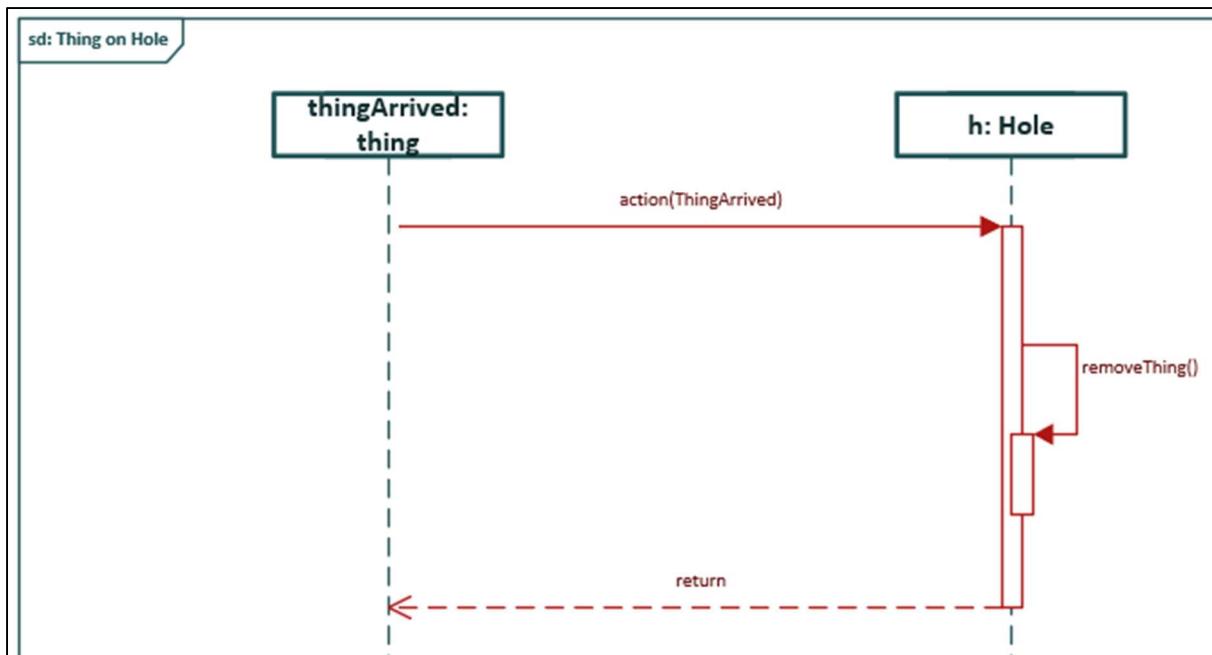
5.3.9. Doboz fallal ütközik



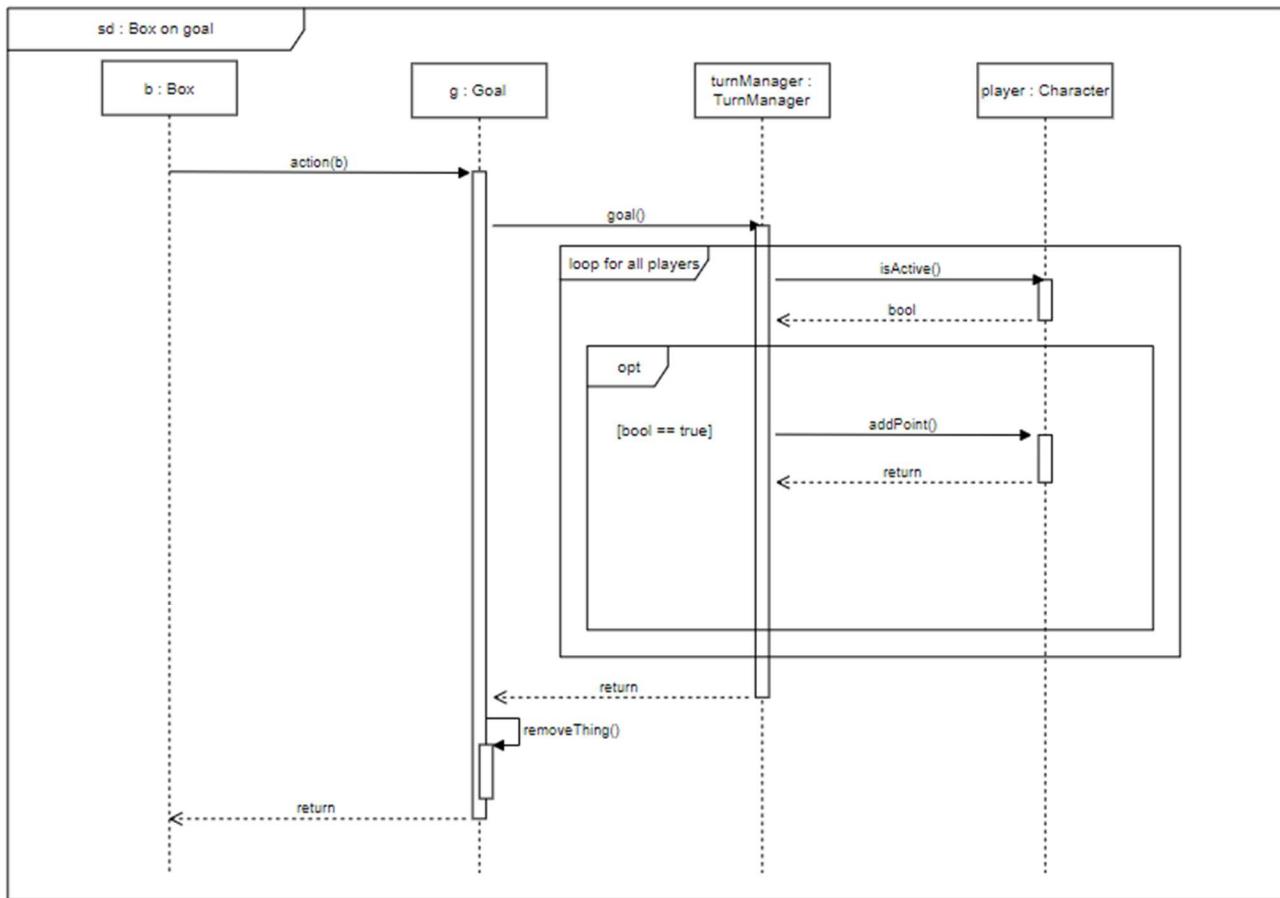
5.3.10. Doboz kapcsolót kapcsol



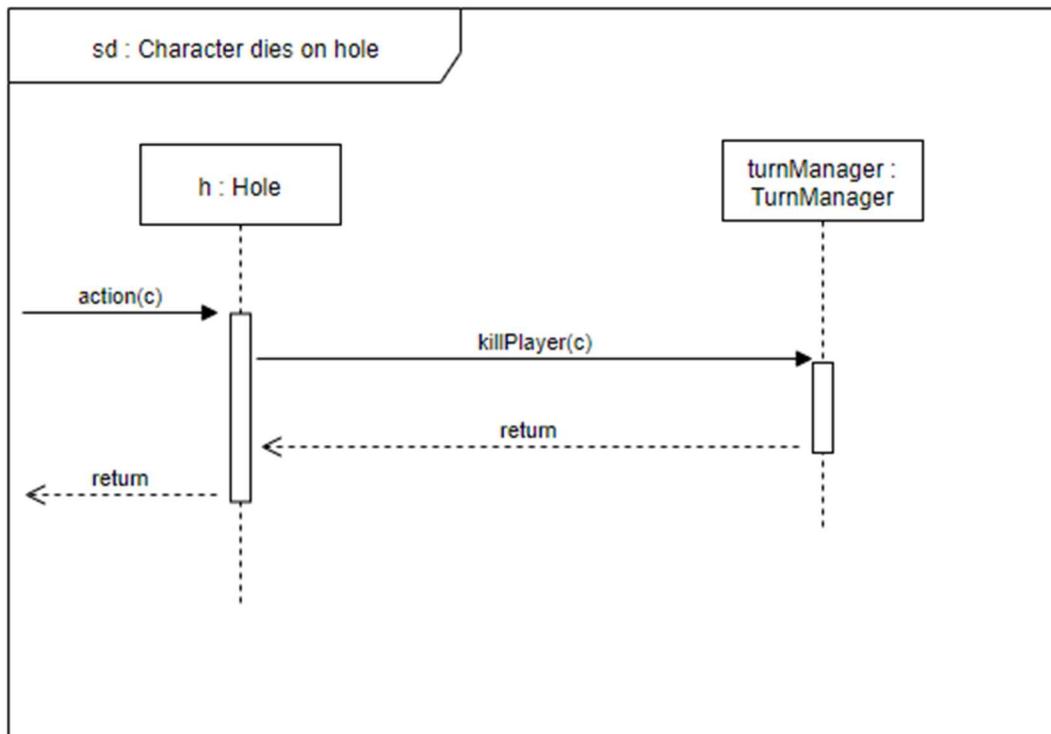
5.3.11. Doboz lyukra lép



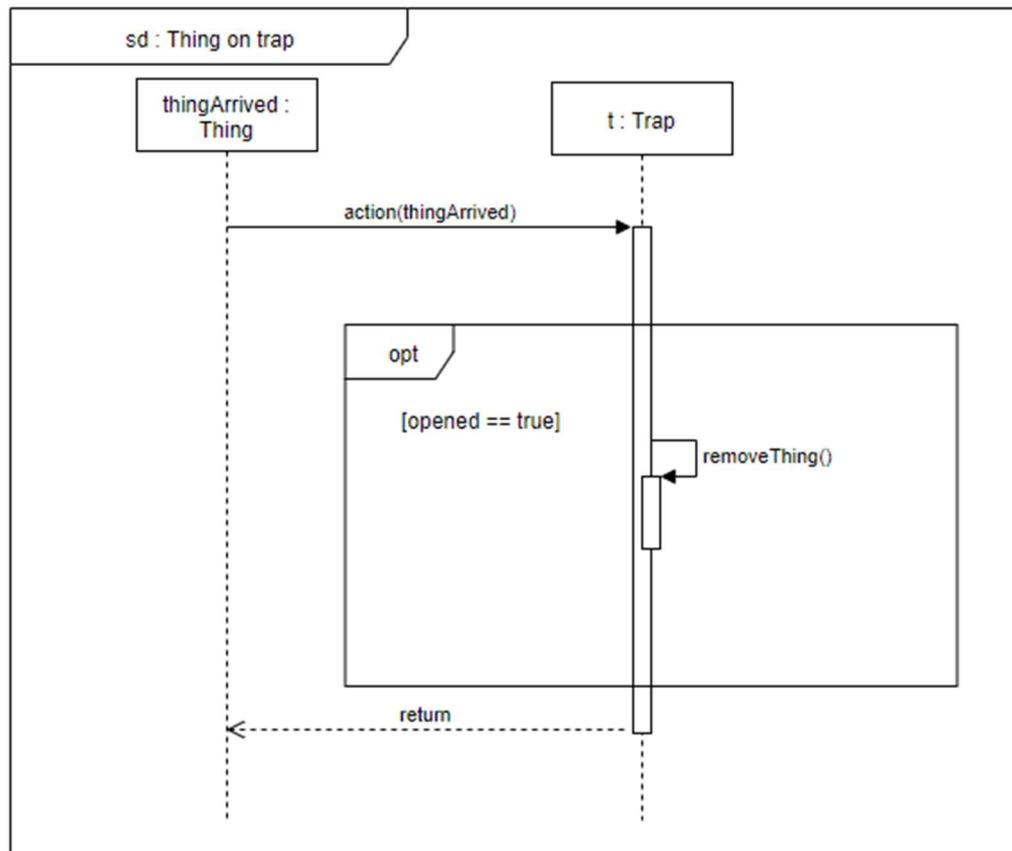
5.3.12. Doboz célba ér / pontszerzés



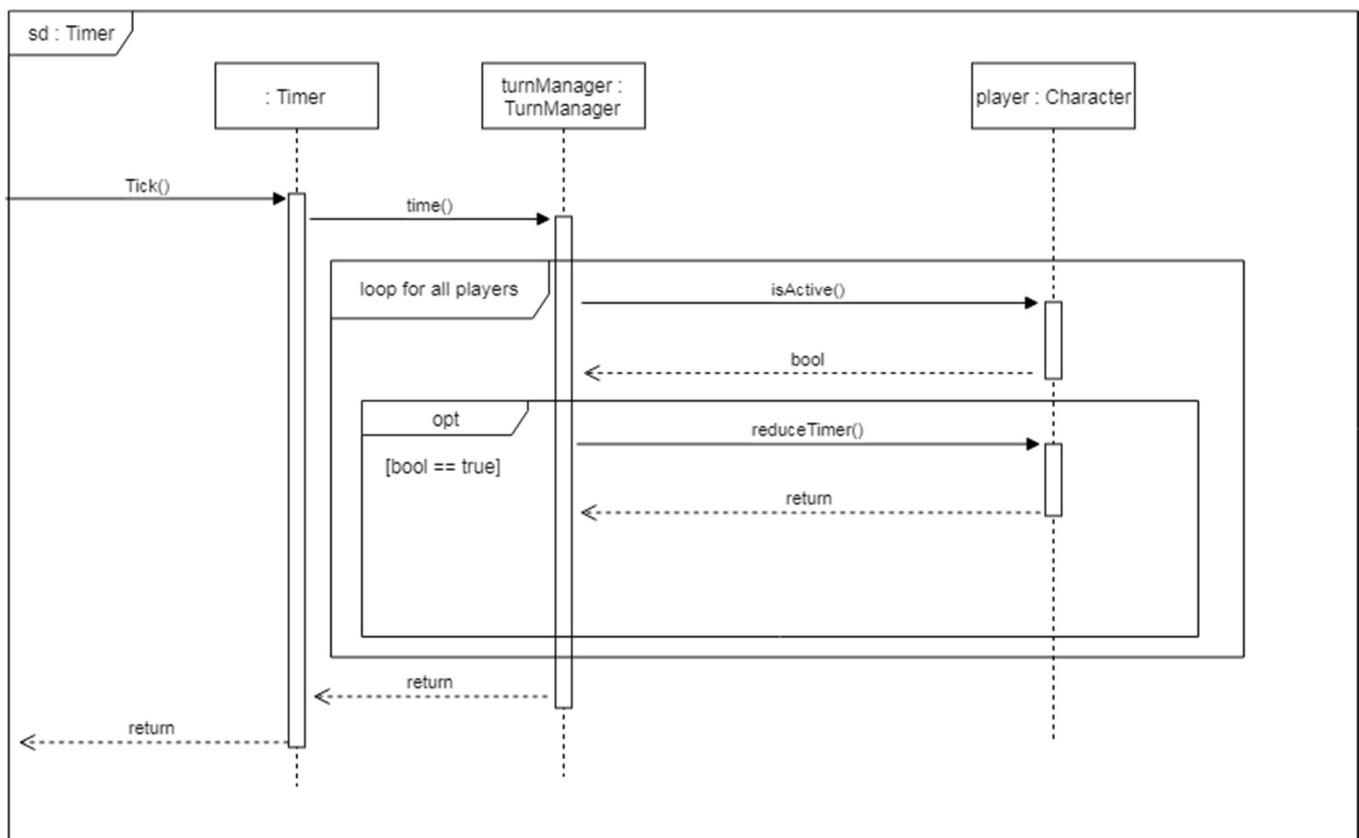
5.3.13. Karakter meghal



5.3.14. Doboz meghal

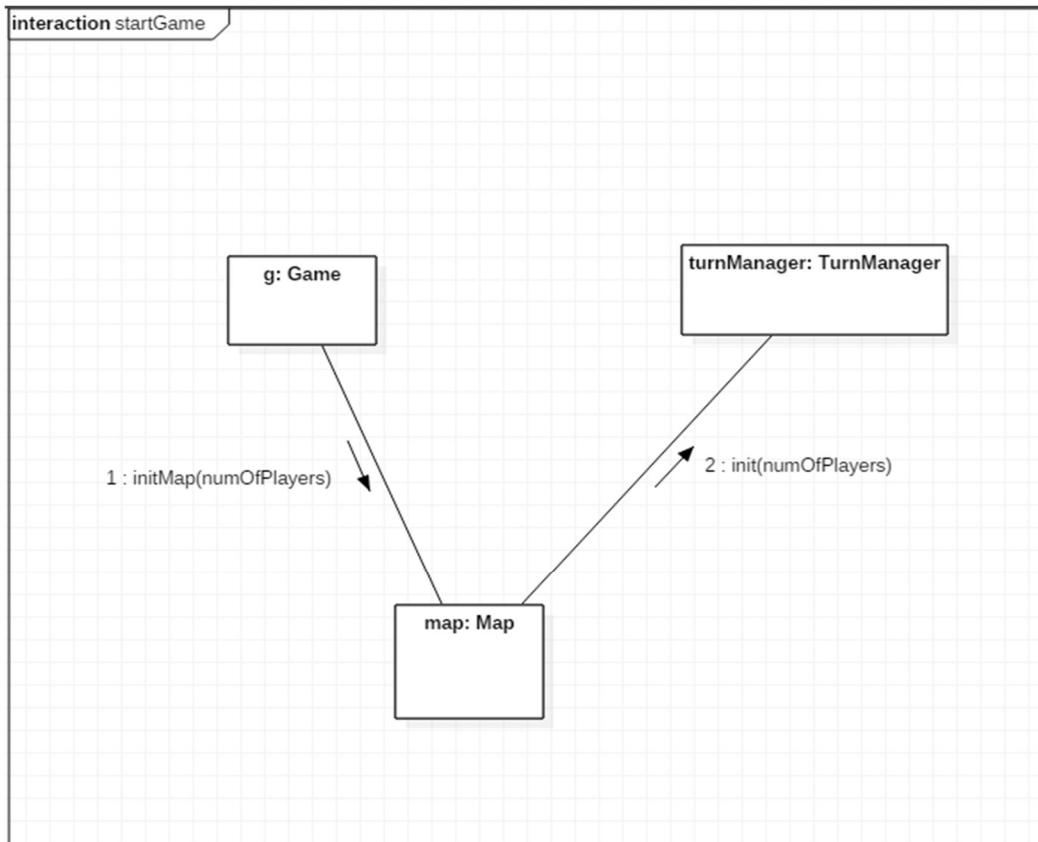


5.3.15. Timer (időzítő)

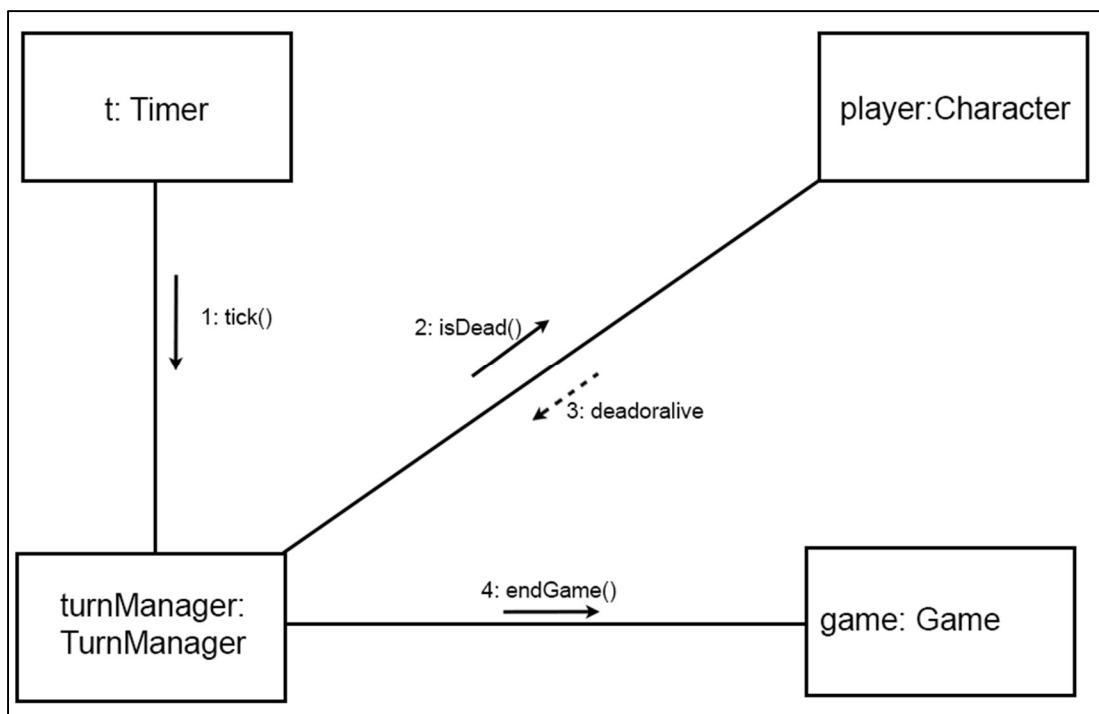


5.4. Kommunikációs diagramok

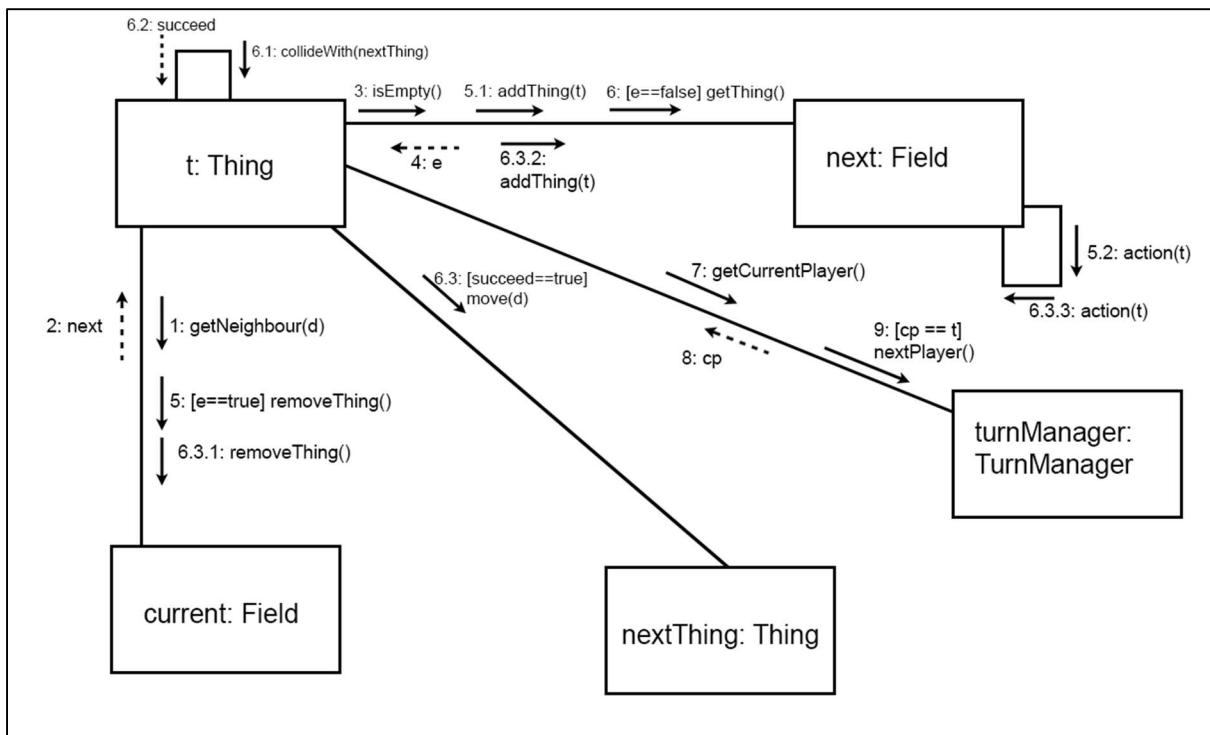
5.4.1. Játék indítása / karakterek számának beállítása



