14. Összefoglalás

67 – srsh

Konzulens:
Tüske Máté Róbert

Csapattagok:

Dombai Tamás	JHXYRE	dombai_tamas@hotmail.com
Erős Ákos	C81KXC	erosakos02@gmail.com
Márton Patrik	MLXD2E	marpataa@yahoo.com
Székely Bálint	WWHK7Q	szekelybalint96@gmail.com
Szilágyi András	DU1V61	szilagyiandras96@freemail.hu

2019. február 17.

Tartalomjegyzék

Ábrák jegyzéke

2. Követelmény, projekt, funkcionalitás

2.1. Bevezetés

2.1.1. Cél

A projekt követelményeinek, funkcionalitásának leírása. Bemutatni a fejlesztés lépéseit, átfogó képet alkotni az egyes szakaszokról, és a kész szoftver működését bemutatni.

2.1.2. Szakterület

Az elkészült szoftver egy játék lesz. Ebből kifolyólag a szakterület nem egyértelműen behatárolható. Első sorban szórakoztató jelleggel készül.

2.1.3. Definíciók, rövidítések

JetBrains IntelliJ - Java fejlesztői környezet
Eclipse - Java fejlesztői környezet
ShareLaTeX - Online LaTeX szerkesztő
StarUML - UML diagram szerkesztő
Git - Verziókezelő
GitHub - Online Git szolgáltatás
Google Dokumentumok - Online dokumentumszerkesztő
Google Táblázatok - Online táblázatkezelő
HSZK - Hallgatói Számítógépes központ
UML - Általános célú modellező nyelv
BME-IIT - Budapesti Műszaki és Gazdaságtudományi Egyetem Irányítástechnika és Informatika Tanszék

2.1.4. Hivatkozások

Szoftver Projekt Laboratórium - https://www.iit.bme.hu/targyak/BMEVIIIAB02?language=hu

2.1.5. Összefoglalás

A dokumentum további részében található:

- Szoftver rövid bemutatása
- Funkcionális és nem funkcionális követelmények leírása
- Fontosabb Use-Case-ek felsorolása
- Szótár a nem megszokott értelemben vett szavak magyarázatával
- Projekt tervezete
- Projektnapló

2.2. Áttekintés

2.2.1. Általános áttekintés

A program főbb részei:

- Játékmotor
- Grafikus felület

A Játékmotor felelős a logika megvalósításáért, a pályák betöltéséért, az alagutak módosításáért, a felhasználói interakciók kezeléséért. Kommunikál a grafikus felülettel, adatokat szolgáltat neki a felület kirajzolásához, parancsokat fogad tőle és ad át neki.

A Grafikus felületről irányítható a Játékmotor, és ez szolgál a játék megjelenítéséért.

2.2.2. Funkciók

A feladat egy terepasztalos játék elkészítése. A terepasztal egy előre felépített környezet, amelyen vonatok és az általuk húzott kocsik mozognak. Az asztalon egyszerre több vonat is lehet, ezek hossza az általuk húzott kocsiktól függ, előfordulhat nagy különbség is két vonat hossza között, így kialakulhat az az eset is hogy egy vonat teljesen eltűnik míg áthalad egy alagúton, azonban egy másik még akkor is látszik a bejáratnál, amikor az eleje már kijött az alagútból.
A terepasztalon való mozgást alapvetően sínek és a sínekre épített váltók biztosítják, illetve az alagutak ki-és bemenetei. A síneket elágazásoknál tehát a váltók kötik össze, ezek állításával lehet biztosítani, hogy a vonat a megfelelő irányba haladjon tovább. A vonat mindig a váltó által mutatott irányba halad tovább, nem fordulhat elő, hogy a másik irányba haladó sínre kerül. A pályán egyszerre maximum két alagút bejárat lehet, ezeket köti össze az alagút, ha a vonat az egyik szájon bemegy, akkor a másikon jön ki. Az alagútban nem lehet megállni, mivel ezt az esetleg az alagútba érkező másik vonat nem észlelheti, és veszélyes szituációt teremthet. A síneken mindkét irányba lehet mozogni, ezért különösen figyelni kell arra hogy a vonatok ne ütközzenek össze. Ez előforduhat akkor például, ha a vonatok egyazon váltóra kerülnek.
A sínek ezen kívül különböző állomásokat kötnek össze, ahova az utasok utaznak, Az állomásoknak van színe, amely egyértelműen azonosítja őket, innen lehet tudni hogy mikor kell leszállnia egy utasnak a vonatról. A vonatok kocsijainak is van színe, ebben azon utasok utaznak, akik a kocsival azonos színű állomáson akarnak leszállni. A megfelelő állomáson csak az azonos színű kocsiban utazók ugorhatnak le a vonatról, a többiek nem, nekik meg kell várni amíg a megfelelő állomásra ér a vonat. A kocsik a mozdonytól kezdődő sorrendben ürülnek ki. Ebből következik, hogy ha egy vonat elér egy állomást, addig nem ürülhet ki egy kocsija, amíg van olyan előtte lévő kocsi, ami nem üres. Ez természetesen akkor is érvényes, ha a kocsi színe azonos az állomás színével, ekkor sikertelen lesz a vonatról való leszállás.
A vonatokat a játékos tudja irányítani, teljes mértékben ő felel a vonatok megfelelő irányításáról. A vonatok irányítása a váltók különböző irányba való állításával lehetséges. természetesen az alagút szájra vonatkozó kikötés itt is érvényes, többet nem tehet egyszerre a pályára. A váltók száma tetszőleges, azonban a váltók csak indokolt esetben működnek, ott ahol nincs mit váltani, nem befolyásolják a vonat irányát. A játékos célja, hogy minden vonat minden utasa elérje a megfelelő állomást, és leszálljon ott. A játék megnyerésére ekkor van lehetősége. Ha két vonat összeütközik, vagy egy vonat kisiklik, akkor a játékos vesztett, hiszen a játék célját nem tudja már teljesíteni. Miután a játékos egy terepasztalt sikeresen teljesített, azaz minden utas célba jutott, továbbléphet a következő, már nehezebb környezettel rendelkező és több vonatot, utast tartalmazó pályára.

2.2.3. Felhasználók

A játékban egy felhasználó vesz részt. A játékos képes a váltók irányát megváltoztatni, illetve az alagutak be és kijáratait változtatni. A játékos célja, hogy minden utas eljusson a megfelelő állomásra, így megnyerje a játékot. A játékos nem képes a váltókat áthelyezni, illetve eltüntetni, ez a játék alapvető struktúrájából adódik és a pályához tartozik. A játékos ezen kívül a pálya struktúráját sem képes átalakítani, ezek előre adottak.

2.2.4. Korlátozások

A játékot egyszerre egy felhasználó játszhatja.
A pálya előzetesen fixen van elkészítve, a játékos nem tudja ezt megváltoztatni, nem tud síneket hozzáadni, illetve eltüntetni, sem váltókat elhelyezni. A játékos egyedül az alagutak be-és kijáratait tudja befolyásolni, mivel ehhez van joga.
A vonatok tetszőleges hosszúak lehetnek, de van fix méretük, mivel befolyásolhatja a játék élményt, ha egy túl hosszú vonat miatt például több nem fér a pályára. A vonat egy mozdonyból, és maximum 10 rácsatolt kocsiból állhat.
Egyes kocsikban csak olyan utasok utazhatnak, akik a megfelelő színű állomáson akarnak leszállni, mások nem. A vonat előről üresedik, minden kocsi csak akkor tud ürülni, ha előtte üres kocsik állnak.
Ha a játékos egy pályát nem teljesített, azt előről kell kezdenie. Egy pálya akkor sikertelen ha két vonat összeütközik rajta, vagy egy vonat kisiklik; illetve ha nem teljesíti a játék célját, azaz nem jut el minden utas a megfelelő állomásra.
További korlátozások:
-Minden vonat azonos, fix, nem változtatható sebességgel közlekedik; nem állhatnak meg.
-A váltók, állomások, alagútszájak pontszerűnek tekinthetők, hosszuk nem értelmezett.
-A sínek csak egyenesek lehetnek.
-Tíz színt különböztetünk meg.

2.2.5. Feltételezések, kapcsolatok

Szoftver projekt laboratórium feladatkiírás:
https://www.iit.bme.hu/targyak/BMEVIIIAB02/feladat?language=hu

2.3. Követelmények

2.3.1. Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
F01	Alagútszájat lehet építeni	bemutatás	alapvető	specifikáció	Alagútszáj építés	-
F02	Alagútszájat lehet törölni	bemutatás	alapvető	specifikáció	Alagútszáj törlés	-
F03	Váltót lehet állítani	bemutatás	alapvető	specifikáció	Váltó állítás	-
F04	Alagútszájat csak erre a célra kijelölt pontra lehet építeni	bemutatás	alapvető	specifikáció	Alagútszáj építés	-
F05	Ha két vonat ütközik, a játékos veszít	bemutatás	alapvető	specifikáció	-	-
F06	Ha egy vonat kisiklik, a játékos veszít	bemutatás	fontos	csapat	-	-
F07	Ha minden vonatról leszálltak, a játékos nyer	bemutatás	alapvető	specifikáció	-	-

F08	Több pálya van	bemutatás	alapvető	specifikáció	-	-
F09	Játék indítható	bemutatás	alapvető	specifikáció	Játék indítás	-
F10	Játék bezárható	bemutatás	alapvető	specifikáció	Játék bezárás	-
F11	Pálya újrakezdhető	bemutatás	opcionális	csapat	Újrakezdés	-
F12	Vonatok mozognak a pályán	bemutatás	alapvető	specifikáció	-	-
F13	Vonat alagútban lévő része nem látszik	bemutatás	alapvető	specifikáció	-	-
F14	Vonatról leszállhatnak az utasok	bemutatás	alapvető	specifikáció	-	-
F15	Különböző hosszúságú vonatok vannak	bemutatás	opcionális	csapat	-	Egy vonat maximum 10 kocsiból állhat, és egy mozdonyból.
F16	Kocsikról csak megfelelő körülmények között szállhatnak le az utasok	bemutatás	alapvető	specifikáció	-	-
F17	Kocsiknak és állomásoknak van színük	bemutatás	alapvető	specifikáció	-	-
F18	Két alagútszáj közé egy alagút épül	bemutatás	alapvető	specifikáció	-	-
F19	Két alagútszáj közé a legrövidebb úton épül alagút	bemutatás	opcionális	csapat	-	-

2.3.2. Erőforrásokkal kapcsolatos követelmények

Azonosító	**Leírás**	**Ellenőrzés**	**Prioritás**	**Forrás**	**Komment**
E01	ShareLaTeX	nincs	opcionális	csapat	Online LaTeX dokumentumszerkesztő
E02	Git	nincs	alapvető	csapat	Verziókezelő

E03	Eclipse	nincs	opcionális	csapat	Java fejlesztőkörnyezet HSZK-ban
E04	IntelliJ IDEA	nincs	opcionális	csapat	Java fejlesztőkörnyezet
E05	Star UML	nincs	alapvető	csapat	UML diagram szerkesztő
E06	Perifériák	nincs	alapvető	csapat	Monitor, egér
E07	HSZK-s, vagy jobb számítógép	bemutatás	alapvető	megrendelő	
E08	Google Dokumentumok	nincs	opcionális	csapat	Online dokumentumszerkesztő
E09	Google Táblázatok	nincs	opcionális	csapat	Online táblázatkezelő

2.3.3. Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
A01	Dokumentáció alapján könnyen telepíthető	bemutatás	fontos	megrendelő	-
A02	JRE8 ellenőrzés	bemutatás	alapvető	megrendelő	-

2.3.4. Egyéb nem funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
N01	Egységes kód	nincs	alapvető	csapat	Egységes, átlátható kód formázás, a könnyebb kezelhetőség érdekében
N02	Könnyű használhatóság	nincs	opcionális	csapat	Előképzettség nélküli felhasználó képes legyen használni a játékot

2.4. Lényeges use-case-ek

2.4.1. Use-case leírások

Use-case neve	Alagútszáj építés
Rövid leírás	Kattinrásra az erre megjelölt különleges helyen létrehozza az alagútszáját.
Aktorok	Player
Forgatókönyv	Ha olyan helyre kattint a játékos, ahol alagútszáj építése lehetséges, és a maximális kettőnél kevesebb van felépítve, akkor építésre kerül.

Use-case neve	Alagútszáj törlés
Rövid leírás	Kattintásra a már felépített alagutszáját eltávolítjuk a pályáról.
Aktorok	Player

Forgatókönyv	Ha a játékos olyan helyre kattint, ahol már van egy felépített alagútszáj, akkor az eltávolításra kerül.

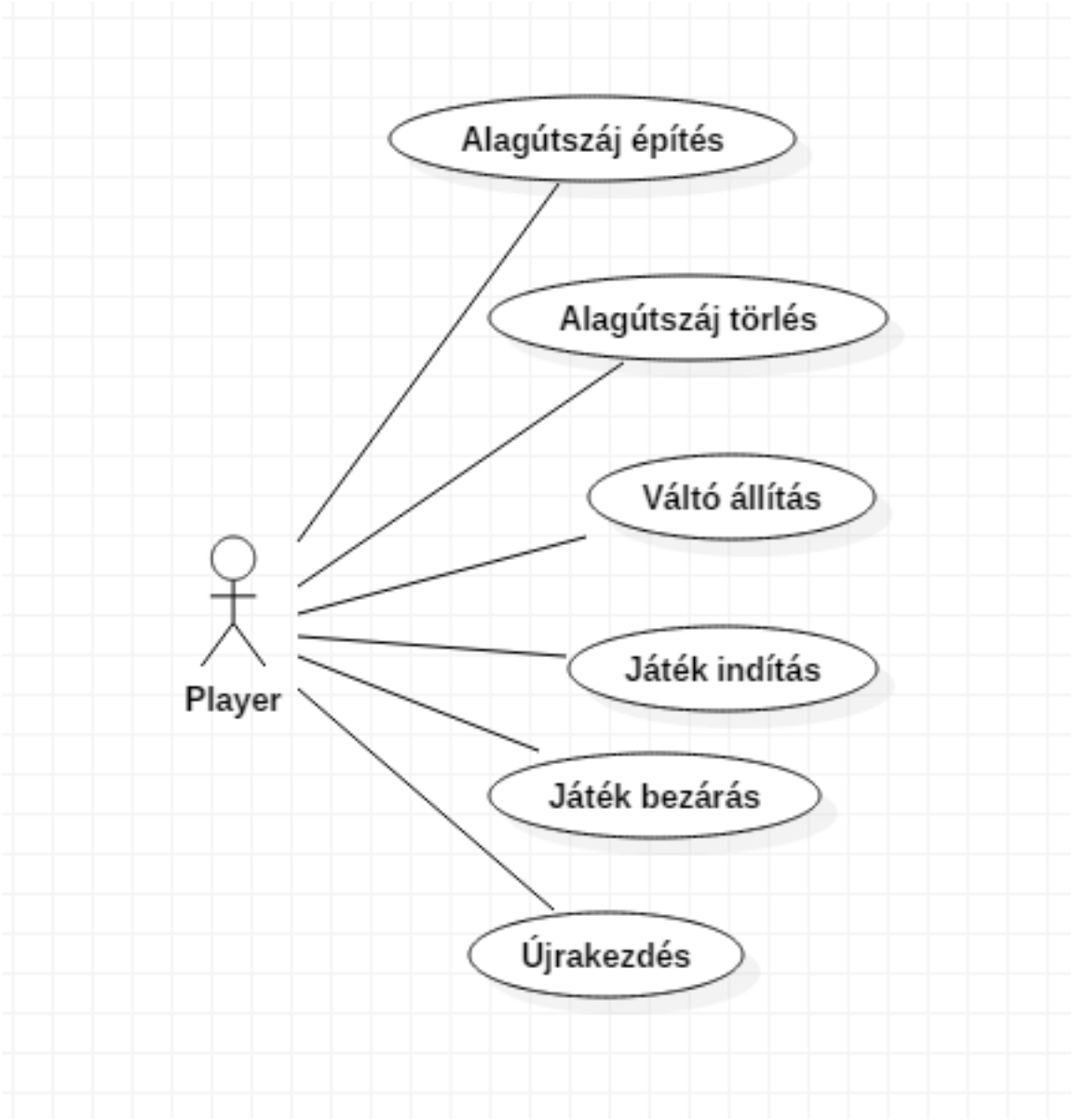
Use-case neve	**Váltó állítás**
Rövid leírás	Kattintásra a váltó állapota az eddigi beállítás ellentettjére módosul.
Aktorok	Player
Forgatókönyv	Ha a játékos a váltóra kattint, és a váltó eddig az első kimenetre továbbította a vonatot, akkor az átállítás után a másodikra fogja, és fordítva.

Use-case neve	**Játék indítás**
Rövid leírás	Gombnyomásra a játék elindul.
Aktorok	Player
Forgatókönyv	A játékos miután a megfelelő gombot megnyomta, a pálya betöltésre kerül, a vonatok elindulnak, a kurzor aktívvá válik és a játékos kezelheti a pályát.

Use-case neve	**Játék bezárás**
Rövid leírás	Gombnyomásra a játék bezárul.
Aktorok	Player
Forgatókönyv	A jelenlegi, még teljesítésre váró pálya sorszáma mentésre kerül, és a játék bezárul, ha a játékos a bezárásra klikkel.

Use-case neve	**Újrakezdés**
Rövid leírás	A játékos újraindíthatja a jelenlegi pályát.
Aktorok	Player
Forgatókönyv	Ha a játékos vesztett, vagy úgy ítéli meg, hogy célszerűbb alaphelyzetből kezdeni a pályát, akkor a megfelelő gomb megnyomásával a pálya kezdeti állapota töltődik be ismét.

2.4.2. Use-case diagram



2.5. Szótár

Alagút	Két alagútszájat összekötő fedett sín(pont). A vonat alagútban lévő része nem látható.
Alagútszáj	Játékos készítheti, illetve szüntetheti meg erre a célre kijelölt speciális helyeken. Egy alagút kezdő ill. végpontja.
Alagútszáj építés	Játékos általi alagútszáj elhelyezés a terepasztalon erre kijelölt speciális helyen.
Alagútszáj megszüntetés	Játékos általi alagútszáj törlés a terepasztalon.

Állomás	Speciális sín(szakasz). Terepasztalon kijelölt speciális hely; az állomás színével azonos színű kocsikból szállhatnak ki az utasok.
Hossz	Sínekhez rendelhető mennyiség, két pont közötti távolságot adja meg.
Játék elvesztése	A játék egy lehetséges végződése, akkor következik be, ha két vonat ütközik, vagy egy vonat kisiklik.
Játék megnyerése	A játék egy lehetséges végződése, akkor következik be, ha minden vonat üres.
Kisiklás	Az az esemény, ami akkor következik be, ha egy vonat a váltón úgy akar áthaladni, hogy azt a váltó állása nem teszi lehetővé.
Kocsi	A terepasztalon síneken haladnak egy mozdonnyal alkotott vonatként. Van színe.
Leszállás	Az az esemény, amikor egy állomáson az utas leszáll a vonatról.
Megrendelő	Laborkonzulens, tárgyfelelős, BME-IIT
Mozdony	A terepasztalon síneken haladnak. Egy mozdonyhoz több kocsi is tartozhat, együtt egy vonatot alkotnak. Utasokat nem szállít, színnel nem rendelkezik.
Sín	A terepasztalon kijelölt sávok, amiken a vonatok haladhatnak. Némelyiken speciális pont található, amire alagútszájat építhet a játékos.
Szín	Kocsikhoz és állomásokhoz rendelhető tulajdonság, mely azok színét adja meg.
Terepasztal	Az a tér amin a játék zajlik.
Utas	Kocsikban utazó személy, aki állomásokon leszállhat a vonatról.
Üres vonat	Az a vonat, amelyiknek egyik kocsijában sem található utas.
Ütközés	Az az esemény ami akkor következik be, ha két vonat azonos sínen azonos ponton található meg.
Váltó	Olyan speciális sín(pont), aminek egy bemenete, és két kimenete lehet. A két kimenet között a játékos képes váltani.
Váltó állítás	Váltó két kimenete közötti váltás.
Vonat	Mozdonyból és max. 10 db kocsiból álló szerelvény (de minimum 1). A terepasztalon síneken haladnak.

2.6. Projekt terv

2.6.1. Ütemterv, lépések

A fejlesztési folyamat három fő lépésből áll, ezek a következők: szkeleton, prototípus, grafikus változat. A részfeladatokat, és a pontos ütemezést a lenti táblázat tartalmazza:

Dátum	**Feladat**	**Ellenőrzés módja**
2017.02.20.	Követelmény, projekt, funkcionalitás	beadás
2017.02.27.	Analízis modell kidolgozása 1.	beadás
2017.03.06.	Analízis modell kidolgozása 2.	beadás
2017.03.13.	Szkeleton tervezése	beadás, dokumentum feltöltés
2017.03.20.	Szkeleton	beadás, forráskód feltöltés
2017.03.27.	Prototípus koncepciója	beadás
2017.04.03.	Részletes tervek	beadás
2017.04.10.	Prototípus készítése, tesztelése	-
2017.04.17.	Prototípus	beadás; forráskód, tesztbemenet, elvárt kimenet feltöltés
2017.04.24.	Grafikus felület specifikációja	beadás
2017.05.01.	Grafikus változat készítése	-
2017.05.08.	Grafikus változat	beadás, forráskód feltöltés
2017.05.10.	Összefoglalás	beadás, feltöltés

2.6.2. Csapat, beosztás

Az **srsh** csapat 5 főből áll. A csapat tagjainak nevei, elérhetőségei, fő feladatai a lenti táblázatban találhatók.

Név	**E-mail**	**Fő szerep**
Dombai Tamás	dombai_tamas@hotmail.com	Dokumentáció, UML, kód
Erős Ákos	erosakos02@gmail.com	Kód, UML, dokumentáció
Márton Patrik	marpataa@yahoo.com	Dokumentáció, kód
Székely Bálint	szekelybalint96@gmail.com	Kód, UML, dokumentáció
Szilágyi András	szilagyiandras96@freemail.hu	Kód, dokumentáció

2.6.3. Eszközök, technikák

Kommunikációhoz alapvetően két szolgáltatást fogunk igénybe venni. **Skype**-on létrehoztunk egy konferenciabeszélgetést, ahol igény szerint bármikor megbeszélést tarthatunk. **Facebook**-on létrehozott csoportos beszélgetés valamint zárt csoport segíti a közös munkát.

Verziókezelést a forráskód szerkesztéséhez használunk. Választásunk a **GitHub**-ra esett, annak egyszerűsé-

ge és közismertsége miatt.

Dokumentumszerkesztéshez a **ShareLaTeX** online LaTex szerkesztőt, valamint a **Google Dokumentumok** szolgáltatását használjuk. A választás azért ezekre esett, mert képesek valós időben mutatni a dokumentumokon végzett változtatásokat, így elkerülve az esetleges félreértéseket a csapattagok között.

Fejlesztőkörnyezetekből a választás alapvetően az **IntelliJ IDEA**-ra esett, de **Eclipse**-t is használni fogunk bizonyos feladatokra.

Naplót fogunk vezetni az elvégzett feladatokról **Google Táblázatok**-ban. Azért ezt választottuk, mert itt valós időben láthatjuk, hogy melyik csapattag éppen melyik feladatrészen dolgozik.

2.7. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017.02.12. 22:00	1 óra	**Dombai** **Erős** **Márton** **Székely** **Szilágyi**	Megbeszélés. Projekttel való ismerkedés, feladat értelmezése. Egységes koncepció kialakítása.
2017.02.16. 19:00	1 óra	**Dombai**	Tevékenység: 2.3.2, 2.5 alpontok megírása.
2017.02.17. 19:00	1 óra	**Dombai**	Tevékenység: 2.6 alpontok megírása.
2017.02.18. 15:00	1 óra	**Márton**	Tevékenység: 2.2.2, 2.2.3, 2.2.4 alpontok megírása.
2017.02.18. 17:00	1 óra	**Dombai**	Tevékenység: 2.2.5, 2.3.1, 2.3.3, 2.3.4 alpontok megírása.
2017.02.18. 18:00	1 óra	**Székely**	Tevékenység: 2.4.1 alpont megírása.
2017.02.18. 18:00	1 óra	**Erős**	Tevékenység: 2.1 alpont megírása.
2017.02.19. 18:00	1 óra	**Székely**	Tevékenység: 2.4.1 kiegészítés, 2.4.2 diagram készítés, ellenőrzés.
2017.02.19. 19:00	1 óra	**Erős**	Tevékenység: 2.2.1 alpont megírása
2017.02.19. 21:00	1 óra	**Dombai** **Erős** **Márton** **Székely** **Szilágyi**	Megbeszélés. Elkészült anyag közös átnézése; hibák keresése, javítása. Előttünk álló feladatok gyors átbeszélése.

3. Analízis modell kidolgozása 1

3.1. Objektum katalógus

3.1.1. Application

Létrehozza a szükséges objektumokat, elindítja a játékot.

3.1.2. Connector

Két sínt köt össze, az egyenes sínek miatt szükséges a kanyarok megvalósításához.

3.1.3. Controller

A játékos ezen keresztül indíthatja el a játékot, irányítja a váltókat és módosíthatja az alagút helyét. A játék megkezdése után elindít egy Timert ami felel a vonatok mozgásáért, és a parancsok olvasásáért.

3.1.4. EndNode

Vakvágány megvalósítása, a végén a vonat kisiklik.

3.1.5. Graph

A vasút hálózat képe, tárolja a síneket, állomásokat, váltókat, csatlakozókat és a vakvágányokat egy gráf formájában a könnyebb használhatóság érdekében.

3.1.6. Map

A vasút hálózatot és a vonatokat tartalmazza.

3.1.7. Rail

Sín amin a vonat közlekedik, csomópontokhoz kapcsolódik, összeköt két elemet.

3.1.8. Station

Állomás, ahol az utasok leszállnak a megfelelő színű kocsiról. Saját színnel rendelkezik(az azonos színű kocsiról szállnak le az utasok ha ez lehetséges).

3.1.9. Switch

Váltó, amivel sínek között lehet váltani, a játékos irányítja így van hatása a vonatok mozgására.

3.1.10. TrainLocomotive

Mozdony amihez vagonokat lehet kötni. Egy jelre mozog és mozgatja a hozzá kapcsolt vagonokat.

3.1.11. Tunnel

Alagút ami alatt a vonat át tud haladni. Az alagút kijáratait a játékos áthelyezheti.

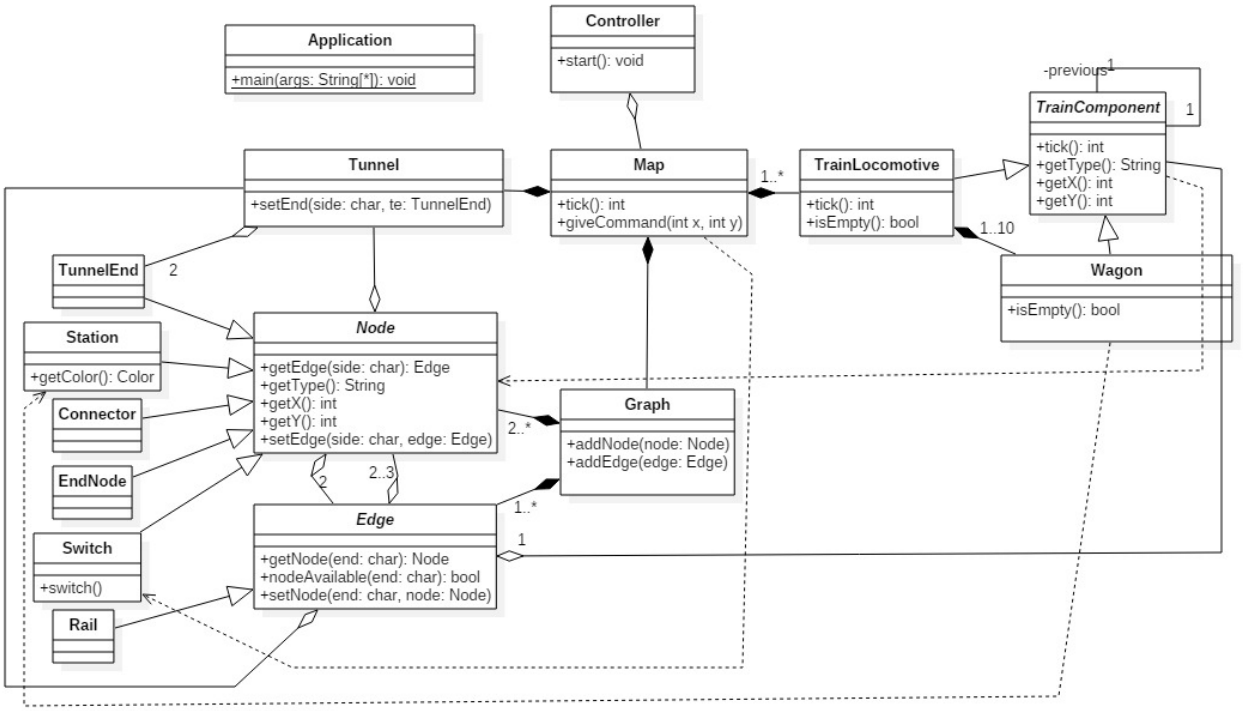
3.1.12. TunnelEnd

Speciális pontok, ahova a játékos alagút szájat építhet.

3.1.13. Wagon

Saját színnel rendelkezik, egy mozdonyhoz vagy egy másik vagonhoz van kapcsolva ami húzza és hozzákapcsolható egy másik vagon. A megfelelő színű állomás mellett elhaladva az utasok leszállnak, ha ez lehetséges.

3.2. Statikus struktúra diagramok



3.1. ábra. Osztálydiagram publikus metódusokkal

3.3. Osztályok leírása

3.3.1. Application

- Felelősség
  Megkezdődik a különböző objektumok létrehozása, itt kezdődik a megfelelő asszociációk felépítése, illetve a standard input figyelése is. Létrehozza a Controller osztály egy példányát, innen megkezdődik a szükséges objektumok létrejötte.
- Ősosztályok
  Az Application osztálynak nincs ősosztálya.
- Interfészek
  Az Application nem valósít meg interface-t.

- Attribútumok
  Nincsenek publikus attribútumok.
- Metódusok
  - void main(args: String[*]): A program main metódusa, a fentiek alapján működik.

3.3.2. Connector

- Felelősség
  A játékban csak egyenes sínek lehetnek. Ahhoz, hogy a különböző sínek össze legyenek kötve, a Connector osztály nyújt segítségét. Gráf szinten a Connector egyfajta Node, ami síneket köt össze.
- Ősosztályok
  A Connector osztály a Node osztályból származik
- Interfészek

  A Connector osztály nem valósít meg interface-t.
- Attribútumok

  Nincsenek publikus Attribútumok.
- Metódusok

  A Node osztályból örökölt metódusok, a pontos leírásuk az ősosztálynál található.
  - Edge getEdge(side: char)
  - String getType()
  - int getX()
  - int getY()
  - void setEdge(side: char, edge: Edge)

3.3.3. Controller

- Felelősség

  A játékos a Controller osztály segítségével tudja irányítani a játékot, ellátni a számára engedélyezett szerepet, módosítani az alagút végét illetve a váltókat állítani.
- Ősosztályok

  A Controller osztálynak nincs ősosztálya.
- Interfészek

  A Controller osztály nem valósít meg interface-t.
- Attribútumok

  Ninscenek publikus attribútumok.

- Metódusok

  - void start(): Egy timer szerint hívja a Map tick() metódusát, valamint a Map giveCommand() metódusát.

3.3.4. Edge

- Felelősség

  A Gráf éleit reprezentáló osztály. Ezek segítségével vannak összekötve a csomópontok (Node), ezért gyakorlatilag a sín szerepét tölti be.

- Ősosztályok

  Az Edge osztálynak nincs ősosztálya.

- Interfészek

  Az Edge osztály nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus Attribútumok.

- Metódusok

  - Node getNode(end: char): Visszaadja a paraméterként megadott végen lévő csomópont.
  - bool nodeAvailable(end: char): Visszaadja, hogy elérhető-e egy csomópont. (pl. váltó nem erre a sínre van állítva)
  - void setNode(end: char, node: Node): Beállít egy paraméterként megadott csomópontot.

3.3.5. EndNode

- Felelősség
  Az EndNode osztály reprezentálja a vakvágány kiindulópontját. Ha a vonat eléri a vakvágány végét, kisiklik és a játéknak vége.

- Ősosztályok

  Az EndNode osztály a Node osztályból származik

- Interfészek

  Az EndNode osztály nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus Attribútumok.

- Metódusok

  A Node osztályból örökölt metódusok, a pontos leírásuk az ősosztálynál található.
  - Edge getEdge(side: char)
  - String getType()
  - int getX()
  - int getY()
  - void setEdge(side: char, edge: Edge)

3.3.6. Graph

- Felelősség

  A pályát (terepasztalt) reprezentáló osztály, amely az terepasztal sajátosságainak megfelelő gráfot tartalmaz. Ez lehetővé teszi a pálya egy egyszerűbb képével való munkát, illetve a pálya megfelelő tárolását a szükséges információkkal.

- Ősosztályok

  A Graph osztálynak nincs ősosztálya,

- Interfészek

  A Graph osztály nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus attrubútumok.

- Metódusok

  - void addNode(node: Node): Csomópont hozzáadását lehetővé tevő metódus.
  - void addEdge(edge: Edge): Él hozzáadását lehetővé tevő metódus.

3.3.7. Map

- Felelősség

  A térképet reprezentáló osztály. A vasúti hálózatot (mint Graph), valamint vonatokat, ill. az alagutat tartalmaza. Gyakorlatilag összefogja a pályát.

- Ősosztályok

  A Map osztálynak nincs ősosztálya.

- Interfészek

  A Map osztály nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus attribútumok.

- Metódusok

  - int tick(): Végighívja az összes komponens tick() metódusát.

3.3.8. Node

- Felelősség
  A vasúti pálya alapvetően egy gráffal van tárolva. A Node osztály a gráf csomópontjait reprezentálja. Öt leszármazottja van, attól függően, hogy milyen típusú az adott csomópont. Az élek (Edge) ezeket a csomópontokat kötik össze.

- Ősosztályok

  A Node osztálynak nincs ősosztálya.

- Interfészek

  A Node osztály nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus attribútumok.

- Metódusok

  - Edge getEdge(side: char): Visszaadja a csomópontba futó paraméterként kapott oldalon álló élet.
  - String getType(): Visszaadja milyen típusú az adott csomópont.
  - int getX(): Visszaadja a csomópont X pozícióját.
  - int getY(): Visszaadja a csomópont Y pozícióját.
  - void setEdge(side: char, edge: Edge): Paraméterként kapott Edge-t ad hozzá a csomópont paraméterként kapott végére.

3.3.9. Rail

- Felelősség

  A Rail osztály egy sínt reprezentál, amin a vonat közlekedik, ez a pálya egyéb elemeit köti össze egymással.

- Ősosztályok

  A Rail osztály az Edge osztályból származik

- Interfészek

  A Rail osztály nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus Attribútumok.

- Metódusok

  Az Edge osztályból örökölt metódusokkal rendelkezik, pontos leírásuk az ősosztálynál.
  - Node getNode(end: char)
  - bool nodeAvailable(end: char)
  - void setNode(end: char, node: Node)

3.3.10. Station

- Felelősség

  A station egy állomást reprezentáló osztály. Az állomásnak saját színe is van, az egyes utasok ezek alapján tudják, hogy leszállnak-e az adott állomáson vagy sem.

- Ősosztályok

  A Station osztály a Node osztályból származik

- Interfészek

  A Station osztály nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus Attribútumok.

- Metódusok

  A Node osztályból örökölt metódusok, a pontos leírásuk az ősosztálynál található (első 5 metódus).
  - Edge getEdge(side: char)
  - String getType()
  - int getX()
  - int getY()
  - void setEdge(side: char, edge: Edge)
  - Color getColor(): Visszaadja az állomás színét.

3.3.11. Switch

- Felelősség

  A Switch egy váltót reprezentáló osztály. A váltóval a sínek között lehet váltani. Az állása a játékostól függ, ő irányítja, ezzel befolyásolva a vonat mozgását.

- Ősosztályok

  A Switch osztály a Node osztályból származik

- Interfészek

  A Switch osztály nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus Attribútumok.

- Metódusok

  A Node osztályból örökölt metódusok, a pontos leírásuk az ősosztálynál található.
  - Edge getEdge(side: char)
  - String getType()
  - int getX()
  - int getY()
  - void setEdge(side: char, edge: Edge)
  - void switch(): Két sín közötti váltást megvalósító metódus.

3.3.12. TrainComponent

- Felelősség

  A TrainComponent osztály a vonat különböző komponenseit reprezentálja. A vonat állhat egy mozdonyból, és az azt követő vagonokból, ezek mind komponensek.

- Ősosztályok

  A TrainComponent osztálynak nincs ősosztálya.

- Interfészek

  A TrainComponent osztály nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus attribútumok.

- Metódusok

  - int tick(): Saját státuszának megfelelő értékkel tér vissza (1 ha kisiklott, 0 ha nem lépett fel probléma), ütközésdetektálás Map-ben.
  - String getType(): Visszaadja az adott komponens típusát.
  - int getX(): Visszaadja az adott komponens X pozícióját.
  - int getY(): Visszaadja az adott komponens Y pozícióját.

3.3.13. TrainLocomotive

- Felelősség

  A TrainLocomotive osztály egy mozdonyt (mint komponens) reprezentál. Ehhez lehet még vagonokat csatlakoztatni, amik együtt, egy jelre mozognak.

- Ősosztályok

  A Locomotive osztály a TrainComponent osztályból származik.

- Interfészek

  A Locomotive osztály nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus attribútumok.

