

Szoftver projekt laboratórium

13 – gg_no_re

Konzulens:

Dudás Ádám

Csapattagok

Bartók Patrik István

Koncsik Norbert

Németh Bence

Pekk János Richárd

Burom Bence

CKC5JZ

DZLUZ7

EMNTY0

F6LQJ2

GNEFKT

2017

2. Követelmény, projekt, funkcionalitás

2.1 Bevezetés

2.1.1 Cél

Követelmények meghatározása, mit hogyan fogunk csinálni, részfeladatok kiosztása a csapaton belül, Use-Case diagram megalkotása.

2.1.2 Szakterület

Játék, szórakoztatás.

2.1.3 Definíciók, rövidítések

UML (Unified Modelling Language) - leíró nyelv

RUP (Rational Unified Process) - fejlesztési folyamat

HSZK - Hallgatói Számítógép Központ

JDK - Java Development Kit

.dat - fájl típus

kiértékelés - szoftveresen ellenőrizhető

bemutató - csak emberi szemmel ellenőrizhető

2.1.4 Hivatkozások

<https://www.iit.bme.hu/targyak/BMEVIIIAB02/feladat> - Feladatkiírás, projekt terv

2.1.5 Összefoglalás

A következőkben a szoftver felépítésének felületes leírása, Use-Case diagram megtervezése és részletes leírása, a szoftver iránti követelmények megfogalmazása, valamint a funkciók meghatározása lesz ismertetve.

2.2 Áttekintés

2.2.1 Általános áttekintés

A fejlesztés Java nyelven fog zajlani. A szoftver internetes hálózatra nem fog csatlakozni, semmilyen hálózati adatforgalom nem lesz. Egy darab pálya lesz a játékban, minden szint ugyan azt használja, amelyet egy .dat fájlban fogunk tárolni, melyet a program képes lesz kezelni. 3 nagyobb blokk fog megvalósulni: vonat, sín, pálya. A pálya kommunikál a vonatokkal a mozgásukkal kapcsolatban, a sínekkel a lehelyezésükkel kapcsolatban. A vonat kommunikál saját magával az ütközések kezeléséhez, és a sínekkel a mozgásuk miatt. A sín csak saját magával, tudja hogy milyen sín van mögötte és utána.

2.2.2 Funkciók

A játék egy terepasztalon játszódó vonatszimulátor. A terepasztalon az állomásokat sínek kötik össze, a vonatok csak ezeken haladhatnak. Ha egy sín elágazik, akkor ott egy váltó köti össze őket. A szerelvény mindig csak abba az irányba haladhat tovább, amerre a váltó áll.

A szerelvényeken különböző színű vagonok találhatók. Az utasok csak akkor szállhatnak le az állomáson, ha az állomás színe megegyezik a szerelvényen lévő első, nem üres vagonnak a színével. Ha minden utas leszállt a vonatról, az adott szintet teljesítettük, és a következőre lépünk. A szintek lépésével a pályák egyre nehezednek, több állomás nyílik meg, illetve több vonat kerül a terepasztalra. A vonatok a terepasztal egy dedikált helyéről indulnak, az adott szinthez meghatározott időközönként.

Miután sikeresen leszállítottunk róluk minden utast, a szerelvények a pályán nem állnak meg, így ügyelnünk kell arra, hogy ne ütközzenek össze semmivel, mert ekkor felrobbannak, és elveszítjük a játékot. Játék végét jelenti még továbbá az is, ha egy szerelvényt kivezetünk a pályáról, vagy ha egy váltót felvágunk. Utóbbi azt jelenti, hogy egy vonat egy olyan váltón halad át szemből, ami nem az ő állásba van kapcsolva. A pályán létrehozhatunk alagutakat. Egyszerre csak maximum 2 alagútszáj lehet a pályán, ezeket köti össze az alagút. Ha 1 alagútszáj van csak a pályán, azon a vonat egyszerűen átmegy rajta. Az alagút lehet rövidebb, mint a vonat hossza, ilyenkor csak az alagútban lévő része nem látszik.

Célja, hogy a vonatokon lévő utasokat a megfelelő színű állomásokra eljuttassuk.

2.2.3 Felhasználók

A felhasználónak rendelkeznie kell olyan képességgel, amivel mozgatni tudja az egeret, valamint képes látni a monitort. Minimális informatikai készség szükséges.

2.2.4 Korlátozások

A HSZK-s számítógépeknek megfelelő Javat verzióval rendelkeznie kell a számítógépnek, amin futtatni akarjuk.

2.2.5 Feltételezések, kapcsolatok

<https://www.iit.bme.hu/targyak/BMEVIIIAB02/feladat> - Feladatkiírás : Megadja a követelményeket, korlátozásokat. Projekt terv: Megadja milyen feladatokat mikorra kell elvégezni.

2.3 Követelmények

2.3.1 Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
2.3.1.1	A vonatokat mozgatni kell.	kiértékelés	alapvető	Megrendelő	train_movement	
2.3.1.2	Csak olyan vonatot lehet generálni, amilyen színű állomás nyitva van.	kiértékelés	alapvető	gg_no_re	generate_train	
2.3.1.3	A különböző állomásokat ki kell nyitni a pálya megkezdésekor.	kiértékelés	alapvető	Megrendelő	init	
2.3.1.4	Ha a vonat eléri a pálya szélét, felrobban és vége a játéknak.	kiértékelés	fontos	gg_no_re	train_movement	
2.3.1.5	Ha a vonat nekimegy egy másik vonatnak, a vonat felrobban és vége a játéknak.	kiértékelés	alapvető	gg_no_re	train_movement	
2.3.1.6	Ha a vonat nekimegy egy váltónak, ami nem az ő sínjére van váltva, a vonat felrobban és vége a játéknak.	kiértékelés	fontos	gg_no_re	train_movement	
2.3.1.7	Ha egy vonat eléri egy állomást és az első nem üres vagonjának a színe	kiértékelés	alapvető	Megrendelő	passenger_delivery	

	megegyezik az állomás színével, a vagon kiürül.					
2.3.1.8	A generált vonatok el tudjanak indulni.	kiértékel és	alapvető	Megrendelő	generate_train	
2.3.1.9	Alagút szakasz nem lehet se váltón, se állomáson.	kiértékel és	fontos	gg_no_re	place_tunel	
2.3.1.10	A váltót állítani lehet.	kiértékel és	alapvető	Megrendelő	use_switch	
2.3.1.11	A váltó ne állítódjon át, ha vonat van rajta.	kiértékel és	fontos	gg_no_re	use_switch	
2.3.1.12	A vonat a váltó állásának megfelelő irányba halad tovább.	kiértékel és	alapvető	Megrendelő	train_movement	
2.3.1.13	Ha bármelyik vonat is felrobban, a játéknak vége.	kiértékel és	alapvető	gg_no_re	simulation	
2.3.1.14	Ha minden vonat kiürült, a játékos megnyerte a szintet.	kiértékel és	alapvető	gg_no_re	simulation	