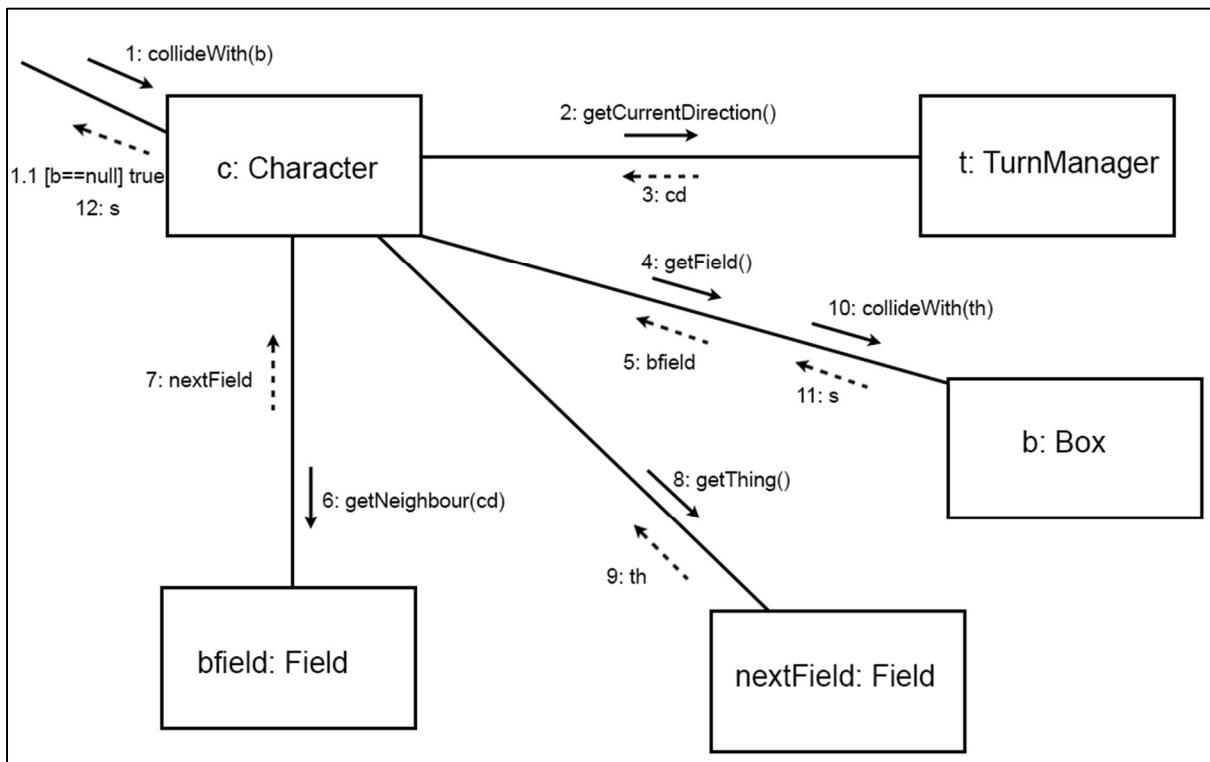
5.4.2. Játék vége (megnyerése / elvesztése)



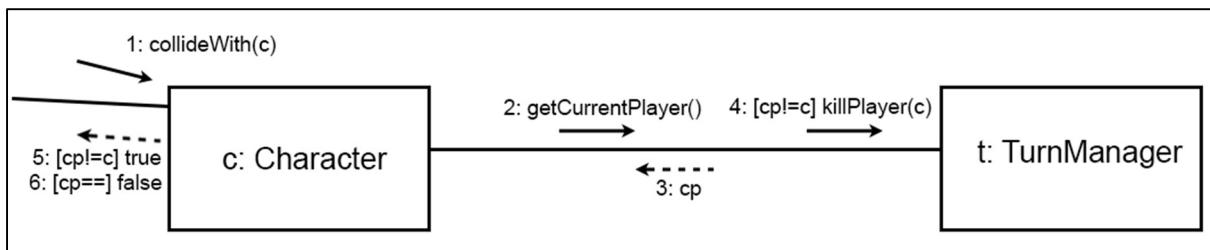
5.4.3. Karakter mozgása



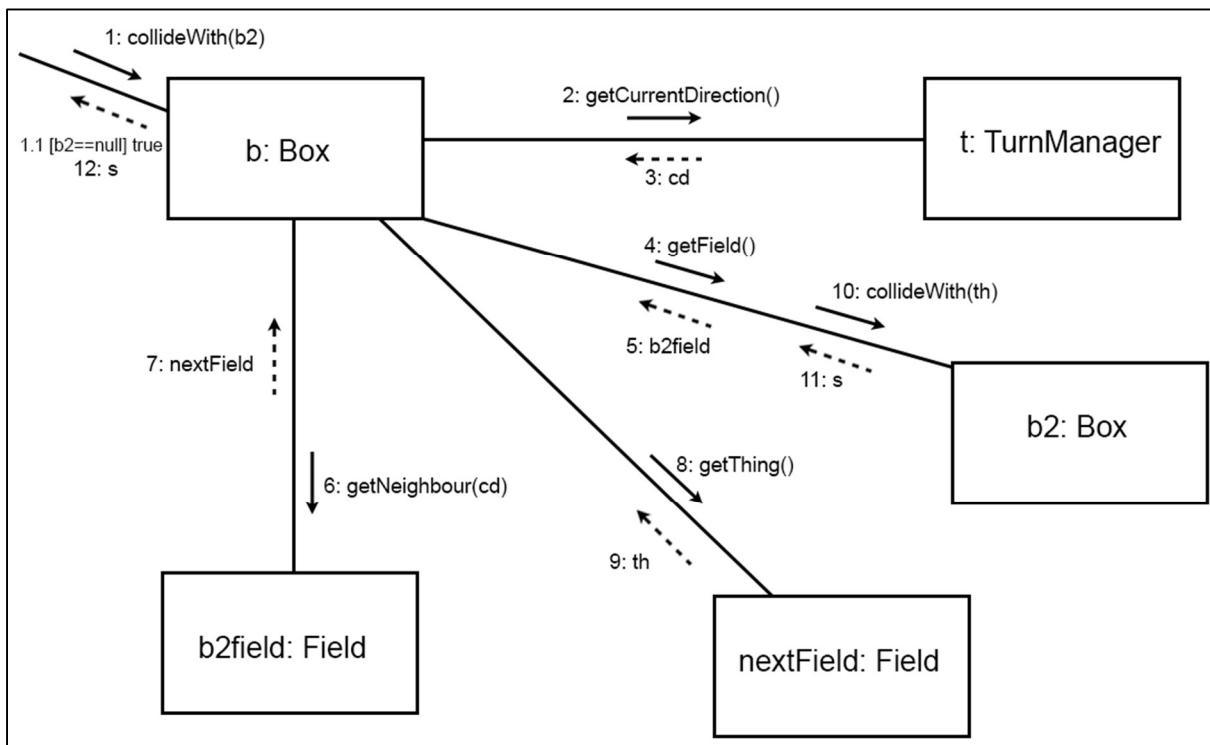
5.4.4. Karakter dobozzal ütközik



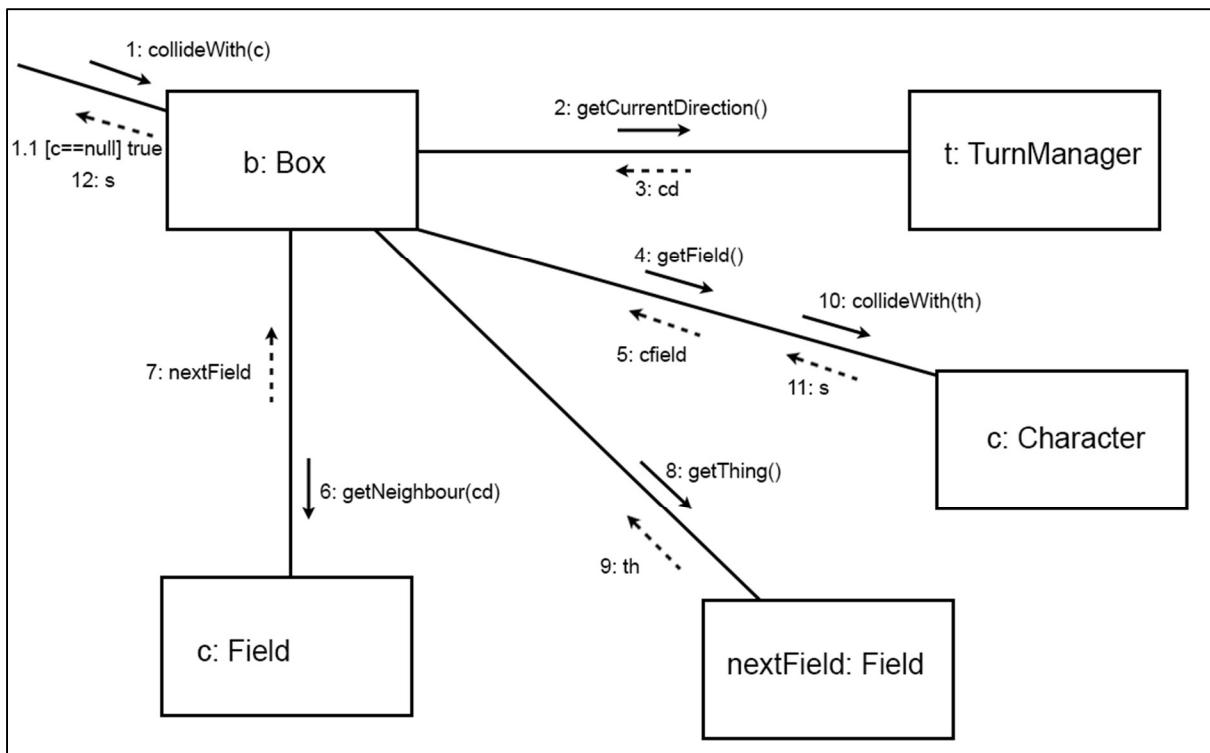
5.4.5. Karakter falnak ütközik



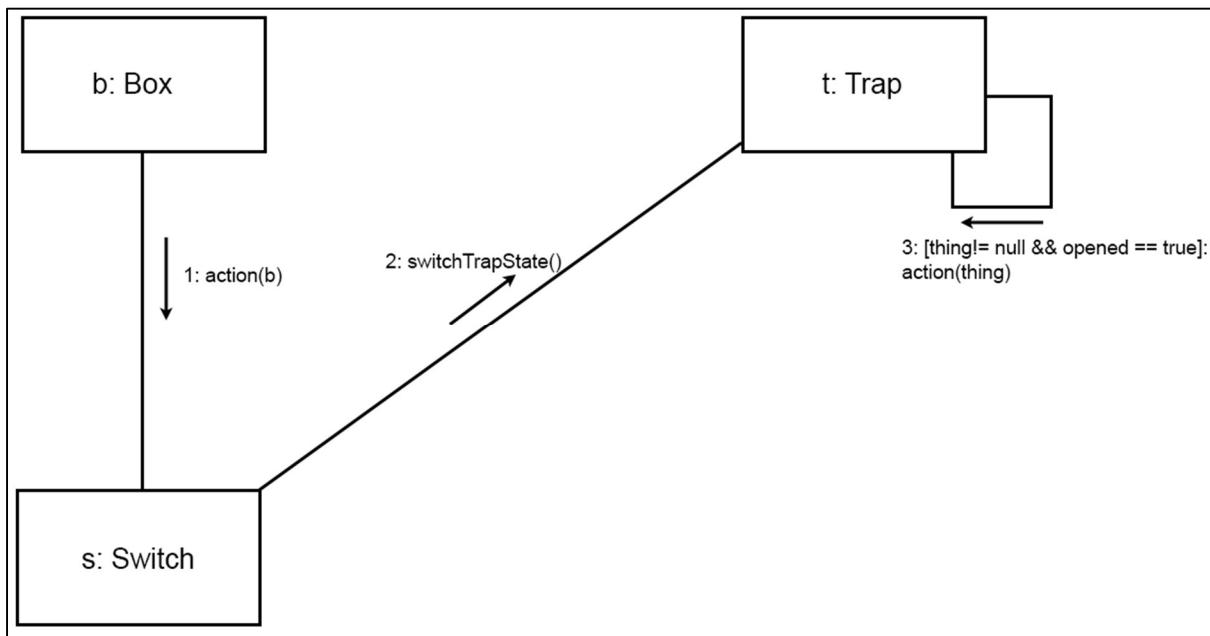
5.4.6. Doboz dobozzal ütközik



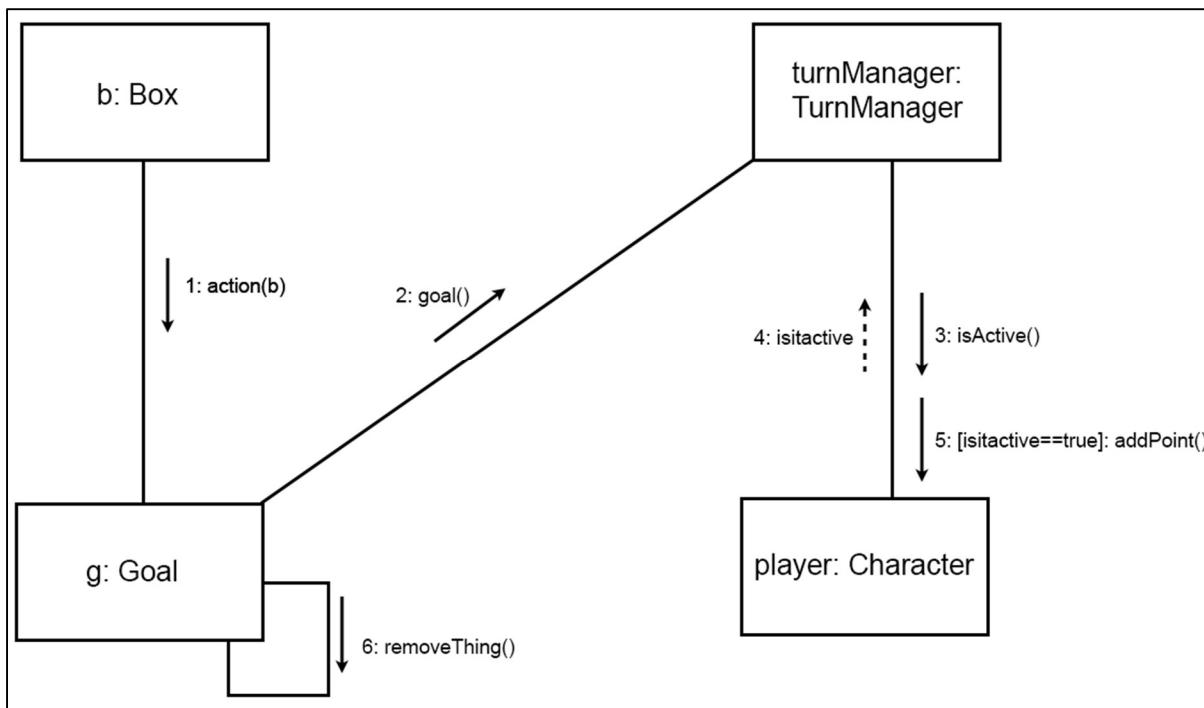
5.4.7. Doboz karakterrel ütközik



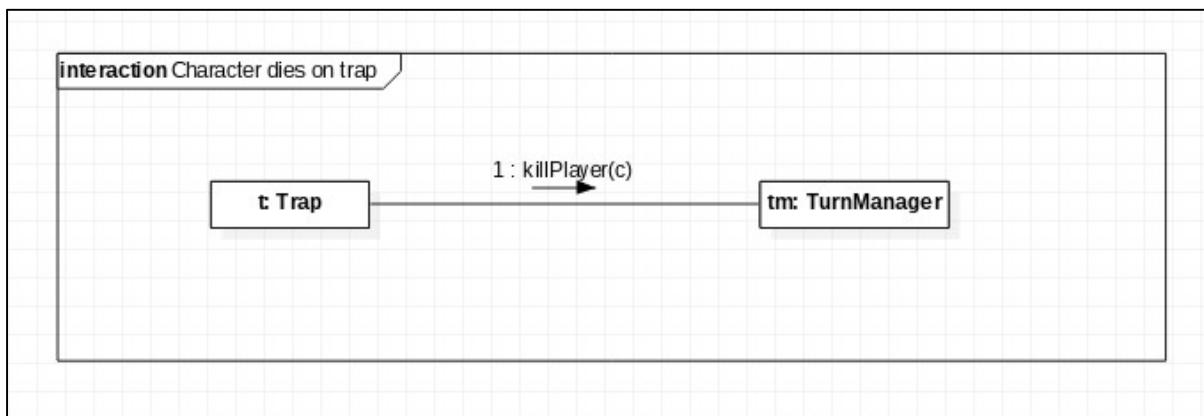
5.4.8. Doboz kapcsolót kapcsol



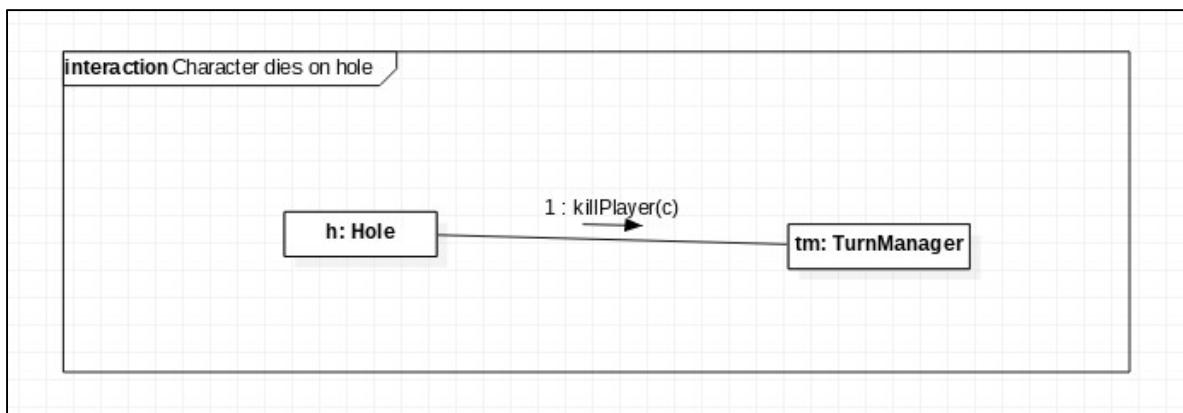
5.4.9. Doboz célba ér / pontszerzés



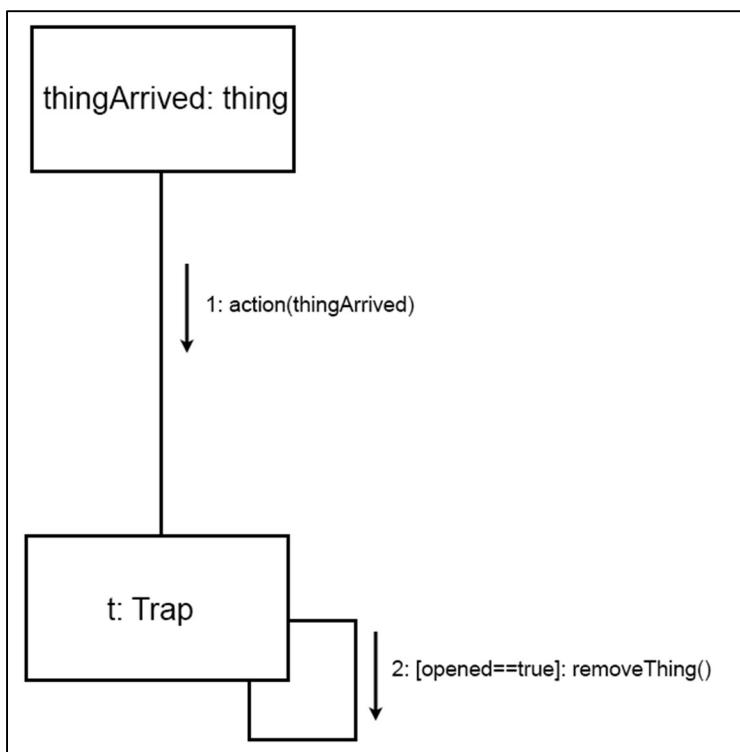
5.4.10. Karakter meghal a csapdán



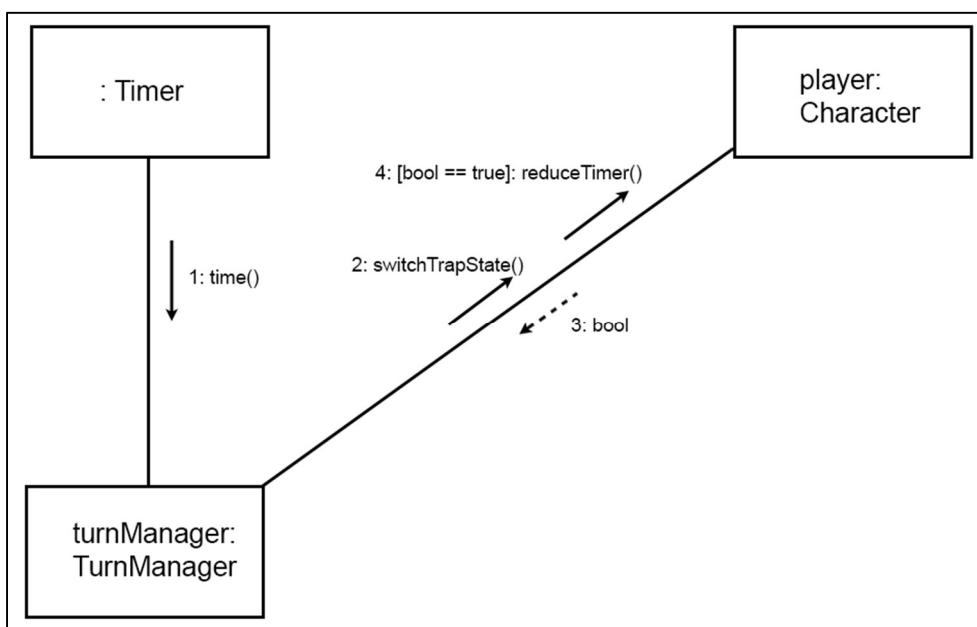
5.4.11. Karakter meghal a lyukon



5.4.12. Doboz meghal a csapdán



5.4.13. Timer (időzítő)



5.5. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.03.10 17:00	6 óra	Králik Bende Szatmáry Máté Kapitány	Értekezlet: Analízis modell hibáinak elemzése, átbeszélése, javítása. Megoldási terv készítése, feladatok kiosztása, use-case diagram tervezése (v1)
2018.03.10 22:00	2 óra	Králik Máté	Értekezlet: Új mozgatási algoritmus (v1) tervezése.
2018.03.11 10:00	1.5 óra	Bende	Use-case diagram elkészítése (végleges)
2018.03.11 10:00	2 óra	Kapitány	Use-case leírások elkészítése
2018.03.11 9:30	1 óra	Szatmáry	Szkeleton felületi terv
2018.03.11 12:00	4.5 óra	Králik Bende Szatmáry Máté Kapitány	Értekezlet: diagramtípusok tanulmányozása, részletes megoldási terv átbeszélése, brainstorming
2018.03.11 17:00	6 óra	Králik Máté	Értekezlet: Szekvencia diagramok megírása
2018.03.11 17:00	5.5 óra	Bende	Kommunikációs diagramok megírása
2018.03.11 17:00	6 óra	Kapitány	Szekvencia diagramok megírása (mozgás, ütközések)
2018.03.11 17:00	5 óra	Szatmáry	Kommunikációs diagramok megírása (mozgás, ütközések)
2018.03.12 10:00	2 óra	Szatmáry	Hibák javítása, formázás, diagramok korrekciói

6. Szkeleton beadás

6.1. Fordítási és futtatási útmutató

A program futtatásához szükségeltetik egy legalább 1.7-es JAVA fejlesztői környezettel (JDK) és futtatói környezettel (JRE) rendelkező személyi számítógép. Továbbá a program gondmentes futtatásához ajánlott Windows, vagy GNU/Linux operációs rendszerrel, és legalább 2GB RAM-mal, Sandy Bridge alapú 2. generációs Intel Core processzorral, és 1280*720 pixel felbontású monitorral felszerelt konfiguráció.

Eclipse fejlesztői környezet megléte különösen ajánlott, de nem feltétel.

