Követelmények, projekt, funkcionalitás

/1. házi feladat/

1 – a_csapat

Konzulens:

Belákovics Ádám

Csapattagok:

Név
Simon Márta
Németh Marcell
Szövő Roland
Nagy Szabina
Majkut Kristóf

2018.02.18

2. Követelmény, projekt, funkcionalitás

Bevezetés

Cél

A projekt követelményeinek, alapvető felépítésének és funkcionalitásának ismertetése; ezek segítségével körbehatárolni a fejlesztés menetét és a végleges program felépítését, működését.

Szakterület

A kialakítandó szoftver egy számítógépes játék, így nem egy kifejezett szakterület részére készül, hanem általános felhasználásra, szórakoztatásra.

Definíciók, rövidítések

architekturális kép A szoftver belső felépítését szemléltető ábra.
BME Budapest Műszaki Egyetem rövidítése.
Eclipse Fejlesztőkörnyezet főként a Java programozási nyelvhez.
ernyő Képernyő rövidítése a dokumentációban.
eseménykezelő Az architektúra alrendszere, feladatai: a felhasználó által bevitt adat feldolgozása, információ továbbítása további alrendszerek felé.
Facebook Internetes közösségi oldal.
fejlesztőkörnyezet Olyan számítógépes program vagy programok összessége, ami lehetővé teszi vagy leegyszerűsíti egy fejlesztési munka végrehajtását.
funkció A program működésének egy külön megfogalmazható része.
Git Egy verziókezelő rendszer.
Git account A verziókezelő rendszer használatához szükséges felhasználói fiók.
GitHub A Git verziókezelő rendszerre épülő internetes szolgáltatás.
Google Docs Interaktív dokumentum szerkesztő rendszer, melynek segítségével a megosztásban résztvevő felhasználók mindegyike (akár egyidejűleg) képes hozzáférni a szerkesztőfelülethez.
GUI Graphical User Interface rövidítése. Felhasználó és számítógép közötti kommunikációt lehetővé tevő felület, mely részben vagy teljesen mértékben grafikus elemekből áll. Az architektúra egyik alrendszerét képezi.
HSZK Hallgatói Számítógép Központ rövidítése.
játékmotor Az architektúra egyik alrendszere. A játék tényleges irányítója, működtetője.
IntelliJ IDEA Fejlesztőkörnyezet főként a Java programozási nyelvhez.
JRE6 Java Runtime Environment 6 rövidítése. Ez egy olyan program, ami szükséges a Java programozási nyelvben írt programok használatához.
multi-player Többjátékos mód.
PC Personal Computer rövidítése, jelentése személyi számítógép.
proto/prototípus A program olyan állapota, amikor minden belső működés meg van valósítva és működik, de grafikus felület még nincsen hozzá.
szekvencia diagram Feladata az objektumok egymás közti üzenetváltásainak ábrázolása egy időtengely mentén.
single-player Egyjátékos mód.
szkeleton A program olyan állapota, amikor a program belső felépítése készen van, de nem csinál semmit.
szoftver Számítógépen futtatható program.
UML Unified Modeling Language rövidítése, egy rendszer modellező eszköz.

Use-Case Egy felhasználó és egy rendszer közötti, adott célt elérő interakció leírása.
verziókezelő Olyan számítógépes program, aminek segítségével eltárolható fájloknak régebbi verziói, így később vissza lehet térni egy adott verzióhoz, illetve nyomon lehet követni egy fájl változását.
WhiteStar UML UML, Use-Case diagrammok készítését lehetővé tevő szoftver.

Hivatkozások

Szoftver projekt labor - https://www.iit.bme.hu/targyak/BMEVIIIAB02/feladat

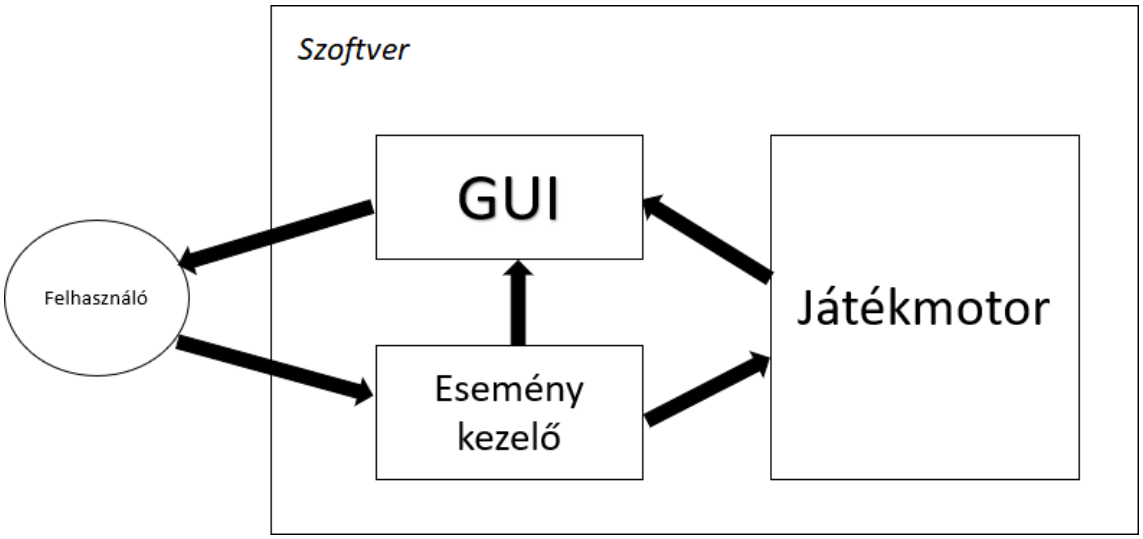
Összefoglalás

- 2.2 Áttekintés: A projekt terveinek, funkcióinak és korlátozásainak áttekintése, felhasználók lehetőségeinek meghatározása
- 2.3 Követelmények: A projekt során elvárt követelmények kidolgozása, külön részletezve a funkcionális, erőforrás-átadással kapcsolatos és egyéb nem funkcionális követelményeket.
- 2.4 Lényeges use-case-ek: A lényeges use-case-ek felsorolása és ábrázolása
- 2.5 Szótár: A projekt során bevezetett fogalmak részletesebb körülírása
- 2.6 Projekt terv: A projekt megvalósításával kapcsolatos feladatkörök és határidők részletes kifejtése
- 2.7 Napló: Az elvégzett feladatok és ráfordított idők felsorolása

Áttekintés

Általános áttekintés

A legmagasabb szintű architektúrális képe a játékszoftvernek:



Az architektúra legfontosabb alrendszere nyilvánvalóan a játékmotor, ő fogja „ismerni a játékszabályokat”, itt lesz maga a játéklogika megvalósítva. A játékmotor figyeli, hogy egy adott lépés szabályos-e, tovább lehet-e haladni abba az irányba, odébb lehet-e tolni azt a ládát, vagy éppen mi történik a dolgozóval, ha egy lyukra lép. Ő fogja kezelni a teljes pályát, követi a játékosok lépéseit, figyeli az esetleges játék végét.

A felhasználói inputokat az eseménykezelő alrendszer fogja feldolgozni, és azoknak megfelelően értesíti a másik két alrendszer egyikét, hogy milyen események mentek végbe. A fő hangsúly ennél a szoftvernél a billentyűzet bementeken van.

A GUI, azaz a grafikus felhasználói interfész mutatja a játékosnak, hogy mi is történik a pályán valójában. A képernyő frissítésére vonatkozó információk jöhetnek közvetlenül az eseménykezelő alrendszertől, vagy a játékmotortól, amely a felhasználói inputoknak megfelelő változtatásokat hajtatja végre a GUI-val.

Hálózati kapcsolatra egyáltalán nincs szükség a játék használatához, illetve adattárolással sem kell foglalkozni, hiszen esetünkben a toplistának nem sok értelme van, a pálya pedig rögzített felépítésű.

Funkciók

A Sokoban nevű játékban eredetileg különböző pályákon, adott elrendezésű falak között kell dobozokat tologatni úgy, hogy azok mindegyikét az kijelölt helyekre eljuttassuk. Két egymás mellett lévő doboz nem tolható el egyszerre. Ügyelni kell, hogy a dobozok ne „ragadjanak be”, azaz ne kerüljenek olyan helyzetbe, ahonnan már ki nem mozdíthatóak, hiszen egy pálya teljesítéséhez az összes dobozt a helyére kell vinni. Ez alapvetően egy *single-player* játék, amelynek egy kicsit megcsavart, *multi-player* verziója a Killer Sokoban.

A játék ezúttal egy téglalap alakú *raktárépület*ben zajlik, ahol *láda*kat tárolnak. A raktárépület padlója négyzetekre van osztva, ezeken állnak a négyzetekkel megegyező alapterületű ládák. A ládák eltolhatóak, de egy *mozgatás*sal mindig csak egy szomszédos négyzetre kerülhetnek.

A raktárban ketten dolgoznak (dolgozók), a két játékos őket tudja irányítani. A játékosok célja a pálya két ellentétes sarkában lévő, számukra kijelölt *nyelő*khöz eljuttatni a ládákat. Amennyiben az egyik játékos a másik nyelőjéhez tol egy ládát, az a másik játékos *pont*jának számít. A két *dolgozó* a számukra kijelölt nyelőkről indul a játék kezdetekor. A raktárnak *fal*ai és *oszlop*ai is vannak, ezeken a ládák nem tolhatók át, illetve (értelemszerűen) a dolgozók sem tudnak áthaladni rajtuk. A ládák egymást el tudják tolni, így egy dolgozó egyszerre több ládát is mozgathat. Fontos azonban, hogy a dolgozók egymást közvetlenül nem tolhatják el, valamint egymással azonos mezőre sem léphetnek. Ellentétben, ha egy dolgozóra ládát tolunk, akkor az automatikusan a szomszédos négyzetre tolódik. Amennyiben a dolgozó már nem tolható a mellette lévő mezőre (mert pl. fal van mellette), a dolgozó meghal. A ládák nem nyomhatók össze, ha fal mellett állnak, abba az irányba tolva őket nem történik semmi.

A padlón egyes helyeken *lyuk*ak találhatók, amelyekre ládát tolva a láda leesik (eltűnik). Ha munkás lép rá, meghal. Némelyik lyuk csak akkor viselkedik lyukként, ha egy *kapcsolón* láda áll, egyébként padlónak tűnik. A kapcsolón ládának kell állnia, hogy kinyissa a lyukat, ha munkás áll a kapcsolóra, akkor nem kapcsol.

A raktárépület felépítése fix, azaz a játék egy pályát tartalmaz, ahol adott rendben helyezkednek el a különböző falak, oszlopok, ládák, lyukak, illetve kapcsolók. Ebben mérkőzhet meg egymással a két dolgozó.

Ha az egyik játékos meghal (lezuhan vagy összenyomják egy ládával), akkor a játék véget ér és az életben maradt játékos nyer. Egy másik esetben akkor lehet vége, ha már nem lehet több ládát tolni. Ez előfordulhat, ha már csak sarkokban vannak ládák, ahonnan nem mozdíthatók ki, vagy

ha a játékosok az összes mozdítható ládát eljuttatták a nyelőkhöz. Ekkor a játékot az nyeri, aki a legtöbb ládát vitte be a számára kijelölt helyre.

Felhasználók

A felhasználók egy-egy munkást irányítanak, így tudják a ládákat tologatni.
A szoftver felhasználói számára nincs szükség különösebb előképzettségre, a használat rövid időn belül elsajátítható, és a játék élvezhető bárki számára, aki a szótárban felsorolt definíciókat képes értelmezni.

Korlátozások

A szoftverre vonatkozó előírás és korlátozás is egyben egyedül az, hogy a standard Java könyvtár készletet kell használnia, és semmilyen más csomagot nem vehet igénybe.

Feltételezések, kapcsolatok

Követelmények

Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
R01	A pálya téglalap alakú.	bemutatás	opcionális	csapat		
R02	A pálya négyzet alakú *mező*kre van felosztva.	bemutatás	alapvető	megrendelő		
R03	Egy-egy mezőn állhat egy dolgozó vagy egy láda, továbbá lehet rajta egy lyuk vagy egy kapcsoló.	kiértékelés	alapvető	megrendelő		
R04	A ládák egy mozgatással csak velük szomszédos mezőre tolhatók.	bemutatás	alapvető	megrendelő	Move Chest	

R05	A raktárban két dolgozó van.	bemutatás	alapvető	csapat		Mindegyiket másik játékos irányítja.
R06	A játékosok egy-egy dolgozót irányítanak.	bemutatás	alapvető	megrendelő	Move Worker	
R07	A cél eljuttatni a ládákat az elfogadóhelyekre (nyelők).	bemutatás	alapvető	megrendelő	Move Chest	
R08	A két nyelő a pálya két ellentétes sarkában van.	bemutatás	fontos	csapat		
R09	Mindkét játékosnak ki van jelölve a saját nyelője, az oda eljuttatott ládák csak a kijelölt játékosnak számítanak pontként.	kiértékelés	alapvető	csapat		
R10	A dolgozók a nekik kijelölt nyelőkről indulnak.	kiértékelés	fontos	csapat	Move Worker	
R11	A raktárnak falai és oszlopai vannak, amelyeken nem lehet átmenni.	bemutatás	alapvető	megrendelő	Move Worker, Move Chest	

R12	Egy dolgozó egyszerre több ládát is eltolhat.	bemutatás	fontos	megrendelő	Move Chest	
R13	A dolgozók egymást nem tudják eltolni.	bemutatás	alapvető	csapat	Move Worker	
R14	Egy mezőn egyszerre csak egy dolgozó állhat.	bemutatás	alapvető	csapat	Move Worker	
R15	Ha egy dolgozóra ládát tolunk, az automatikusan a következő mezőre tolódik.	bemutatás	alapvető	megrendelő	Move Chest	
R16	Ha a dolgozó nem tolható a mellette lévő mezőre, amikor rátolnak egy ládát, meghal.	bemutatás	alapvető	megrendelő	Move Chest	
R17	A ládák nem nyomhatók össze, ha nem tolhatók tovább, nem történik velük semmi.	bemutatás	alapvető	megrendelő	Move Chest	
R18	Ha egy lyukra ládát tolunk, akkor az eltűnik.	bemutatás	alapvető	megrendelő	Move Chest	

R19	Ha dolgozó lép a lyukra, meghal.	bemutatás	alapvető	megrendelő	Move Worker	
R20	Van olyan lyuk, amely csak akkor viselkedik lyukként, ha egy kapcsoló aktiválva van.	bemutatás	alapvető	megrendelő	Activate Switch, Open Hole	
R21	Egy kapcsoló akkor aktív, ha egy láda áll rajta, dolgozó hiába áll rá, nem történik semmi.	bemutatás	alapvető	megrendelő	Move Chest, Activate Switch	
R22	A játék egy fix felépítésű (falak, oszlopok, ládák, lyukak és kapcsolók elhelyezkedése tekintetében) pályát tartalmaz.	bemutatás	fontos	csapat		
R23	Ha egy játékos meghal, vége a játéknak, és a másik győzött.	bemutatás	alapvető	csapat		
R24	Ha úgy ér véget a játék, hogy már nem lehet többet tolni, akkor az nyer, aki a legtöbb ládát vitte el a saját nyelőjéhez.	bemutatás	alapvető	megrendelő	Move Chest	

R25	Egy láda akkor nem mozdítható, ha legalább két, nem szemközti oldalról is fal/oszlop határolja.	bemutatás	alapvető	csapat	Move Chest	
R26	A dolgozók csak velük közvetlenül szomszédos irányokban (jobbra, balra, fel, le) mozoghatnak.	bemutatás	alapvető	megrendelő	Move Worker	
R27	A játékot el lehet indítani.	bemutatás	alapvető	csapat	Start Game	

Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
R1	Git	nincs	alapvető	csapat	verziókezelő
R2	Git account	nincs	alapvető	csapat	tárhely
R3	JRE6	bemutatás	alapvető	megrendelő	
R4	Eclipse	nincs	opcionális	csapat	Java IDE
R5	IntelliJ IDEA	nincs	opcionális	csapat	Java IDE
R6	HSZK-ban találhatókkal azonos vagy jobb teljesítményű PC	bemutatás	alapvető	megrendelő	

R7	Monitor	nincs	alapvető	csapat	
R8	Egér	nincs	alapvető	csapat	
R9	Billentyűzet	nincs	alapvető	csapat	
R10	White Star UML	nincs	opcionális	csapat	

Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
3.01	Szkeleton átadás	bemutatás	alapvető	megrendelő	március 12.
3.02	Proto átadás	bemutatás	alapvető	megrendelő	április 9.
3.03	Teljes program átadása	bemutatás	alapvető	megrendelő	május 1.
3.04	Útmutató alapján telepíthető, külső segítség nélkül	bemutatás	fontos	megrendelő	

Egyéb nem funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
NFR01	Problémamentes tesztelés	tesztelés	fontos	specifikáció	
NFR02	Hibamentes futás	tesztelés	fontos	specifikáció	
NFR03	Felhasználó	futtatás	alapvető		ismerje a játék irányításához szükséges billentyűket

Lényeges use-case-ek

Use-case leírások

Cím	Dolgozót mozgat (Move Worker)
Leírás	A játékos egy dolgozót irányít a pályán.
Aktorok	Játékos
Főforgatókönyv	1. A választott irányban (jobb, bal, fel, le) a szomszédos mező üres. 2. A dolgozó a választott irányban a szomszédos mezőre mozog.
Alternatív forgatókönyv	A választott irányban a dolgozó egy falba/oszlopba ütközik. A dolgozó nem tud a választott irányba továbblépni, egy helyben marad.
	A választott irányban a másik játékos dolgozója áll a mezőn. Abban az irányban nem tud továbbhaladni a dolgozó.
	A választott irányba továbblép egy „lyuk” mezőre. A dolgozó meghal.

Cím	Ládát eltol (Move Chest)
Leírás	A dolgozó odébb tolja az útjába kerülő ládát.
Aktorok	Játékos
Főforgatókönyv	1. A választott irányban (jobb, bal, fel, le) a ládával szomszédos mező üres. 2. A dolgozó a haladási irányában a ládát a vele szomszédos mezőre továbbtolja.

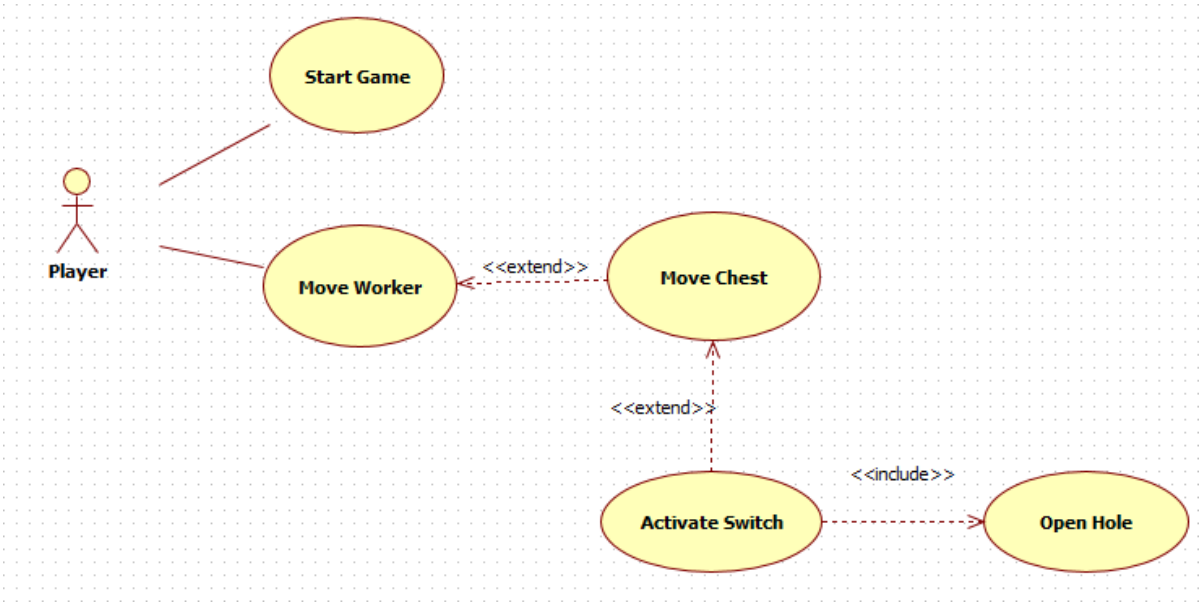
Alternatív forgatókönyv	A választott irányban egy fal/oszlop áll. A ládát ekkor nem lehet elmozdítani.
	1.B.1 A választott irányban egy dolgozó van. 1.B.2.A.1 A dolgozó melletti mező üres a választott irányban, ekkor a ládát és a dolgozót is odébb toljuk a velük szomszédos mezőkre. 1.B.2.B.1 A dolgozó melletti mező egy fal/oszlop, a láda ugyan odébb csúszik, de a dolgozó, akire rátolták, meghal.

Cím	Kapcsoló aktiválása (Activate Switch)
Leírás	A dolgozó aktiválja a kapcsolót.
Aktorok	Játékos
Főforgatókönyv	1. A dolgozó rátol egy ládát a kapcsolóra. 2. A kapcsoló aktiválódik.
Alternatív forgatókönyv	1.A.1 A kapcsolón egy dolgozó áll. 1.A.2.A.1 A dolgozó melletti mező üres a választott irányban, ekkor a ládát és a dolgozót is odébb toljuk a velük szomszédos mezőkre. 1.A.2.B.1 A dolgozó melletti mező egy fal/oszlop/láda, a láda ugyan odébb csúszik, de a dolgozó, akire rátolták, meghal.

Cím	Lyuk kinyitása/mutatása (Open Hole)
Leírás	A kapcsoló kinyitja a megfelelő lyukat.
Aktorok	-
Főforgatókönyv	A kapcsoló aktiválásakor a megfelelő lyuk kinyílik.

Cím	Játék indítása (Start Game)
Leírás	A játék indítása.
Aktorok	Játékos
Főforgatókönyv	A játékos elindítja a játékot.

Use-case diagram



Szótár

dolgozó A játékos által irányított karakter.
elfogadóhely Olyan mező, amelyre ha a dolgozó ládát tol, akkor pontot szerez.
fal A pályát határoló mező, amelyekre nem lehet lépni, sem ládát tolni.
halál Olyan esemény, amely során egy dolgozó lekerül a pályáról és a játékosa elveszíti a játékot. Akkor következhet be, ha egy dolgozóra ládát tolnak és nem tud hova ellépni előle, vagy lyukra lép.
kapcsoló Olyan elem, amelyre ha ládát tol a dolgozó, akkor egy mező lyukként kezd viselkedni, ameddig a láda rajta marad.
láda Olyan elem, amely célba juttatásával nyerhető meg a játék.
lyuk Olyan elem, amelyre ha ládát tol a dolgozó, akkor a láda lekerül a pályáról, vagy ha játékos lép rá akkor meghal.
mező A pályát alkotó egységnyi területek, falak és oszlopok, illetve amelyeken állhat játékos vagy láda, emellett lehetnek lyukak vagy elfogadóhelyek.

mozgatás Olyan esemény, amely alatt a dolgozó egy ládát egy szomszédos mezőre helyez át.
nyelő Elfogadóhely szinonímája.
nyerés Az az esemény, amely során a játékosok pontjai összehasonlításra kerülnek, és a több pontot elért játékost hirdetik győztesnek.
oszlop Olyan mező a pályán belül, amelyre nem lehet lépni, sem ládát tolni.
pálya Az a grafikus terület, ahol a játékmenet zajlik, a raktár maga.

Projekt terv

Csapat

A csapat 5 főből áll. A feladatokat mindenki saját magának jelöli ki, lehetőleg úgy, hogy mindenkinek azonos mennyiségű és nehézségű feladata legyen.

Név	Felelősségek
Majkut Kristóf (csapatvezető)	Menedzsment, UML, kód
Nagy Szabina	Dokumentáció, kód
Németh Marcell	Kód, dokumentáció, UML
Simon Márta	Kód, dokumentáció, UML
Szövő Roland	Kód, dokumentáció

Kommunikáció

Verziókezelés: Mivel többen dolgozunk a projekten, ezért fontos, hogy mindig mindenkinek elérhető legyen a legfrissebb forráskód, illetve dokumentáció. Ezért valamilyen módon meg kell osztanunk egymással az elkészített dokumentumokat, programkódokat.
Erre a Git elosztott verziókezelő szoftvert választottuk. Ehhez a központi tárhelyet a GitHub biztosítja. Azért esett erre a megoldásra a választás, mert így könnyen tudjuk követni ki mit csinált, egyszerűen tudunk visszaállni egy korábbi verzióra és közösen meg tudjuk vitatni, hogy mely változások kerüljenek be a végleges projektbe.

Facebook: A csapatnak létrehoztunk egy privát facebook beszélgetést, ahova mindenki írhat. Itt beszéljük meg az esetleges felmerülő kérdéseket.

Használt programok

Verziókezelés: a verziókezeléshez a feljebb bővebben kifejtett Git-et használjuk.

Dokumentáció: a dokumentációhoz Google Docs-ot használunk. Így mindenki el tudja érni és szerkeszteni a dokumentumot. Az UML diagramok készítéséhez a WhiteStarUML nevű szoftvert használtuk.

Fejlesztőkörnyezet: a forráskódot Eclipse, illetve IntelliJ IDEA nevű fejlesztőkörnyezetekben készítjük.

Mérföldkövek, határidők

Dátum	Leírás	Ellenőrzés
febr. 19.	Követelmény, projekt, funkcionalitás	beadás
febr. 26	Analízis modell kidolgozása 1.	beadás
márc. 5.	Analízis modell kidolgozása 2.	beadás
márc. 12.	Szkeleton tervezése	beadás
márc. 19.	**Szkeleton**	beadás és a forráskód herculesre való feltöltése
márc. 26.	Prototípus koncepciója	beadás
ápr. 9.	Részletes tervek	beadás
ápr. 16.	Prototípus készítése, tesztelése	nem kell anyagot beadni
ápr. 23.	**Prototípus**	beadás és a forráskód, a tesztbemenetek és az elvárt kimenetelek herculesre való feltöltése
máj. 2.	Grafikus felület specifikációja	beadás a laboralkalommal
máj. 7.	Grafikus változat készítése	nem kell anyagot beadni
máj. 14.	**Grafikus változat**	beadás és a forráskód herculesre való feltöltése

máj. 18.	**Összefoglalás**	beadás és feltöltés

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 kiírja 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 eljárni. A szkeletonnak alkalmasnak kell lenni arra, hogy a különböző forgatókönyvek és szekvencia diagramok ellenőrizhetők legyenek. Csak karakteres ernyőkezelés 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 interfészt. A változat tervezési szempontból elkészült, az ütemezés, az aktív objektumok kezelése megoldott. A business objektumok - a megjelenítésre vonatkozó részeket kivéve - valamennyi metódusa a végleges algoritmusokat tartalmazza. A megjelenítés és működtetés egy alfanumerikus ernyőn 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** (teljes) változat a prototípustól elvileg csak a kezelői felület minőségében különbözhet. Ennek változatnak az értékelésekor a hangsúlyt sokkal inkább a megvalósítás belső szerkezetére, semmint a külalakra kell helyezni.

Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.02.13.	2 óra	Szövő	Bevezetés elkészítése (2.1.1, 2.1.2, 2.1.4 2.1.5)
2018.02.16.	4 óra	Majkut	Áttekintés elkezdése (2.2.1, 2.2.2) Követelménynek (2.3.1) Lényeges use-case-ek (2.4)
2018.02.17.	3óra	Szövő	Áttekintés befejezése(2.2.3, 2.2.4, 2.2.5, 2.3.3) Projekt terv elkészítése (2.6)
2018.02.17	3 óra	Németh	Erőforrások összeírása (2.3.2) Szótár elkészítése (2.5)

2018.02.17	4 óra	Simon	Dokumentáció átnézése, ellenőrzése, Lényeges use-case-ek (2.4), Funkcionális követelmények (2.3.1)
2018.02.17	3 óra	Nagy	Definíciók, rövidítések (2.1.3); Egyéb, nem funkcionális követelmények (2.3.4)

Analízis modell kidolgozása II.

/3. házi feladat/

1 – a_csapat

Konzulens:

Belákovics Ádám

Csapattagok:

Név
Simon Márta
Németh Marcell
Szövő Roland
Nagy Szabina

Majkut Kristóf	UDAXJO	kristof.majkut@gmail.com

2018.03.04.

4. Analízis modell kidolgozása

1. Objektum katalógus

1. Storehouse

Ez a játék logikáját magába foglaló osztály. Számon tartja a pályán lévő mozdítható elemeket és mezőket.

2. Chest

A Chest egy ládát megvalósító mozdítható pályaelem. Ellenőrzi, hogy a láda mozdítható-e, tehát nincs-e beragadva. Egy Chest objektum mozdítható/eltolható egy Worker objektum által. Ha Hole mezőre kerül az objektum, akkor lekerül a pályáról.

3. Worker

A Worker objektum a felhasználó által irányított dolgozót valósítja meg, mint mozdítható pályaelemet. Képes a Chest-ek eltolására, valamint ha Hole-ra lép vagy egy Chest-tel nekinyomják egy Wall vagy egy másik Chest objektumnak, akkor meghal, vagyis lekerül a pályáról.

4. Field

Üres mező. Ellenőrzi, hogy amelyik mozdítható pályaelem rá akar kerülni/lépni, az megteheti-e. Saját maga kezeli az Elementek ráhelyeződését.

5. Wall

Pálya szélét határoló, valamint falként (vagy oszlopként) is funkcionáló mező. Nem lehet rálépni, se mozdítható pályaelemet rátolni. A Field osztály leszármazottja, emiatt magnak ellenőrzi az esetleges rálépést/rátolást.

6. Hole

Lyukként funkcionáló mező. Ha rálép egy Worker vagy rátolnak egy Chest-et, akkor azok objektumpéldánya “leesik” a pályáról, azaz kikerül a kollekcióból, amiben a pályán lévő elemek nyilván vannak tartva. Kapcsolóval (Switch) nyitható és csukható.

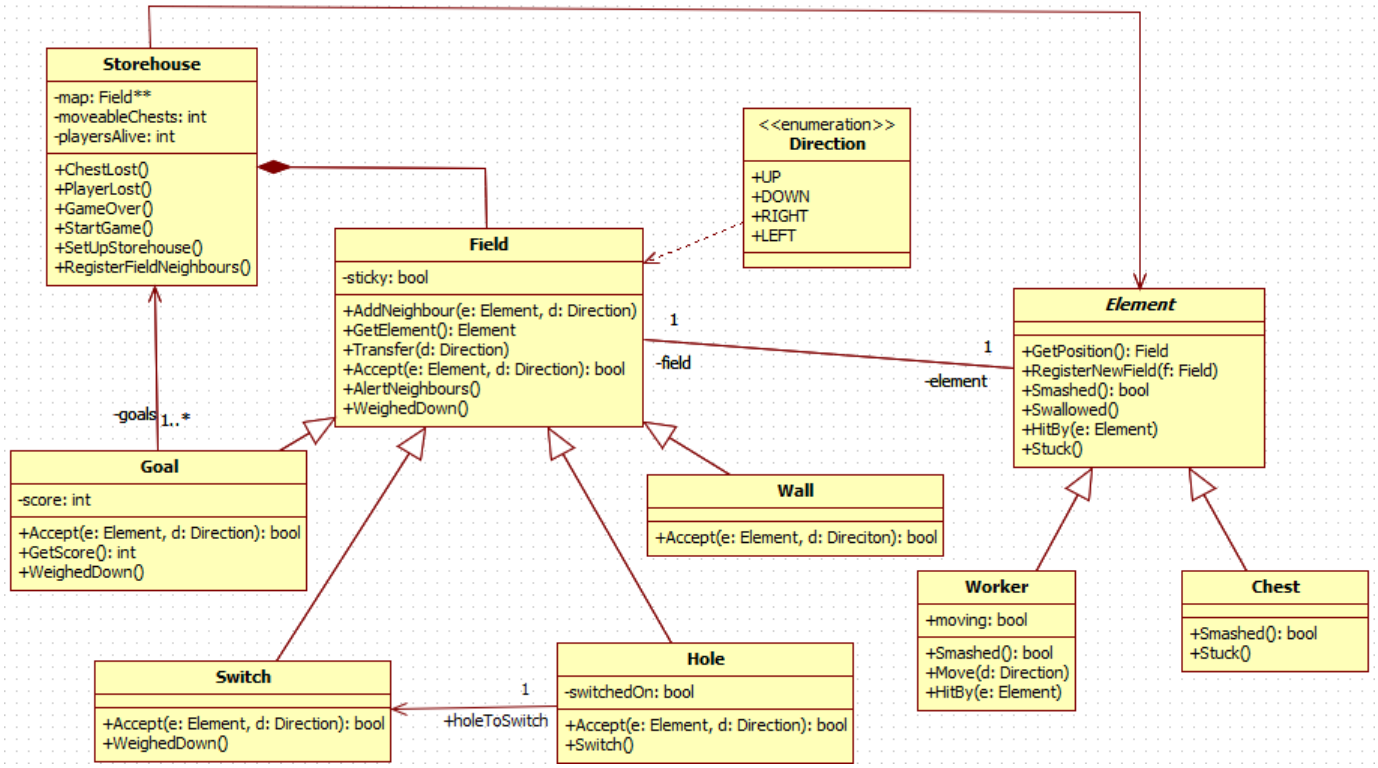
7. Switch

Kapcsolóként funkcionáló mező. Ha egy Worker rátol egy Chest-et, akkor kinyitja a hozzá tartozó Hole-t. Ha Worker lép rá, akkor nem fog történni semmi.