2.3.2 Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
2.3.2.1	Java Development Kit	kiértékelés	alapvető	Megrendelő	
2.3.2.2	egér	bemutató	alapvető	Megrendelő	
2.3.2.3	megjelenítő	bemutató	fontos	gg_no_re	

2.3.3 Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
2.3.3.1	Java verzió, ami kompatibilis a HSZK-s Java verzióval..	kiértékelés	alapvető	Megrendelő	

2.3.4 Egyéb nem funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
2.3.4.1	Hordozható	kiértékelés	alapvető	Megrendelő	Olyan gépen is lefut a fordított program, ahol nem fejlesztették
2.3.4.2	Felhasználói előélet	bemutató	fontos	gg_no_re	A felhasználó tudja, hogy mire való ez a program és arra akarja használni
2.3.4.3	Megbízható	kiértékelés	alapvető	gg_no_re	Egy gépen, amin van előírt java verzió, elfut a szoftver.

2.4 Lényeges use-case-ek

2.4.1 Use-case leírások

Use-case neve	init
Rövid leírás	Felépíti az adott pályát.
Aktorok	User
Forgatókönyv	Felépíti az adott pályát. Megnyitja az állomásokat.

Use-case neve	place_tunnel
Rövid leírás	A felhasználó leteheti az alagutat.
Aktorok	User
Forgatókönyv	Egyesével rakhatja le a bejáratot és a kijáratot. Ha mindkettő lerakásra került, kirajzolódik az alagút közé.

Use-case neve	use_switch
Rövid leírás	A felhasználó átválthatja a váltókat.
Aktorok	User
Forgatókönyv	A váltók 2 állásúak, kattintásra lehet változtatni a állásukat.

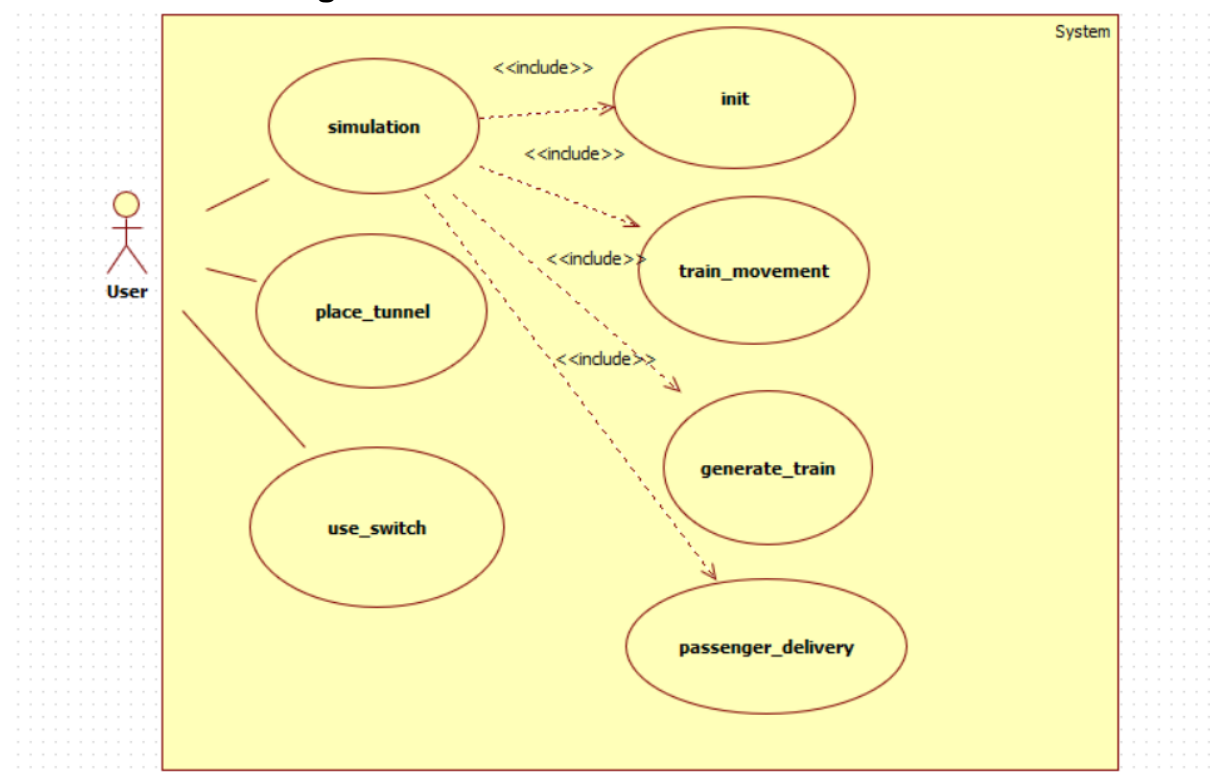
Use-case neve	simulation
Rövid leírás	A játék folyamatának szimulálása.
Aktorok	User
Forgatókönyv	A pálya működése a következő három include-olt használati esetben valósul meg: train_movement, generate_train, passenger_delivery

Use-case neve	train_movement
Rövid leírás	A vonatok mozgatása.
Aktorok	User
Forgatókönyv	A vonatok következő állapotba mozgatása, a sín típusának függvényében.

Use-case neve	generate_train
Rövid leírás	A vonatok generálása.
Aktorok	User
Forgatókönyv	Új vonatok generálása, majd indítása a nyitva lévő állomások függvényében (csak olyan színű vonatok indulnak, amilyen színű állomás nyitva van).

Use-case neve	passenger_delivery
Rövid leírás	Az utasok leszállítása.
Aktorok	User
Forgatókönyv	Utasok leszállítsa, ha lehetséges. Akkor lehetséges, ha az állomás színe megegyezik, a vonat első nem üres vagonjának színével.

2.4.2 Use-case diagram



2.5 Szótár

include	tartalmaz
Java Development Kit	Java fejlesztőkörnyezet
modell	a valóság leképezése
szoftver	számítógépen futtatható egység
UML (Unified Modelling Language)	szabványos, általános célú modellező nyelv
RUP (Rational Unified Process)	szoftverfejlesztési folyamat
Google Docs	közösen szerkeszthető szövegfájl
GIT	verziókezelő szoftver

2.6 Projekt terv

Az alkalmazást objektum orientáltan készítjük Java-ban megvalósítva, UML leírással, RUP processz szerint. A dokumentumokat közösen szerkeszthető Google Docs-ban készítjük el. A forráskódok megosztásához GIT rendszert fogunk használni.