6.1.1. Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Box.java	2765 byte	2018.03.16	A pályán tologatható dobozok Box osztálya.
Character.java	5960 byte	2018.03.16	A játékos által közvetlenül irányított karakterek Character osztálya.
Collidable.java	164 byte	2018.03.16	Az ütközésre képes Thing osztályokat összekapcsoló Interface.
Direction.java	110 byte	2018.03.15	Azon irányok felsorolása, amerre a pályán Thing-ek mozoghatnak.
Field.java	2136 byte	2018.03.15	Absztrakt ősosztály a pálya mezőire. Ilyen mező objektumokra kerülhetnek rá a Thing-ek.
Game.java	13259 byte	2018.03.18	Game osztály. A program belépési pontja.
Goal.java	721 byte	2018.03.16	Cél osztály, cél pályaelemre kell tolni a dobozokat.
Hole.java	695 byte	2018.03.16	Lyuk osztály.
Map.java	686 byte	2018.03.15	A játék színtere, Map osztály.
NormalField.java	284 byte	2018.03.16	Az üres mezőt jelképező NormalField osztály.
Switch.java	684 byte	2018.03.16	A csapda (Trap) be, illetve kikapcsolásáért felelős Switch osztály.
Thing.java	938 byte	2018.03.16	Absztrakt ősosztály az egyes Field mezőkre rákerülhető dolgoknak.
Timer.java	854 byte	2018.03.15	A játékosok idejeinek csökkentéséért felelős Timer osztály.
Trap.java	1649 byte	2018.03.16	A pályán megjelenő csapdák Trap osztálya.
TurnManager.java	5514 byte	2018.03.15	A körök és lépésekkel menedzseléséért felelős TurnManager osztály.
Wall.java	985 byte	2018.03.16	Mozdíthatatlan fal elemek Wall osztálya.

6.1.2. Fordítás

Eclipse vagy egyéb Java fejlesztői környezetben a projekt megnyitása, avagy új projekt létrehozása, és az összes a projekthez tartozó forrásfájlok hozzáadása után a projekt/Build All (CTRL+B) menüpont lefuttatása.

Alternatívaként a 6.1.3 pont szerinti útmutató követése, mivel Eclipse alatt a run gomb egyben fordít is.

6.1.3. Futtatás

Eclipse vagy egyéb fejlesztői környezetben a már lefordult program elindítása a Run gomb (tipikusan kerek, zöld, kicsiny háromszöggel a közepében) lenyomása által.

A program elindítása után a parancssoron vár számokat bemenetként, amivel a megfelelő tesztelési menüpontok kiválaszthatóak.

6.2. Értékelés

Tag neve	Munka százalékban
Bende	19,67%
Szatmáry	21,51%
Máté	23,79%
Králik	18,76%
Kapitány	16,24%

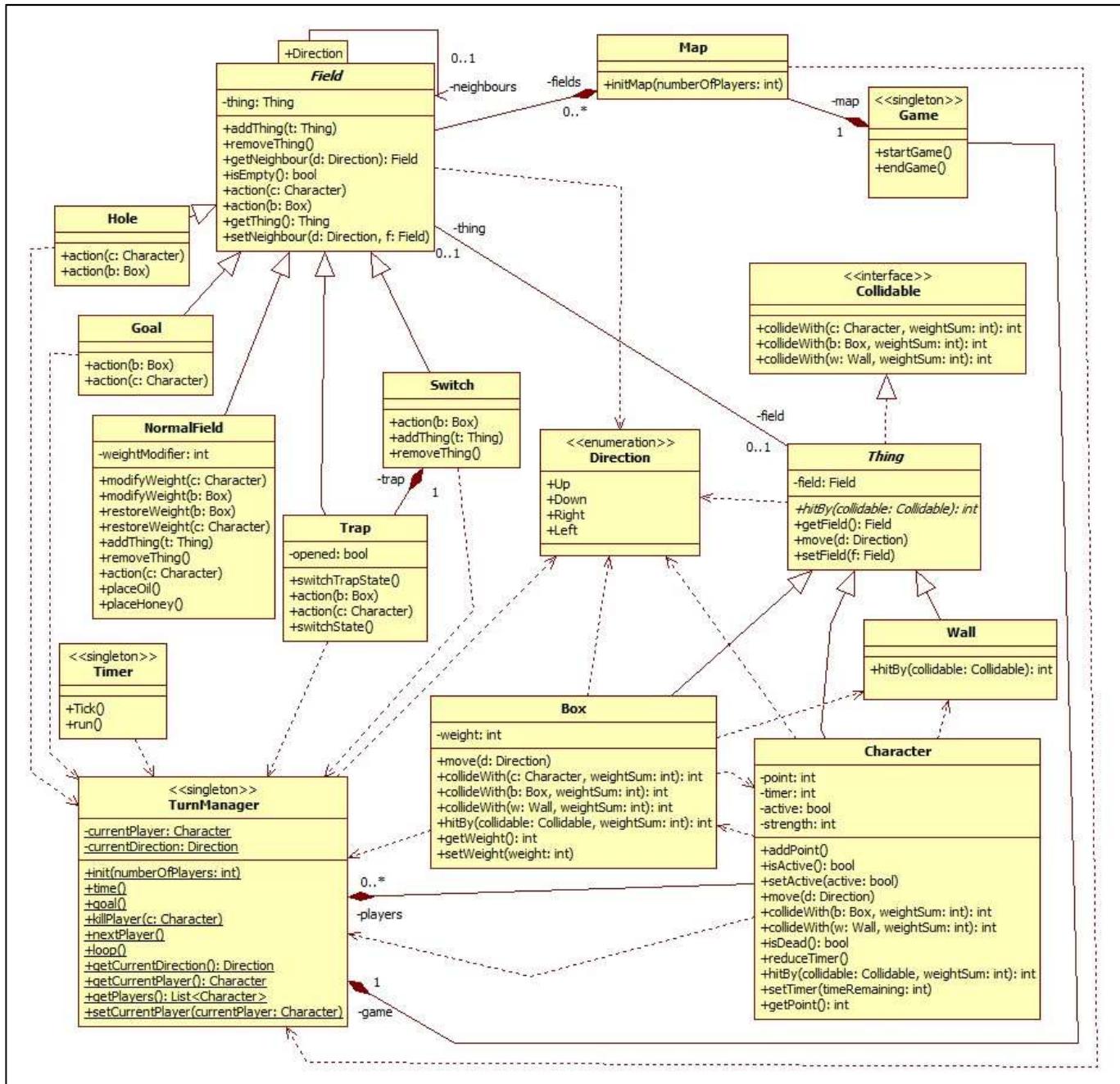
6.3. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.03.14 21:00	2 óra	Bende Szatmáry Máté Králik Kapitány	Értekezlet: Heti teendők megvitatása, végrehajtási terv elkészítése.
2018.03.15 14:00	1 óra	Bende Králik	Értekezlet: kör vezérlő logikai kód elkészítése.
2018.03.15 20:00	6 óra	Máté	Oszályok programkódjainak megírása.
2018.03.18 23:00	2 óra	Szatmáry	Kommentek, leírások elkészítése a beadandó munkához.
2018.03.18 20:00	1 óra	Kapitány	Game, szkeleton teszt elkészítése
2018.03.19: 10:00	2 óra	Máté Szatmáry	Értekezlet: utolsó simítások, korrektúrázások, jegyzőkönyv elkészítése.

7. Prototípus koncepciója

7.0. Változás hatása a modellre

7.0.1. Módosult osztálydiagram



Az osztályokhoz és a nevesített asszociációvégekhez implicit getter és setter függvények tartoznak, amelyeket a diagram az olvashatóság kedvéért nem minden esetben jelöl.

7.0.2. Új vagy megváltozó metódusok

Thing absztrakt osztály:

public abstract int hitBy(Collidable collidable, int weightSum);

Akkor hívódik meg, ha az adott Thing ütközik egy másik Collidable interfacet megvalósító objektummal. Visszatérési értéke annak az ütközés súlyának értéke.

Collidable interface:

public int collideWith(Character c, int weightSum);

Collidable interfacet megvalósító Thing karakterrel ütközik. Visszatérési értéke függ az ütközésben résztvevő Thing-ek típusától, és súlyától.

public int collideWith(Box b, int weightSum);

Collidable interfacet megvalósító Thing dobozzal ütközik. Visszatérési értéke függ az ütközésben résztvevő Thing-ek típusától, és súlyától.

public int collideWith(Wall w, int weightSum);

Collidable interfacet megvalósító Thing fallal ütközik. A fal (Wall) mindenképp mozdíthatatlan.

Field absztrakt osztály:

public void placeOil();

A Character placeOil() függvénye hívja meg, az éppen aktuális Field (NormalField) placeOil() függvényét, ezzel csökkentve a weightModifier tagváltozójának értékét.

public void placeHoney();

A Character placeHoney() függvénye hívja meg, az éppen aktuális Field (NormalField) placeHoney() függvényét, ezzel csökkentve a weightModifier tagváltozójának értékét.

Character osztály:

public void placeHoney();

A játékos által irányított karakter mézet lerakó függvénye. minden játékosnak előre meghatározott darabszámu méze van, ezeket NormalField mezőre rakhatja le, ezáltal pozitív irányba változtatva annak a weightModifier koefficiensének értékét.

public void placeOil()

A játékos által irányított karakter olajat lerakó függvénye. minden játékosnak előre meghatározott darabszámu olaja van, ezeket NormalField mezőre rakhatja le, ezáltal negatív irányba változtatva annak a weightModifier koefficiensének értékét.

public void move(Direction d)

A játékos karakterét mozgató függvény. Ha az általa meghívott mozgatások és ütközések sorozatának szummája nem éri el a játékosra jellemző erő (strength) változó értékét, akkor garantáltan meghiúsul a mozgás.

Box osztály:

public void move(Direction d)

A dobozt mozgató függvény. Tárolja és viszi tovább a mozgatási együtthatók kiszámításáért és a következő mező ellenőrzéséért felelős változókat.

Mozgatását csak a falba ütközés állítja meg.

NormalField osztály:

```
public void modifyWeight(Box b);
```

Az aktuális NormalField (saját) értékét változtatja meg, hozzáadva a weightModifier tagváltozójához a rajta lévő doboznak a súlyát.

```
public void modifyWeight(Character c);
```

Az aktuális NormalField (saját) értékét "változtatja meg", hozzáadva a weightModifier tagváltozójához a rajta levő karakter súlyát, ami 0.

A karaktereknek nincsen súlya.

```
public void restoreWeight(Box b);
```

Az aktuális NormalField (saját) értékét változtatja meg akkor, amikor lekerül a mezőről egy Box, levonva az annak eddig hozzáadott súlyát a NormalField weightModifier változójának értékéből.

```
public void restoreWeight(Character c);
```

Az aktuális NormalField (saját) értékét "változtatja meg" akkor, amikor lekerül a mezőről egy Character, levonva az annak eddig "hozzáadott" súlyát a NormalField weightModifier változójának értékéből. A karaktereknek nincsen súlya.

```
public void addThing(Thing t);
```

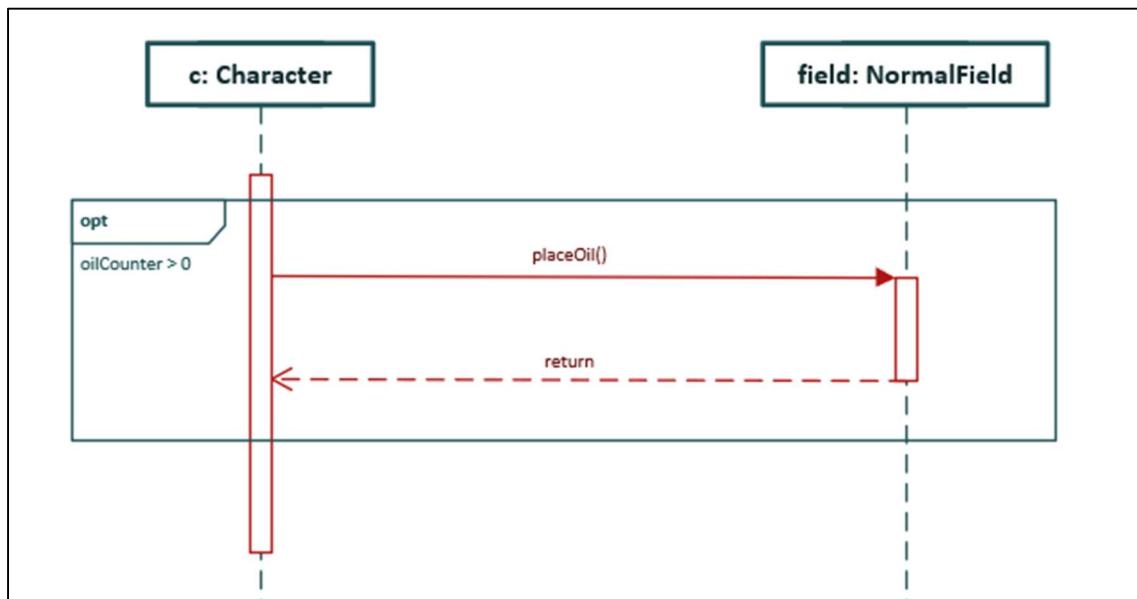
Rákerül egy NormalField-re egy Thing, beállítja annak a helyét erre a mezőre, majd meghívja a súlyokat hozzáadó és számoló függvényt.

```
public void removeThing();
```

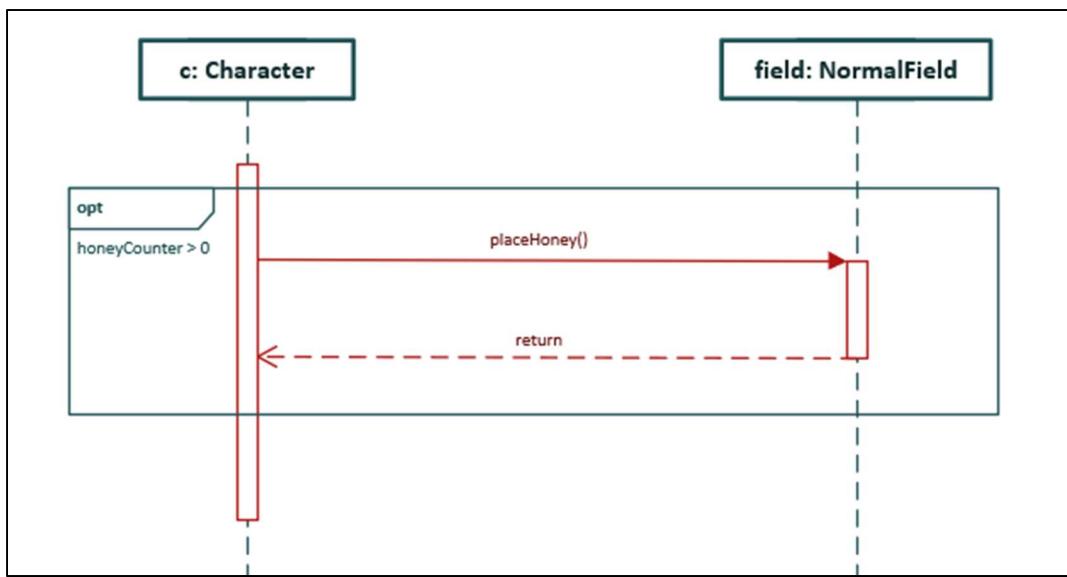
Lekerül egy NormalField-ről egy Thing, kitörli a Thing aktuális pozíóját, továbbá levonja erről a NormalField mezőről a Thing súlyát.

7.0.3. Szekvencia diagramok

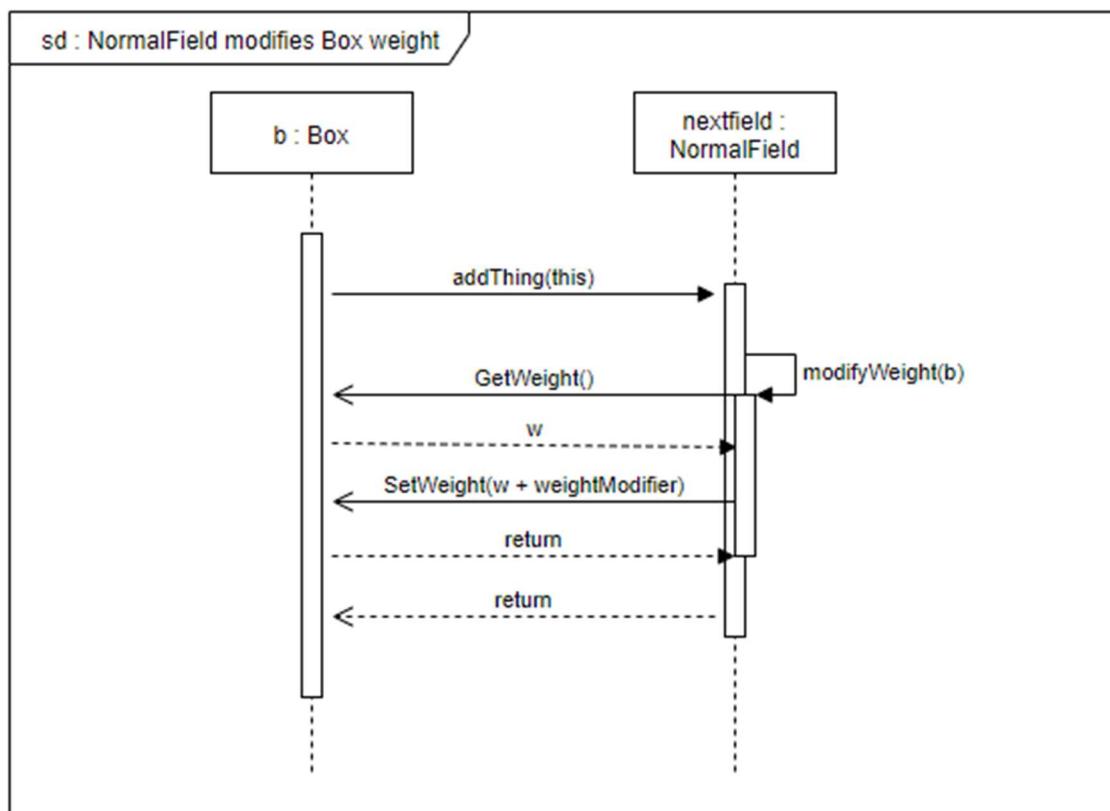
7.0.3.1. placeOil



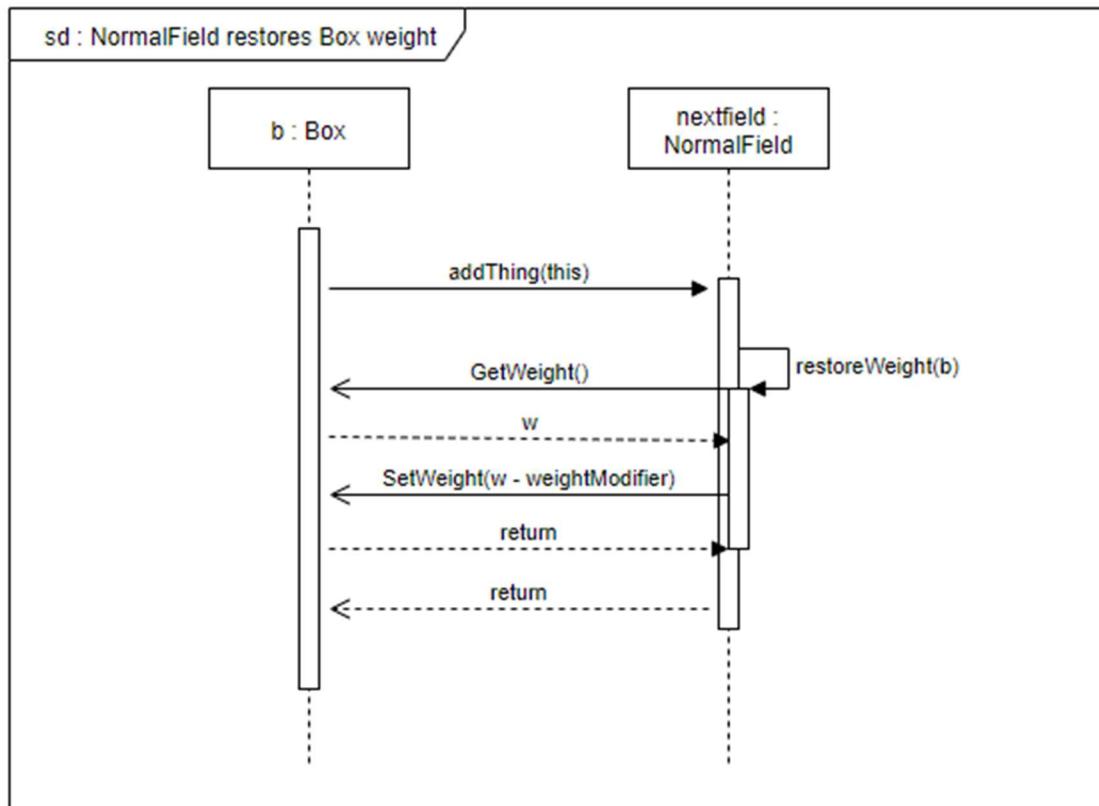
7.0.3.2. placeOil



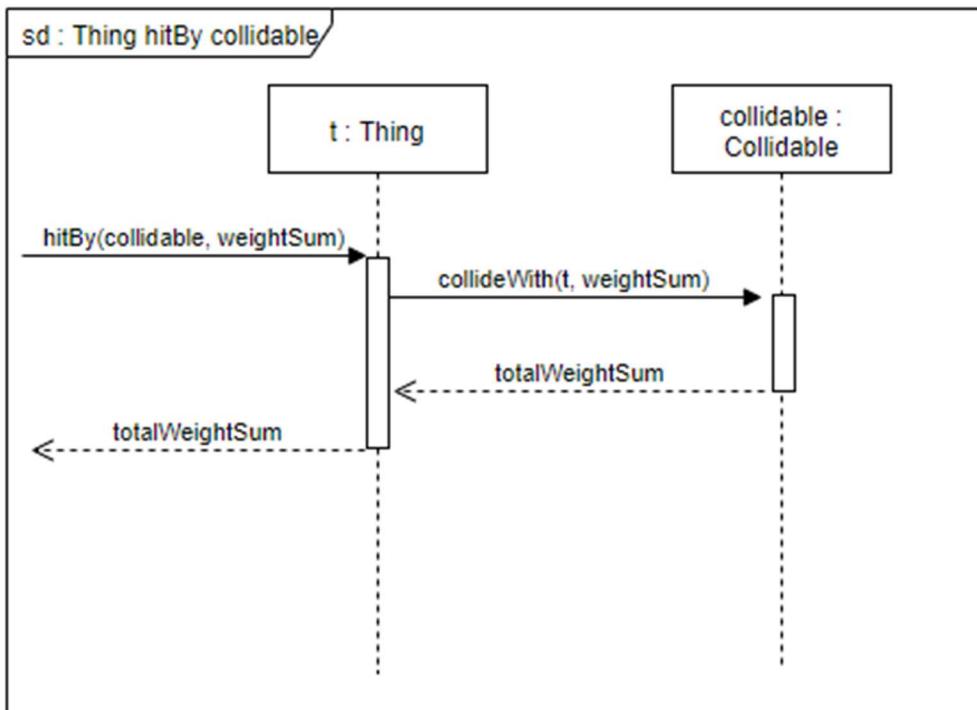
7.0.3.3. NormalField modifies Box weight



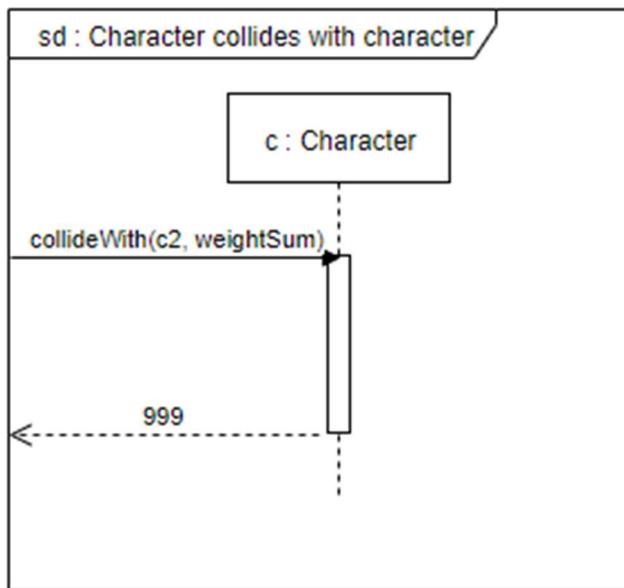
7.0.3.4. NormalField restores box weight