- Metódusok

  Csak örökölt metódusokkal rendelkezik, ezek pontos leírása az ősosztálynál található.
  - bool tick()
  - String getType()
  - int getX()
  - int getY()

3.3.14. Tunnel

- Felelősség

  A Tunnel egy pályán elhelyezhető alagutat reprezentáló osztály. A vonatok át tudnak haladni ezen az alagúton. A játékos jogosultságai közé tartozik, hogy áthelyezheti az alagút egyes végeit, így változtatva a pályát.

- Ősosztályok

  A Tunnel osztálynak nincs ősosztálya.

- Interfészek

  A Tunnel osztály nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus attribútumok.

- Metódusok

  - TunnelEnd setEnd(side: char, te: TunnelEnd): Beállítható az alagút egy vége.

3.3.15. TunnelEnd

- Felelősség

  A TunnelEnd az alagút egy végét reprezentálja. A játékosnak lehetősége van áthelyezni.

- Ősosztályok

  A TunnelEnd osztály a Node osztályból származik.

- Interfészek

  A TunnelEnd osztály nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus Attribútumok.

- Metódusok

  A Node osztályból örökölt metódusok, a pontos leírásuk az ősosztálynál található.
  - Edge getEdge(side: char)
  - String getType()
  - int getX()
  - int getY()
  - void setEdge(side: char, edge: Edge)

3.3.16. Wagon

- Felelősség

  A Wagon osztály a mozdonyhoz csatlakoztatható vagont reprezentálja. A vagon a mozdonyhoz hasonlóan egy komponens, ezeket egybe lehet építeni. A vagonban utaznak az utasok. Saját színnel rendelkezik, így az azonos színű állomáson az utasok leszállhatnak, ha ez lehetséges.

- Ősosztályok

  A Wagon osztály a TrainComponent osztályból származik.

- Interfészek

  A Wagon osztály nem valósít meg interface-t.

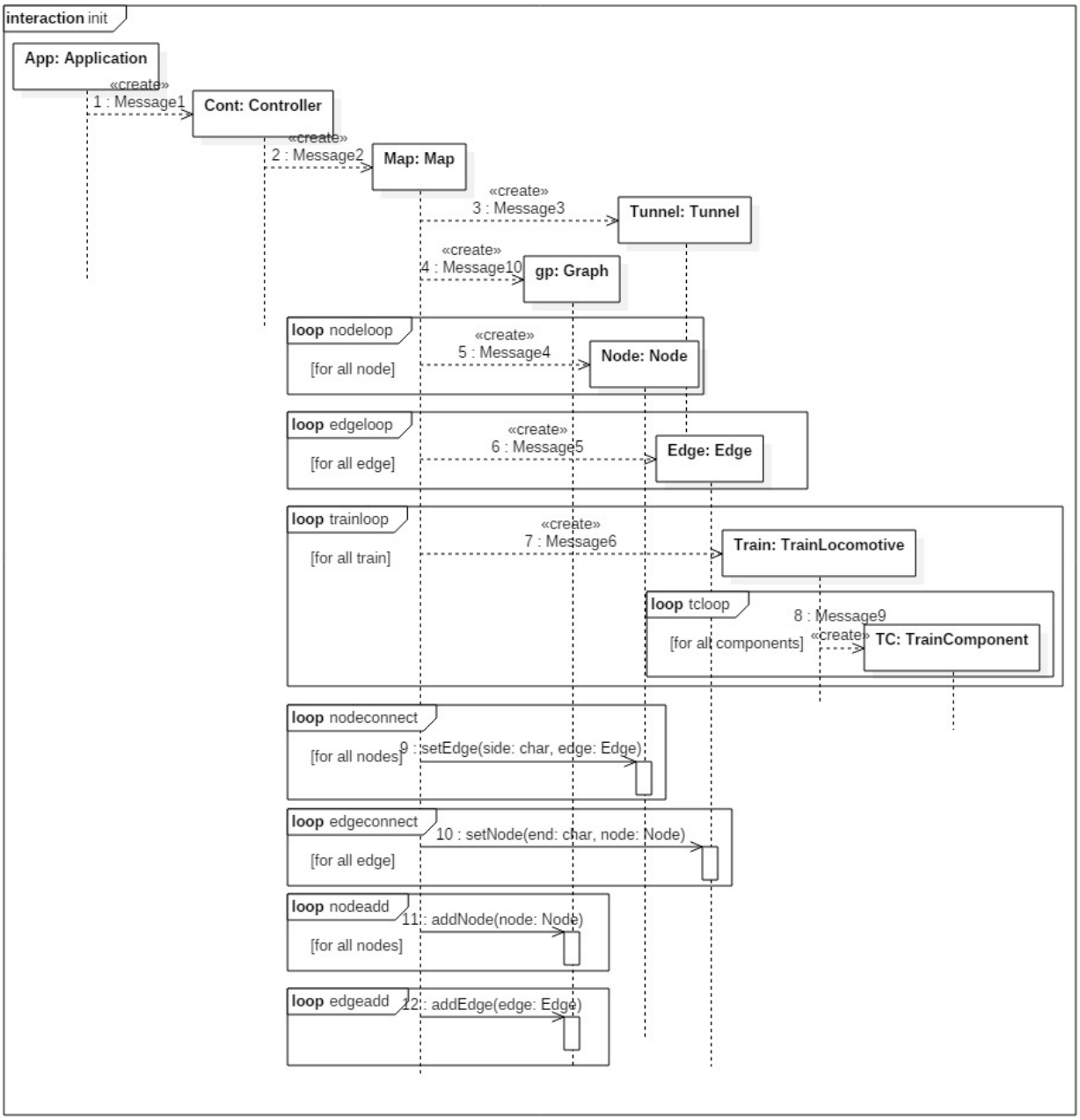
- Attribútumok

  Nincsenek publikus attribútumok.

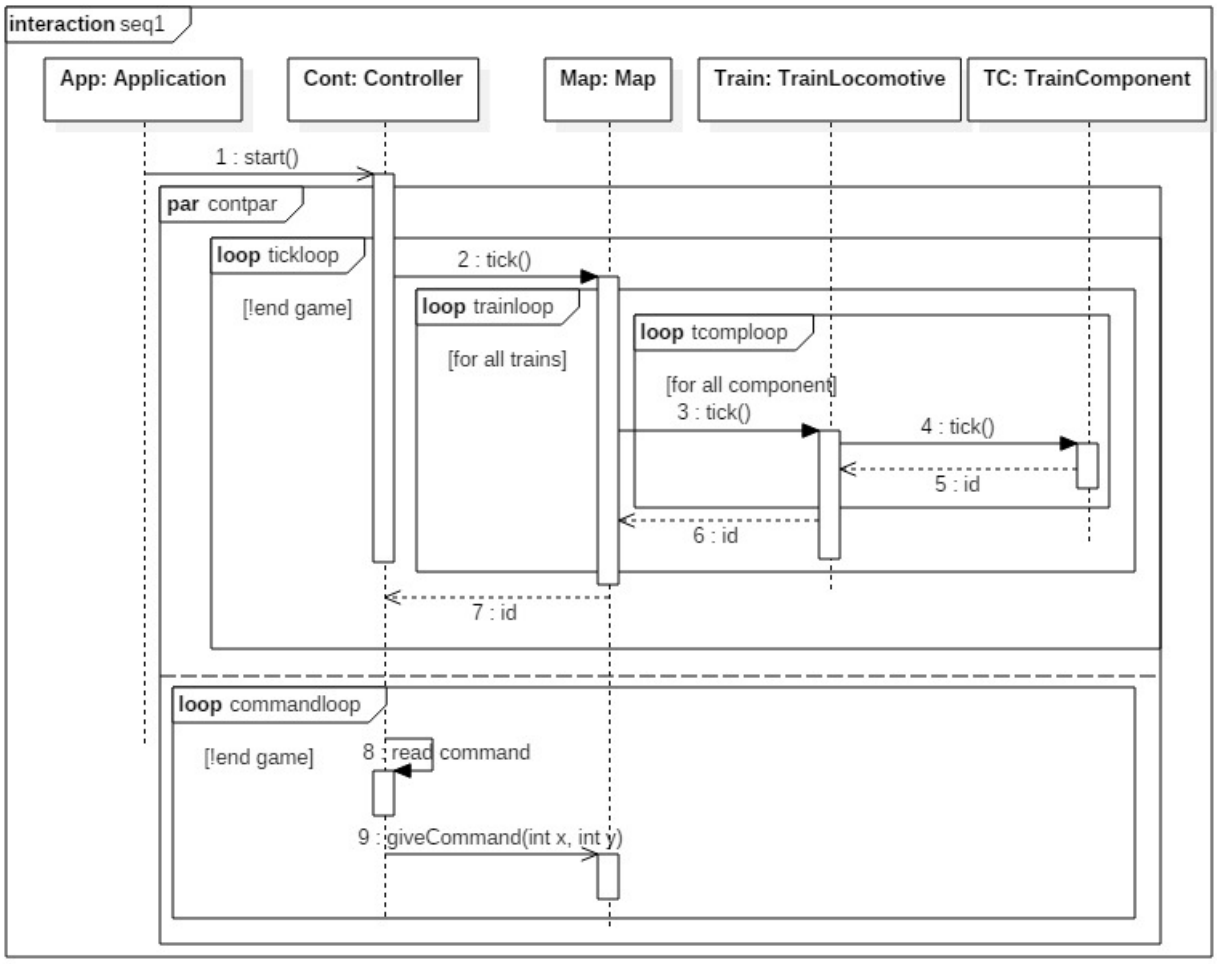
- Metódusok

- bool isEmpty(): Visszaad egy logikai értéket, hogy az adott vagon üres vagy sem.
- bool tick(): A TrainComponent osztályból örökölt metódus.
- String getType(): A TrainComponent osztályból örökölt metódus.
- int getX(): A TrainComponent osztályból örökölt metódus.
- int getY(): A TrainComponent osztályból örökölt metódus.

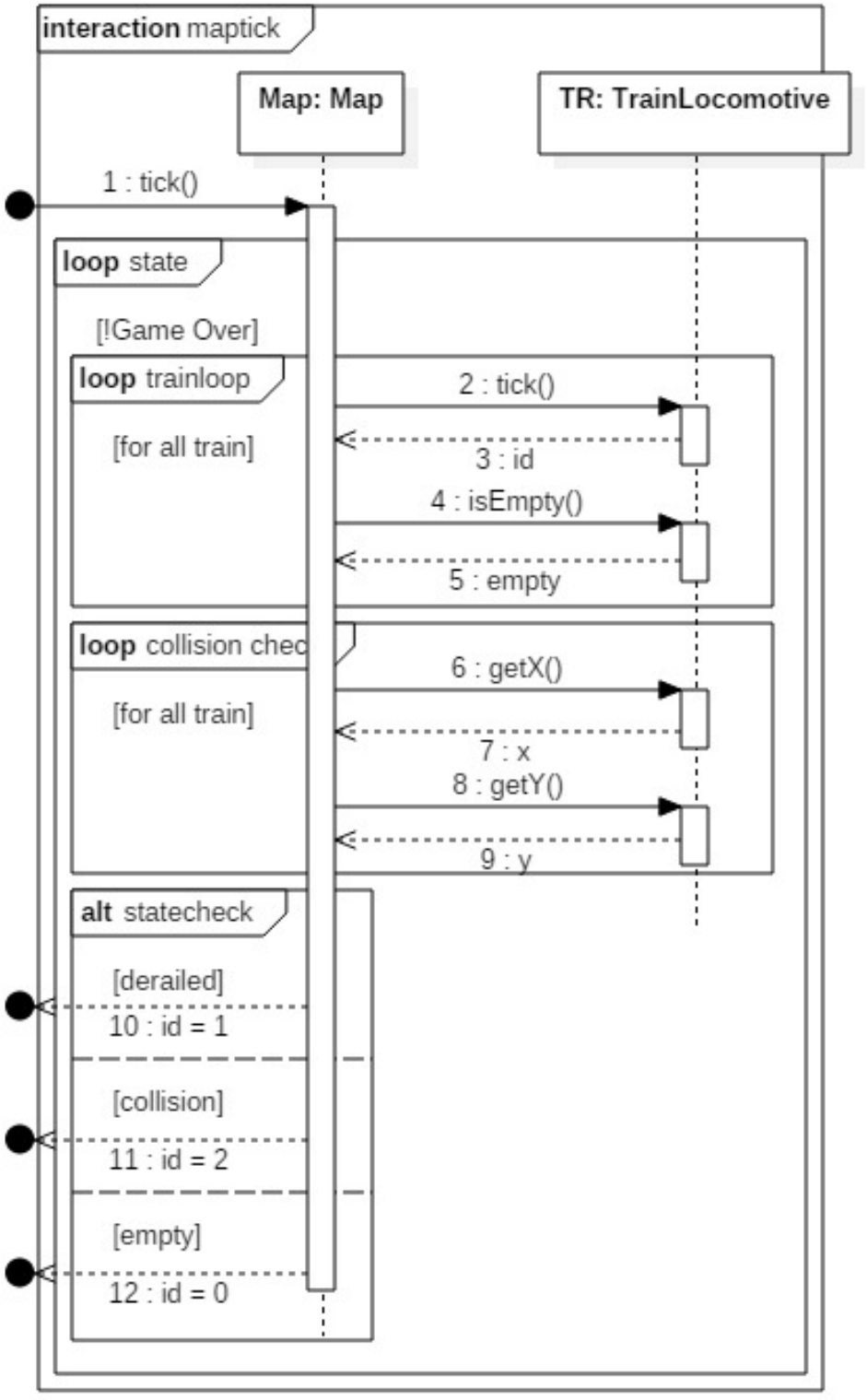
3.4. Szekvencia diagramok



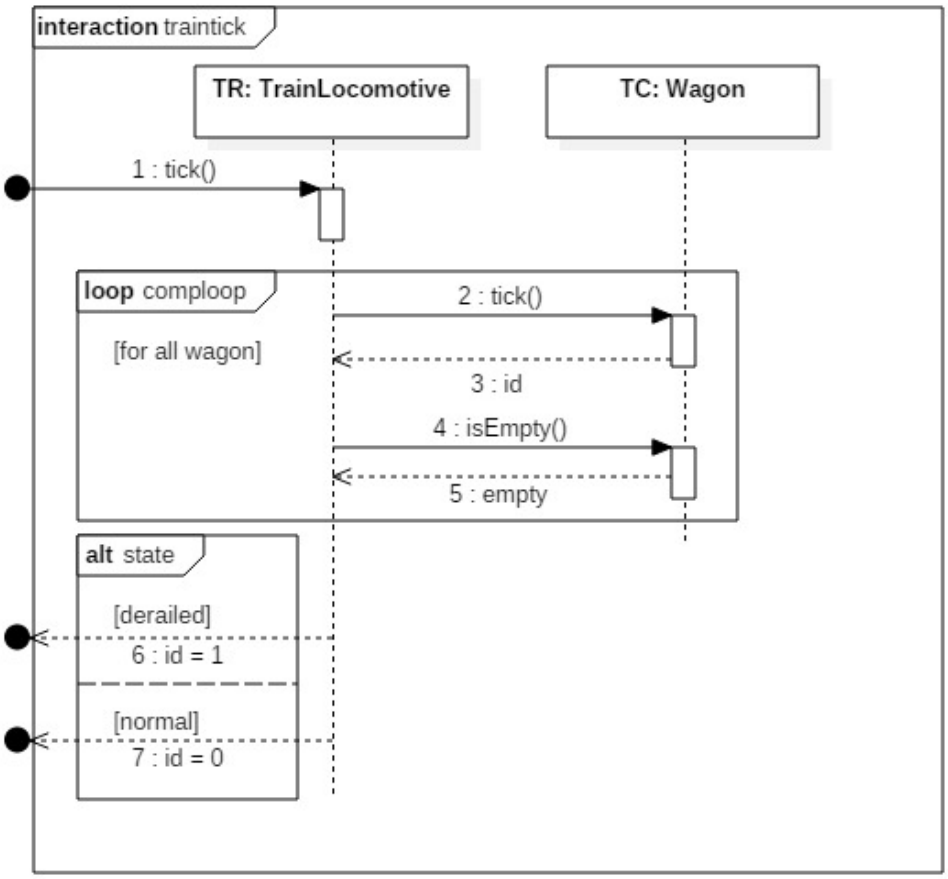
3.2. ábra. Init szekvencia



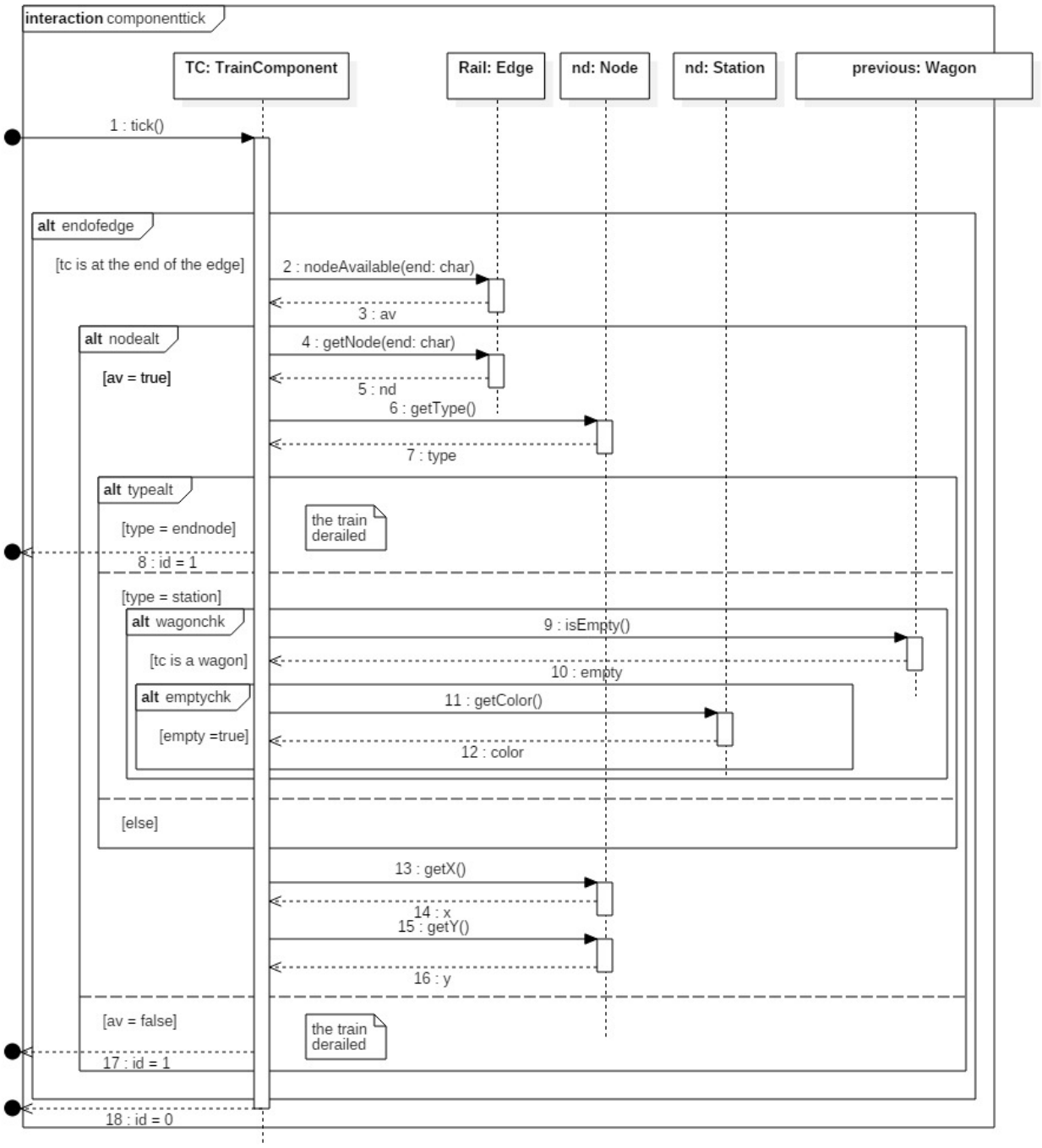
3.3. ábra. Tick szekvencia átfogó képe



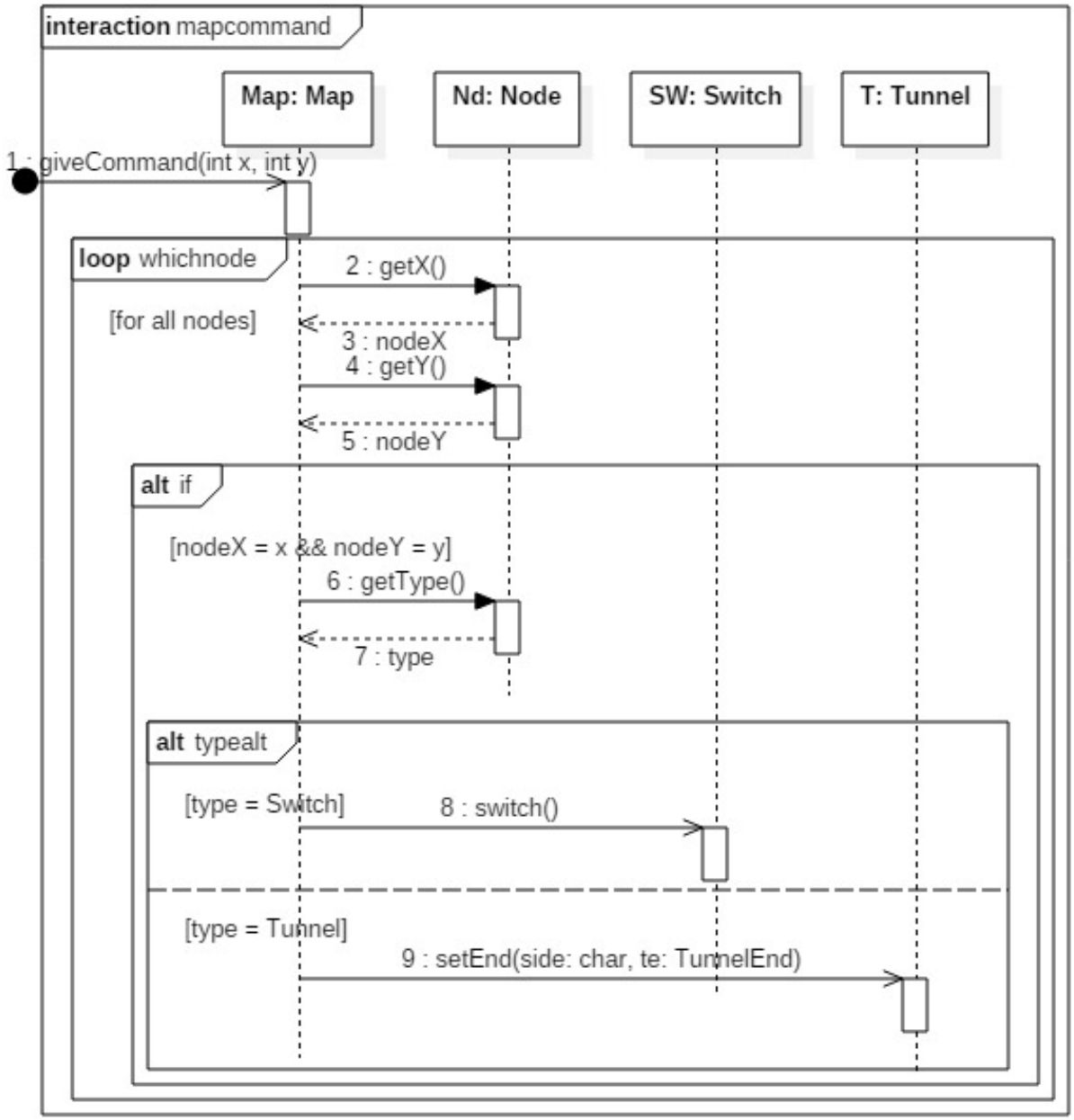
3.4. ábra. Map tick metódusának átfogó képe



3.5. ábra. Train tick metódusának átfogó képe

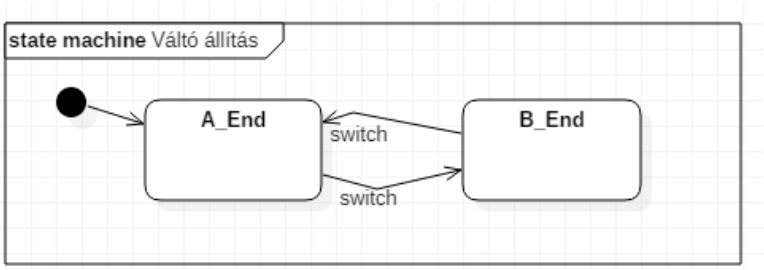


3.6. ábra. TrainComponent tick metódusának átfogó képe

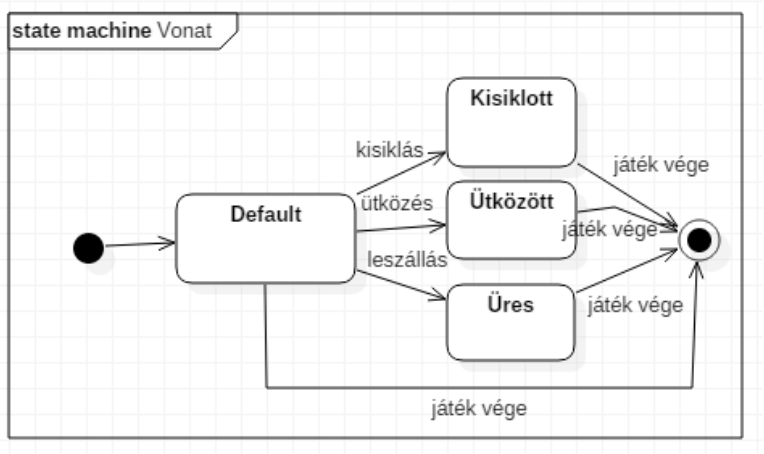


3.7. ábra. Map giveCommand metódusának átfogó képe

3.5. State-chartok



3.8. ábra. Váltó állapotdiagramja



3.9. ábra. Vonat állapotdiagramja

3.6. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017.02.22. 21:00	1 óra	**Dombai** **Erős** **Székely** **Szilágyi**	Értekezlet. Közös koncepció megbeszélése, feladatok elosztása. Döntés: Dombai és Erős elkészíti az osztálydiagramot. Következő megbeszélés szombaton (2017-02-25).
2017.02.23. 20:00	2 óra	**Dombai**	3.2 - statikus struktúra diagram készítésének elkezdése.
2017.02.23. 22:00	2 óra	**Erős**	3.2 - statikus struktúra diagram készítésének befejezése.
2017.02.24. 16:00	1 óra	**Dombai**	3.5 - 3.2, 3.3 ábra elkészítése.
2017.02.24. 19:00	2 óra	**Dombai**	3.5 - 3.4, 3.5, 3.6 ábra elkészítése, 3.3 javítása.

Kezdet	Időtartam	Résztvevők	Leírás
2017.02.25. 10:00	1 óra	**Márton**	3.3 alpont elkezdése.
2017.02.25. 14:00	2 óra	**Márton**	3.3 alpont folytatása.
2017.02.25. 15:00	1 óra	**Szilágyi**	3.1 alpont elkezdése.
2017.02.25. 20:00	1 óra	**Dombai** **Márton** **Székely** **Szilágyi**	Megbeszélés. Eddig elkészített pontok ismertetése a többieknek, javításra szoruló pontok megbeszélése.
2017.02.25. 21:00	1 óra	**Dombai**	3.2, 3.4 javítása.
2017.02.25. 21:00	1 óra	**Szilágyi**	3.1 javítás
2017.02.26. 9:00	1 óra	**Márton**	3.3 javítása, kiegészítése.
2017.02.26. 21:00	1 óra	**Szilágyi**	3.1 javítása.
2017.02.27. 00:00	1 óra	**Székely**	3.5 elkészítése.

4. Analízis modell kidolgozása 2

4.1. Objektum katalógus

4.1.1. Locomotive

A mozdonyok a vasúton haladnak és húzzák a hozzájuk kapcsolt vagonokat.

4.1.2. Rail

A síneken haladnak a mozdonyok és a vagonok.Az állomások, váltók és a vasút részei sínekkel vannak összekötve.

4.1.3. Siding

A vakvágány egy olyan sín melynek nincsen folytatása, a végén a vonat és a vagonok kisiklanak.

4.1.4. Station

Az állomásoknál tudnak leszállni a vagonokról az utasok, mikor a vonat elhalad mellettük.

4.1.5. Switch

A váltók segítségével lehet a vonatokat irányítani azzal, hogy melyik sínen haladjanak tovább

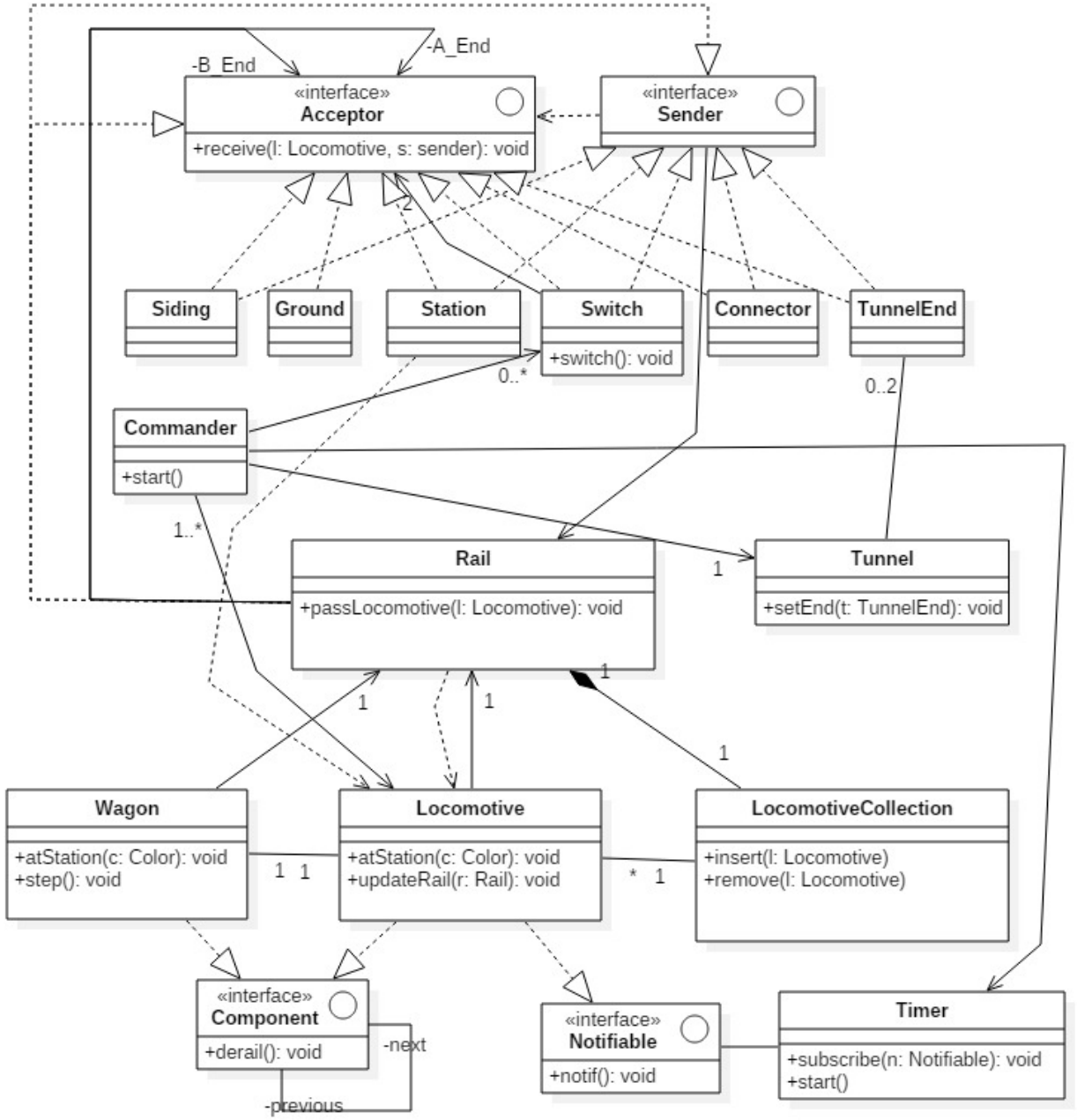
4.1.6. Tunnel

A játékban van egy alagút amiben sín fut. A vonatok belemennek az alagútba, majd elhagyják azt.

4.1.7. Wagon

A vagonok össze lehetnek kötve egymással és egy mozdony húzza őket. A vagonokban utaznak az utasok, akik a megfelelő állomáshoz érve leszállhatnak.

4.2. Statikus struktúra diagramok



4.1. ábra. Statikus struktúra diagram

4.3. Osztályok leírása

Az egyes osztályok örökölt metódusainak leírása minden esetben azok ősénél található.

4.3.1. Acceptor «interface»

- Felelősség
  Interface, definiálja a mozdony fogadására képes eszközöket.
- Ősosztályok

  Nincs ős.
- Metódusok

  – void receive(l: Locomotive) : Az interface-t megvalósító objektumnak alkalmasnak kell lennie mozdony fogadására, amit paraméterként kap meg ebben a metódusban. Saját szerepének megfelelően kezeli a fogadott mozdonyt.

4.3.2. Commander

- Felelősség
  A kezelőtől ciklusban olvassa be a parancsokat. A parancsban szereplő váltó, vagy alagút konkrét példányát az asszociáción keresztül éri el, és meghívja megfelelő metódusát.
- Ősosztályok

  Nincs ősosztály.
- Interfészek

  Nem valósít meg interface-t.
- Attribútumok

  Nincsenek publikus attribútumok.
- Metódusok

  – void start(): Commander megkapja a vezérlést az App-tól, várjuk a parancsokat.

4.3.3. Component «interface»

- Felelősség

  Interface,a vonat különböző komponenseit definiálja, amelyek ki tudnak siklani. A vonat állhat egy mozdonyból, és az azt követő vagonokból, ezek mind megvalósítják a Component interface-t. Szerepe, hogy egy elem kisiklása esetén a teljes vonaton "végigfuthasson" a kisiklás.
- Ősosztályok

  Nincs ősosztály.

- Metódusok

  - void derail(): Vonatkomponens kisiklatása. Az előtte lévő komponenseket szintén kisiklatja.

4.3.4. Connector

- Felelősség
  Két sín összekötéséért felelős, mozdonyokat csupán továbbadja a következő Acceptor-nak. Így valósulhatnak meg a kanyarok.
- Ősosztályok
  Nincs ősosztály.
- Interfészek
  Acceptor, Sender
- Metódusok

4.3.5. Ground

- Felelősség
  A kisiklott vonatok fogadására szolgáló singleton föld osztály. Ide érkeznek a vakvágányon áthaladó mozdonyok.
- Ősosztályok

  Nincs ősosztály.
- Interfészek

  Acceptor
- Attribútumok

  Nincsenek publikus attribútumok.
- Metódusok
  Nem rendelkezik az implementált interface-től eltérő metódusokkal.

4.3.6. Locomotive

- Felelősség

  A Locomotive osztály egy mozdonyt (mint komponens) reprezentál. Ehhez lehet még vagonokat csatlakoztatni, amik együtt, egy jelre mozognak.
- Ősosztályok

  Nincs ősosztály.

- Interfészek

  Component

- Attribútumok

  Nincsenek publikus attribútumok.

- Metódusok

  - void atStation(c: Color): Amikor a mozdony beér egy állomásra, meghívódik a metódus, a paraméterben az aktuális állomás színével, azok az utasok akiknek az adott színű állomás volt az úticélja, leszállnak.
  - void updateRail(r: Rail): Beállítja a paraméterben kapott sínt mint aktuális sínpárt, amin a mozdony jelenleg halad.

4.3.7. LocomotiveCollection

- Felelősség
  Egy-egy sínen utazó mozdonyok kollekciója.

- Ősosztályok

  Nincs ősosztály.

- Interfészek

  Nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus attribútumok.

- Metódusok

  - void insert(l: Locomotive): A paraméterként kapott mozdonyt hozzáadja a kollekcióhoz.
  - void remove(l:Locomotive): A paraméterként kapott mozdonyt eltávolítja a kollekcióból.

4.3.8. Notifiable «interface»

- Felelősség
  Interface. A Timer-től eredő értesítéseket fogadó osztályok valósítják meg. A Timer-re való feliratkozás után az azok notif() metódusát hívja beállított időközönként. Lehet singleton.

- Ősosztályok
  Nincs ősosztály.

- Metódusok

  - void notif(): a Timer jelzi az implementornak, hogy a beállított késleltetés letelt.

4.3.9. Rail

- Felelősség
  A Rail osztály egy sínt reprezentál, amin a vonat közlekedik, ez a pálya egyéb elemeit köti össze egymással.
- Ősosztályok

  Nincs ősosztály.
- Interfészek Sender, Notifiable, Acceptor
- Attribútumok

  Nincsenek publikus attribútumok.
- Metódusok
  - void passLocomotive(l: Locomotive): Továbbadja a paraméterként kapott mozdonyt az irányának megfelelő következő Accreptor-nak.

4.3.10. Sender «interface»

- Felelősség
  Interface, definiálja a mozdony átadására képes eszközöket. Ismeretet biztosí
- Ősoor interfészhez.sztályot az Acceptk

  Nincs ősosztály.
- Metódusok
  Nem rendelkezik metódusokkal.