A további részfeladatok kiosztása később fog megtörténni.

Feladat	Felelős személy	Dátum
Követelmény	Pekk	2017.02.20.
Projekt terv	Bartók	2017.02.20.
Funkcionalitás	Németh	2017.02.20.
Use-casek	Koncsik	2017.02.20.
Use-case diagram	Burom	2017.02.20.
Analízis modell kidolgozása	Bartók, Burom, Koncsik, Németh, Pekk	2017.02.27.
Szkeleton tervezése	Bartók, Burom, Koncsik, Németh, Pekk	2017.03.13.
Skeleton	Bartók, Burom, Koncsik, Németh, Pekk	2017.03.20.
Prototípus koncepciója	Bartók, Burom, Koncsik, Németh, Pekk	2017.03.27.
Részletes tervek	Bartók, Burom, Koncsik, Németh, Pekk	2017.04.03.
Prototípus készítése, tesztelése	Bartók, Burom, Koncsik, Németh, Pekk	2017.04.10.
Prototípus	Bartók, Burom, Koncsik, Németh, Pekk	2017.04.17.
Grafikus felület specifikációja	Bartók, Burom, Koncsik, Németh, Pekk	2017.04.24.
Grafikus változat készítése	Bartók, Burom, Koncsik, Németh, Pekk	2017.05.01.
Grafikus változat	Bartók, Burom, Koncsik, Németh, Pekk	2017.05.08.
Összefoglalás	Bartók, Burom, Koncsik, Németh, Pekk	2017.05.10.

2.7 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.02.18. 19:00	30 perc	Bartók Koncsik Németh Pekk Burom	Értekezlet. Feladat értelmezése, részfeladatok kiosztása
2017.02.18. 19:30	2 óra	Bartók	Átadás és egyéb követelmények
2017.02.18. 19:30	2 óra	Koncsik	Use-case- ek, diagram
2017.02.18. 19:30	2 óra	Németh	Funkciók, funkcionális követelmények
2017.02.18. 19:30	2 óra	Pekk	Bevezetés, áttekintés, szótár
2017.02.18. 19:30	2 óra	Burom	Bevezetés, erőforrások, projekt terv

3. Analízis modell kidolgozása

3.1 Objektum katalógus

3.1.1 Color

A Color enum tárolja a színeket.

3.1.2 Direction

A Direction enum tárolja, hogy az egyes vonatok milyen irányba közlekednek.

3.1.3 Rail

A Rail osztály példányai a sínek. A vonatok ezeken tudnak közlekedni. Speciális esetei a váltó, az állomás, és a bemenet. A váltón lehet a vonatot egy másik irányba tovább vezetni, az állomáson történik majd az utasok leszállítása, a bemeneti sínről pedig indulnak majd a szerelvények a pályára.

3.1.4 Stage

A Stage osztály példánya foglalja magába a játék logikáját. Itt történik meg a pálya inicializálása, a vonatok generálása, itt hívódnak meg a játék működéséhez szükséges függvények.

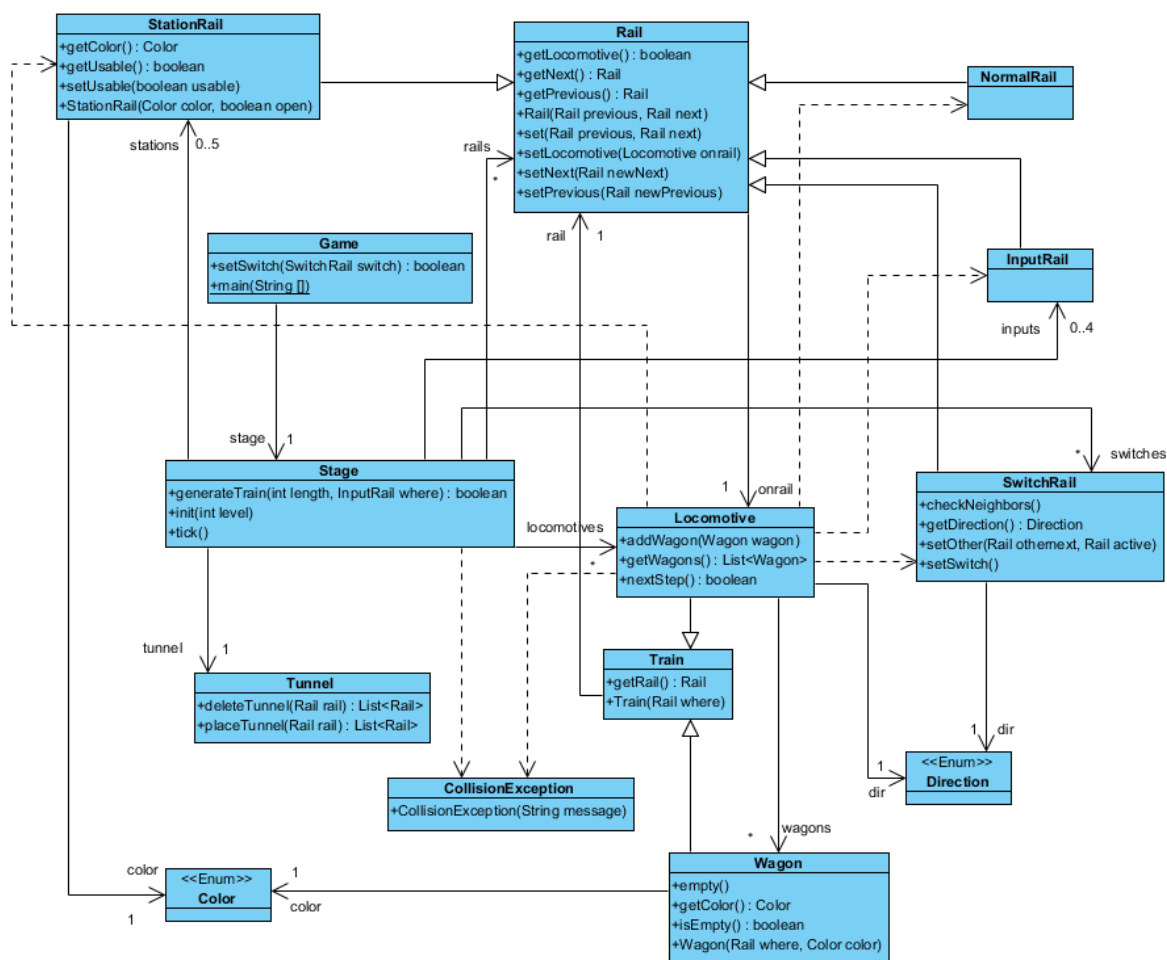
3.1.5 Train

A Train osztály példányai valósítják meg a sínen járó eszközöket. Ez lehet a Locomotive, vagy a Wagon. A Locomotive-ok fogják tárolni egy listán a rájuk csatolt vagonokat.