8. Goal

Célmezőként, nyelőként funkcionáló mező. Ha egy Worker a számára kijelölt célmezőre tol egy Chest-et, akkor a pontszáma megnő eggyel, és a célmezőre tolt Chest lekerül a pályáról. Amennyiben a Worker az ellenfél Worker célmezőjére tolja a Chest-et, akkor azzal az ellenfélnek szerez plusz egy pontot.

2. *Statikus struktúra diagramok*



3.1 ábra. Osztálydiagram

3. *Osztályok leírása*

1. Chest

- **Felelősség**
  Ládaként funkcionáló mozdítható pályaelemet megvalósító osztály. A Worker-ek el tudják tolni.
- **Ősosztályok**
  Element -> Chest
- **Metódusok**
  - **bool Smashed():** False-szal tér vissza, hiszen a láda nem nyomható össze.
  - **void Stuck():** Szól a Storehouse-nak, hogy egy újabb láda beragadt.

2. Direction

- **Felelősség**
  Az irányokat nyilvántartó enumeráció
- **Attribútumok**
  - **UP:** Felfelé irány.
  - **DOWN:** Lefelé irány.
  - **RIGHT:** Jobbra irány.

- **LEFT:** Balra irány.

3. Element

- **Felelősség**
  Mozdítható pályaelemek absztrakt ősosztálya.
- **Attribútumok**
  - **Field field:** Ebben tárolja, hogy épp melyik mezőn áll a pályaelem.
- **Metódusok**
  - **bool Smashed():** Amennyiben az adott pályaelemet nem tudta továbbtenni a mező, a rajta álló pályaelemnek ezt a metódusát hívja, ahol megnézi, hogy összenyomható-e az adott Element
  - **bool Swallowed():** Akkor hívjuk meg, ha az elem lyukra vagy nyelőre kerül.
  - **void RegisterNewField(Field f):** Ha az adott pályaelem egy új mezőre kerül, a mező ezzel tudja beregisztrálni magát az Elementbe.
  - **Field GetPosition():** Visszatér azzal a mezővel, amin áll.
  - **void HitBy(Element e):** Összeütköztet magával egy másik pályaelemet. Alapesetben, azaz ha az ütköztető pályaelem nem egy Worker, egyszerűen felteszi a saját mezőjére az átvett Element-et.
  - **void Stuck():** Figyelmezteti a pályaelemet, hogy beragadós mezőre lépett.

4. Field

- **Felelősség**
  A pályát alkotó mezők ősosztálya.
- **Attribútumok**
  - **bool sticky:** Jelzi, hogy a mezőre tolva egy ládát onnan elmozdítható lesz-e még.
  - **Element element:** Az éppen a mezőn álló Element.
- **Metódusok**
  - **bool Accept(Element e, Direction d):** Megállapítja, hogy az adott Element ráléphet-e.
  - **void Transfer(Direction d):** A mezőn álló Element-et elmozdítja az adott irányba.
  - **void AlertNeighbours():** Ha egy ládát tolnak a mezőre, és a sticky attribútuma igaz, értesíti a körülötte lévőket, hogy kvázi már ő is falnak számít.
  - **void AddNeighbour(Field f, Direction d):**
  - **Element GetElement():** Visszatér a rajta álló pályaelemmel.
  - **void WeighedDown():** A ládával történő kézfogáskor hívódó függvény.

5. Goal

- **Felelősség**
  Célmezőként, nyelőként funkcionáló mezőt megvalósító osztály. Ha egy Worker rátol egy Chest-et, akkor a pontszáma megnő eggyel, és a Chest lekerül a pályáról.
- **Ősosztályok**
  Field -> Goal
- **Attribútumok**
  - **int score:** Azon játékos pontszáma, akinek a célja ebbe a nyelőbe eljuttatni a dobozokat
- **Metódusok**
  - **bool Accept(Element e, Direction d):** Megállapítja, hogy az adott Element ráléphet-e.
  - **int GetScore():** A rátolt ládák számával, azaz az adott játékos pontjával tér vissza.

- **void WeighedDown():** Ekkora tudja a nyelő, hogy láda lépett rá, így az adott játékos pontszámát növeli eggyel, illetve elnyeli az adott ládát.

6. Hole

- **Felelősség**
  Lyukként funkcionáló mezőt megvalósító osztály. Ha rálép egy Element, akkor az le fog esni a pályáról. Kapcsolóval nyitható és csukható.
- **Ősosztályok**
  Field -> Hole
- **Attribútumok**
  - **bool switchedOn:** Tárolja, hogy a hozzá tartozó kapcsoló fel van-e kapcsolva.
- **Metódusok**
  - **bool Accept(Element e, Direction d):** Minden pályaelemet elfogad,tehát true-val tér vissza és elnyeli őket, azaz nem tárolja le őket.
  - **void Switch():** A switchedOn attribútum értékét negálja.

7. Storehouse

- **Felelősség**
  Nyilvántartja a mezőket, a mozdítható ládák és az aktuális játékosok számát. A játékbeli események vezérléséért felelős osztály.
- **Attribútumok**
  - **Field[][] map:** A mezőket tároló két dimenziós Field tömb.
  - **int moveableChests:** Tárolja a mozdítható ládák számát.
  - **int playersAlive:** Tárolja, hogy hány játékos van a játékban.
  - **Goal[] goals:** A kihelyezett nyelők tömbje.
- **Metódusok**
  - **void ChestLost():** Csökkenti a mozdítható ládák számát.
  - **void PlayerLost():** Csökkenti a játékban lévő játékosok számát. Ha eléri ez az 1-et akkor meghívja a GameOver() függvényt.
  - **void GameOver():** Befejezi a játékot és kiírja a győztest.
  - **void StartGame():** Elindítja a játékot, felépíti a pályát.
  - **void SetUpStoreHouse():** Elhelyezi a pályán a mezőket a megfelelő elrendezésben.
  - **void RegisterFieldNeighbours():** A pályán való elhelyezkedés alapján beregisztrálja az összes mezőhöz annak szomszédait.

8. Switch

- **Felelősség**
  Kapcsolóként funkcionáló mezőt megvalósító osztály. Ha rátolunk egy Chest-et, akkor kinyitja a hozzá tartozó Hole-t. Ha Worker lép rá, akkor nem történik semmi.
- **Ősosztályok**
  Field -> Switch
- **Attribútumok**
  - **Hole holeToSwitch:** Az általa kapcsolt lyuk.
- **Metódusok**
  - **bool Accept(Element e, Direction d):** Megállapítja, hogy az adott Element ráléphet-e.

- **void WeighedDown():** Ezt a függvényt csak láda hívja, emiatt úgy definiáljuk felül, hogy ez meghívja a hozzá tartozó lyuk Switch() függvényét.

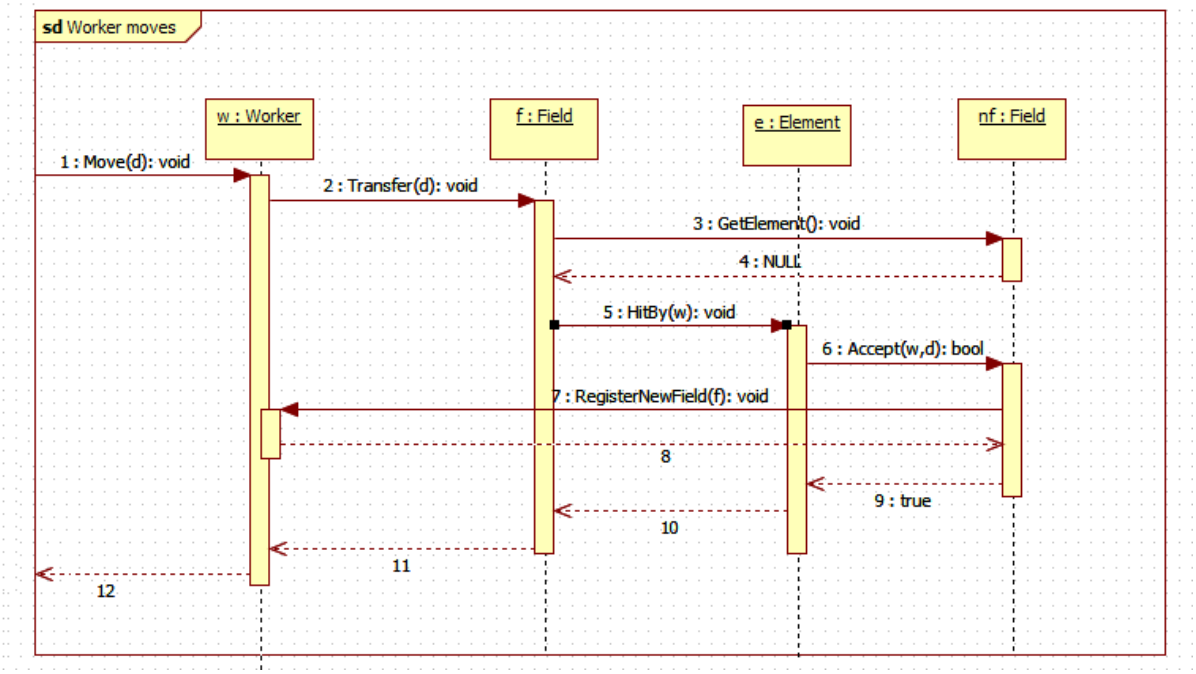
9. Wall

- **Felelősség**
  Pálya szélét határoló, valamint falként (oszlopként) funkcionáló mezőt megvalósító osztály. Nem léphet rá semmilyen Element.
- **Ősosztályok**
  Field -> Wall
- **Metódusok**
- **bool Accept(Element e, Direction d):** Mindig false-szal tér vissza, mást nem csinál.

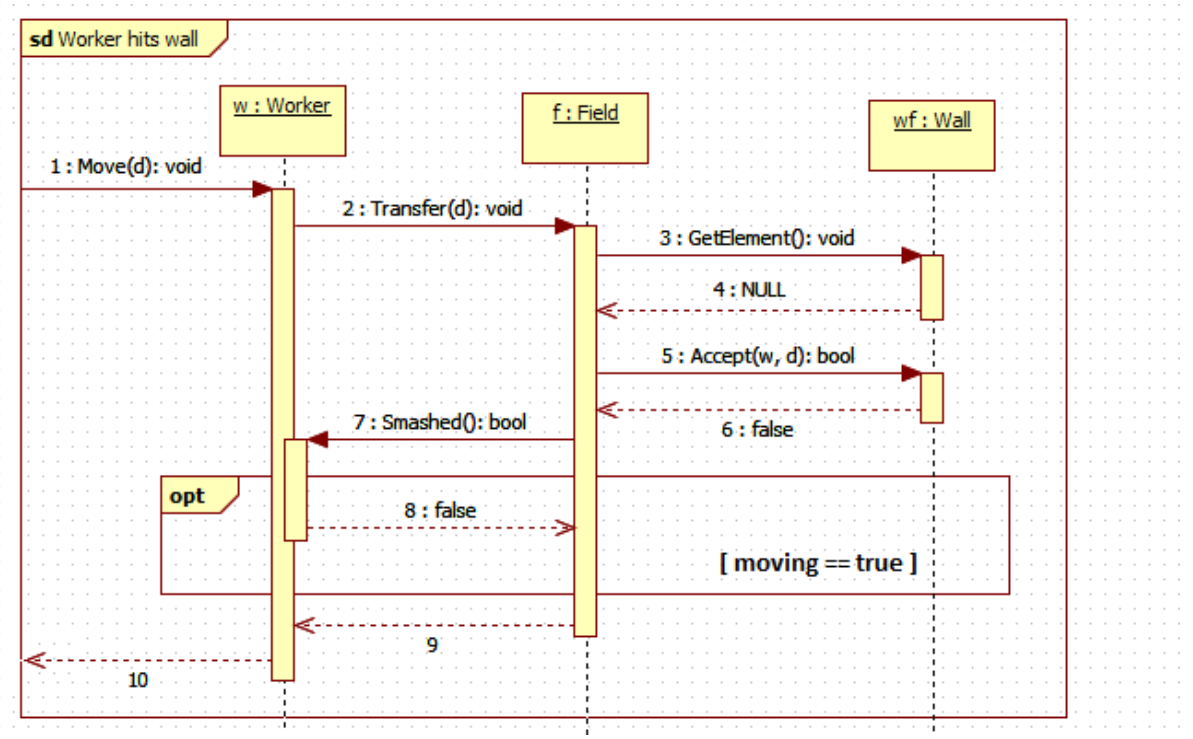
10. Worker

- **Felelősség**
  Dolgozót, mint mozdítható pályaelemet megvalósító osztály. Képes a Chest-ek eltolására.
- **Ősosztályok**
  Element -> Worker
- **Attribútumok**
  - **bool moving:** Jelzi, hogy éppen a dolgozó lép-e, vagy pedig csak tolják őt
- **Metódusok**
  - **bool Smashed():** Annak érdekében, hogy ne haljon meg a dolgozó amikor csak simán falnak megy, figyeli a moving értékét, és amennyiben false, azaz a dolgozó éppen megtolták, igazzal tér vissza, mivel a dolgozó összenyomható. Végül meghívja a Storehouse PlayerLost() függvényét.
  - **void Move(Direction d):** Külső input esetén ez a függvény hívodik meg a dolgozó mozgatására.
  - **void HitBy(Element e):** Amennyiben meghívódik, semmit nem csinál, hiszen ezt csak egy másik Worker Transfer(Direction d) függvénye hívhatta, ezzel érhető el, hogy a két munkás ne tudja eltolni egymást.