4.3.11. Siding

- Felelősség
  Vakvágányokat reprezentáló osztály. Beérkező mozdonyokat Ground-ra továbbítja.
- Ősosztályok

  Nincs ősosztály.
- Interfészek

  Acceptor, Sender
- Attribútumok

  Nincsenek publikus attribútumok.
- Metódusok

4.3.12. Station

- Felelősség
  A station egy állomást reprezentáló osztály. Színnel rendelkezik, ezt a beérkező mozdonyoknak átadja, így valósul meg a leszállás.
- Ősosztályok

  Nincs ősosztály.
- Interfészek

  Acceptor, Sender
- Attribútumok

  Nincsenek publikus attribútumok.
- Metódusok

4.3.13. Switch

- Felelősség
  A Switch egy váltót reprezentáló osztály. A váltóval a sínek között lehet váltani. Az állása a játékostól függ, ő irányítja, ezzel befolyásolva a vonat mozgását.
- Ősosztályok

  Nincs ősosztály.
- Interfészek

  Acceptor, Sender
- Attribútumok

  Nincsenek publikus attribútumok.
- Metódusok
  - void switch(): A váltás műveletét megvalósító metódus.

4.3.14. Timer

- Felelősség
  Időzítés. A beállított késleltetés leteltével a rá feliratkozott, Notifiable interfészt megvalósító objektumok notif() metódusát hívja. metódusát.
- Ősosztályok

  Nincs ősosztály.

- Interfészek

  Nem valósít meg interface-t.

- Attribútumok
  Nincsenek publikus attribútumok.

- Metódusok

  - void start(n: Notifiable): Elindul a Timer, előre definiált késleltetésenként hívja minden rá feliratkozott objektum notif függvényét.

4.3.15. Tunnel

- Felelősség

  A Tunnel egy pályán elhelyezhető alagutat reprezentáló osztály. A vonatok át tudnak haladni ezen az alagúton. A játékos jogosultságai közé tartozik, hogy áthelyezheti az alagút egyes végeit, így változtatva a pályát.

- Ősosztályok

  Nincs ősosztály.

- Interfészek
  Nem valósít meg interface-t.

- Attribútumok

  Nincsenek publikus attribútumok.

- Metódusok

  - void setEnd(t: TunnelEnd): Beállítható az alagút egy vége.

4.3.16. TunnelEnd

- Felelősség

- Olyan pontokat reprezentál ahova alagútszáj helyezhető.

- Ősosztályok

  Nincs ősosztály.

- Interfészek

  Acceptor, Sender

- Attribútumok

  Nincsenek publikus attribútumok.

- Metódusok

4.3.17. Wagon

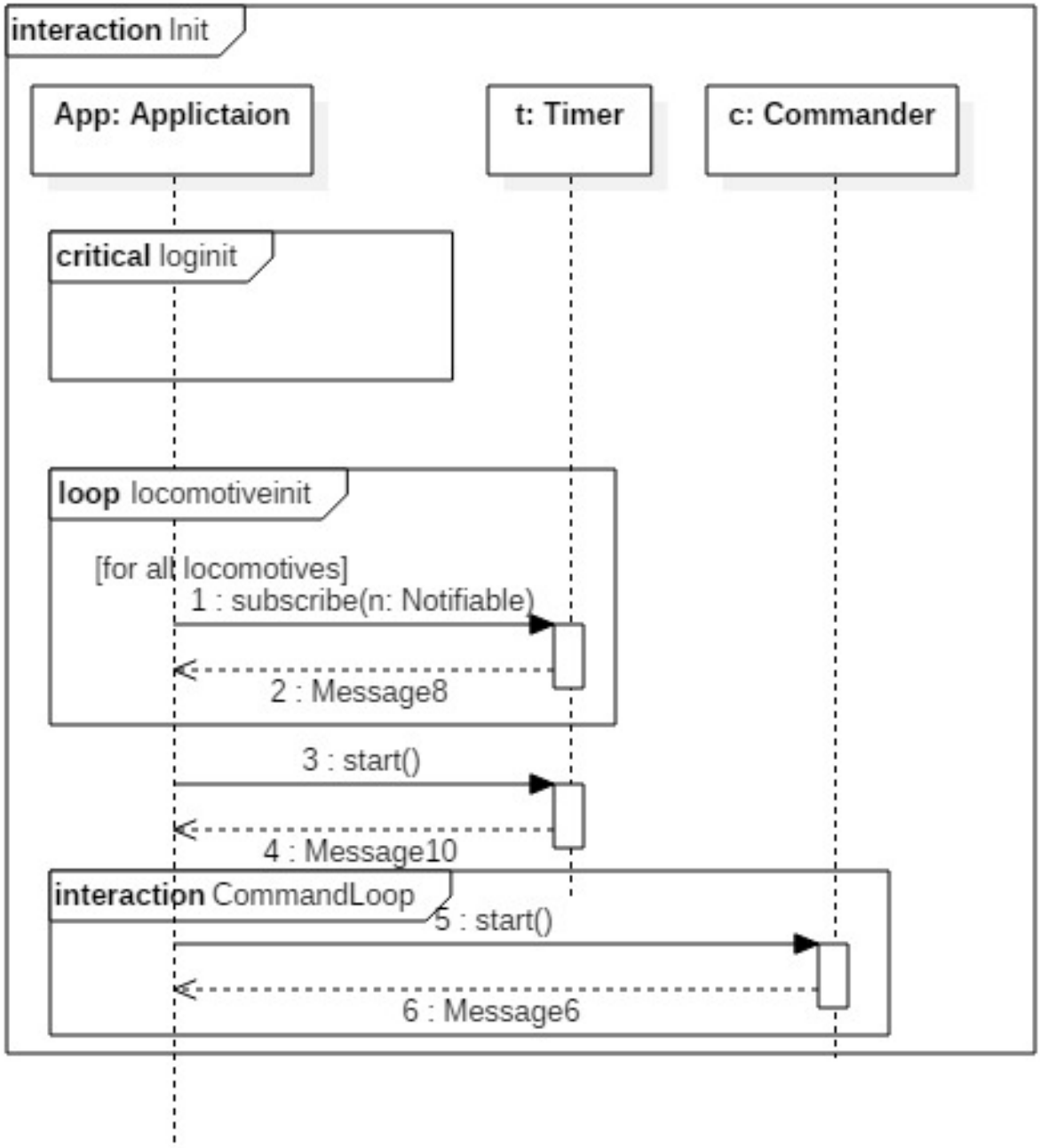
- Felelősség
  A Wagon osztály a mozdonyhoz csatlakoztatható vagont reprezentálja. A vagon a mozdonyhoz hasonlóan egy komponens, ezeket egymás után lehet kötni. A vagonban utaznak az utasok. Saját színnel rendelkezik, így az azonos színű állomáson az utasok leszállhatnak, ha ez lehetséges.
- Ősosztályok

  Nincs ősosztály.
- Interfészek
  Component
- Attribútumok

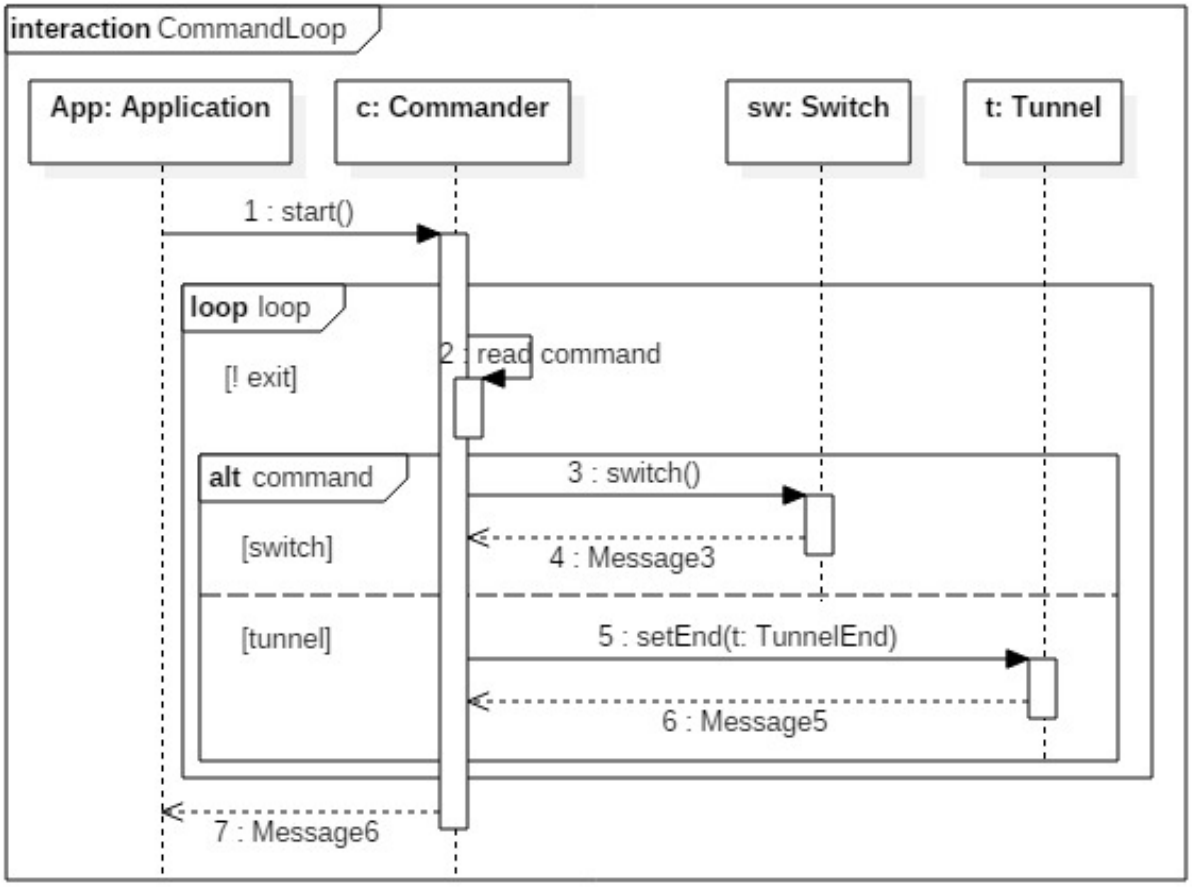
  Nincsenek publikus attribútumok.
- Metódusok
  - void atStation(c: Color):Amikor a mozdony beér egy állomásra, meghívódik a metódus (a vonatban előtte álló komponens hívja meg), a paraméterben az aktuális állomás színével, azok az utasok akiknek az adott színű állomás volt az úticélja, leszállnak.
  - void step(): Továbblépteti a vagont.

4.4. Szekvencia diagramok

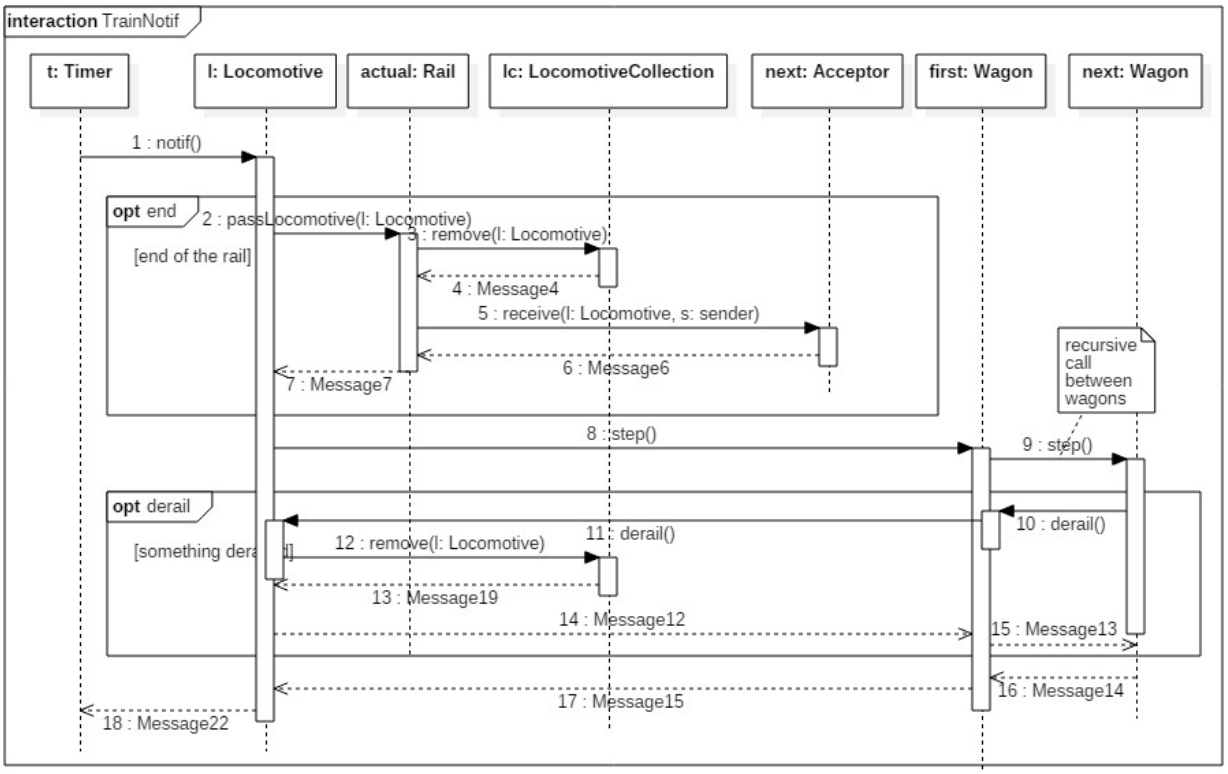
A "loginit" feladata a terepasztalt alkotó speciális pontokból, sínekből valamint vonatokból álló "hálózat" megfelelő összekapcsolása. Erre két megoldás elképzelhető: előre definiálunk egy pályát a program kódjában, vagy megfelelő formális nyelvű dokumentumból olvassuk be azt, majd építjük fel.



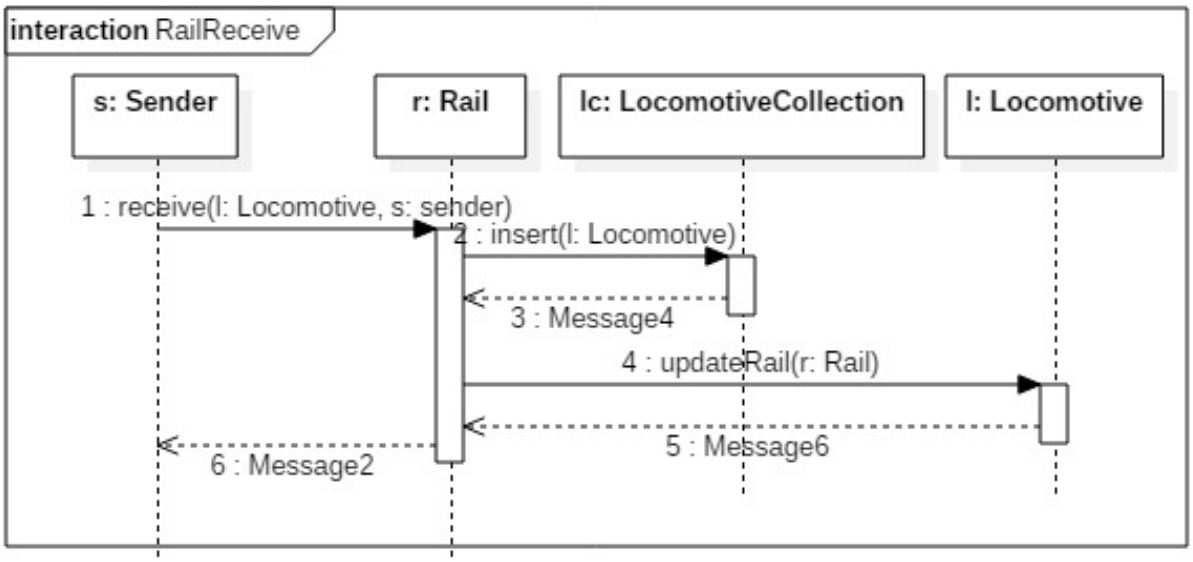
4.2. ábra. Init szekvencia diagram



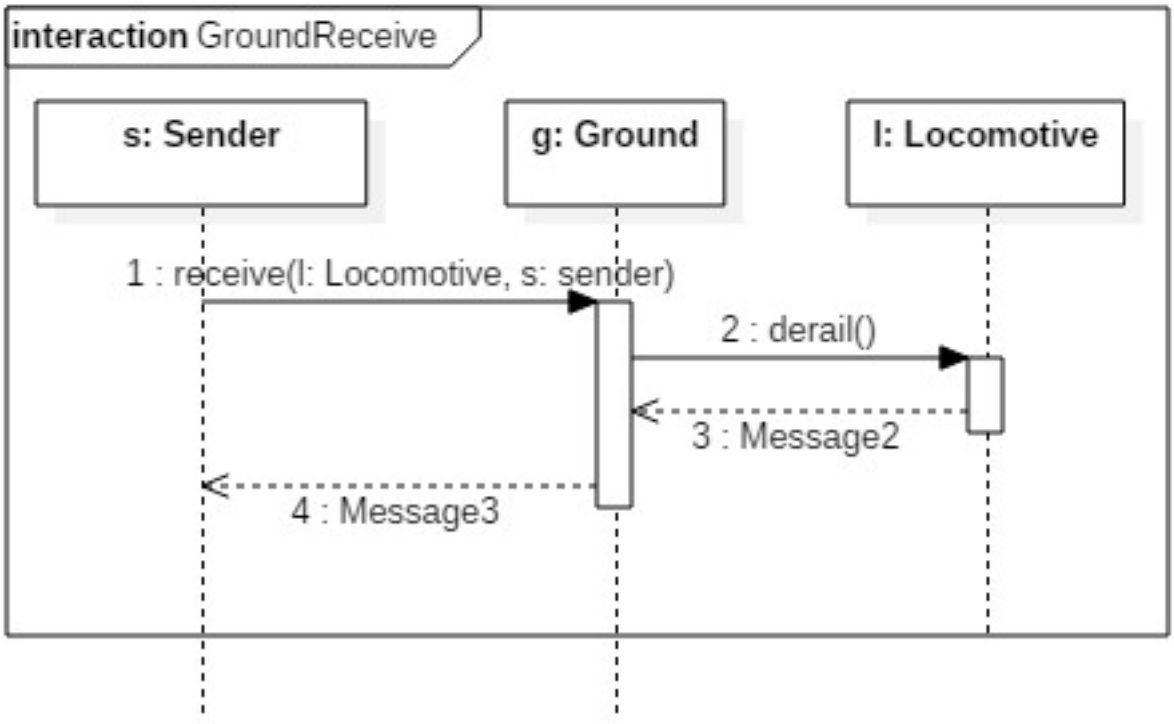
4.3. ábra. Vezérlés



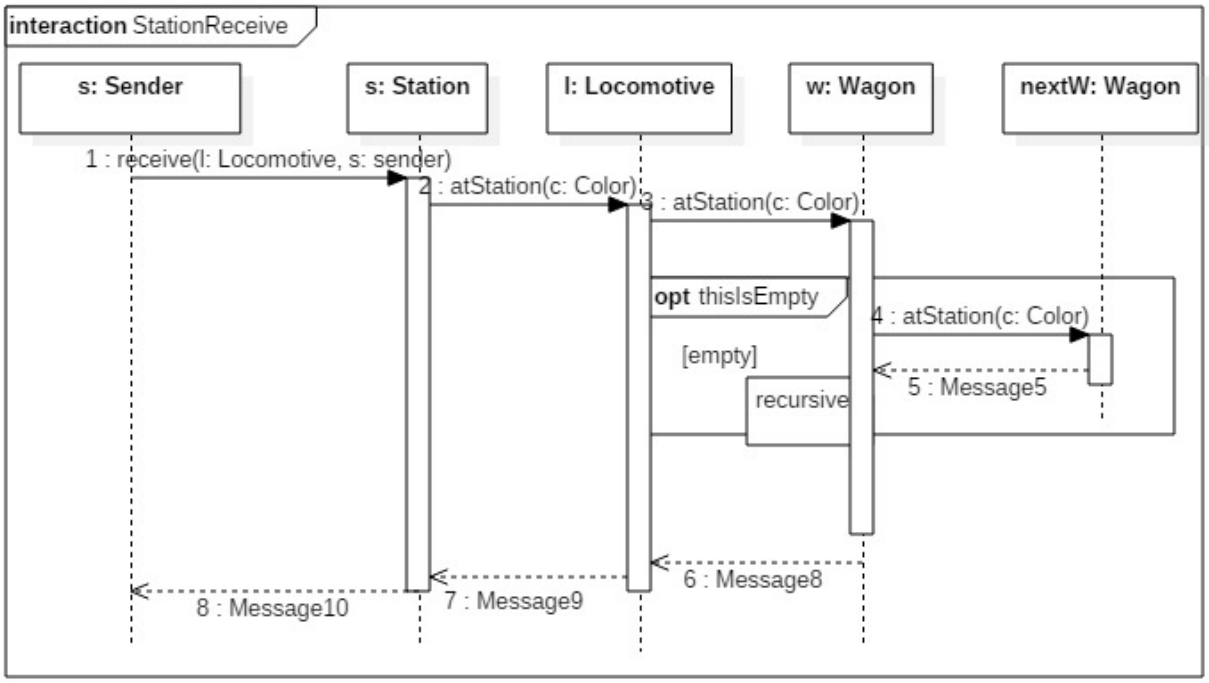
4.4. ábra. Locomotive Notif szekvenciája



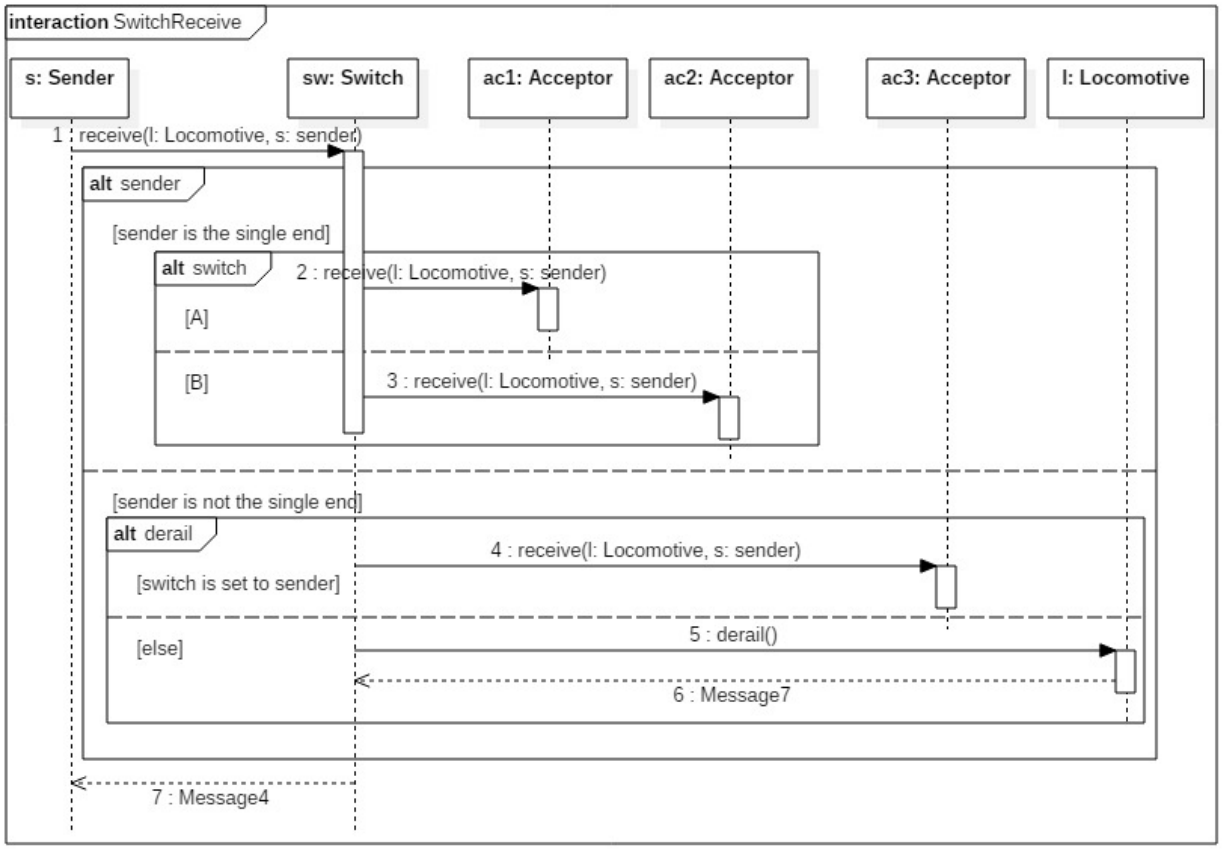
4.5. ábra. Rail léptetési szekvenciája



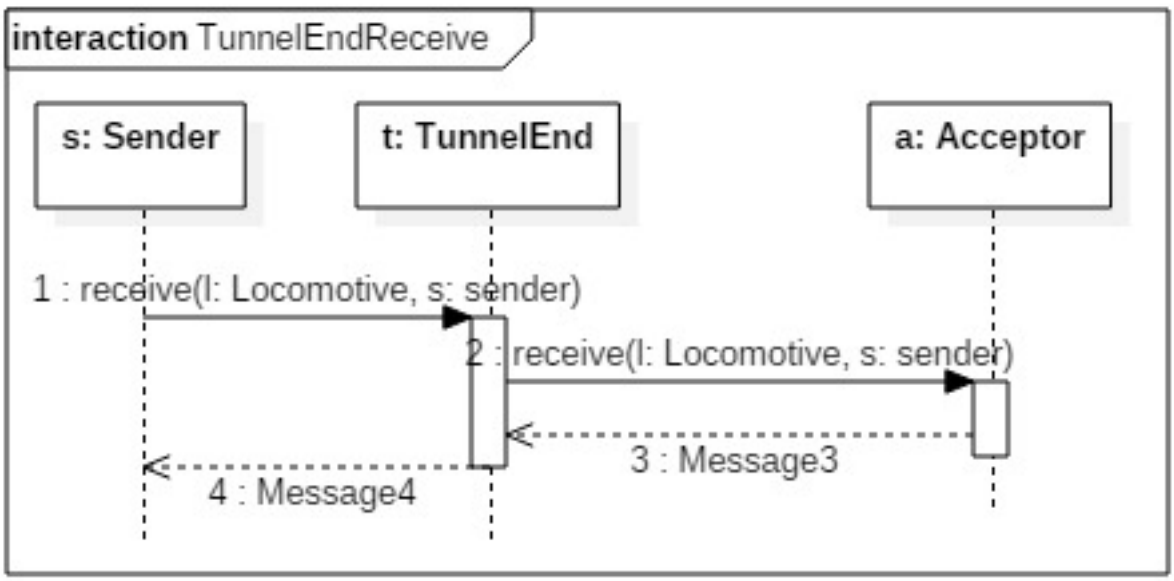
4.6. ábra. Ground szekvenciája



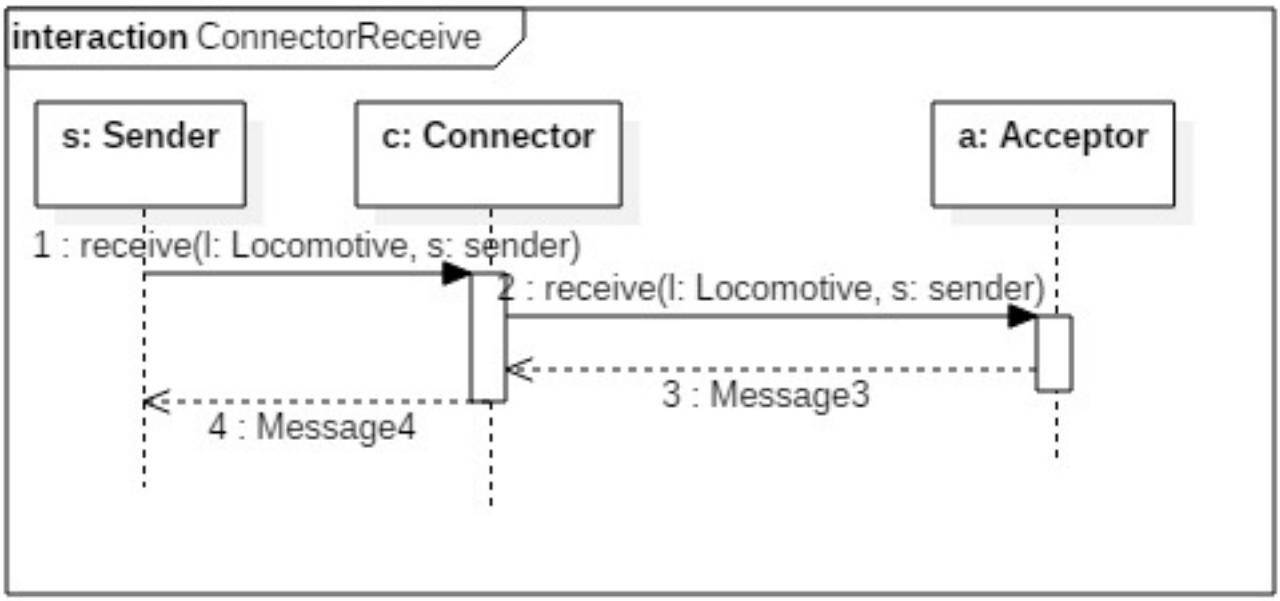
4.7. ábra. Station szekvenciája



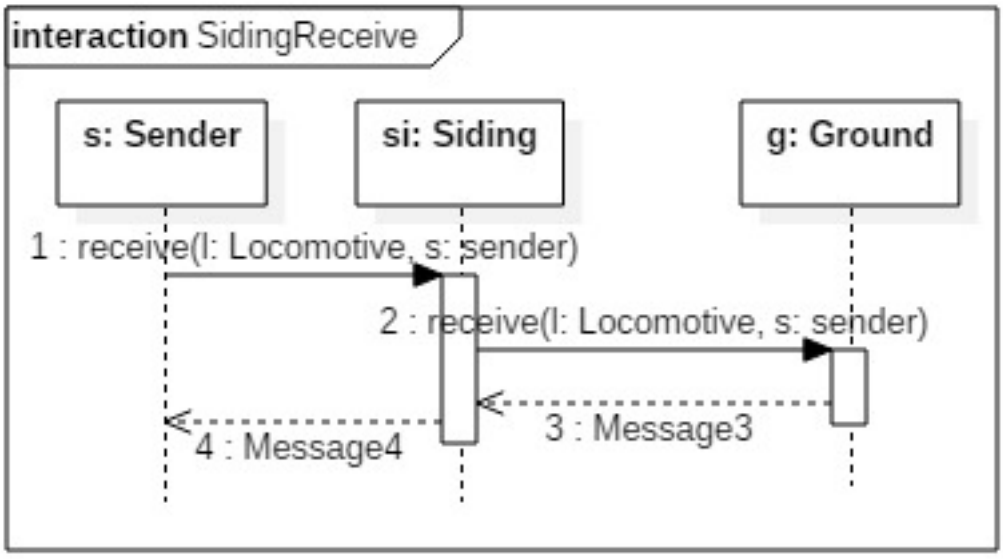
4.8. ábra. Switch szekvenciája



4.9. ábra. TunnelEnd szekvenciája



4.10. ábra. Connector szekvenciája



4.11. ábra. Siding szekvenciája

4.5. State-chartok

Nincs olyan state-chart amit értelme lenne ábrázolni.

4.6. Kiegészítések

A feladatot a következő információkkal pontosítjuk: a vonatról való leszállás akkor lehetséges, amikor a mozdony beérkezik az állomásra. A leszálláshoz szükséges többi feltétel természetesen nem változik. Az, hogy az alagút egyik vége van csak kijelölve nem befolyásolja a vonatok mozgását.

4.7. Napló

Kezdet	**Időtartam**	**Résztvevők**	**Leírás**
2010.03.01. 22:00	1 óra	**Dombai**	2.2 elkezdése
2010.03.01. 23:00	1 óra	**Erős**	2.2 befejezése
2010.03.03. 17:00	2 óra	**Dombai**	2.4
2010.03.04. 16:00	2 óra	**Székely**	2.2, 2.4 ellenőrzés. Javaslatok készítése.
2010.03.04. 18:00	1 óra	**Székely**	4.3 elkezdése
2010.03.04. 15:00	1 óra	**Szilágyi**	4.1
2010.03.04. 19:00	2 óra	**Márton**	4.3 befejezése
2010.03.06. 06:00	1 óra	**Dombai**	Ellenőrzés, javítások.

5. Szkeleton tervezése

5.0. Kiegészítések, változtatások

Alagút csak egy sín két vége közé építhető. A speciális pontok csak két sínt köthetnek össze. A sínek legalább olyan hosszúak hogy 3 step metódusból ne lehessen átérni.

Mivel jelentős változások történtek az osztályok szerepében, ezért ez a dokumentum is tartalmazza az osztálydiagramot.

Az új függvények és szerepváltozások gyors bemutatása:

5.0.1. Component

Korábbi Acceptor és Sender interfész összevonása és kiegészítése. getNext() metódusa paraméterként megkapja, hogy melyik vonatelem kéri a következő elemet, valamint hogy az melyik elemről került a jelenlegire. operateOn() metódusa saját magára jellemző módon módosítja a mozdony (ill. ezzel a vonat tulajdonságait). getCollection() visszaadja a pályaelemen lévő vonatelemeket.

5.0.2. Collection

Egykori LocomotiveCollection. Mostmár nem csak mozdonyokat, hanem vagonokat is tárol. Megvalósítja az ütközésdetektálást, ezt a függvényt a mozdony step metódusa hívja, visszatérési az a mozdony amivel ütközött, vagy null. myComponentAtEnd-et hívja a komponensek step metódusa, megadja hogy a vonat elérte-e a pályaelem végét, ha igen true-val tér vissza, ha nem akkor növeli a pozícióját.

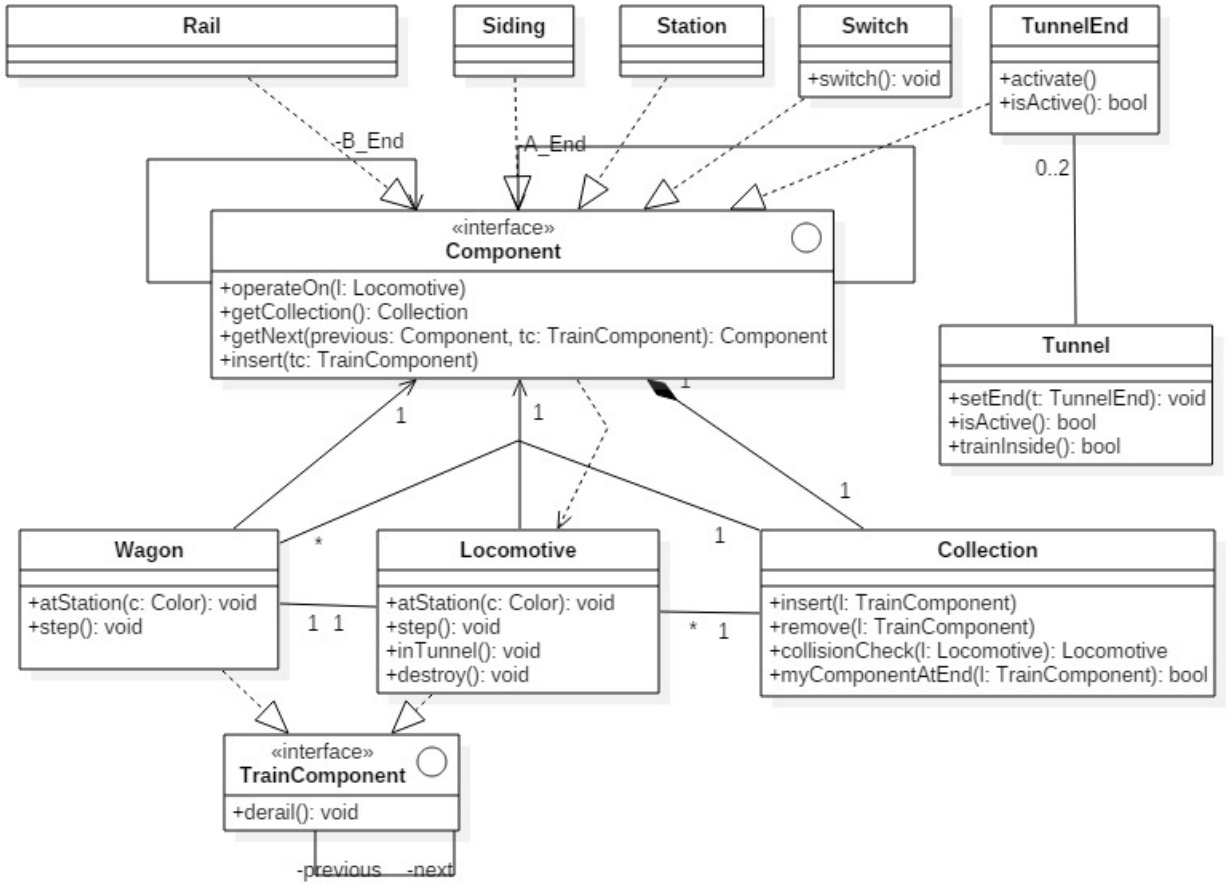
5.0.3. Locomotive

Hozzáadásra került inTunnel metódus, ami gyakorlatilag egy flag-et állít attól függően hogy alagútba be vagy ki lép a vonat. destroy hívódik ha a mozdony egy másik mozdonnyal ütközött.

5.0.4. Switch

Switch metódus hívása esetén minden olyan vonatelemet kisisklat ami rajta van (mivel hossza valószínűleg zérus lesz, ezért ez gyakorlatilag egy elemet jelent).