3.1.6 Tunnel

A Tunnel osztály példányai az alagút elemei. Egy pályán egyszerre csak két bejárata lehet, amit összeköt az alagút.

3.2 Statikus struktúra diagramok



3.1. ábra. Class diagram

3.3 Osztályok leírása

3.3.1 CollisionException

- **Felelősség**

Saját kivétel osztály. Akkor dobódik, ha egy vonat ütközik valamivel: váltóval, másik vonattal, a pálya szélével.

- **Ősosztályok**

Exception

- **Metódusok**

- **CollisionException(String message):** Létrehoz egy CollisionException osztályt a paraméterben kapott üzenettel.

3.3.2 Color

- **Felelősség**

Az állomások, vagonok lehetséges színei.

- **Attribútumok**

- **RED**
- **GREEN**
- **BLUE**
- **YELLOW**
- **ORANGE**

3.3.3 Direction

- **Felelősség**

A vonatok lehetséges mozgási irányait tárolja. Egy mozdony SwitchRailenként kap új Directiont.

- **Attribútumok**

- **NEXT:** A sínek next-jére kell lépjen. (A váltó főágán, vagy az előre kapcsolt mellékágon lépked.)
- **PREVIOUS:** Ha váltón áll, akkor még egyszer a next-re kell lépni, de az utána következő sínek previous-án folytatja. (A visszakapcsolt mellékágon lépked.)
- **PREPRE:** A sínek previous-ára kell lépjen. (A váltó főágán visszafele lépked.)

3.3.4 Game

- **Felelősség**

A játék futtatásáért felel. Itt lehet a felhasználónak inputot adnia a játék számára: Switch állítása, alagút manipulálása.

- **Metódusok**

- **static void main(String[] args):** A játék futtatása.

3.3.5 InputRail

- **Felelősség**

Bemeneti sín osztály. Ennek a példányára fog kerülni a mozdony mielőtt belépne a pályára. Ha a már pályán lévő vonatok erre a sínre lépnek, felrobbannak.

- **Ősosztályok**

Rail

3.3.6 Locomotive

- **Felelősség**

Mozdony osztály.

- **Ősosztályok**

Train

- **Attribútumok**

- **List<Wagon> wagons:** A mozdonyhoz tartozó vagonokat tárolja, olyan sorrendben, hogy a kisebb indexű vagon közelebb van a mozdonyhoz.
- **Direction dir:** A mozgás irányát tárolja.

- **Metódusok**

- **Locomotive(InputRail where):** Létrehoz egy Locomotive objektumot és a paraméterként kapott bemeneti sínre rakja, majd beállítja a mozgási irányát.
- **void addWagon(Wagon wagon):** Felfűzi a paraméterként kapott vagonat a wagons lista végére.
- **List<Wagon> getWagons():** Visszaadja a mozdonyra felcsatolt vagonok listáját.
- **boolean nextStep():** A teljes szerelvény léptetését végzi, valamint ütközést érzékel, leszállítja az utasokat ha lehetséges.

3.3.7 NormalRail

- **Felelősség**

Normál sín osztály. Alagutat csak ilyen sín fölé lehet helyezni.

- **Ősosztályok**

Rail

3.3.8 Rail

- **Felelősség**

A sín osztály. Ezen fognak közlekedni a vonatok.

- **Attribútumok**

- **Rail previous:** Tárolja, hogy melyik sínbe csatlakozik.
- **Rail next:** Tárolja, hogy melyik sín csatlakozik bele.
- **Locomotive onrail:** Tárolja, hogy melyik vonat áll a sínen.

- **Metódusok**

- **Rail getPrevious():** Visszaadja, hogy melyik sínbe csatlakozik.
- **Rail getNext():** Visszaadja, hogy melyik sín csatlakozik bele.
- **Rail set(Rail previous, Rail next):** Beállítja a csatlakozásokat.
- **Rail setPrevious(Rail newprevious):** Csak az előzőt állítja.
- **Rail setNext(Rail newnext):** Csak a következőt állítja.
- **boolean getLocomotive():** Visszaadja, hogy van-e vonat az adott sínen.
- **void setLocomotive(Locomotive onrail):** Rálépteti a vonatot a sínre.
 - Erre azért van szükség, mert így könnyebb detektálni, hogy a váltónál melyik irányból jött a vonat.

3.3.9 Stage

- **Felelősség**

Az adott pályát és szintet jellemzi. Legenerálja a pályát, majd a játék folyamatától függően növeli a szintet a maximumig, utána tartja a max szintet. Szimulálja a folyamatokat.

- **Attribútumok**

- **List<Locomotive> locomotives:**
- **List<StationRail> stations:**
- **List<Rail> rails:**
- **List<SwitchRail> switches:**
- **List<InputRail> inputs:**
- **Tunnel tunnel:**

- **Metódusok**

- **boolean generateTrain(int length, InputRail where):** Generál egy vonatot, csak olyan színű vagonokkal, amilyen állomások aktívak, és lerakja a paraméterként kapott bemeneti sínre.
- **void init(int level):** Létrehozza, felépíti a pályát.
- **boolean tick():** Következő időpillanatba lépés: vonatok léptetése, switchek felmérése.
- **boolean setSwitch(int number):** A switches listában lévő sorrend szerinti paraméterben kapott számú váltót váltja.

3.3.10 StationRail

- **Felelősség**

Állomás osztály. Ha az első nem üres vagon színe megegyezik az állomás színével, az adott vagonról az utasok itt leszállnak.

- **Ősosztályok**

Rail

- **Attribútumok**

- **Color color:** Tárolja az állomások színét.
- **boolean open:** Tárolja, hogy aktív-e az állomás.

- **Metódusok**

- **StationRail (String color, boolean open):** Meghívja az ősosztály konstruktorát, majd beállítja az állomás színét, és hogy nyitva van-e.
- **boolean getUsable():** Visszaadja, hogy aktív-e az állomás.
- **void setUsable(boolean usable):** Aktiválja/deaktiválja az állomást.
- **Color getColor():** Visszaadja az állomás színét.