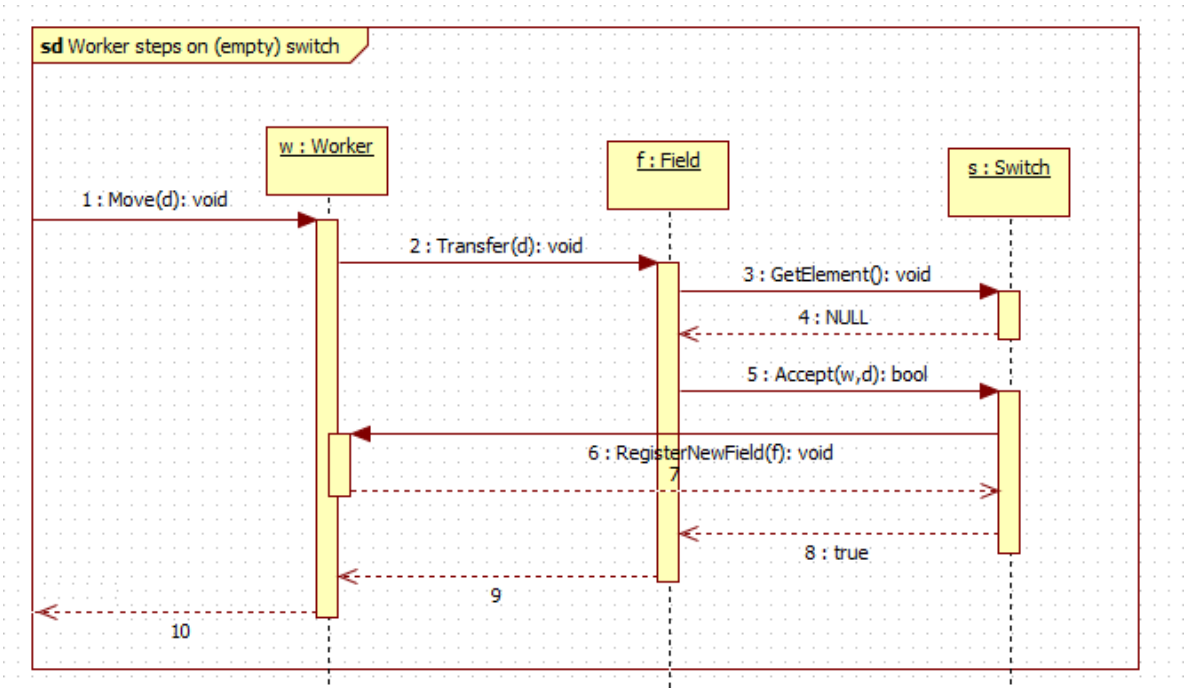
4. Szekvencia diagramok



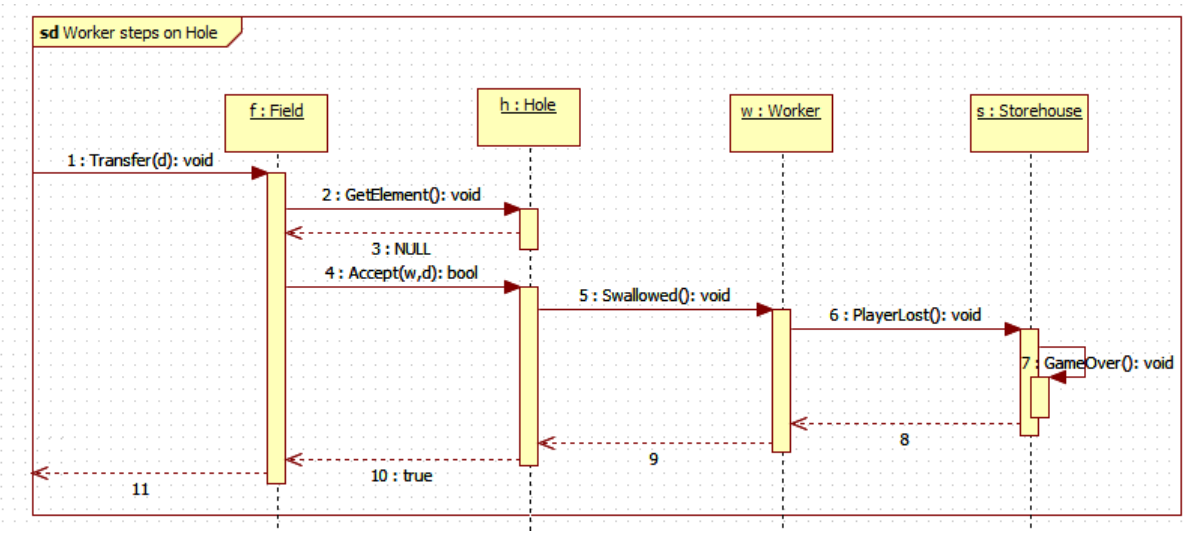
4.4.1. A dolgozó lép



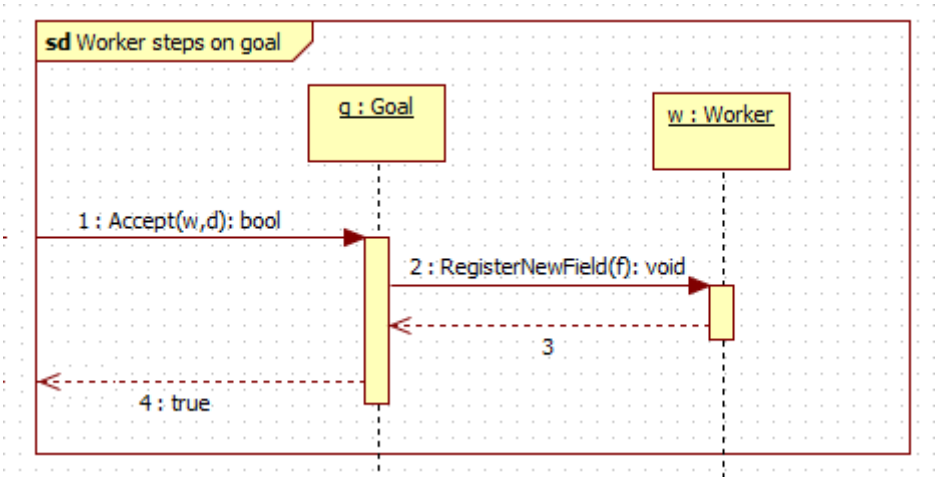
A dolgozó falnak ütközik



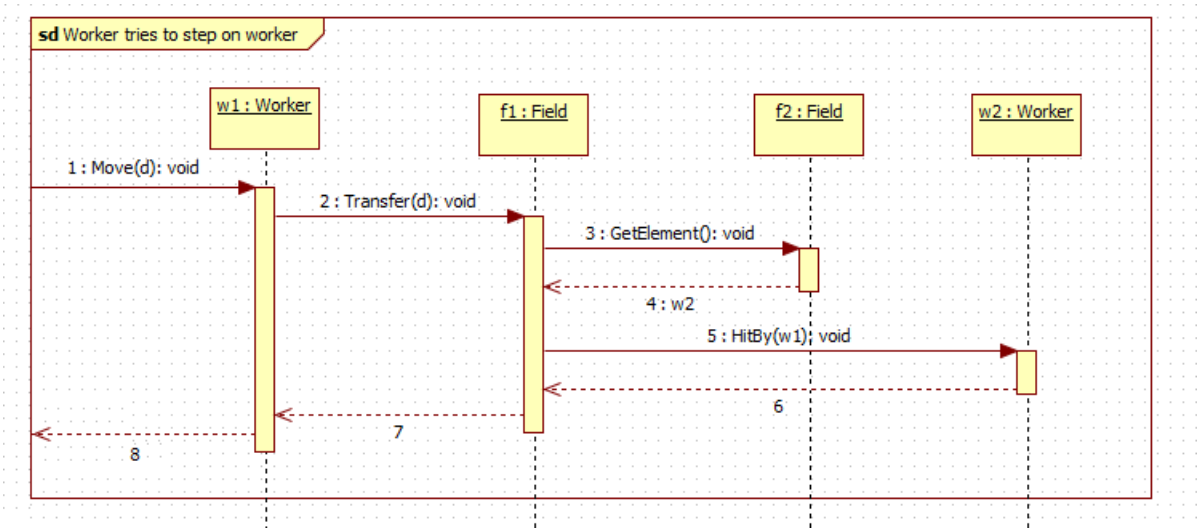
A dolgozó kapcsolóra lép



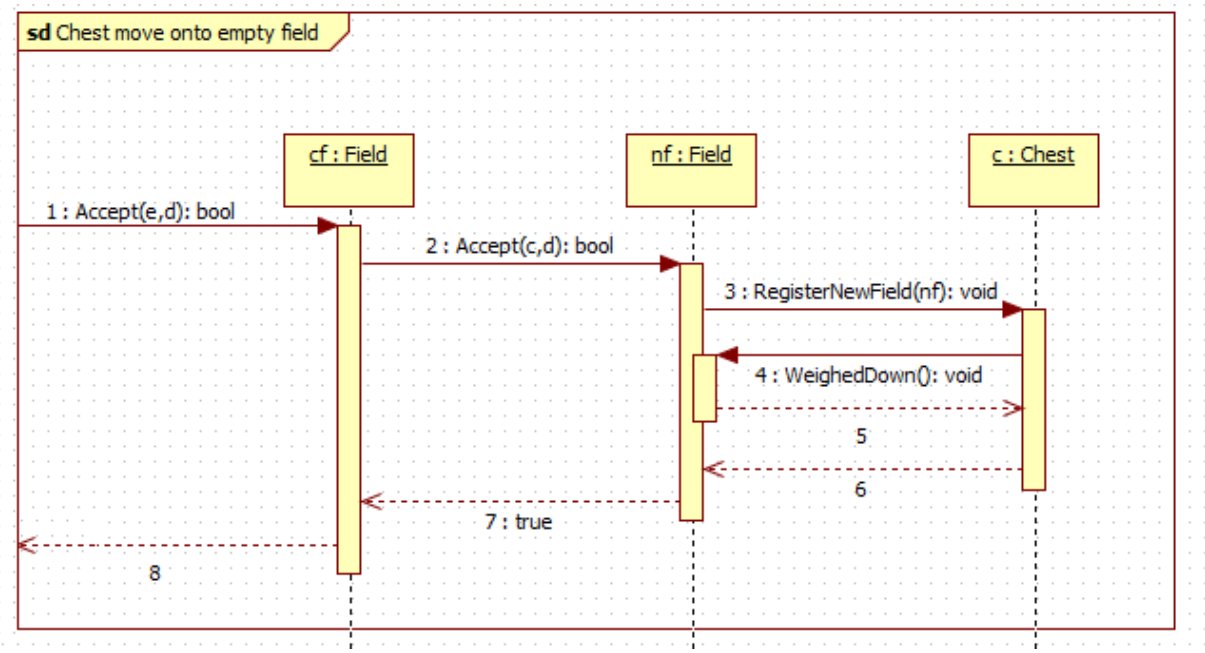
A dolgozó egy lyukra lép és meghal



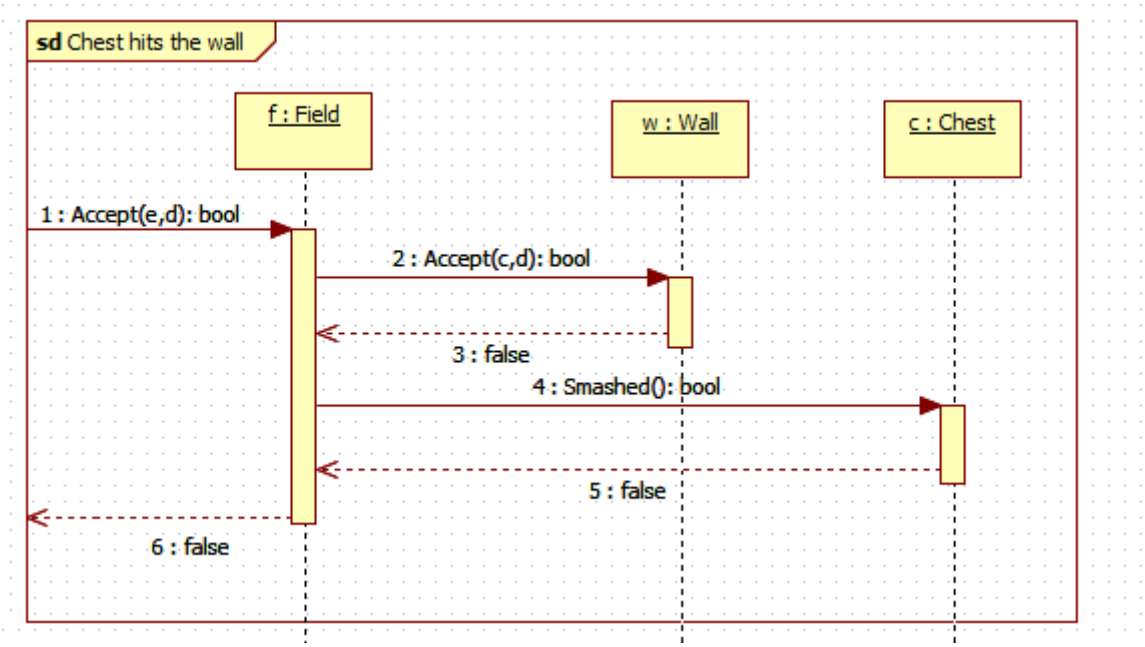
A dolgozó a nyelőre lép



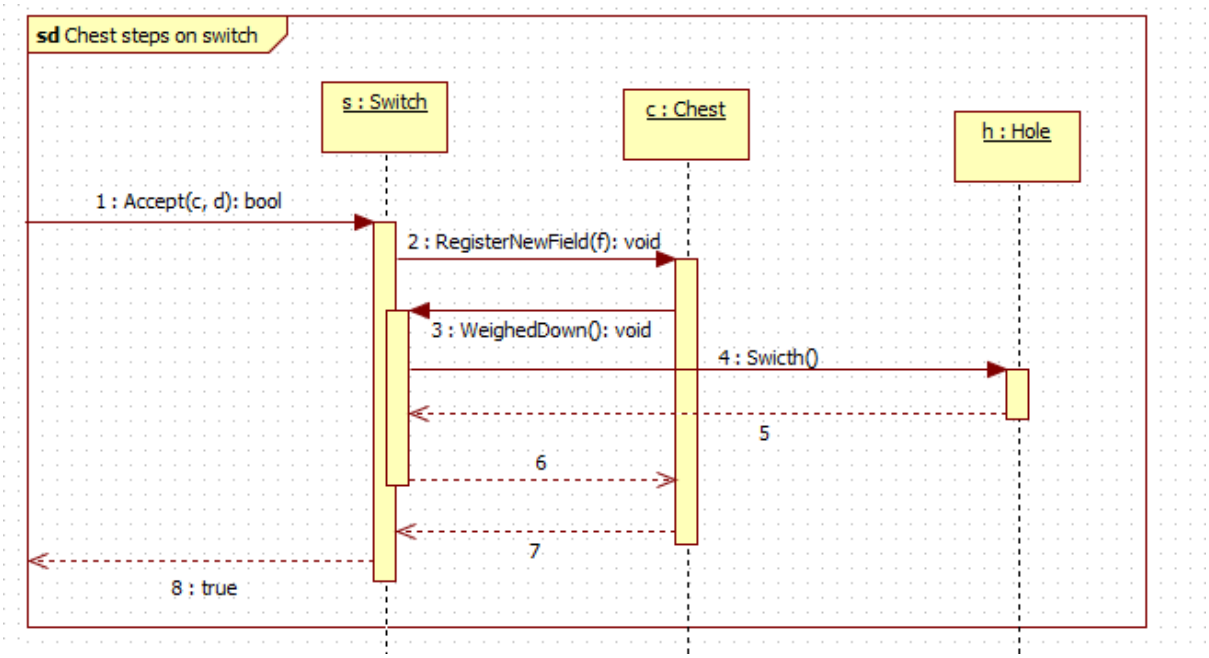
A dolgozó nekimegy egy másik dolgozónak



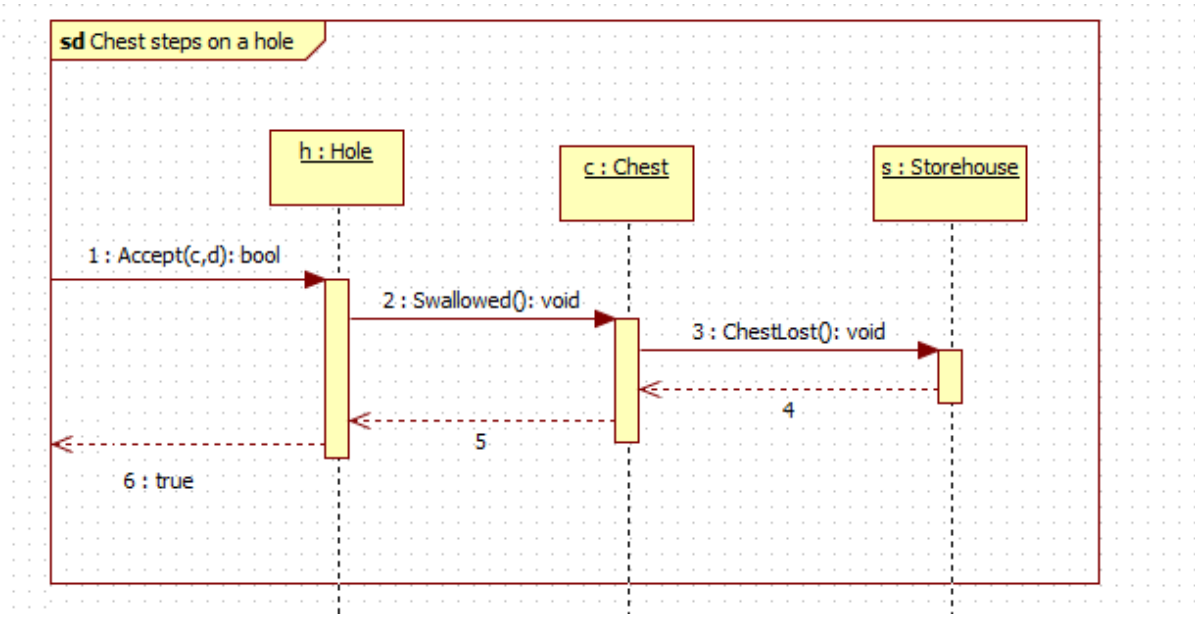
A láda eltolódik



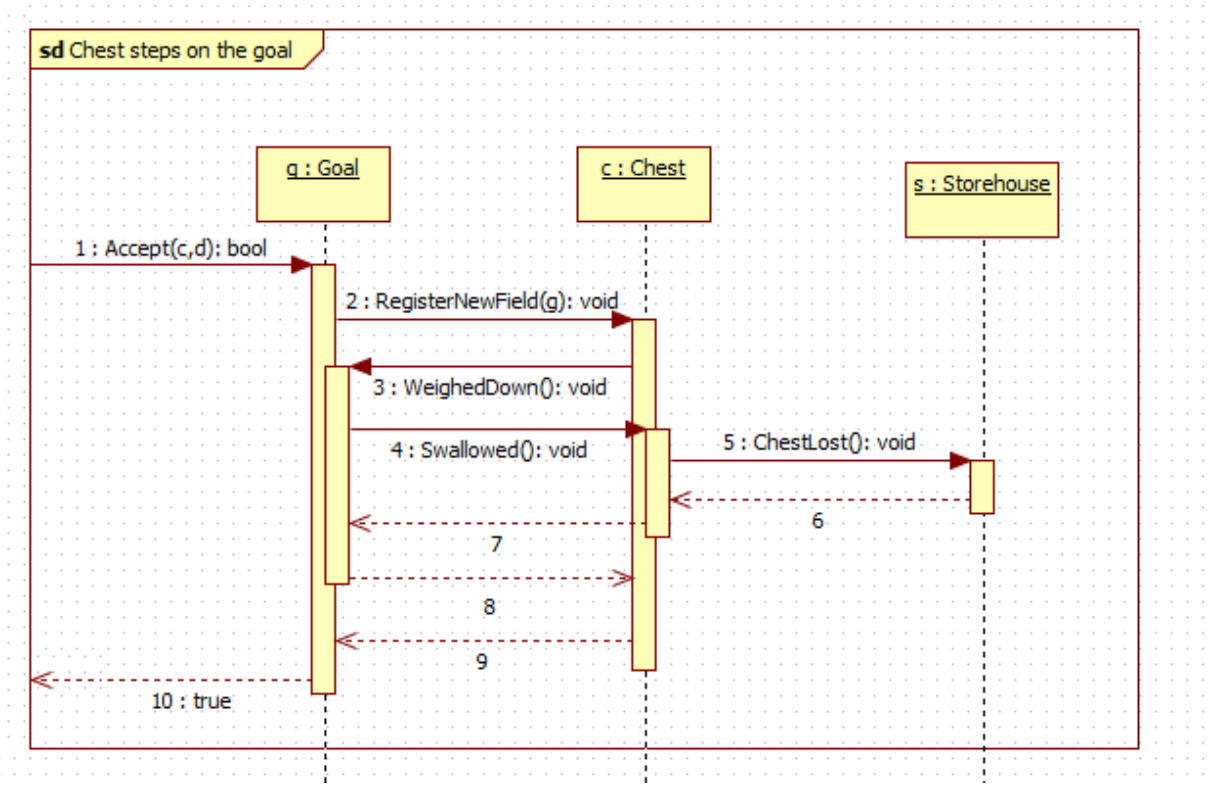
A ládát falnak tolják



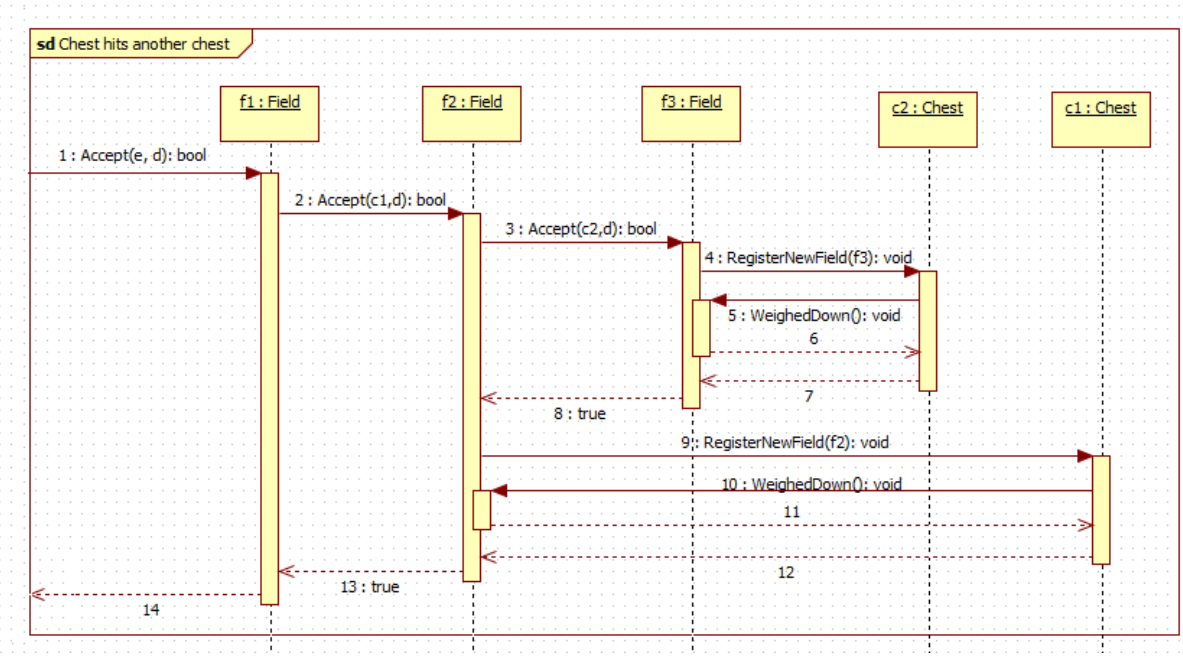
A ládát kapcsolóra tolják, ami így aktiválódik



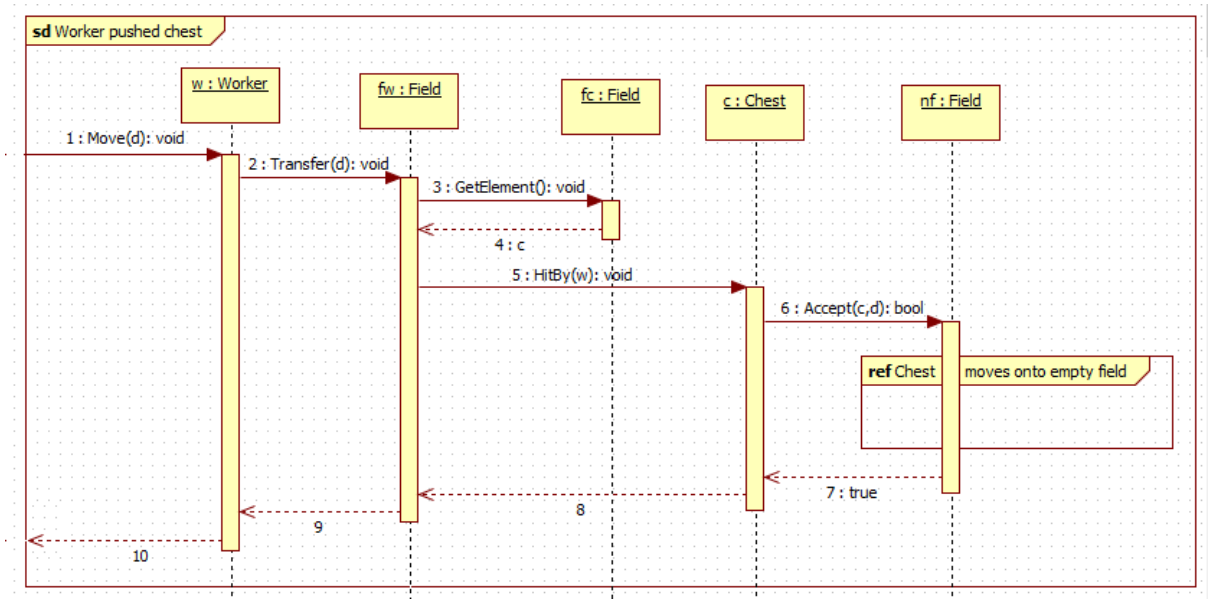
A ládát lyukra tolják, és a lyuk elnyeli (megsemmisül)



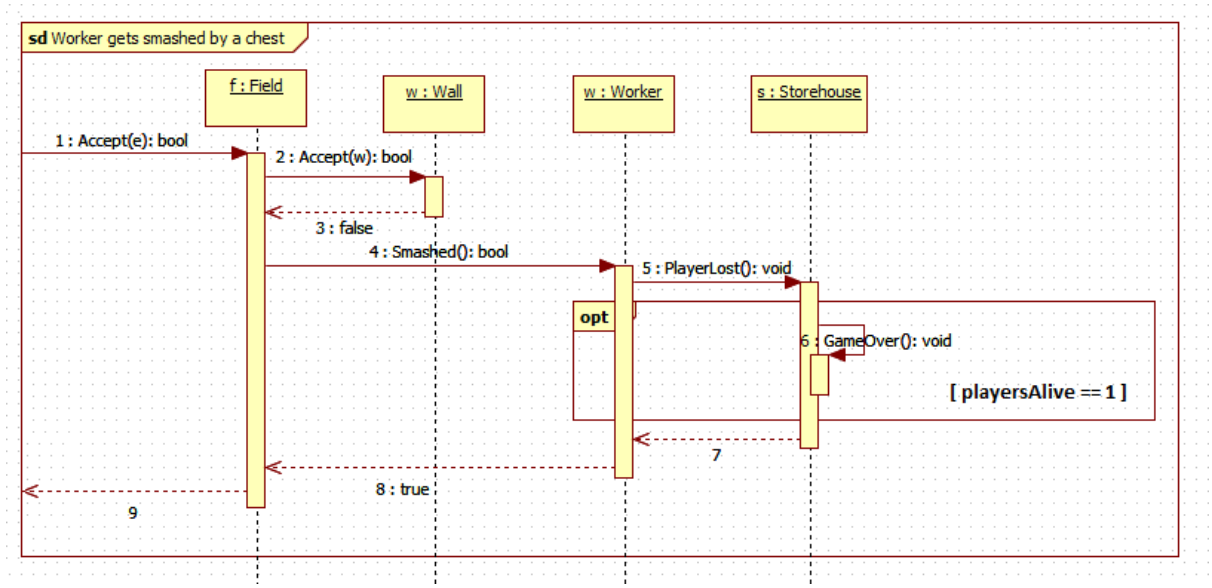
A ládát nyelőre tolják és a nyelő elfogadja



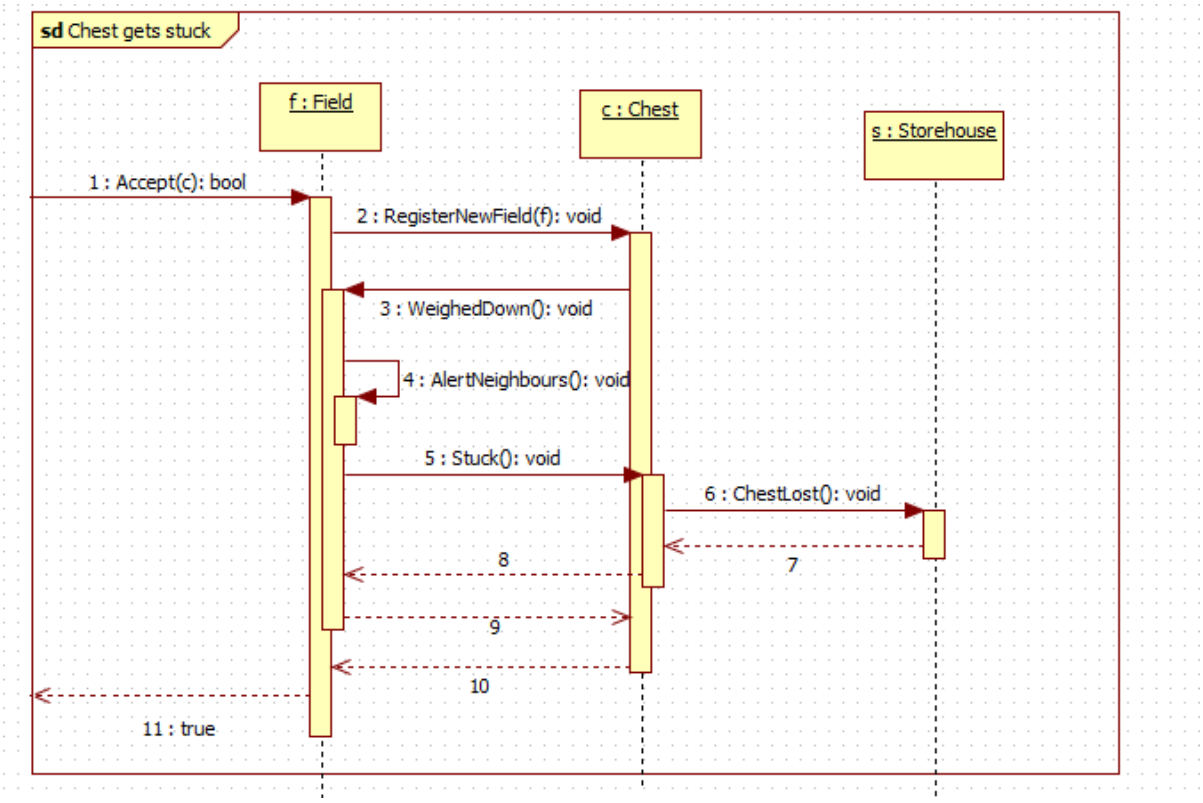
Egyik ládát a másiknak tolják



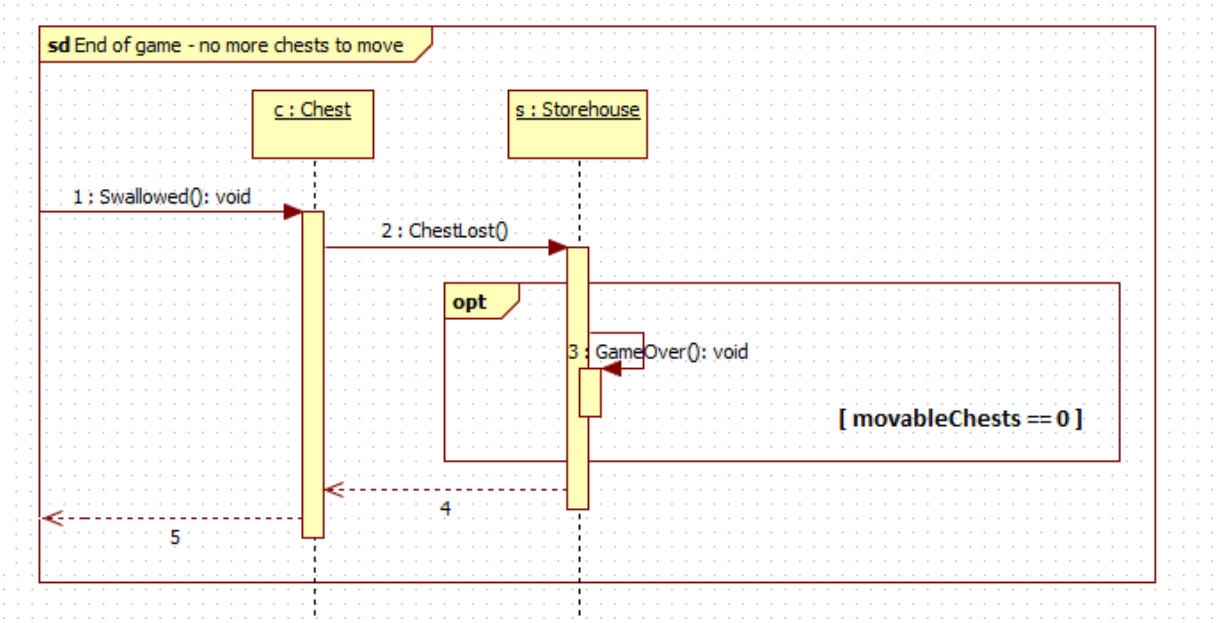
A dolgozó eltol egy ládát



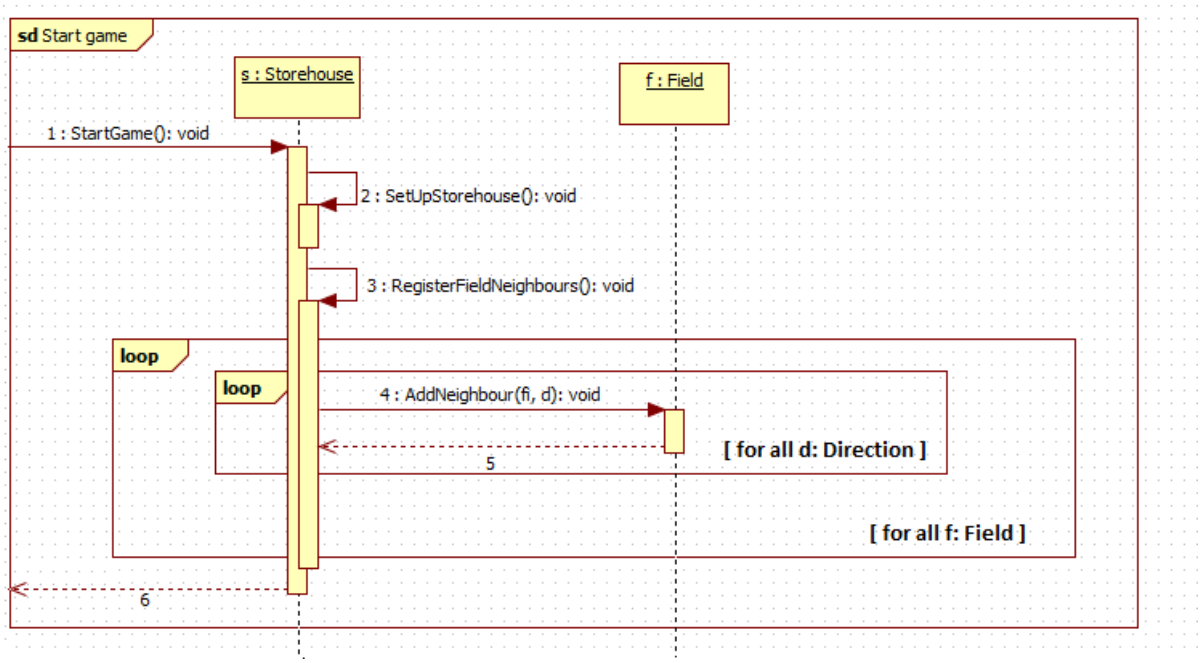
Egy fal mellett álló dolgozónak tolnak egy ládát, és a dolgozót összenyomja (megsemmisül)



Egy olyan mezőre tolnak ládát, ahonnan nem lehet kimozdítani

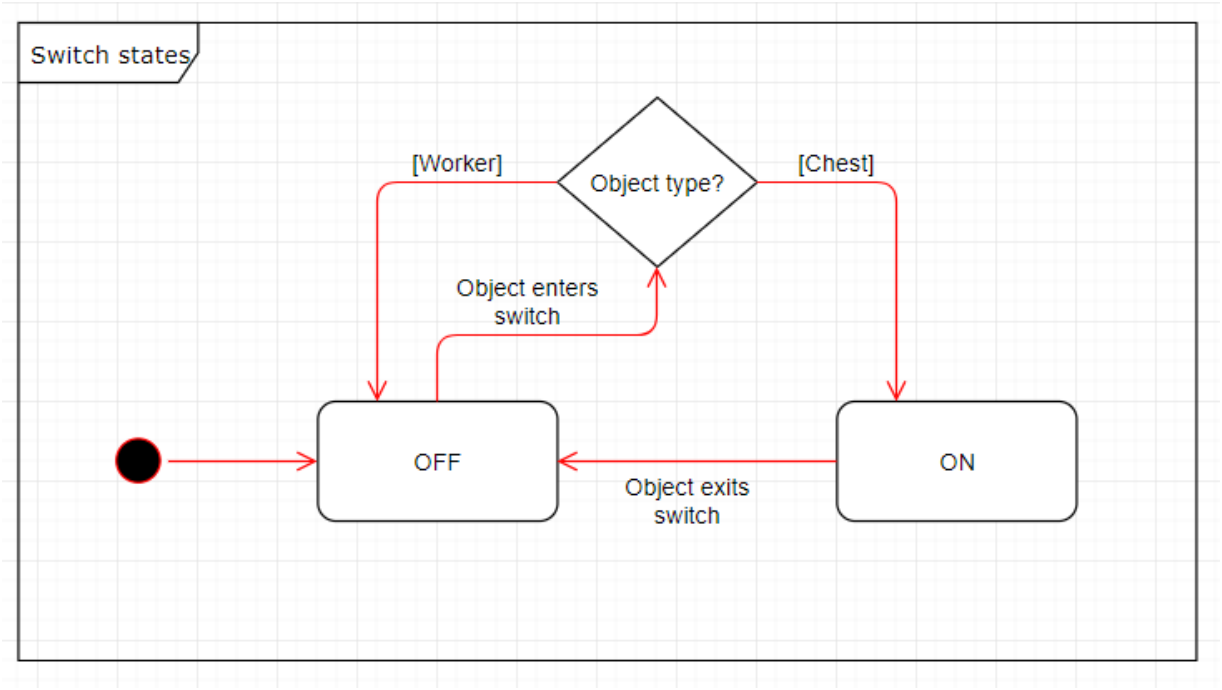


Elfogynak a mozdítható ládák a pályán (játék vége)



3.4.17 A játék indulása, a pálya felépülése

State-chart



Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.02.23.	3 óra	Németh, Majkut,	Osztály diagram tervezése, elkészítése (4.2);

		Szövő, Nagy	Objektum katalógus (4.1), Osztályok leírásának elkezdése (4.3)
2018.02.24.	3 óra	Simon	Objektum katalógus javítása (4.1), Osztályok leírásának javítása (4.3)
2018.02.24.	5 óra	Majkut	Osztály diagram javítása (4.2), Szekvencia diagrammok készítése (4.4)
2018.02.25.	1 óra	Németh, Szövő, Majkut, Simon	Osztályok leírásának véglegesítése (4.3)
2018.02.25.	1 óra	Nagy	Dokumentum szerkesztése (teljes)
2018.03.04.	2 óra	Majkut, Nagy	Szekvencia diagrammok (4.4), State chart (4.5)
2018.03.04.	1 óra	Németh, Szövő, Simon	Dokumentum szerkesztése

Szkeleton tervezése

/5. házi feladat/

1 – a_csapat

Konzulens:

Belákovics Ádám

Csapattagok:

Név
Simon Márta
Németh Marcell
Szövő Roland

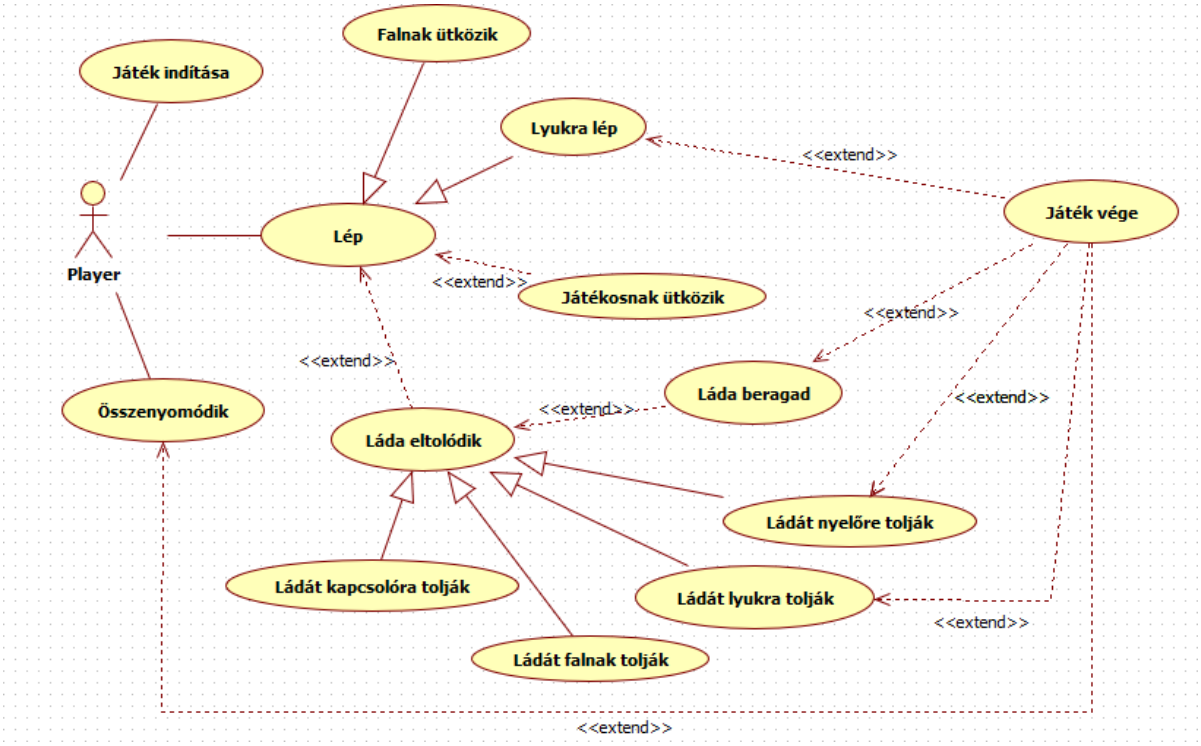
Nagy Szabina
Majkut Kristóf

2018.03.11.

5. Szkeleton tervezése

1. A szkeleton modell valóságos use-case-ei

1. Use-case diagram



2. Use-case leírások

Use-case neve	Játék indítása
Rövid leírás	A játékos elindítja az új játékot.
Aktorok	Player
Forgatókönyv	Új játék kezdésekor felépül a pálya. A ládák, nyelők, kapcsolók, lyukak, falak és játékosok a helyükre kerülnek.

Use-case neve	Összenyomódik
Rövid leírás	A játékos meghal.
Aktorok	Player
Forgatókönyv	A játékost fal, vagy mozdíthatatlan láda közé szorítja egy másik játékos. A játékosok száma csökken eggyel, ha egy mozgatható játékos maradt, a játék véget ér.

Use-case neve	Lép
Rövid leírás	A játékos lép.
Aktorok	Player
Forgatókönyv	A játékos az általa választott irányba lévő legközelebbi szabad mezőre lép.

Use-case neve	Falnak ütközik
Rövid leírás	A játékos falnak ütközik.
Aktorok	Player
Forgatókönyv	A játékos lépni kíván az adott irányba, de azon a mezőn fal található, így a lépés meghiúsul.

Use-case neve	Lyukra lép
Rövid leírás	A játékos lyukra lép.
Aktorok	Player
Forgatókönyv	A játékos egy nyitott állapotban lévő lyukra lép, majd meghal. A játékosok száma csökken eggyel, ha egy mozgatható játékos maradt, a játék véget ér.

Use-case neve	Játékosnak ütközik
Rövid leírás	A játékos egy másik játékosnak ütközik.
Aktorok	Player
Forgatókönyv	A játékos lépni kíván az adott irányba, de azon a mezőn egy másik játékos áll, így a lépés meghiúsul.

Use-case neve	Láda eltolódik
Rövid leírás	A játékos eltol egy ládát.
Aktorok	Player
Forgatókönyv	A játékos az adott irányba kíván eltolni egy mozgatható ládát.

Use-case neve	Láda beragad
Rövid leírás	A láda nem tolható tovább.
Aktorok	-
Forgatókönyv	A ládát akadály veszi körül (fal vagy mozdíthatatlan láda). Ha mindegyik pályán lévő láda mozdíthatatlan, a játék véget ér.

Use-case neve	Ládát nyelőre tolják
Rövid leírás	A láda eltűnik.
Aktorok	-
Forgatókönyv	A ládát egy nyelő mezőre tolja egy játékos. A láda eltűnik, az érte járó pont hozzáadódik az adott nyelőhöz tartozó játékos pontszámához. Ha nincs több mozdítható láda a pályán, a játék véget ér.

Use-case neve	Ládát lyukra tolják
Rövid leírás	A láda eltűnik.
Aktorok	-
Forgatókönyv	A ládát egy nyitott állapotban lévő lyukra tolja egy játékos. A láda eltűnik, pontot egyik játékos sem kap.

Use-case neve	Ládát falnak tolják
Rövid leírás	A láda falba ütközik.
Aktorok	-
Forgatókönyv	A ládát nem lehet a megadott irányba léptetni, hiszen azon a mezőn fal található, így a lépés meghiúsul.

Use-case neve	Ládát kapcsolóra tolják
Rövid leírás	A ládát kapcsolóra tolják.
Aktorok	-
Forgatókönyv	A ládát olyan mezőre tolják, amelyen kapcsoló található, ekkor a kapcsolóhoz tartozó lyuk kinyílik.

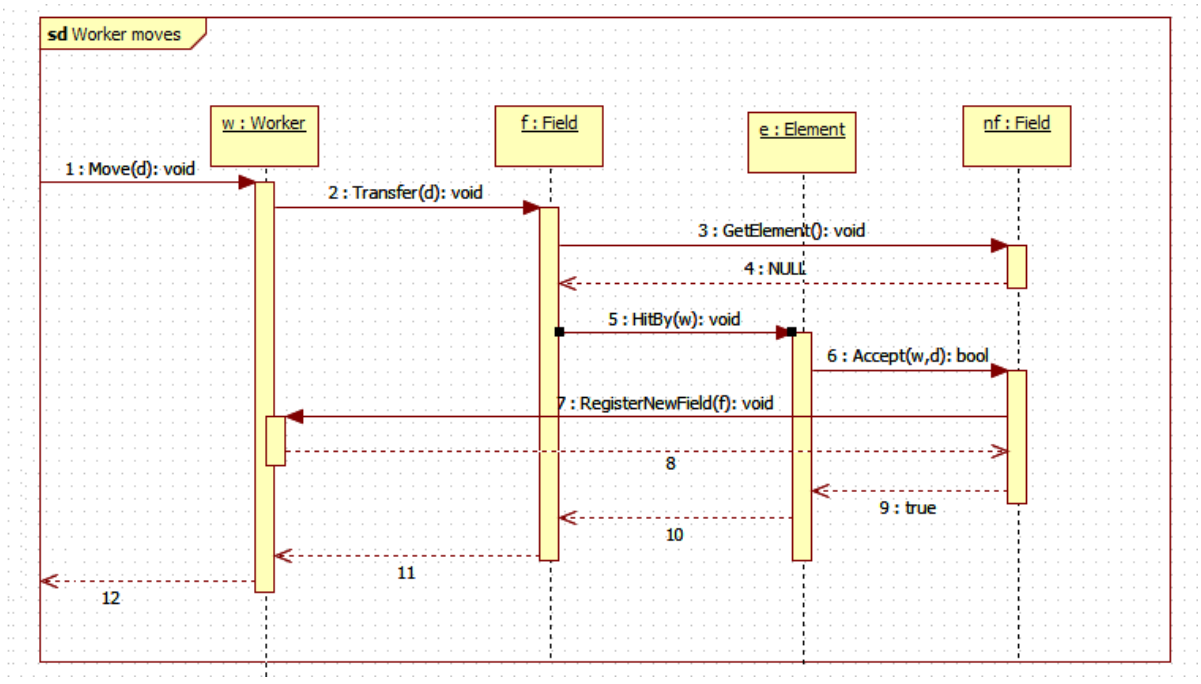
Use-case neve	Játék vége
Rövid leírás	A játék véget ér.
Aktorok	-
Forgatókönyv	A játék véget érhet a következő módon: 1. Egy játékos marad a pályán. 2. Az összes pályán maradt láda mozdíthatatlan.