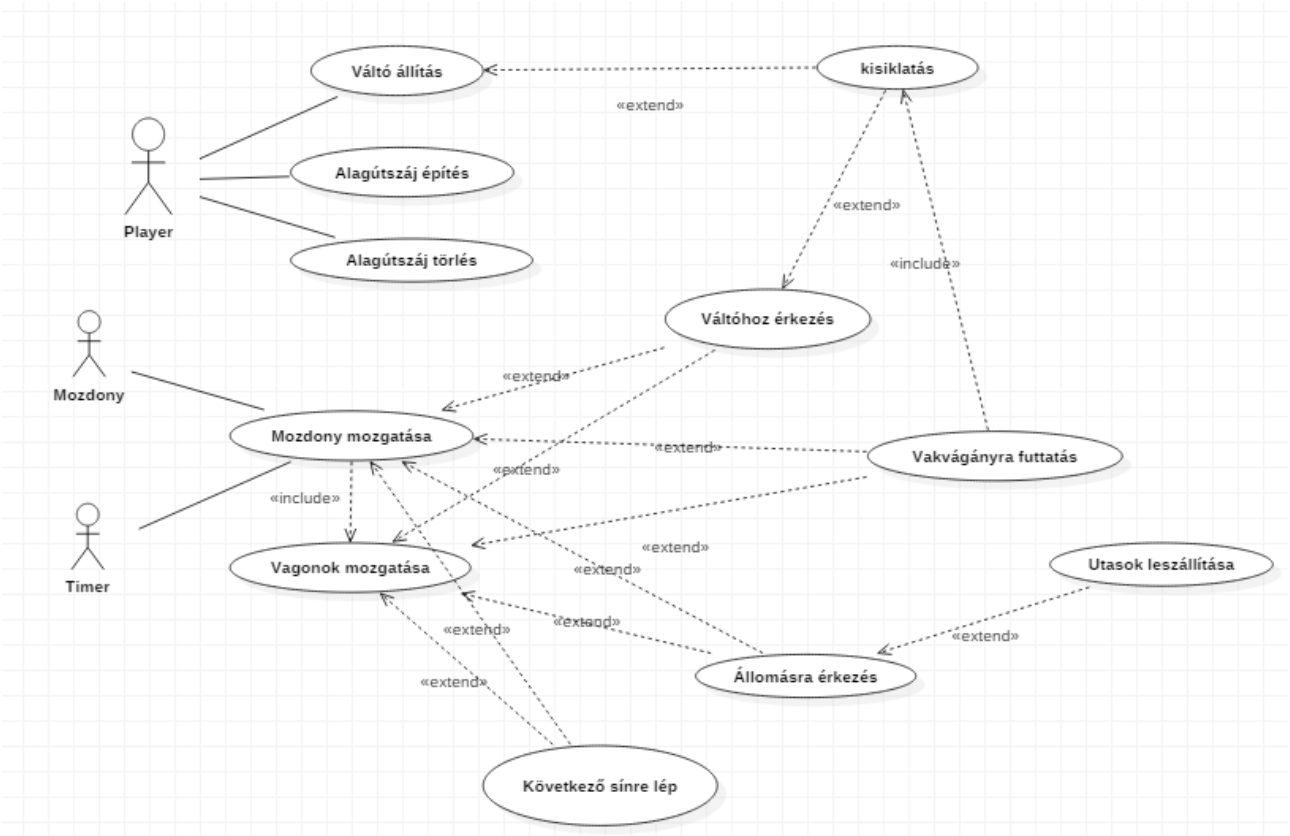
A többi változatás és azok szerepe egyszerűen kikövetkeztethető a szekvenciadiagramokból.



5.1. ábra. Javított osztálydiagram

5.1. A szkeleton modell valóságos use-case-ei

5.1.1. Use-case diagram



5.2. ábra. Use-case diagram

5.1.2. Use-case leírások

Use-case neve	**Alagútszáj építés**
Rövid leírás	Az erre megjelölt különleges helyen létrehozza az alagútszájat.
Aktorok	Player
Forgatókönyv	Ahol alagútszáj építése lehetséges, és a maximális kettőnél kevesebb van felépítve, akkor építésre kerül.

Use-case neve	**Alagútszáj törlés**
Rövid leírás	A már felépített alagutszájat eltávolítjuk a pályáról.
Aktorok	Player

Forgatókönyv	A játékos olyan helyet jelöl ki, ahol már van egy felépített alagútszáj, akkor az eltávolításra kerül.

Use-case neve	**Váltó állítás**
Rövid leírás	A váltó állapota az eddigi beállítás ellentettjére módosul.
Aktorok	Player
Forgatókönyv	Ha a játékos interakcióba kerül a váltóval és a váltó eddig az első kimenetre továbbította a vonatot, akkor az átállítás után a másodikra fogja, és fordítva.

Use-case neve	**Mozdony mozgatása**
Rövid leírás	A mozdony halad tovább a pályán.
Aktorok	Mozdony,Timer
Forgatókönyv	Ellenőrizzük, hogy egy pályaelem végéhez értünk-e, ha igen akkor lekérjük a következőt és átlépünk rá, ha nem, akkor az aktuálison haladunk tovább, amennyiben nem történt ütközés.

Use-case neve	**Vagonok mozgatása**
Rövid leírás	A mozdony húzza maga mögött a vagonokat.
Aktorok	Mozdony,Timer
Forgatókönyv	A mozdony húzza a maga mögött lévő kocsit, a kocsik pedig mind a mögöttük lévőket, ellenőrizzük azt is, hogy nincs-e három egymást követő vagon mind különböző pályaelemen, ekkor ugyanis kisiklik.

Use-case neve	**Vakvágányra futtatás**
Rövid leírás	A vonat vakvágányra fut.
Aktorok	Mozdony, Timer
Forgatókönyv	A mozdony, majd vagonok vakvágányra futnak, és kisiklanak.

Use-case neve	**Kisiklatás**
Rövid leírás	A vonat kisiklik, mert vagy vakvágányra futott, vagy alatta állították a váltót.

Aktorok	Mozdony, Timer
Forgatókönyv	A mozdony, majd vagonok kisiklanak, a játék véget ér.

Use-case neve	**Váltóhoz érkezés**
Rövid leírás	A vonat váltóhoz érkezik.
Aktorok	Mozdony,Timer
Forgatókönyv	A mozdony, majd vagonok egy váltóhoz érkeznek, és a váltó állásának megfelelően haladnak tovább. Ha rosszul haladna át valamelyik vonatelem a váltón, akkor kisiklik.

Use-case neve	**Állomásra érkezés**
Rövid leírás	A vonat beérkezik egy állomásra.
Aktorok	Mozdony
Forgatókönyv	A vonat megérkezk az állomásra, ahol ha van olyan utas, aki ilyen színű állomásra akart eljutni, akkor leszáll.

Use-case neve	**Utasok leszállítása**
Rövid leírás	Az ide eljutni kívánó utasok leszállnak.
Aktorok	Timer
Forgatókönyv	Állomás színe megegyezik a vagonnal, az utasok leszállnak, a játékosnak jóváíródnak a pontjai.

Use-case neve	**Következő sínre lépés**
Rövid leírás	Az előző pályaelemről továbblépve a sínre érkezik a vonat.
Aktorok	Mozdony, Timer
Forgatókönyv	Az állomást, váltót elhagyva a következő sínpáron halad tovább a vonat

5.2. A szkeleton kezelői felületének terve, dialógusok

A szkeleton menüvezérelt módon fog működni. A felhasználónak meg kell adnia a kívánt parancs számát, majd a program lefuttatja azt. A menü felépítése alább látható.

1. Váltó állítás

2. Alagút építés
    1 Alagútszáj megadása
    2 Alagútszáj törlése
3. Mozdony mozgatása
4. Mozdony állomásra ér
5. Pályaelemre lép

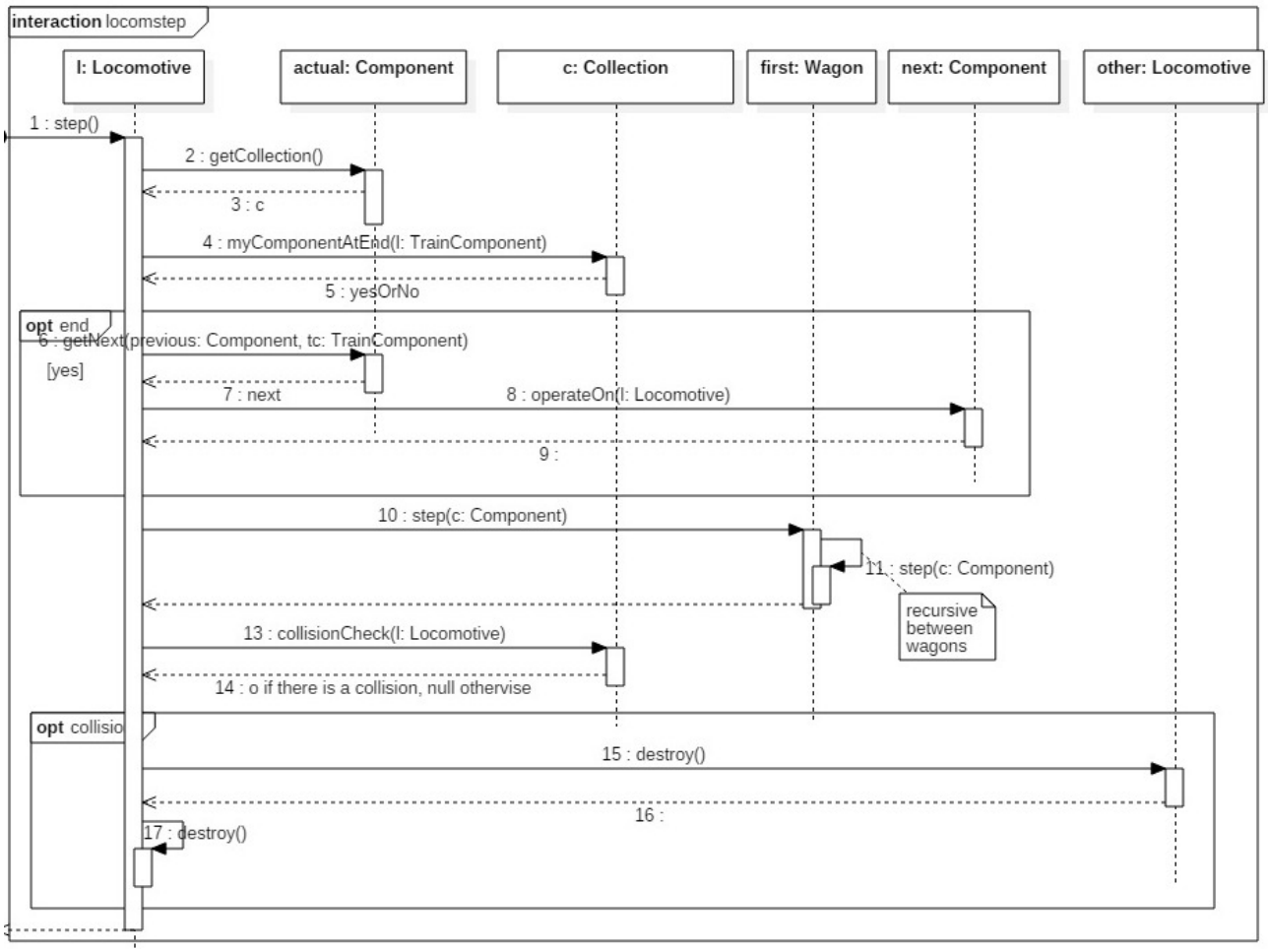
Az alábbi példa egy olyan interakciót mutat be, amikor két vonat összeütközik.

```
? Adja meg a parancs kódját: 3
>         [:Locomotive].step():
>             [actual:Component].getCollection()
>                 [c:Collection].myComponentAtEnd(I: TrainComponent):
?                         Elértük a végét? I/N: I
>                 [actual:Component].getNext(previous: Component,
                tc: TrainComponent)
                //Rekurzívan fut egészen a collection végéig
>                 [first:Wagon].step(c: Component)
>                 [c:Collection].collisionCheck(I: Locomotive):
?                         Történt ütközés? I/N: I
>                 [other:Locomotive].destroy()
>                 [:Locomotive].destroy()
```

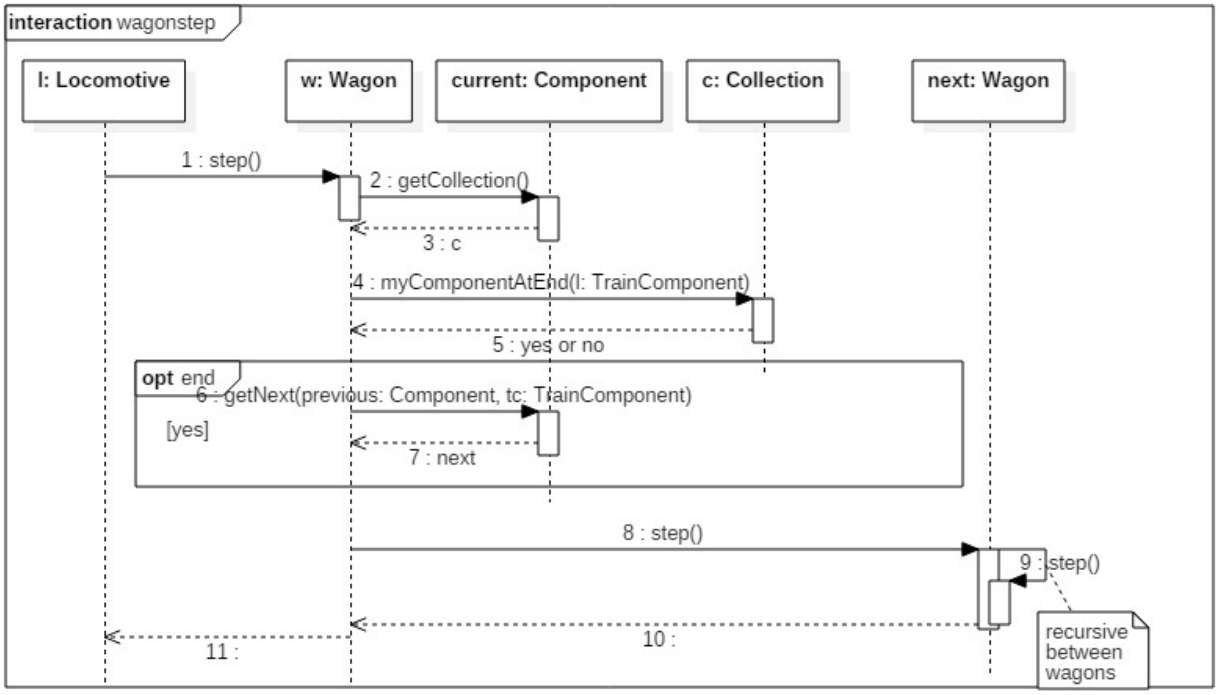
A sorok elején 1 karakter jelzi a sor típusát.

1. ? Kérdés
2. - Szöveges kimenet a konzolra
3. > bemenet

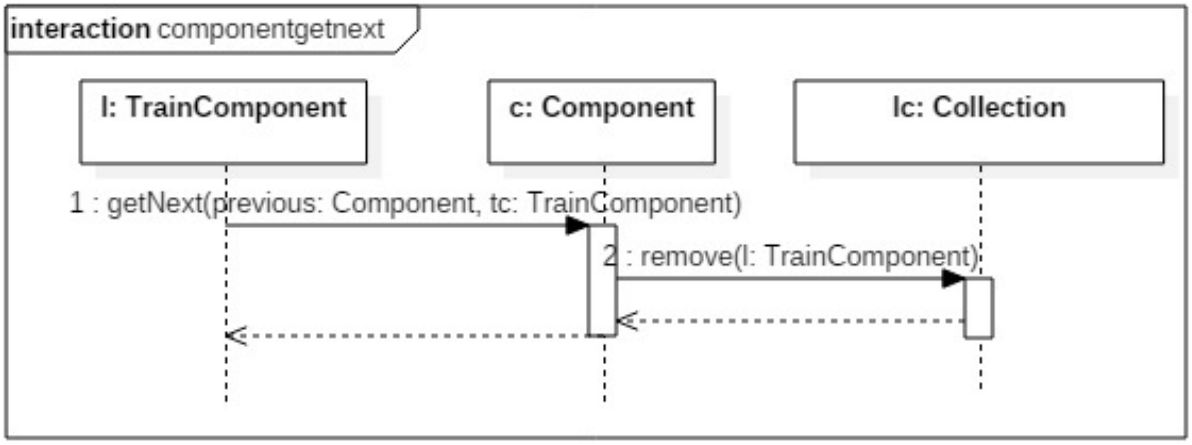
5.3. Szekvencia diagramok a belső működésre



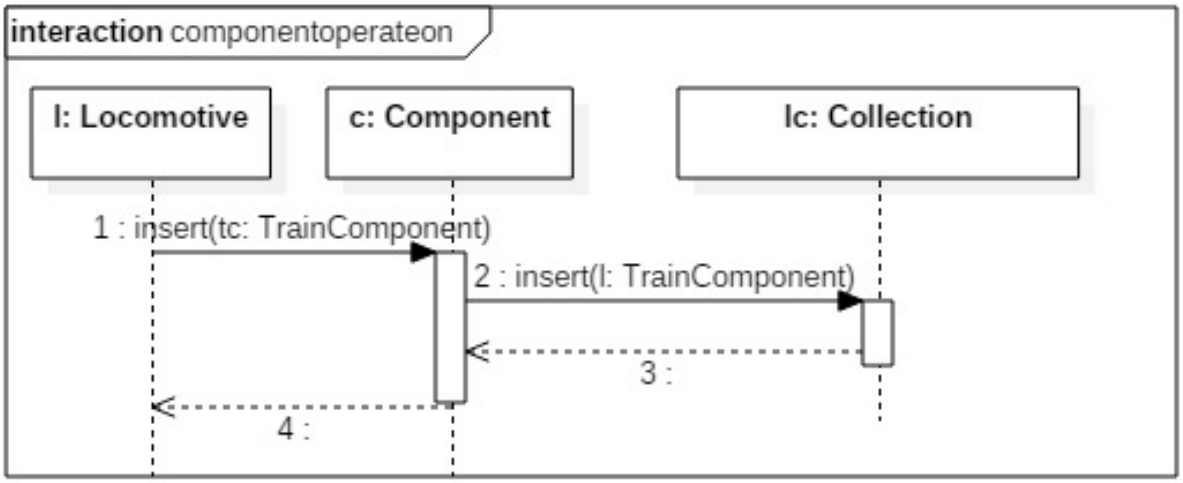
5.3. ábra. Mozdony mozgatása.



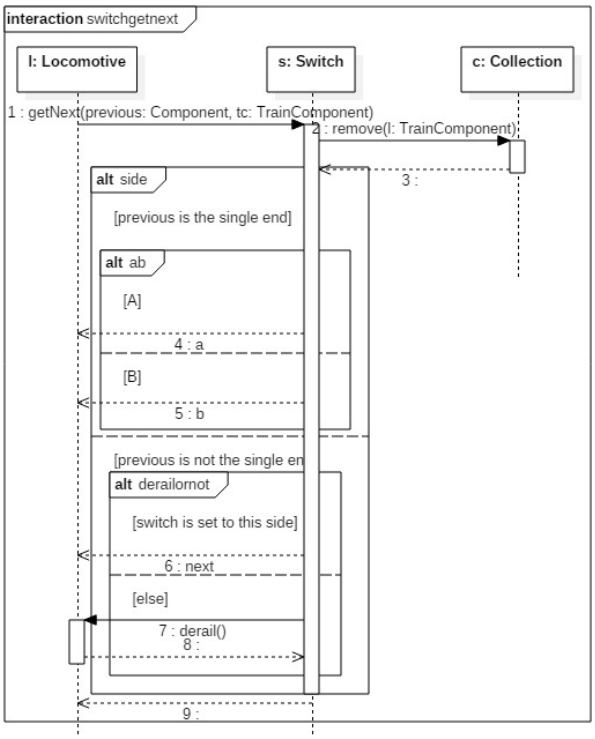
5.4. ábra. Vagonok mozgatása.



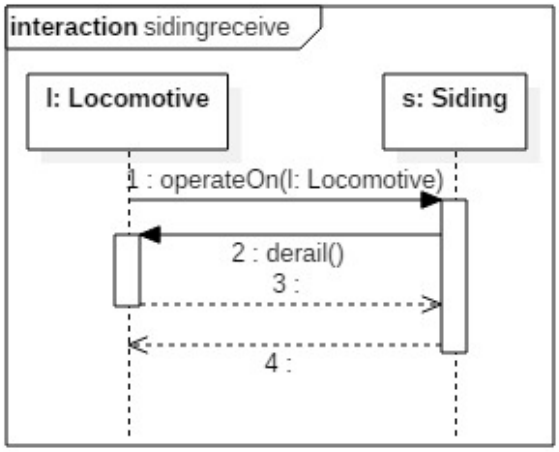
5.5. ábra. Minden pályaelemre jellemző kapcsolódó elem lekérési módja.



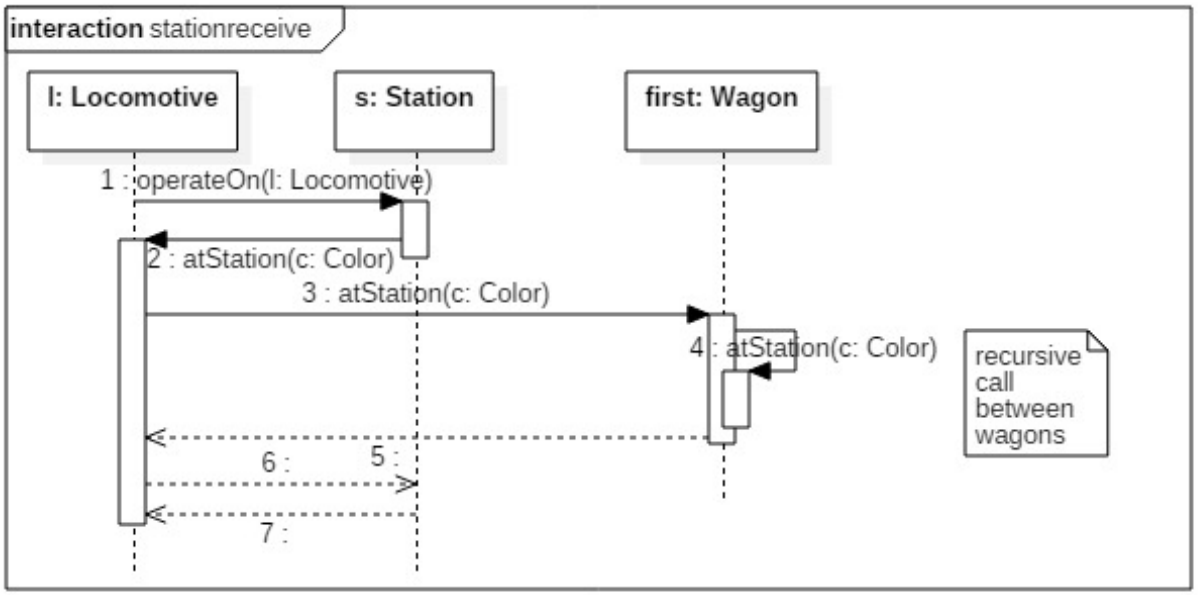
5.6. ábra. Minden pályaelemre jellemző vonatelem "befogadási" mód.



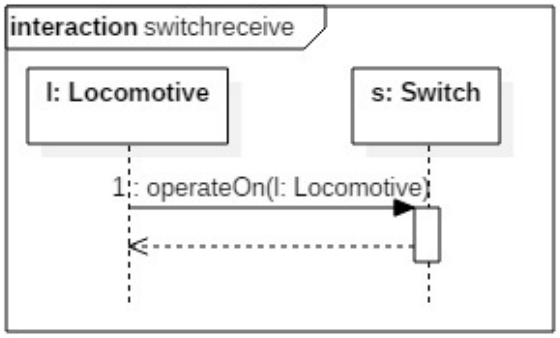
5.7. ábra. Váltóra jellemző kapcsolódó elem lekérési módja. Kisiklatja a rosszul áthaladó elemet.



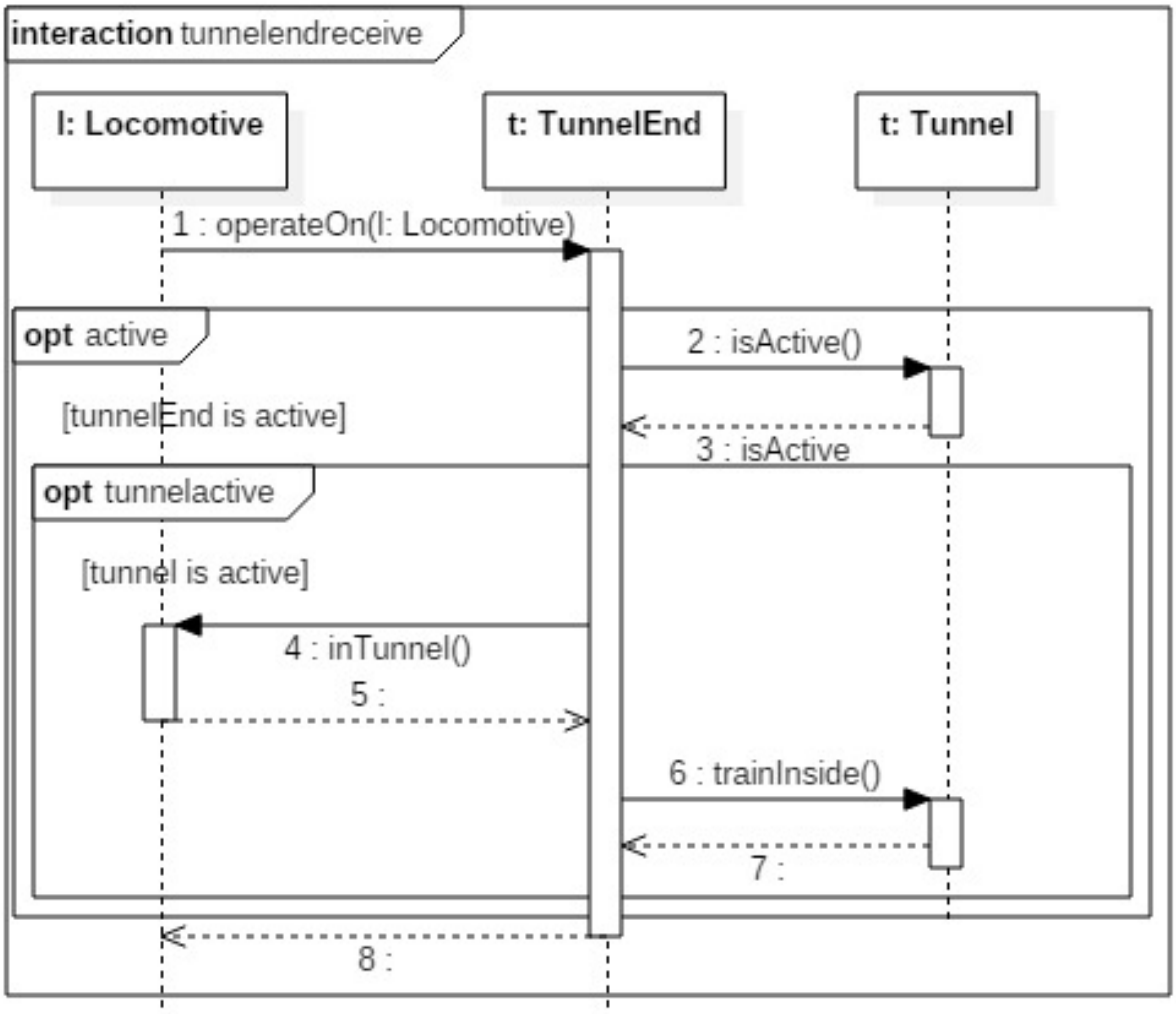
5.8. ábra. Mozdony kisiklatása amennyiben vakvágányra fut.



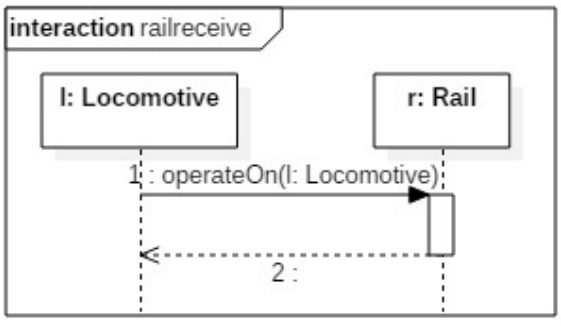
5.9. ábra. Jelzés a mozdonynak (ill. ezzel vonatnak) hogy állomásra ért.



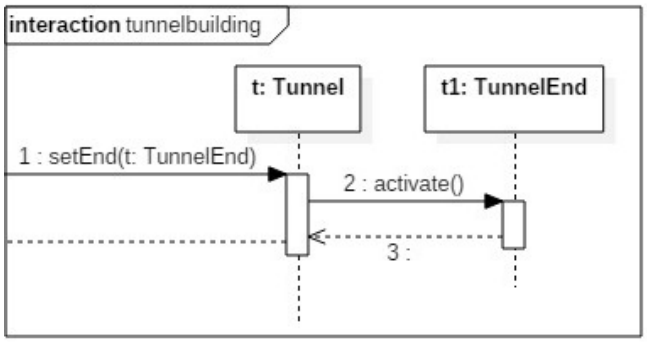
5.10. ábra. Váltó nem végez műveletet a mozdonyon.



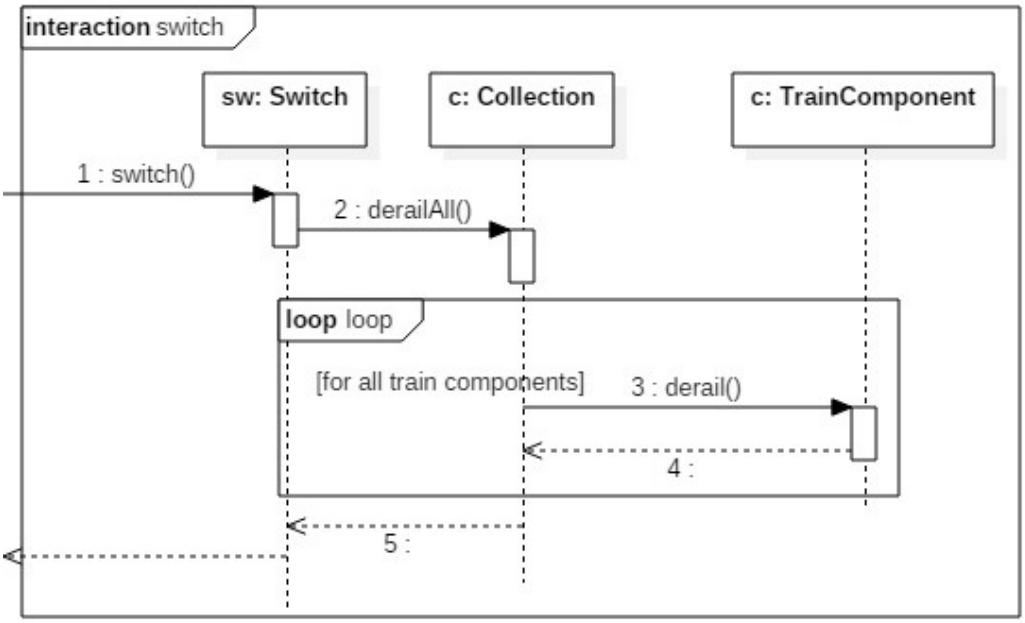
5.11. ábra. Aktív alagútszájba való be-ill. kihaladás esetén a vonatot értesítjük.



5.12. ábra. Sín nem végez műveletet a mozdonyon.

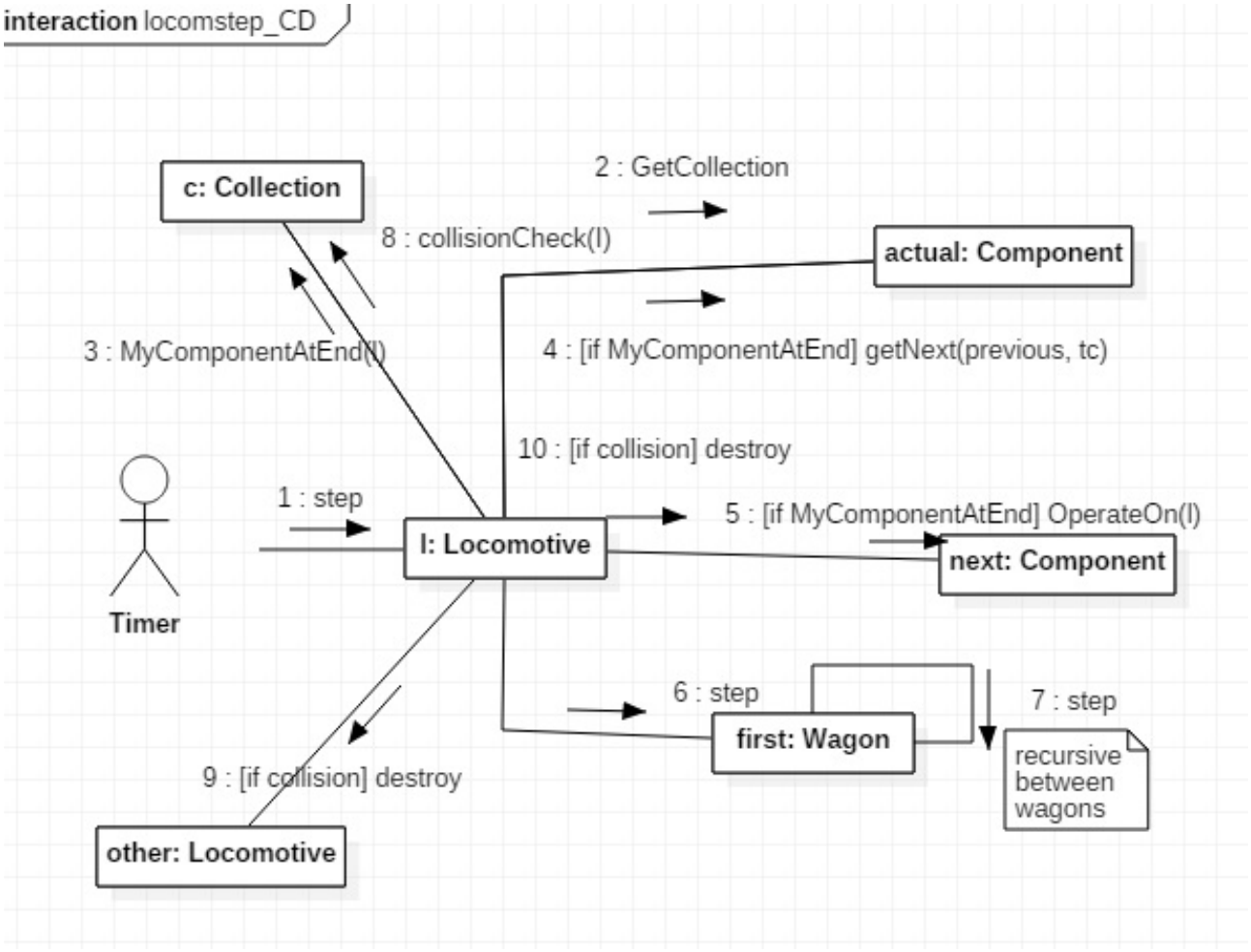


5.13. ábra. Alagútépítés.

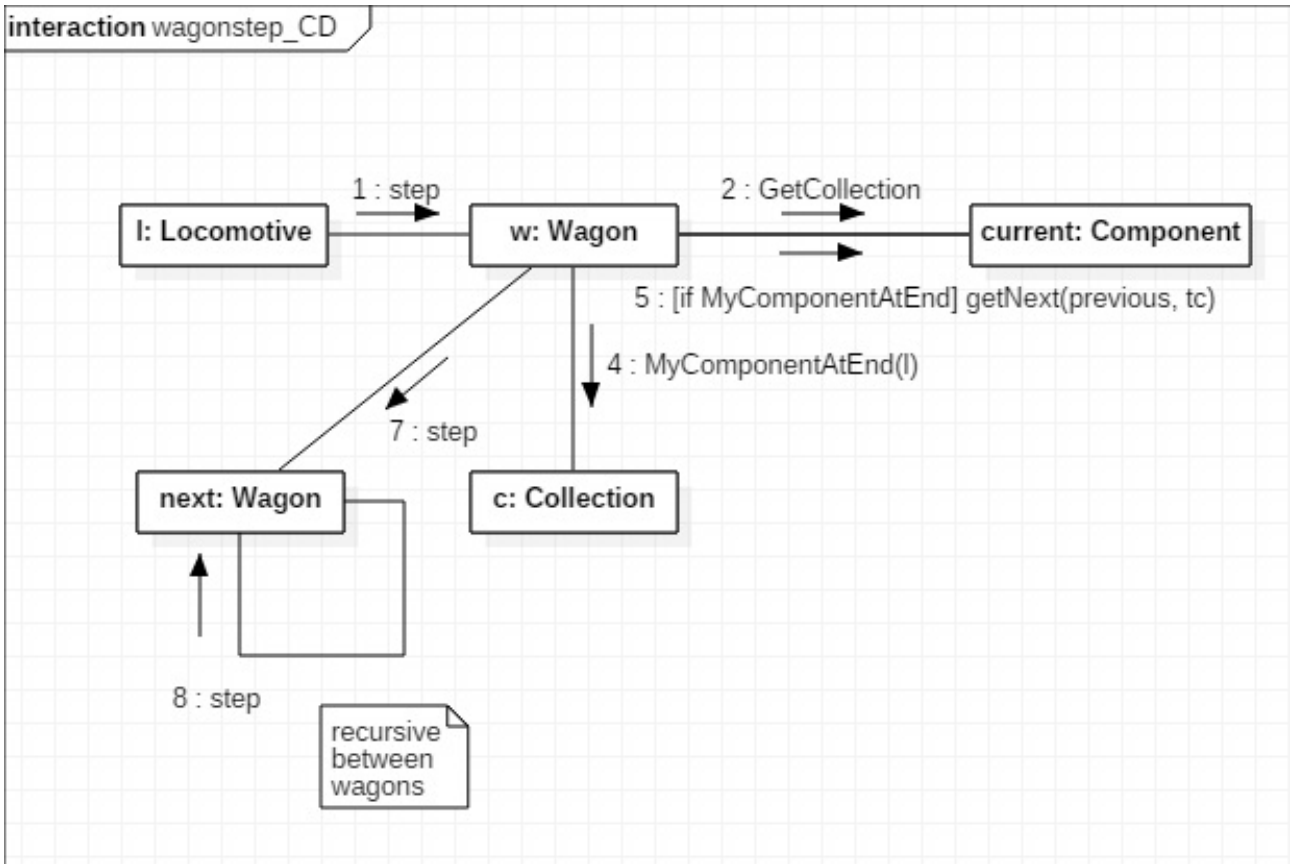


5.14. ábra. Váltó állítása, kisiklat minden rajta lévő vonatelemet.