3.3.11 SwitchRail

- **Felelősség**

Váltó osztály. Itt történik meg a váltás, illetve tudja, hogy az adott vonatok melyik irányból jöttek, és hogy melyik irányba közlekednek tovább.

- **Ősosztályok**

Rail

- **Attribútumok**

- **Rail active:** Tárolja, hogy melyik irányba áll a váltó.
- **Rail othernext:** A váltó másik iránya.
- **Direction dir:** Az irány amerre a vonat tovább halad.

- **Metódusok**

- **void setOther(Rail othernext, Rail active):** Beállítja a váltó másik állásának megfelelő sánt, illetve hogy melyik az aktív pillanatnyilag.
- **boolean setSwitch():** Megvizsgálja, hogy van-e rajta vonat. Ha van rajta, akkor nem lehet váltani, ha nincs rajta, akkor átváltja a másik állásba.
- **Direction getDirection():** Visszaadja a mozgási irányt.
- **void checkNeighbors():** Megnézi, hogy a váltó előtti síneken tartózkodik-e vonat.
- **Rail getNext():** Visszaadja, hogy melyik váltó aktív.

3.3.12 Train

- **Felelősség**

Sínen járó osztály.

- **Attribútumok**

- **Rail rail**: Minden vonat tárolja, hogy melyik sínen tartózkodik.

- **Metódusok**

- **Train(Rail)**: Létrehoz egy Train objektumot és a paraméterként kapott sínre helyezi.
- **Rail getRail()**: Visszaadja, hogy melyik sínen áll.

3.3.13 Tunnel

- **Felelősség**

Alagút osztály. Csak normál sínek fölé helyezhető el.

- **Attribútumok**

- **NormalRail entrance**: Alagút bejáratának helye.
- **NormalRail exit**: Alagút kijáratának a helye.

- **Metódusok**

- **List<Rail> placeTunnel(Rail rail)**: Alagút létrehozása, valamint törlése. Ha a paraméterként kapott függvény már része az alagútnak, akkor kitörli. Ha nem része, és vagy a bejárat, vagy a kijárat még szabad, valamint az alagút minden pontja érvényes helyen van, létrehozza az alagutat.

3.3.14 Wagon

- **Felelősség**

Vagon osztály.

- **Össosztályok**

Train

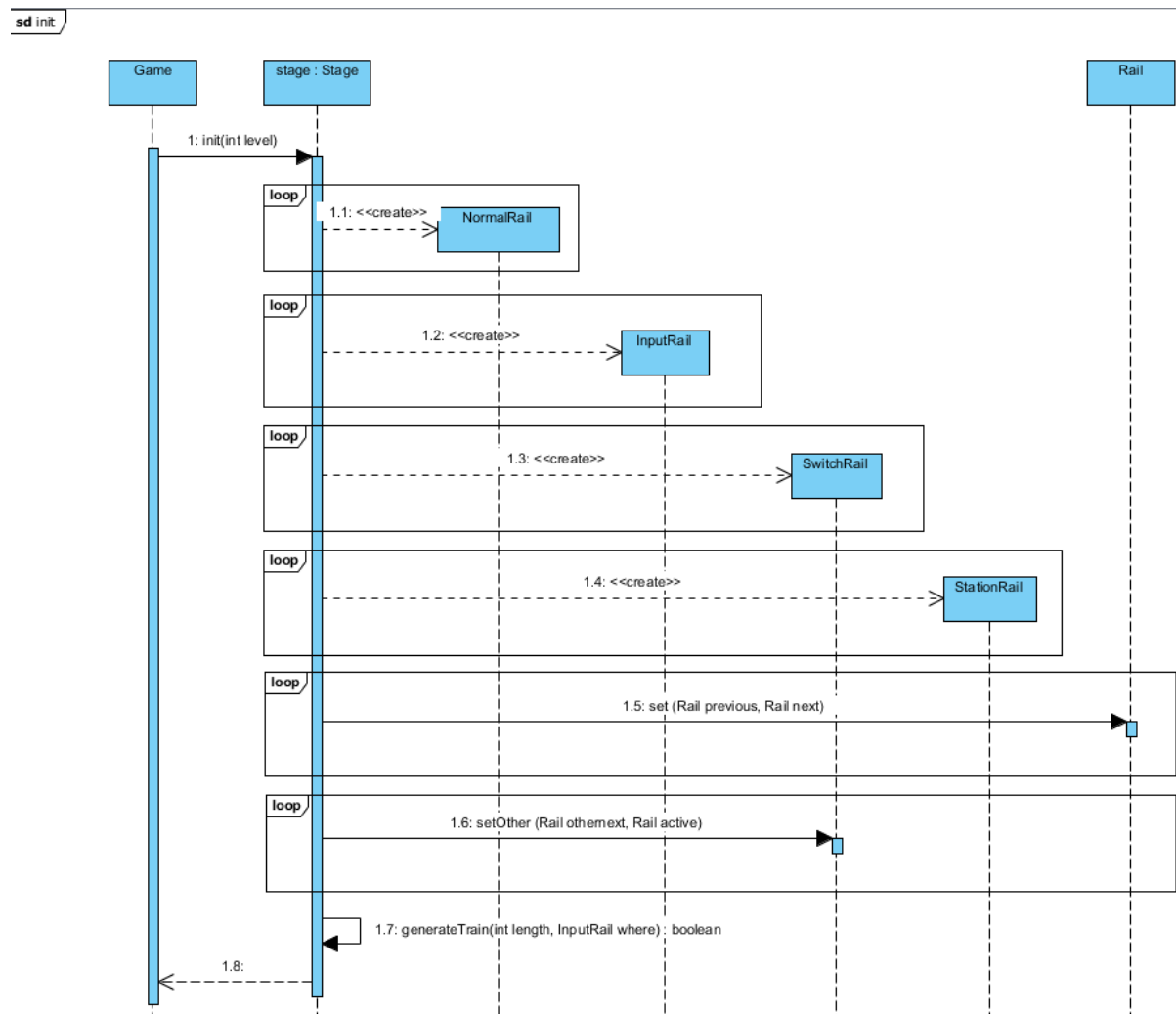
- **Attribútumok**

- **Color color**: A vagon színe.
- **boolean isEmpty**: Üres-e a vagon.