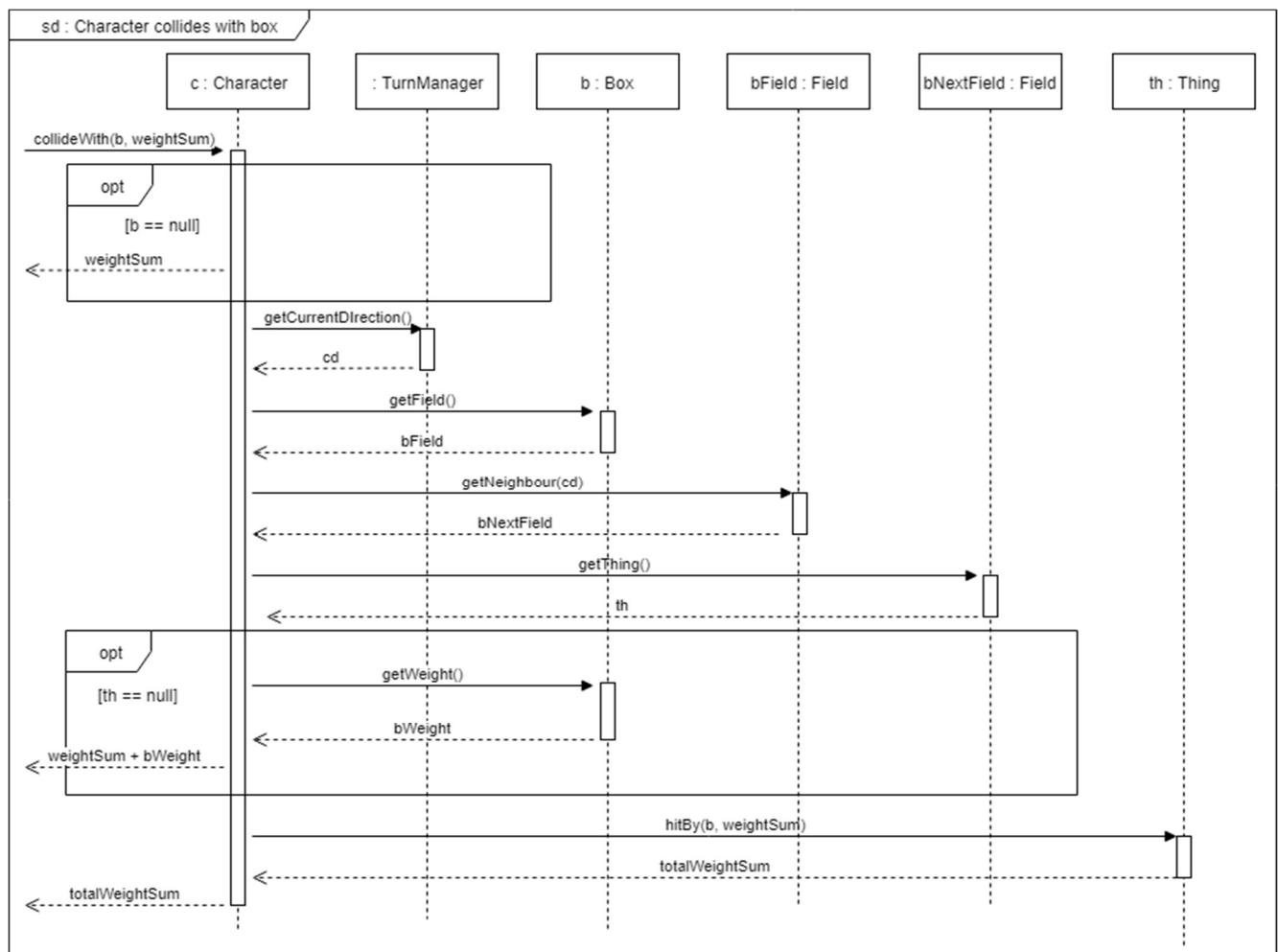
7.0.3.5. Thing hitBy collidable



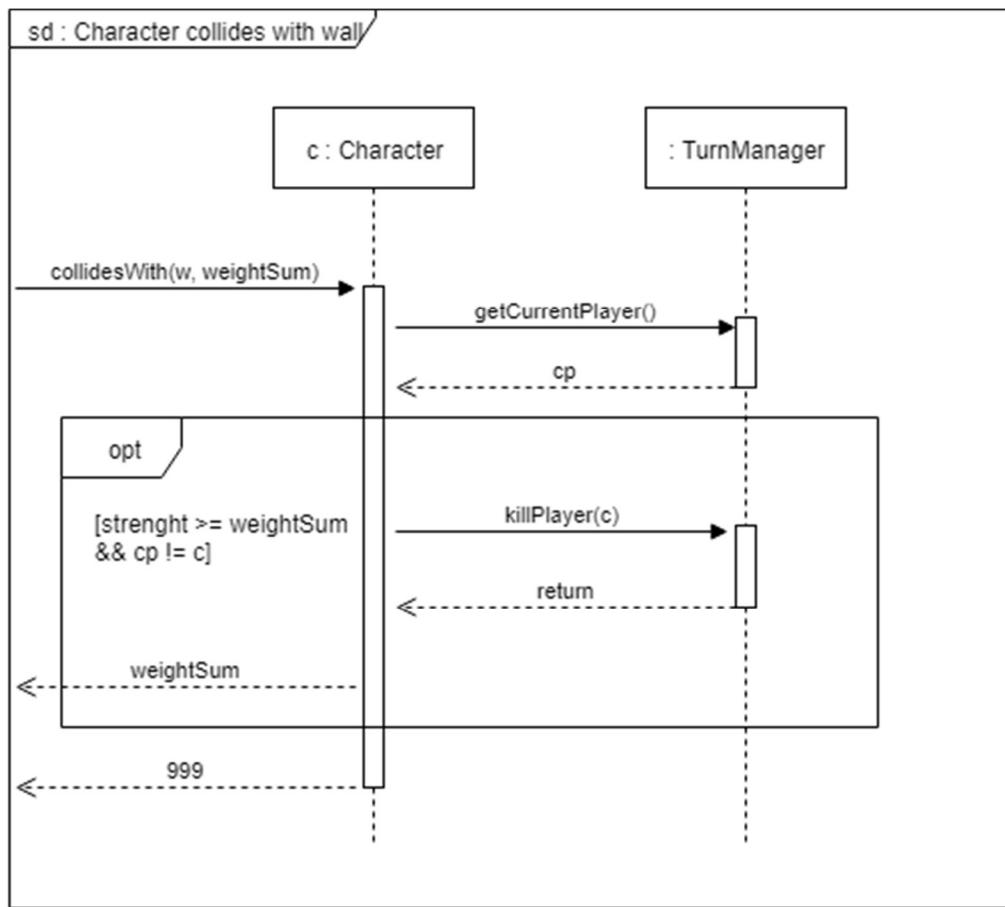
7.0.3.6. Character collides with character



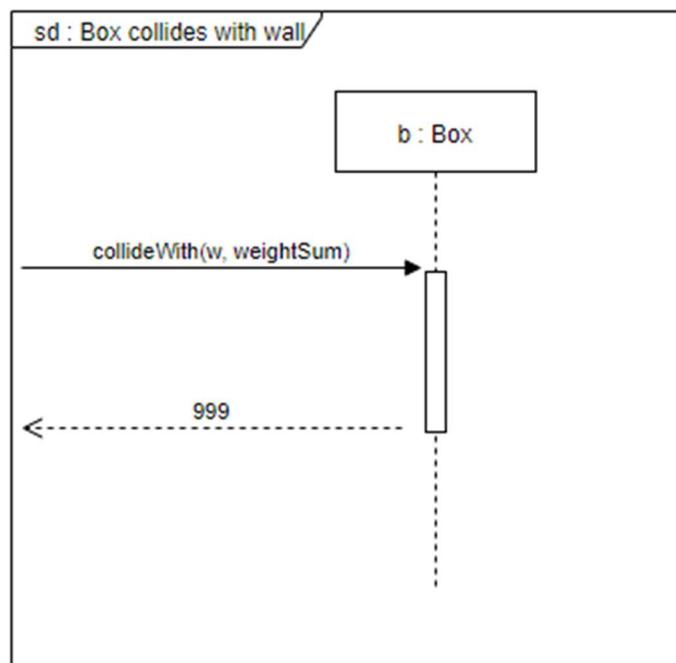
7.0.3.7. Character collides with box



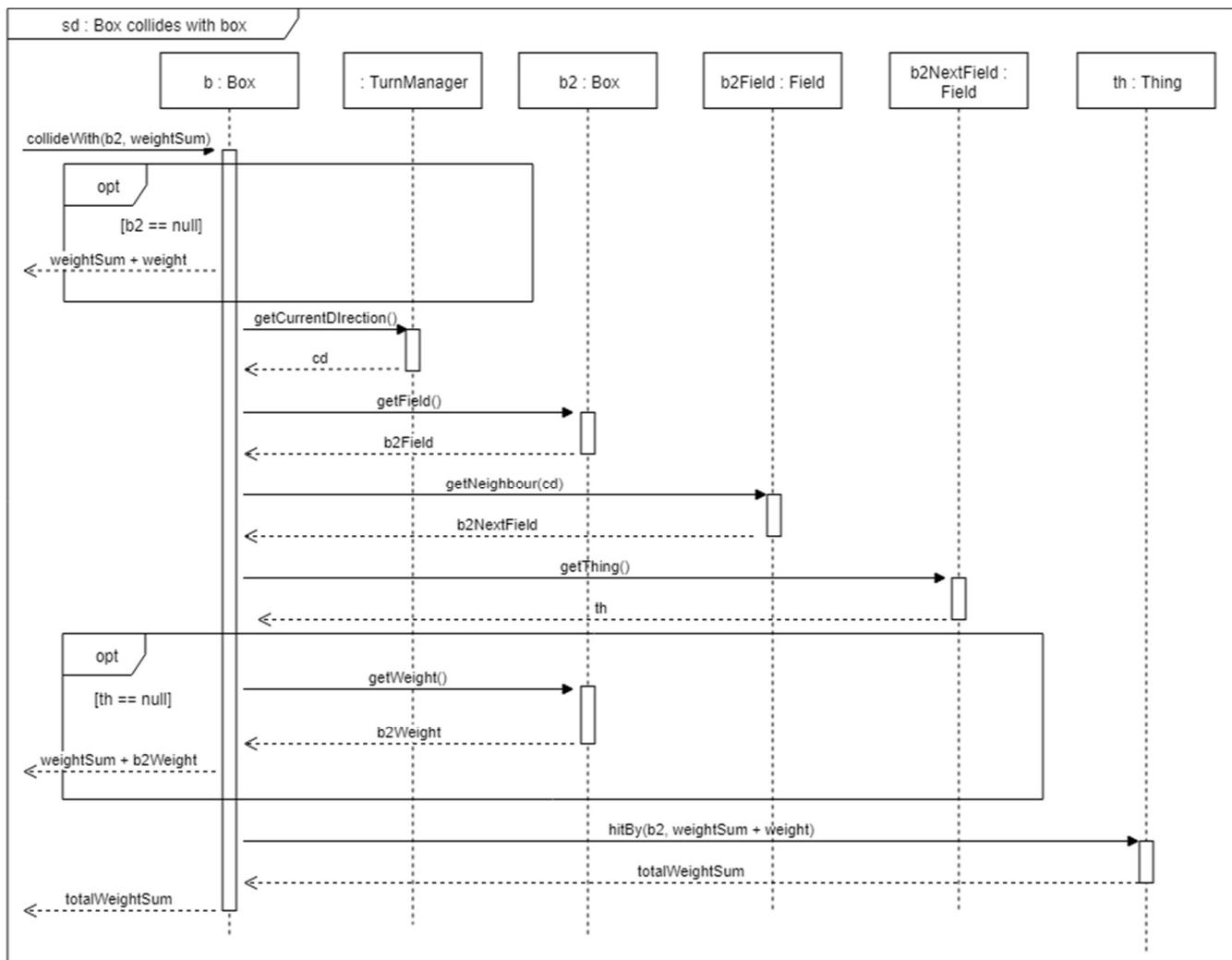
7.0.3.8. Character collides with wall



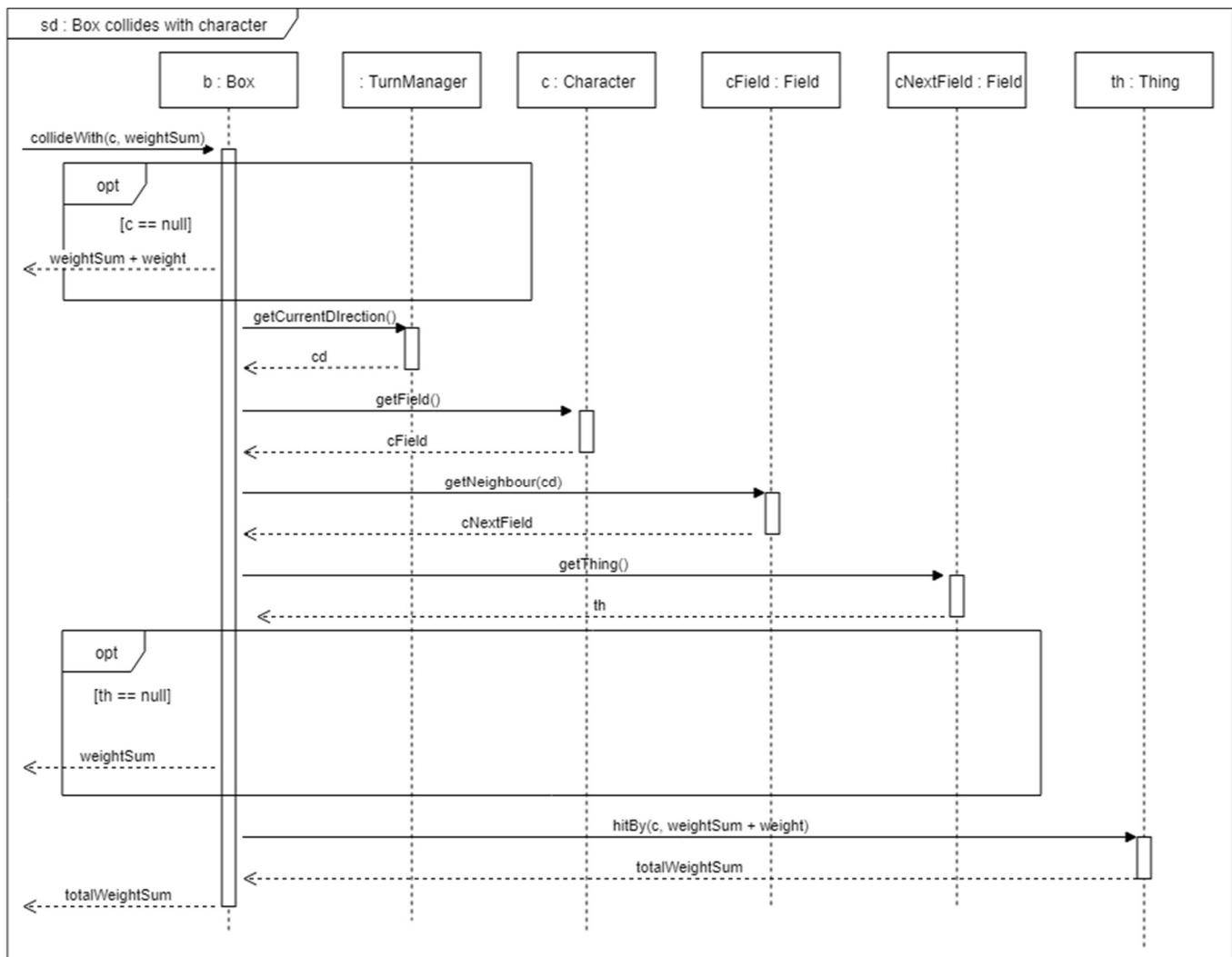
7.0.3.9. Box collides with wall



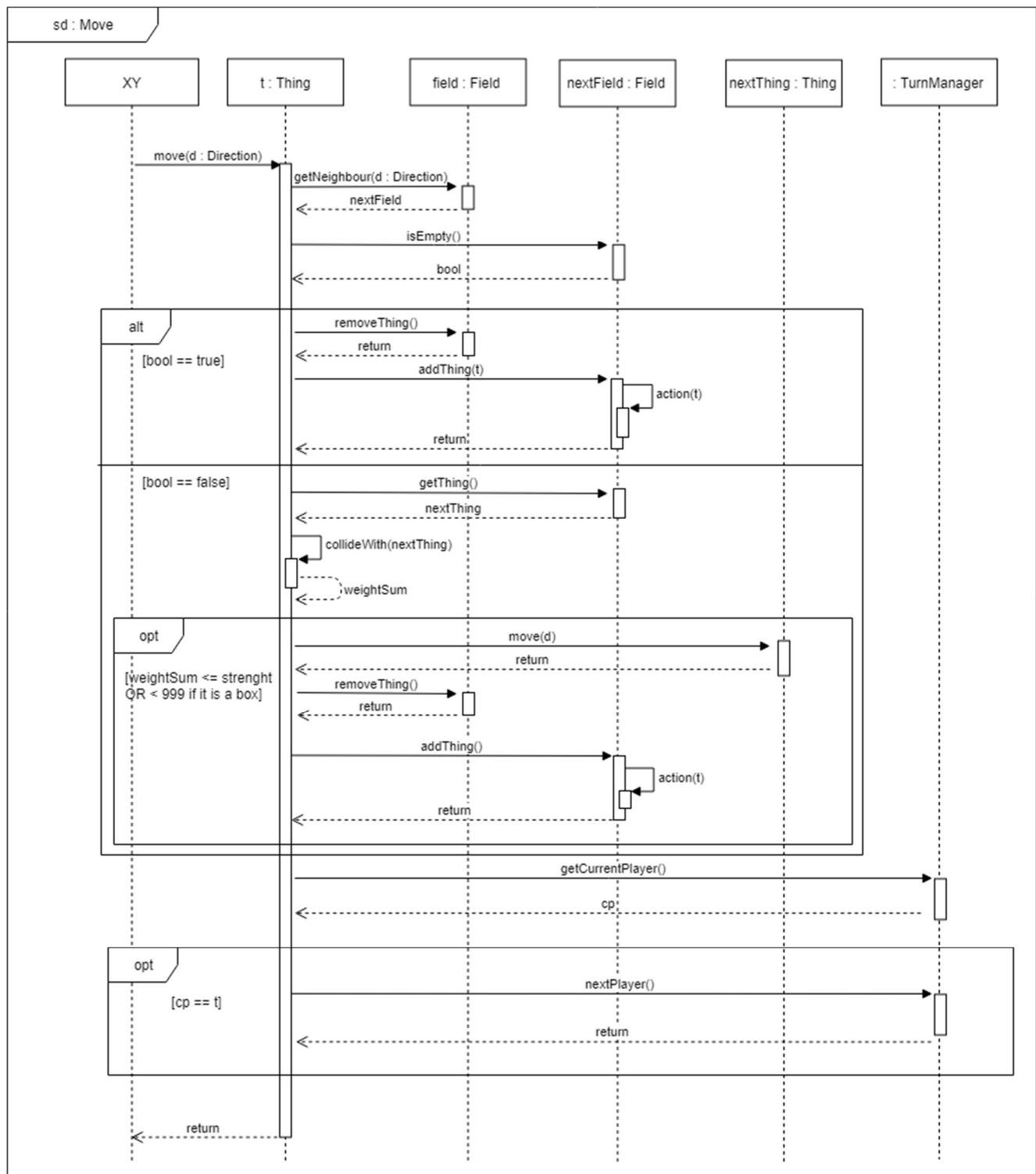
7.0.3.10. Box collides with box



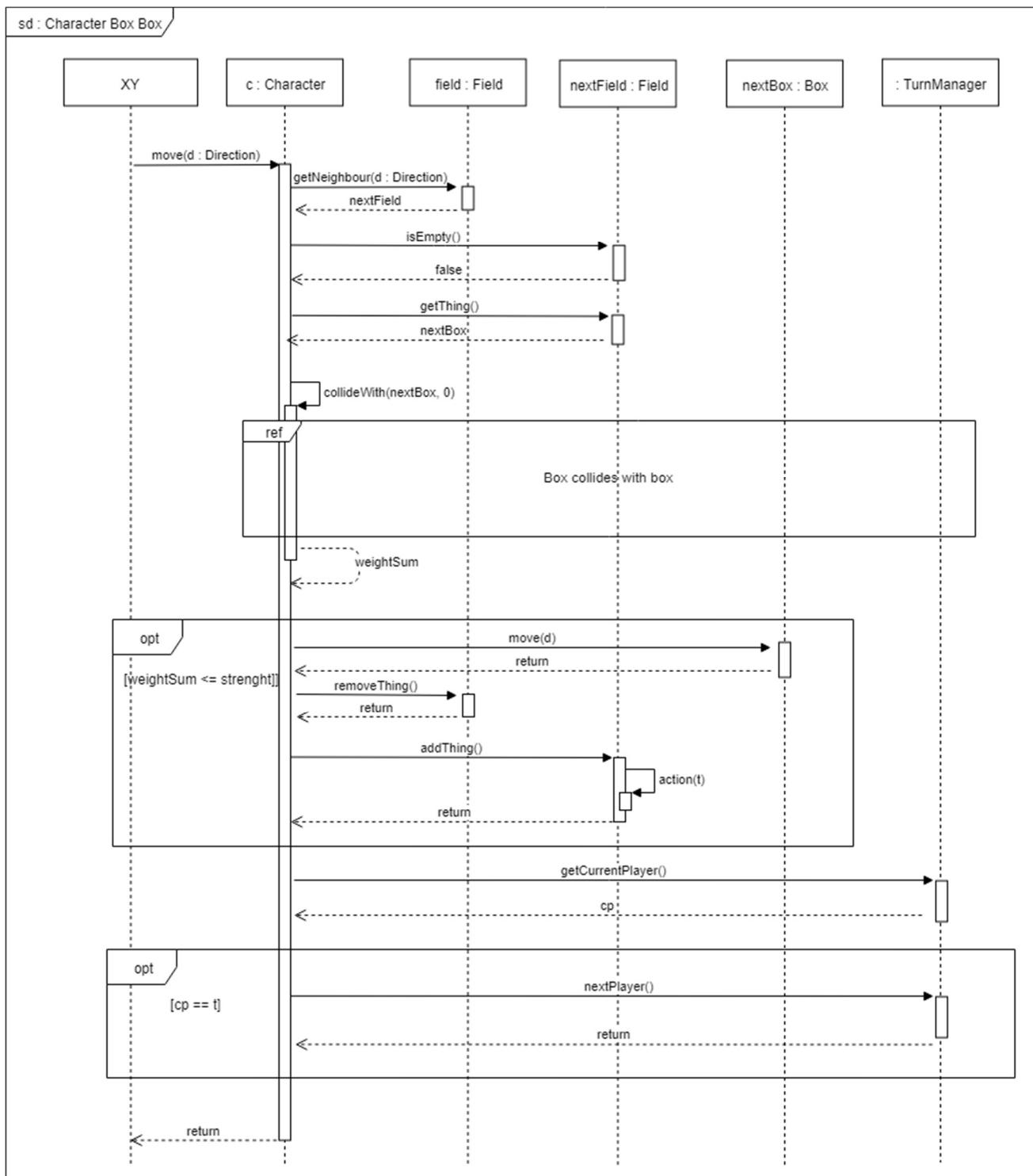
7.0.3.11. Box collides with character



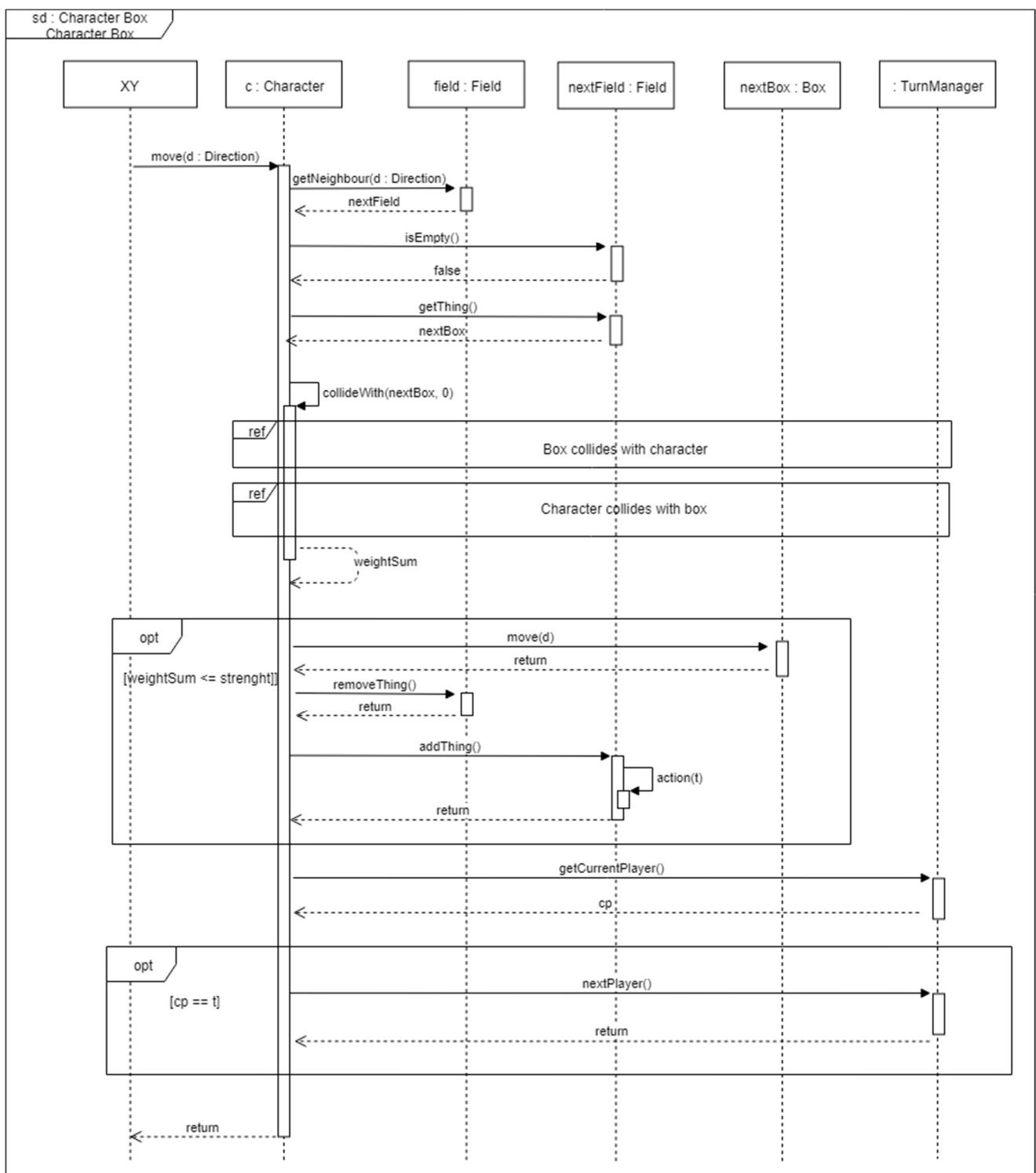
7.0.3.12. Move



7.0.3.13. Character box box



7.0.3.14. Character box character box



7.1. Prototípus interface-definíciója

7.1.1. Az interfész általános leírása

Az interfész a szabványos bemenetről fogad parancsokat, és a szabványos kimenetre írja ki az esetleges kimenetet. Így van mód file-ból olvasni, továbbá file-ba írni parancssorból a szabvány ki és bemenet átirányításával az automatikus teszteléshez előre elkészített teszteseteket. A tesztesetek a programnak parancsokat adhatnak, és a kapott kimenetet összehasonlíják az elvárt kimenettel. A tesztesetek célja a program helyes működéséről való megbizonyosodás.