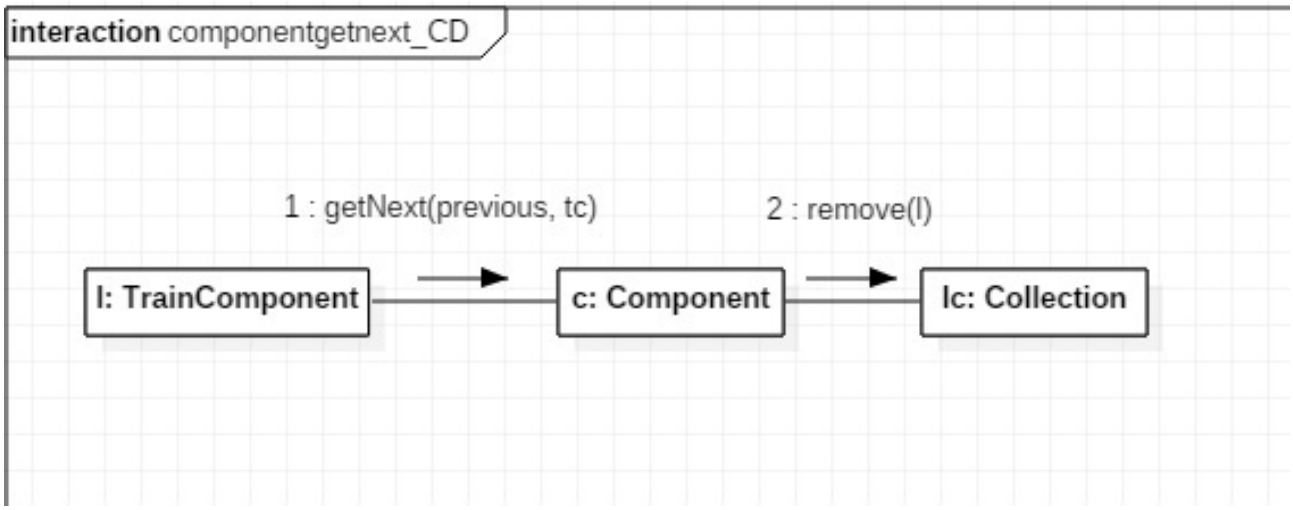
5.4. Kommunikációs diagramok



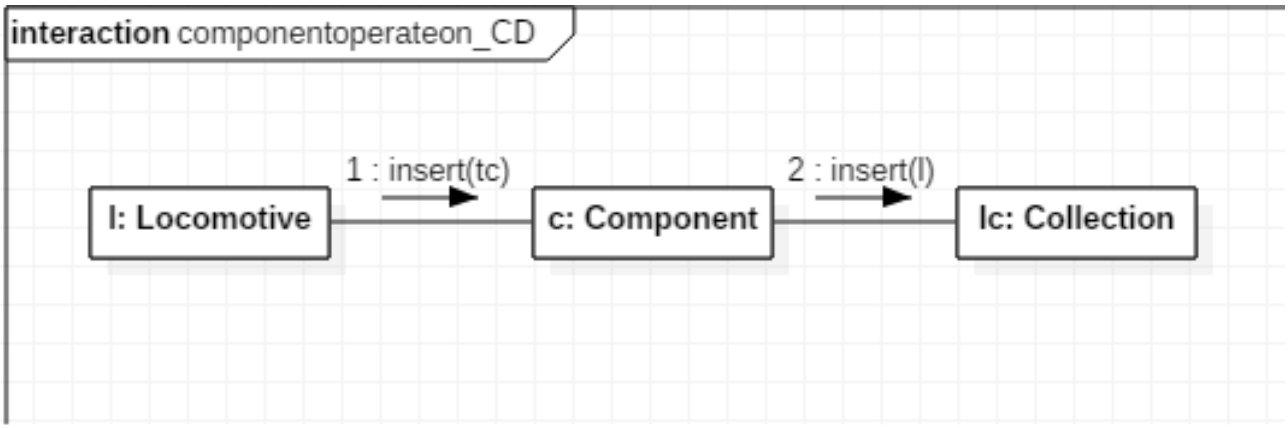
5.15. ábra. Mozdony mozgatása.



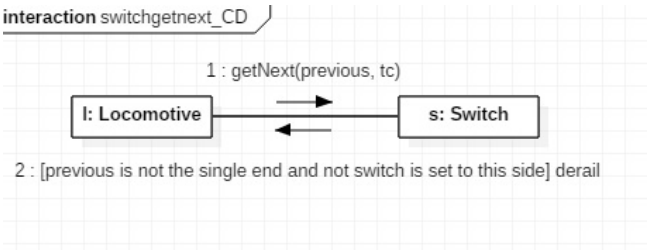
5.16. ábra. Vagonok mozgatása.



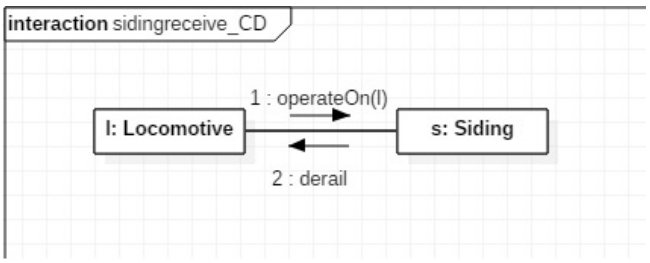
5.17. ábra. Minden pályaelemre jellemző kapcsolódó elem lekérési módja.



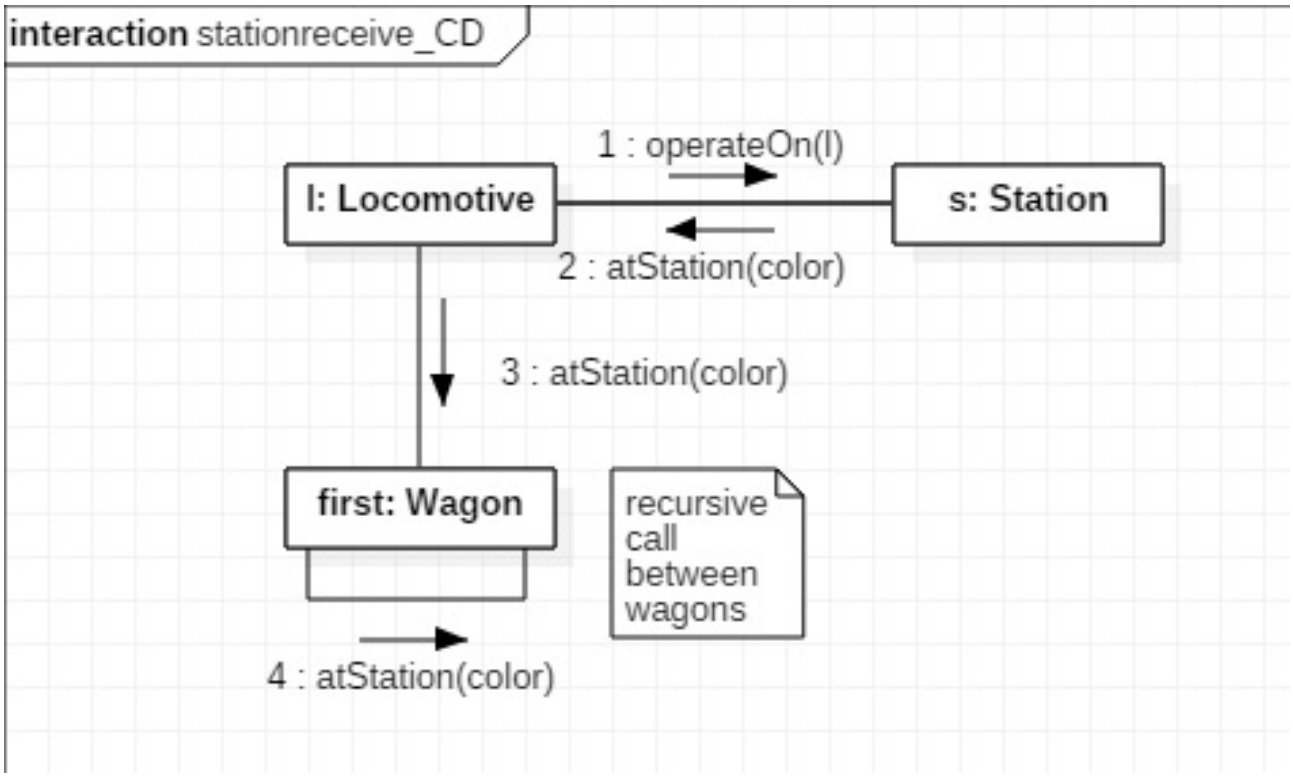
5.18. ábra. Minden pályaelemre jellemző vonatelem "befogadási" mód.



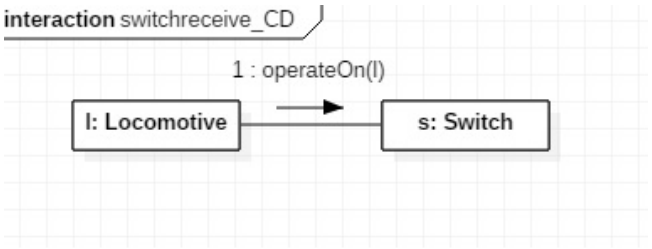
5.19. ábra. Váltóra jellemző kapcsolódó elem lekérési módja. Kisiklatja a rosszul áthaladó elemet.



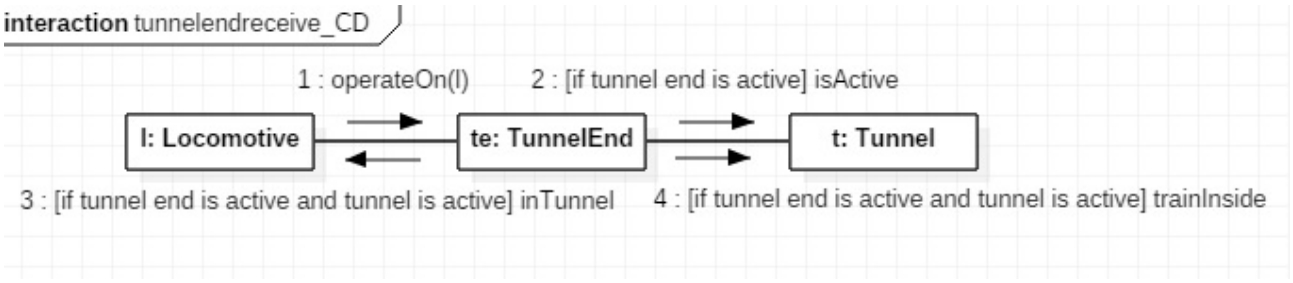
5.20. ábra. Mozdony kisiklatása amennyiben vakvágányra fut.



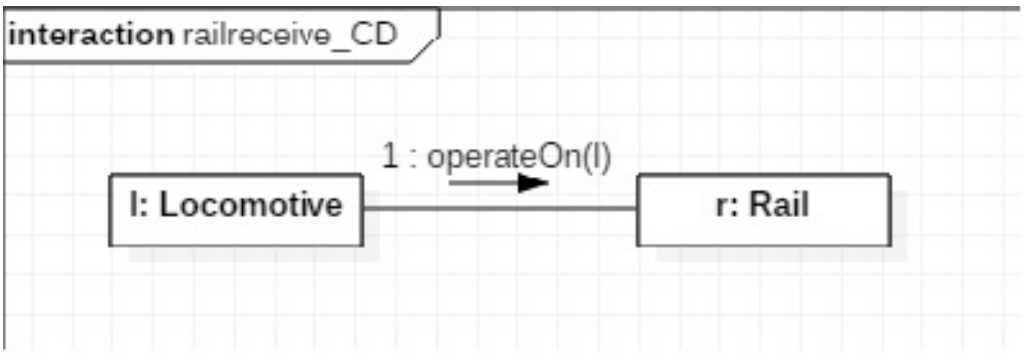
5.21. ábra. Jelzés a mozdonynak (ill. ezzel vonatnak) hogy állomásra ért.



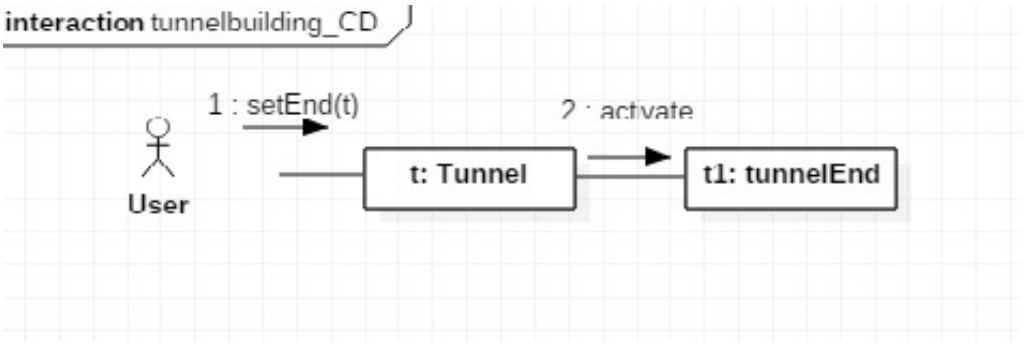
5.22. ábra. Váltó nem végez műveletet a mozdonyon.



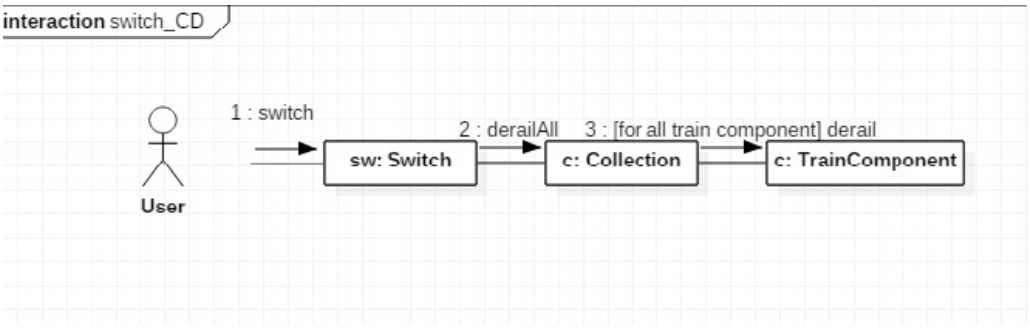
5.23. ábra. Aktív alagútszájba való be-ill. kihaladás esetén a vonatot értesítjük.



5.24. ábra. Sín nem végez műveletet a mozdonyon.



5.25. ábra. Alagútépítés.



5.26. ábra. Váltó állítása

5.5. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017.03.11. 18:00	1 óra	**Dombai** **Székely**	Rögtönzött értekezlet. Javítások megbeszélése. Use-case-ek összeírása
2017.03.11. 19:00	1 óra	**Dombai**	Tevékenység: 5.1 ábra, 5.3 szekvencia diagramok
2017.03.12. 14:00	1 óra	**Dombai** **Erős** **Székely** **Szilágyi**	Rögtönzött értekezlet. Elkészült diagramok ellenőrzése, javítások javaslása.
2017.03.12. 18:00	1 óra	**Dombai**	Tevékenység: véglegesítések szekvenciadiagramon
2017.03.12. 18:00	1 óra	**Erős**	Tevékenység: további use-case-ek keresése, 5.2 elkezdése
2017.03.12. 22:00	2 óra	**Szilágyi**	Tevékenység: 5.4. Kommunikációs diagramok elkészítése
2017.03.13. 9:00	1 óra	**Erős**	Tevékenység: 5.2 befejezése
2017.03.13. 9:00	1 óra	**Székely**	Tevékenység: 5.1 befejezése

6. Szkeleton beadás

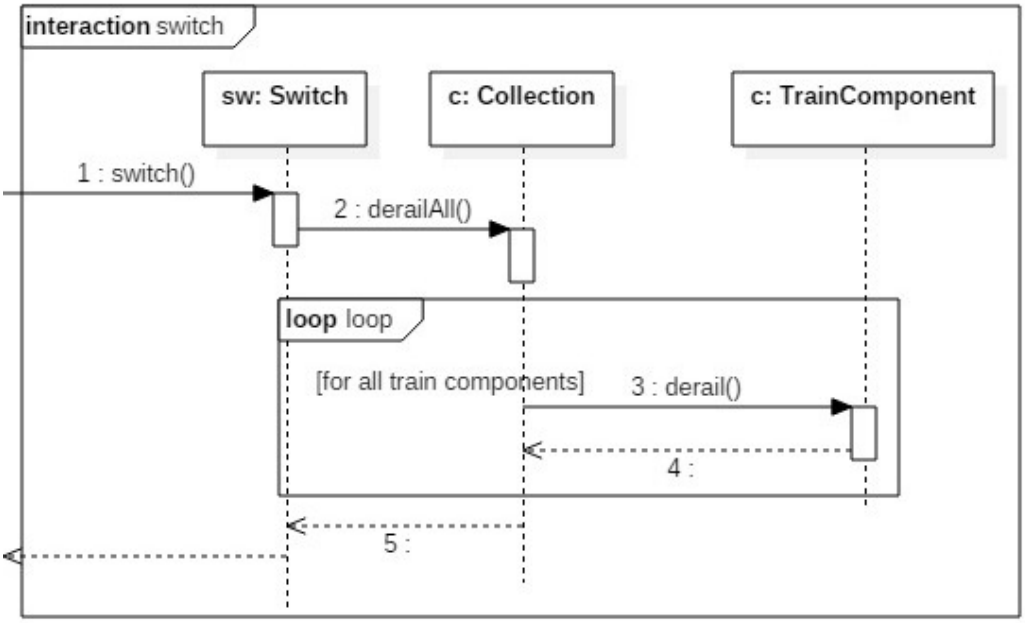
6.0. Kiegészítések, változtatások

Component absztrakt osztály lett hogy elkerülhessük a kódismétléseket, illetve bekerült pár tesztelést elősegítendő függvény.

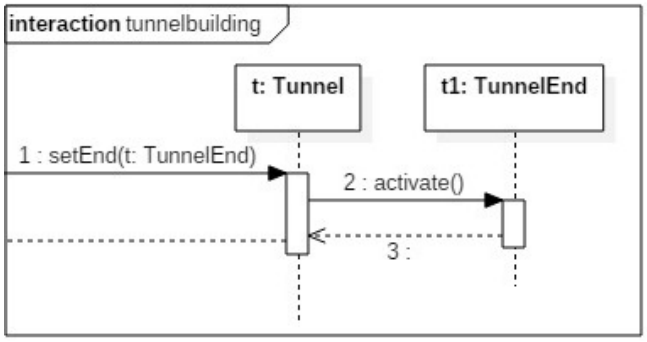
Kezelői felület tervének változása:

1. Váltó állítás
    *1.1 Van rajta vonatelem? I/N
2. Alagút építése
    *2.1 Töröljük is nyomban utána? I/N
3. Vonat mozgatása
    *3.1 Jelenlegi elem? Sín, váltó, állomás, alagútszáj. S/V/A/L
    *3.2 Történik ütközés a lépés közben? I/N
    *3.3 Jelenlegi elem végére kerül? I/N
        *3.3.1 Következő elem? (Sín, váltó, állomás, alagútszáj, vakvágány) S/V/A/L/K
            *3.2.1.1 Leszállás a vonatról (állomás esetén)
            *3.2.1.2 Aktiválatlan lagútszájba lépés (alagútszáj esetén)
            *3.2.1.3 Kisiklás (vakvágány esetén)
        *3.3.2 Következő elemre lépés.

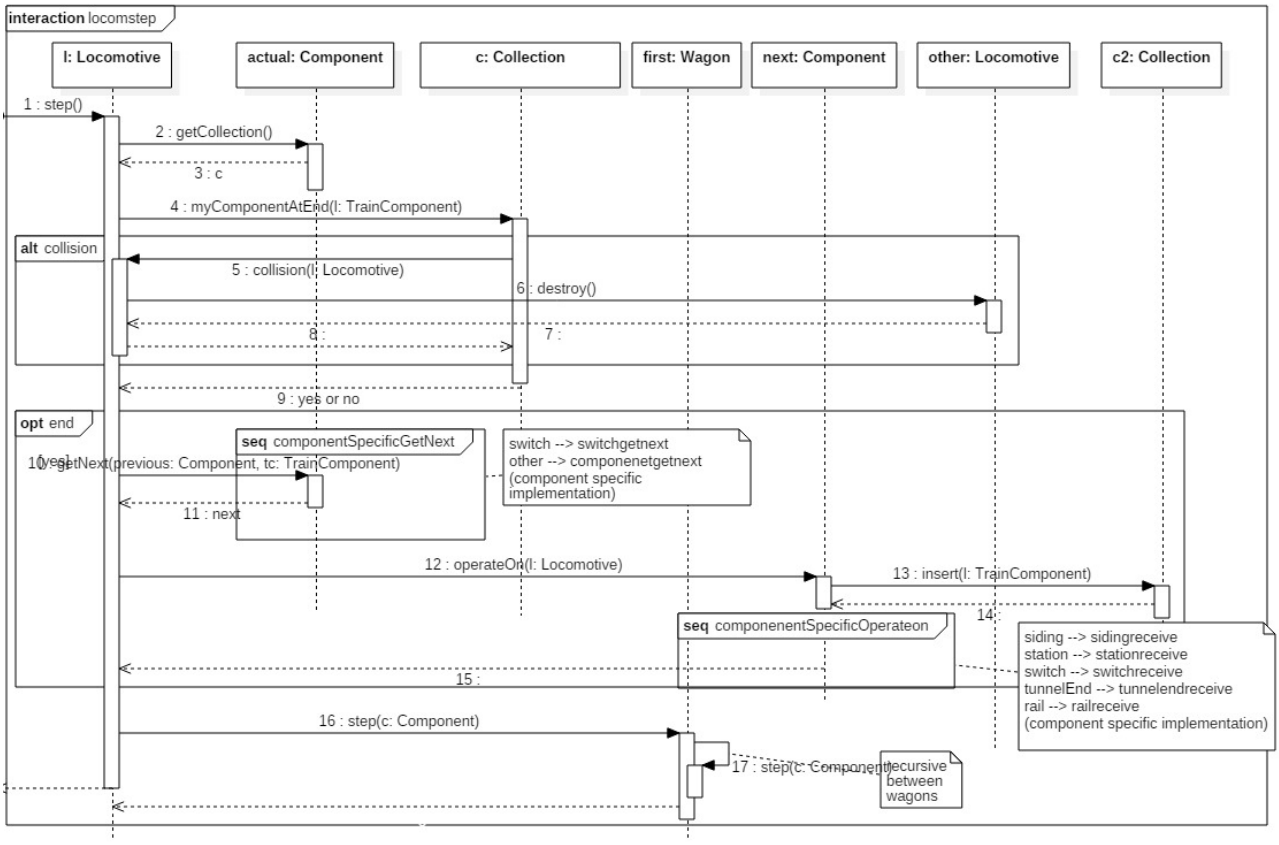
A*-al jelöltek automatikusan hívódnak.



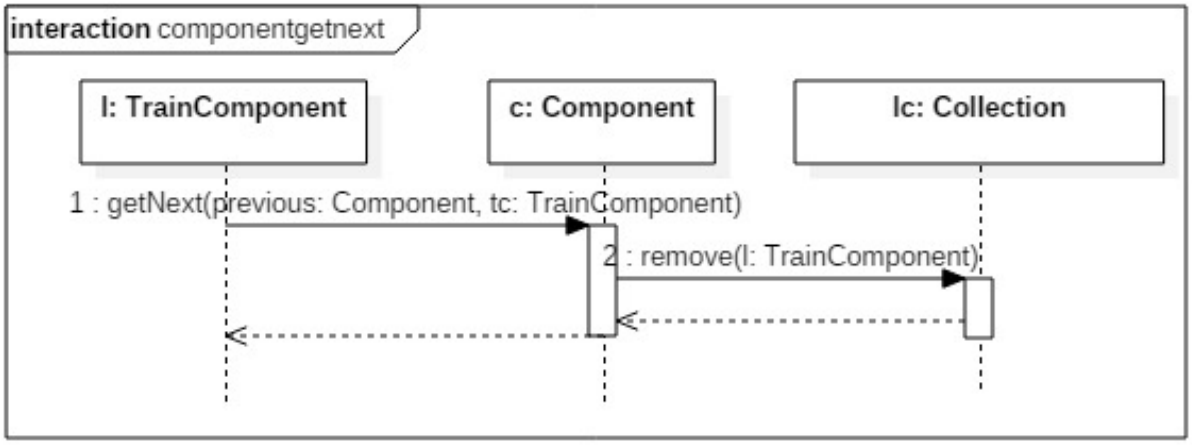
6.1. ábra. 1. teszteset - Váltó állítása, kisiklat minden rajta lévő vonatelemet.



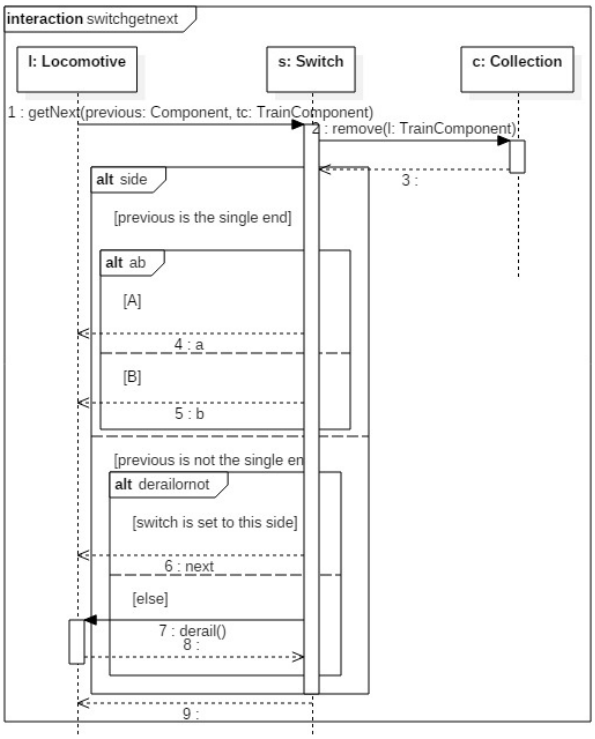
6.2. ábra. 2. teszteset - Alagútépítés.



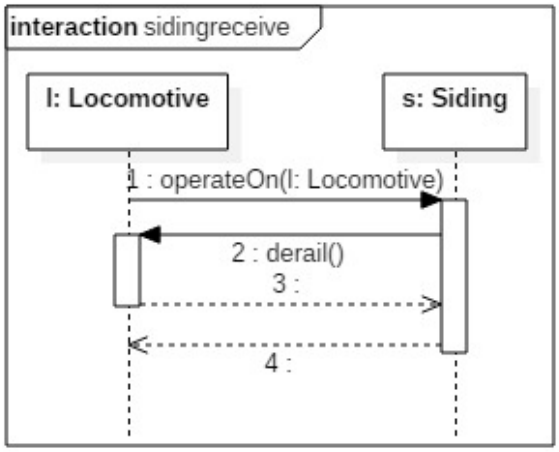
6.3. ábra. 3. teszeset - Mozdony mozgatása.



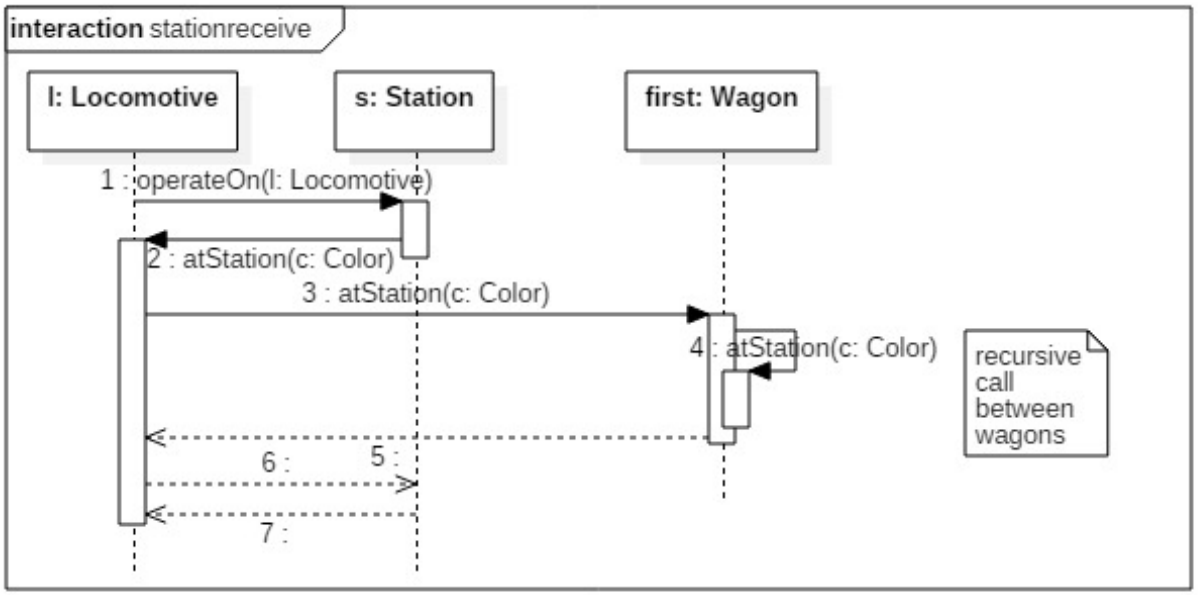
6.4. ábra. 3. teszteset hivatkozik rá - componentgetnext



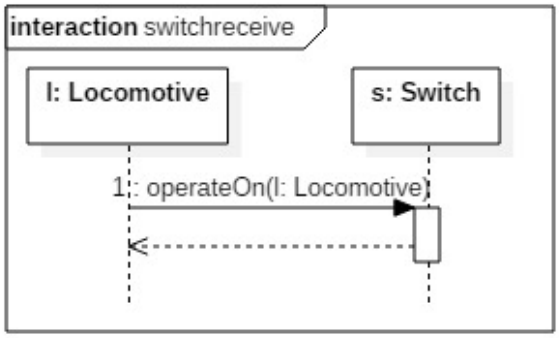
6.5. ábra. 3. teszteset hivatkozik rá - switchgetnext - Váltó rossz áthaladás esetén kisiklatja a mozdonyt.



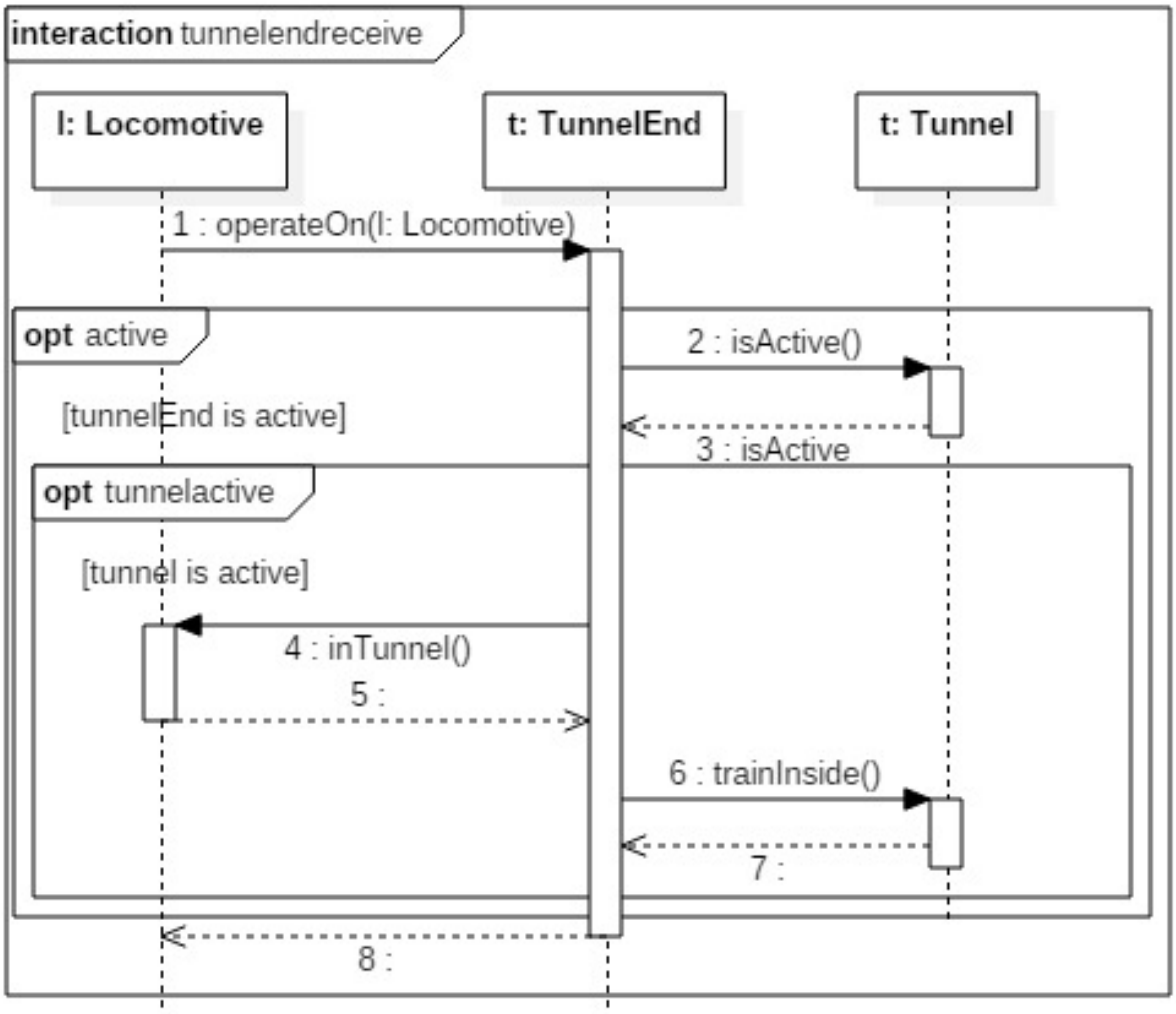
6.6. ábra. 3. teszteset hivatkozik rá - Mozdony kisiklatása amennyiben vakvágányra fut.



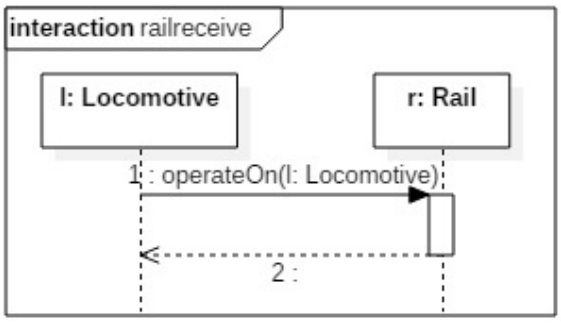
6.7. ábra. 3. teszteset hivatkozik rá - Jelzés a mozdonynak (ill. ezzel vonatnak) hogy állomásra ért.



6.8. ábra. 3. teszteset hivatkozik rá - Váltó nem végez műveletet a mozdonyon.



6.9. ábra. 3. teszteset hivatkozik rá - Aktív alagútszájba való be-ill. kihaladás esetén a vonatot értesítjük.



6.10. ábra. 3. teszteset hivatkozik rá - Sín nem végez műveletet a mozdonyon.

6.1. Fordítási és futtatási útmutató

6.1.1. Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Application.java	xx	2017.03.20 - xx	A tesztkörnyezet megvalósítása, a bemeneteknek megfelelően futtatja le a tesztet.
Collection.java	xx	2017.03.20 - xx	Egy tároló, a mozdonyoknak és a vagonoknak.
Component.java	xx	2017.03.20 - xx	Absztrakt osztály a vasút elemeihez.
Locomotive.java	xx	2017.03.20 - xx	A mozdony osztálya.
Rail.java	xx	2017.03.20 - xx	Component-ből származó vasúti elem, sín.
Siding.java	xx	2017.03.20 - xx	Component-ből származó vasúti elem, vak vágány.
Station.java	xx	2017.03.20 - xx	k
Component-ből származó vasúti elem, saját színnel rendelkező állomás. Switch.java	xx	2017.03.20 - xx	Component-ből származó vasúti elem, váltó.
TrainComponent.java	xx	2017.03.20 - xx	A vonat részeihez interface.
Tunnel.java	xx	2017.03.20 - xx	Az alagút osztálya.
TunnelEnd.java	xx	2017.03.20 - xx	Az alagút vége.
Wagon.java	xx	2017.03.20 - xx	TrainComponet-ből származó elem, vagon.

Végleges fájlok nem készültek el a nyomtatásig, ezért a méretüket és időpontjaikat nem tartalmazza ez a dokumentum.