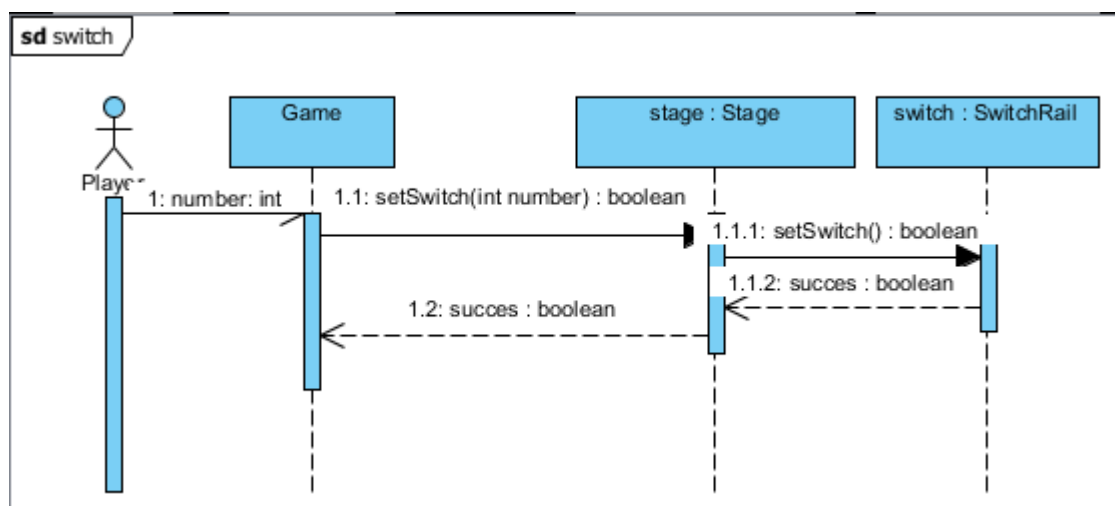
- **Metódusok**

- **Wagon(Rail where, String color)**: Konstruktor, beállítja, hogy melyik sínre legyen lehelyezve, és hogy milyen legyen a színe.
- **boolean isEmpty()**: Visszaadja, hogy üres-e a vagon.
- **String getColor()**: Visszaadja a vagon színét.
- **void empty()**: Kiüríti a vagon.

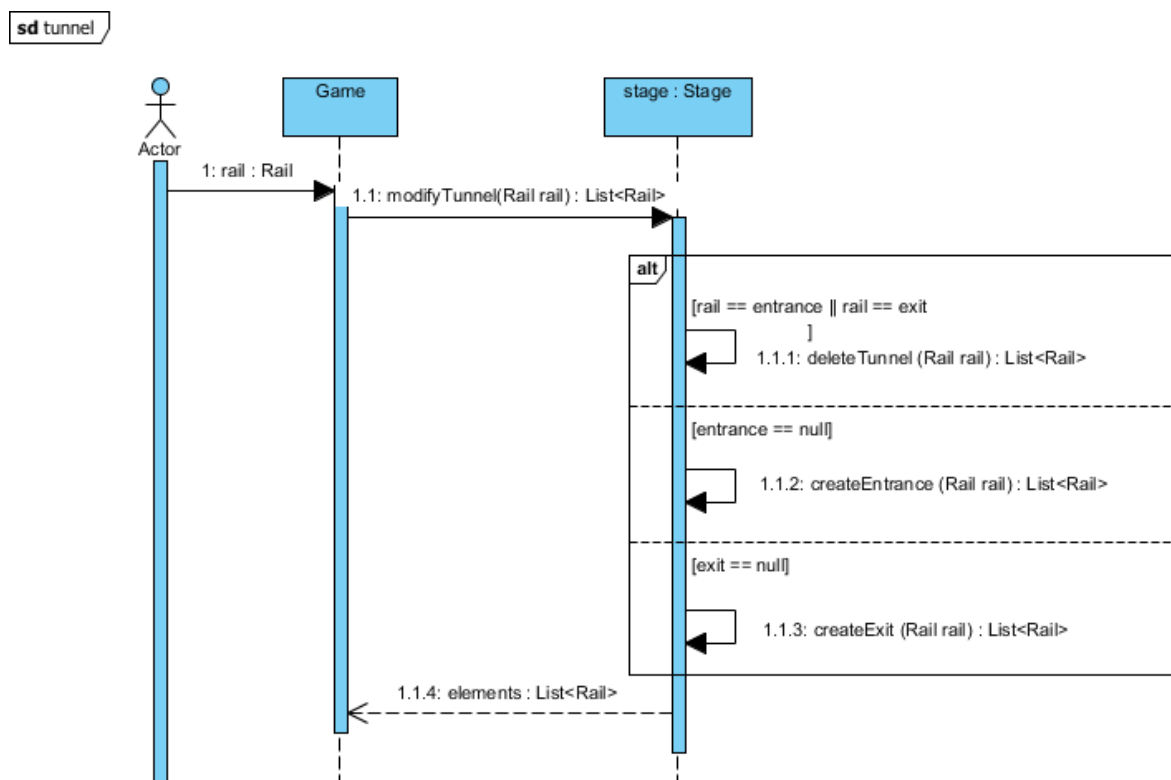
3.4 Szekvencia diagramok



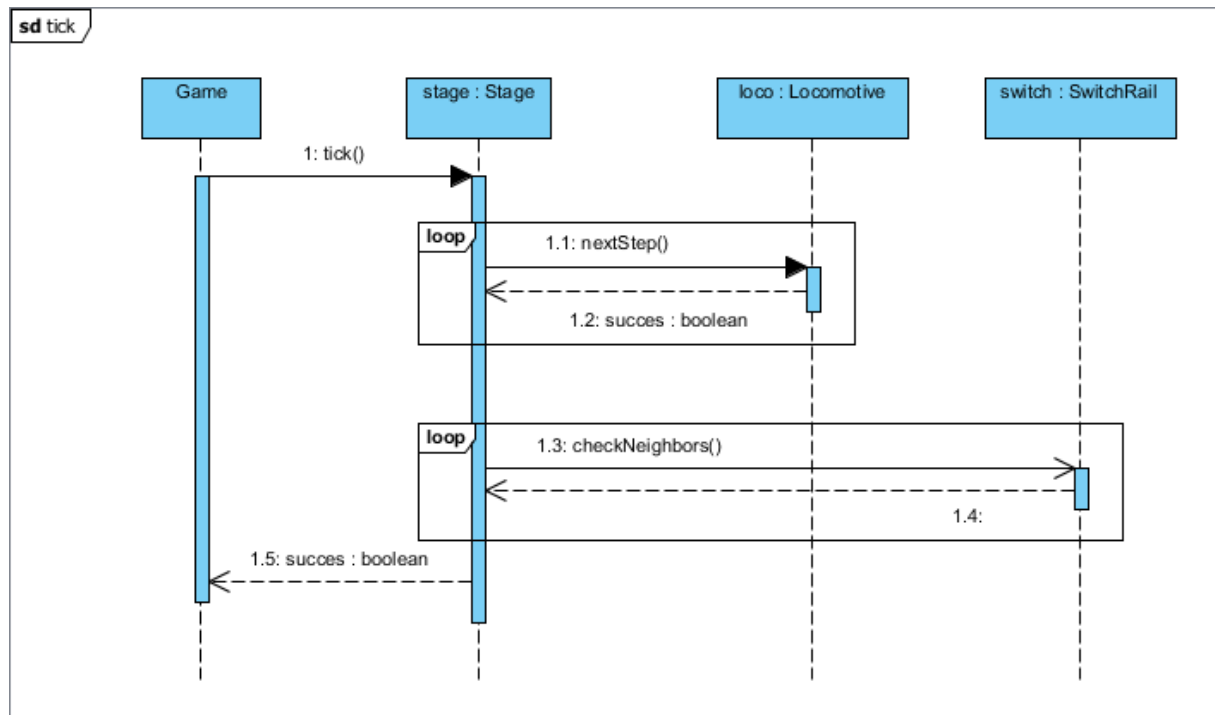
3.2 ábra. Inicializálás szekvenciadiagramja