7.1.2. Bemeneti nyelv

loadGame

Leírás: Egy fájlból betölti a függvény a pályát.

Opciók: -

step

Leírás: A játékos lép a karakterrel.

Opciók: A lépés iránya

placeOil

Leírás: A karakter olajat rak le a mezőre.

Opciók: -

placeHoney

Leírás: A játékos mézet rak le a mezőre.

Opciók: -

listMap

Leírás: 2D-s rajzot készít karakterekből, amin különböző karakterek jelzik a különböző dolgokat a pályán (doboz, játékos, fal stb.)

Opciók: -

listPlayers

Leírás: kiírja a játékosok adatait (az Ő köre jön-e, mennyi ideje van hátra, hány pontja van, él-e még)

Opciók: 1-4-ig a játékos sorszámai.

listCurrentPlayer

Leírás: a soron következő játékosról lehet adatokat lekérni.

Opciók: -

7.1.3. Kimeneti nyelv

Kimentet a következő parancsok adnak:

listMap

2d-s ASCII karakterekből álló rajzot készít karakterekből, amin különböző karakterek jelzik a különböző dolgokat a pályán (doboz, játékos, fal stb.)

Karakter - C, Doboz - B, Csapda - T, Kapcsoló - S, Lyuk - H, Fal - X, ÜresMező - <semmi>

listPlayers

Leírja a játékosok adatait az alábbi formában:

[játékos száma] - [él-e] - [aktív-e] - [hátralevő ideje] - [pontszáma]

listCurrentPlayer

Leírja az éppen aktív játékos adatait az alábbi formában:

[játékos száma] - [él-e] - [hátralevő ideje] - [pontszáma]

7.2. Összes részletes use-case

Use-case neve	loadGame
Rövid leírás	pálya betöltése, játék elindítása.
Aktorok	Player
Forgatókönyv	A játékos elindítja a programot, és betölti a pályát.

Use-case neve	step
Rövid leírás	A játékos léptetése
Aktorok	Player
Forgatókönyvek	<ul style="list-style-type: none"> 1. A játékos egy másik mezőre lép. 2. A játékos egy dobozt tol el. 3. A játékos falnak próbál menni. 4. A játékos egy másik játékosnak próbál menni.

Use-case neve	placeOil
Rövid leírás	Olaj lerakása egy mezőre.
Aktorok	Player
Forgatókönyv	A játékos olajat rak le azon mezőre, amelyen áll.

Use-case neve	placeHoney
Rövid leírás	Méz lerakása egy mezőre.
Aktorok	Player
Forgatókönyv	A játékos mézet rak le azon mezőre, amelyen áll.

Use-case neve	listMap
Rövid leírás	2D-s rajzot készít karakterekből, ami a pályán elhelyezett mutatja
Aktorok	Player
Forgatókönyv	Kirajzolja a kimenetre karakterekből a jelen pályát (2D-ben), bizonyos karakterek szimbolizálják a pályán lévő dolgokat.

Use-case neve	listPlayer
Rövid leírás	Kilistázza egy játékos adatait.
Aktorok	Player
Forgatókönyv	Kiírja a kimenetre valamely játékos adatait részletes formában.

Use-case neve	listCurrentPlayer
Rövid leírás	Kilistázza a soron lévő játékos adatait.
Aktorok	Player
Forgatókönyv	Kiírja a kimentre a soron lévő játékos részletes adatait részletes formában.

7.3. Tesztelési terv

Teszt-eset neve	Játékos mozgatása üres mezőre
Rövid leírás	teszteljük a játékos üres mezőre lépését.
Teszt célja	Teszteljük, hogy helyesen működik-e a program, ha a játékos üres mezőre lép.

Teszt-eset neve	Játékos mozgatása falnak
Rövid leírás	a játékos megpróbáljuk
Teszt célja	Teszteljük, hogy helyesen működik-e a program, vagyis hogy a játékos nem tud falat tartalmazó mezőre lépni.

Teszt-eset neve	Játékos-Doboz-Doboz tolás
Rövid leírás	A Játékos megpróbál eltolni egy dobozt, ami egy másik dobozt tol.
Teszt célja	Teszteljük, hogy el lehet-e tolni több dobozt egyszerre,

Teszt-eset neve	Méz lerakása
Rövid leírás	A játékos mézet önt a rajta álló mezőre.
Teszt célja	Teszteljük, hogy méz lerakásakor csökken-e a játékos mézkészlete és jobban tapad-e a mező amin ezt tette.

Teszt-eset neve	Olaj lerakása
Rövid leírás	A játékos olajat önt a rajta álló mezőre
Teszt célja	Teszteljük, hogy olaj lerakásakor csökken-e a játékos olajkészlete és jobban csúszik-e a mező amin ezt tette.

Teszt-eset neve	Játékos-Doboz-Játékos-Doboz interakció
Rövid leírás	A játékos egy dobozon keresztül eltol egy másik játékest, ami egy doboznak ütközik.
Teszt célja	Teszteljük, hogy ha dobozt nekitolunk egy játékosnak, akkor megfelelően viselkedik, vagyis hogy eltolja-e a második játékos a mögötte levő dobozt.

Teszt-eset neve	Játékos lyukra lép
Rövid leírás	A játékos rálép a lyuk mezőre.
Teszt célja	Teszteljük, hogy a játékos meghal-e, amikor lyuk mezőre lép.

Teszt-eset neve	Doboz mozgatása üres mezőre
Rövid leírás	A játékos eltol egy dobozt egy üres mezőre
Teszt célja	Teszteljük, hogy a játékos akadálymentesen el tud-e tolni egy dobozt egy üres mezőre.

Teszt-eset neve	Játékos-Játékos találkozás.
Rövid leírás	Játékos másik játékos mezőjére akar lépni
Teszt célja	Teszteljük, hogy helyesen működik-e a program ha egy játékos egy másik játékos mezőjére akar lépni.

Teszt-eset neve	Játékos nyitott csapdára lép.
Rövid leírás	A Játékos nyitott csapdára lép.
Teszt célja	teszteljük, hogy helyesen működik-e a program, vagyis hogy meghal-e a játékos.

Teszt-eset neve	Játékos látát tol egy kapcsolóra.
Rövid leírás	Láda kerül a kapcsolóra.
Teszt célja	Teszteljük, hogy a kapcsoló élesíti-e a csapdáját, ha rá kerül egy doboz.

7.4. Tesztelést támogató segéd- és fordítóprogramok specifikálása

A teszteket egy saját parancssoros segédprogrammal fogjuk futtatni, ami a standard bemenetről fogad szöveget, végrehajtja a teszt lépéseit, majd az outputot összehasonlítja az elvárt kimenettel. Ha hibás a sőkban (a játék maga) program, akkor jelez a segédprogram. Így nem kell repetitív módon nekünk végigjátszani a újra a teszteket, ahányszor módosítunk valamit a programon.

8. Részletes tervezettség

8.1. Osztályok és metódusok tervezettsége

8.1.1. Box

- Felelősség

A pályán található dobozokért felelős osztály. A játékosok tudják őket egy karakter vagy egy másik doboz nekitolásával mozgatni a Direction által meghatározott irányokba, amennyiben a dobozok pillanatnyi súlyozása és a toló karakter ereje ezt lehetővé teszi. Ezeket kell a játékosnak megfelelő helyre tolnia, ahol eltűnnek és pontot kap az adott doboz a helyére toló játékos. Kapcsoló mezőre érkezve, invertálja annak állapotát (ki, illetve bekapcsolt lehet), aktív csapdára vagy lyukra érkezve megsemmisül és eltűnik a pályáról. Más doboz nem, viszont más karaktert össze tud nyomni (megsemmisíteni), falnak ütközve megáll.

- Ősosztályok

Thing

- Interfészek

Collidable

- Attribútumok

-Field **field**: Annak a mezőnek a referenciajára, amin az adott doboz éppen tartózkodik.
-int **weight**: A doboz pillanatnyi súlya.

- Metódusok

+int **collideWith(Character c, int weightSum)**: Doboz karakterrel ütközik, a súlyozások alapján kiszámolja, hogy tudnak-e tovább mozogni.
+int **collideWith(Box b, int weightSum)**: Doboz dobozzal ütközik, a súlyozások alapján kiszámolja, hogy tudnak-e tovább mozogni.
+int **collideWith(Wall w, int weightSum)**: Doboz fallal ütközik, nem történik semmi, mert a fal mozdíthatatlan, a doboz pedig nem törhet össze.
+int **hitBy(Collidable collidable, int weightSum)**: Akkor hívódik meg, ha egy doboznak nekiütközik egy másik dolog. Visszatér a collideWith függvény által kalkulált mozgatási értékkel.
+void **move(Direction d)**: A doboz kísérletet tesz a mozgásra a paraméterként megadott d irányba.
+void **setField(Field f)**: A doboz objektum megkapja referenciajáért azon pályamezőt, amin éppen tartózkodik.
+Field **getField()**: Lekérdezi azt a mezőt, amin a doboz éppen tartózkodik.
+int **getWeight()**: Lekérdezi az adott doboz pillanatnyi súlyát.
+void **setWeight(Weight w)**: Beállítja a doboz súlyát.

8.1.2. Character

- **Felelősség**

A játékosokért felelős osztály, a pályán található játékosokhoz tartozó objektum. A karakterek azok a játékelemek, amiket a játékos közvetlenül tud irányítani. minden játékosnak minden körben pontosan egy darab gomblenyomásra van lehetősége. Ha a játékos mozdíthatatlan pályaelem, mint fal, oszlop, ezek mellett levő doboz felé mozgatná a karaktert, a rendszer azt is lépésnek veszi, tehát az adott körben a lépését elhasználta, ennek ellenére nem mozgott a pályán. Doboznak való mozgás által, amennyiben azok pillanatnyi súlyozása és a karakter ereje azt lehetővé teszi, mozgatni tudja azokat. Ha karakter célra tol egy dobozt, akkor pontot kap. Karakter aktív csapdára vagy lyukra érkezve meghal és eltűnik a pályáról.

- **Ősosztályok**

Thing

- **Interfészek**

Collidable

- **Attribútumok**

- Field **field**: Annak a mezőnek a referenciája, amin az adott karakter éppen tartózkodik.
- int **point**: Adott karakter pillanatnyi pontjainak a száma.
- int **timer**: Adott karakter hátralevő játékidejének a mennyisége másodpercben.
- bool **active**: Aktív-e az adott karakter.
- int **strength**: A karakterre jellemző erő. Ha kisebb, mint a dolgok sorozata, amit mozdítani próbál, akkor a mozgatás sikertelen lesz.
- int **honeyCounter**: Adott karakter letehető mézeinek a száma darabban.
- int **oilCounter**: Adott karakter letehető olajainak a száma darabban.

- **Metódusok**

- +int **collideWith(Character c, int weightSum)**: Karakter másik karakterrel ütközik. Karakter közvetlenül nem tud másik karaktert elmozdítani, ezért ilyenkor nem történik semmi.
- +int **collideWith(Box b, int weightSum)**: Karakter dobozzal ütközik, a súlyozások alapján kiszámolja, hogy tudnak-e tovább mozogni.
- +int **collideWith(Wall w, int weightSum)**: Karakter fallal ütközik. Ha a karakter éppen aktív, akkor nem történik semmi, mert a játékos önszántából ment a falnak, viszont ha inaktív, tehát más dolog tolta a falnak, akkor meghal.
- +int **hitBy(Collidable collidable, int weightSum)**: Akkor hívódik meg, ha egy karakterek nekiütközik egy másik dolgot. Visszatér a collideWith függvény által kikalkulált mozgatási értékkel.
- +void **move(Direction d)**: A karakter kísérletet tesz a mozgásra a paraméterként megadott d irányba.
- +void **addPoint()**: Karakter pontot szerzett doboz a cérla tolásával.
- +int **getPoint()**: Lekérdezi az aktuális karakter aktuális pontjainak a számát.
- +bool **isActive()**: Aktív-e az adott karakter.
- +bool **isDead()**: Él-e az adott karakter.
- +void **reduceTimer()**: Csökkenti az aktív karakter hátralévő idejét 1 másodperccel.

- +void setTimer(int timeRemaining):** A játék elején inicializálja a TurnManager az összes karakter kezdeti idejét ugyanazzal az értékkel.
- +void setField(Field f):** A karakter objektum megkapja referenciaként azon pályamezőt, amin éppen tartózkodik.
- +Field getField():** Lekérdezi azt a mezőt, amin a karakter éppen tartózkodik.

8.1.3. Collidable

- **Felelősség**

Az ütközésre képes pályaelemek valósítják meg ezt az interfészét, majd definiálják felül működéseik szerint a collideWith metódusait.

- **Metódusok**

- +int collideWith(Character c, int weightSum):** Üres, felüldefiniálandó függvényfejléc, akkor hívódik meg, ha Collidable interfész megvalósító osztály objektuma karakterrel ütközik.
- +int collideWith(Box b, int weightSum):** Üres, felüldefiniálandó függvényfejléc, akkor hívódik meg, ha Collidable interfész megvalósító osztály objektuma dobozzal ütközik.
- +int collideWith(Wall w, int weightSum):** Üres, felüldefiniálandó függvényfejléc, akkor hívódik meg, ha Collidable interfész megvalósító osztály objektuma fallal ütközik.

8.1.4. Direction

- **Felelősség**

A pályára vonatkozó közlekedési irányok enumerációját testesíti meg.

- **Attribútumok**

- +Up, Down, Left, Right:** A négy irány, amerre a játékban mozogni lehet.

8.1.5. Field

- **Felelősség**

A Map osztályban található mezőknek az absztrakt alaposztálya. Összefoglaló osztály, ezek tömbjét tárolja a Map osztály egy adattagjában.

- **Attribútumok**

- #Thing thing:** Annak a dolognak a referenciaja, ami adott időpillanatban az adott mezőn található.
- Map<Direction, Field> neighbours:** A mező 4 égtáj szerinti szomszédjai a Direction enumeráció elemeihez társítva.

- **Metódusok**

- +void addThing(Thing t):** Egy dolog rákerül a mezőre, eltárolja annak a referenciaját.

+void removeThing(): Akkor hívódik meg, ha egy Character ellép, vagy egy Box eltolódik azaz valamilyen doleg lekerül az adott mezőről.

+Field getNeighbour(Direction d): A paraméterként kapott irányba található szomszédját adja vissza.

+void setNeighbour(Direction d, Field f): A paraméterként kapott irányba található szomszédját állítja be.

+bool isEmpty(): Lekérdezi, hogy éppen üres-e az adott mező.

+void action(Character c): Üres, felüldefiniálandó függvényfejléc. Akkor hívódik meg, amikor egy karakter kerül a mezőre.

+void action(Box b): Üres, felüldefiniálandó függvényfejléc. Akkor hívódik meg, amikor egy doboz kerül a mezőre.

+void switchState(): Üres, felüldefiniálandó függvényfejléc. Adott csapda mezőnek váltja az állapotát aktív és inaktív között.

+void placeOil(): Üres, felüldefiniálandó függvényfejléc. Adott mezőre olajat helyez le.

+void placeHoney(): Üres, felüldefiniálandó függvényfejléc. Adott mezőre mézet helyez le.

8.1.6. Game

- **Felelősség**

A pálya nyilvántartásáért és a játékmenet legfőbb funkcióiért (elindítása vagy lezárása) felelős. Meghívja további osztályok inicializáló függvényeit.

- **Attribútumok**

-Map map: Számoltartja a játékhoz tartozó pályát.

- **Metódusok**

+void startGame(): A teljes játék elindításáért felelős. Bekéri a felhasználótól a játékosok számát, majd meghívja a Map osztály initMap függvényét.

+void endGame(): A játék lezárásáért felelős. Akkor fut le, amikor minden játékosnak elfogy a saját ideje. Ilyenkor összegzi az eredményeket, kiírja a győztest, vagy győzteseket és visszalép a játék főmenüjébe.

8.1.7. Goal

- **Felelősség**

A pályának azon, megkülönböztetett színnel és mintával jelzett mezői, ahova a játékosoknak dobozokat kell tologatniuk, hogy pontokat kapjanak. A játék célja, hogy adott játékos minél több célmezőre, minél több dobozt toljon.

- **Ősosztályok**

Field

- **Metódusok**

+void action(Character c): Ha karakter kerül a célmezőre, akkor nem történik semmi, mivel csak a dobozárt jár pont, a pontosztás kivételével pedig a cél csak egy sima mező.

+void action(Box b): Ha doboz kerül a célmezőre, akkor a doboz lekerül a pályáról, és a dobozt a célba toló játékosnak pontot ír jóvá a turnManager.

8.1.8. Hole

- **Felelősség**

Olyan Field mező, amire ha rákerül Character vagy Box objektum, akkor azt megsemmisül/meghal. Abban különbözik a Trap osztály példányaitól, hogy míg azok a Switch-el való aktiválásig NormalField-ként viselkednek (nem jelentenek veszélyt a rájuk tollt Box vagy Character-re), addig a Hole az konstans lyukként viselkedik, bármelyik időpillanatban megsemmisít, bármilyen rá kerülő objektumot.

- **Ősosztályok**

Field

- **Metódusok**

+void action(Character c): Ha egy karakter kerül a lyukra, akkor az meghal, és eltűnik a pályáról, adott játékos számára véget ér a játék.

+void action(Box b): Ha egy doboz kerül a lyukra, akkor az megsemmisül, és eltűnik a pályáról.

8.1.9. Map

- **Felelősség**

A pálya, amin a játék zajlik. Mezők nxn méretű mátrixából áll, amin megjelenhetnek falak, dobozok, csapdák, kapcsolók, lyukak, karakterek, és üres mezők. A pályát ő maga generálja le megadott szabályoknak és kikötéseknek megfelelően.

- **Attribútumok**

-Field[][] fields: A pálya mátrixát tartalmazó tömb. A pálya kirajzolását segíti, a játékmenet logikája a getNeighbour() függvényen alapul.

- **Metódusok**

+void initMap(NumberOfPlayers int): A pályát legeneráló és létrehozó függvény. Ezen felül paraméterül kapja a Game singleton-tól az elindítani kívánt játékban résztvevő játékosok számát, amiket aztán tovább is ad a TurnManager osztálynak, ami azt az init() függvényében használ fel.

8.1.10. NormalField

- **Felelősség**

A pályán található sima üres mezőkért felelős osztály. A játékosok és a dobozok ezeken tudnak szabadon közlekedni, az erre való rálépéskor nem történik semmilyen speciális esemény, viszont erre a mezőre tehet le a játékos karaktere mézet vagy olajat.

- **Ősosztályok**

Field

- **Attribútumok**

-int weightModifier: Adott mező súlymodifikációs változója. A karakterek által lehelyezhető méz, illetve olaj módosíthatja ezt rendre pozitív, illetve negatív irányba.

- **Metódusok**

+void addThing(Thing t): Egy doleg rákerül a mezőre, eltárolja annak a referenciáját, és amennyiben található ezen mezőn méz vagy olaj, akkor azok hatásainak eredőjével változtatja a mezőre került doboz súlyát.

+void removeThing(): Akkor hívódik meg, ha egy Character ellép, vagy egy Box eltolódik azaz valamilyen doleg lekerül az adott mezőről. A referencia törlése előtt visszaállítja a súlyukat a módosítás előtti állapotra.

+void placeOil(): Negatív irányba módosítja a mező súlymodifikációs változójának az értékét.

+void placeHoney(): Pozitív irányba módosítja a mező súlymodifikációs változójának az értékét.

+void modifyWeight(Box b): Változtatja a mezőre került doboz súlyát a súlymodifikációs változó pillanatnyi értékével.

+void restoreWeight(Box b): Visszaállítja a mezőn lévő, azaz a mezőről éppen lekerülő doboz súlyát a rákerülés előtti állapotra.

+void modifyWeight(Character c): A karakternek nincsen súlya, ezért nem adunk hozzá semmit.

+void restoreWeight(Character c): A karakternek nincsen súlya, nem adtunk hozzá semmit, ezért nincs is mit visszaállítani.

8.1.11. Switch

- **Felelősség**

A pályán megtalálható kapcsoló mező, amelyhez pontosan egy csapda tartozik. Akkor aktiválódik, ha egy dobozt tolnak rá, majd deaktiválódik a következő rátolásnál. Játékosként rálépe nem történik semmi.

- **Ősosztályok**

Field

- **Attribútumok**

-Trap trap: A kapcsolóhoz tartozó csapda referenciajára.

- **Metódusok**

+void addThing(Thing t): Egy doleg rákerül a mezőre, eltárolja annak a referenciáját, és meghívja rá az action függvényt.

+void removeThing(): Akkor hívódik meg ha karakter ellép, vagy doboz lekerül a mezőről. Ilyenkor is meghívja az action függvényt.

+void action(Character c): Ha karakter kerül a kapcsolóra, nem történik semmi, mivel irányába NormalField-ként viselkedik.

+void action(Box b): Ha doboz kerül a kapcsolóra, akkor meghívja a kapcsolóhoz társított csapda switchState() függvényét.

8.1.12. Thing

- **Felelősség**

Gyűjtőfogalom, absztrakt ōsosztály a mezőkön megtalálható “dolgok”-ra. Ilyen lehet például fal (Wall), doboz (Box) és játékos (Character). Esetleges ütközések esetén meghatározza, hogy milyen cselekmény fog végbe menni.

- **Interfészek**

Collidable

- **Attribútumok**

#Field field: A mező, amelyen adott időpillanatban egy adott dolog található.