6.1.2. Fordítás

```
javac -encoding ISO-8859-2 src/*.java
```

6.1.3. Futtatás

```
cd src
java -Dfile.encoding=ISO-8859-2 Application
```

6.2. Értékelés

Tag	Munka százalékban	Aláírás
Dombai	32.0 %	
Erős	16.67 %	
Márton	15.4 %	
Székely	18.0 %	
Szilágyi	18.0 %	

6.3. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017.03.19. 16:00	2 óra	**Dombai** **Erős** **Székely** **Szilágyi**	Értekezlet. Döntés: Erős elkészíti a felhasználói bemenetet, Dombai a teszteket.
2017.03.19. 18:00	1 óra	**Dombai**	test1, test2, test3; egyéb osztályok kiegészítése
2017.03.19. 18:00	2 óra	**Erős**	Main fgv.
2017.03.19 20:00	1 óra	**Márton**	Osztályok kiegészítése, kommentek
2017.03.20. 8:00	1 óra	**Dombai**	Kiegészítések Locomotive Step-jéhez, naplózás
2017.03.20 10:00	1 óra	**Márton**	Kommentelés
2017.03.20 11:00	1 óra	**Szilágyi**	Kommentelés

7. Prototípus koncepciója

7.0. Kiegészítések, változtatások

7.0.1. Feladat változásai

Keresztező sín

"Sheldon új pályaelemet, kereszteződő síneket vásárolt. A kereszteződés egyszintű, a különböző irányokból jövő vonatok a kereszteződésben ütköznek."

A keresztező sínek megvalósításához létrehozunk egy *(Intersection)* osztályt, mely a *Component* absztrakt osztályból fog származni a többi pályaelemhez hasonlóan. Ez az eddigi elemektől eltérően 4 irány beállító setterel fog rendelkezni. Az eddigi korlátozások miatt csak két mozdony ütközhetett, azonban a kereszteződő sínekkel ez megváltozik: mostmár a mozdonyok kocsikkal is ütközhetnek.

Ez lényegi módosítást nem igényel, mivel az egyes pályaelemeken lévő vonatelemek már tárolva vannak, így az ütközés detektálása változtatás nélkül megoldható.

Felszállás a vonatra

"Egyes állomásokon utasok a megfelelő színű üres kocsikba (a kocsi szerelvényben elfoglalt helyzetétől függetlenül) fel tudnak szállni."

További megkötések tőlünk: Az állomáson található összes utas felszáll a vonatra (ha a kocsik színe ezt lehetővé teszi); az egyes kocsik kapacitása végtelennek tekinthető, ezzel biztosítva azt hogy minden utas felszállhasson a vonatra.

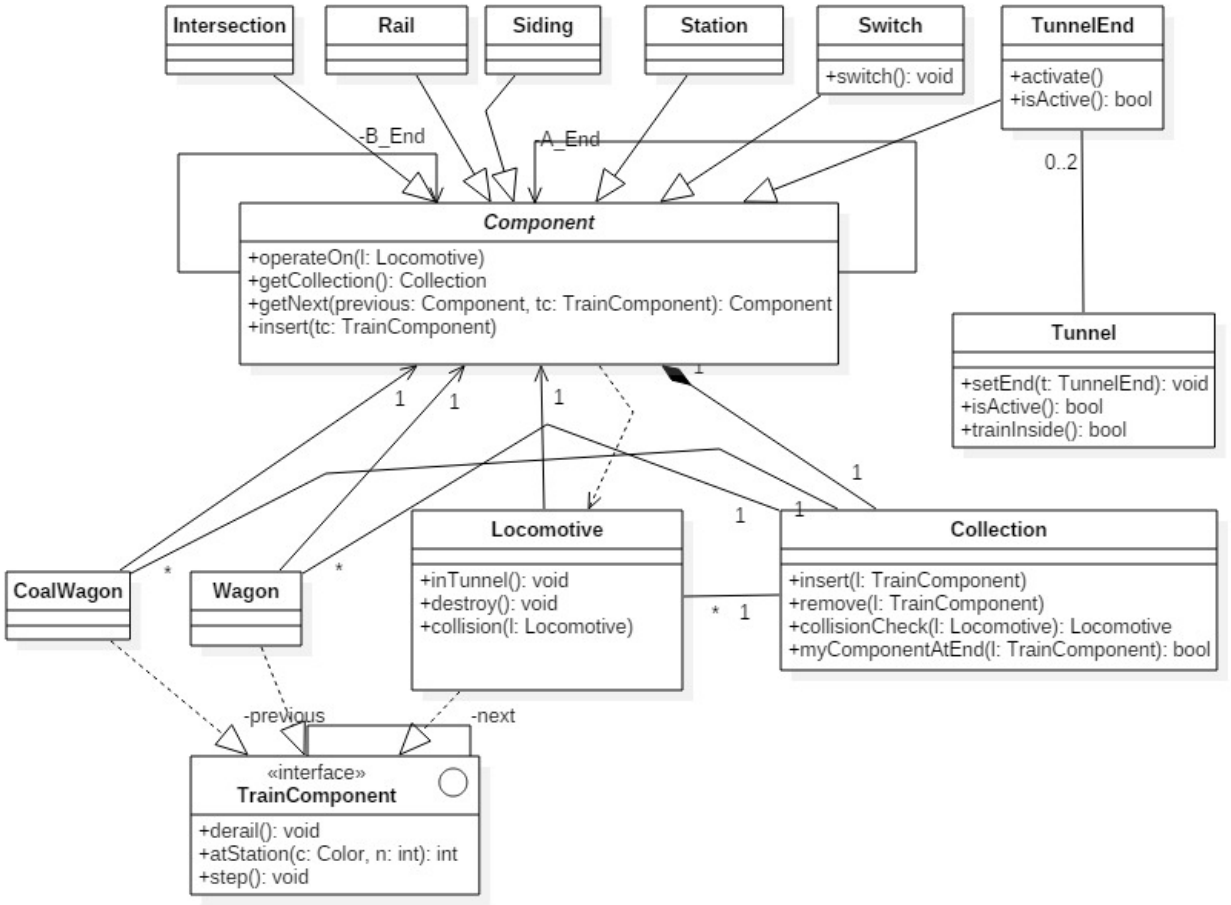
Az egyedüli változtatás a *TrainComponent atStation* metódusában lesz: az állomás színe mellett paraméterként megkapja hogy hány utas van az állomáson (ez az érték felszállás esetén csökkenhet a vagonok közötti hívások során); visszatérési értéke pedig a leszállt utasok száma. A felszállás az utasok leszállása után fog megtörténni, ezzel biztosítva azt, hogy 1-1 utas ne tudjon fel- és leszállni a vonatról azonos lépésben.

Szeneskocsi

"Sheldon bővítette a vagonkészletét. Vett szeneskocsikat, amiken nem utaznak utasok, nem is tudnak felszállni. Az utasok leszállásánál az ilyen vagonokat nem vesszük figyelembe."

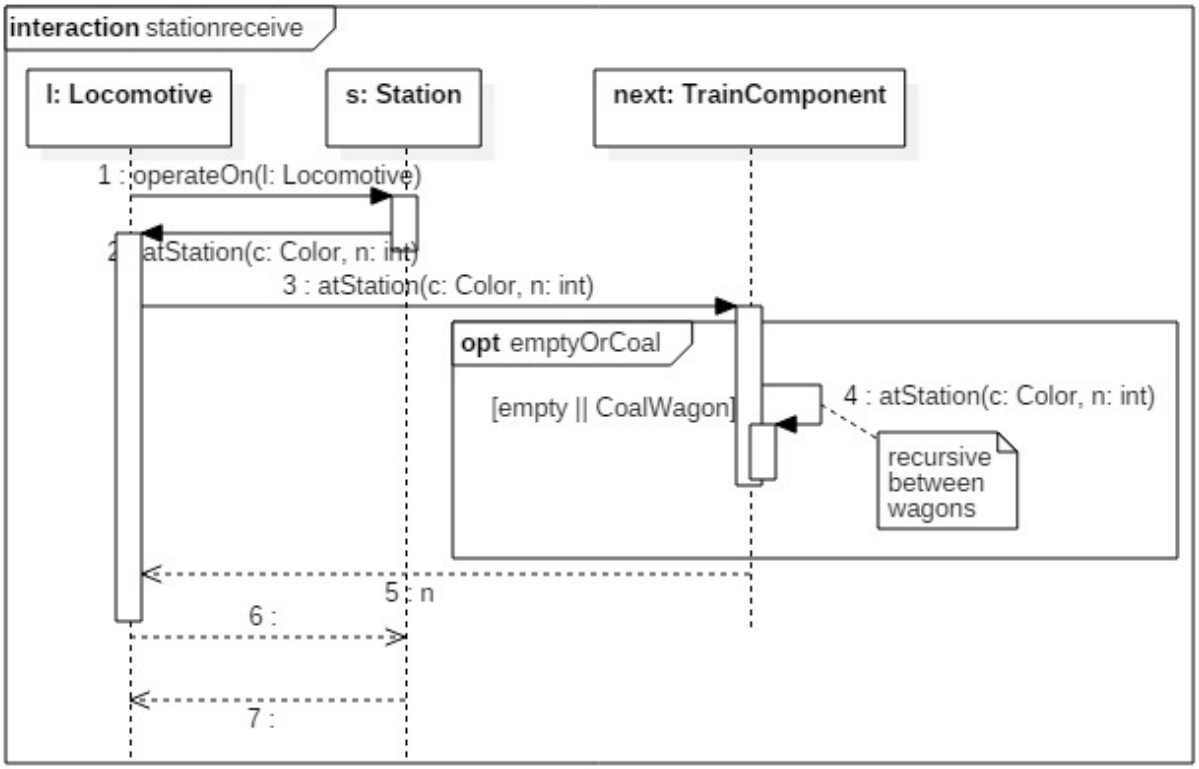
A szeneskocsi megvalósítához egy új *TrainComponent* interfésszel rendelkező osztályt hozunk létre *CoalWagon* néven. Mivel ez gyakorlatilag egy utasok nélküli kocsit jelent, ezért amikor meghívódik az *atStation* metódusa az mindig meghívja a következő kocsi *atStation* metódusát (amennyiben létezik következő kocsi)). A szeneskocsi többi metódusa megegyezik a *Wagon*-éval.

7.0.2. Módosult osztálydiagram



7.1. ábra. Osztálydiagram

7.0.3. Módosult és új szekvenciadiagamok



7.2. ábra. Állomásra érkezés szekvenciája

Megjegyzés: a felszállás ill. leszállás privát metódusokkal valósul meg.

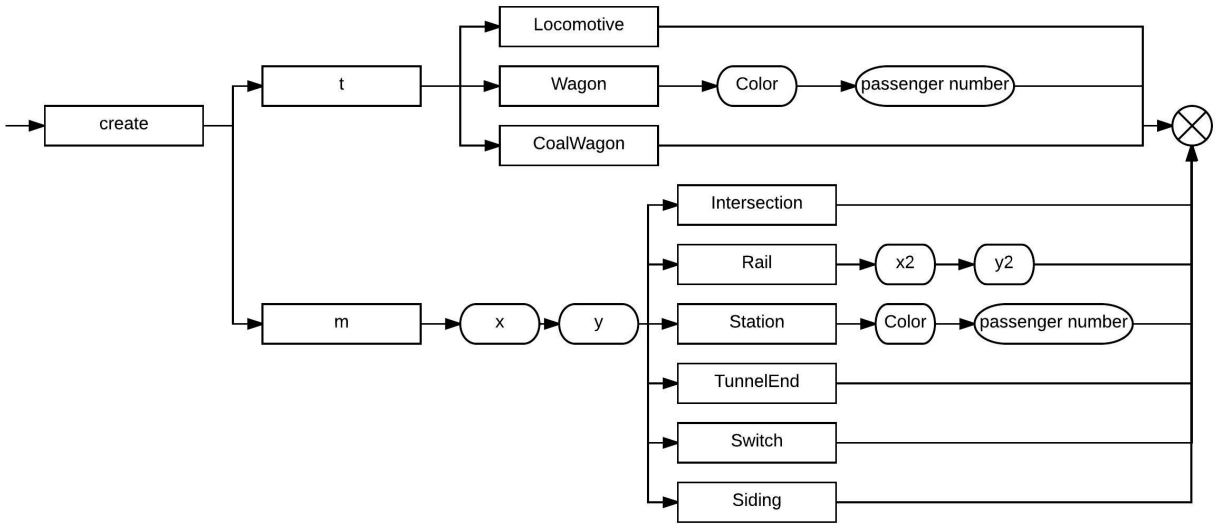
7.1. Prototípus interface-definíciója

7.1.1. Az interfész általános leírása

Az interface a szabványos bemenetéről fogad parancsokat, utasításokat (ezek is előre meg vannak határozva), valamint a parancsok függvényében generált kimenetet a szabványos kimeneten jeleníti meg. Előre készített tesztesetek alapján lehetőség van a tesztelésre is. A tesztelő tetszőleges szituációk szimulációjára képes, ennek függvényében kap egy kimenetet. Az output külön átirányítható fájlba is. Ez által le lehet ellenőrizni az adott kimenetet, elmenteni azt. A tesztek kimenetele előre meghatározott, így azoknak egyeznie kell a várt eredménnyel.

7.1.2. Bemeneti nyelv

create



Leírás
Pályát ill. vonatokat alkotó elemek létrehozása. A létrehozott pályelemekhez létrehoz egy-egy típusként egyedi ID-t (vonatot alkotó id-k, pályát alkotó id-k), ami az összekapcsolásokhoz és azonosításhoz szükséges. Az új elem ID-je megegyezik a create hívási számával az adott típuson (m/t).

Opciók

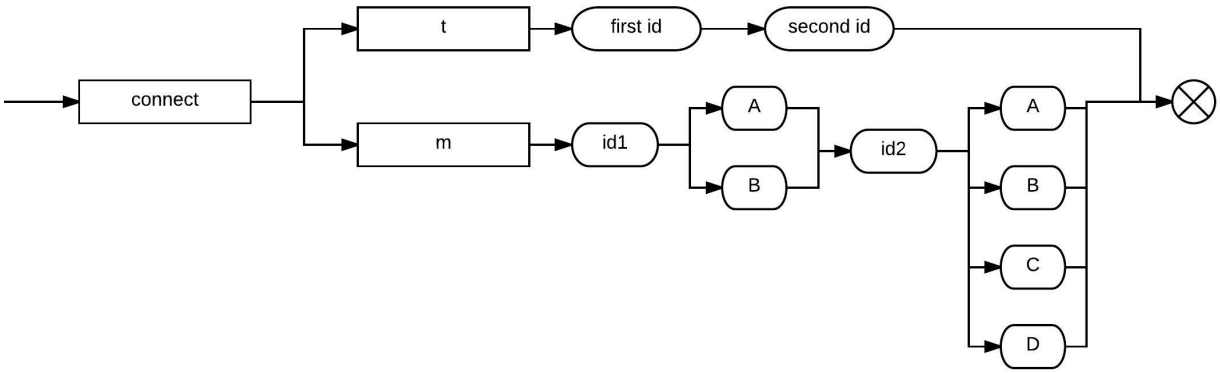
- m - pályát alkotó elem
  - x - x koordináta
  - y - y koordináta
    - Intersection - kereszteződés (pontszerű)
    - Rail - sín (hosszal rendelkezik)
      - x2 - a sín másik végének x koordinátája
      - y2 - a sín másik végének y koordinátája
    - Station (pontszerű)
      - Color - az állomás színe
      - passenger number - az állomáson lévő utasok száma
    - TunnelEnd - alagútvég (pontszerű)
    - Switch - váltó (pontszerű)
    - Siding - vakvágány (pontszerű)
- t - vonatot alkotó elem
  - Locomotive - mozdony
  - Wagon - vagon
    - Color - a kocsi színe
    - passenger number - a kocsiban lévő utasok száma induláskor
  - CoalWagon - szeneskocsi

Példa parancsok

- create m 4.43 5.12 Station red 10
- create t CoalWagon
- create m 2.12 3.34 Intersection
- create m 67.1 45.3 Rail 87.3 98.3
- create t Wagon blue 10

---

connect



Leírás
Pályát ill. vonatokat alkotó elemek egymáshoz kapcsolása (pályaelem-pályaelem, vonatelem-vonatelem). A create paranccsal létrehozott elemeket kapcsolja egymáshoz, az ott generált ID alapján.

Opciók

- t - vonatot alkotó elem
  - first id - A vonat elejéhez közelebb található elem ID-je (neki next-je a second id)
  - second id - A végéhez közelebb található elem ID-je (neki previous-a a first id).
- m - pályát alkotó elem
  - id1 - Az egyik pályaelem ID-je
    - A - Ha az "A" oldalához akarjuk kapcsolni az id2-t
    - B - Ha a "B" oldalához akarjuk kapcsolni az id2-t
  - id2 - A másik pályaelem ID-je
    - A - Ha az "A" oldalához akarjuk kapcsolni az id1-t
    - B - Ha a "B" oldalához akarjuk kapcsolni az id1-t.
    - C - Ha id2 intersection, akkor "C" oldalára kapcsoljuk az id1-et, amúgy "A"-ra.
    - D - Ha id2 intersection, akkor "D" oldalára kapcsoljuk az id1-et, amúgy "B"-ra.

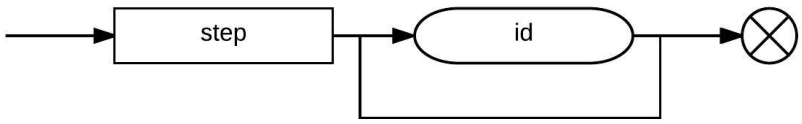
Megjegyzés: Intersection esetén csak A<—>B illetve C<—->D irányban lehet áthaladni.

Példa parancsok

- connect t 2 3
- connect m 6 A 7 B
- connect m 9 A 12 D

---

step



Leírás
A teljes terepasztal, vagy egy-egy mozdony (ill. ezzel vonat) léptetésére szolgál.

Opciók

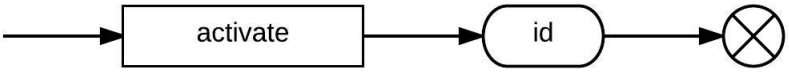
- id - Léptetendő mozdony ID-je. (opcionális)

Példa parancsok

- step 2
- step

---

activate



Leírás
Egy-egy alagútvég aktiválására ill. deaktiválására szolgál.

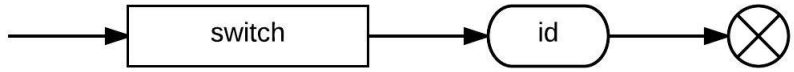
Opciók

- id - Aktiválandó alagútszáj ID-je.

Példa parancsok

- activate 3

switch



Leírás
Egy-egy váltó átállítására szolgál.

Opciók

- id - Átváltandó váltó ID-je.

Példa parancsok

- switch 5

---

place

Leírás
Megadott ID-vel rendelkező mozdonyt, illetve a hozzá kapcsolódó kocsikat elhelyezi a megadott ID-vel rendelkező sín megadott oldalára. A sín hossza legalább akkora mint a teljes vonat.

Opciók

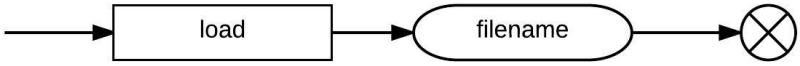
- locomotive ID - A mozdony ID-je
- rail ID - a sín ID-je
- side - sín A/B oldala

Példa parancsok

- place 2 4 A

---

load



Leírás
Egy pálya betöltésére szolgál.

Opciók

- filename - A fájl elérési útvonala

Példa parancsok

- load map1.txt

---

retry

Leírás
Jelenlegi pálya újrakezdésére szolgál

Opciók

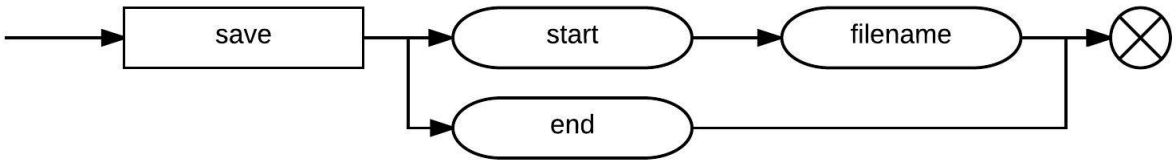
- nincs paramétere

Példa parancsok

- retry

---

save



Leírás
Konzolra írt parancsok mentésére szolgál. *save start fájlnév* kiadásától *save end* kiadásáig minden konzolra beírt parancsot elment a fájlba.

Opciók

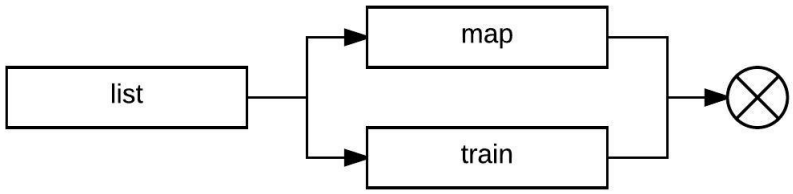
- start - Mentés kezdete
  - filename - Fájl elérési útvonala
- end - Mentés vége

Példa parancsok

- save start palya1.txt
- save end

---

list



Leírás
Létrehozott pályát- és vonatokat alkotó elemek valamit az alagút státuszának lekérdezésére szolgál.

Opciók

- map - Pályát alkotó elemek, valamint alagút lekérdezése
- train - Vonatokat alkotó elemek lekérdezése

Példa parancsok

- list train
- list map

7.1.3. Pályát leíró fájlok

A pályákat a fent ismertetett parancsokkal írjuk le, nem hozunk létre külön xml fájlokat.

7.1.4. Kimeneti nyelv

create

created <típus> with id: <id>

connect

Pályelemek esetén:
connected (<id1> to <side> side of <id2>) and (<id2> to <side> of <id1>)
Vonatelemek esetén:
connected <id1> to <id2>
megj.: vonat eleje<—[id1]-[id2]<—-vonat vége

step

Példa egy mozdony léptetésére:
stepped <mozdony id> new position: <pályaelem id> <pos>
which moved: <kocsi id> new position: <pályaelem id> <pos>
which moved: <kocsi id> new position: <pályaelem id> <pos>

activate

Sikeres alagútszáj aktiválás esetén (true ha aktiváltuk, false ha deaktiváltuk):
activated <alagútszáj id> <true/false>
Sikertelen alagútszáj aktiválás esetén (ha nem egy sín két oldalán találhatót aktiváltunk):
couldn't activate <id>

switch

switched <váltó id> new position: <A/B>

place

Nincs kimenete.

load

Nincs kimenete.

retry

Nincs kimenete.

save

Nincs kimenete.

list

Típusoknak megfelelő sorokat ír egymás után, az alábbi formátumokban:
Kereszteződés esetén:
Intersection <id> at <x>, <y>
A side connected to <id>
B side connected to <id>
C side connected to <id>
D side connected to <id>

Sín esetén:
Rail <id> from <x>, <y> to <x>, <y>
A side connected to <id>
B side connected to <id>

Állomás esetén:
Station <id> at <x>, <y> with <n> passengers and <color> color
A side connected to <id>

B side connected to <id>

Alagútvég esetén:
TunnelEnd <id> at <x>, <y>: <activated/deactivated>
A side connected to <id>
B side connected to <id>

Váltó esetén:
Switch <id> at <x>, <y> switched to: <A/B>
A side connected to <id>
B side connected to <id>
C side connected to <id>

Vakvágány esetén:
Intersection <id> at <x>, <y>
A side connected to <id>
B side connected to <id>

Alagút kiírása:
Tunnel is <activated/deactivated>
End1: <id>; End2<id>
Train inside <true/false>

Mozdony esetén:
Locomotive <id> on <map component id> at <pos>; next: <next id>;
collided <true/false>; destroyed <true/false>; derailed <true/false>

Vagon esetén:
Wagon <id> on <map component id> at <pos>; next: <next id>, previous: <previous id>;
collided <true/false>; destroyed <true/false>; derailed <true/false>;
passenger number: <n>, color: <color>

Szeneskocsi esetén:
CoalWagon <id> on <map component id> at <pos>; next: <next id>; previous: <previous id>;
collided <true/false>; destroyed <true/false>; derailed <true/false>

7.2. Összes részletes use-case

Use-case neve	**create**
Rövid leírás	A vonatokat és a pályát felépítő egységek létrehozása.
Aktorok	Tesztelő
Forgatókönyv	Létrehoz új pálya-elemeket. Minden létrehozott pálya-elemhez rendel egy egyedi azonosítót amelyek segítségével történik az összekapcsolásuk és azonosításuk.

Use-case neve	**connect**
Rövid leírás	A vonatokat és a pályát felépítő egységek összekapcsolása.
Aktorok	Tesztelő
Forgatókönyv	Pálya-elemhez másik pálya-elemet, vonat-elemhez pedig másik vonat-elemet kapcsol hozzá a create-ben létrehozott ID-k alapján.

Use-case neve	**step**
Rövid leírás	A terapasztalon lévő összes, vagy egy-egy vonat léptetése.
Aktorok	Tesztelő
Forgatókönyv	Amennyiben paraméterként kap egy ID-t akkor csak azt a mozdonyt (és általa a teljes vonatot) lépteti, alap esetben viszont az összes terepasztalon lévő vonatot.

Use-case neve	**activate**
Rövid leírás	Alagútszáj aktiválása/deaktiválása.
Aktorok	Tesztelő
Forgatókönyv	A paraméterként kapott ID-jű alagútszáj aktiválása illetve deaktiválása.

Use-case neve	**switch**
Rövid leírás	Váltó állítása egyik állapotból a másikba.
Aktorok	Tesztelő
Forgatókönyv	A paraméterként kapott ID-jű váltó átállítása az ellenkező állásba.

Use-case neve	**place**
Rövid leírás	Mozdonyt elhelyezi egy sínpár egyik felére/menetirányába.
Aktorok	Tesztelő
Forgatókönyv	A paraméterként kapott ID-jű mozdonyt és a hozzá kapcsolódó kocsikat elhelyezi a szintén paraméterként átadott ID-jű sín megfelelő oldalára.

Use-case neve	**retry**

Rövid leírás	Pálya újrakezdése.
Aktorok	Tesztelő
Forgatókönyv	A jelenlegi pálya kezdőállapotát tölti vissza, újbóli próbálkozás szándéka esetén.

Use-case neve	**save**
Rövid leírás	Konzolra írt parancsok mentésére szolgál.
Aktorok	Tesztelő
Forgatókönyv	save start fájlnév kiadásától save end kiadásáig minden konzolra beírt parancsot elment a fájlba. A fájl elérési útvonalát a start paranccsal együtt szükséges megadni.

Use-case neve	**list**
Rövid leírás	Létrehozott pályát és vonatokat alkotó elemek listázására szolgál
Aktorok	Tesztelő
Forgatókönyv	Típusoknak megfelelő státusz lekérdezésére szolgál, amivel könnyen ellenőrizhető a futás eredménye. Pályát alkotó elemek listázása esetén az alagút státusza is megjelenik.

7.3. Tesztelési terv

Teszt-eset neve	**Vonat haladása sínváltás nélkül**
Rövid leírás	Vonat mozgatásának tesztelése
Teszt célja	Ellenőrzi, hogy a vonat megfelelően halad-e előre a síneken alapesetben, ha nem megy át váltón, nem érkezik állomásra, stb.

Teszt-eset neve	**Vonat haladása sínváltással**
Rövid leírás	Ellenőrzi, hogy a vonat megfelelően halad-e az egyes elemek között.
Teszt célja	

Teszt-eset neve	**Alagút áthelyezése**
Rövid leírás	Az alagútvég áthelyezésének tesztelése

Teszt célja	Ellenőrizhető, hogy az alagút egy végének áthelyezése sikeres-e a vonatok tudják-e azon keresztül folytatni útjukat. Továbbá az is ellenőrizhető, hogy érvényes helyre épül-e az adott alagútszáj.

Teszt-eset neve	**Állomásra érkezés**
Rövid leírás	Az állomással való interakciók tesztelése
Teszt célja	Ellenőrizhető, hogy a vonat állomásra érkezés esetén megtörténnek-e specifikált interakciók, hogy pl. szenes kocsikba nem történik fel/leszállás

Teszt-eset neve	**Váltás**
Rövid leírás	Váltó átállításának tesztelése
Teszt célja	Annak ellenőrzésére szolgál, hogy megtörténik-e a váltók átállítása

Teszt-eset neve	**Kocsik húzása**
Rövid leírás	Vonat egyes elemei egymást húzzák
Teszt célja	Annak ellenőrzése, hogy az összekapcsolt vonat képes-e együtt mozogni a terepasztalon, azaz az elől lévő kocsik húzzák-e a mögöttük lévőket.

Teszt-eset neve	**Vonatok ütközése**
Rövid leírás	Vonatok ütközésének a tesztelése
Teszt célja	Ellenőrizhető,hogy két vonat ütközése esetén azok ténylegesen megsemmisülnek-e.

Teszt-eset neve	**Vakvágányra lépés**
Rövid leírás	A vakvágányra lépés tesztelése
Teszt célja	Tesztelhető, hogy ha a vonat vakvágányra lép, akkor ténylegesen kisiklik-e, és nem tudja folytatni az útját.

Teszt-eset neve	**Utasok leszállása**
Rövid leírás	Az utasok megfelelő leszállásának ellenőrzése

Teszt célja	Tesztelhető, hogy az utasok a megfelelő színű állomásokon szállnak-e le, illetve olyan utas nem tud leszállni, aminek színe nem egyezik.

Teszt-eset neve	**Váltón kisiklás**
Rövid leírás	Váltó átállítása úgy, hogy azon vonat halad át
Teszt célja	Ellenőrizhetjük, hogy egy vonat kisiklik-e ha átállítunk egy váltót amin éppen áthalad.

7.4. Tesztelést támogató segéd- és fordítóprogramok specifikálása

A tesztelés manuálisan nehézkes lenne. A támogató segédprogram tartalmaz különböző opciókat, többet között a kimenet fájlba való mentését is. Ennek során össze lehet vetni a tényleges aktuális kimenetet a várt kimenettel, és megállapítható, hogy sikeres-e egy teszteset vagy sem. A tesztelés természetesen determinisztikus, így minden esetben van várható kimenet, ami egyben fix is, így nem szabad eltérnie a kapott kimenetnek ettől. Ha mégis, abban az esetben hibás a teszt.

7.5. Napló

Kezdet	**Időtartam**	**Résztvevők**	**Leírás**
2017.03.24. 21:00	1 óra	**Dombai Márton**	Értekezlet.
2017.03.25. 12:00	2 óra	**Dombai**	7.0 pont, 7.1.2
2017.03.26. 11:00	1 óra	**Dombai**	7.3, kiegészítések
2017.03.26. 19:00	1 óra	**Márton**	7.1.1, 7.4
2017.03.26. 20:00	1 óra	**Székely**	7.2
2017.03.27. 9:00	1 óra	**Márton**	7.3

8. Részletes tervek

8.1. Osztályok és metódusok tervei

8.1.1. Component

- Felelősség

  Definiálja a mozdony fogadására/átadásra képes eszközöket.

- Ősosztályok
  Nincs ős.

- Interfészek
  Nem valósít meg interfészt

- Attribútumok

  - Component AEnd: Megadja a komponens egyik végét.
  - Component BEnd: Megadja a komponens másik végét.
  - double length: Komponens hossza.
  - double x0, y0, x1, y1 : Komponens pozíciójának koordinátái.

- Metódusok

  + void operateOn(l: Locomotive) : Saját magára jellemző módon módosítja a mozdony (illetve ezzel a vonat) tulajdonságait
  + Collection getCollection() : Visszaadja a pályaelemen lévő vonatelemeket.
  + Component getNext(previous: Component, tc: TrainComponent) : Paraméterként megkapja, hogy melyik vonatelem kéri a következő elemet, valamint hogy az melyik elemről került a jelenlegire.
  + void insert(tc: TrainComponent) : Amikor egy kocsi rálép a következő elemre, akkor a következő elem ezt hívja meg.
  + void setEnd(side: char, end: Component): Beállítja az adott komponenes egyik végét.
  + Component(x0: int, y0: int, x1: int, y1: int): Konstruktor. Beállítja az aktuális komponens pozícióját.

8.1.2. Collection

- Felelősség

  Egy-egy sínen utazó mozdonyok/vagonok kollekciója. Megvalósítja az ütközésdetektálást.

- Ősosztályok

  Nincs ősosztály.

- Interfészek
  Nem valósít meg interface-t.

- Attribútumok
  Nincsenek publikus attribútumok.
  - TrainComponent[*] trainComponents: tárolja a vonat elemeit
  - float[*]: tárolja a vonat elemeinek a pozícióját
- Metódusok
  + void insert(l: TrainComponent): A paraméterként kapott mozdonyt/vagont hozzáadja a kollekcióhoz.
  + void remove(l: TrainComponent): A paraméterként kapott mozdonyt/vagont eltávolítja a kollekcióból.
  + bool myComponentAtEnd(l: TrainComponent): Megadja hogy a vonat elérte-e a pályaelem végét, ha igen true-val tér vissza, ha nem akkor növeli a pozícióját. A komponenesek step metódusa hívja meg.
  - void updatePositionOf(l: TrainComponent): Frissíti az adott vonat pozícióját
  - void checkForComponentAt(position: float): Megnézi, hogy a beállítandó pozíción van-e másik vonat.

8.1.3. CoalWagon

- Felelősség
  Szeneskocsi reprezentációja, utasok nem utaznak ezekben a kocsikban.
- Ősosztályok

  TrainComponent
- Interfészek
- Attribútumok
  Nincsenek olyan attribútumai ami nem az ősosztályból származik.
- Metódusok
  Metódusok leírása ősosztálynál.

8.1.4. Intersection

- Felelősség
  Kereszteződő sínek reprezentációja.
- Ősosztályok

  Component
- Interfészek
  Nem valósít meg interface-t.
- Attribútumok
  - Component CEnd: Kereszteződés 3. vége

- Component DEnd: Kereszteződés 4. vége

- Metódusok

  Nincsenek metódusai.

8.1.5. Locomotive

- Felelősség
  A Locomotive osztály egy mozdonyt (mint komponens) reprezentál. Ehhez lehet még vagonokat csatlakoztatni, amik együtt mozognak.

- Ősosztályok

  TrainComponent

- Interfészek

- Attribútumok
  Nem rendelkezik ősosztályon túli attribútumokkal.

- Metódusok

  + void inTunnel(): Egy flag-et állít attól függően hogy alagútba be vagy ki lép a vonat.
  + void destroy(): Akkor hívódik ha a mozdony egy másik mozdonnyal ütközött.
  + void collision(l: Locomotive): Ütközés esetén hívódik az őt tároló Collection-től, paraméterként az ütközésben részvevő másik mozdonnyal.
  + void placeLocomotive(current: Component): Pálya létrehozásakor megadott elemre helyezi a mozdonyt. (Rail placeLocomotive-ja hívja, ő hívja a mögötte lévő elemek place-ét)

8.1.6. Rail

- Felelősség
  A Rail osztály egy sínt reprezentál, amin a vonat közlekedik, ez a pálya egyéb elemeit köti össze egymással.

- Ősosztályok

  Component

- Interfészek
  Nem valósít meg interface-t.

- Attribútumok
  Nincsenek attribútumai.

- Metódusok
  Nincsenek publikus metódusai.
  - void inTunel(): Egy flag-et állít attól függően hogy alagútba be vagy ki lép a vonat.
  - void destroy(): Akkor hívódik ha a mozdony egy másik mozdonnyal ütközött.

- collision(l: Locomotive): Ütközés detektálás

+ placeLocomotive(side: char, locomotive: Locomotive): Meghívja a Locomotive azonos nevű függvényét.

8.1.7. Siding

- Felelősség
  Vakvágányokat reprezentáló osztály.
- Ősosztályok

  Component
- Interfészek
  Nem valósít meg interface-t.
- Attribútumok
  Nincsenek attribútumok.
- Metódusok
  Nincsenek metódusai.

8.1.8. Station

- Felelősség
  A station egy állomást reprezentáló osztály. Színnel rendelkezik, ezt a beérkező mozdonyoknak átadja, így valósul meg a leszállás.
- Ősosztályok

  Component
- Interfészek
  Nem valósít meg interface-t.
- Attribútumok

  - Color color: Megadja az állomás színét
- Metódusok

  + Station(x0: int, y0: int, x1: int, y1: int, color: Color): Konstruktor. Beállítja az állomás pozícióját és színét.

8.1.9. Switch

- Felelősség
  A Switch egy váltót reprezentáló osztály. A váltóval a sínek között lehet váltani. Az állása a játékostól függ, ő irányítja, ezzel befolyásolva a vonat mozgását.
- Ősosztályok

  Component

- Interfészek
  Nem valósít meg interface-t.
- Attribútumok

  - bool state: Váltó aktuális állapota
  - Component CEnd: Váltó 3. oldala

- Metódusok

  + void switch(): A váltás műveletét megvalósító metódus.

8.1.10. TrainComponent

- Felelősség
  A vonat alkotóelemeihez absztrakt ősosztály.
- Ősosztályok

  Nincs ősosztály

- Interfészek
  Nem valósít meg interface-t.
- Attribútumok

  - TrainComponent previous: vonatban őt megelőző elem
  - TrainComponent next: vonatban őt követő elem
  - Component current: az a pályaelem amin épp rajta van
  - bool destroyed: flag, jelzi hogy a vonat ütközött-e
  - bool derailed: flag, jelzi hogy a vonat kisiklott-e
  - bool inTunnel: flag, jelzi hogy a vonat alagútban van-e

- Metódusok

  + void void derail(): kisiklatja az adott elemet (és ezzel a vonatot)
  + void int atStation(Color color, int passengerNumber): jelzés, hogy állomásra ért a vonat, paraméterként kapja az állomás színét és az ott lévő utasok számát
  + void void step(): lépteti az elemet
  + void connect(char side, TrainComponent tc): kocsik egymáshoz kapcsolását teszi lehetővé side oldalra kapcsolja tc-t
  + void place(current: Component): pálya létrehozása során beállítja az elem kezdőhelyzetét

8.1.11. Tunnel

- Felelősség
  A Tunnel egy pályán elhelyezhető alagutat reprezentáló osztály. A vonatok át tudnak haladni ezen az alagúton. A játékos jogosultságai közé tartozik, hogy áthelyezheti az alagút egyes végeit, így változtatva a pályát.
- Ősosztályok

  Nincs ősosztály
- Interfészek
  Nem valósít meg interface-t.
- Attribútumok

  - TunnelEnd EndA: Tárolja az alagút egyik végét.
  - TunnelEnd EndB: Tárolja az alagút másik végét.
  - bool trainInside: Megmondja, hogy ven-e vonat az alagútban.
  - bool active: Megadja, hogy aktív-e az alagút.
- Metódusok

  + void setEnd(t: TunnelEnd): Beállítható az alagút egy vége.
  + bool isActive(): visszatér az aktuális alagútvég állapotával
  + bool trainInside(): visszatér igaz/hamis értékkel, attól függően, hogy van-e az alagútban vonat.