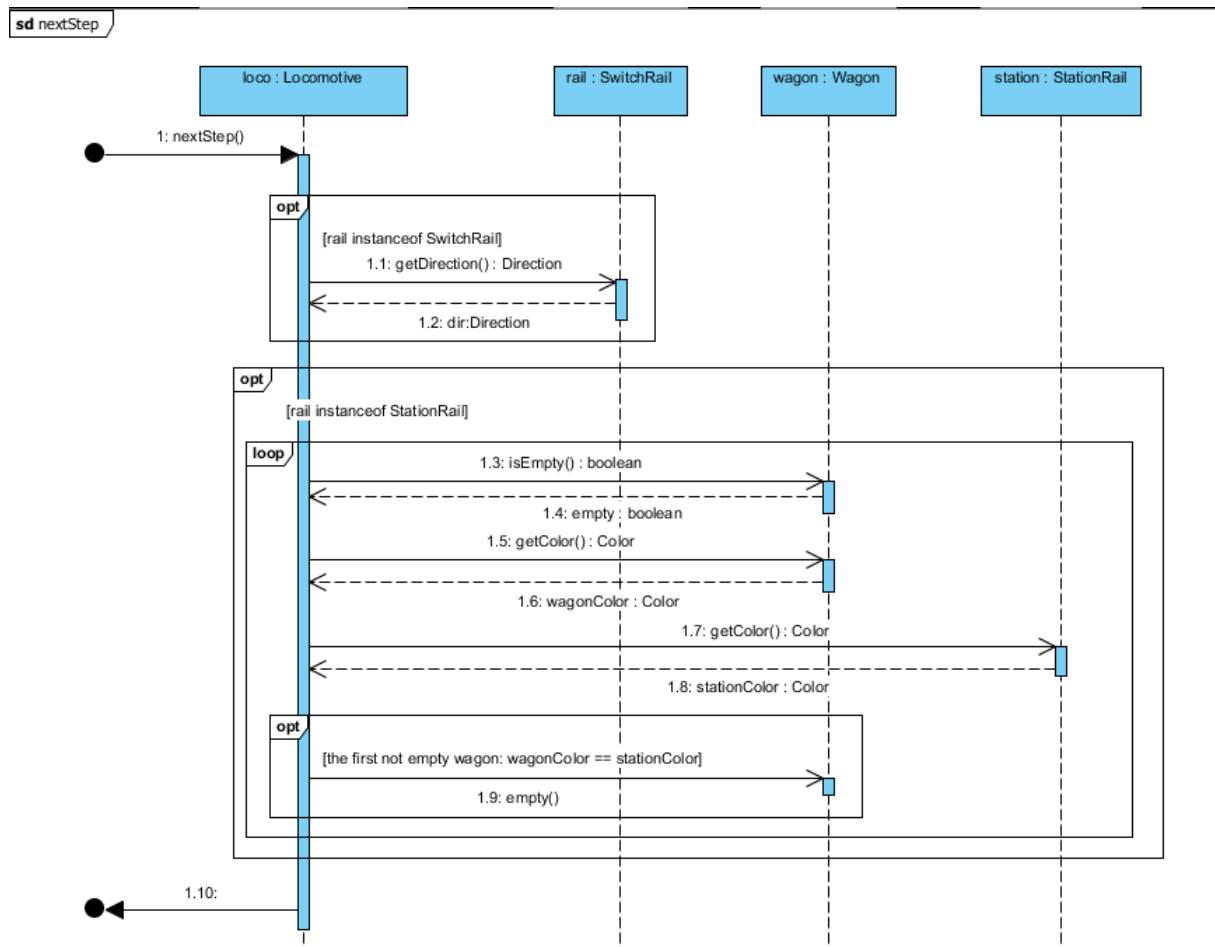
3.3 ábra. Váltás szekvenciadiagramja



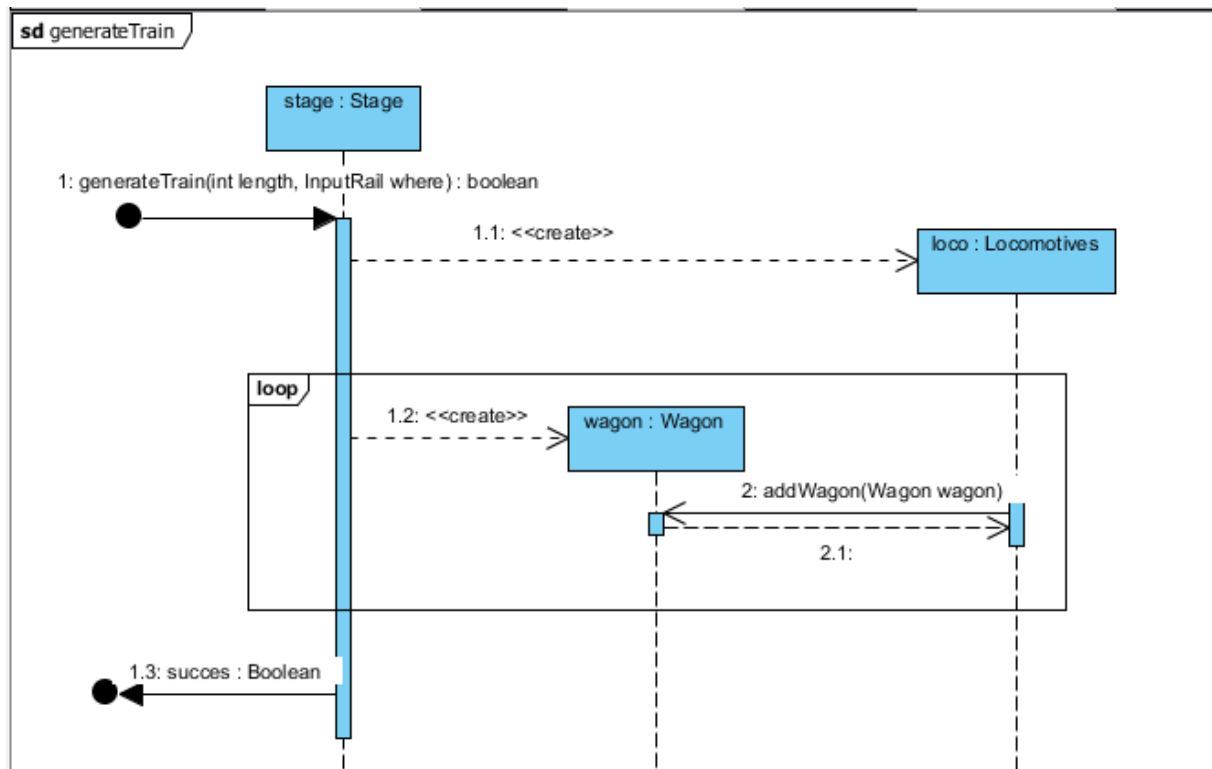
3.4. ábra. Alagút lerakás szekvenciadiagramja



3.5 ábra. Következő időpillanatba lépés szekvenciadiagramja

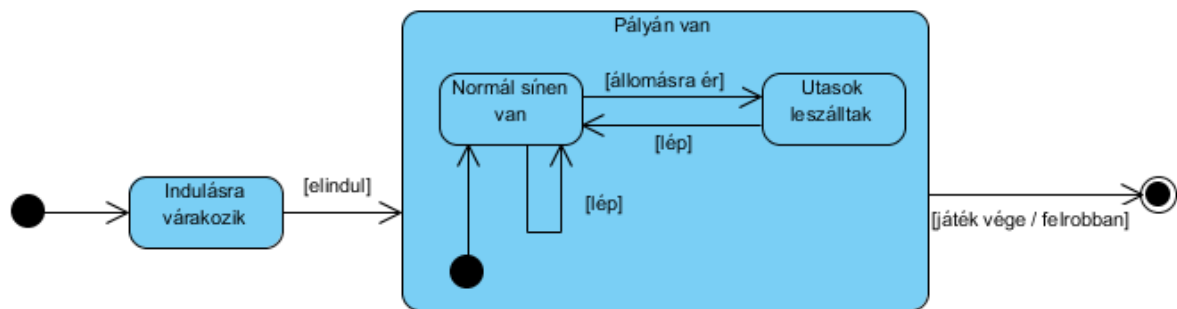


3.6 ábra. Vonatok lépésének szekvenciadiagramja

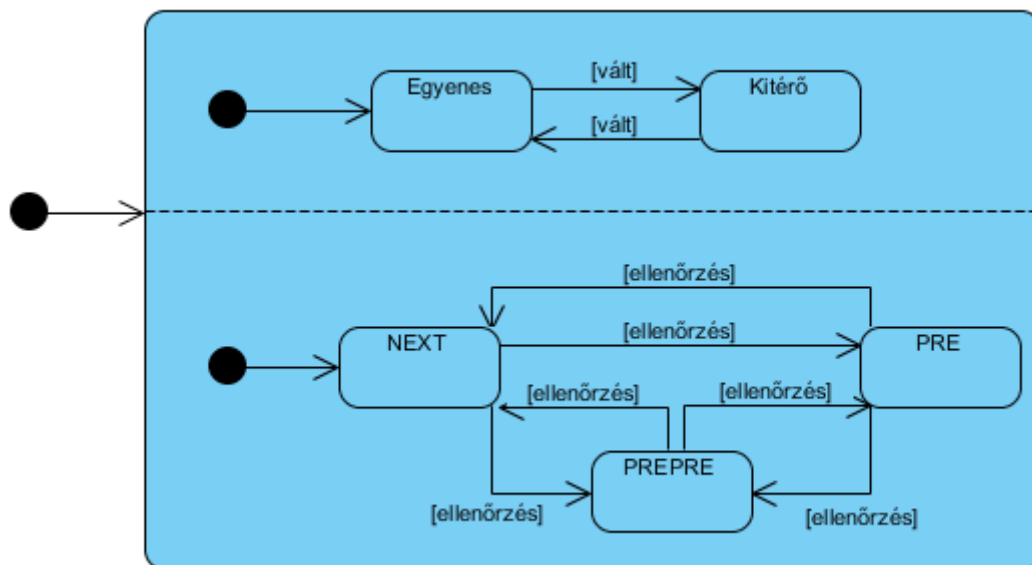