- **Metódusok**

+Field getField(): Lekérdezi azt a mezőt, amin a dolog éppen tartózkodik.

+void setField(Field f): A dolog megkapja és beállítja magának referenciajént azon pályamezőt, amin éppen tartózkodik.

+void move(Direction d): Üres, felüldefiniálandó függvényfejléc. Egy adott dolog mozgásáért felel.

8.1.13. Timer

- **Felelősség**

A számítógép belső óráját kérdezi le, és értesíti a TurnManager osztályt arról, hogy eltelt egy másodperc. minden játékosnak külön számlálója és hátralevő ideje van. Ha a játékos ideje elfogy, akkor többet már nem léphet, viszont a játékot ettől függetlenül még megnyerheti.

- **Metódusok**

void run(): Végtelen ciklusban fut a háttérben, és másodpercenként meghívja a tick függvényt, ami pedig a turnManager time() függvénye segítségével dekrementálja az aktuális játékos hátralevő idejét.

8.1.14. Trap

- **Felelősség**

A pályán megtalálható csapda játékelemekért felelős osztály. Ha egy játékos vagy doboz belekerül ha aktivált állapotban van, az megsemmisül/meghal. minden kapcsolóhoz (switch) pontosan egy csapda, és minden csapdához pontosan egy kapcsoló tartozik, az állítja a csapda állapotát ki- és bekapcsolt között.

- **Ósosztályok**

Field

- **Attribútumok**

-bool opened: Nyitott vagy csukott állapotban van-e éppen a csapda.

- **Metódusok**

- void switchState()**: Aktiválja vagy deaktiválja magát a csapda a linkelt Switch-től származó jel függvényében.
- void action(Box b)**: Ha egy doboz kerül a csapdára, amikor az nyitott állapotban van, akkor az a doboz megsemmisül és lekerül a pályáról. Csukott állapot esetében üres NormalField mezőként viselkedik, és egyszerűen rákerül.
- void action(Character c)**: Ha egy karakter kerül a csapdára, amikor az nyitott állapotban van, akkor az a karakter meghal és lekerül a pályáról. Csukott állapot esetében üres NormalField mezőként viselkedik, és egyszerűen rákerül a karakter.

8.1.15. TurnManager

- **Felelősség**

Megkapja a játékosok számát, majd menedzseli a játékmenetet. Tudja, és tárolja az aktuális játékost, az adott körben releváns mozgási irányt, továbbá tárolja az összes játékost tartalmazó listáját. Felel azért, hogy melyik játékosnak a köre következik, ki és ki az aktuális játékos, aki az adott körben léphet.

A Timer segítségével fix időközönként (másodperc) dekrementálja a saját maga által tárolt, és éppen aktív karakterek hátralevő idejét, annak a tick() függvénye segítségével.

- **Attribútumok**

- Game game:** A pálya nyilvántartásáért és a játékmenet legfőbb funkcióiért felelős game singleton referencia.
- Character currentPlayer:** Az éppen aktív játékost tároló referencia.
- Direction currentDirection:** Az adott kör mozgási irányát tároló referencia.
- List<Character> players:** Az összes karaktert tartalmazó lista.

- **Metódusok**

- +void init(int numberOfPlayers):** Létrehozza a paraméterül kapott számú játékosok karaktereit, hozzáadja őket az azokat tároló listához, az első játékos karakterét aktívvá teszi, majd elindítja a játek futásának magjaként tekinthető loop() függvényt.
- +void loop():** Ezt hívja meg az init() függvény, végételen ciklusba olvassa az inputot, szól az aktuális karaktereknek, hogy lépjön, majd lépteti magát a következő játékosra.
- +void time():** Amikor a Timer jelez, akkor levon az aktív játék hátralevő idejéből egy másodperct.
- +void goal():** Ha egy karakter egy célmezőre tol egy dobozt, akkor meghívódik ez a függvény és pontot ír jóvá neki.
- +void killPlayer(Character c):** Megölünk egy játékost. Ennek a működése úgy került megvalósításra, hogy a hátralevő idejét 0-ra állítjuk, ezáltal a TurnManager már nem fog számára több lépései lehetőséget biztosítani.
- +void nextPlayer():** Átadja a lépései jogot a soron következő még élő játékosnak.
- +Direction getCurrentDirection():** Lekérdezi az adott kör mozgási irányát. Egy körben egy irányba lehet csak maximum egy mezőnyit mozogni. A karakter által közvetetten mozgatott dolgok sem mozoghatnak más irányba.
- +Character getCurrentPlayer():** Lekérdezi, hogy ki éppen az aktuális karakter.
- +List<Character> getPlayers():** Visszaadja az összes játékost tartalmazó listát.
- +void setCurrentPlayer(Character currentPlayer):** Beállítja a soron következő karaktert aktívnak.

8.1.16. Wall

- **Felelősség**

A pályán megtalálható mozdíthatatlan, és áthatolhatatlan fal objektumokért felelős. Ha egy falszakasz nem több, egymással összefüggő komponensből áll (1×1 mező méretű), akkor oszlopnak nevezzük. Egy oszlop a méretétől eltekintve minden más tulajdonságában megegyezik a fallal. Ha magától felnak menne a játékos által irányított karakter, akkor beleütközik, nem történik semmi. Hasonlóan a dobozzal is, viszont amennyiben egy karakter, avagy egy doboz tol egy másik karaktert felnak, az meghal.

- **Ősosztályok**

Thing

- **Interfészek**

Collidable

- **Metódusok**

+int **hitBy(Collidable collidable)**: Felnak ütközik egy dolog.

Visszatér a collideWith

függvény általkikalkulált mozgatási értékkel.

+int **collideWith(Character c, int weightSum)**: A fal karakterrel ütközik. A fal mozdíthatatlan, ezért maximálissúllyal tér vissza, megakadályozva ezzel a mozgást.

+int **collideWith(Box b, int weightSum)**: A fal dobozzal ütközik. A fal mozdíthatatlan, ezért maximálissúllyal tér vissza, megakadályozva ezzel a mozgást.

+int **collideWith(Wall w, int weightSum)**: A fal fallal ütközik. A fal mozdíthatatlan, ezért maximálissúllyal tér vissza, megakadályozva ezzel a mozgást.

8.2. A tesztek részletes tervei, leírások a teszt nyelvén

8.2.1. Játékos mozgatása üres mezőre

- **Leírás**

Teszteljük a játékos üres mezőre lépését. A játékosnak át kell kerülnie az üres mezőre, majd az előzőről eltűnni.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Alapvető mozgást vizsgálja. A játékos nem mozdul a helyéről.

- **Bemenet**

createmap

add player 2 4 d

command currentplayer move right

finish

- **Elvárt kimenet**

currentplayer 3 4

8.2.2. Játékos mozgatása falnak

- **Leírás**

Teszteljük a játékos falnak mozgatását. A játékosnak meg kell akadnia, és a lépését abban a körben elveszteni ezzel.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A játékos falnak menését vizsgálja. Ha a játékos elmozdul a helyéről az hibás működést jelent.

- **Bemenet**

```
createmap  
add player 2 4  
add wall 3 4  
command currentplayer move right  
finish
```

- **Elvárt kimenet**

```
currentplayer 2 4
```

8.2.3. Játékos-Doboz-Doboz tolás

- **Leírás**

A játékos 2 dobozt próbál egyszerre tolni. Amennyiben utánuk üres mező van, a játékos eltolja a dobozokat. (A játékos elég erős, alap ereje 2-es)

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A játékos és több doboz interakcióját vizsgálja. Hiba, ha a doboz eltolódik abba az irányba, amely irányba mellette fal található, vagy ha nem mozdul úgy, hogy üres mező van mellette.

- **Bemenet**

```
createmap  
add player 2 4  
add box 3 4  
add box 4 4  
command currentplayer move right  
finish
```

- **Elvárt kimenet**

```
currentplayer 3 4  
box 4 4  
box 5 4
```

8.2.4. Méz lerakása

- **Leírás**

A játékos mézet próbál lerakni. Normál mezőn sikeres.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A játékos méz lerakási funkcióját vizsgálja. Hiba, ha a játékos sima mezőre nem tud lerakni mézet.

- **Bemenet**

```
createmap  
add player 1 2  
command currentplayer place honey  
add switch 1 3  
command currentplayer move down  
command currentplayer place honey  
finish
```

- **Elvárt kimenet**

```
currentplayer 1 2  
true  
switch 1 3  
currentplayer 1 3  
false
```

8.2.5. Olaj lerakása

- **Leírás**

A játékos olajat próbál lerakni. Normál mezőn sikeres.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A játékos olaj lerakási funkcióját vizsgálja. Hiba, ha a játékos sima mezőre nem tud lerakni olajat.

- **Bemenet**

```
createmap  
add player 1 2  
command currentplayer place oil  
add switch 1 3  
command currentplayer move down  
command currentplayer place oil  
finish
```

- **Elvárt kimenet**

```
currentplayer 1 2  
true  
currentplayer 1 3  
false
```

8.2.6. Játékos-Doboz-Játékos-Doboz interakció

- **Leírás**

A játékos dobozt, játékost, dobozt próbál egyszerre tolni. Amennyiben utánuk üres mező van, a játékos eltolja a dobozokat, illetve az inaktív játékost. (A játékos elég erős, alap ereje 2-es)

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A játékos doboz-játékos-doboz eltolási funkcióját vizsgálja. Hiba, ha a dobozt rá tudja tolni a játékosra, és a játékos meghal. Szintén hibás működés, ha fal van a konvoj tolási irányában, és bele tudja a játékos a dobozt a falba tolni.

- **Bemenet**

```
createmap
add player 2 4
add box 3 4
add player 4 4
add box 5 4
command currentplayer move right
finish
```

- **Elvárt kimenet**

```
currentplayer 3 4
box 4 4
player 5 4
box 6 4
```

8.2.7. Játékos lyukra lép

- **Leírás**

A játékos lyuk mezőre lép. Ekkor a játékosnak meg kell halni, ideje nullázódik.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A játékos és a lyuk kapcsolatát vizsgálja. Hiba, ha a játékos nem mozdul (a lyukra) vagy lyukra lép és nem hal meg.

- **Bemenet**

```
createmap
add player 2 4
add hole 3 4
command currentplayer move right
finish
```

- **Elvárt kimenet**

```
hole 3 4
currentplayer died
```

8.2.8. Doboz mozgatása üres mezőre

- **Leírás**

A játékos egy dobozt üres mezőre tol. A játékosnak a doboz helyére kell kerülnie, a doboznak pedig az üres mezőre.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A játékos és egy doboz alapvető tolási kapcsolatát vizsgálja. Hiba, ha a játékos nem kerül a doboz helyére, és/vagy a doboz nem tolódik el.

- **Bemenet**

createmap
 add player 2 4
 add box 2 5
 command currentplayer move down
 finish

- **Elvárt kimenet**

currentplayer 2 5
 box 2 6

8.2.9. Játékos-játékos találkozás

- **Leírás**

A játékos játékost próbál tolni. Ennek sikertelennek kell lennie.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Két játékos kapcsolatát vizsgálja. Hiba, ha a játékosok közül bármelyik is elmozdul a helyéről.

- **Bemenet**

createmap
 add player 2 4
 add player 3 4
 command currentplayer move right
 finish

- **Elvárt kimenet**

currentplayer 2 4
 player 3 4

8.2.10. Játékos nyitott csapdára lép

- **Leírás**

A játékos nyitott lyuk mezőre lép. Ekkor a játékosnak meg kell halni, ideje nullázódik.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A játékos és a kapcsoló által vezérelt csapda kapcsolatát vizsgálja. Hiba, ha a játékos, nem tud rálépni a nyitott csapdára vagy rálépve nem hal meg.

- **Bemenet**

createmap
 add player 2 5
 add switch 2 3
 add trap 3 4
 add box 2 4
 command currentplayer move up
 command currentplayer move right
 finish

- **Elvárt kimenet**

switch 2 3
 box 2 3
 currentplayer 2 4
 true
 currentplayer died

8.2.11. Játékos lánát tol kapcsolóra

- **Leírás**

A játékos lánát tol kapcsoló mezőre. Ekkor a kapcsoló mezőre kell a lánának kerülni, azt kapcsolni, majd a játékosnak a doboz eredeti helyére kerülni.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A játékos, doboz és kapcsoló hármas kapcsolatát vizsgálja. Hibás működést jelent, ha a játékos nem mozdul a helyéről, ha a doboz nem tolódik a kapcsolóra és ha a kapcsoló nem nyitja ki a hozzá tartozó csapdát.

- **Bemenet**

```
createmap
add player 2 5
add switch 2 3
add trap 3 4
add box 2 4
command currentplayer move up
finish
```

- **Elvárt kimenet**

```
box 2 3
currentplayer 2 4
true
```

8.3. A tesztelést támogató programok tervezése

A tesztelést támogató programunk parancsok megadása után a finish parancs hatására leellenőrzi a változásokat az adott tesztseteknél, és a kimenetre írja azokat. A finish előtti parancsokkal még a finish be nem lett gépelve még nem foglalkozik, kivéve ha azok nem értelmezhetőek, ekkor a program visszaadja a "hibás parancs" üzenetet, és az adott sort nem veszi figyelembe. A kimenetet utána összehasonlítja a várt eredménnyel.

Minden parancs a createmap-el kell induljon, amely egy fix, csak normalfield-ból álló pályát hoz létre, ahol a koordináták bal fenti sarokból 0,0-val kezdődnek.

A parancsoknál az első hozzáadott player lesz a currentplayer, aki a tesztelésnél többet léphet, nincs játékos váltás.

A parancsoknál az összetartozó switch-et és trap-et az egymás után való addolással jelezzük, ahol a switchet addoljuk elsőnek.

8.4. Napló

Kezdet	Időtartam	Résznevők	Leírás
2018.04.05. 12:00	5 óra	Szatmáry	Osztályok és metódusok tervezése
2018.04.07. 15:30	0.5 óra	Kapitány	Egyes tesztsetek
2018.04.08. 17:00	2 óra	Kapitány	Egyes tesztsetek
2018.04.08. 17:00	2 óra	Králik	Tesztsetek, kisebb formázások, rendbeszedések
2018.04.08. 17:00	2 óra	Máté	Egyes tesztsetek
2018.04.08. 18:00	1 óra	Bende	Osztályleírások bővítése

10. Prototípus beadása

10.1. Fordítási és futtatási útmutató

10.1.1. Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Box.java	6546 bytes	2018.04.23	A pályán tologatható dobozok Box osztálya.
Character.java	9923 bytes	2018.04.23	A játékos által közvetlenül irányított karakterek Character osztálya.
Collidable.java	1035 bytes	2018.04.23	Az ütközésre képes Thing osztályokat összekapcsoló Interface.
Direction.java	202 bytes	2018.04.23	Azon irányok felsorolása, amerre a pályán Thing-ek mozoghatnak.
Field.java	6235 bytes	2018.04.23	Absztrakt ősosztály a pálya mezőire. Ilyen mező objektumokra kerülhetnek rá a Thing-ek.
Game.java	15499 bytes	2018.04.23	Game osztály. A program belépési pontja.
Goal.java	1238 bytes	2018.04.23	Cél osztály, cél pályaelemre kell tolni a dobozokat.
Hole.java	730 bytes	2018.04.23	Lyuk osztály.
Map.java	19239 bytes	2018.04.23	A játék színtere, Map osztály.
NormalField.java	3632 bytes	2018.04.23	Az üres mezőt jelképező NormalField osztály.
Switch.java	2749 bytes	2018.04.23	A csapda (Trap) be, illetve kikapcsolásáért felelős Switch osztály.
Thing.java	1984 bytes	2018.04.23	Absztrakt ősosztály az egyes Field mezőkre rákerülhető dolgoknak.
Timer.java	1090 bytes	2018.04.23	A játékosok idejeinek csökkentéséért felelős Timer osztály.
Trap.java	2300 bytes	2018.04.23	A pályán megjelenő csapdák Trap osztály.
TurnManager.java	12106 bytes	2018.04.23	A körök és lépések menedzseléséért felelős TurnManager osztály.
Visitable.java	1546 bytes	2018.04.23	Olyan osztályok interface, amire dolgok kerülhetnek.
Wall.java	2603 bytes	2018.04.23	Mozdíthatatlan fal elemek Wall osztálya.

10.1.2. Fordítás

Eclipse vagy egyéb Java fejlesztői környezetben a projekt megnyitása, avagy új projekt létrehozása, és az összes a projekthez tartozó forrásfájlok hozzáadása után a projekt/Build All (CTRL+B) menüpont lefuttatása.

Alternatívaként a 6.1.3 pont szerinti útmutató követése, mivel Eclipse alatt a run gomb egyben fordít is.

10.1.3. Futtatás

Eclipse vagy egyéb fejlesztői környezetben a már lefordult program elindítása a Run gomb (tipikusan kerek, zöld, kicsiny háromszöggel a közepében) lenyomása által.

A program elindítása után a parancssoron vár számokat bemenetként, amivel a megfelelő tesztelési menüpontok kiválaszthatóak.

10.2. Fordítási és futtatási útmutató

A tesztesetek vezérlése a részletes tervek beadandó óta a következőkben módosult:

A játékot elindító parancs a “createmap x”-re módosult, ahol az “x” a játékosok számát jelenti.

Ezek után az értelmező a játékosok lehelyezését várja a tesztelőtől, soronként “add player x y” formában, ahol az x és az y az x és y tengely szerinti koordináták. A pálya alapértelmezett statikus mérete 20*20, fix fallal határolt, tehát az x és y is rendre 1-18 értékeket vehet fel.

A karakterek megadása után az objektumok és mezők megadása következik, a következő formában:

“add obj x y”, ahol az obj helyére a következő szavak helyettesíthetők be: “hole”, “goal”, “switch”, “trap”, “wall”, “box”.

Az objektumok és mezők megadását a tesztelő a “finish” parancs megadásával jelezheti, innentől kezdve elindult a játék futtatásáért felelős függvény.

A karaktereket a “w”, “a”, “s”, illetve “d” parancsok megadásával lehet. Ezután a tesztelőnek lehetősége van a méz, avagy olaj lehelyezésére az aktív karakter aktuális pozíciójába a “place honey”, illetve “place oil” parancsokkal, ami után, vagy helyett a mozgást az “f” parancssal végezheti.

A tesztesetek lépésenkénti kimenetét a drawMapTest függvény által körönként generált ASCII karakteres pálya pillanatkép képzi.

10.2.1. Játékos mozgatása üres mezőre

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23

10.2.2. Játékos mozgatása falnak

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23

10.2.3. Játékos-doboz-doboz-tolás

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23
Teszt eredménye	A teszt kimenete a dobozok súlyától függ. A speciális külső hatásoktól mint méz vagy olaj eltekintve egy karakter egy dobozt biztosan el fog tudni tolni, viszont a pontos karakter erők és doboz súlyok specifikálásáig a jelenlegi állás szerint (2018.04.23) két dobozt már nem tud egy karakter eltolni.
Lehetséges hibaok	Karakter erejének és doboz súlyának pontos specifikálása
Változtatások	Végeleges verzióra kóbe véssett értékek.

10.2.4. Méz lerakása

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23
Teszt eredménye	Csak a NormalField mezőnek van weightSum adattagja, így játékos olajat vagy mézet csak arra tud lerakni. Más Field-en nincs értelmezve ez a művelet. Mivel a mezők nem írják ki a súlyukat (így is tele van a pálya elemekkel), ezért erről megbizonyosodni csak a mézezett vagy olajozott nem-NormalField-en keresztül tologatott dobozokkal lehetséges.
Lehetséges hibaok	
Változtatások	

10.2.5. Olaj lerakása

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23
Teszt eredménye	Csak a NormalField mezőnek van weightSum adattagja, így játékos olajat vagy mézet csak arra tud lerakni. Más Field-en nincs értelmezve ez a művelet. Mivel a mezők nem írják ki a súlyukat (így is tele van a pálya elemekkel), ezért erről megbizonyosodni csak a mézezett vagy olajozott nem-NormalField-en keresztül tologatott dobozokkal lehetséges.
Lehetséges hibaok	
Változtatások	

10.2.6. Játékos-doboz-játékos-doboz interakció

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23

10.2.7. Játékos lyukra lép

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23

10.2.8. Doboz mozgatása üres mezőre

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23

10.2.9. Játékos-játékos találkozás

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23

10.2.10. Játékos nyitott csapdára lép

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23

10.2.11. Játékos látótól kapcsolóra

Tesztelő neve	Szatmáry Péter
Teszt időpontja	2018.04.23

10.3. Értékelés

Tag neve	Munka százalékban
Szatmáry	21.36%
Máté	20.36%
Bende	19.36%
Králik	19.53%
Kapitány	19.36%

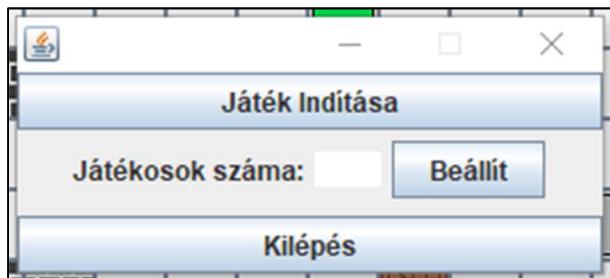
10.4. Napló

Kezdet	Időtartam	Résznevők	Leírás
2018.04.23. 10:00	3 óra	Szatmáry	Korrektúrázás, javítások, osztályokban segédkezés
2018.04.22. 15:00	4 óra	Máté	TurnManager osztályok
2018.04.22. 12:00	7 óra	Bende	Field osztályok
2018.04.22. 12:00	8.5 óra	Králik	Map osztály
2018.04.22. 10:00	10 óra	Kapitány	Thing osztályok

11. Grafikus felület specifikációja

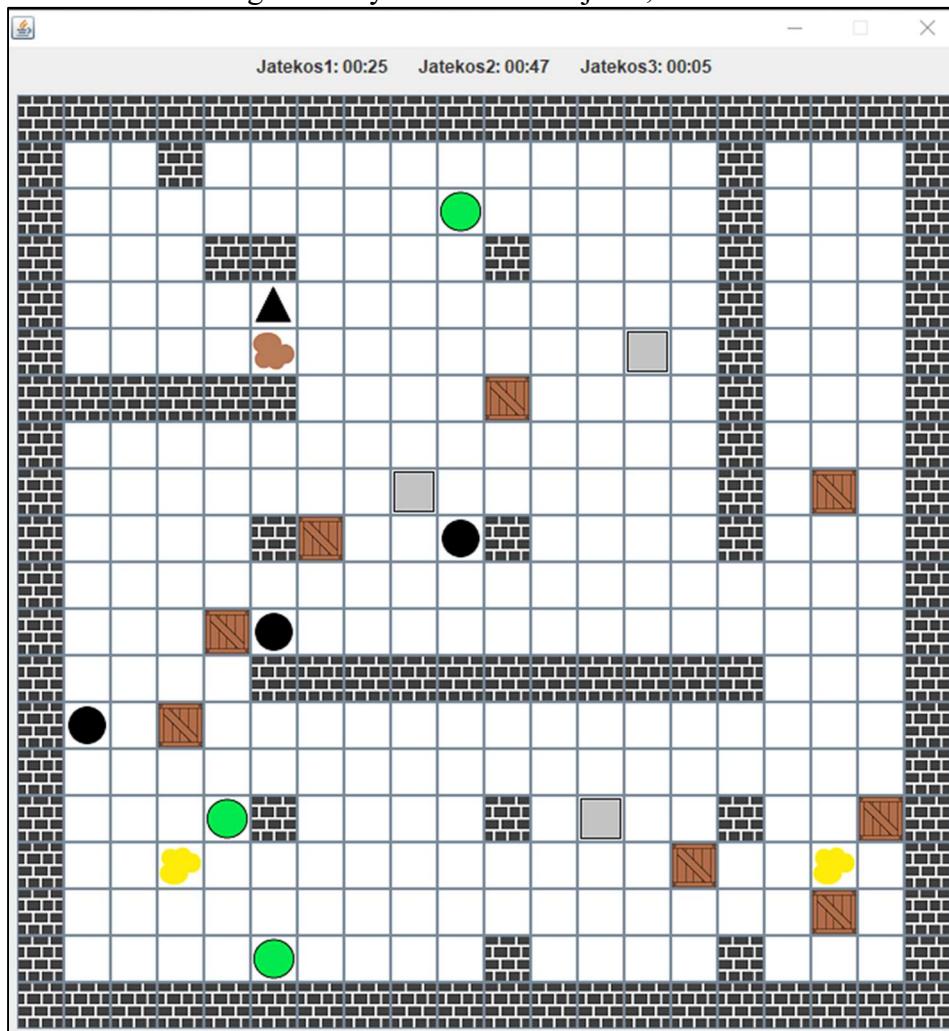
11.1. A grafikus interfész

A szoftvert elindítva a következő grafikus ablak jelenik meg:



Az alap menüben három dolgot tehetünk. A legfelső gombbal ("Játék Indítása") elindíthatjuk a játékot. Középen a "Játékosok száma:" pont mellet megadhatjuk a játékban résztvevő játékosok számát, majd a "Beállít" gombbal rögzíthetjük is ezt. Ezen kívül még egy gomb található legalul, a "Kilépés" gomb, ami értelemszerűen kilép a játékból, bezárja az alkalmazást.