8.1.12. TunnelEnd

- Felelősség
  Olyan pontokat reprezentál ahova alagútszáj helyezhető.
- Ősosztályok

  Component
- Interfészek
  Nem valósít meg interface-t.
- Attribútumok

  - bool active: Aktív-e vagy nem.
- Metódusok

  – void activate(): Aktiválja az aktuális pontot alagútszájnak
  – bool isActive(): Megmondja, hogy aktív-e az aktuális pont.

8.1.13. Wagon

- Felelősség
  A Wagon osztály a mozdonyhoz csatlakoztatható vagont reprezentálja. A vagon a mozdonyhoz hasonlóan egy komponens, ezeket egymás után lehet kötni. A vagonban utaznak az utasok. Saját színnel rendelkezik, így az azonos színű állomáson az utasok leszállhatnak, ha ez lehetséges.
- Ősosztályok

  TrainComponent
- Interfészek
- Attribútumok

  - Color color - a kocsi színe
  - int passengerNumber - utasok száma
- Metódusok

  - void getOn(int n) - n számú utas felszállása a kocsira
  - void getOff(int n) - n számú utas leszállása a kocsiról
  + Wagon(int passengerNumber, Color color) - kontruktorm, beállítja az utasok kezdeti számát és a kocsi színét

  Többi metódus leírása ősosztálynál.

8.2. A tesztek részletes tervei, leírásuk a teszt nyelvén

Vonatot alkotó kocsik hossza egységnyi, a kiindulási sínpárjuk legalább olyan hosszú hogy a teljes vonat elférjen rajta.

8.2.1. Alagútvég elhelyezése

- Leírás
  Az alagútvég áthelyezésének tesztelése.

  Ellenőrizhető, hogy az alagút egy végének elhelyezése sikeres-e, a vonatok tudják-e azon keresztül folytatni útjukat.
- Bemenet

  create m 3.38 3.75 Rail 6.64 6.12
  create m 6.64 6.12 TunnelEnd
  create m 8.89 8.75 TunnelEnd
  create m 8.89 8.75 Rail 9.76 9.49
  connect m 1 B 2 A
  connect m 2 B 3 A
  connect m 3 B 4 A
  activate 2

- Elvárt kimenet
  created Locomotive with id: 1
  created Rail with id: 1
  created TunnelEnd with id: 2
  created TunnelEnd with id: 3
  created Rail with id: 4
  connect (1 to A side of 2) and (2 to B side of 1)
  connect (2 to A side of 3) and (3 to B side of 2)
  connect (3 to A side of 4) and (4 to B side of 3)
  activated 2 true

8.2.2. Állomásra érkezés

- Leírás
  Ellenőrizhető, hogy a vonat állomásra érkezés esetén megtörténnek-e specifikált interakciók, hogy pl. szenes kocsikba nem történik fel/leszállás.

- Bemenet
  create m 0.15 0.03 Rail 4.43 5.12
  create m 4.43 5.12 Station red 3
  create m 4.43 5.12 Rail 6.62 6.57
  create m 6.62 6.57 Station blue 2
  connect m 1 B 2 A
  connect m 2 B 3 A
  create t Locomotive
  create t CoalWagon
  create t Wagon red 10
  connect t 1 2 connect t 2 3 place 1 1 B
  step
  ..
  step
  list map

- Elvárt kimenet
  created Rail with id: 1
  created Station with id: 2
  created Rail with id: 3
  created Station with id: 4
  connected 2 to A side of 3 and 3 to B side of 2
  connected 3 to A side of 4 and 4 to B side of 3
  created Locomotive with id: 1
  created Coalwagon with id: 2
  created Wagon with id: 3
  ..
  Rail 1 from 0.15 0.03 to 4.43 5.12
  A side connected to null
  B side connected to 2
  Station 2 at 4.43, 5.12 with 3 passengers and red color
  A side connected to 1
  B side connected to 3
  Rail 3 from 4.43 5.12 to 6.62 6.57

A side connected to 2
B side connected to 4
Station 4 at 4.43, 5.12 with 0 passengers and blue color A side connected to 3
B side connected to null

8.2.3. Váltás

- Leírás
  Váltó átállításának tesztelése. Annak ellenőrzésére szolgál, hogy megtörténik-e a váltók átállítása.

- Bemenet
  create t Locomotive
  create m 3.38 3.75 Rail 6.64 6.12
  create m 6.64 6.12 Switch
  create m 6.64 6.12 Rail 9.76 9.49
  create m 6.64 6.12 Rail 4.52 4.78
  connect m 1 B 2 A
  connect m 2 B 3 A
  connect m 2 C 4 A
  place 1 1 B switch 2
  step 1 (sokszor)

- Elvárt kimenet
  created Locomotive with id: 1
  created Rail with id: 1
  created Switch with id: 2
  created Rail with id: 3
  created Rail with id: 4
  connected (1 to A side of 2) and (2 to B side of 1)
  connected (2 to A side of 3) and (3 to B side of 2)
  connected (2 to A side of 4) and (4 to B side of 2)
  switched 2 new position: C
  sok stepped... (hatodikra lép 4-es id-re:)
  stepped 1 new position 4 0.0

8.2.4. Vonat haladása sínváltás nélkül

- Leírás
  Vonat mozgatásának tesztelése. Ellenőrzi, hogy a vonat megfelelően halad-e előre a síneken alapesetben, ha nem megy át váltón, nem érkezik állomásra, stb.

- Bemenet
  create t Locomotive
  create m 3.38 3.75 Rail 6.64 6.12
  connect 1 B 2 A
  place 1 1 B step 1
  ..

- Elvárt kimenet
  created Locomotive with id: 1
  created Rail with id: 1
  stepped 1 new position 1 1.0

8.2.5. Vonat haladása sínváltással

- Leírás
  Ellenőrzi, hogy a vonat megfelelően halad-e az egyes elemek között.

- Bemenet
  create t Locomotive
  create m 3.38 3.75 Rail 6.64 6.12
  create m 6.64 6.12 Rail 9.76 9.49
  create m 9.76 9.49 Rail 12.3 12.75
  create m 9.76 9.49 Switch
  connect m 1 A 4 B
  connect m 2 A 4 A
  connect m 3 A 4 C
  place 1 1 B
  step 1 (sokszor, egymás után)

- Elvárt kimenet
  created Locomotive with id: 1
  created Rail with id: 1
  created Rail with id: 2
  created Rail with id: 3
  created Switch with id: 4 connected (1 to B side of 4) and (4 to A side of 1)
  connected (2 to A side of 4) and (4 to A side of 2)
  connected (3 to C side of 4) and (4 to A side of 3)
  Első step-re:
  stepped 1 new position 1 1.0
  Utána sokadikra:
  stepped 1 new position 4 0.0
  Utána közvetlen stepped 1 new position 2 0.0

8.2.6. Kocsik húzása

- Leírás
  Annak ellenőrzése, hogy az összekapcsolt vonat képes-e együtt mozogni a terepasztalon, azaz az elől lévő kocsik húzzák-e a mögöttük lévőket.

- Bemenet
  create t Locomotive
  create t Wagon blue 10
  create t Wagon red 10
  connect t 1 2 connect t 2 3 create m 3.38 3.75 Rail 6.64 6.12
  place 1 1 A

step 1
list train

- Elvárt kimenet
  created Locomotive with id: 1
  created Wagon with id: 2
  created Wagon with id: 3
  created Rail with id: 1
  stepped 1 new position 1 1.0
  which moved 2 new position 1 1.0
  which moved 3 new position 1 1.0
  Locomotive 1 on 1 at 1.0 next: 2;
  collided false; destroyed false; derailed false
  Wagon 2 on 1 at 1.0 next: 3, previous 1;
  collided false; destroyed false; derailed false
  passenger number: 10, color: java.awt.Color[r=0,g=0,b=255]
  Wagon 3 on 1 at 1.0 next: 0, previous 2;
  collided false; destroyed false; derailed false
  passenger number: 10, color: java.awt.Color[r=255,g=0,b=0]

8.2.7. Vonatok ütközése

- Leírás
  Vonatok ütközésének a tesztelése. Ellenőrizhető,hogy két vonat ütközése esetén azok ténylegesen megsemmisülnek-e.

- Bemenet
  create t Locomotive
  create t Locomotive
  create m 1.0 1.0 Rail 1.0 3.0
  create m 1.0 3.0 Rail 1.0 5.0
  connect m 1 B 2 A
  place 1 1 B
  place 2 2 B
  step 1
  step 2
  list train
  step 1
  list train
  .
  .
  step 1
  list train

- Elvárt kimenet
  created Locomotive with id: 1
  created Locomotive with id: 2
  created Rail with id: 1

created Rail with id: 2
connected (1 to A side of 2) and (2 to B side of 1)
stepped 1 new position 2 6.64 6.12
Locomotive 1 on 1 at 1.0 next: 0;
collided true; destroyed true; derailed false

8.2.8. Vakvágányra lépés

- Leírás
  A vakvágányra lépés tesztelése. Tesztelhető, hogy ha a vonat vakvágányra lép, akkor ténylegesen kisiklik-e, és nem tudja folytatni az útját.

- Bemenet
  create m 4 4 Rail 5 5
  create m 5 5 Siding
  connect m 1 B 2 A
  create t Locomotive
  place 1 1 B
  step 1 step 1 list train (kilistázzuk a státuszát)

- Elvárt kimenet
  created Rail with id: 1
  created Siding with id: 2
  connected (1 to A side of 2) and (2 to B side of 1)
  created Locomotive with id 1:
  stepped 1 new position: 1 1.0
  stepped 1 new position: 2 0.0
  Locomotive 1 on 2 at 0.0; next: 0;
  collided false; destroyed true; derailed true

8.2.9. Utasok leszállása

- Leírás
  Az utasok megfelelő leszállásának ellenőrzése. Tesztelhető, hogy az utasok a megfelelő színű állomásokon szállnak-e le, illetve olyan utas nem tud leszállni, aminek színe nem egyezik.

- Bemenet
  8.2.2 teszteset tartalmazza ezt is.

- Elvárt kimenet
  8.2.2 teszteset tartalmazza ezt is.

8.2.10. Váltón kisiklás

- Leírás
  Váltó átállítása úgy, hogy azon vonat halad át. Ellenőrizhetjük, hogy egy vonat kisiklik-e ha átállítunk egy váltót amin éppen áthalad.

- Bemenet
  create m 5 5 Switch
  create m 4 4 Rail 5 5
  connect m 2 B 1 A
  create t Locomotive
  place 1 2 B
  step
  step
  switch 1 (átállítjuk a váltót)
  list train (kilistázzuk a státuszát)

- Elvárt kimenet
  created Switch with id: 1
  created Rail with id: 2
  connected (2 to A side of 1) and (1 to B side of 2)
  created Locomotive with id: 1
  stepped 1 new position 1.0
  stepped 1 new position 1.0
  [Switch].switch()
  Locomotive 1 on 1 at 0.0;
  next: 0; collided false; destroyed true; derailed true

8.3. A tesztelést támogató programok tervei

A tesztelést egy segédprogram fogja segíteni. A program tervezi mellett bemutatásra kerülnek az elvek, ami alapján működik, illetve hogy kimenetét hogyan célszerű lekezelni. A tesztelőnek lehetősége lesz fix teszteseteket elindítani. A program generál a belső működés alapján egy kimentet. A kimenetben egyértelműen látszik, hogy mi történt az utasítások végrehajtása során.

Példa a kiemenetre: stepped <mozdony id> new position: 3 1

A kimenetből ezáltal kiolvasható, hogy melyik objektum pontosan mit csinált, milyen változás történt vele.

Ha a teszteset lefutott, az megjelenik a konzolos kimeneten. Ezzel egyidőben egy kimeneti fájlba is íródik, így a program bezárása után is elérhető lesz a futtatott teszt kimenetele. Ennek során a tesztelő, akár a megfelelő diagramokat tekintve össze tudja hasonlítani, hogy a program adott esetben helyesen, vagy hibásan működik. A tesztesetekhez biztosítva van minden esetben azokat leíró szekvencia, hiszen a szekvenciákon leírt működés tesztelhető. Természetesen pontos egyezést is lehet vizsgálni, minden egyes tesztesethez van elvárt kimenet. Tehát nem magához a tesztesethez van diagram, hanem ahhoz, amit ábrázol, így könnyen megállapítható hogy helyes-e a működés. Ez csak egy plusz opciója az ellenőrzésnek. Amennyiben az elvárt kimenetre hagyatkozunk, akkor elég csak a kapottat összehasonlítani azzal.

A tesztelés menete a következő: A program felvázolja a teszteseteket. A tesztelő kiválaszt egy adott esetet, majd lefuttatja. Ezek az esetek természetesen fixek, vagy minimálisan, vagy egyáltalán nem várnak felhasználói inputot. Ezzel együtt viszonylag egyszerűek is, például egy mozdonyhoz 2 kocsi hozzácsatolása ilyen téren nem különbözik 9 kocsi hozzácsatolásától, hiszen csak ugyan az a folyamat játszódna le többször, és átláthatatlanabbá tenné a kimenetet. A lényeg a funkciók megfelelő tesztelésén van, így a program esetei arra törekednek, hogy a lényeges funkciókat érintsék. Lehetőség van több fájlba is menteni, valamint tetszőleges fájlba menteni szintén. Ugyanazon esetet többször is le lehet futtatni a program segítségével, így meggyőződni,

hogy minden alkalommal ugyan azt a kimenetet produkálja.

8.4. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017.04.01 13:00	2 óra	**Erős**	Tevékenység: 8.1 elkezdése
2017.04.03 21:00	1 óra	**Erős**	Tevékenység: 8.1 befejezés
2017.04.03 21:00	2 óra	**Dombai**	Tevékenység: 8.1 befejezés
2017.04.03 22:00	1 óra	**Márton**	Tevékenység: 8.4 a segédprogram tervei
2017.04.04 9:00	1 óra	**Székely**	Tevékenység: 8.2
2017.04.04 10:00	1 óra	**Székely** **Erős** **Dombai**	Megbeszélés; 8.2 véglegesítése, javítások.

10. Prototípus beadása

10.0. Tesztesetekben történő változások

A teszteset ki és bemenetei a korábban említett implementációs kérdések változása miatt történt (sebesség, pályahosszak, stb. implementáció függő) - azonban a tesztek lényege nem változott.

10.0.1. Alagútvég elhelyezése

- Leírás
  Az alagútvég áthelyezésének tesztelése.

  Ellenőrizhető, hogy az alagút egy végének elhelyezése sikeres-e, a vonatok tudják-e azon keresztül folytatni útjukat.

- Bemenet

  create m 3.38 3.75 Rail 6.64 6.12
  create m 6.64 6.12 TunnelEnd
  create m 8.89 8.75 TunnelEnd
  create m 8.89 8.75 Rail 9.76 9.49
  connect m 1 B 2 A
  connect m 2 B 3 A
  connect m 3 B 4 A
  activate 2

- Elvárt kimenet
  created Locomotive with id: 1
  created Rail with id: 1
  created TunnelEnd with id: 2
  created TunnelEnd with id: 3
  created Rail with id: 4
  connect (1 to A side of 2) and (2 to B side of 1)
  connect (2 to A side of 3) and (3 to B side of 2)
  connect (3 to A side of 4) and (4 to B side of 3)
  activated 2 true

10.0.2. Állomásra érkezés

- Leírás
  Ellenőrizhető, hogy a vonat állomásra érkezés esetén megtörténnek-e specifikált interakciók, hogy pl. szenes kocsikba nem történik fel/leszállás.

- Bemenet
  create m 0.15 0.03 Rail 4.43 5.12
  create m 4.43 5.12 Station red 3
  create m 4.43 5.12 Rail 6.62 6.57
  create m 6.62 6.57 Station blue 2
  connect m 1 B 2 A

connect m 2 B 3 A
create t Locomotive
create t CoalWagon
create t Wagon red 10
connect t 1 2
connect t 2 3
place 1 1 B
step
..
step (sokszor (7-szer))
list train
list map

- Elvárt kimenet
  created Rail with id: 1
  created Station with id: 2
  created Rail with id: 3
  created Station with id: 4
  connected (1 to A side of 2) and (2 to B side of 1)
  connected (2 to A side of 3) and (3 to B side of 2)
  created Locomotive with id: 1
  created Coalwagon with id: 2
  created Wagon with id: 3
  stepped 1 new position 1 2.0
  which moved 2 new position 1 2.0
  which moved 3 new position 1 2.0
  .. (sokszor)
  Locomotive 1 on 2 at 0.0 next: 2;
  collided false; destroyed false; derailed false
  Wagon 2 on 2 at pos; next: 3, previous: 1; collied false; destoyed false; derailed false
  Wagon 3 on 2 at 0.0 next: 0, previous 2; collided false; destroyed false; derailed false
  passenger number: 3, color: java.awt.Color[r=255,g=0,b=0]
  Rail 1 at 0.15, 0.03 to 4.43, 5.12
  A side connected to 0
  B side connected to 2
  Station 2 at 4.43, 5.12 with 10 passengers and java.awt.Color[r=255,g=0,b=0] color
  A side connected to 1
  B side connected to 3
  Rail 3 at 4.43, 5.12 to 6.62, 6.57
  A side connected to 2
  B side connected to 0
  Station 4 at 6.62, 6.57 with 2 passengers and java.awt.Color[r=0,g=0,b=255] color
  A side connected to 0
  B side connected to 0
  Tunnel is deactivated
  A side connected to 0
  B side connected to 0

10.0.3. Váltás

- Leírás
  Váltó átállításának tesztelése. Annak ellenőrzésére szolgál, hogy megtörténik-e a váltók átállítása.

- Bemenet
  create t Locomotive
  create m 3.38 3.75 Rail 6.64 6.12
  create m 6.64 6.12 Switch
  create m 6.64 6.12 Rail 9.76 9.49
  create m 6.64 6.12 Rail 4.52 4.78
  connect m 1 B 2 A
  connect m 2 B 3 A
  connect m 2 C 4 A
  place 1 1 B
  switch 2
  step 1 (sokszor hatodikra lép át a váltón)

- Elvárt kimenet
  created Locomotive with id: 1
  created Rail with id: 1
  created Switch with id: 2
  created Rail with id: 3
  created Rail with id: 4
  connected (1 to A side of 2) and (2 to B side of 1)
  connected (2 to A side of 3) and (3 to B side of 2)
  connected (2 to A side of 4) and (4 to C side of 2)
  switched 2 new position: C
  sok stepped... (hatodikra lép 4-es id-re:)
  stepped 1 new position 4 0.0

10.0.4. Vonat haladása sínváltás nélkül

- Leírás
  Vonat mozgatásának tesztelése. Ellenőrzi, hogy a vonat megfelelően halad-e előre a síneken alapesetben, ha nem megy át váltón, nem érkezik állomásra, stb.

- Bemenet
  create t Locomotive
  create m 3.38 3.75 Rail 6.64 6.12
  connect 1 B 2 A
  place 1 1 B
  step 1
  ..

- Elvárt kimenet
  created Locomotive with id: 1
  created Rail with id: 1
  stepped 1 new position 1 1.0

10.0.5. Vonat haladása sínváltással

- Leírás
  Ellenőrzi, hogy a vonat megfelelően halad-e az egyes elemek között.

- Bemenet
  create t Locomotive
  create m 3.38 3.75 Rail 6.64 6.12
  create m 6.64 6.12 Rail 9.76 9.49
  create m 9.76 9.49 Rail 12.3 12.75
  create m 9.76 9.49 Switch
  connect m 1 A 4 B
  connect m 2 A 4 A
  connect m 3 A 4 C
  place 1 1 B
  step 1 (sokszor, egymás után)

- Elvárt kimenet
  created Locomotive with id: 1
  created Rail with id: 1
  created Rail with id: 2
  created Rail with id: 3
  created Switch with id: 4
  connected (1 to B side of 4) and (4 to A side of 1)
  connected (2 to A side of 4) and (4 to A side of 2)
  connected (3 to C side of 4) and (4 to A side of 3)
  Első step-re:
  stepped 1 new position 1 1.0
  Utána sokadikra:
  stepped 1 new position 4 0.0
  Utána közvetlen
  stepped 1 new position 2 0.0

10.0.6. Kocsik húzása

- Leírás
  Annak ellenőrzése, hogy az összekapcsolt vonat képes-e együtt mozogni a terepasztalon, azaz az elől lévő kocsik húzzák-e a mögöttük lévőket.

- Bemenet
  create t Locomotive
  create t Wagon blue 10
  create t Wagon red 10
  create m 3.38 3.75 Rail 6.64 6.12
  connect t 1 2
  connect t 2 3
  place 1 1 A
  step 1

list train

- Elvárt kimenet
  created Locomotive with id: 1
  created Wagon with id: 2
  created Wagon with id: 3
  created Rail with id: 1
  connected 0 to 1
  connected 1 to 2
  stepped 1 new position 1 1.0
  which moved 2 new position 1 1.0
  which moved 3 new position 1 1.0
  Locomotive 1 on 1 at 1.0 next: 2;
  collided false; destroyed false; derailed false
  Wagon 2 on 1 at 1.0 next: 3, previous 1;
  collided false; destroyed false; derailed false
  passenger number: 10, color: java.awt.Color[r=0,g=0,b=255]
  Wagon 3 on 1 at 1.0 next: 0, previous 2;
  collided false; destroyed false; derailed false
  passenger number: 10, color: java.awt.Color[r=255,g=0,b=0]

10.0.7. Vonatok ütközése

Ez a teszt nem működik.

- Leírás
  Vonatok ütközésének a tesztelése. Ellenőrizhető,hogy két vonat ütközése esetén azok ténylegesen megsemmisülnek-e.

- Bemenet
  create t Locomotive
  create t Locomotive
  create m 3.38 3.75 Rail 6.64 6.12
  create m 6.64 6.12 Rail 9.76 9.49
  connect m 1 B 2 A
  place 1 1 B
  place 2 2 A
  step 1
  list train

- Elvárt kimenet
  created Locomotive with id: 1
  created Locomotive with id: 2
  created Rail with id: 1
  created Rail with id: 2
  connected (1 to A side of 2) and (2 to B side of 1)
  stepped 1 new position 2 6.64 6.12
  Locomotive 1 on 1 at 1.0 next: 0;
  collided true; destroyed true; derailed false

10.0.8. Vakvágányra lépés

- Leírás
  A vakvágányra lépés tesztelése. Tesztelhető, hogy ha a vonat vakvágányra lép, akkor ténylegesen kisiklik-e, és nem tudja folytatni az útját.

- Bemenet
  create m 4 4 Rail 5 5
  create m 5 5 Siding
  connect m 1 B 2 A
  create t Locomotive
  place 1 1 B
  step 1
  step 1
  list train (kilistázzuk a státuszát)

- Elvárt kimenet
  created Rail with id: 1
  created Siding with id: 2
  connected (1 to A side of 2) and (2 to B side of 1)
  created Locomotive with id 1:
  stepped 1 new position: 1 1.0
  stepped 1 new position: 2 0.0
  Locomotive 1 on 2 at 0.0; next: 0;
  collided false; destroyed true; derailed true

10.0.9. Utasok leszállása

- Leírás
  Az utasok megfelelő leszállásának ellenőrzése. Tesztelhető, hogy az utasok a megfelelő színű állomásokon szállnak-e le, illetve olyan utas nem tud leszállni, aminek színe nem egyezik.

- Bemenet
  8.2.2 teszteset tartalmazza ezt is.

- Elvárt kimenet
  8.2.2 teszteset tartalmazza ezt is.

10.0.10. Váltón kisiklás

- Leírás
  Váltó átállítása úgy, hogy azon vonat halad át. Ellenőrizhetjük, hogy egy vonat kisiklik-e ha átállítunk egy váltót amin éppen áthalad.

- Bemenet
  create m 5 5 Switch
  create m 4 4 Rail 5 5
  connect m 2 B 1 A

create t Locomotive
place 1 2 B
step
step
switch 1 (átállítjuk a váltót)
list train (kilistázzuk a státuszát)

- Elvárt kimenet
created Switch with id: 1
created Rail with id: 2
connected (2 to A side of 1) and (1 to B side of 2)
created Locomotive with id: 1
stepped 1 new position 2 1.0
stepped 1 new position 1 0.0
Locomotive 1 on 1 at 0.0; next: 0;
collided false; destroyed true; derailed true

10.1. Fordítási és futtatási útmutató

10.1.1. Fájllista

Fájl neve	**Méret**	**Keletkezés ideje**	**Tartalom**
Application.java	8 kilobyte	2017.04.17 13:56	Main függvény, Game osztály példányosítása
CoalWagon.java	3 kilobyte	2017.04.17 21:40	Szeneskocsi függvényei
Collection.java	5 kilobyte	2017.04.17 21:40	Szükséges metódusok
Component.java	5 kilobyte	2017.04.17 21:40	Component osztály definiált metódusai
Game.java	10 kilobyte	2017.04.17 21:32	Bemenetek kezelése, osztályok példányosítása, és a megfelelő parancsok lekezelése
Intersection.java	1 kilobyte	2017.04.17 21:40	Kereszteződés függvényei
Locomotive.java	3 kilobyte	2017.04.17 21:40	Mozdony függvényei
Rail.java	1 kilobyte	2017.04.17 21:40	Sín függvényei
Siding.java	2 kilobyte	2017.04.17 21:40	Vakvágány függvényei
Station.java	2 kilobyte	2017.04.17 21:40	Állomás függvényei
Switch.java	2 kilobyte	2017.04.17 21:40	Váltó függvényei
TrainComponent.java	3 kilobyte	2017.04.17 21:40	Vonat elemeinek függvényei
Tunnel.java	2 kilobyte	2017.04.17 21:40	Alagút függvényei
TunnelEnd.java	2 kilobyte	2017.04.17 21:40	Alagútvég függvényei
Wagon.java	4 kilobyte	2017.04.17 21:40	Kocsik függvényei

10.1.2. Fordítás

```
set PATH="%PATH%;C:\Program_Files\Java\jdk1.6.0_18\bin"
chcp 65001
javac -encoding ISO-8859-2 src/*.java
```

10.1.3. Futtatás

```
cd src
java -Dfile.encoding=ISO-8859-2 Application
```

A program a pdf-ből való másolással is tesztelhető, egyszerre több sort másolva is. Minden teszt után újra kell indítani a programot.

10.2. Tesztek jegyzőkönyvei

10.2.1. Alagútvég áthelyezés

Tesztelő neve	**Dombai, Erős, Székely**
Teszt időpontja	2017.04.18. 9:00

10.2.2. Állomásra érkezés

Tesztelő neve	**Dombai, Erős, Székely**
Teszt időpontja	2017.04.18. 9:10

10.2.3. Váltás

Tesztelő neve	**Dombai, Erős, Székely**
Teszt időpontja	2017.04.18. 9:20

10.2.4. Vonat haladása sínváltás nélkül

Tesztelő neve	**Dombai, Erős, Székely**
Teszt időpontja	2017.04.18. 9:30

10.2.5. Vonat haladása sínváltással

Tesztelő neve	**Dombai, Erős, Székely**
Teszt időpontja	2017.04.18. 10:00

10.2.6. Kocsik húzása

Tesztelő neve	**Dombai, Erős, Székely**
Teszt időpontja	2017.04.18. 10:10

10.2.7. Vonatok ütközése

Tesztelő neve	**Dombai, Erős, Székely**
Teszt időpontja	2017.04.18. 10:10
Teszt eredménye	Sikertelen
Lehetséges hibaok	Collection.java-ban ütközésdetektálás nem megfelelő
Változtatások	Nincs idő változtatásra.

10.2.8. Vakvágányra lépés

Tesztelő neve	**Dombai, Erős, Székely**
Teszt időpontja	2017.04.18. 11:30

10.2.9. Utasok leszállása

Tesztelő neve	**Dombai, Erős, Székely**
Teszt időpontja	2017.04.18. 9:50

10.2.10. Váltón kisiklás

Tesztelő neve	**Dombai, Erős, Székely**
Teszt időpontja	2017.04.18. 9:40

10.3. Értékelés

Összesen, a projekt kezdete óta:

Tag	**Munka százalékban**	**Aláírás**
Dombai	32 %	
Erős	18 %	
Márton	15 %	
Székely	15 %	

Tag	Munka százalékban	Aláírás
Szilágyi	17 %	

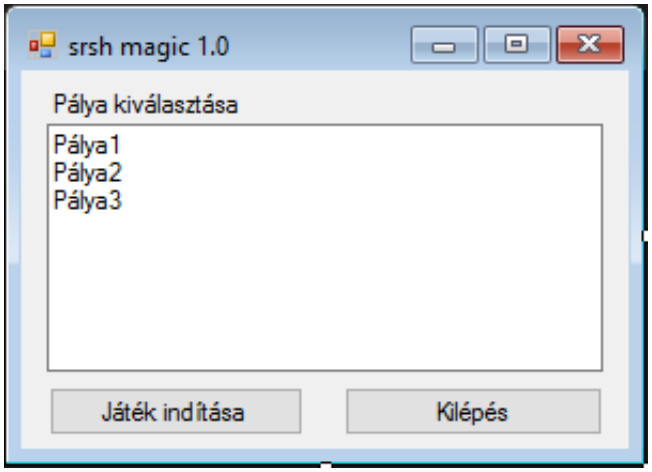
10.4. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017.04.14. 11:00	1 óra	**Dombai**	Tevékenység: Game.java elkezdése
2017.02.15. 14:00	3 óra	**Márton**	Tevékenység: Tesztprogram
2017.02.15. 14:00	2 óra	**Szilágyi**	Tevékenység: Osztályok implementálása 1.0
2017.02.16. 9:00	1 óra	**Márton**	Tevékenység: tesztprogram
2017.02.16. 20:00	1 óra	**Dombai**	Tevékenység: Game.java, Locomotive.java
2017.02.16. 21:00	1 óra	**Márton**	Tevékenység: tesztprogram
2017.02.16. 23:00	1 óra	**Dombai**	Tevékenység: connect, create parancshoz tartozó függvények
2017.02.17. 9:00	2 óra	**Márton**	Tevékenység: tesztprogram
2017.02.17. 11:00	2 óra	**Dombai**	Tevékenység: minden osztályhoz implementációk, list parancs
2017.02.17. 18:00	3 óra	**Erős**	Tevékenység: további függvények
2017.02.17. 18:00	3 óra	**Szilágyi**	Tevékenység: collection.java
2017.02.18. 10:00	3 óra	**Dombai**	Tevékenység: tesztelés, javítások
2017.02.18. 10:00	3 óra	**Erős**	Tevékenység: tesztelés, javítások, fordítási útmutató, fájllista
2017.02.18. 10:00	2 óra	**Székely**	Tevékenység: tesztelés
2017.02.18. 10:00	2 óra	**Szilágyi**	Tevékenység: collection javítása

11. Grafikus felület specifikációja

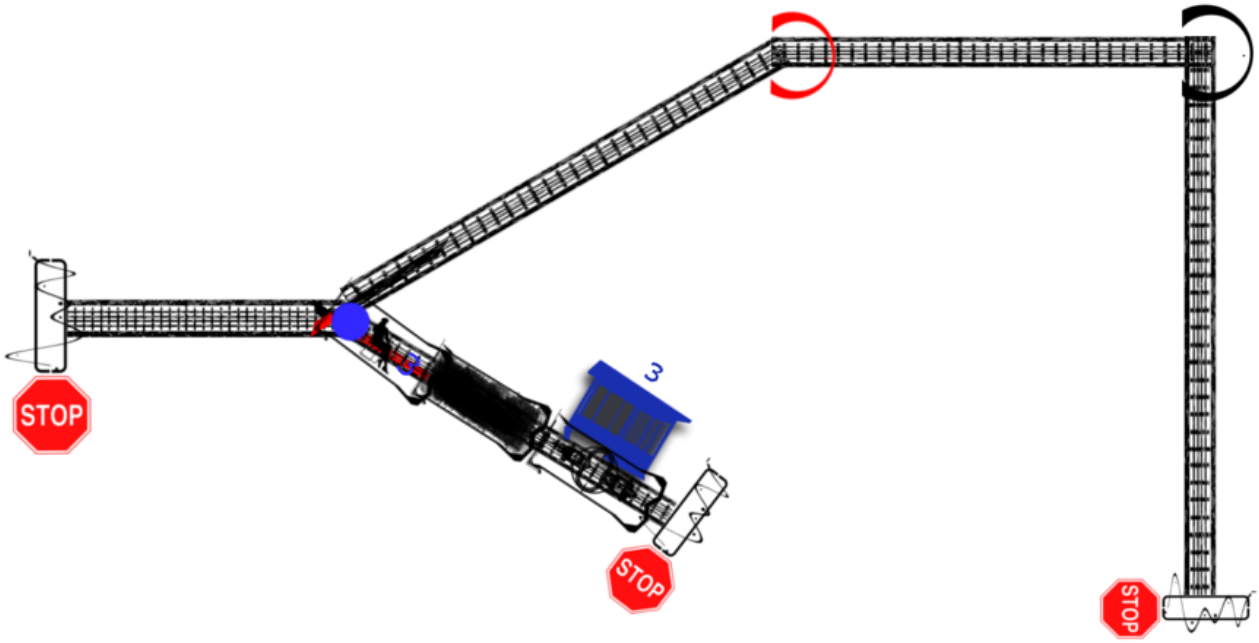
11.1. A grafikus interfész

A szoftvert elindítva az alábbi grafikus ablak jelenik meg:



11.1. ábra. Menü kinézete

Ezen a listából kiválasztva a kívánt pályát a játék indítása gombra kattintva elindítható az adott játék. A kilépés feliratú gombra kattintva kilép az alkalmazás.



11.2. ábra. Példapálya - **csupán illusztráció!**

A játékot megvalósító ablak csupán egy grafikus panelből fog állni. A játéktéren a játék pillanatnyi állapota jelenik meg, az egyes elemekhez ezeket a képeket rendelve:

11.3. ábra. Egyes modellben szereplő elemekhez tartozó képek

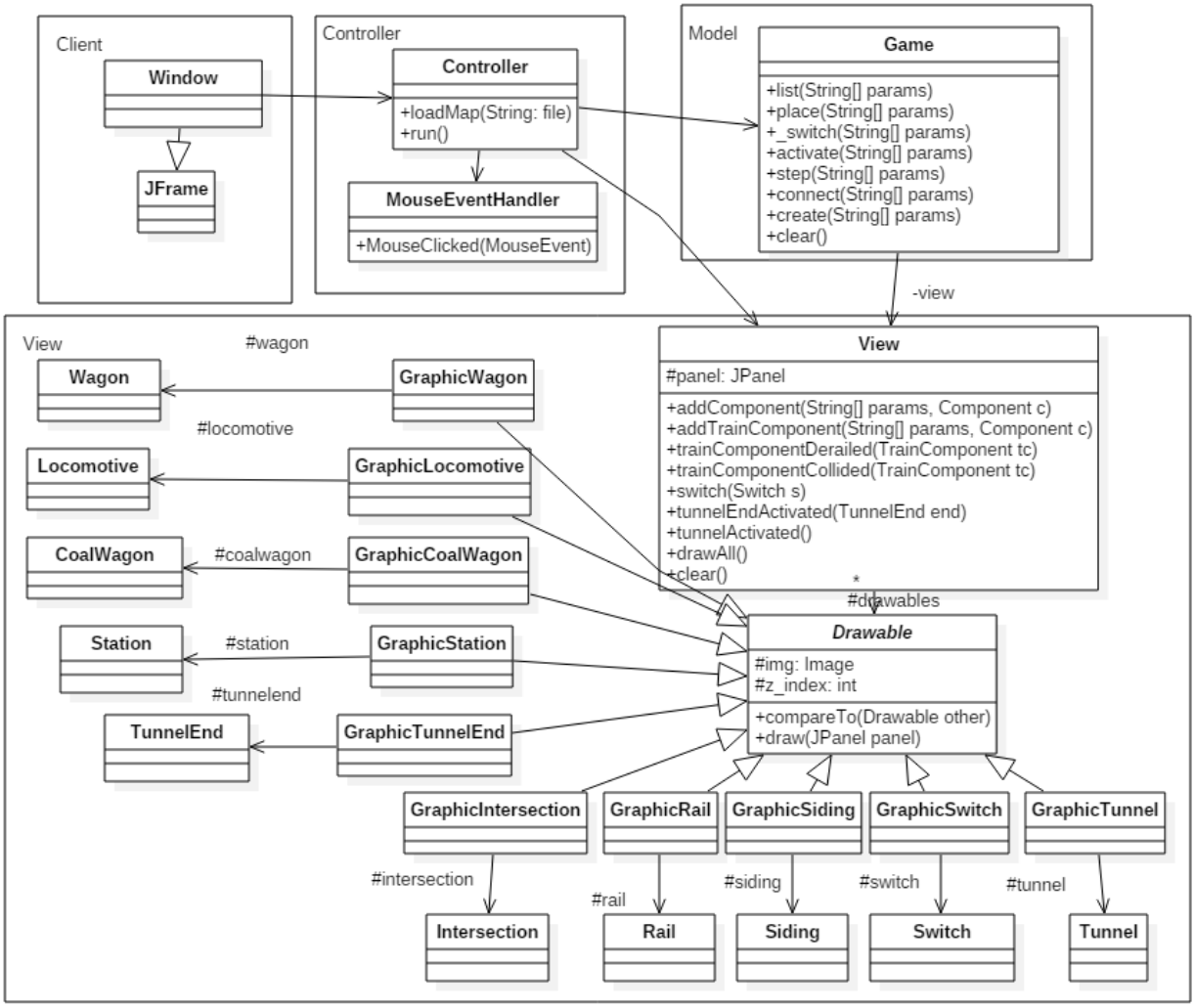
Ezek jelentéstartalma balról jobbra: aktivált alagútvég, deaktivált alagútvég, kocsi, mozdony, sín, állomás, szeneskocsi, vakvágány, váltó.
A kocsik és állomások színe változó, valamint az utasok száma is, ezért az itt nincs ábrázolva.