2. A szkeleton kezelői felületének terve, dialógusok

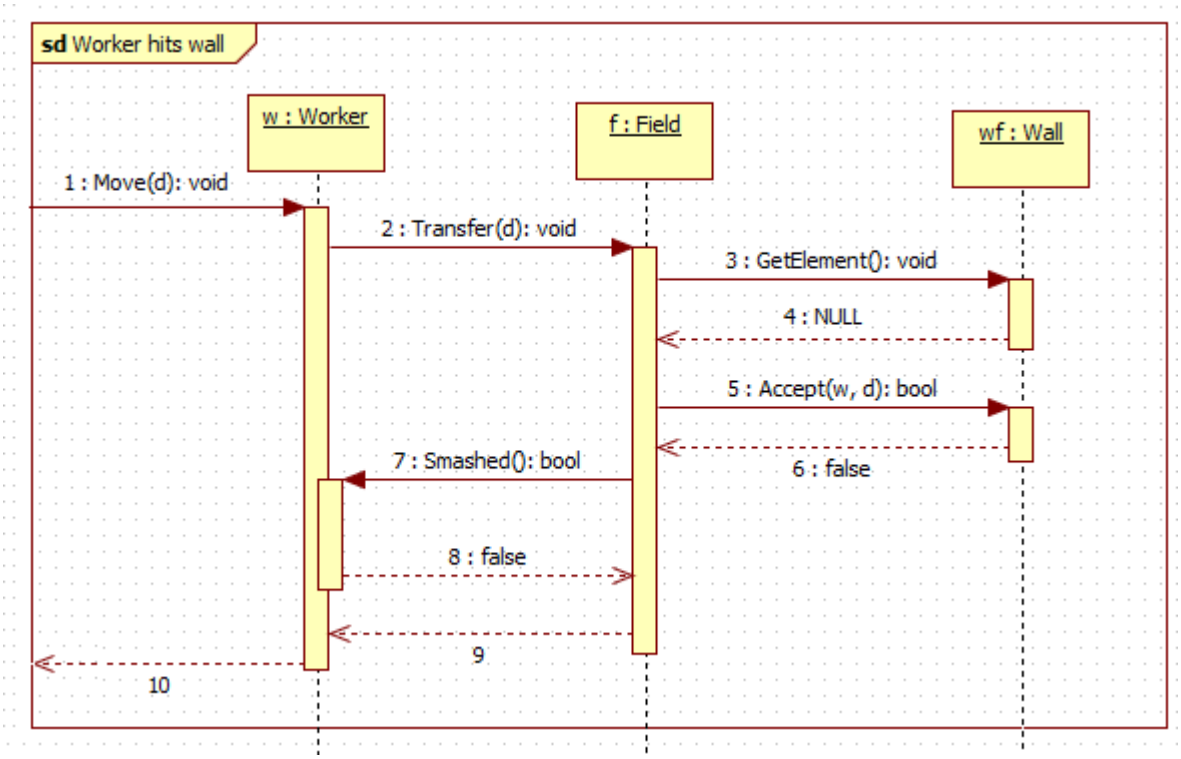
1. Játékos lép
    1. Sima mezőre
    2. Lyukra
        1. Utolsó játékos? (I/N)
    3. Kapcsolóra

4. Nyelőre
2. Játékos (csak) ládát tol
    1. Egy ládát tol
        1. Láda beragad? (I/N)
            1. Ha igen, utolsó láda volt? (I/N)
    2. Két (vagy több) ládát tol
    3. Ládát falnak
    4. Ládát kapcsolóra
    5. Ládát nyelőre
        1. Utolsó láda? (I/N)
    6. Ládát lyukra
        1. Utolsó láda? (I/N)
3. Játékos játékost tol
    1. Közvetlenül
    2. Ládával
        1. Üresre
        2. Falnak
        3. Ládának
            1. Falnak? (I/N)

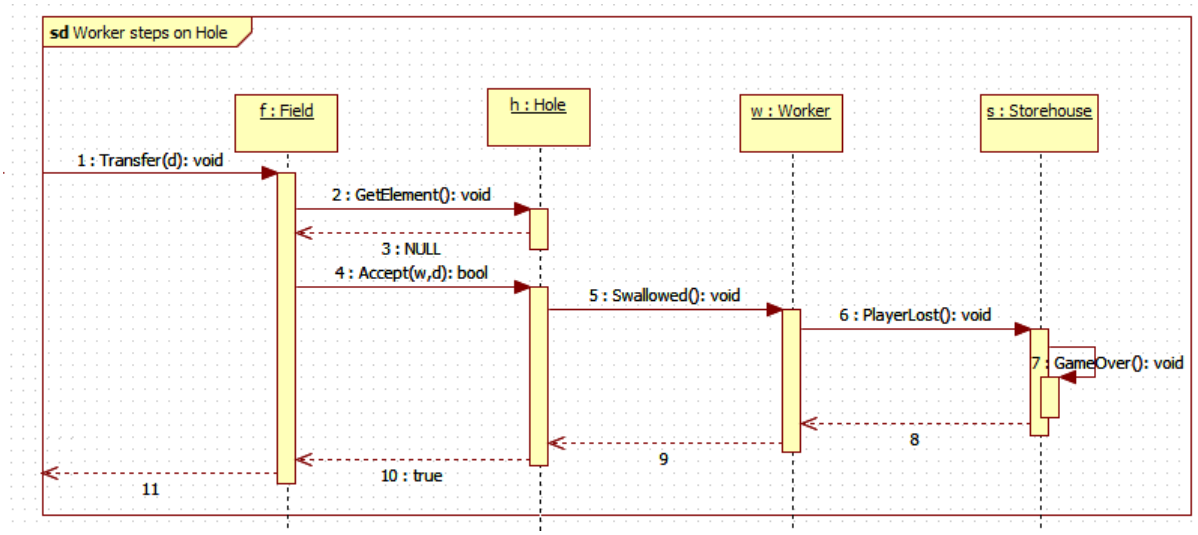
3. Szekvencia diagramok a belső működésre



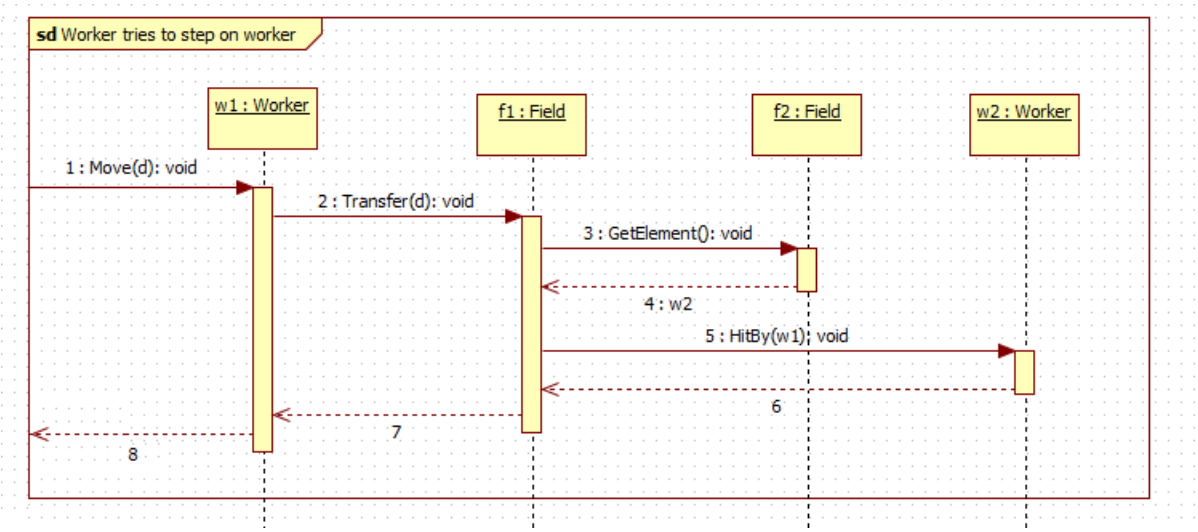
5.3.1. A dolgozó lép



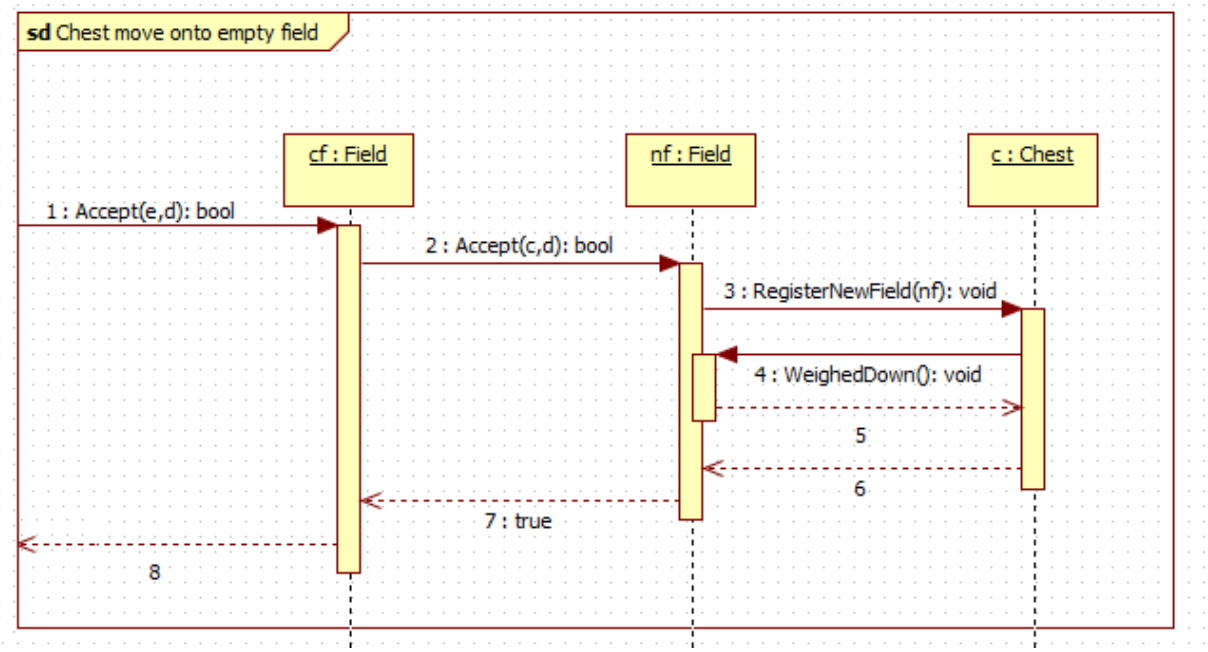
A dolgozó falnak ütközik



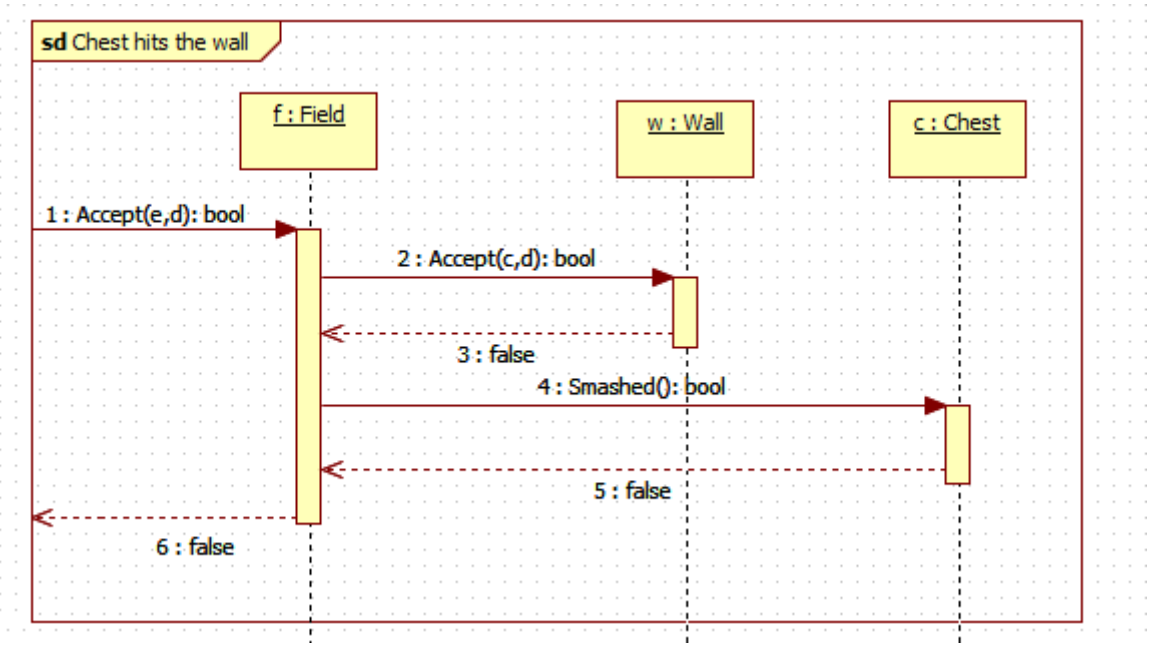
A dolgozó egy lyukra lép és meghal



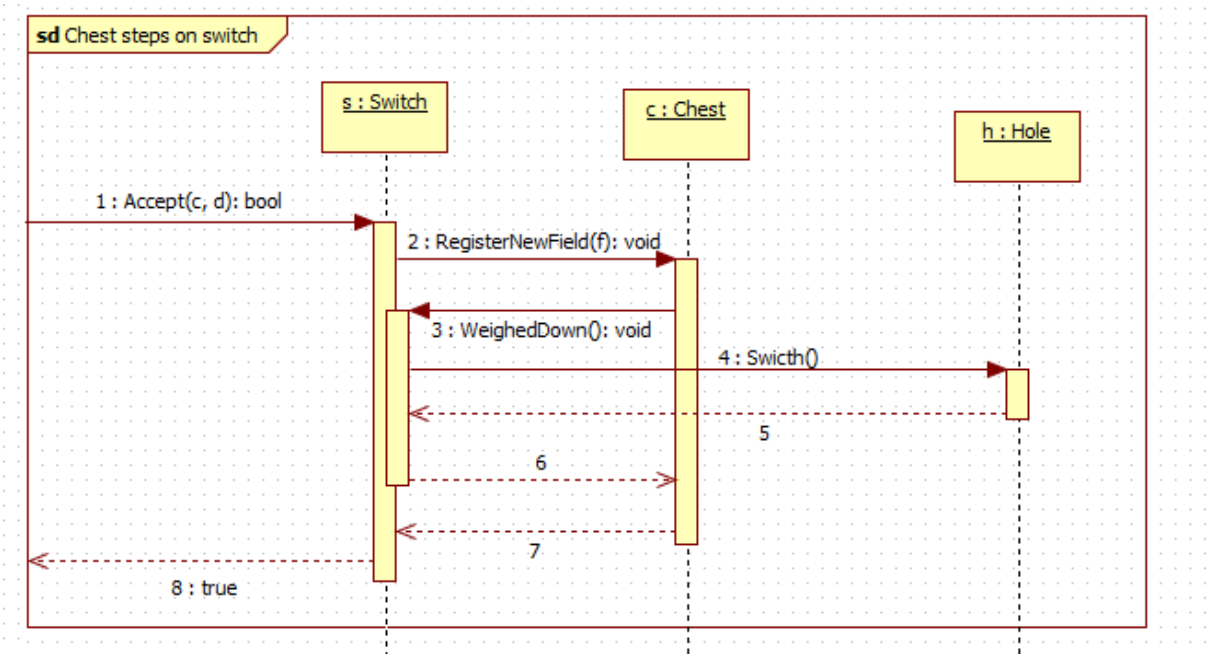
A dolgozó nekimegy egy másik dolgozónak



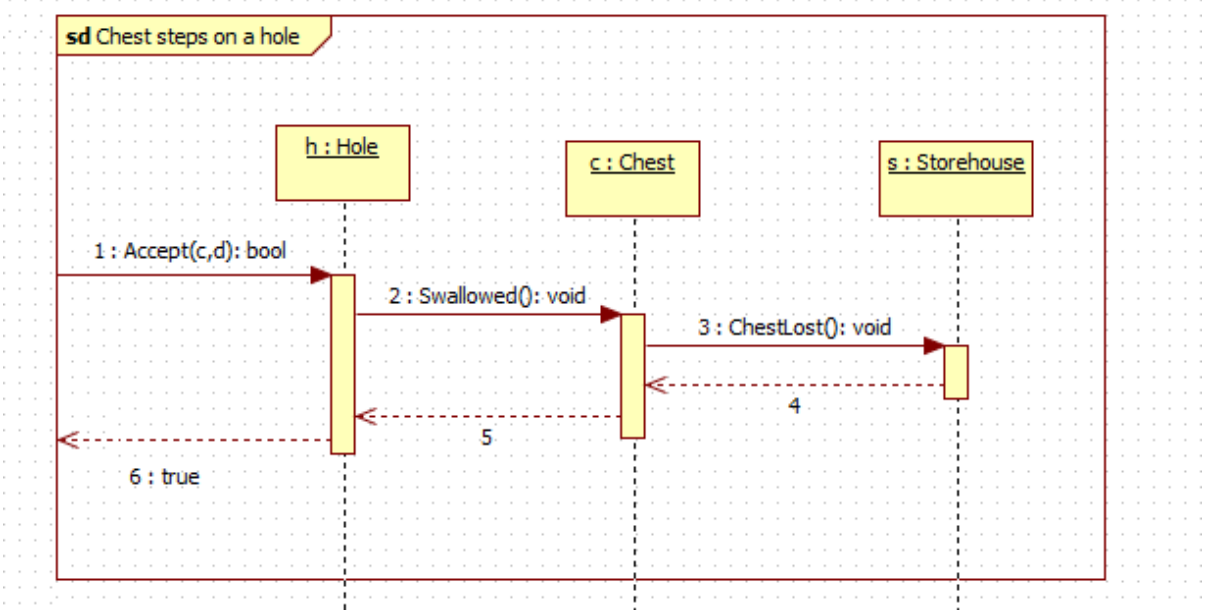
A láda eltolódik



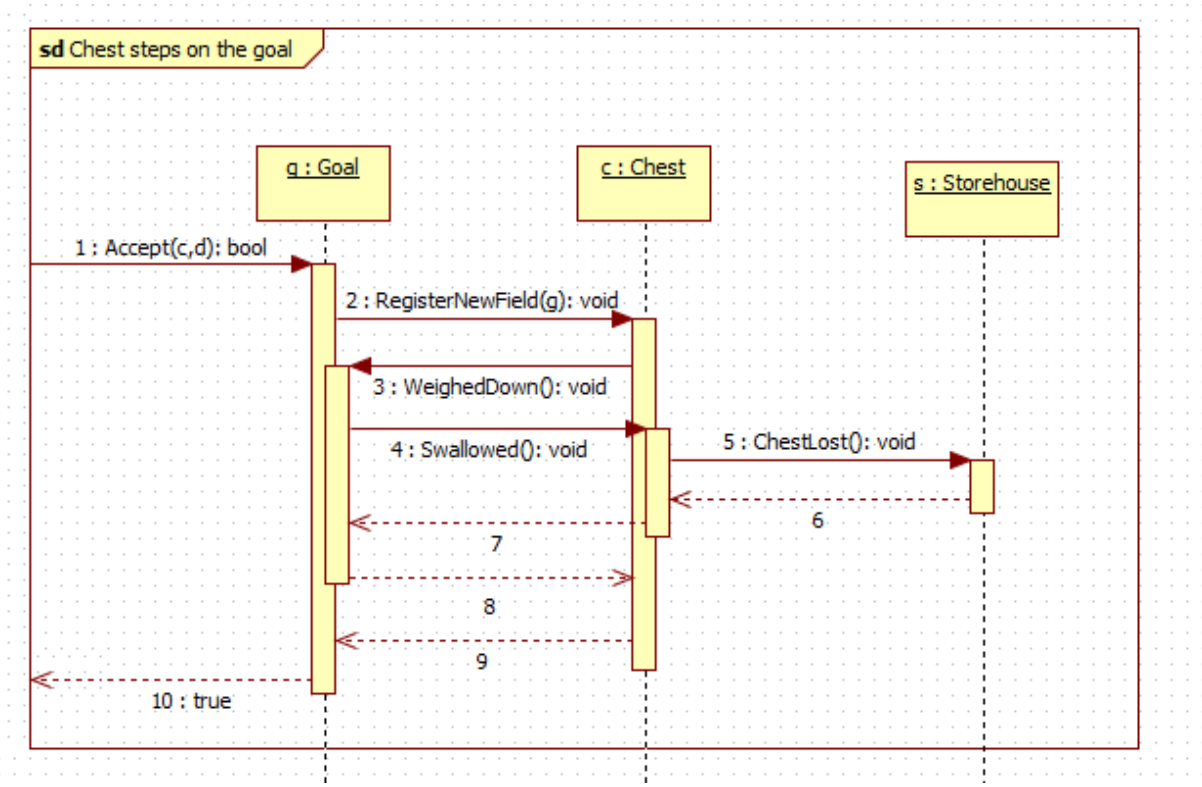
A ládát falnak tolják



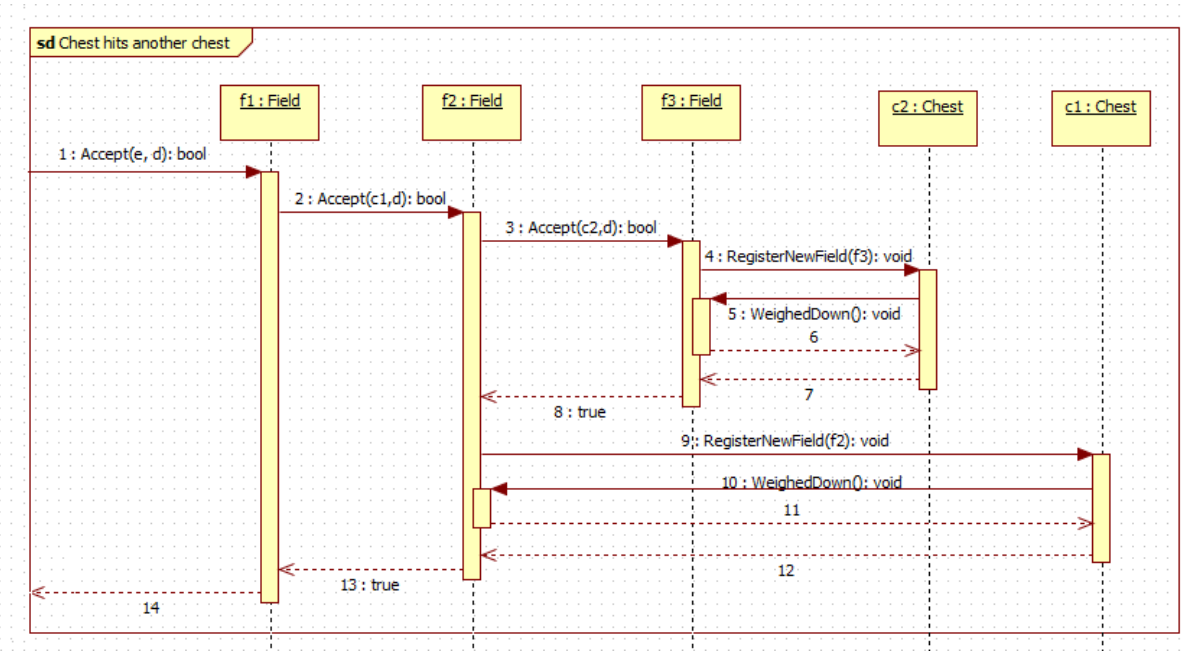
A ládát kapcsolóra tolják, ami így aktiválódik



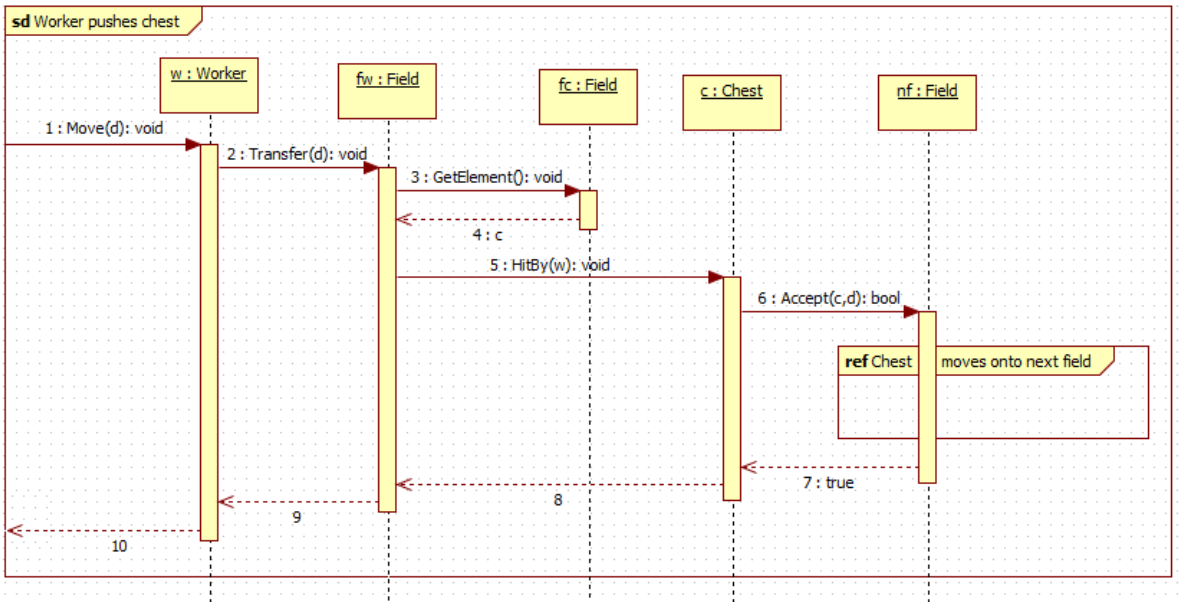
A ládát lyukra tolják, és a lyuk elnyeli (megsemmisül)



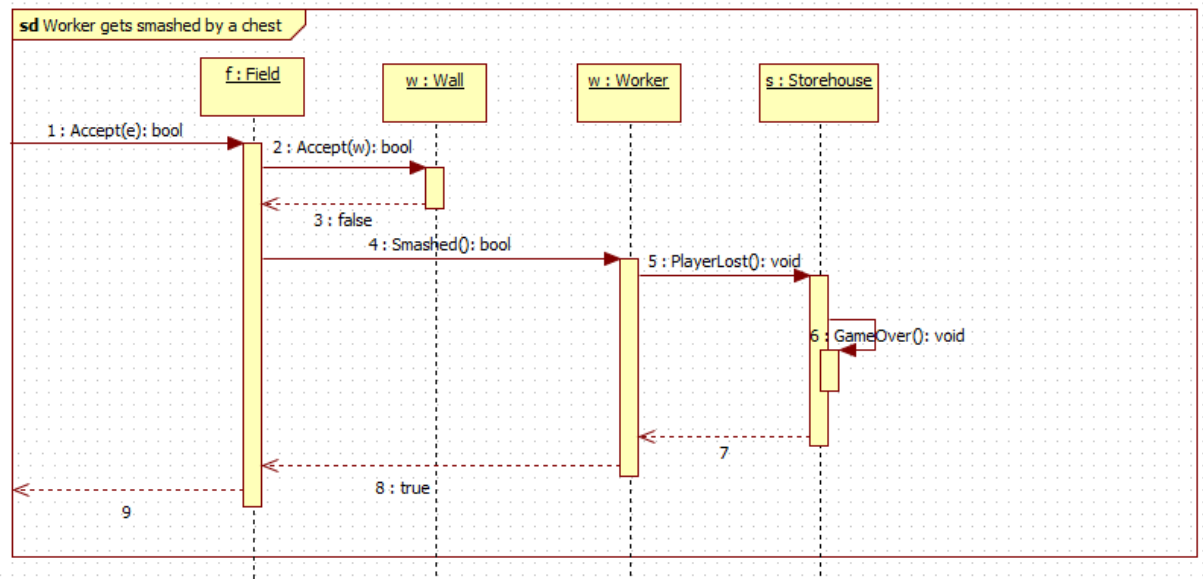
A ládát nyelőre tolják és a nyelő elfogadja