A "Játék Indítása" gombra nyomva elindul a játék, és az alábbihoz hasonló ablak jelenik meg:

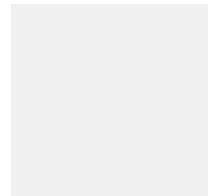
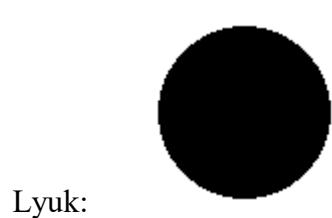
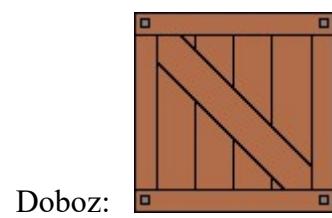
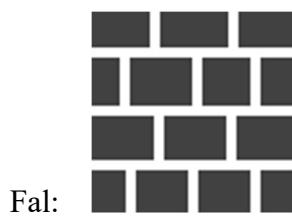


Ez csupán egy referencia pálya, a pályán szereplő elemeket szemlélteti, a végleges változat ettől eltérhet.

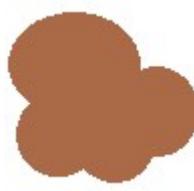
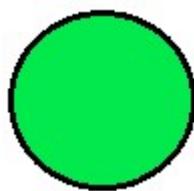
Az ablak tetején található egy panel, amely a játékosok nevét és hátralévő idejét jeleníti meg. Ez csak információk jelzésére szolgál.

Alatta található maga a teljes játéktér (a pálya) és rajta az elhelyezett objektumok. Ezeket az objektumokat magától értetődő ábrák jelzik, amelyek később, *esztétikai okokból még változhatnak*, hogy a felhasználók többsége "jól kinézőnek" találja.

Áttekintés a jelenlegi (nem végleges) ábrákról:



Játékos (különböző színű háromszögek):



11.2. A grafikus rendszer architektúrája

A játék grafikus megjelenítésére a Java Swing-et használjuk.

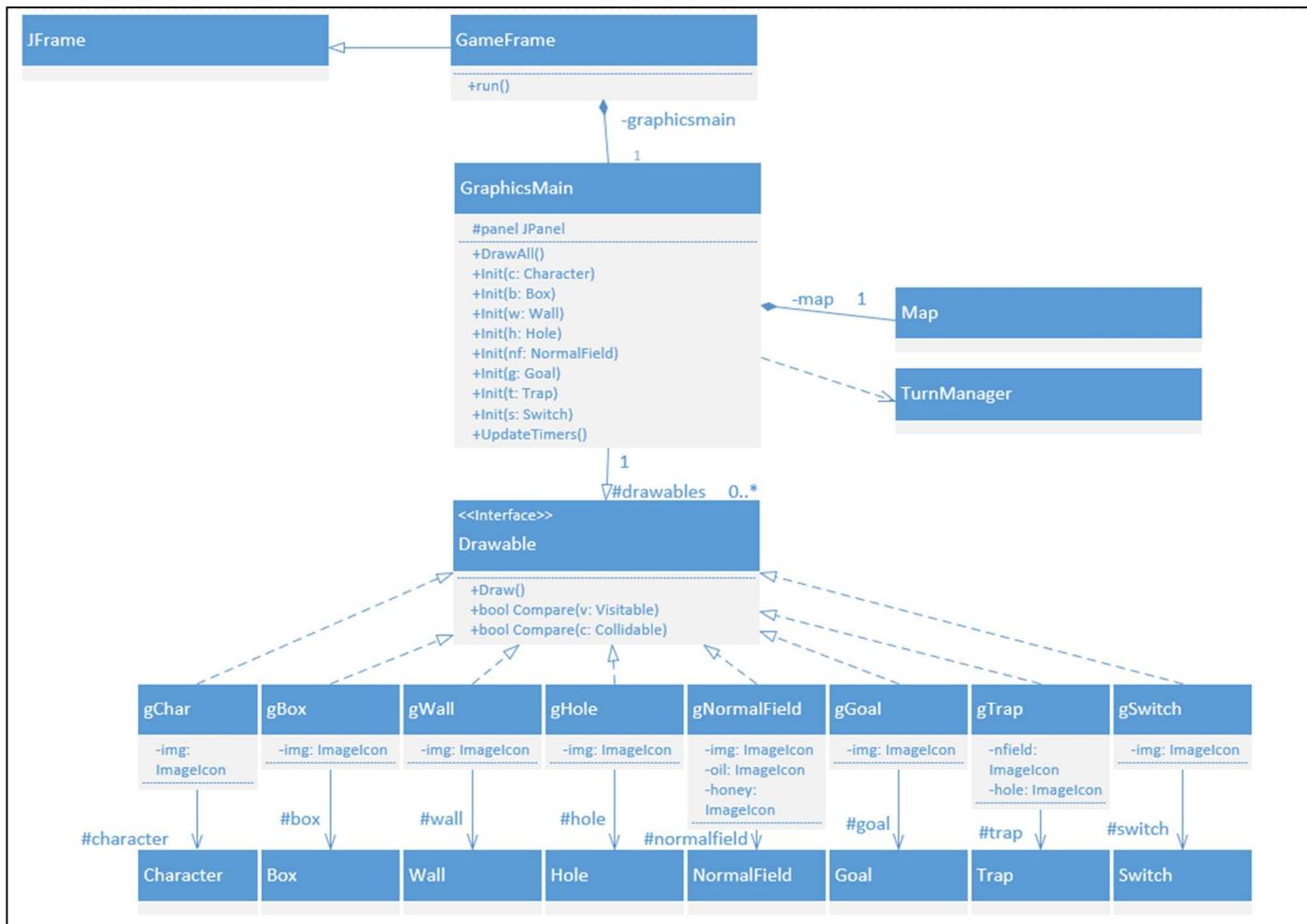
11.2.1. A felület működési elve

A grafikai felülete a programnak modell-view-controller (modell-nézet-vezérlő) szerkezeti mintát követi. Miszerint a nézet és a modell szétválasztódik, "függetlenek" egymástól.

A programunk minden másodpercben kirajzolja a pályát ötször, ezt egy előre elindított szál biztosítja, amely öt másodpercenként szól a GraphicsMain osztálynak rajzolás történjen.

Kirajzoláskor egy a pálya mezőit tartalmazó mátrix segítségével rajzolunk, amely kirajzolja a mátrix jelenlegi elemét, amennyiben semmit nem tartalmazó Field, a Fieldet, típusától függően, amennyiben tartalmaz valamit, a tartalmazott dolgot, szintén típusától függően.

11.2.2. A felület osztály-struktúrája



11.3. A grafikus objektumok felsorolása

11.3.1. gChar

- **Felelősség**

Egy karakter kirajzolása.

- **Interfészek**

Drawable

- **Attribútumok**

-img: ImageIcon : A karakter képe.

#character: Character : Referencia arra a karakterre, amit grafikusan reprezentál.

- **Metódusok**

+Draw(): void : Kirajzolja a karaktert miután lekérdezte aktív-e, ha aktív, kirajzolja a sima ImageIcon-t, ha nem aktív, akkor módosítja a színét az adott karakter egyedi színére.

+Compare(v: Visitable): bool : Mivel Visitable .nem egyezhet meg Character-rel soha, visszatér false-al.

+Compare(c: Collidable): bool : Megvizsgálja a paraméterként kapott Collidable megegyezik-e a Character tagváltozóval, amennyiben igen, visszatér true-val, ha nem, visszatér false-al.

11.3.2. gBox

- **Felelősség**

Egy doboz kirajzolása.

- **Interfész**

Drawable

- **Attribútumok**

-img: ImageIcon : A doboz képe.

#box: Box : Referencia arra a dobozra, amit grafikusan reprezentál.

- **Metódusok**

+Draw(): void : Kirajzolja a dobozt.

+Compare(v: Visitable): bool : Mivel Visitable nem egyezhet meg Box-szal soha, visszatér false-al.

+Compare(c: Collidable): bool : Megvizsgálja a paraméterként kapott Collidable megegyezik-e a Box tagváltozóval, amennyiben igen, visszatér true-val, ha nem, visszatér false-al.

11.3.3. gWall

- **Felelősség**

Egy fal kirajzolása.

- **Interfészek**

Drawable

- **Attribútumok**

-img: ImageIcon : A fal képe.

#wall: Wall : Referencia arra a falra, amit grafikusan reprezentál.

- **Metódusok**

+Draw(): void : Kirajzolja a falat.

+Compare(v: Visitable): bool : Mivel Visitable nem egyezhet meg Wall-lal soha, visszatér false-al.

+Compare(c: Collidable): bool : Megvizsgálja a paraméterként kapott Collidable megegyezik-e a Wall tagváltozóval, amennyiben igen, visszatér true-val, ha nem, visszatér false-al.

11.3.4. gHole

- **Felelősség**

Egy lyuk kirajzolása.

- **Interfészek**

Drawable

- **Attribútumok**

-img: ImageIcon : A lyuk képe.

#hole: Hole : Referencia arra a lyukra, amit grafikusan reprezentál.

- **Metódusok**

+Draw(): void : Kirajzolja a lyukat.

+Compare(v: Visitable): bool : Megvizsgálja a paraméterként kapott Visitable megegyezik-e a Hole tagváltozóval, amennyiben igen, visszatér true-val, ha nem, visszatér false-al

+Compare(c: Collidable): bool : Mivel Collidable nem egyezhet meg Hole-lal soha, visszatér false-al.

11.3.5. gNormalField

- **Felelősség**

Egy sima mező kirajzolása.

- **Interfészek**

Drawable

- **Attribútumok**

-img: ImageIcon : A sima mező képe.

-oil: ImageIcon : Az olajos mező képe.

-honey: ImageIcon : A mézes mező képe.

#normalfield: NormalField : Referencia arra a sima mezőre, amit grafikusan reprezentál.

- **Metódusok**

+Draw(): void : Kirajzolja a megfelelő mezőt.

+Compare(v: Visitable): bool : Megvizsgálja a paraméterként kapott Visitable megegyezik-e a Hole tagváltozóval, amennyiben igen, visszatér **true**-val, ha nem, visszatér **false**-al

+Compare(c: Collidable): bool : Mivel Collidable nem egyezhet meg Hole-lal soha, visszatér false-al.

11.3.6. gGoal

- **Felelősség**

Egy célmező kirajzolása.

- **Interfészek**

Drawable

- **Attribútumok**

-img ImageIcon : A cél mező képe

#goal: Goal : Referencia arra a cél mezőre, amit grafikusan reprezentál.

- **Metódusok**

+Draw(): void : Kirajzolja a célmezőt.

+Compare(v: Visitable): bool : Megvizsgálja a paraméterként kapott Visitable megegyezik-e a Hole tagváltozóval, amennyiben igen, visszatér **true**-val, ha nem, visszatér **false**-al

+Compare(c: Collidable): bool : Mivel Collidable nem egyezhet meg Hole-lal soha, visszatér false-al.

11.3.7. gTrap

- **Felelősség**

Egy csapda kirajzolása.

- **Interfészek**

Drawable

- **Attribútumok**

-field: ImageIcon : A sima mező képe.

-hole: ImageIcon : A lyuk képe.

#trap: Trap : Referencia arra a csapdára, amit grafikusan reprezentál.

- **Metódusok**

+Draw(): void : Kirajzolja a csapdát a megfelelő állapotban, ezt a trap tagváltozó getState() függvényhívásával dönti el.

+Compare(v: Visitable): bool : Megvizsgálja a paraméterként kapott Visitable megegyezik-e a Hole tagváltozóval, amennyiben igen, visszatér **true**-val, ha nem, visszatér **false**-al

+Compare(c: Collidable): bool : Mivel Collidable nem egyezhet meg Hole-lal soha, visszatér false-al.

11.3.8. gSwitch

- **Felelősség**

Egy kapcsoló kirajzolása.

- **Interfészek**

Drawable

- **Attribútumok**

-img: ImageIcon : A kapcsoló képe

#switch: Switch : Referencia arra a kapcsolóra, amit grafikusan reprezentál.

- **Metódusok**

+Draw(): void : Kirajzolja a kapcsolót.

+Compare(v: Visitable): bool : Megvizsgálja a paraméterként kapott Visitable megegyezik-e a Hole tagváltozóval, amennyiben igen, visszatér **true**-val, ha nem, visszatér **false**-al

+Compare(c: Collidable): bool : Mivel Collidable nem egyezhet meg Hole-lal soha, visszatér false-al.

11.3.9. GameFrame

- **Felelősség**

A grafikus megjelenítés alapja, ezen található minden elem.

- **Ősosztályok**

JFrame

- **Attribútumok**

-graphicsMain: GraphicsMain: A grafikus felület fő tartalmát (JPanel és elemei), továbbá a működési logikát tartalmazó objektum.

- **Metódusok**

+run(): A grafikus réteg üzleti logikájának komponenseit egyesíti, rendezи egy nagy egésszé. Inicializálja az objektumokat, időzít, meghívja a kirajzolandó objektumoknak a kirajzoló metódusait.

11.3.10. GraphicsMain

- **Felelősség**

A pályána elemeinek megfelelő grafikus elemek kirajzolását végzi, és frissíti a játékosok idejét.

- **Ősosztályok**

-

- **Interfészek**

-

- **Attribútumok**

map: Map : A pályán található objektumok pozícióját tartalmazó osztály referenciája.

panel: JPanel : Ezen a panelen jelennek meg a Fieldeket és Thingeket jelképező ábrák.

- **Metódusok**

- **+DrawAll(): void**

Végigfut a pálya fieldjein és lekérdezi üresek-e. Ha üres, meghívja az Init() függvényt az aktuális Field paraméterezésével, ha nem üres, a getThing() függvény által visszaadott Thing paraméterezésével hívja meg.

- **+Init(c : Character): void**

Összehasonlíta a drawables mátrix aktuális elemének tagváltozóját, megegyezik-e a kapott (Character) paraméterrel. Ha nem egyezik meg, a Compare függvény visszatér hamisként, és töröljük a drawables mátrix helytelen aktuális elemét, majd létrehozunk egy újat, jelen esetben gChar-t a paraméterként kapott Character paraméterezésével, és meghívjuk a Draw() függvényét.

Ha megegyezik a Character a drawables mátrix adott elem tagváltozójával, akkor nem kell kirajzolni újra, de mivel ez Character, aktív/inaktív mivoltja változhat, ezért minden kirajzoljuk, és ekkor is meghívjuk a Draw()-t.

- **+Init(b : Box): void**

Összehasonlítja a drawables mátrix aktuális elemének tagváltozóját, megegyezik-e a kapott (Box) paraméterrel. Ha nem egyezik meg, a Compare függvény visszatér hamisként, és töröljük a drawables mátrix helytelen aktuális elemét, majd létrehozunk egy újat, jelen esetben gBox-t a paraméterként kapott Boxparaméterezésével, és meghívjuk a Draw() függvényét.

Ha megegyezik a Box a drawables mátrix adott elem tagváltozójával, akkor nem kell kirajzolni újra.

- **+Init(w : Wall): void**

Összehasonlítja a drawables mátrix aktuális elemének tagváltozóját, megegyezik-e a kapott (Wall) paraméterrel. Ha nem egyezik meg, a Compare függvény visszatér hamisként, és töröljük a drawables mátrix helytelen aktuális elemét, majd létrehozunk egy újat, jelen esetben gWall-t a paraméterként kapott Wall paraméterezésével, és meghívjuk a Draw() függvényét.

Ha megegyezik a Wall a drawables mátrix adott elem tagváltozójával, akkor nem kell kirajzolni újra.

- **+Init(h : Hole): void**

Összehasonlítja a drawables mátrix aktuális elemének tagváltozóját, megegyezik-e a kapott (Hole) paraméterrel. Ha nem egyezik meg, a Compare függvény visszatér hamisként, és töröljük a drawables mátrix helytelen aktuális elemét, majd létrehozunk egy újat, jelen esetben gHole-t a paraméterként kapott Hole paraméterezésével, és meghívjuk a Draw() függvényét.

Ha megegyezik a Wall a drawables mátrix adott elem tagváltozójával, akkor nem kell kirajzolni újra.

- **+Init(nf : NormalField): void**

Összehasonlítja a drawables mátrix aktuális elemének tagváltozóját, megegyezik-e a kapott (NormalField) paraméterrel. Ha nem egyezik meg, a Compare függvény visszatér hamisként, és töröljük a drawables mátrix helytelen aktuális elemét, majd létrehozunk egy újat, jelen esetben gNormalField-t a paraméterként kapott NormalField paraméterezésével, és meghívjuk a Draw() függvényét.

Ha megegyezik a NormalField a drawables mátrix adott elem tagváltozójával, akkor nem kell kirajzolni újra.

- **+Init(g : Goal): void**

Összehasonlítja a drawables mátrix aktuális elemének tagváltozóját, megegyezik-e a kapott (Goal) paraméterrel. Ha nem egyezik meg, a Compare függvény visszatér hamisként, és töröljük a drawables mátrix helytelen aktuális elemét, majd létrehozunk egy újat, jelen esetben gGoal-t a paraméterként kapott Goal paraméterezésével, és meghívjuk a Draw() függvényét.

Ha megegyezik a Goal a drawables mátrix adott elem tagváltozójával, akkor nem kell kirajzolni újra.

- **+Init(t : Trap): void**

Összehasonlítja a drawables mátrix aktuális elemének tagváltozóját, megegyezik-e a kapott (Trap) paraméterrel. Ha nem egyezik meg, a Compare függvény visszatér hamisként, és töröljük a drawables mátrix helytelen aktuális elemét, majd létrehozunk egy újat, jelen esetben gTrap-t a paraméterként kapott Trap paraméterezésével, és meghívjuk a Draw() függvényét.

Ha megegyezik a Trap a drawables mátrix adott elem tagváltozójával, akkor nem kell kirajzolni újra, de a Trap esetében kirajzoltatjuk (Draw()), mert az állapota nyitott/zárt ezen felül változhatott.

- **+Init(s : Switch): void**

Összehasonlítja a drawables mátrix aktuális elemének tagváltozóját, megegyezik-e a kapott (Switch) paraméterrel. Ha nem egyezik meg, a Compare függvény visszatér hamisként, és töröljük a drawables mátrix helytelen aktuális elemét, majd létrehozunk egy újat, jelen esetben gSwitch-t a paraméterként kapott Switch paraméterevezésével, és meghívjuk a Draw() függvényét.

Ha megegyezik a Switch a drawables mátrix adott elem tagváltozójával, akkor nem kell kirajzolni újra.

- **+UpdateTimers(): void**

Frissíti a grafikán az összes játékos idejét.

11.3.11. Drawable

- **Felelősség**

Interface. minden grafikusan kirajzolható osztály megvalósítja, ennek segítségével történik meg a kirajzolás.

- **Metódusok**

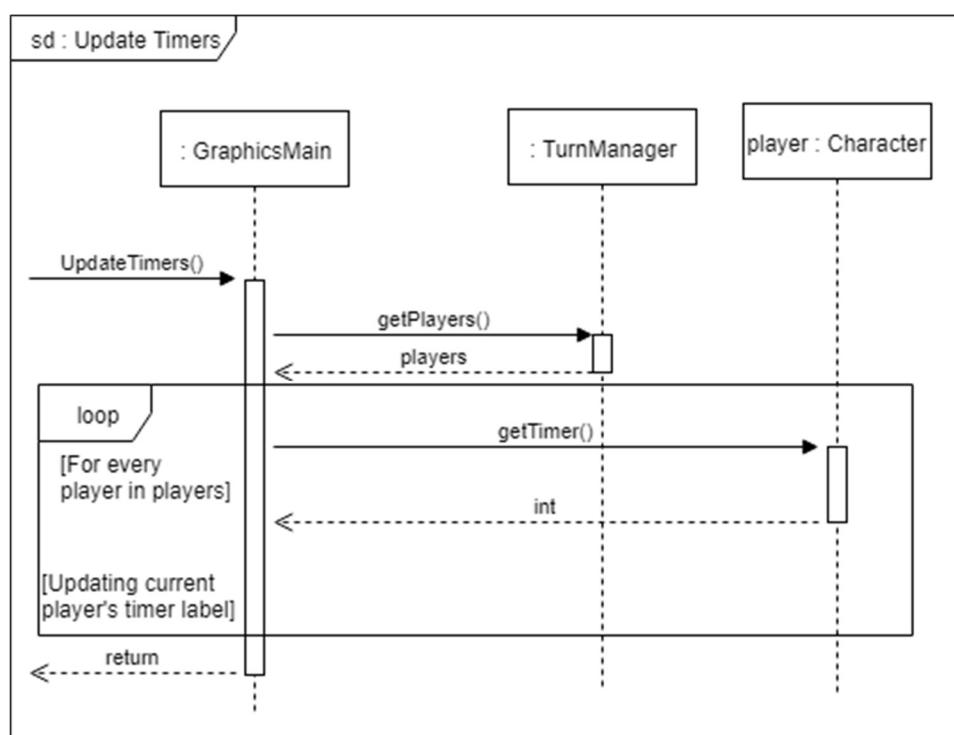
+ Draw(): void : Különböző objektumok kirajzolásáért felelős az egyes pályaelemekhez tartozó kirajzoló osztályokban.

+ Compare(v: Visitable) : bool: Összehasonlít két Visitable interfész megvalósító objektumot.

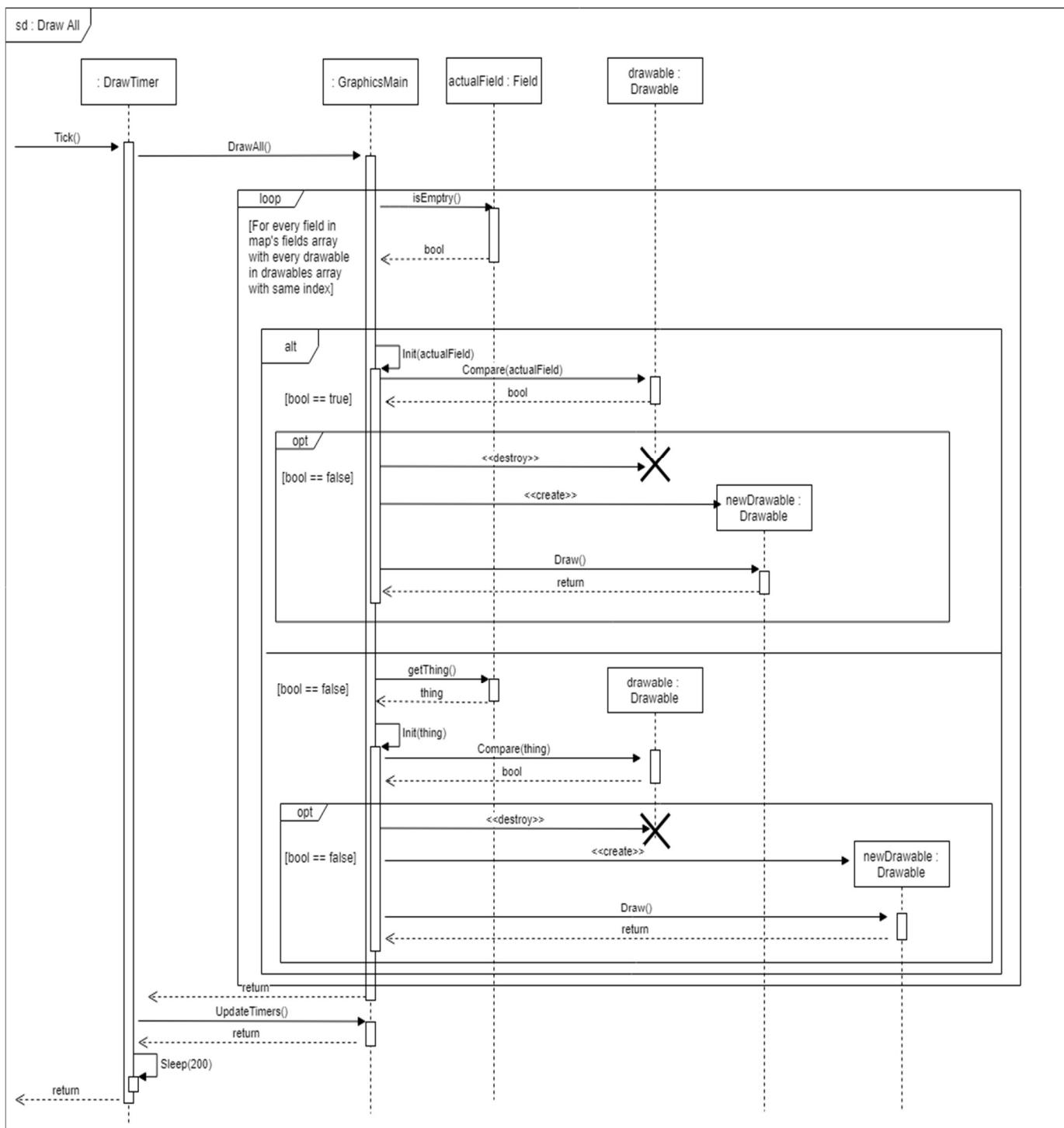
+ Compare(c: Collidable) : bool: Összehasonlít két Collidable interfész megvalósító objektumot.