11.2. A grafikus rendszer architektúrája

11.2.1. A felület működési elve

A grafikus felületet MVC architektúrának megfelelően próbáltuk megtervezni. Minden képernyőn megjelenő objektumnak van egy grafikus párja, ami a **Drawable** ősosztály leszármazottja. A **View** osztály ilyen objektumokat tárol, és minden kirajzolásnál meghívja ezen objektumok draw metódusát. A megjelenítés push alapú. Ha a modelben történt valami változás (amit hívhatunk egy eseménynek), akkor értesíti a **View**-t, hogy valami esemény történt, úgy hogy meghívja az eseménynek megfelelő metódust. Kirajzolásnál minden grafikus objektum, lekérdezi a **Game**-ben lévő párjától az adatait, majd ez alapján kirajzolja magát. A **Controller** osztály a program elején, a **Game** és **View** objektumok létrehozása után feliratkozik a neki megfelelő eseményekre. Ez után már fogadja is az eseményeket. Tervezés során figyeltünk arra is, hogy könnyen bővíthető, cserélhető legyen a rendszer. A **View** és **Graphic** osztályokból való származtatással új GUI-t lehet készíteni, anélkül, hogy meglévő kódban sokat kéne változtatni.

11.2.2. A felület osztály-struktúrája



11.3. A grafikus objektumok felsorolása

11.3.1. Osztály1

- Felelősség
- Ősosztályok
- Interfészek
- Attribútumok
  - attribútum1: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa

  - attribútum2: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa

- Metódusok

  - int foo(Osztály3 o1, Osztály4 o2): metódus leírása, láthatósága (UML jelöléssel)
  - int bar(Osztály5 o1): metódus leírása, láthatósága (UML jelöléssel)

11.3.2. Osztály2

- Felelősség

- Ősosztályok

- Interfészek

- Attribútumok

  - attribútum1: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa
  - attribútum2: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa

- Metódusok

  - int foo(Osztály3 o1, Osztály4 o2): metódus leírása, láthatósága (UML jelöléssel)
  - int bar(Osztály5 o1): metódus leírása, láthatósága (UML jelöléssel)

11.3.3. Controller

- Felelősség
  Az irányításért felelős osztály. Itt történik többek között a pálya betöltése is.
- Ősosztályok
  Nincs ősosztály.
- Interfészek
  Nem valósít meg inteface-t.
- Attribútumok
  Nincsenek publikus attribútumok.
- Metódusok

  + void loadMap(String file): Betölti a paraméterként megadott file-t, ami egy páylát tartalmaz.
  + void run(): Elindítja a játékot.

11.3.4. Drawable

- Felelősség
  A grafikus megjelenítést lehetővé tévő ős osztály, leszármazottjainak biztosítja a grafikus megjelenítéshez szükséges attribútumukat és metódusokat.
- Ősosztályok
  Nincs ősosztálya.
- Interfészek
  Nem valósít meg interfacet.
- Attribútumok
  - # Image img: Egy képet tárol a megjelenítendő objektumról.
  - # int z_index: Megadja az elem másokhoz viszonyított pozícióját (Előrébb vagy hátrébb van-e)
- Metódusok
  - + compareTo(Drawable other): Két drawable összehasonlítása z_index alapján
  - + draw(JPanel panel): Adott elem kirajzolása egy JPanel-ra

11.3.5. Game

- Felelősség
  Ebben az osztályban vannak létrehozva a különböző vezérlő utasítások. A játék működését lehet befolyásolni.
- Ősosztályok
  Nincs ősosztály.
- Interfészek
  Nem valósít meg inteface-t.
- Attribútumok
  Nincsenek publikus attribútumok.
- Metódusok
  - + void create(String[] params): A paraméterként megadott pálya elemet hozza létre a megfelelő (szintén megadott) adatokkal.
  - + void connect(String[] params): Összekapcsol két pálya elemet, az összekapcsolás adatait paraméterként kapja meg.
  - + void step(String[] params): A játékban való léptetést végrehajtó metódus. Kiszámolja, hogy az adott elemet hova kell léptetni, és új pozíciót ad neki.
  - + void activate(String[] params): Egy-egy alagút vég aktiválására ill. deaktiválására szolgál.
  - + void switch(String[] params): Egy váltó állásának megváltoztatására szolgál.
  - + void place(String[] params): Megadott ID-vel rendelkező mozdonyt, illetve a hozzá kapcsolódó kocsikat elhelyezi a sín megadott oldalára. A sín hossza legalább akkora mint a teljes vonat.
  - + void list(String[] params): Létrehozott pályát- és vonatokat alkotó elemek valamit az alagút státuszának lekérdezésére szolgál.

11.3.6. GraphicCoalWagon

- Felelősség
  Szenes vagon grafikus megjelenítésért felelős.
- Ősosztályok
  Drawable az ősosztálya
- Interfészek
  Nem valósít meg interfacet.
- Attribútumok

  # Image img: Egy képet tárol a megjelenítendő objektumról.
  # int z_index: Megadja az elem másokhoz viszonyított pozícióját (Előrébb vagy hátrébb van-e)
- Metódusok

  + compareTo(Drawable other): Két drawable összehasonlítása z_index alapján
  + draw(JPanel panel): Adott elem kirajzolása egy JPanel-ra

11.3.7. GraphicIntersection

- Felelősség
  Kereszteződés grafikus megjelenítésért felelős.
- Ősosztályok
  Drawable az ősosztálya
- Interfészek
  Nem valósít meg interfacet.
- Attribútumok

  # Image img: Egy képet tárol a megjelenítendő objektumról.
  # int z_index: Megadja az elem másokhoz viszonyított pozícióját (Előrébb vagy hátrébb van-e)
- Metódusok

  + compareTo(Drawable other): Két drawable összehasonlítása z_index alapján
  + draw(JPanel panel): Adott elem kirajzolása egy JPanel-ra

11.3.8. GraphicLocomotive

- Felelősség
  Mozdony vagon grafikus megjelenítésért felelős.
- Ősosztályok
  Drawable az ősosztálya

- Interfészek
  Nem valósít meg interfacet.
- Attribútumok

  # Image img: Egy képet tárol a megjelenítendő objektumról.
  # int z_index: Megadja az elem másokhoz viszonyított pozícióját (Előrébb vagy hátrébb van-e)
- Metódusok

  + compareTo(Drawable other): Két drawable összehasonlítása z_index alapján
  + draw(JPanel panel): Adott elem kirajzolása egy JPanel-ra

11.3.9. GraphicRail

- Felelősség
  Sín grafikus megjelenítésért felelős.
- Ősosztályok
  Drawable az ősosztálya
- Interfészek
  Nem valósít meg interfacet.
- Attribútumok

  # Image img: Egy képet tárol a megjelenítendő objektumról.
  # int z_index: Megadja az elem másokhoz viszonyított pozícióját (Előrébb vagy hátrébb van-e)
- Metódusok

  + compareTo(Drawable other): Két drawable összehasonlítása z_index alapján
  + draw(JPanel panel): Adott elem kirajzolása egy JPanel-ra

11.3.10. GraphicSiding

- Felelősség
  Vak vágány grafikus megjelenítésért felelős.
- Ősosztályok
  Drawable az ősosztálya
- Interfészek
  Nem valósít meg interfacet.
- Attribútumok

  # Image img: Egy képet tárol a megjelenítendő objektumról.
  # int z_index: Megadja az elem másokhoz viszonyított pozícióját (Előrébb vagy hátrébb van-e)

- Metódusok

  + compareTo(Drawable other): Két drawable összehasonlítása z_index alapján
  + draw(JPanel panel): Adott elem kirajzolása egy JPanel-ra

11.3.11. GraphicStation

- Felelősség
  Állomás grafikus megjelenítésért felelős.
- Ősosztályok
  Drawable az ősosztálya

- Interfészek
  Nem valósít meg interfacet.
- Attribútumok

  # Image img: Egy képet tárol a megjelenítendő objektumról.
  # int z_index: Megadja az elem másokhoz viszonyított pozícióját (Előrébb vagy hátrébb van-e)

- Metódusok

  + compareTo(Drawable other): Két drawable összehasonlítása z_index alapján
  + draw(JPanel panel): Adott elem kirajzolása egy JPanel-ra

11.3.12. GraphicSwitch

- Felelősség
  Váltó grafikus megjelenítésért felelős.
- Ősosztályok
  Drawable az ősosztálya

- Interfészek
  Nem valósít meg interfacet.
- Attribútumok

  # Image img: Egy képet tárol a megjelenítendő objektumról.
  # int z_index: Megadja az elem másokhoz viszonyított pozícióját (Előrébb vagy hátrébb van-e)

- Metódusok

  + compareTo(Drawable other): Két drawable összehasonlítása z_index alapján
  + draw(JPanel panel): Adott elem kirajzolása egy JPanel-ra

11.3.13. GraphicTunnel

- Felelősség
  Alagút grafikus megjelenítésért felelős.
- Ősosztályok
  Drawable az ősosztálya
- Interfészek
  Nem valósít meg interfacet.
- Attribútumok

  # Image img: Egy képet tárol a megjelenítendő objektumról.
  # int z_index: Megadja az elem másokhoz viszonyított pozícióját (Előrébb vagy hátrébb van-e)
- Metódusok

  + compareTo(Drawable other): Két drawable összehasonlítása z_index alapján
  + draw(JPanel panel): Adott elem kirajzolása egy JPanel-ra

11.3.14. GraphicTunnelEnd

- Felelősség
  Alagútvég grafikus megjelenítésért felelős.
- Ősosztályok
  Drawable az ősosztálya
- Interfészek
  Nem valósít meg interfacet.
- Attribútumok

  # Image img: Egy képet tárol a megjelenítendő objektumról.
  # int z_index: Megadja az elem másokhoz viszonyított pozícióját (Előrébb vagy hátrébb van-e)
- Metódusok

  + compareTo(Drawable other): Két drawable összehasonlítása z_index alapján
  + draw(JPanel panel): Adott elem kirajzolása egy JPanel-ra

11.3.15. GraphicWagon

- Felelősség
  Vagon grafikus megjelenítésért felelős.
- Ősosztályok
  Drawable az ősosztálya

- Interfészek
  Nem valósít meg interfacet.
- Attribútumok

  # Image img: Egy képet tárol a megjelenítendő objektumról.
  # int z_index: Megadja az elem másokhoz viszonyított pozícióját (Előrébb vagy hátrébb van-e)

- Metódusok

  + compareTo(Drawable other): Két drawable összehasonlítása z_index alapján
  + draw(JPanel panel): Adott elem kirajzolása egy JPanel-ra

11.3.16. MouseEventHandler

- Felelősség
  Az egér eseményeit lekezelő osztály. A felhasználó által rendszerbe bevitt eseményeket kezeli le a megfelelő módon.
- Ősosztályok
  Nincs ősosztály.
- Interfészek
  Nem valósít meg inteface-t.
- Attribútumok
  Nincsenek publikus attribútumok.
- Metódusok

  + void MouseClicked(MosueEvent e): Esemény (kattintás) hatására megkapja paraméterként az eseményt, és annak információi alapján lekezeli azt.

11.3.17. View

- Felelősség
  A megjelenítésért felelős osztály.
- Ősosztályok
  Drawable az ősosztálya

- Interfészek
  Nem valósít meg interfacet.
- Attribútumok

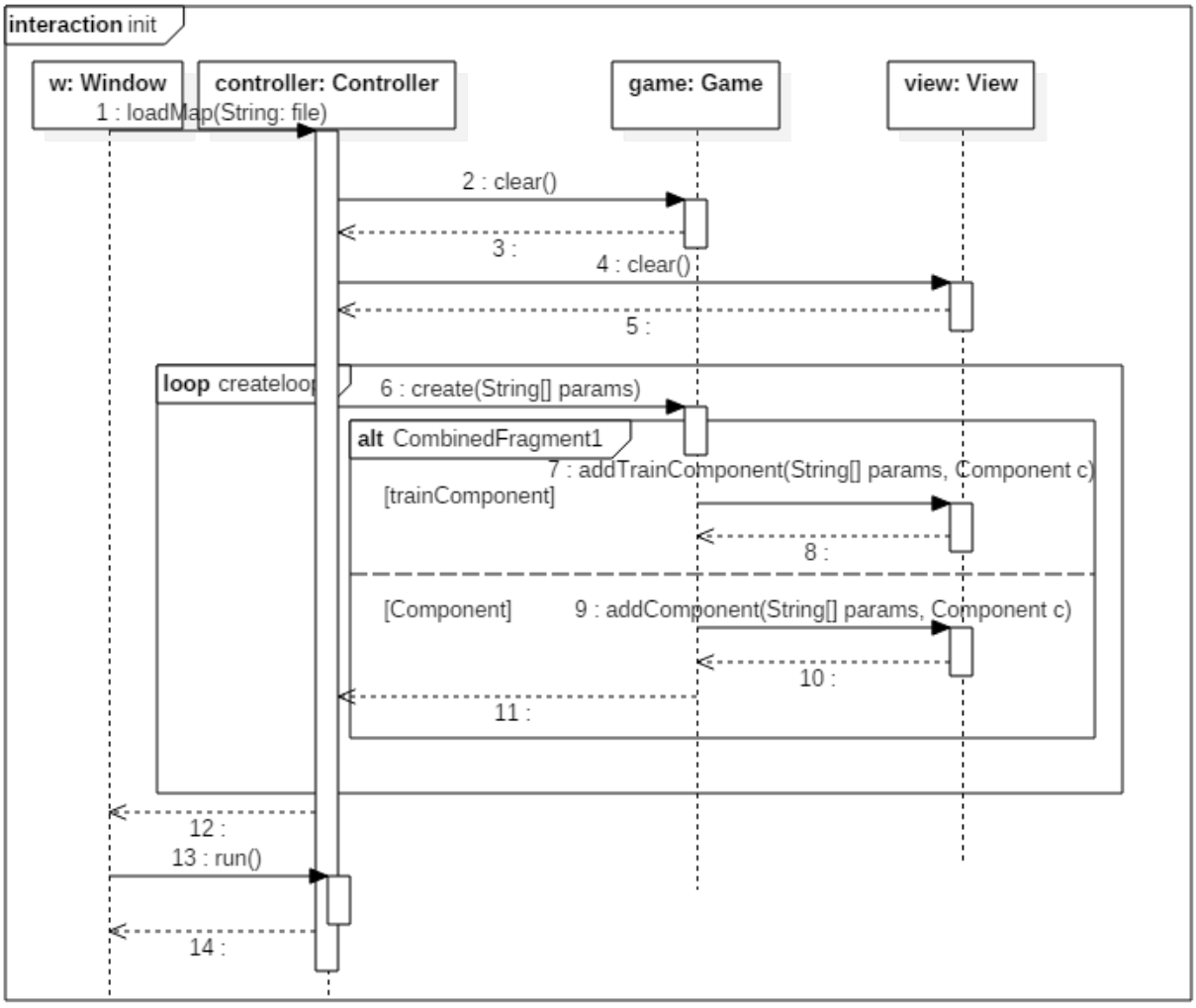
  # JPanel panel: Egy JPanel elemet tárol a megjelenítés miatt.

- Metódusok

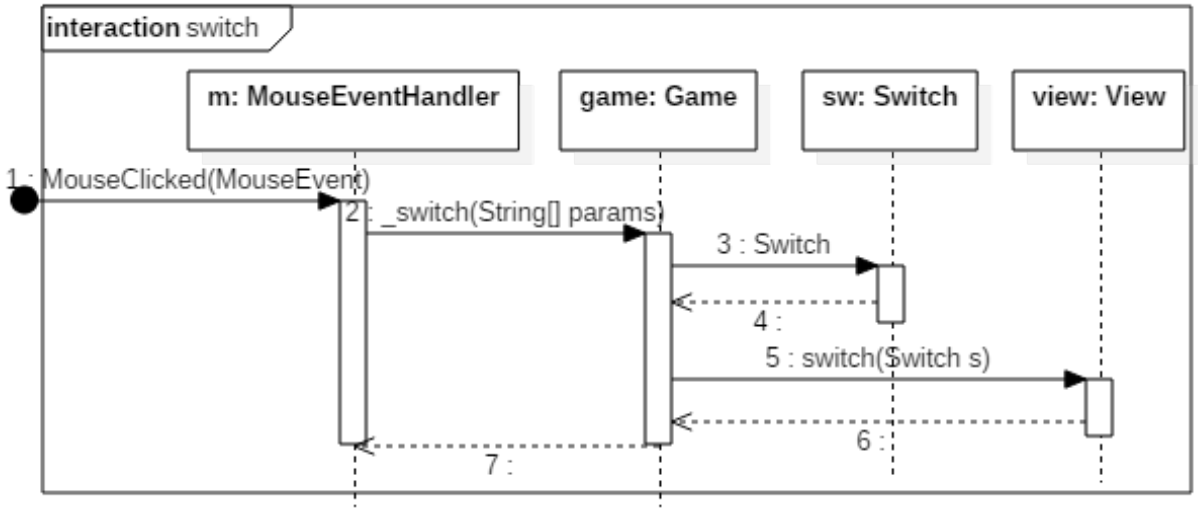
  + addComponent(String[] params, Component c): A paraméterként kapott Componentent-et a params alapján kirajzolja.

+ addTrainComponent(String[] params, TrainComponent c): A paraméterként kapott TrainComponentent-et a params alapján kirajzolja.
+ trainComponentDerailed(TrainComponent tc): Kirajzolja a paraméterként átadott vonat elem kisiklását.
+ trainComponentCollided(TrainComponent tc): Kirajzolja a paraméterként átadott vonat elem ütközését.
+ switch(Switch s): Kirajzolja a paraméterként átadott váltó állapotának megfelelő képet.
+ tunnelEndActivated(TunnelEnd end): Kirajzolja az aktivált alagút véget.
+ tunnelActivated(): Kirajzolja az alagutat.

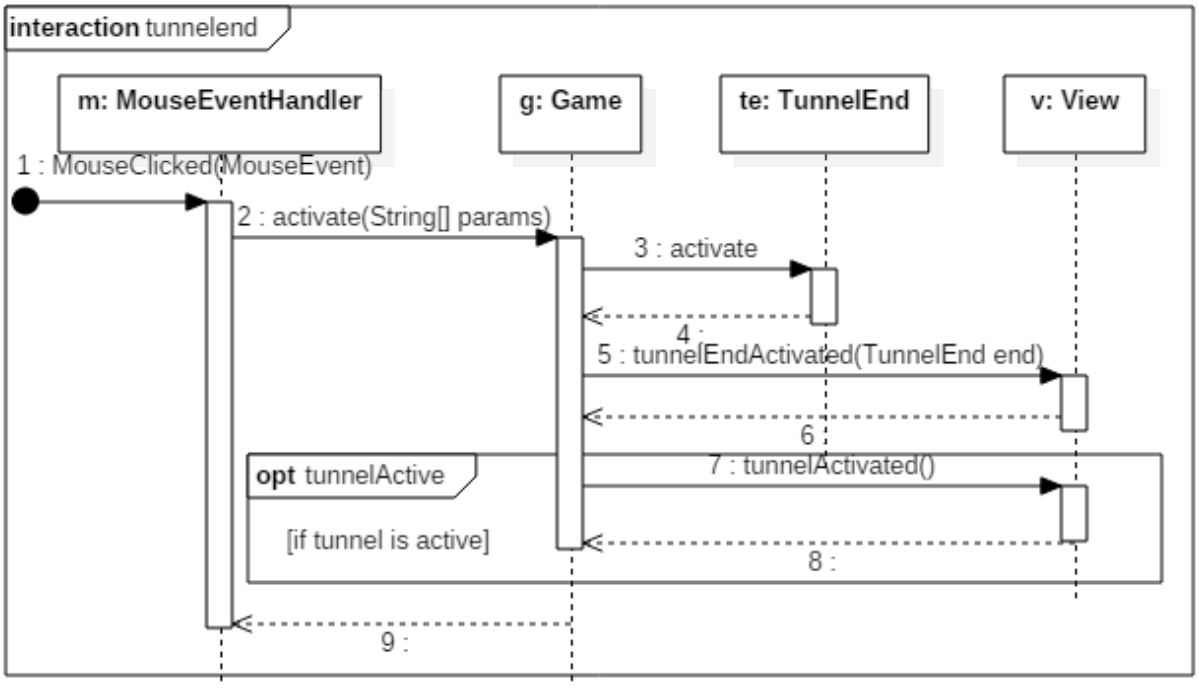
11.4. Kapcsolat az alkalmazói rendszerrel



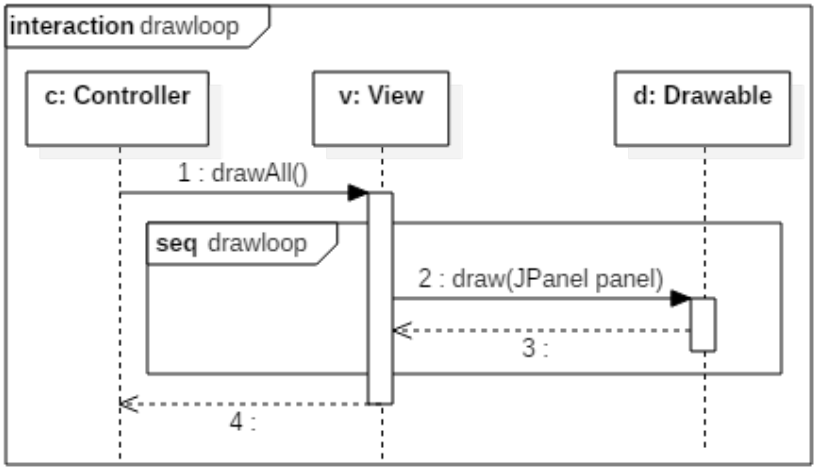
11.4. ábra. Pálya betöltése



11.5. ábra. Váltás



11.6. ábra. Alagútvég aktiválása



11.7. ábra. Rajzolás

11.5. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017.04.21. 18:00	0,5 óra	**Dombai** **Erős** **Szilágyi**	Értekezlet. Döntés: Erős elkezdi az osztálydiagramot.
2017.04.21. 19:00	1,5 óra	**Erős**	Tevékenység: Osztálydiagram elkezdése
2017.04.22. 18:00	1 óra	**Székely**	Tevékenység: Grafikus elemek készítése
2017.04.23. 21:00	1 óra	**Dombai**	Tevékenység: osztálydiagram befejezése, szekvenciák
2017.04.23. 21:00	1 óra	**Erős**	Tevékenység: 11.2, osztálydiagram átnézése
2017.04.23. 22:00	1 óra	**Erős**	Tevékenység: 11.3
2017.04.23. 22:00	1 óra	**Márton**	Tevékenység: Osztályok leírása
2017.04.23. 22:00	1 óra	**Szilágyi**	Tevékenység: Osztályok leírása
2017.04.24. 8:00	2 óra	**Székely**	Tevéken3ység: Grafikus elemek befejezése
2017.04.24. 11:00	1 óra	**Dombai**	Tevékenység: Példapálya, menü, átnézés

13. Grafikus felület specifikációja

13.0. Grafikus felület változások

- fekete vonal - sín
- piros kitöltetlen kör - vakvágány
- fekete kitöltött kör - kereszteződés
- sínen lévő színes kitöltött négyzet - állomás
- piros kitöltött kör - aktiválatlan alagútvég
- zöld kitöltött kör - aktivált alagútvég
- "CYAN" négyzet - mozdony
- Feketen égyzet - szeneskocsi
- egyéb színes négyzet - kocsi
- Széles szürke téglalap - alagút
- kereszteződésben narancs téglalap - váltó fix vége
- kereszteződésében zöld téglalap - váltó aktív, változó vége

13.1. Fordítási és futtatási útmutató

13.1.1. Fájllista

Controller package:

Fájl neve	Méret	Keletkezés ideje	Tartalom
Application.java	1 KB	2017.05.07 14:00	Alkalmazás
Controller.java	3 KB	2017.05.07 14:00	Vezérlő
Application.java	2 KB	2017.05.07 14:00	Kezdőablak

Model package:

Fájl neve	Méret	Keletkezés ideje	Tartalom
CoalWagon.java	3 KB	2017.05.07 14:00	Szeneskocsi
Collection.java	6 KB	2017.05.07 14:00	Pályaelemeken található vonatelemek kollekciója

Fájl neve	Méret	Keletkezés ideje	Tartalom
Component.java	6 KB	2017.05.07 14:00	Pályaelemek absztrakt ősosztálya
Game.java	12 KB	2017.05.07 14:00	Fő játék osztály
Intersection.java	2 KB	2017.05.07 14:00	Kereszteződés
Locomotive.java	4 KB	2017.05.07 14:00	Mozdony
Rail.java	2 KB	2017.05.07 14:00	Sín
Siding.java	2 KB	2017.05.07 14:00	Vakvágány
Station.java	2 KB	2017.05.07 14:00	Állomás
Switch.java	3 KB	2017.05.07 14:00	Váltó
TrainComponent.java	4 KB	2017.05.07 14:00	Vonatot alkotó elemek absztrakt ősosztálya
Tunnel.java	3 KB	2017.05.07 14:00	Alagút
TunnelEnd.java	2 KB	2017.05.07 14:00	Alagútvég
Wagon.java	4 KB	2017.05.07 14:00	Vagon

View package:

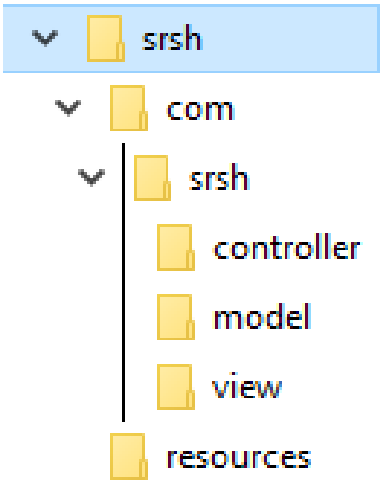
Fájl neve	Méret	Keletkezés ideje	Tartalom
Drawable.java	2 KB	2017.05.07 14:00	Rajzolható elemek absztrakt ősosztálya
DrawableComponent.java	2 KB	2017.05.07 14:00	Rajzolható pályát alkotó elemek absztrakt ősosztálya
DrawableTrainComponent.java	1 KB	2017.05.07 14:00	Rajzolható vonatot alkotó elemek absztrakt ősosztálya
GraphicCoalWagon.java	1 KB	2017.05.07 14:00	Szeneskocsi
GraphicIntersection.java	1 KB	2017.05.07 14:00	Kereszteződés
GraphicLocomotive.java	2 KB	2017.05.07 14:00	Mozdony
GraphicRail.java	1 KB	2017.05.07 14:00	Sín
GraphicSiding.java	1 KB	2017.05.07 14:00	Vakvágány
GraphicStation.java	2 KB	2017.05.07 14:00	Állomás
GraphicSwitch.java	3 KB	2017.05.07 14:00	Váltó
GraphicTunnel.java	1 KB	2017.05.07 14:00	Alagút
GraphicTunnelEnd.java	1 KB	2017.05.07 14:00	Alagútvég

Fájl neve	Méret	Keletkezés ideje	Tartalom
GraphicWagon.java	1 KB	2017.05.07 14:00	Kocsi
View.java	6 KB	2017.05.07 14:00	Játék ablak, View

Resources mappa:

Fájl neve	Méret	Keletkezés ideje	Tartalom
palya1.txt	1 KB	2017.05.07 14:00	Első pálya
palya2.txt	1 KB	2017.05.07 14:00	Második pálya
palya3.txt	1 KB	2017.05.07 14:00	Harmadik pálya

13.1.2. Fordítás



13.1. ábra. Fordításhoz és futtatáshoz szükséges mappaszerkezet

com mappa felett állva:
JDK telepítési útvonala ettől eltérhet, ebben az esetben azzal útvonallal kell beállítani.

```
set PATH="%PATH%;C:\Program_Files\Java\jdk1.6.0_18\bin"
chcp 65001
javac -encoding UTF-8 com/srsh/controller/*.java com/srsh/model/*.java
     com/srsh/view/*.java
```

13.1.3. Futtatás

```
java -Dfile.encoding=UTF-8 com.srsh.controller.Application
```

13.2. Értékelés

Tag	**Munka százalékban**	**Aláírás**
Dombai	33.8 %	
Erős	19.0 %	
Márton	16.9 %	
Székely	14.8 %	
Szilágyi	15.4 %	

13.3. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017.05.05. 10:00	1 óra	**Erős**	View osztályok létrehozása, azok elkezdése
2017.05.05. 14:30	3 óra	**Dombai**	Controller package, View minimálisan
2017.05.05. 14:30	3,5 óra	**Erős**	View package
2017.05.05. 16:00	1 óra	**Szilágyi**	Collection osztály javítások
2017.05.06. 18:00	2 óra	**Dombai**	Draw metódusok átalakítása, ehhez szükséges metódusok Modell-ben, alagút javítás
2017.05.06. 21:00	2 óra	**Szilágyi**	Place
2017.05.06. 21:00	4 óra	**Márton**	Grafika, magic
2017.05.06. 23:00	2 óra	**Dombai**	Minden.
2017.05.07. 10:00	3 óra	**Dombai**	Grafikus javítás, tolatás patch, utasszám megjelenítés, pályák szerkesztése, így előjövő bugok javítása
2017.05.07. 20:00	2 óra	**Dombai**	z index, váltón kisiklás patch, clear fgv-ek, felugró ablakok, fordítási ill. futtatási útmutató, fájllista
2017.05.07. 23:00	1 óra	**Székely**	Pálya
2017.05.08. 10:00	1,5 óra	**Székely**	Kommentezés
2017.05.08. 10:00	1,5 óra	**Dombai**	Kommentezés, napló, doksi véglegesítés

14. Összefoglalás

14.1. Projekt összegzés

Tag	Munkaidő (óra)
Dombai	56
Erős	31.5
Márton	28
Székely	24.5
Szilágyi	25.5
Összesen:	**165.5**

Fázis	Forrássor
Szkeleton	748
Prototípus	1554
Grafikus változat	2846
Összesen	5148

14.2. Dombai

- Mit tanultak a projektből konkrétan és általában?
  Végre valóban volt értelme verziókezelést használni, ShareLatex használatával nagyjából megtanulhattuk a LaTeXhasználatát. Korrekt bevezetés volt a csapatmunka világába, és egyértelműen kiderült hogy az a valóságban nem úgy működik "ahogy az a nagykönyvben meg van írva".

- Mi volt a legnehezebb és a legkönnyebb?
  Szkeleton elkeszítése, azon belül is az analízis modellek voltak a legnehezebbek, az egy-egy hetes határidő szűk ahhoz hogy jól átgondolt, kellő részletességgel elkészített terveket lehessen készíteni, illetve azokat diagramokkal tudjuk ábrázolni. Egyúttal véleményem szerint a kapott "hint"-ek nem részletesek, nincs mindenhol egyértelműen definiálva hogy mi a követelmény (különösen első pár doksinál). Az elvárt objektumorientált tervezési szintet tisztázni kellene már a projekt legelején, hogy ne később hulljanak ki csontvázak a szekrényből.
  A grafikus változat elkészítése volt a legkönnyebb, hiszen az ahhoz szükséges kód jelentős része már rendelkezésre állt, minimális plusz tervezést igényelt.

- Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?
  Igen.
- Ha nem, akkor hol okozott ez nehézséget?
  -
- Milyen változtatási javaslatuk van?
  Bővebb tájékoztatás az elvárásokról, akár a doksikban lévő hint-ek bővítésével.
- Milyen feladatot ajánlanának a projektre?
  Pac-Man szerű játék procedurálisan generált térképekkel, extrákkal meghintve.

14.3. Erős

- Mit tanultak a projektből konkrétan és általában?
  Meganultuk használni a LaTex-et a Git-et és nyújtott egy átfogó képet arról, hogy milyen is csapatban dolgozni. Minden nehézségével együtt egy jó tapasztalatot adott a csapatmunkával kapcsolatban.
- Mi volt a legnehezebb és a legkönnyebb?
  A legnehezebb összehangolni azt, hogy ki mikor ér rá, és összehozni a meetingeket. Emellett nehéz volt az analízis modell kidolgozása is. Ezen felül a sceleton sem úgy sikerült teljesen, ahogy később megvalósult, de összességében elégedettek lehetünk. A legkönnyebb talán a programozás része volt a dolognak.
- Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?
  A nagy egészt nézve igen. Voltak részek, ahol talán több idő kellett volna, volt ahol kevesebb is elegendő lenne, de összességében jó volt így.
- Ha nem, akkor hol okozott ez nehézséget?
  Nem okozott.
- Milyen változtatási javaslatuk van?
  Hercules fejlesztése, esetleges lecserélése. Ezen felül a Dokumentációt túl részletesnek érzem, nem feltétlen kéne ennyire részletesen megtervezni, hisz a végső kód az esetek többségében nem áll 100%-os átfedésben a tervekkel.
- Milyen feladatot ajánlanának a projektre?
  Raktár kezelő szoftver megvalósítása, akár grafikus alapokon.

14.4. Márton

- Mit tanultak a projektből konkrétan és általában?
  Megtanultuk használni az RUP fejlesztési módszertant, és a verziókezelést is. Megtapasztalhattuk hogy milyen csapatban dolgozni és hogyan kell összeegyeztetni egymás munkáját hogy végül működő szoftvert kapjunk, továbbá megtanultuk alkalmazni a szoftvermodellezési technikákat.
- Mi volt a legnehezebb és a legkönnyebb?
  A legnehezebb az volt, hogy megfelelően összehangoljuk a csapatmunkát. Tekintve hogy nagyon sok rész összefüggött, sok időt kellett belefektetni abba hogy megfelelően felosszuk egymás között a munkát, illetve sokszor csak azután lehetett egy adott részt elkezdeni, miután valaki befejezte azt, amire épül. Miután megfelelően összehangoltuk a fejlesztést, és elkészítettük a modelleket, már könnyen ment a feladat implementálása (kódolás).
- Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?
  Összességében igen.

- Ha nem, akkor hol okozott ez nehézséget?
  Nem okozott.

- Milyen változtatási javaslatuk van?
  A beadandó sablonban megfogalmazott (és kék színnel jelölt) követelmények sokszor nem voltak egyértelműek, pontosak, így meg volt az esélye annak hogy rosszul oldunk meg egy-egy részt, mivel az általunk logikus értelmezés ellentmondott az egyébként elvártnak.

- Milyen feladatot ajánlanának a projektre?
  Egy repülőtér irányítása, fel és leszálló gépek kezelése, illetve a terminálok felé közlekedő gépek megfelelő irányítása, hogy ne ütközzenek össze, az utasok fel-és leszálljanak majd a gépek ismét útjukra induljanak.

14.5. Székely

- Mit tanultak a projektből konkrétan és általában?
  Verziókezelés (Git) használatát, gyakorlati megvalósítását a szoftvertechnológiákban tanult tervezési módszereknek, határidőkhöz és egymáshoz való alkalmazkodást. Továbbá, mivel mi LaTeX-ben készítettük a dokumentumokat, és én először dolgoztam ebben, így ebben is sokat fejlődtem.

- Mi volt a legnehezebb és a legkönnyebb?
  Rájöttem, hogy mennyire nehéz egy szoftvert "megrendelésre" készíteni, ahol az ember sok esetben nem mehet a saját feje után, hanem pontosan azt kell megvalósítani amit kérnek, és úgy, ahogyan azt kérik, és nem csak az a lényeg, hogy működjön. Nehéz volt mindig következetesen dolgozni is, ezen a téren még sokat kell fejlődnünk, sajnos gyakran előfordultak az utólagos módosítások, de úgy érzem, hogy most már van egy átfogóbb képünk a tervezés egészével kapcsolatban, így előrelátóbban fogunk tudni majd dolgozni a továbbiakban. Túl könnyűnek egyik részfeladatot se nevezném, de szerintem a tesztelés okozta a legkevesebb problémát.

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

- Ha nem, akkor hol okozott ez nehézséget?
  Nem okozott nehézséget.

- Milyen változtatási javaslatuk van?
  A sablonok kommentjei alapján nem mindig egyértelmű, hogy mit is kér az adott pontban, példákkal lehetne segíteni ezen.

- Milyen feladatot ajánlanának a projektre?
  Egy város közlekedésének (lámpák, közösségi közlekedés, stb..) összehangolását lebonyolító szoftver.

14.6. Szilágyi

- Mit tanultak a projektből konkrétan és általában?
  Megtanultuk használni a verzió kezelést és tapasztalatot szereztünk, arról, hogy milyen csapatban dolgozni.

- Mi volt a legnehezebb és a legkönnyebb?
  Nehéz volt olyan időpontot találni ami mindenkinek megfelelt, valamint a skeletonnal voltak problémák és emiatt a megvalósítás sem volt egyszerű. Miután ezeket javítottuk a kódolás már nem volt nehéz.

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

- Ha nem, akkor hol okozott ez nehézséget?
  Nem okozott.
- Milyen változtatási javaslatuk van?
  Nem volt mindig egyértelmű, hogy mit kér tőlünk a feladat.
- Milyen feladatot ajánlanának a projektre?
  Egy településen a postás munkáját utánzó játékot.