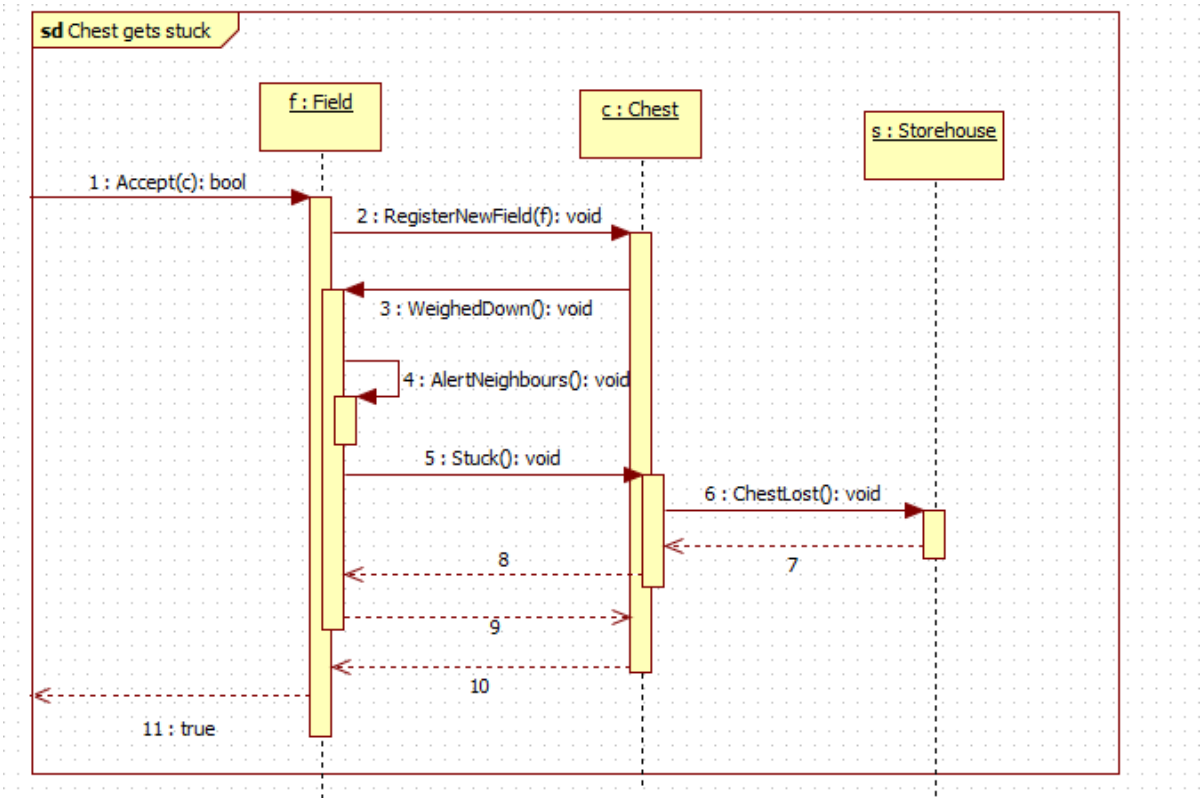
Egyik ládát a másiknak tolják



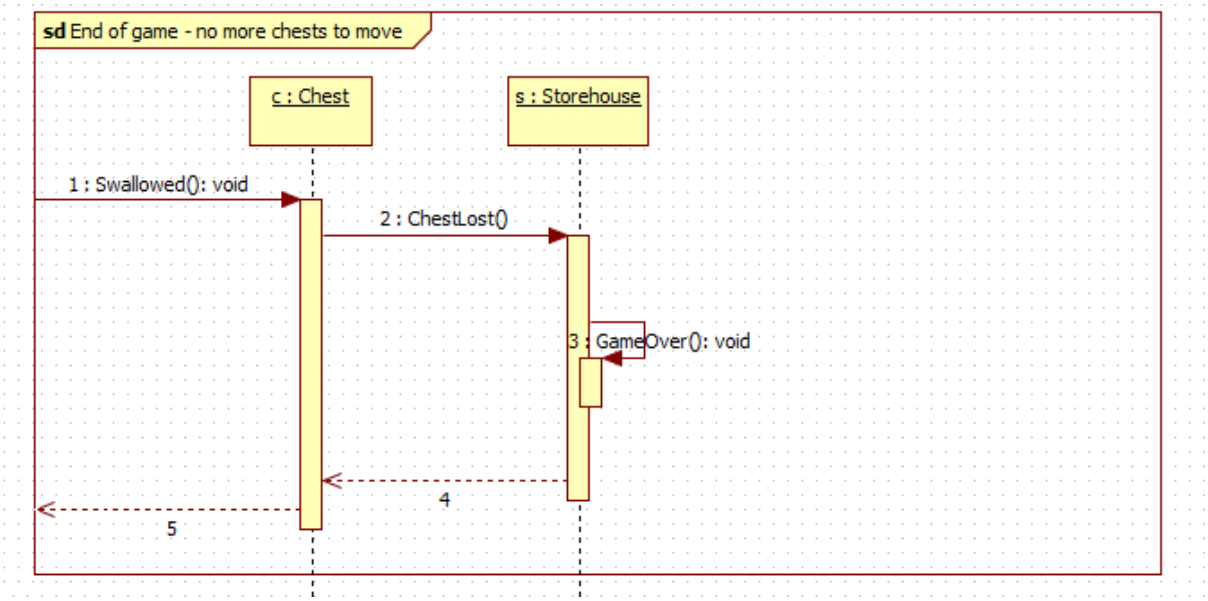
A dolgozó eltol egy ládát



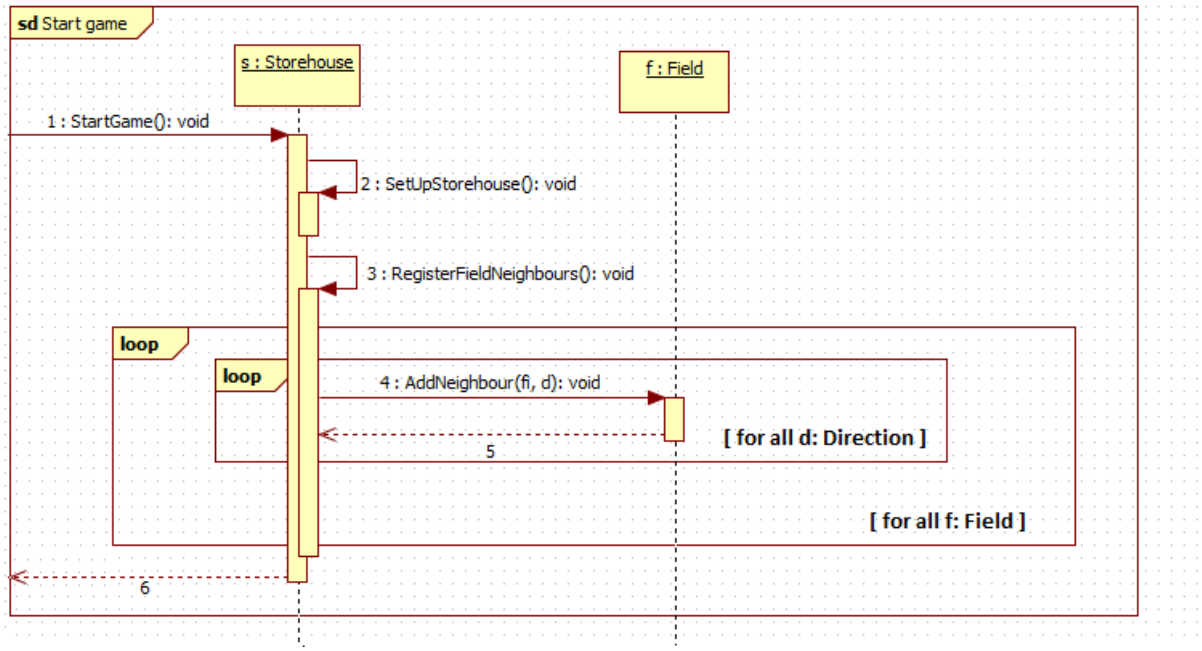
Egy fal mellett álló dolgozónak tolnak egy ládát, és a dolgozót összenyomja (megsemmisül)



Egy olyan mezőre tolnak ládát, ahonnan nem lehet kimozdítani

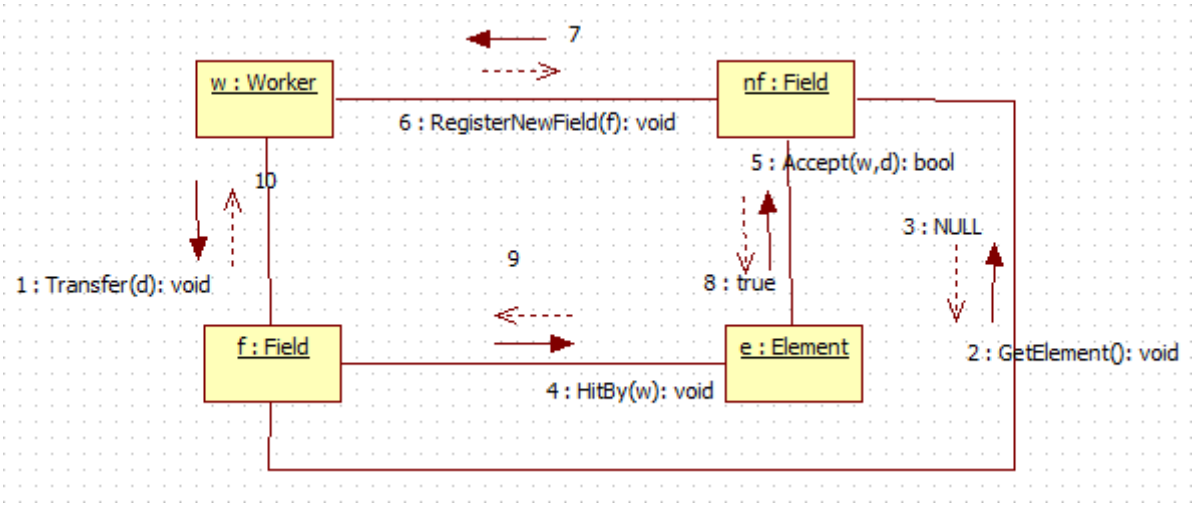


Elfogynak a mozdítható ládák a pályán (játék vége)

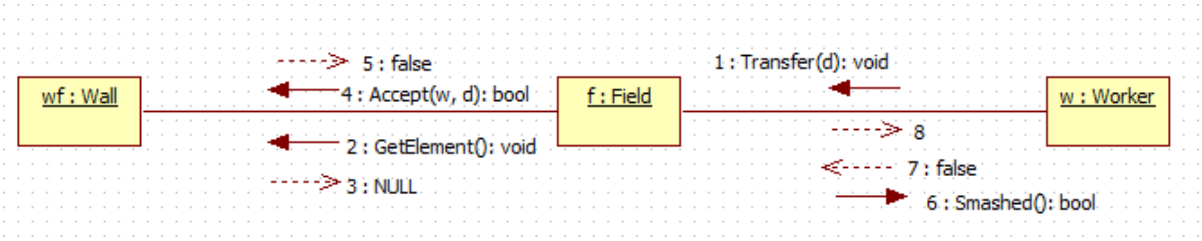


A játék indulása, a pálya felépülése

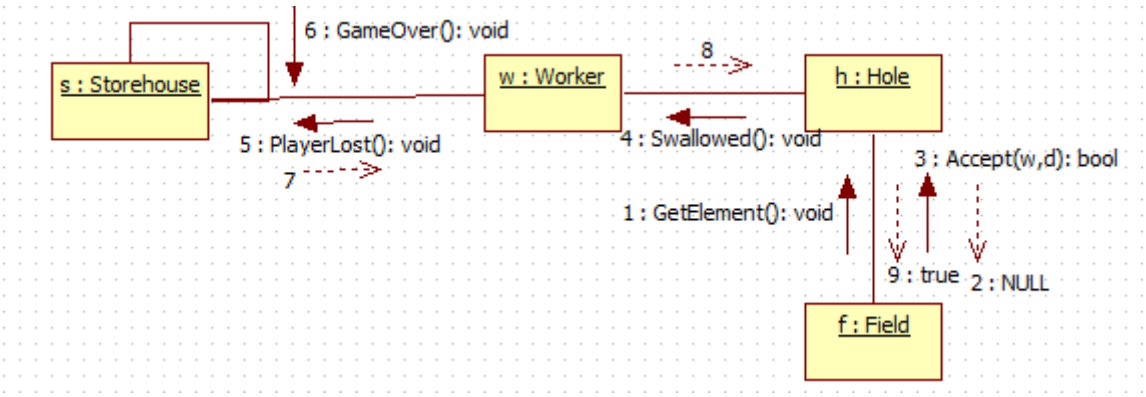
4. Kommunikációs diagramok



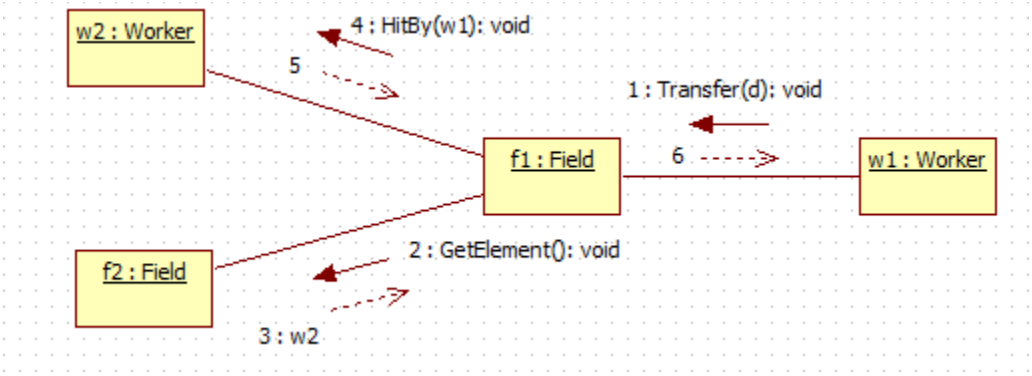
A dolgozó lép



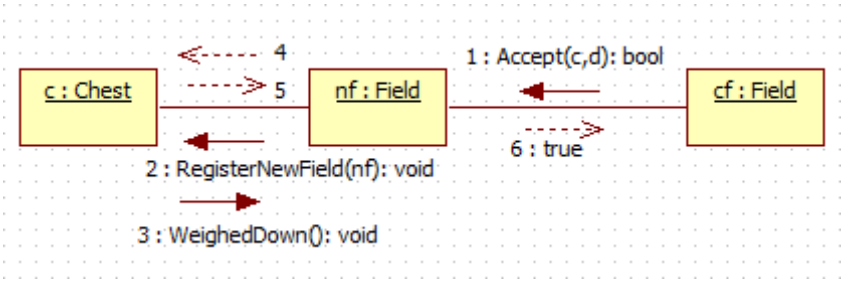
A dolgozó falnak ütközik



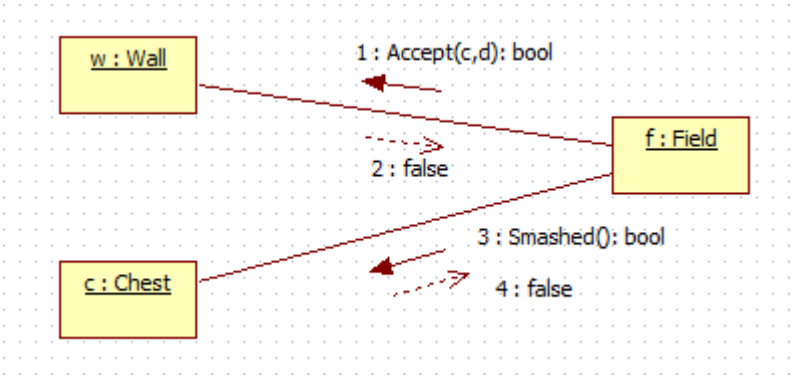
A dolgozó egy lyukra lép és meghal



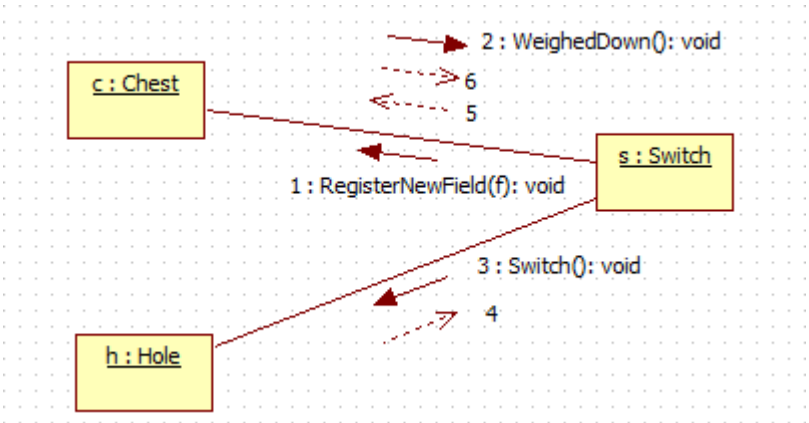
A dolgozó nekimegy egy másik dolgozónak



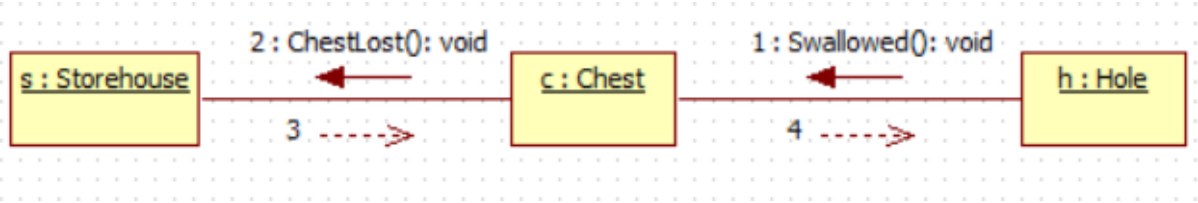
A láda eltolódik



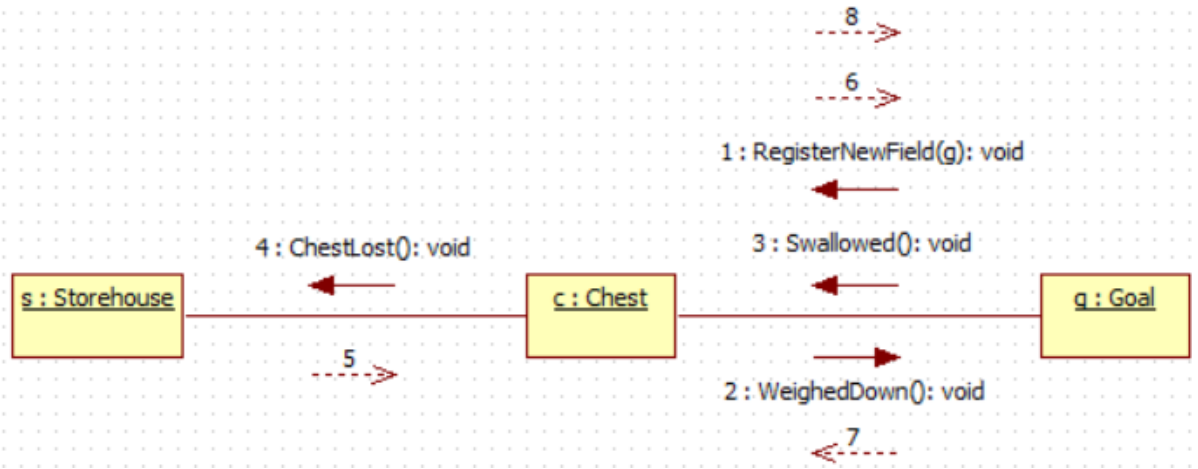
A ládát falnak tolják



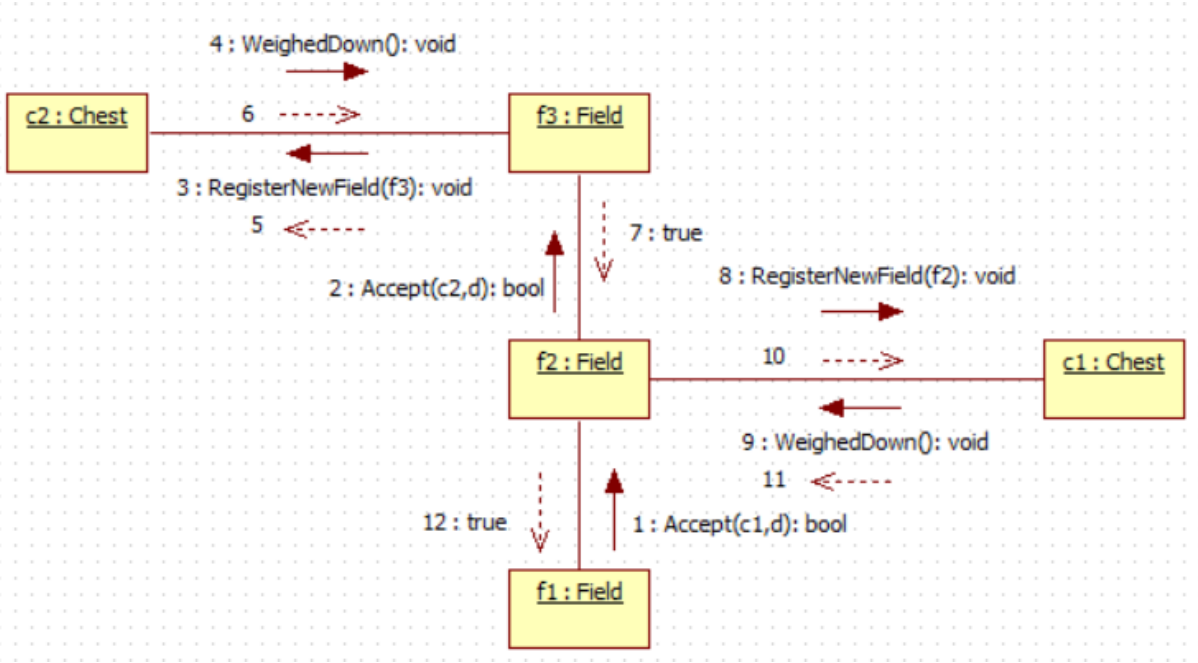
A ládát kapcsolóra tolják, ami így aktiválódik



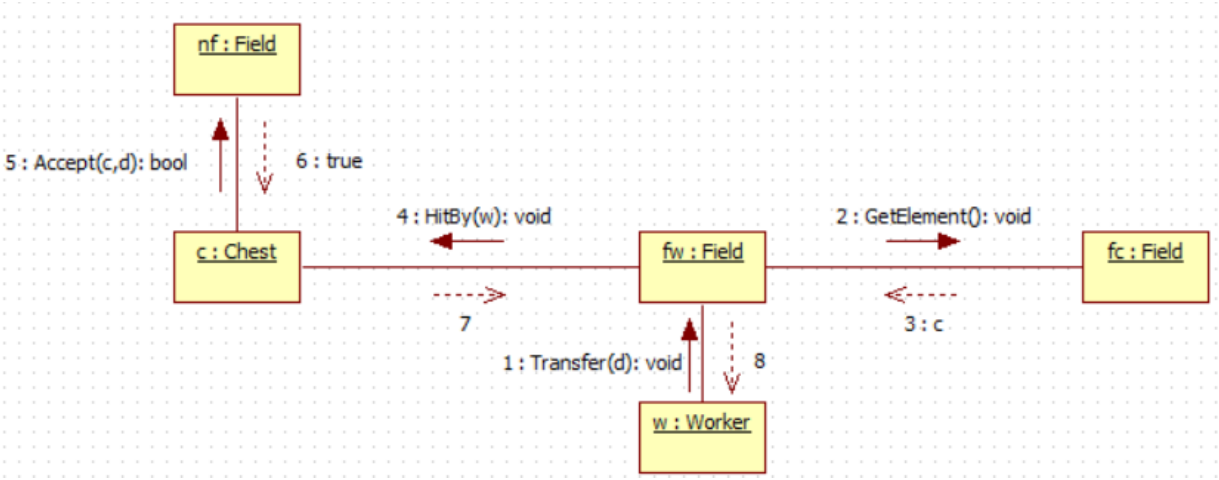
A ládát lyukra tolják, és a lyuk elnyeli (megsemmisül)



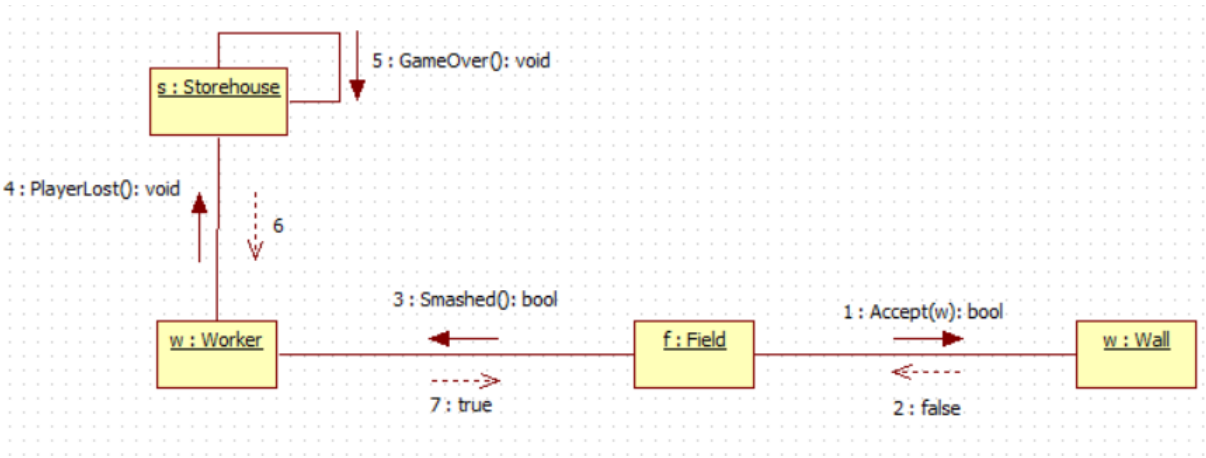
A ládát nyelőre tolják és a nyelő elfogadja



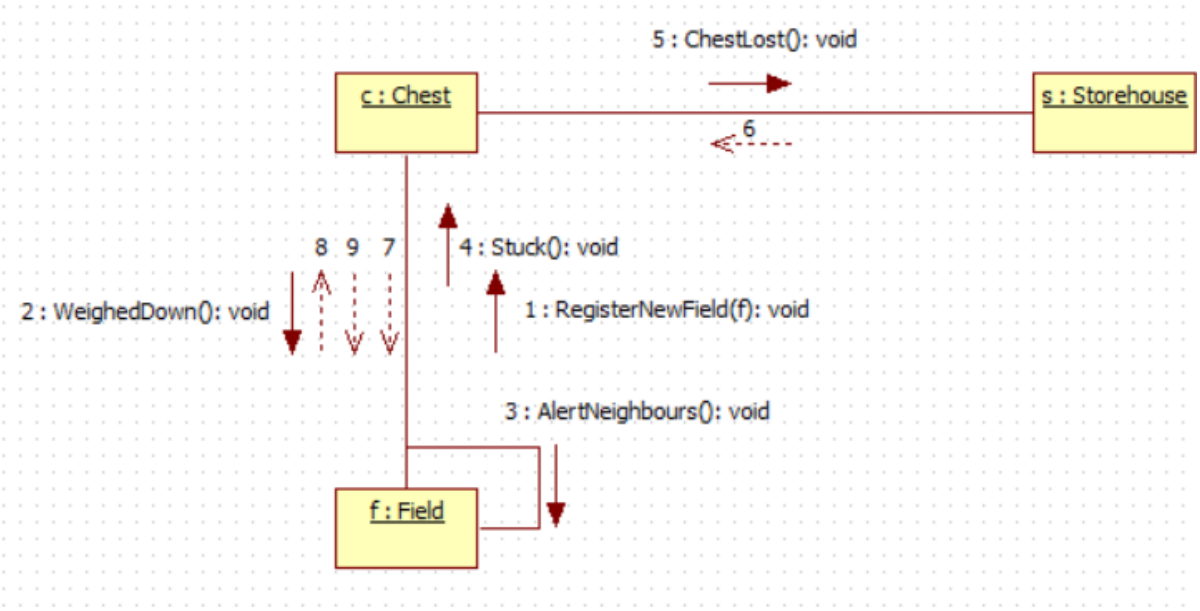
Egyik ládát a másiknak tolják



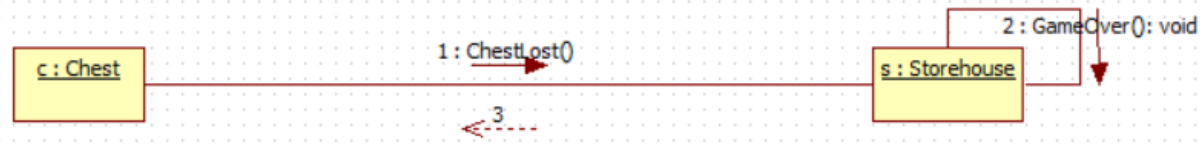
A dolgozó eltol egy ládát



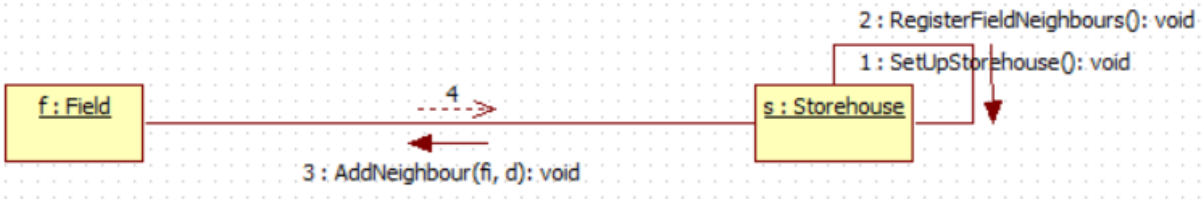
Egy fal mellett álló dolgozónak tolnak egy ládát, és a dolgozót összenyomja (megsemmisül)



Egy olyan mezőre tolnak ládát, ahonnan nem lehet kimozdítani



Elfogynak a mozdítható ládák a pályán (játék vége)



A játék indulása, a pálya felépülése

Megjegyzés: A White Star UML modellezőszoftver saját függvényszámozást használ ezért tér el minimálisan a kommunikációs diagrammokon látható jelölés a megszokottól.

5. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.03.11	4 óra	Szövő, Majkut	5.1.1 Use-case diagram
2018.03.10	1 óra	Majkut	5.2 Kezelői felület terve
2018.03.11	3 óra	Simon, Németh	5.4 Kommunikációs diagrammok
2018.03.11	2 óra	Nagy	5.1.2 Use-case leírások

Szkeleton tervezése

/6. házi feladat/

1 – a_csapat

Konzulens:

Belákovics Ádám

Csapattagok:

Név
Simon Márta
Németh Marcell
Szövő Roland
Nagy Szabina

Majkut Kristóf

2018.03.11.

6. Szkeleton beadás

1. Fordítási és futtatási útmutató

1. Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Chest.java	828 bájt	2018. március 17	Chest osztály
Direction.java	76 bájt	2018. március 17	Direction enumeráció
Element.java	811 bájt	2018. március 17	Element osztály
Field.java	1925 bájt	2018. március 17	Field osztály
Goal.java	347 bájt	2018. március 17	Goal osztály
Hole.java	429 bájt	2018. március 17	Hole osztály
Storehouse.java	700 bájt	2018. március 17	Storehouse osztály
Switch.java	280 bájt	2018. március 17	Switch osztály
UI.java	5437 bájt	2018. március 17	User Interface
Wall.java	256 bájt	2018. március 17	Wall osztály
Worker.java	4040 bájt	2018. március 17	Worker osztály

2. Fordítás

```
set PATH=%PATH%;C:\Program Files\Java\jdk1.8.0_121\bin
set JAVA_HOME=C:\Program Files\Java\jdk1.8.0_121
set CLASSPATH=%CLASSPATH%;

mkdir bin
javac -d bin src/com/projlab/acsapat/*.java
```

3. Futtatás

```
cd bin
java com.projlab.acsapat.UI

cd ..
```

2. Értékelés

Tag neve	Munka százalékban
Nagy	20%
Németh	20%
Majkut	20%
Simon	20%
Szövő	20%

3. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.03.15	6 óra	Majkut, Simon, Nagy, Németh, Szövő	Program megírása és kommentezés
2018.03.15	3óra	Majkut	User Interface készítése
2018.03.17	1 óra	Simon, Szövő	Dokumentáció szerkesztése

Prototípus koncepciója

/7. házi feladat/

1 – a_csapat

Konzulens:

Belákovics Ádám

Csapattagok:

Név
Simon Márta

Németh Marcell
Szövő Roland
Nagy Szabina
Majkut Kristóf

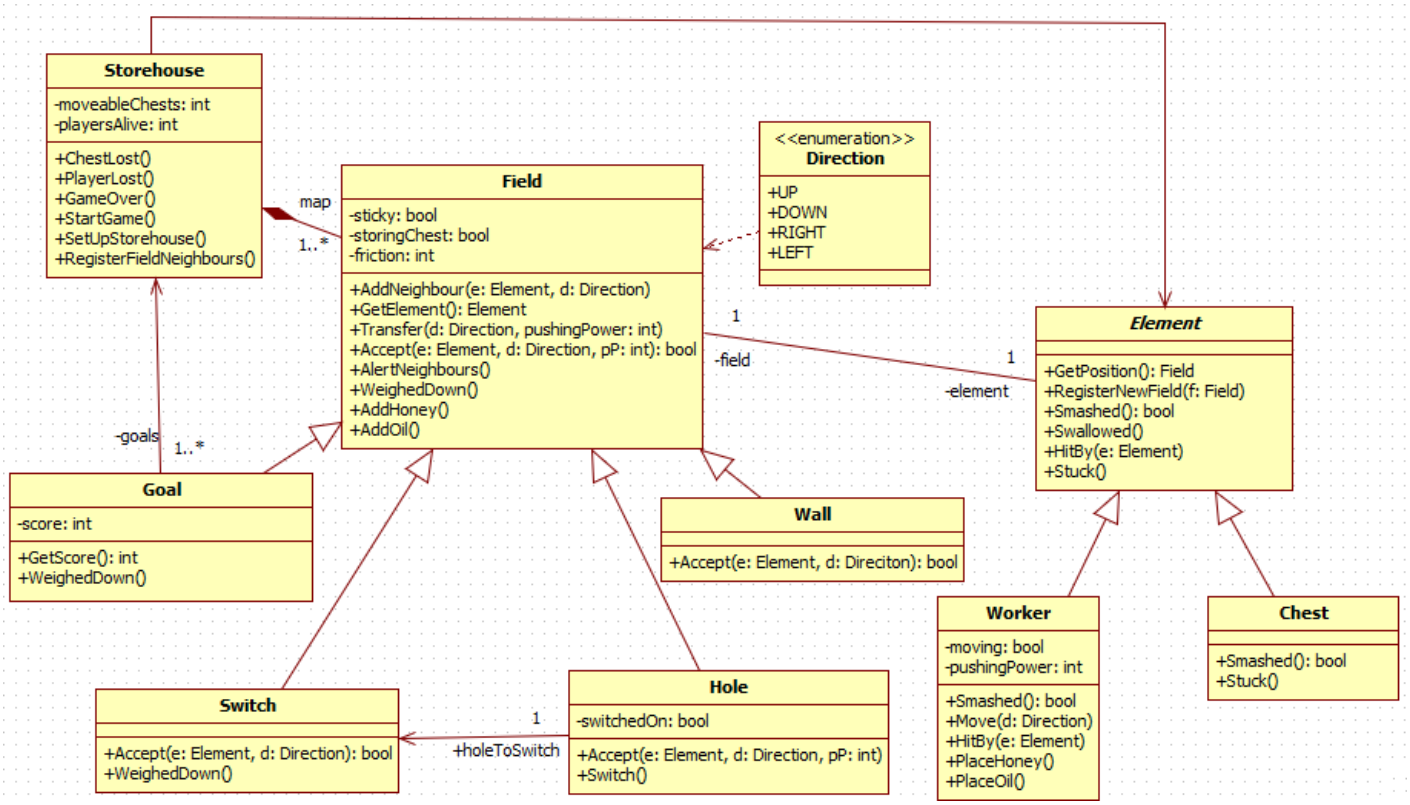
2018.03.26.