11.4. Kapcsolat az alkalmazói rendszerrel

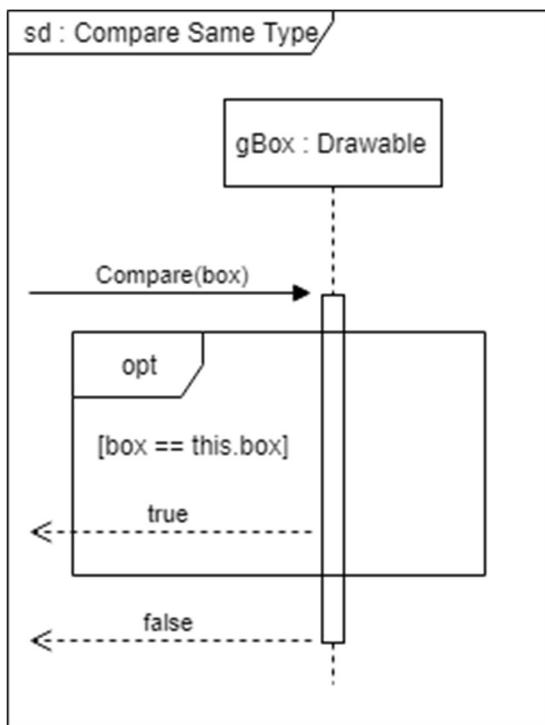
11.4.1. Update Timers



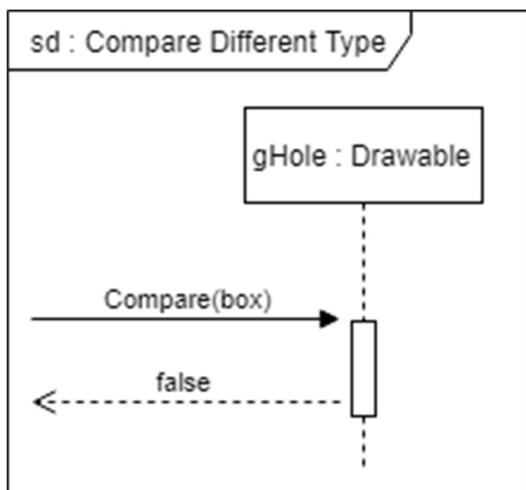
11.4.2. Draw All



10.1.1. Compare Same Type



10.1.2. Compare Different Type



11.5. Napló

Kezdet	Időtartam	Résznevők	Leírás
2018.04.30 16:00	1 óra	Szatmáry Máté Bende Králik Kapitány	Értekezlet: Feladat megbeszélése, megoldási terv készítése
2018.04.30 17:00	2 óra	Szatmáry	Osztálydiagram elkészítése
2018.04.30 19:00	2 óra	Máté	Működési terv és architektúra
2018.05.01 14:30	4 óra	Bende	Osztályleírások írása
2018.05.01 17:00	3 óra	Králik	Grafikus interface megtervezése
2018.05.01 14:30	3.5 óra	Kapitány	Szekvenciadiagramok megtervezése

13. Grafikus változat beadása

13.1. Fordítási és futtatási útmutató

13.1.1. Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Box.java	4791 byte	2018.05.13	A pályán található dobozok osztálya.
Character.java	7989 byte	2018.05.13	A pályán található karakterek osztálya.
Collidable.java	1035 byte	2018.05.13	Ütközésre képes Thing-ek interface.
Direction.java	202 byte	2018.05.13	A mozgási irányok felsorolása.
Drawable.java	636 byte	2018.05.13	A kirajzolandó osztályok interface.
Field.java	5017 byte	2018.05.13	Absztrakt ösosztály a pálya mezőire.
Game.java	1438 byte	2018.05.13	A program belépési pontja.
GameFrame.java	9572 byte	2018.05.13	A grafika alapja és inputkezelés.
gBox.java	1124 byte	2018.05.13	A kirajzolandó dobozok osztálya.
gChar.java	1309 byte	2018.05.13	A kirajzolandó karakterek osztálya.
gGoal.java	1133 byte	2018.05.13	A kirajzolandó célok osztálya.
gHole.java	1134 byte	2018.05.13	A kirajzolandó lyukak osztálya.
gNormalField.java	1656 byte	2018.05.13	A kirajzolandó üres mezők osztálya.
gSwitch.java	1282 byte	2018.05.13	A kirajzolandó kapcsolók osztálya.
gTrap.java	1269 byte	2018.05.13	A kirajzolandó csapdák osztálya.
gWall.java	1117 byte	2018.05.13	A kirajzolandó falak osztálya.
Goal.java	1270 byte	2018.05.13	A pályán található célok osztálya.
Graphics.java	1919 byte	2018.05.13	Grafikai segédinterface.
GraphicsMain.java	9041 byte	2018.05.13	A grafika kirajzolásáért felelős.
Hole.java	1111 byte	2018.05.13	A pályán található lyukak osztálya.
Map.java	11883 byte	2018.05.13	A játék színtérét tartalmazza.
NormalField.java	3546 byte	2018.05.13	A pályán található üres mezők osztálya.
Switch.java	2072 byte	2018.05.13	A pályán található kapcsolók osztálya.
Thing.java	1704 byte	2018.05.13	Absztrakt ösosztály a pálya elemeire.
Trap.java	2250 byte	2018.05.13	A pályán található csapdák osztálya.
TurnManager.java	6146 byte	2018.05.13	A körökért és lépésekért felel.
Visitable.java	1579 byte	2018.05.13	Interface a Field mezőkhöz.
Wall.java	2541 byte	2018.05.13	A pályán található falak osztálya.
Cel30.png	945 byte	2018.05.08	A cél mezők kisképe.
Doboz30.png	1022 byte	2018.05.08	A dobozok kisképe.
Fal230.png	775 byte	2018.05.08	A falak kisképe.
Jatekos30.png	447 byte	2018.05.08	Az inaktív játékosok kisképe.
JatekosActive30.png	230 byte	2018.05.08	Az aktív játékos kisképe.
Kapcsolo30.png	243 byte	2018.05.08	A kapcsoló mezők kisképe.
Lyuk30.png	442 byte	2018.05.08	A lyuk mezők kisképe.
Mezesmezo30.png	780 byte	2018.05.08	A mezőkre letett méz kisképe.
NormalField1.png	320 byte	2018.05.08	Az üres mezők kisképe.
Olajosmezo30.png	908 byte	2018.05.08	A mezőkre letett olaj kisképe.
readme.txt	195 byte	2018.05.13	Képelhelyezési útmutató
Timer.java	0 byte	2018.05.13	nem használt, benne maradt...
DrawTimer.java	0 byte	2018.05.13	nem használt, benne maradt...

13.1.2. Fordítás és telepítés

Eclipse vagy egyéb Java fejlesztői környezetben a projekt megnyitása, avagy új projekt létrehozása, és az összes a projekthez tartozó forrásfájlok hozzáadása után a projekt/Build All (CTRL+B) menüpont lefuttatása.

Alternatívaként a 6.1.3 pont szerinti útmutató követése, mivel Eclipse alatt a run gomb egyben fordít is.

Esetlegesen szükséges lehet a forráskódok mellett beadott játékelemeket szimbolizáló kisképek a (readme.txt-ben is leírt módon) projekt gyökérmapjába való áthelyezésére.

13.1.3. Futtatás

Eclipse vagy egyéb fejlesztői környezetben a már lefordult program elindítása a Run gomb (tipikusan kerek, zöld, kicsiny háromszöggel a közepében) lenyomása által.

A program elindítása után a megjelenő ablakban meg kell adni a játékosok számát, majd a "Beállít" gomb lenyomása után, a "Játék indítása" feliratú gombra balklikkelve elindul a játék. A mindenkor aktív karaktert sárga színű háromszög jelzi, azt a W, A, S, illetve D billentyűk lenyomásával lehet irányítani, továbbá mézet vagy olajat rendre a H, illetve O gombok lenyomásával lehet lehelyezni.

A játék akkor ér véget, ha minden karakter vagy meghal, vagy lejár az összes ideje. Ilyenkor az eredmények ablakot bezárva vagy az OK gombra kattitva lehet kilépni a programból.

13.2. Értékelés

Tag neve	Munka százalékban
Szatmáry	20.00 %
Máté	20.00 %
Bende	20.00 %
Králik	20.00 %
Kapitány	20.00 %

13.3. Napló

Kezdet	Időtartam	Résznevők	Leírás
2018.05.13 15:00	3 óra	Szatmáry	Eseménykezelők megírása, hibák javítása, kommentezés.
2018.05.13 15:00	6 óra	Máté	Tesztelés, hibák keresése.
2018.05.13 14:00	7 óra	Bende	Játék kezdeti és végi logikájának implementálása.
2018.05.13 14:00	7.5 óra	Králik	Grafikus osztályok implementálása.
2018.05.13 14:00	7.5 óra	Kapitány	Grafikus osztályok implementálása.

14. Összefoglalás

14.1. A projektre fordított összes munkaidő

Tag neve	Munkaidő (óra)
Bende Zoltán	70
Szatmáry Péter	70
Králik Bálint Bendegúz	70
Máté László	70
Kapitány Gergely	70
Összesen	350

- A feltöltött programok forrássorainak száma

Fázis	Kódsorok száma
Szkeleton	731 kód + 182 komment + 182 üres sor
Prototípus	739 kód + 1277 komment + 279 üres sor
Grafikus változat	1279 kód + 930 komment + 345 üres sor
Összesen	2749 kód + 2389 komment + 806 üres sor

14.2. Projekt összegzés

14.2.1. Mit tanultak a projektből konkrétan és általában?

Bende Zoltán: Hogy milyen nehézségekbe ütközik egy projekt elkészítése csapatmunkával, illetve hogy milyen problémák lépnek fel egy viszonylag nagy méretű projekt elkészítése közben. Megtanultam, mennyire fontos, hogy előre megtervezzük a programot, és ne azonnal kezdjük el irni a kódot.

Szatmáry Péter: A csapatmunkának, a feladatok közös átbeszélésének, és megfelelő kiosztásának fontosságát. mindenki másban jobb, ne féljük segítséget kérni, vagy másra rábízni az adott részfeladatot, ha tudjuk, hogy úgy effektívebbek vagyunk vagy jobb lesz a végeredmény.

Továbbá időmenedzsmentet - így sem buktunk el rajta, de ha minden időben elkezdtünk volna, hiszem, hogy még sokkal jobb lehetett volna a projekt végterméke. Fontos, hogy jól meg tudjuk előre becsülni, hogy adott részfeladat mennyi időt fog felemészteni, s ez által mikor érdemes elkezdeni vele foglalkozni. A feladat részletes tervezése bár iszonyatosan sok időt vesz el, de a kódolás utána már ment, mint a karikacsapás – úgy érzem jobbára megérte a sok szenvedést.

Králik Bálint Bendegúz: A projekt felvilágosított arról hogy egy projekt fejlesztése igen komplikált dolog. A fejlesztés során baromi sok komplikáció lépett fel, olyanok is amilyenekre egyáltalán nem gondoltunk. Emellett nehéz volt öt ember idejét úgy beosztani, hogy egy adott időben mindenjában ráérjünk.

Máté László: Általánosságban annyit, hogy meg lehet bízni a csapattársakban. Lehet, hogy csak szerencsésen jött ki (más csapatoknál akadtak gondok), de senki nem lógott el, mindenki csinálta a feladatot a megbeszéltek szerint. A projektből azt tanultam meg leginkább, hogy nem szabad az utolsó pillanatra halogatni a dolgot, menet közben sok tervezési/figyelmetlenségi hiba előjön, jobban be kell osztani az időt.

Kapitány Gergely: Segített megtapasztalni milyen csapatban dolgozni, mekkora előnyei vannak. Segített megtanulni hogyan kell nekikezdeni egy projektnek, mikre kell figyelni, és mennyire jó ha szervezett a munka, mindenki egy olyan részen dolgozik amihez jobban ért, és hogy ezáltal mennyivel könnyebb lehet egy csoport munka.

14.2.2. Mi volt a legnehezebb és a legkönnyebb?

Bende Zoltán: A legnehezebb a rendszer megtervezése volt, a különböző osztálydiagramok kitalálása volt, illetve a szekvenciadiagramok kitalálása és létrehozása volt. A legkönnyebb pedig a feladat részletes specifikációja volt.

Szatmáry Péter: A legnehezebb a projektmenedzsment, vagy ahogyan már említettem az időmendzsment volt. A legtöbb gondot messze az okozta, hogy összeegyeztessük hétről hétre azokat az időintervallumokat, amikor mind, de legalább többen ráérünk. mindenkinél máskor vannak más tárgyai, máskor megy laborra, máskor ír zárhelyiket, és a többi... Emiatt többször is kissé megcsúsztunk az idővel - ami szerencsére ez sosem ment a teljes leadandó rovására, de okozott pár kellemetlen átdolgozott vasárnap éjszakát a hétfői leadási határidő előtt. A legkönnyebb szerintem is a specifikációk megírása volt. A kódolás sem volt kifejezetten nehéz, de az már időnként okozott kisebb bonyodalmakat, mikor adott függvény nem tudott különbséget tenni az egyes leszármazott osztályok között, aztán új interfaceket, és más módszereket kellett keresnünk.

Králik Bálint Bendegúz: A legnehezebb talán a játék osztály diagramjának összerakása volt, itt ki kellett találni, hogy mi, mit, mikor és hogyan ér el, amely jelentős gondolkodást igényelt az egész csapatból. A legkönnyebb a konkrét kódolás volt, hiszen ilyenkor már rendelkezésre álltak a diagramok, amelyek alapján szinte már csak agy nélkül gépelni kellet. Persze ilyenkor derültek ki olyan hibák amelyekre nem számítottunk és a működést akadályozták.

Máté László: A legnehezebb egyértelműen a (sokszor feleslegesnek tűnő) leírások és diagramok elkészítése volt szerintem. A legkönnyebb része a diagramok alapján kódolás volt, rutinszerű feladatnak tűnt.

Kapitány Gergely: A legnehezebb esetleg az időbeosztás. Mivel még nem tapasztaltuk egy-egy rész mennyire hosszú időt ölel fel, volt hogy elcsúsztunk és még dolgozni kellett rajta közel beadás előtt. Legkönnyebb része talán a már ismert program működés, diagramok alapján a kód megírása volt.

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

Bende Zoltán: A pontszámmal lehet, hogy összhangban áll, de a kreditek számával már kevésbé, ez a tárgy, pláne a sok dokumentációval együtt semmiképp se egy három kredites tárgy.

Szatmáry Péter: A pontszámokkal összhangban állt; papíron a kreditértékkal is, mivel ha megnézzük a fejénként belefektetett órák számát, akkor még pozitívan is jöttünk ki belőle. Gyakorlatilag viszont mindenki tudja, hogy a tárgyakba ténylegesen is befektetendő munka általában köszönőviszonyban sincs a kreditértékekkel. A BSc képzés egyik leg-szakmaibb téma lenne, én úgy gondolom, hogy a 3 kreditnél ez talán többet érhetne - eleve több ilyenfajta gyakorlatias téma lenne szükség a BProf tanterv mintájára ezen a képzésen is.

Králik Bálint Bendegúz: Szerintem igen, csak kevéssére jutott eszembe ilyesmi, hogy jó lett volna több idő, de akkor sem a feladat nehézsége okozta a problémát, hanem hogy más téma ból közeledett valami határidő, zh, ilyesmi. A pontszámok szerintem megfelelően voltak kiosztva feladatonként.

Máté László: Igen, a feladatokra adott idő általában rendben volt, a gondot inkább az jelentette esetenként, hogy más tantárgyakból egyenlőtlennel voltak ütemezve a feladatok, volt, hogy ráérősebben voltunk, volt, hogy nagyon nehéz volt időpontot találni. A pontszámokkal személy szerint nem foglalkoztam igazán, erről nem tudok véleményt mondani.

Kapitány Gergely: Általában igen, inkább néha hamarabb kellett volna elkezdeni, de az idő és a pontszám lehet mondani összhangban állt az elvégzendő feladatokkal.

14.2.4. Ha nem, akkor hol okozott ez nehézséget?

Bende Zoltán: Időnként rosszul osztottuk be az időket, és sokkal több időt szántunk a feladatra, mint amennyivel meg lehetett volna csinálni azt.

Szatmáry Péter: Főleg a saját hibáinknak köszönhető, de gyakran kellett az utolsó pillanatban is (értsd. hétfőn $\frac{1}{2}$ 2-kor javítani dolgokat, vagy készíteni el az utolsó grafikont) foglalkozni vele.

Králik Bálint Bendegúz: -

Máté László: Nem emlékszem pontosan.

Kapitány Gergely: Talán a legelső alkalomnál amikor a program kódot kellett beadni, hamarabb kellett volna nekiesni, mivel lehet az volt a leghosszabb, illetve ekkor derültek ki esetleges hibák, amiknek volt, hogy teljes átváriálására volt szükség, ami még több időt vett igénybe.

14.2.5. Milyen változtatási javaslatuk van?

Bende Zoltán: Mindenképpen kevesebb felesleges dokumentáció kéne a téma ból, egy jó részénél nem érzem, hogy bármilyen valós hasznuk lenne, és ténylegesen a mai ipari viszonyokra készitené fel a hallgatókat.

Szatmáry Péter: Ha nem is direktben a Szoftver Projekt Laboratórium, de úgy vélem, hogy a Szoftvertechnológia téma keretében érdemes lett volna foglalkozni, a név elhadárasán túl is kicsit a verziókezelő rendszerekkel. minden cég használja őket az iparban, mi is használtuk őket valamilyen módon - de lehet egyedül vagyok vele, nekem még mindig nem teljesen világos a használatuk mélylélektana: ki, mikor, milyen branch-en dolgozzon, hogyan oldjuk

meg merge conflict-okat, meg ilyenek. Többször azon kaptam magam, hogy minden létező módosítást csak a master-be töltöttünk fel, ha meg sipákolt a git, hogy conflict van, akkor nekiestem a kódnak az online editorban ctrl+c, ctrl+v-vel felvértezve.

Jó lenne ha egy online felületen nyomon lehetne kötni a határidőket (visszaszámlálóval és értesítésekkel), az eddig megszerzett pontokat.

Továbbá a Hercules írhatná egyértelmű és könnyen emészthető módon a feltöltések határidejeit.

Králik Bálint Bendegúz: Érdemes lehet olyan feladatokat megfogalmazni, amelyek törekednek elérni, hogy a program esztétikailag felhasználó barát legyen. Ha a google-be beírjuk hogy sokban, baromi sok olyan lefejlesztett játék jön ki ami “ronda”.

Máté László: Hercules felületen lehessen hétről hétre követni a pontszámokat és javaslatokat.

Kapitány Gergely: Kisebb javaslat nagyobb feladat részknél a korai elkezdésre. Tapasztalat hiján nem könnyű előrebecsülni, és volt hogy leadás előtt dolgoztunk rajta.

14.2.6. Milyen feladatot ajánlanának a projektre?

Bende Zoltán: Valamilyen felturbózott tetris játékot lehetne feladni feladatként.

Szatmáry Péter: Egyetértek a többiekkel, valami egyszerűbb játék: Space Impact, esetleg valami NES időkből származó felülnézetes 2D-s mászkálós kalandjáték (Final Fantasy) vagy platformer (Donkey Kong).

Králik Bálint Bendegúz: Létezik egy “Bomberman” nevezetű játék, ami mindig is a klasszikus játékok közül az egyik kedvencem volt. Szerintem annak a lefejlesztése nem igényel nagyobb munkát mint egy sokban. De bármilyen játék fejlesztése jó ötlet, hiszen az nem unalmas.

Máté László: mindenépp szórakoztató, játék jellegű feladatot. Irodai, banki alkalmazást unalmasabb lenne fejleszteni (szerintem).

Kapitány Gergely: mindenépp a több játékos jó ötlet. És továbbra is játékok legyenek mindenépp. Akár aktuális felkapott játékokhoz hasonló játék/játék mód, ez lehet jobban érdekli majd a csoportot, illetve látják hogyan lehet megprogramozni egyes részeit aktuális játékoknak, amelyekkel lehet ők is játszanak.

14.2.7. Egyéb kritika és javaslat

Bende Zoltán: Hadd ismételjem meg mégegyszer magam: **Túl sok diagramot kell csinálni.** Ennyi idő alatt a Doom-ot is lefejlesztettük volna.

Szatmáry Péter:



(a fejem amikor megláttam hány diagramot kell megírni)

A kommunikációs diagramok megrajzolását egy felesleges, redundáns feladatnak éreztem, szerintem semmit nem tettek hozzá érdemben a tervezéshez, másolás volt kicsit a külcsínt változtatva a szekvencia diagramok után.

Mivel egyikünknek sem volt még tapasztalata hasonló vagy nagyobb volumenű projekteken, sokszor nagyon el voltunk veszve, hogy mit kellene adott beadandó dokumentum adott feladatának pontjaiba írni. Sok esetben abszolút kevésnek tartottuk az egyes részfeladatokhoz tartozó 2-3 mondatos feladatleírást, így gyakran a vik.wiki-n található régebbi évek feladatainak dokumentációira kellett hagyatkoznunk, mint megoldási útmutatás, hogy az adott alfeladat mit is akar, hogy eleve milyen irányba induljunk el. Ez különösen igaznak éreztem a szkeleton és a prototípus tesztnyelvezetével kapcsolatos modulokra.

Králik Bálint Bendegúz: Egy “nem változtatási javaslat”, hogy tartsák meg azt a jó szokásukat, hogy valamilyen játék fejlesztését írják elő, mert ezek fejlesztése a diákoknak is úgy mond szórakoztató.



Ez egy jó macsek.

Máté László: Nem kimondottan ehhez a témahez, inkább mint “folyamathoz”: Szoftvertechnológia tárgyból modern alkalmazásokon mutassák be az anyag azon részeit, amit lehet/érdekes. Így egyrészt kézzelfoghatóbbnak tűnne a megszerzett tudás, másrészt jobb elképzelésekkel tudnának a hallgatók nekikezdeni a szoftver projekt labor tárgynak.

Kapitány Gergely: Szerintem ez a Sokoban jó feladat ötlet volt projektnek, hasonló feladatok biztos érdeklik majd a csapatokat a jövőben.

14.X. Hibajavítások, megjegyzések

- A grafikus változat a Hercules portálon leadott .zip állományába hibásan került bele a Timer.java, illetve a DrawTimer.java – ezért lettek feltüntetve 0 byte-os fájlonként a fájllistában.
- A leadandó programkód véglegesítésekor kimaradt a legutolsó a projekthez történt commit, a Game.java 15. sora módosult erről:
`public static void endGame() {`
 a következőre, így nem kezdi el a második szál is futtatni az endGame függvényt, ezzel másodjára is kiírva a búcsúzóüzenetet.
`public static synchronized void endGame() {`