7. Prototípus koncepciója

7.0 Változás hatása a modellre

A játék úgy módosul, hogy a pályán lévő mezők kiegészülnek egy új tulajdonsággal, a súrlódással (friction), a dolgozók pedig egy "tolási erővel" (pushingPower). Ennek a két attribútumnak a megjelenése megakadályozza, hogy egy dolgozó akárhány ládát eltoljon, az adott dolgozó és az általa tolt láda mezőjének súrlódása fogja meghatározni, hány láda lesz éppen akkor eltolható. A dolgozó befolyásolhatja az egyes mezők súrlódását oly módon, hogy a mézet vagy olajat helyezhet el rajtuk, ezzel növelve, illetve csökkentve azok súrlódását.
Az osztályok ennek megfelelően megkapják az új attribútumokat, illetve metódusokat, a Field Transfer() függvénye és Accept() függvénye pedig kiegészül egy további paraméterrel, a dolgozó tolóerejével.

Módosult osztálydiagram



Új vagy megváltozó metódusok

7.0.2.1 Field

- **Felelősség**
  A pályát alkotó mezők ősosztálya.
- **Új attribútumok**
  - **bool storingChest:** Tárolja, hogy éppen van-e rajta láda.
  - **int friction:** A mező surlódása, ilyen nehéz az adott mezőről továbbtolni egy ládát.
- **Új metódusok**
  - **void AddHoney():** A mező súrlódását (int friction) növeli meg egy adott értékkel.
  - **void AddOil():** A mező surlódását (int friction) csökkenti egy adott értékkel.
- **Módosított metódusok**
  - **void Transfer(Direction d, *int pushingPower*):** A transfer függvény meghívásakor a Worker most már azt is továbbadja paraméterként, hogy mekkora erővel tolja meg az előtte lévő ládát.
  - **bool Accept(Element e, Direction d, *int pushingPower*):** Egy láda csak akkor tolható tovább, ha az őt érő tolóerő nagyobb, mint a ládát tartalmazó mező surlódása, ezért az Accept() metódus is megkapja paraméterként a tolóerőt.

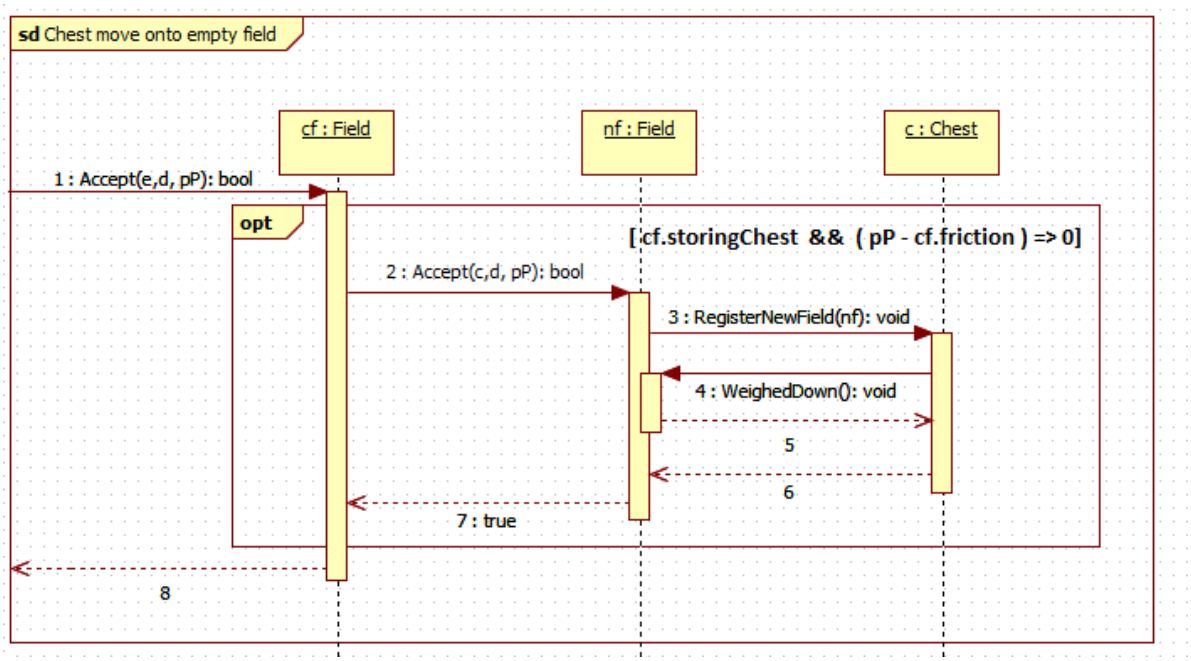
7.0.2.1 Worker

- **Felelősség**

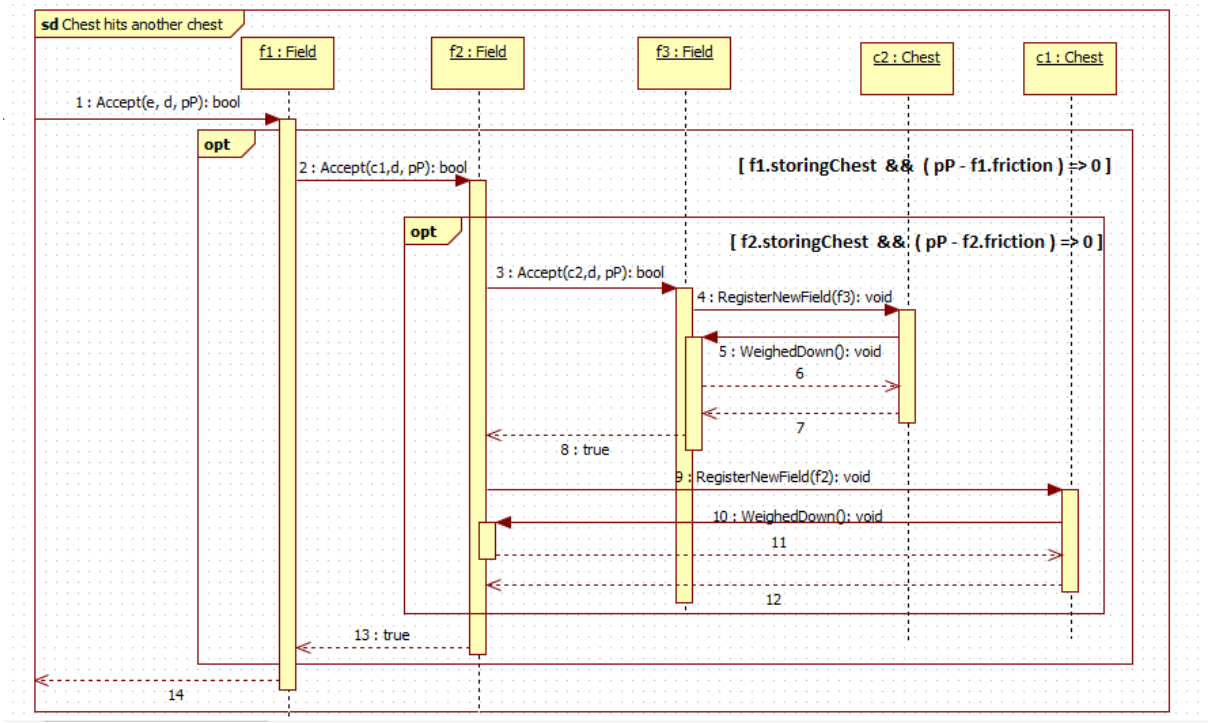
Dolgozót, mint mozdítható pályaelemet megvalósító osztály. Képes a Chest-ek eltolására.

- **Új attribútumok**
  - **int pushingPower:** Az az erő, amellyel megtolja az előtte lévő ládát
- **Metódusok**
  - **void PlaceHoney():** Mézet tesz a éppen őt tartalmazó mezőre, ezzel megnövelve annak a surlódását.
  - **void PlaceOil():** Olajat tesz a éppen őt tartalmazó mezőre, ezzel csökkentve annak a surlódását.

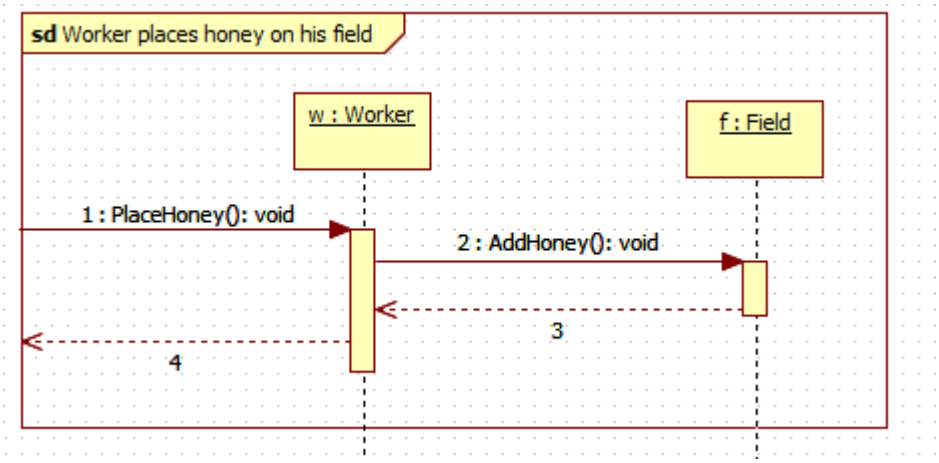
Szekvencia-diagramok



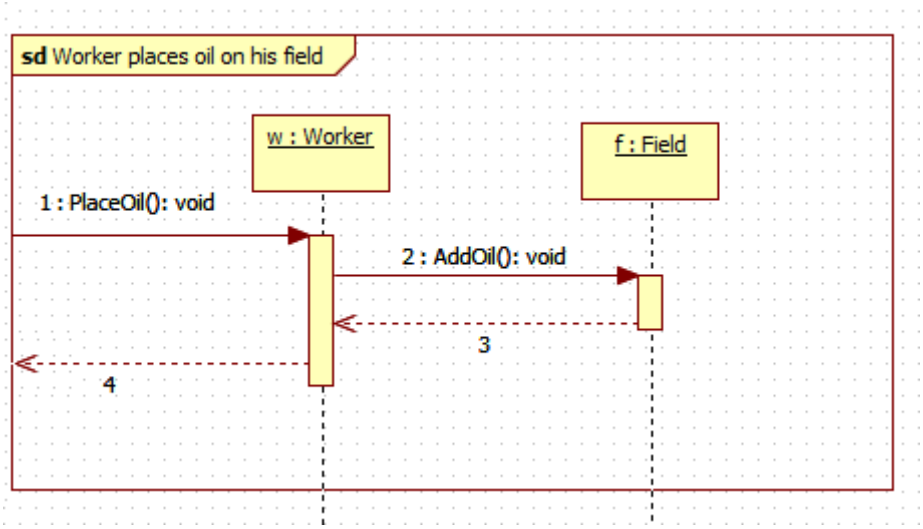
7.0.3.7 A láda eltolódik



7.0.3.12 Egyik ládát a másiknak tolják



A dolgozó mézet tesz az őt tároló mezőre



A dolgozó olajat tesz az őt tároló mezőre

Prototípus interface-definíciója

Az interfész általános leírása

A prototípus a program vázának, a szkeletonnak az implementációja, a grafikus felület használata nélkül. Ennek megfelelően a program karakteres, parancssoros irterfésszel rendelkezik.
A szabványos bemenetről fogadja a parancsokat, illetve a szabványos kimeneten jeleníti meg az esetleges kimenetet. Ezen kívül használhatunk fájlokat is, amelyekből a bemeneti parancsokat kiolvassuk, illetve az eredmény is fájlba kiíratható. Ezzel lehetővé tesszük az automatikus tesztelést, az előre elkészített tesztesetek felhasználásával. Egy teszteset tartalmazza a szükséges bemeneti parancsokat, valamint az elvárt eredményt.
A prototípust előre meghatározott parancsokkal használhatjuk, melyeknek paramétereik is lehetnek. Ezek a parancsok teszik lehetővé a játék irányítását. A pálya szerkezetét a konzolon karakter mátrixként jeleníthetjük meg.

Bemeneti nyelv

- **load**
  - **Leírás:** Egy pálya betöltése, amin irányíthatjuk a játékos(oka)t.
  - **Opciók:** A kiválasztott pálya (1 / 2 / …)
- **bind**
  - **Leírás: A mozgatni kívánt dolgozó kijelölése.**
  - **Opciók: A dolgozó “neve” (1 / 2 / …)**
- **move**
  - **Leírás: A dolgozó mozgatása.**
  - **Opciók: A mozgás iránya (U / D / L / R)**
- **drop**
  - **Leírás: A mező súrlódásának megváltoztatása.**
  - **Opciók: Méz(H) vagy olaj(O)**

- **show**
  - **Leírás: A pálya aktuális állapotát megjeleníti.**
  - **Opciók: -**
- **listFields**
  - **Leírás: A mezők kilistázása.**
  - **Opciók: -**
- **listWorkers**
  - **Leírás: A munkások kilistázása.**
  - **Opciók: -**
- **listChests**
  - **Leírás: A ládák kilistázása.**
  - **Opciók: -**
- **help**
  - **Leírás: A parancsok kilistázása a konzolra.**
  - **Opciók: -**
- **exit**
  - **Leírás: Kilépés a programból**
  - **Opciók: -**

Kimeneti nyelv

- **move**
  - **A játékos mozgatása miatt bekövetkezett változások kiírása a szabványos kimenetre szöveges formában.**
- **show**
  - **A pályán lévő mezők és a rajtuk lévő elemek elhelyezkedésének megjelenítése, 4 karakterben / mező:**
    <mező típusa (1) ><mezőn álló páyaelem / mező típusa (2) ><a mező súrlódását változtató dolog / a mező típusa(1)>
    Pl. FFFF (üres mező), FFFH (üres mézes mező), SP1O (olajos switch-en játékos), FC1F (üres mezőn láda)
- **listFields**
  - **Kilistázza az összes mezőt.**
    <típus><x kordináta>,<y koordináta>
- **listWorkers**
  - **Kilistázza a munkásokat.**
    <object name><x kordináta>,<y koordináta>
- **listChests**
  - **Kilistázza a ládákat.**
    <object name><x kordináta>,<y koordináta>
- **help**
  - **Kilistázz a parancsokat**
    <parancs neve> <parancs paraméterei> : <parancs fukciója>

Összes részletes use-case

Use-case neve	drop
Rövid leírás	Megváltoztatja az adott mező surlódását.
Aktorok	Tesztelő
Forgatókönyv	Amennyiben mézet tesz a mezőre, megnöveli a súrlódást, ellenkező esetben csökkenti.

Use-case neve	load
Rövid leírás	Betölt egy pályát.
Aktorok	Tesztelő
Forgatókönyv	A paraméterben megkapott pályát betölti, elhelyezi rajta a pályaelemeket, majd ettől fogva irányítható a játék a többi paranccsal.

Use-case neve	move
Rövid leírás	A dolgozó mozgatása meghatározott irányba.
Aktorok	Tesztelő
Forgatókönyv	Mozgatja a dolgozót a következők szerint: a. A mező üres, a dolgozó az adott mezőre lép. b. A mezőn fal, vagy mozdíthatatlan láda áll, a dolgozó nem tud lépni az adott mezőre. c. A mezőn mozdítható láda áll,a dolgozó eltolja a ládát az adott irányba. d. A mezőn nyitott állapotban lévő lyuk található, a dolgozó meghalt.

Use-case neve	listFields
Rövid leírás	Kilistázza a mezőket.
Aktorok	Tesztelő

Forgatókönyv	Kiírja a játékban lévő mezőket a kimenetre.

Use-case neve	listWorkers
Rövid leírás	Kilistázza a dolgozókat
Aktorok	Tesztelő
Forgatókönyv	Kiírja a kimenetre a játékban lévő dolgozókat

Use-case neve	listChests
Rövid leírás	Kilistázza a ládákat.
Aktorok	Tesztelő
Forgatókönyv	Kiírja a kimenetre a játékban lévő ládákat.

Use-case neve	bind
Rövid leírás	Kijelöli a mozgatni kívánt dolgozót.
Aktorok	Tesztelő
Forgatókönyv	Kijelöli a mozgatni kívánt dolgozót.

Tesztelési terv

Teszt-eset neve	Dolgozó mozgatása (üres mezőre)
Rövid leírás	Dolgozó mozgásának tesztelése.

Teszt célja	Teszteli, hogy a dolgozó tud-e üres mezőre lépni.

Teszt-eset neve	Dolgozó ládát tol
Rövid leírás	A Dolgozó ládatolásának tesztelése.
Teszt célja	Teszteli, hogy a dolgozó el tudja-e tolni a ládát.

Teszt-eset neve	Dolgozó dolgozónak megy
Rövid leírás	Dolgozó dolgozónak menésének tesztelése.
Teszt célja	Teszteli, hogy egy dolgozó nem tolhat el egy másik dolgozót.

Teszt-eset neve	Láda üres mezőre tolása
Rövid leírás	A láda mozgatásának tesztelése.
Teszt célja	Teszteli, hogy a dolgozó adott irányba képes-e üres mezőre tolni a ládát.

Teszt-eset neve	Láda kapcsolóra tolása
Rövid leírás	A kapcsoló működésének tesztelése.
Teszt célja	Teszteli, hogy amikor a dolgozó egy kapcsolót tartalmazó mezőre tol egy ládát, akkor a kapcsolóhoz tartozó lyuk aktiválódik-e.

Teszt-eset neve	Láda letolása kapcsolóról
Rövid leírás	A kapcsoló működésének tesztelése.
Teszt célja	Teszteli, hogy amikor a dolgozó egy kapcsolót tartalmazó mezőről letol egy ládát, akkor a kapcsolóhoz tartozó lyuk újra sima mezőként viselkedik-e..

Teszt-eset neve	Láda falnak tolása
Rövid leírás	A láda mozgatásának tesztelése.
Teszt célja	Teszteli, hogy a láda falnak tolásakor mi történik.

Teszt-eset neve	Láda nyelőre tolása
Rövid leírás	A láda nyelőre tolásának tesztelése.
Teszt célja	Teszteli, hogy a láda nyelőre tolásakor mi történik, magának a nyelőnek a tesztelése.

Teszt-eset neve	Láda ládának tolása
Rövid leírás	Láda ládának tolásának tesztelése.
Teszt célja	Teszteli, hogy ha láda ládát tol, mi történik.

Teszt-eset neve	Mező súrlódásának növelése
Rövid leírás	Méz helyezése a mezőre.
Teszt célja	Teszteli, hogy a mézzel növelhető a súrlódás

Teszt-eset neve	Mező súrlódásának csökkentése
Rövid leírás	Olaj helyezése a mezőre.
Teszt célja	Teszteli, hogy az olajjal csökkentető a súrlódás

Teszt-eset neve	Kapcsolható lyuk elnyel egy pályaelemet
Rövid leírás	Egy pályaelem (doboz/játékos) alatt kinyílik egy kapcsolható lyuk.
Teszt célja	A switch ténylegesen aktiválja-e a hozzárendelt lyukat

Teszt-eset neve	A maximálsian tolható dobozok eltolása sima mezőn
Rövid leírás	Megnézi, hogy ténylegesen odébbtolható-e a specifikált számú láda sima mezőkön.
Teszt célja	Az alap mező súrlódásának tesztelése.

Teszt-eset neve	A maximálisan tolható dobozok eltolása olajos mezőn
Rövid leírás	A sima mezőkön eltolható ládák száma megnő azzal, hogyha valamelyik láda/ládák olajos mezőn állnak.
Teszt célja	Valóban csökkenti-e a súrlódást a mezőre helyezett olaj.

Tesztelést támogató segéd- és fordítóprogramok specifikálása

A program determinisztikus működése miatt segédprogram használata lehetséges és ajánlott. A segédprogram teszt bemeneteket tárol, hozzájuk rendelve a várt kimenettel. A segédprogram lefuttatja a megadott tesztet (vagy többet), és összeveti a kimenetüket az elvárt kimenettel.. Amennyiben a kimenet nem egyezik meg az elvárt kimenettel, úgy a teszt nem sikerült, ellenkező esetben a teszt sikeres. Hiba esetén a program kijelzi, hogy hol, és milyen eltérést talált.

Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.03.23	1 óra	Majkut	7.0
2018.03.24	3 óra	Nagy, Szövő	7.1.2. 7.2
2018.03.24	1 óra	Nagy, Simon, Szövő	7.3

2018.03.24	1 óra	Németh, Szövő	7.1.3
2018.03.24	2 óra	Majkut	7.1, 7.3
2018.03.25	1 óra	Németh	7.4
2018.03.25	1 óra	Nagy	7.1.1, Teljes dokumentum szerkesztése

Részletes tervek

/8. házi feladat/

1 – a_csapat

Konzulens:

Belákovics Ádám

Csapattagok:

Név
Simon Márta
Németh Marcell
Szövő Roland

Nagy Szabina
Majkut Kristóf

2018.04.08.

8. Részletes tervek

Osztályok és metódusok tervei

Chest

- ***Felelősség***

Ládaként funkcionáló mozdítható pályaelemet megvalósító osztály. A Worker-ek el tudják tolni.

- ***Ősosztályok***
  - Element
- ***Metódusok***
  - **+ Smashed(): bool**: False-szal tér vissza, hiszen a láda nem nyomható össze.
  - **+ Stuck(): void**: Szól a Storehouse-nak, hogy egy újabb láda beragadt.

Direction

- ***Felelősség***

Az irányokat nyilvántartó enumeráció.

- ***Attribútumok***
  - **UP:** Felfelé irány.
  - **DOWN:** Lefelé irány.
  - **RIGHT:** Jobbra irány.
  - **LEFT:** Balra irány.

Element

- ***Felelősség***

Mozdítható pályaelemek absztrakt ősosztálya.

- ***Attribútumok***
  - **# storeHouse: StoreHouse**: Ha meghal egy játékos vagy chest megsemmisül ezen keresztül hívódik a PlayerLost()/ChestLost() függvény.
  - **# ownField: Field**: Tárolja, hogy az element melyik mezőn áll.
- ***Metódusok***

- **+ GetPosition():Field :** Visszatér azzal a Field-el amelyiken az element áll.
- **+ RegisterNewField(Field f):void :** Ha az adott pályaelem egy új mezőre kerül, a mező ezzel tudja beregisztrálni magát az Elementbe..
- **+ RegisterStorehouse(Storehouse sH):void:** Ha az adott pályaelem egy új mezőre kerül, a mező ezzel tudja beregisztrálni magát az Elementbe.
- **+ HitBy(Element e, Direction d):void:** Összeütköztet magával egy másik pályaelemet. Alapesetben, azaz ha az ütköztető pályaelem nem egy Worker, egyszerűen felteszi a saját mezőjére az átvett Element-et.
- **+ Stuck(): :** Figyelmezteti a pályaelemet, hogy beragadós mezőre lépett.
- **+ Smashed(): abstract boolean** : Amennyiben az adott pályaelemet nem tudta továbbtenni a mező, a rajta álló pályaelemnek ezt a metódusát hívja, ahol megnézi, hogy összenyomható-e az adott Element
- **+ Swallowed(): abstract void :**Akkor hívjuk meg, ha az elem lyukra vagy nyelőre kerül.

Field

- ***Felelősségek***

A pályát alkotó mezők ősosztálya.

- ***Attribútumok***
  - **- sticky: boolean**: Jelzi, hogy a mezőre tolva egy ládát onnan elmozdítható lesz-e még.
  - **# element: Element**: Az éppen a mezőn álló Element.
  - **- storingChest: bool**: Tárolja, hogy éppen van-e rajta láda.
  - **- friction: int**: A mező surlódása, ilyen nehéz az adott mezőről továbbtolni egy ládát.
- ***Metódusok***
  - **+ GetElement(): Element:** Visszatér a rajta álló pályaelemmel
  - **+ Accept(Element e, Direction d, int pushingPower): bool:** Egy láda csak akkor tolható tovább, ha az őt érő tolóerő nagyobb, mint a ládát tartalmazó mező surlódása, ezért az Accept() metódus is megkapja paraméterként a tolóerőt.
  - **+ Transfer(Direction d, int pushingPower): void:** A transfer függvény meghívásakor a Worker most már azt is továbbadja paraméterként, hogy mekkora erővel tolja meg az előtte lévő ládát.
  - **+ AlertNeighbour(f: Field, d: Direction): void:**Ha egy ládát tolnak a mezőre, és a sticky attribútuma igaz, értesíti a körülötte lévőket, hogy kvázi már ő is falnak számít.
  - **+ AddNeighbour(f: Field, d: Direction): void:** Hozzáadja a szomszédot a szomszédokat tároló tömbhöz
  - **+ WeighedDown(): void:** A ládával történő kézfogáskor hívódó függvény.
  - **+ AddHoney(): void:** A mező súrlódását (int friction) növeli meg egy adott értékkel.
  - **+ AddOil(): void:** A mező surlódását (int friction) csökkenti egy adott értékkel.

Goal

- ***Felelősség***

Célmezőként, nyelőként funkcionáló mezőt megvalósító osztály. Ha egy Worker rátol egy Chest-et, akkor a pontszáma megnő eggyel, és a Chest lekerül a pályáról.

- ***Ősosztályok***
  - Field
- ***Attribútumok***
  - **- score: int:** Azon játékos pontszáma, akinek a célja ebbe a nyelőbe eljuttatni a dobozokat.
- ***Metódusok***
  - **+ GetScore(): int:** Adott játékos pontszámát adja vissza.
  - **+ WeighedDown(): void:** Megnöveli a score attribútumot eggyel.

Hole

- ***Felelősség***

Lyukként funkcionáló mezőt megvalósító osztály. Ha rálép egy Element, akkor az le fog esni a pályáról. Kapcsolóval nyitható és csukható.

- ***Ősosztályok***
  - Field
- ***Attribútumok***
  - **- switchedOn: boolean:** Tárolja, hogy a hozzá tartozó kapcsoló fel van-e kapcsolva.
- ***Metódusok***
  - **+ Switch(): void:** A switchedOn attribútum értékét negálja.
  - **+ Accept(Element e, Direction d): boolean:** A lyukba tolt láda elfogadása, eltüntetése.

Storehouse

- ***Felelősségek***

Nyilvántartja a mezőket, a mozdítható ládák és az aktuális játékosok számát. A játékbeli események vezérléséért felelős osztály.

- ***Attribútumok***
  - **- map: Field[][]:** A mezőket tároló két dimenziós Field tömb.
  - **- moveableChests: int:** Tárolja a mozdítható ládák számát.
  - **- playersAlive: int:** Tárolja, hogy hány játékos van a játékban.
  - **- goals: Goal[]:** A kihelyezett nyelők tömbje.
- ***Metódusok***
  - **+ ChestLost():void:** Csökkenti a mozdítható ládák számát.
  - **+ PlayerLost():void:** Csökkenti a játékban lévő játékosok számát. Ha eléri ez az 1-et akkor meghívja a GameOver() függvényt.
  - **+ GameOver():void :** Befejezi a játékot és kiírja a győztest.
  - **+ StartGame():void :** Elindítja a játékot, felépíti a pályát.
  - **+ SetUpStoreHouse():void:** Elhelyezi a pályán a mezőket a megfelelő elrendezésben.
  - **+ void RegisterFieldNeighbours():** A pályán való elhelyezkedés alapján beregisztrálja az összes mezőhöz annak szomszédait.

Switch

- ***Felelősség***

Kapcsolóként funkcionáló mezőt megvalósító osztály. Ha rátolunk egy Chest-et, akkor kinyitja a hozzá tartozó Hole-t. Ha Worker lép rá, akkor nem történik semmi.

- ***Ősosztályok***
  - Field
- ***Attribútumok***
  - **+ holeToSwitch: Hole**: Az általa kapcsolt lyuk.
- ***Metódusok***
  - **+ Accept(Element e, Direction d): bool**: Megállapítja, hogy az adott Element ráléphet-e.
  - **+ WeighedDown(): void**: Ezt a függvényt csak láda hívja, emiatt úgy definiáljuk felül, hogy ez meghívja a hozzá tartozó lyuk Switch() függvényét.

Wall

- ***Felelősség***

Pálya szélét határoló, valamint falként (oszlopként) funkcionáló mezőt megvalósító osztály. Nem léphet rá semmilyen Element (nem tolható rá láda se).

- ***Ősosztályok***
  - Field
- ***Metódusok***
  - **+ Accept(Element e, Direction d): bool**: Mindig false-szal tér vissza, mást nem csinál ,mert nem enged magára léptetni semmiylen Element példányt.

Worker

- ***Felelősség***

Dolgozót, mint mozdítható pályaelemet megvalósító osztály. Képes a Chest-ek eltolására.

- ***Ősosztályok***
  - Element
- ***Attribútumok***
  - **- moving: boolean:** Jelzi, hogy éppen a dolgozó lép-e, vagy pedig csak tolják őt
  - **- pushingPower: int:** Az az erő, amellyel megtolja az előtte lévő ládát
- ***Metódusok***
  - **+ Smashed(): boolean:** Annak érdekében, hogy ne haljon meg a dolgozó amikor csak simán falnak megy, figyeli a moving értékét, és amennyiben false, azaz a dolgozó éppen megtolták, igazzal tér vissza, mivel a dolgozó összenyomható. Végül meghívja a Storehouse PlayerLost() függvényét.
  - **+ Move(Direction d): void :** Külső input esetén ez a függvény hívodik meg a dolgozó mozgatására.
  - **+ HitBy(Element e): void :** Amennyiben meghívódik, semmit nem csinál, hiszen ezt csak egy másik Worker Transfer(Direction d) függvénye hívhatta, ezzel érhető el, hogy a két munkás ne tudja eltolni egymást.
  - **+ PlaceHoney(): void:** Mézet tesz a éppen őt tartalmazó mezőre, ezzel megnövelve annak a surlódását.
  - **+ PlaceOil(): void:** Olajat tesz a éppen őt tartalmazó mezőre, ezzel csökkentve annak a surlódását.

A tesztek részletes tervei, leírásuk a teszt nyelvén

Teszteset1

- **Leírás**

A tesztpálya egy “rövid folyosó”(, csak jobbra balra lehet benne mozogni), rajta két dolgozó van, illetve egy darab láda. A dolgozó először nekimegy a másiknak, és megpróbálja eltolni, majd a másik irányba indulva az egyetlen ládát nekitolja a falnak, ami így beragad, és a játék véget ér.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszt ellenőrzi, hogy játékos valóban nem tud-e tolni játékost, valamint azt, hogy egy láda beragad, és az volt az utolsó mozdítható, tényleg véget ér-e a játék.

- **Bemenet**

moveL
moveR

- **Elvárt kimenet**

WWW FFFF G**P1**G GGGG F**P2**F F**C1**F WWWW

Teszteset2

- **Leírás**

A tesztpálya egy “rövid folyosó”(, csak jobbra balra lehet benne mozogni), rajta két dolgozó van, illetve egy darab láda. Az egyik játékos egy kapcsolható lyukon áll. A másik játékos egy ládát tol a hozzá tartozó kapcsolóra, aminek hatására a játékos lezuhan és a játék véget ér.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszt ellenőrzi, hogy működik-e a kapcsolható lyuk, illetve azt, hogy vége van-e a játéknak, amennyiben csak egy játékos marad a pályán.

- **Bemenet**

moveR

- **Elvárt kimenet**

WWWW IIII GGGG GGGG F**P2**F S**C1**S WWWW

Teszteset3

- **Leírás**

A tesztpálya egy “rövid folyosó”(, csak jobbra balra lehet benne mozogni), rajta két dolgozó van, illetve egy darab láda. Az egyik dolgozó egy nyelőre tolja a ládát, aminek következtében megnő a pontja, és a játék véget ér több mozdítható láda hiányában.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszt ellenőrzi, hogy a tolható ládák hiányában véget ér-e a játék, hogy a nyelő valóban elnyeli-e a ládát, és hogy a megfelelő játékost kiáltja-e ki győztesnek.

- **Bemenet**

moveR

- **Elvárt kimenet**

WWWW G**P1**G FFFF F**P2**F GGGG WWWW

Teszteset4

- **Leírás**

A tesztpálya egy “rövid folyosó”(, csak jobbra balra lehet benne mozogni), rajta két dolgozó van, illetve egy darab láda. Az egyik dolgozó a ládával a falhoz nyomja a másikat, aki így meghal, és a játék véget ér.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszt ellenőrzi, hogy tolhatóak-e a ládák, hogy valóban összenyomható-e a dolgozó, illetve azt, hogy amennyiben csak egy játékos maradt, vége van a játéknak.

- **Bemenet**

moveR

- **Elvárt kimenet**

WWWW GGGG GGGG F**P1**F F**C1**F WWWW

Teszteset5

- **Leírás**

A tesztpálya egy “folyosó”(, csak jobbra-balra lehet benne mozogni), rajta egy dolgozó van, illetve az egyik irányba négy láda négy tiszta mezőn, a másik irányba kettő két mézes mezőn és egy nem mézesen. A négy üres mezőn állót eltolja a dolgozó, majd a két mézes mezőn állót is. Ekkor három láda kerül egymás mellé, kettő mézes mezőn, egy pedig simán. Ezeket is megpróbálja eltolni a dolgozó, de ezt már nem tudja.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszt ellenőrzi, hogy a dolgozó valóban eltudja-e tolni azt a specifikációban leírt számú ládát, illetve, hogy a méz megnöveli-e a súrlódást.

- **Bemenet**

moveL
moveR
moveR
moveR

- **Elvárt kimenet**

WWWW F**C1**F F**C2**F F**C3**F F**C4**F FFFF GGGG F**P1**M F**C5**M F**C6**M F**C7**F FFFF WWWW

Teszteset6

- **Leírás**

A tesztpálya egy “folyosó”(, csak jobbra-balra lehet benne mozogni), rajta egy dolgozó van, illetve az egyik irányba öt láda öt tiszta mezőn, a másik irányba öt láda egy mézes, és négy olajos mezőn. Az öt sima mezőről a dolgozó már nem tudja eltolni az öt ládát, viszont a másik irányba el tudja tolni az öt ládát úgy is, hogy az egyik mézesen van.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszt ellenőrzi, hogy a dolgozó valóban csak a specifikációban leírt számú ládát tudja-e eltolni, illetve, hogy az olaj csökkenti-e a súrlódást.

- **Bemenet**

moveL
moveR

- **Elvárt kimenet**

WWWW FFFF F**C1**F F**C2**F F**C3**F F**C4**F F**C5**F GGGG F**P1**M F**C6**O F**C7**O F**C8**O F**C9**O F**C0**F WWWW

Teszteset7

- **Leírás**

A tesztpálya egy “folyosó”(, csak jobbra-balra lehet benne mozogni), három játékossal és öt ládával. A felállás a következő: C1 P1 C2 P2 C3 C4 C5 P3 G G, ahol C1 egy kapcsolható lyukon áll, P3 pedig egy kapcsolón, ami mézes. P2 először jobbra tolja a három ládát, aminek hatására bekapcsol a lyuk és elnyeli C1-et. Ezután a dolgozó megpróbálja odébbtolni a ládákat, de azok nem mennek odébb a mézes mező miatt. Itt váltunk P1-re, aki rátol egy ládát P2-re, és megöli. Ezután belesétál a lyukba, és vége a játéknak, hiszen egy játékos maradt.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszt ellenőrzi, hogy jól működik-e a súrlódás és a kapcsoló, elnyeli-e a ládát a kapcsolható lyuk, mozgatható-e a többi játékos is.

- **Bemenet**

moveR
moveR
bindP2
moveR
moveR
moveL
moveL
moveL

- **Elvárt kimenet**

WWWW IIII GGGG FFFF FFFF F**C2**F F**C3**F F**C4**F S**C5**M G**P3**G GGGG WWWW

Teszteset8

- **Leírás**

[szöveges leírás, kb. 1-5 mondat.]

- **Ellenőrzött funkcionalitás, várható hibahelyek**
- **Bemenet**

[a proto bemeneti nyelvén megadva (lásd előző anyag)]

- **Elvárt kimenet**

[a proto kimeneti nyelvén megadva (lásd előző anyag)]

A tesztelést támogató programok tervei

A prototípus indításakor egy főmenüből van lehetőségünk kiválasztani, hogy az előre elkészített teszteseteket akarjuk-e futtatni, vagy saját kezűleg akarjuk-e tesztelni az egyes funkciókat.

Amennyiben egy megírt tesztesetet választunk, a programunk beolvassa a hozzá tartozó .txt fájlt. Ez tartalmazza annak a pályának a leírását, amin a tesztesetet végighajtjuk, és a tesztben használt parancsokat. A tesztekhez tartozó szövegfájlok a következőképpen épülnek fel:

- Az első két szám (x és y) határozza meg a pálya méretét, azaz, hogy mekkora lesz a Field-ekből álló kétdimenziós tömbünk
- A harmadik szám (w) a játékosok (, és ezzel együtt a nyelők) számát határozza meg
- A negyedik szám (s) a kapcsolható lyukak és a hozzájuk tartozó kapcsolók számát határozza meg
- Az ötödik szám (c) a tesztben kiadott parancsok számát határozza meg
- A hatodik szám (l) a pályára felkerülő ládák száma

  --> Ezeket meghatározva rendelkezésünkre áll a pálya felépítéséhez szükséges összes adat, és létrehozza a program a szükséges tömböket

  - Field[x][y] map : maga a pálya
  - string[x][y] expectedOutput: a teszt végén az elvárt felállás a pályán
  - Worker[w] players, illetve Goal[w] goals
  - Switch[s] switches, illetve Hole[s] holesToSwitch
  - string[c] commands: a kiadandó parancsok a teszt során

- Ezután következik a x*y darab mező leírásának beolvasása. A mezőket 4 számjegy reprezentálja, a következők szerint:
  - Az első számjegy határozza meg a Field típusát:
    0 – Field , 1 – Goal, 2 – Switch, 3 – Hole, 4 – Wall
  - A második számjegy határozza meg a mezőn álló pályaelem típusát:
    0 – üres, 1 – Worker, 2 – Chest
  - A harmadik számjegy határozza meg a mező súrlódását:
    0 – tiszta, 1 – mézes, 2 – olajos
  - A negyedik mező egy opcionális paraméter
    Alapértelmezetten 0, ha nem használják. A Goal-ok esetén azt jelzi, hogy hanyadik játékoshoz tartozik (1,2,3,…), Switch-ek és Hole-ok esetén hasonlóan, az azonos számúak tartoznak össze

- Ezek alapján felépíti a program a pályát, elhelyezi a pályaelemeket a megfelelő mezőkön, összeköti a kapcsolókat a hozzájuk tartozó lyukakkal
- Ezután a .txt fájl tartalmazza a kiadandó parancsokat
- Végül pedig az elvárt kimenetet, string formátumban

Az így felépített pályán végrehajtja a program a beolvasott parancsokat egymás után, majd végigmegy a pálya összes mezőjén, lekérdezi azoknak tartalmát, és összehasonlítja a fájlból olvasott expectedOutput megfelelő elemével. A teszt sikerességét pedig közli a tesztelővel. A Field-ek string formátuma – az előző dokumentációban leírtak szerint- <mező típusa (1 char)><pályaelem neve / ha üres, a mező típusa kétszer (2 char)><a mező súrlódását változtató anyag / ha tiszta, a mező típusa (1 char)>.

Amennyiben a sajátkezű tesztelést választjuk, egy előre definiált pályán próbálgathatjuk végig a játék egyes funkcióit, a megfelelő parancsok kiadásával.

Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.04.08.	2-3 óra	Majkut, Szövő, Németh, Nagy, Simon	8.1, 8.2, 8.3

Prototípus beadása

/10. házi feladat/

1 – a_csapat

Konzulens:

Belákovics Ádám

Csapattagok:

Név
Simon Márta
Németh Marcell

Szövő Roland
Nagy Szabina
Majkut Kristóf

2018.04.22.

10. Prototípus beadása

Fordítási és futtatási útmutató

Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Chest.java		2018.04.22	Láda osztály
Direction.java		2018.04.22	Direction Enumeráció
Element.java		2018.04.22	Element osztály
Field.java		2018.04.22	Field osztály
Goal.java		2018.04.22	Goal osztály
Hole.java		2018.04.22	Hole osztály
UI.java		2018.04.22	User Interface
Wall.java		2018.04.22	Wall osztály
Worker.java		2018.04.22	Worker osztály

Storehouse.java		2018.04.22	Storehouse osztály
Switch.java		2018.04.22	Switch osztály

Fordítás

```
@ECHO OFF

set PATH=%PATH%;C:\Program Files\Java\jdk1.8.0_121\bin
set JAVA_HOME=C:\Program Files\Java\jdk1.8.0_121
set CLASSPATH=%CLASSPATH%;

javac -d bin src/src/*.java
```

Futtatás

```
cd bin
java src.UI

cd ..
```

Tesztek jegyzőkönyvei

Teszteset1

Tesztelő neve	Nagy Szabina
Teszt időpontja	2018.04.22

Teszteset2

Tesztelő neve	Nagy Szabina
Teszt időpontja	2018.04.22

Teszteset3

Tesztelő neve	Szövő Roland
Teszt időpontja	2018.04.22

Teszteset4

Tesztelő neve	Németh Marcell
Teszt időpontja	2018.04.22

Teszteset5

Tesztelő neve	Németh Marcell
Teszt időpontja	2018.04.22

Teszteset6

Tesztelő neve	Németh Marcell
Teszt időpontja	2018.04.22

Teszteset7

Tesztelő neve	Szövő Roland
Teszt időpontja	2018.04.22

Teszteset8

Tesztelő neve	Szövő Roland

Teszt időpontja	2018.04.22

Értékelés

Tag neve	Munka százalékban
Nagy Szabina	20%
Németh Marcell	20%
Majkut Kristóf	20%
Simon Márta	20%
Szövő Roland	20%

Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.04.22	4 óra	Nagy, Németh, Szövő, Simon, Majkut	Tesztesetek megírása, tesztelése

Prototípus beadása

/11. házi feladat/

1 – a_csapat

Konzulens:

Belákovics Ádám

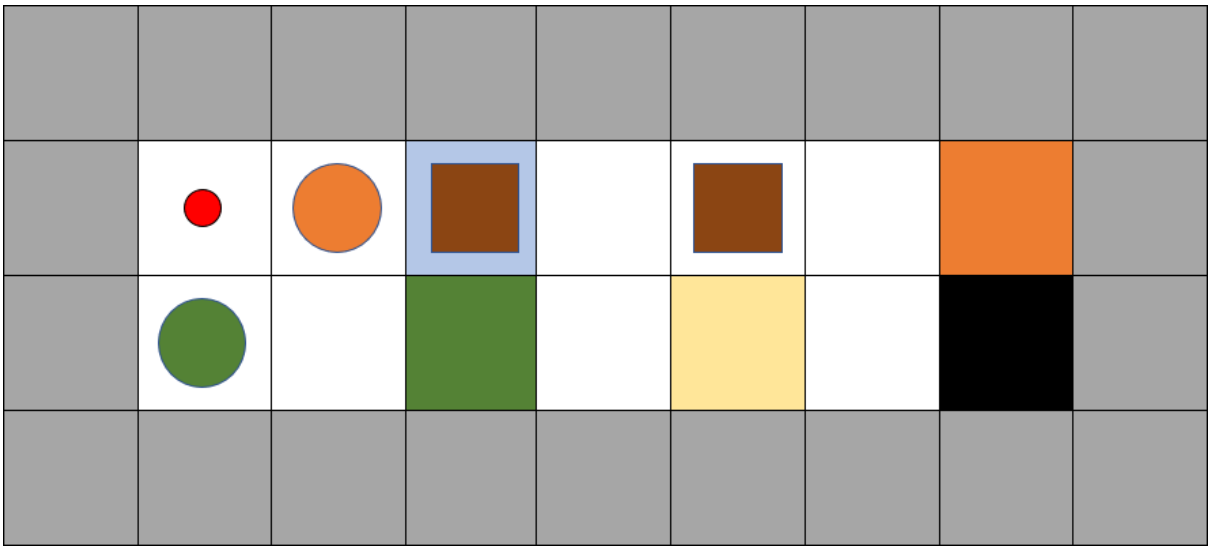
Csapattagok:

Név
Simon Márta
Németh Marcell
Szövő Roland
Nagy Szabina
Majkut Kristóf

2018.05.02.

11. Grafikus felület specifikációja

A grafikus interfész



A pályát alkotó mezőket a négyzetek ábrázolják:

- Field: fehér színű négyzet
- Hole: fekete színű négyzet
- Swicth: fehér színű négyzet egy kis piros körrel a közepén
- Goal: annak a játékosnak a színével megegyező, amelyiknek a pontjait számolja
- Wall: szürke színű négyzet
- Olajos/mézes mező: amennyiben egy sima mezőn, vagy egy kapcsolón mézet/olajat helyez el egy játékos, a mező színe sárga/kék lesz (egy lyukra értelemszerűen nem kerülhet semmi, nyelő esetén pedig lényegtelen annak súrlódása, hiszen úgyis elnyeli rögtön a ládát így ezek színe nem változik)

A mezőkön elhelyezhető pályaelemek:

- Worker: különböző színű körök ábrázolják
- Chest: barna színű kis négyzet

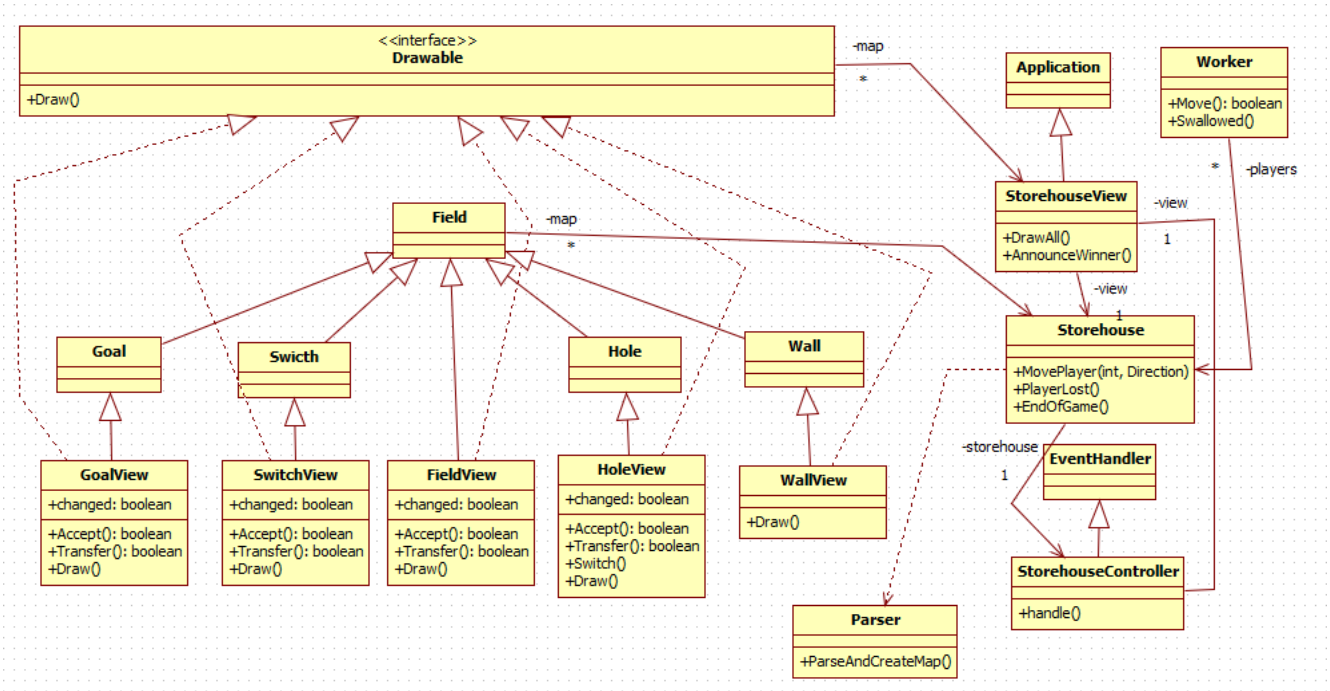
A játéknak alapvetően nincs semmiféle menürendszere, az Escape lenyomásával hívhatunk elő egy párbeszédablakot, amellyel újraindíthatjuk a játékot, vagy kiléphetünk a programból.

A grafikus rendszer architektúrája

A felület működési elve

Amikor a program meglévő prototípusához egy grafikus felület illesztettünk, a cél az MVC architektúra kialakítása volt. Ebből a modellünk már készen volt, a megjelenítésért, illetve az eseménykezelésért felelős osztályokra volt még szükség.
Ehhez bevezettünk öt új osztályt, egyet-egyet mindegyik mezőtípushoz, amelyek a kinézetükért felelnek. Az egyes xView objektumoknak a megfelelő mező leszármazottak az ősosztályai, így ezek hozzáférnek a Field protected tagváltozóihoz is. Továbbá, mindegyik megvalósítja a Drawable interfészt, aminek csupán egyetlen Draw() metódusa van. A közös interfésznek köszönhetően a StorehouseView egy heterogén kollekcióban tudja ezeket eltárolni.
A pályalelemekhez nincs külön View objektum, ezek kirajzolásáért az őt tároló “mezőView” felelős.
Az eseménykezelésért felelős EventHandler leszármazott, a StorehouseController fog reagálni a felhasználói inputokra, a billentyűk lenyomására. Az inputok az egyes dolgozók mozgatásához rendelt betűk, illetve az Escape.
Az értesítéséket tekintve a program kevert. A képernyő frissítésére akkor kerül sor, ha valamelyik játékos lép. Ekkora a StorehouseContorller értesíti a modellt, ami ennek megfelelően lépteti a játékosok egyikét, majd szól a StorehouseView-nak, hogy frissítse a pályát. Az egyes xView objektumok csak annyiban definiálják felül a Field Accept(), illetve Transfer() metódusait, hogy amennyiben az ős ezen függvényei igazzal térnek vissza, azaz a mezőre felkerül, vagy onnan lekerült egy pályaelem, akkor a changed flag-et igazra billenti. Ennek köszönhetően a StorehouseView mikor végigfut a Drawable objektumokat tároló kollekcióján, a legtöbb esetben nem is lesz szükség újrarajzolásra. A Storehouse-nak azért lesz szüksége a StorehouseView ismeretére, mert amennyiben a játék valamilyen okból véget ér, így tud jelezni a megjelenítésnek.

A felület osztály-struktúrája



A grafikus objektumok felsorolása

GoalView

- ***Felelősség***
  - A nyelő kirajzolása
- ***Ősosztályok***
  - Field -> Goal -> GoalView
- ***Interfészek***
  - Drawable
- ***Attribútumok***
  - **+boolean changed:** a legutóbbi rajzolás óta került-e fel rá/le róla pályaelem
- ***Metódusok***
  - **+ boolean Accept():** Amennyiben az ős Accept()-je igazzal tér vissza, igazba billenti a changed flaget
  - **+ boolean Transfer():** Amennyiben az ős Transfer()-je igazzal tér vissza, igazba billenti a changed flaget
  - **+ void Draw():** Felrajzol egy nyelőt a pályára

SwitchView

- ***Felelősség***
  - A kapcsoló kirajzolása
- ***Ősosztályok***
  - Field -> Switch
- ***Interfészek***
  - Drawable
- ***Attribútumok***
  - **+ boolena changed:** a legutóbbi rajzolás óta került-e fel rá/le róla pályaelem
- ***Metódusok***
  - **+ boolean Accept():** Amennyiben az ős Accept()-je igazzal tér vissza, igazba billenti a changed flaget
  - **+ boolean Transfer():** Amennyiben az ős Transfer()-je igazzal tér vissza, igazba billenti a changed flaget
  - **+ void Draw():** Felrajzol egy kapcsolót a pályára

FieldView

- ***Felelősség***
  - A mező kirajzolása
- ***Ősosztályok***
  - Field
- ***Interfészek***
  - Drawable
- ***Attribútumok***
  - **+ boolean changed:** a legutóbbi rajzolás óta került-e fel rá/le róla pályaelem
- ***Metódusok***
  - **+ boolean Accept():** Amennyiben az ős Accept()-je igazzal tér vissza, igazba billenti a changed flaget
  - **+ boolean Transfer():** Amennyiben az ős Transfer()-je igazzal tér vissza, igazba billenti a changed flaget
  - **+ void Draw():** Felrajzol egy mezőt a pályára

HoleView

- ***Felelősség***
  - A lyukak kirajzolása.
- ***Ősosztályok***
  - Field -> Hole
- ***Interfészek***
  - Drawable
- ***Attribútumok***
  - **- boolean changed:** a legutóbbi rajzolás óta került-e fel rá/le róla pályaelem, csak akkor érdekes, ha a lyuk ki van kapcsolva
- ***Metódusok***
  - **+ boolean Accept():** Ha a lyuk ki van kapcsolva, arra az esetre a sima mezővel azonos viselkedés
  - **+ boolean Transfer():** Ha a lyuk ki van kapcsolva, arra az esetre a sima mezővel azonos viselkedés
  - **+ void Draw():** Felrajzol egy lyukat a pályára

WallView

- ***Felelősség***
  - A falak kirajzolása.

· ***Ősosztályok***

- Field -> Wall -> WallView

· ***Interfészek***

- Drawable

· ***Metódusok***

- **+ void Draw():** Felrajzol egy falat a pályára.

Drawable

· ***Felelősség***

- Közös alapot biztosít a játékban az összes kirajzolható objektumnak.

· ***Metódusok***

- **+ void Draw():** Egy rajzoló metódus

StorehouseController

· ***Felelősség***

Kezeli azokat az eseményeket (billentyűleütés, kattintás), amelyek a raktárban lévő elemek pozícióját, viselkedését megváltoztatják.

· ***Ősosztályok***

EventHandler

· ***Attribútumok***

- **- StorehouseView view:** a pálya megjelenítéséért felelős osztály
- **- Storehouse storehouse:** a pálya modellje

· ***Metódusok***

- **+handle():** a felhasználói események kezelése

Storehouse

· ***Felelősség***

A raktár és elemeinek kezelője, a modell

· ***Attribútumok***

- **- Field[][] map:** a pályát alkotó kétdimenziós Field tömb
- **- Worker[] players:** a játékosok által irányított dolgozók tömbje

· ***Metódusok***

- **+MovePlayer(int,Direction):** a megfelelő sorszámú játékos mozgatása megadott irányba

StorehouseView

· ***Felelősség***

A grafikus megjelenítésért felelős osztály.

· ***Ősosztályok***

Application

· ***Attribútumok***

- **- Drawable[][] map**: a Drawable interfészt implementáló objektumokat tároló kétdimenziós tömb

· ***Metódusok***

- **+ DrawAll(): void**: végigszalad a map-en, és meghívja az összes elemének a Draw() metódusát
- **+ AnnounceWinner(): void:** egy felugró ablakban kihirdeti a játék győztesét

Application

· ***Felelősség***

JavaFX alkalmazások belépési pontjaként funkcionáló absztrakt osztály.

Worker

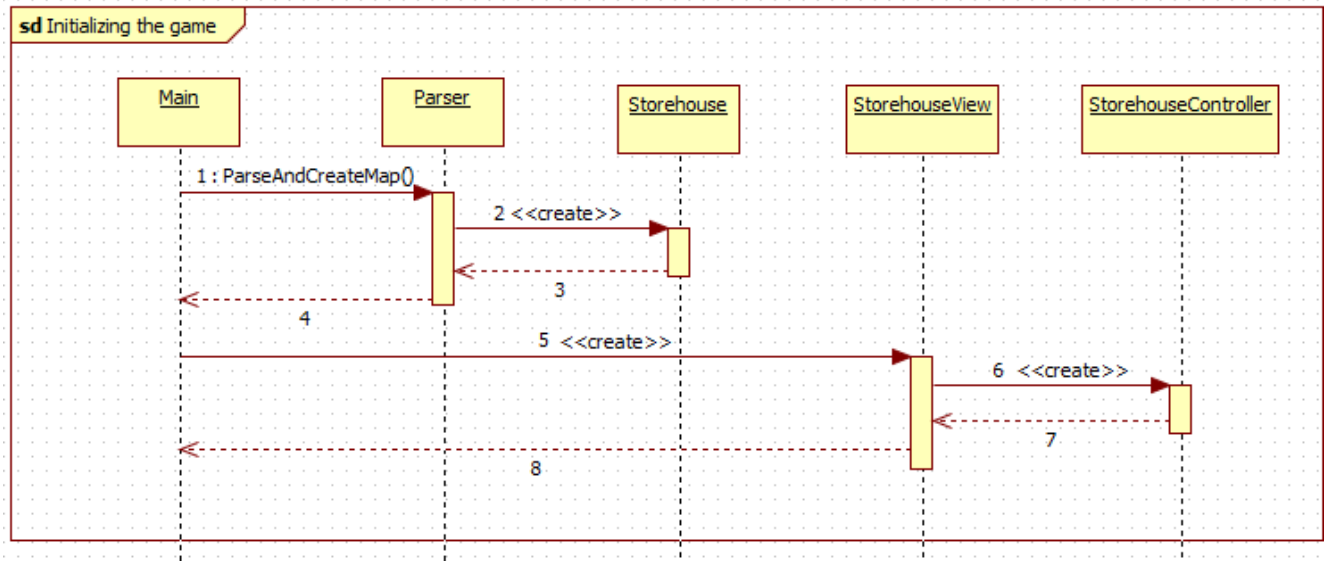
· ***Felelősség***

Dolgozót megvalósító osztály.

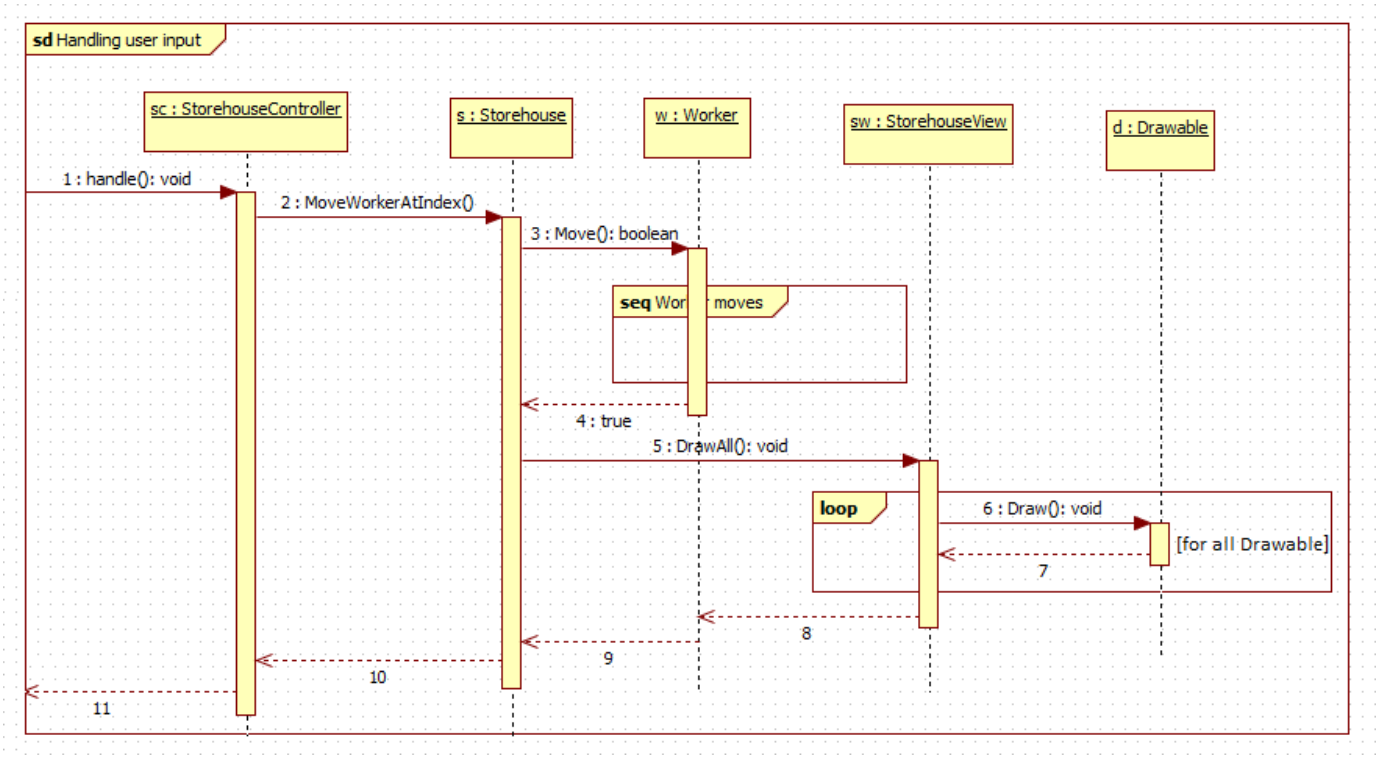
· ***Metódusok***

- **+ Swallowed(): abstract void:** Akkor hívódik meg, ha a dolgozó lyukra vagy nyelőre kerül.
- **+ Move(): boolean:** Dolgozó mozgatása.

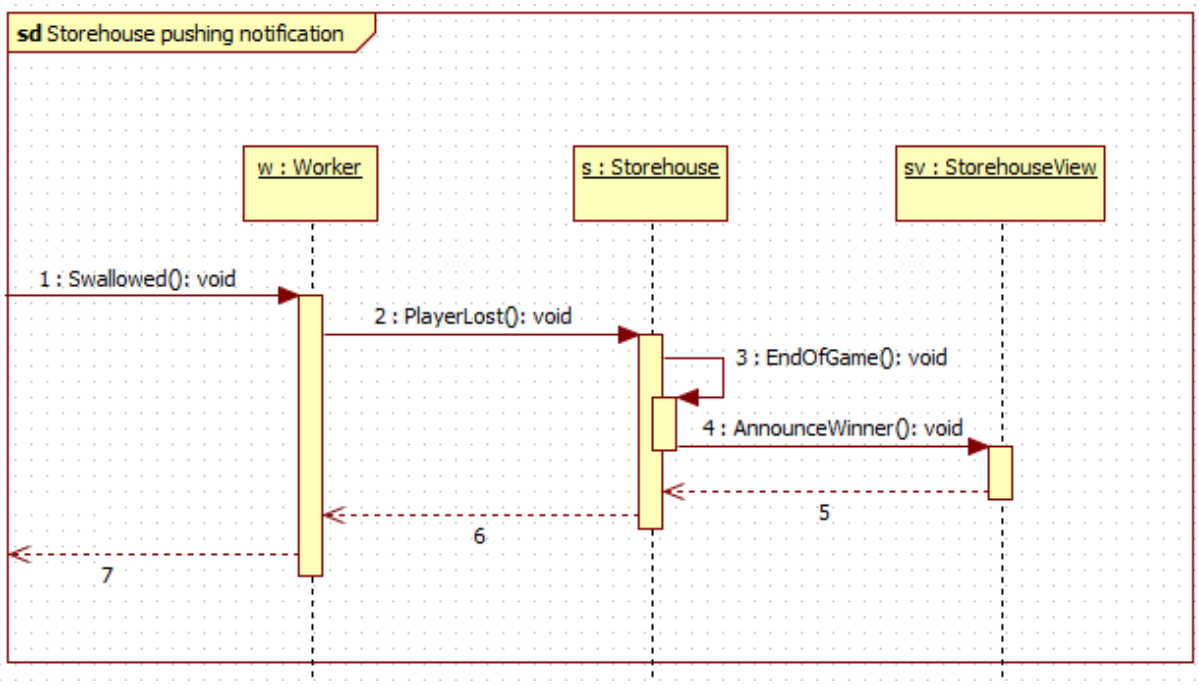
Kapcsolat az alkalmazói rendszerrel



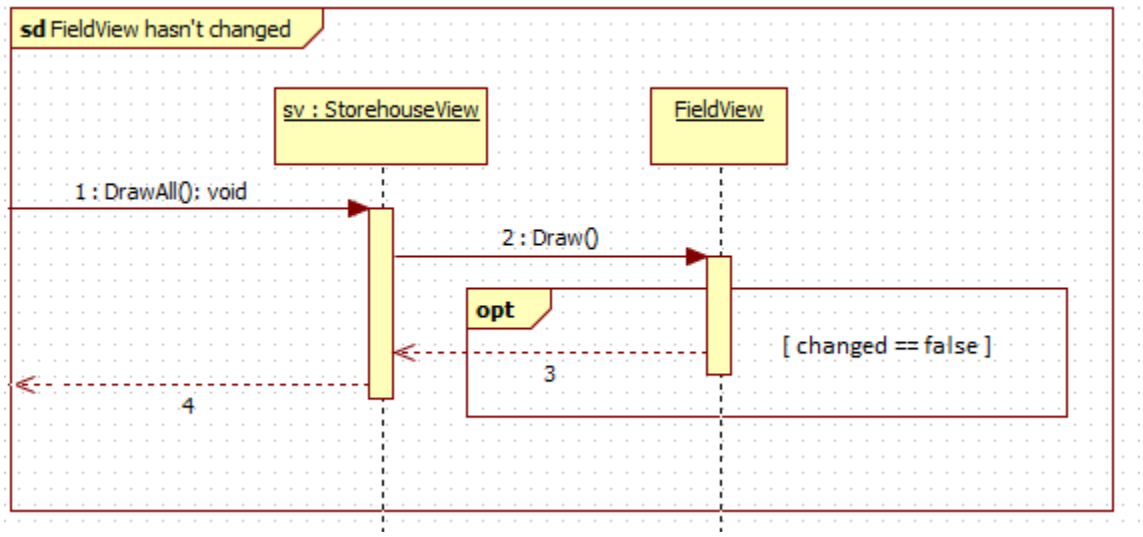
A játék indulása, az egyes komponensek létrehozása



A billentyűk lenyomásának kezelése



A Storehouse értesíti a View-t a játék végéről



11.4.4 Amennyiben a legutolsó rajzolás óra nem változott a mező, nem rajzolja újra magát

Napló

Kezdet	Időtartam	Résztvevők	Leírás

2018.04.27	5 óra	Németh, Simon, Nagy, Szövő, Majkut	Grafikus felület kitalálása 11.1 11.2
2018.04.28	1 óra	Németh, Simon, Nagy, Szövő	11.3
2018.04.28	1 óra	Majkut	11.4

Grafikus felület specifikációja

/11. házi feladat/

1 – a_csapat

Konzulens:

Belákovics Ádám

Csapattagok:

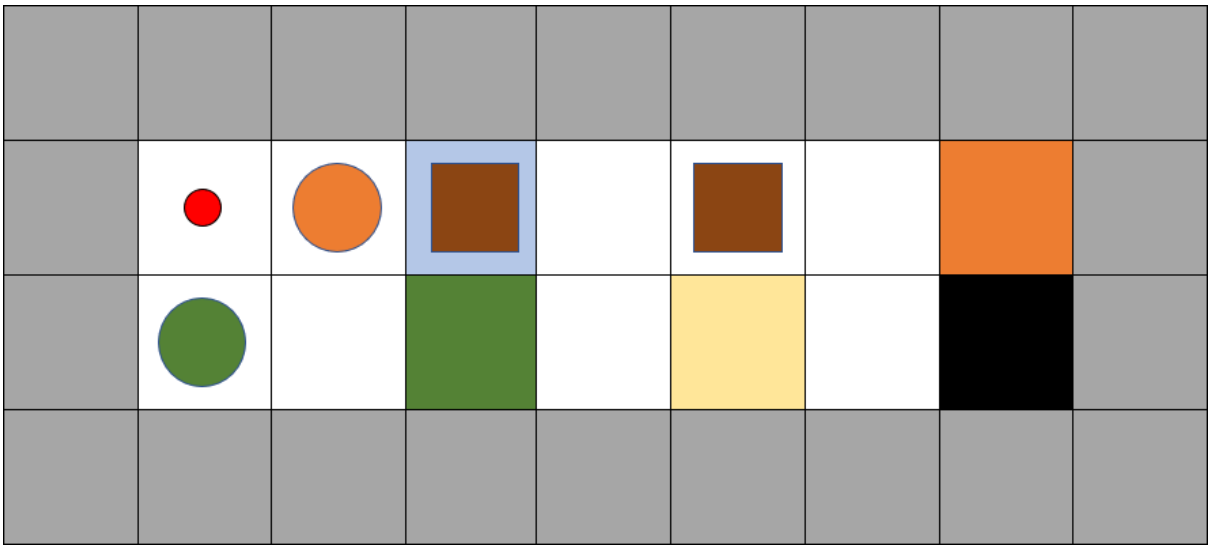
Név
Simon Márta

Németh Marcell
Szövő Roland
Nagy Szabina
Majkut Kristóf

2018.05.02.

11. Grafikus felület specifikációja

A grafikus interfész



A pályát alkotó mezőket a négyzetek ábrázolják:

- Field: fehér színű négyzet
- Hole: fekete színű négyzet
- Swicth: fehér színű négyzet egy kis piros körrel a közepén
- Goal: annak a játékosnak a színével megegyező, amelyiknek a pontjait számolja
- Wall: szürke színű négyzet
- Olajos/mézes mező: amennyiben egy sima mezőn, vagy egy kapcsolón mézet/olajat helyez el egy játékos, a mező színe sárga/kék lesz (egy lyukra értelemszerűen

nem kerülhet semmi, nyelő esetén pedig lényegtelen annak súrlódása, hiszen úgyis elnyeli rögtön a ládát így ezek színe nem változik)

A mezőkön elhelyezhető pályaelemek:

- Worker: különböző színű körök ábrázolják
- Chest: barna színű kis négyzet

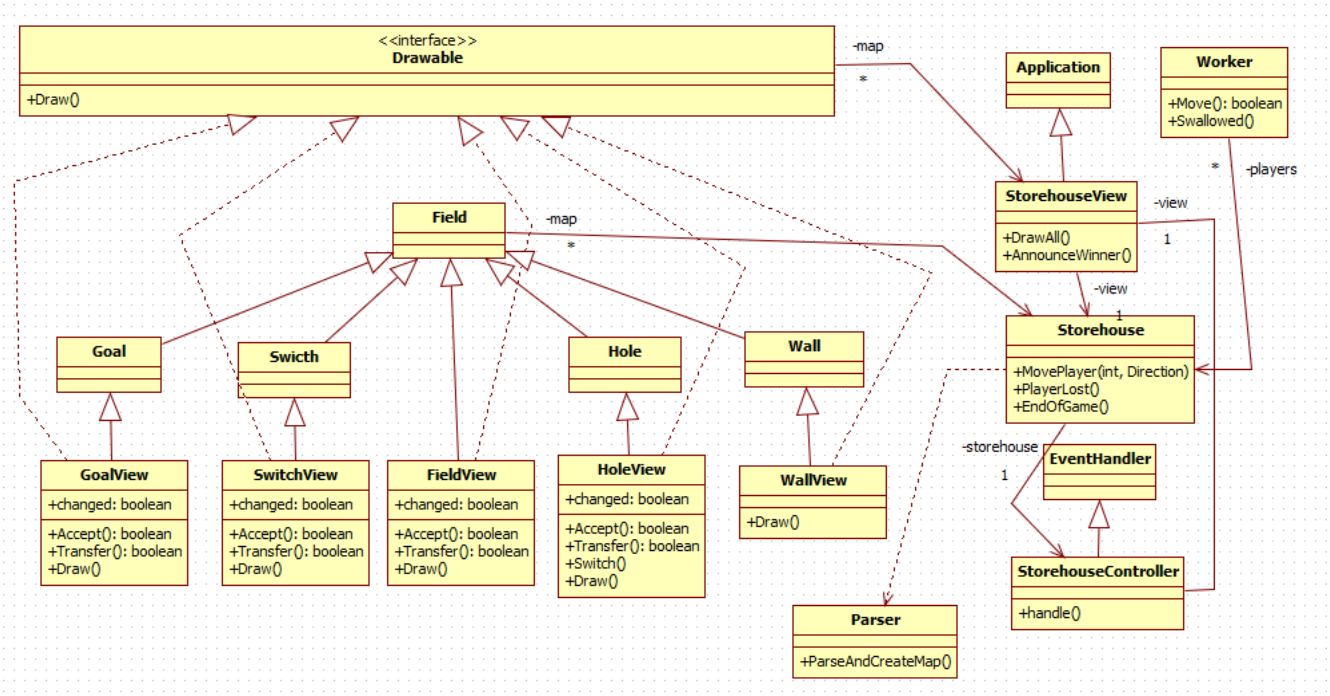
A játéknak alapvetően nincs semmiféle menürendszere, az Escape lenyomásával hívhatunk elő egy párbeszédablakot, amellyel újraindíthatjuk a játékot, vagy kiléphetünk a programból.

A grafikus rendszer architektúrája

A felület működési elve

Amikor a program meglévő prototípusához egy grafikus felület illesztettünk, a cél az MVC architektúra kialakítása volt. Ebből a modellünk már készen volt, a megjelenítésért, illetve az eseménykezelésért felelős osztályokra volt még szükség.
Ehhez bevezettünk öt új osztályt, egyet-egyet mindegyik mezőtípushoz, amelyek a kinézetükért felelnek. Az egyes xView objektumoknak a megfelelő mező leszármazottak az ősosztályai, így ezek hozzáférnek a Field protected tagváltozóihoz is. Továbbá, mindegyik megvalósítja a Drawable interfészt, aminek csupán egyetlen Draw() metódusa van. A közös interfésznek köszönhetően a StorehouseView egy heterogén kollekcióban tudja ezeket eltárolni.
A páyalelemekhez nincs külön View objektum, ezek kirajzolásáért az őt tároló “mezőView” felelős.
Az eseménykezelésért felelős EventHandler leszármazott, a StorehouseController fog reagálni a felhasználói inputokra, a billentyűk lenyomására. Az inputok az egyes dolgozók mozgatásához rendelt betűk, illetve az Escape.
Az értesítéséket tekintve a program kevert. A képernyő frissítésére akkor kerül sor, ha valamelyik játékos lép. Ekkora a StorehouseContorller értesíti a modellt, ami ennek megfelelően lépteti a játékosok egyikét, majd szól a StorehouseView-nak, hogy frissítse a pályát. Az egyes xView objektumok csak annyiban definiálják felül a Field Accept(), illetve Transfer() metódusait, hogy amennyiben az ős ezen függvényei igazzal térnek vissza, azaz a mezőre felkerül, vagy onnan lekerült egy pályaelem, akkor a changed flag-et igazra billenti. Ennek köszönhetően a StorehouseView mikor végigfut a Drawable objektumokat tároló kollekcióján, a legtöbb esetben nem is lesz szükség újrarajzolásra. A Storehouse-nak azért lesz szüksége a StorehouseView ismeretére, mert amennyiben a játék valamilyen okból véget ér, így tud jelezni a megjelenítésnek.

A felület osztály-struktúrája



A grafikus objektumok felsorolása

GoalView

- ***Felelősség***
  - A nyelő kirajzolása
- ***Ősosztályok***
  - Field -> Goal -> GoalView
- ***Interfészek***
  - Drawable
- ***Attribútumok***
  - **+boolean changed:** a legutóbbi rajzolás óta került-e fel rá/le róla pályaelem
- ***Metódusok***

  - **+ boolean Accept():** Amennyiben az ős Accept()-je igazzal tér vissza, igazba billenti a changed flaget
  - **+ boolean Transfer():** Amennyiben az ős Transfer()-je igazzal tér vissza, igazba billenti a changed flaget
  - **+ void Draw():** Felrajzol egy nyelőt a pályára

SwitchView

- ***Felelősség***
  - A kapcsoló kirajzolása
- ***Ősosztályok***
  - Field -> Switch
- ***Interfészek***
  - Drawable
- ***Attribútumok***
  - **+ boolena changed:** a legutóbbi rajzolás óta került-e fel rá/le róla pályaelem
- ***Metódusok***
  - **+ boolean Accept():** Amennyiben az ős Accept()-je igazzal tér vissza, igazba billenti a changed flaget
  - **+ boolean Transfer():** Amennyiben az ős Transfer()-je igazzal tér vissza, igazba billenti a changed flaget
  - **+ void Draw():** Felrajzol egy kapcsolót a pályára

FieldView

- ***Felelősség***
  - A mező kirajzolása
- ***Ősosztályok***
  - Field
- ***Interfészek***
  - Drawable
- ***Attribútumok***
  - **+ boolean changed:** a legutóbbi rajzolás óta került-e fel rá/le róla pályaelem
- ***Metódusok***
  - **+ boolean Accept():** Amennyiben az ős Accept()-je igazzal tér vissza, igazba billenti a changed flaget
  - **+ boolean Transfer():** Amennyiben az ős Transfer()-je igazzal tér vissza, igazba billenti a changed flaget
  - **+ void Draw():** Felrajzol egy mezőt a pályára

HoleView

- ***Felelősség***
  - A lyukak kirajzolása.

· ***Ősosztályok***

- Field -> Hole

· ***Interfészek***

- Drawable

· ***Attribútumok***

- **- boolean changed:** a legutóbbi rajzolás óta került-e fel rá/le róla pályaelem, csak akkor érdekes, ha a lyuk ki van kapcsolva

· ***Metódusok***

- **+ boolean Accept():** Ha a lyuk ki van kapcsolva, arra az esetre a sima mezővelazonos viselkedés
- **+ boolean Transfer():** Ha a lyuk ki van kapcsolva, arra az esetre a sima mezővel azonos viselkedés
- **+ void Draw():** Felrajzol egy lyukat a pályára

WallView

· ***Felelősség***

- A falak kirajzolása.

· ***Ősosztályok***

- Field -> Wall -> WallView

· ***Interfészek***

- Drawable

· ***Metódusok***

- **+ void Draw():** Felrajzol egy falat a pályára.

Drawable

· ***Felelősség***

- Közös alapot biztosít a játékban az összes kirajzolható objektumnak.

· ***Metódusok***

- **+ void Draw():** Egy rajzoló metódus

StorehouseController

· ***Felelősség***

Kezeli azokat az eseményeket (billentyűleütés, kattintás), amelyek a raktárban lévő elemek pozícióját, viselkedését megváltoztatják.

· ***Ősosztályok***

EventHandler

· ***Attribútumok***

- **- StorehouseView view:** a pálya megjelenítéséért felelős osztály
- **- Storehouse storehouse:** a pálya modellje

· ***Metódusok***

- **+handle():** a felhasználói események kezelése

Storehouse

· ***Felelősség***

A raktár és elemeinek kezelője, a modell

· ***Attribútumok***

- **- Field[][] map:** a pályát alkotó kétdimenziós Field tömb
- **- Worker[] players:** a játékosok által irányított dolgozók tömbje

· ***Metódusok***

- **+MovePlayer(int,Direction):** a megfelelő sorszámú játékos mozgatása megadott irányba

StorehouseView

· ***Felelősség***

A grafikus megjelenítésért felelős osztály.

· ***Ősosztályok***

Application

· ***Attribútumok***

- **- Drawable[][] map**: a Drawable interfészt implementáló objektumokat tároló kétdimenziós tömb

· ***Metódusok***

- **+ DrawAll(): void**: végigszalad a map-en, és meghívja az összes elemének a Draw() metódusát
- **+ AnnounceWinner(): void:** egy felugró ablakban kihirdeti a játék győztesét

Application

· ***Felelősség***

JavaFX alkalmazások belépési pontjaként funkcionáló absztrakt osztály.

Worker

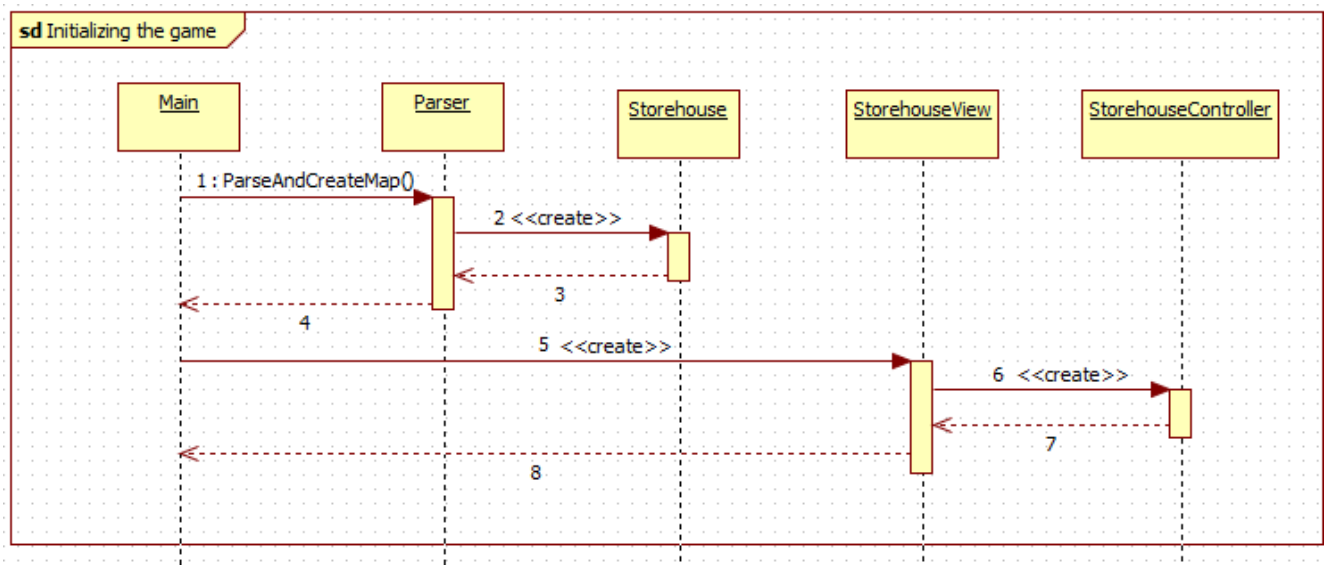
· ***Felelősség***

Dolgozót megvalósító osztály.

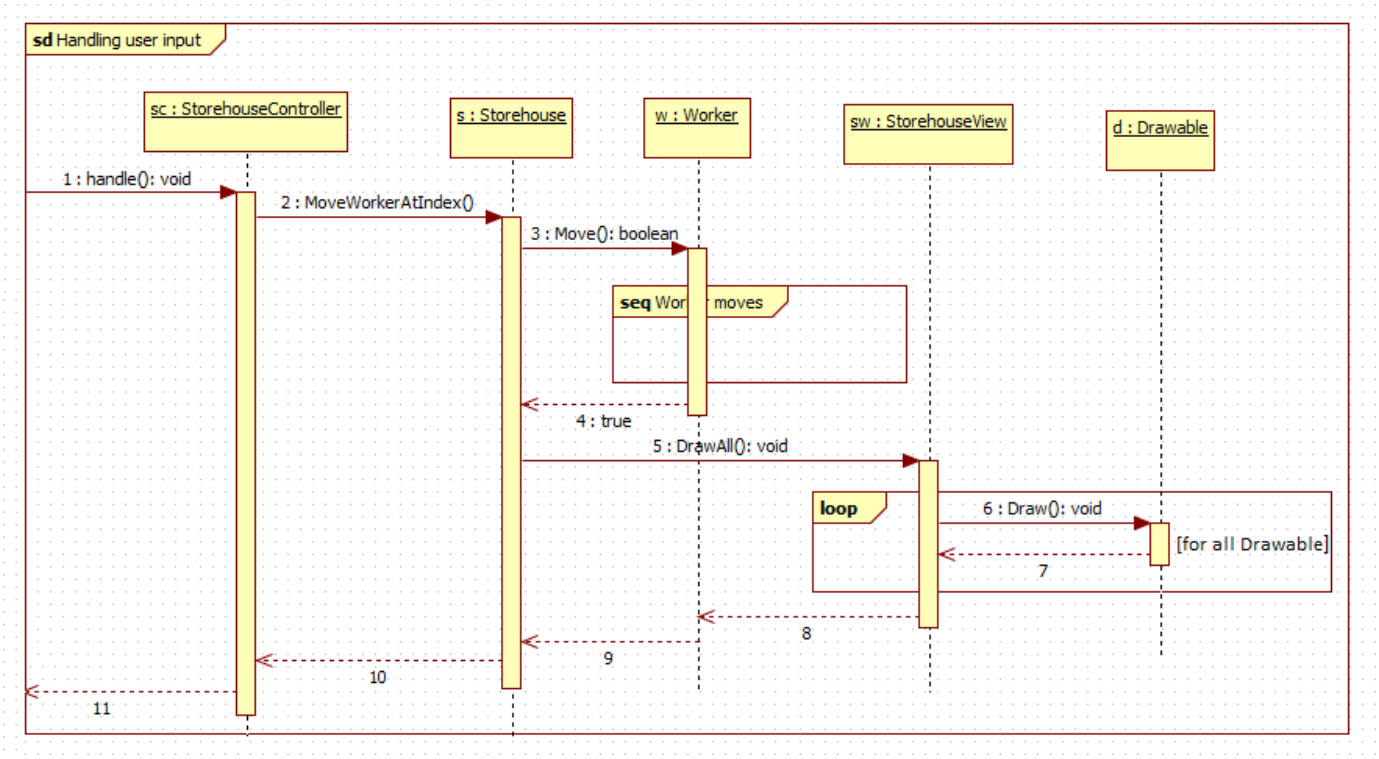
· ***Metódusok***

- **+ Swallowed(): abstract void:** Akkor hívódik meg, ha a dolgozó lyukra vagy nyelőre kerül.
- **+ Move(): boolean:** Dolgozó mozgatása.

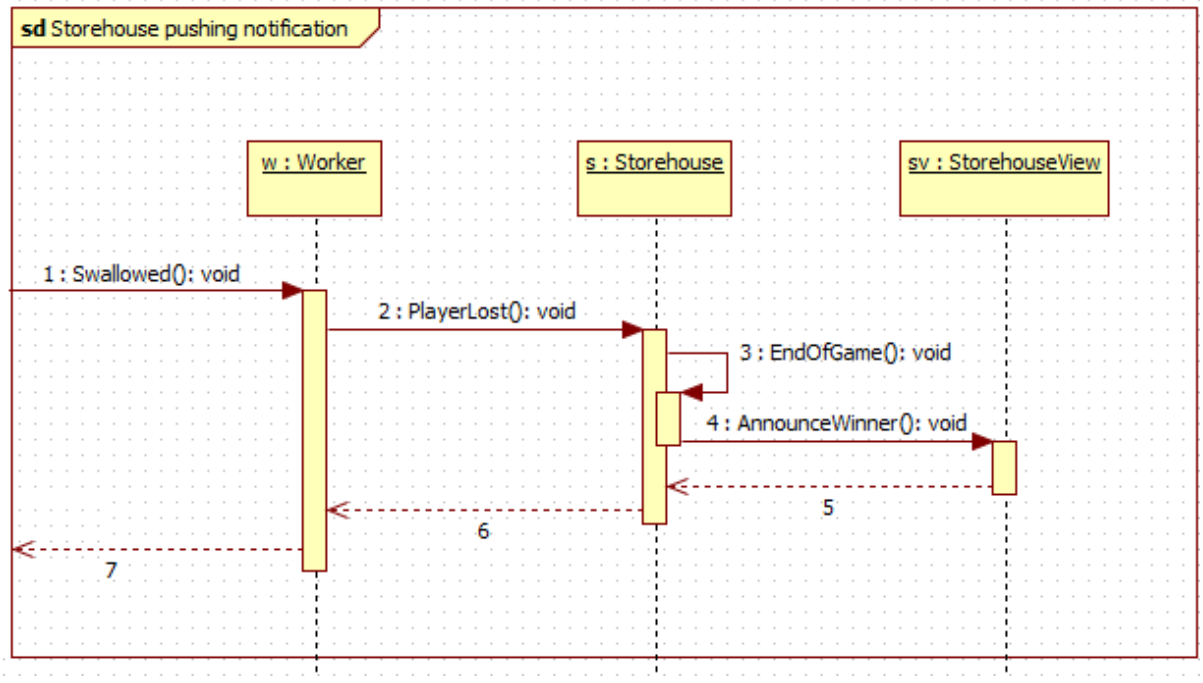
Kapcsolat az alkalmazói rendszerrel



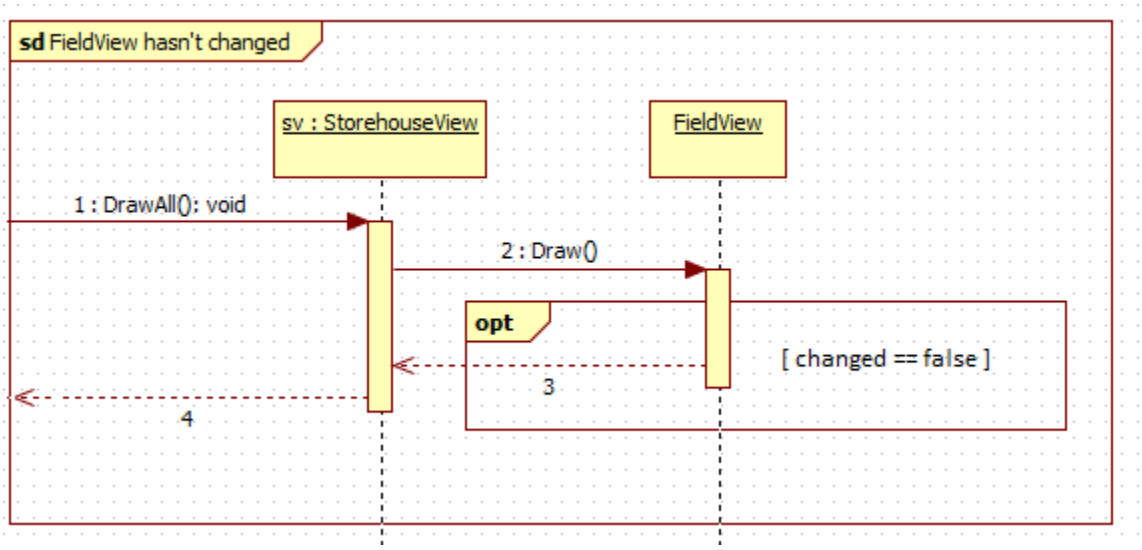
A játék indulása, az egyes komponensek létrehozása



A billentyűk lenyomásának kezelése



A Storehouse értesíti a View-t a játék végéről



11.4.4 Amennyiben a legutolsó rajzolás óra nem változott a mező, nem rajzolja újra magát

Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.04.27	5 óra	Németh, Simon, Nagy, Szövő, Majkut	Grafikus felület kitalálása 11.1 11.2
2018.04.28	1 óra	Németh, Simon, Nagy, Szövő	11.3
2018.04.28	1 óra	Majkut	11.4

Grafikus változat beadása

/13. házi feladat/

1 – a_csapat

Konzulens:

Belákovics Ádám

Csapattagok:

Név
Simon Márta
Németh Marcell
Szövő Roland
Nagy Szabina
Majkut Kristóf

2018.05.14.

13. Grafikus változat beadása

Fordítási és futtatási útmutató

Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom

Chest.java	1 175	2018.05.12	Ládákat megvalósító osztály.
ChestView.java	2 196	2018.05.13	A ládák kirajzolásáért felelős osztály.
Direction.java	76	2018.03.16	Irányokat nyilvántartó osztály.
Drawable.java	257	2018.05.12	Összes kirajzolható objektum interfésze.
Element.java	1 681	2018.05.13	Pályaelemeket megvalósító ősosztály.
Field.java	6 608	2018.05.13	Mezőket megvalósító ősosztály.
FieldView.java	1 758	2018.05.13	A mezők kirajzolásáért felelős osztály.
Goal.java	734	2018.05.13	Nyelőket megvalósító osztály.
GoalView.java	1 623	2018.05.12	A nyelők kirajzolásáért felelős osztály.
Hole.java	1 353	2018.05.13	Lyukakat megvalósító osztály.
HoleView.java	1 934	2018.05.12	A lyukak kirajzolásáért felelős osztály.
InitState.java	164	2018.05.13	Kezdő állapotot tároló osztály.
Moveable.java	259	2018.05.12	Mozdítható objektumok interfésze.
Parser.java	7 978	2018.05.13	Pálya létrehozásáért felelős osztály.
Storehouse.java	5 797	2018.05.13	Mozdítható elemek nyilvántartásáért, játék vezérléséért felelős osztály.
StorehouseController.java	3 123	2018.05.13	Eseményeket kezelő osztály.
StorehouseView.java	4 295	2018.05.13	A grafikus megjelenítésért felelős osztály.
Switch.java	1 984	2018.05.13	Kapcsolókat megvalósító osztály.

SwitchView.java	1 825	2018.05.13	A kapcsolók kirajzolásáért felelős osztály.
Wall.java	902	2018.04.23	Falakat megvalósító osztály.
WallView.java	773	2018.05.12	A falak kirajzolásáért felelős osztály.
Worker.java	2 862	2018.05.12	Dolgozókat megvalósító osztály.
WorkerView.java	2 718	2018.05.13	A dolgozók kirajzolásáért felelős osztály.

Fordítás és telepítés

```
@ECHO OFF

set PATH=%PATH%;C:\Program Files\Java\jdk1.8.0_121\bin
set JAVA_HOME=C:\Program Files\Java\jdk1.8.0_121
set CLASSPATH=%CLASSPATH%;

javac -d bin src/src/*.java
```

Futtatás

```
cd bin
java src.StorehouseView

cd ..
```

Értékelés

Tag neve	Munka százalékban
Nagy Szabina	20%
Németh Marcell	20%
Majkut Kristóf	20%

Simon Márta	20%
Szövő Roland	20%

Napló

Kezdet	Időtartam	Résztvevők	Leírás
2018.05.11	6 óra	Nagy, Németh, Szövő, Simon, Majkut	Kódolás, Dokumentum szerkesztése