3.7. ábra. Vonatok generálásának szekvenciadiagramja

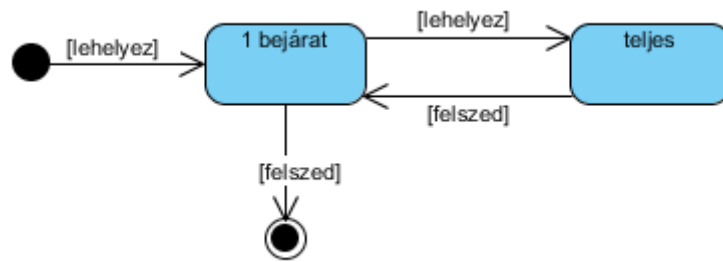
3.5 State-chartok



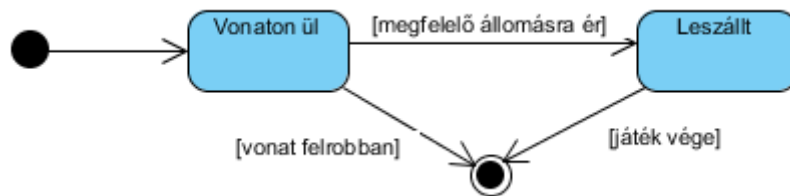
3.8. ábra. Vonat állapotdiagramja.



3.9. ábra. Váltó állapotdiagramja.



3.10. ábra. Alagút állapotdiagramja.



3.11. ábra. Utasok állapotdiagramja.

3.6 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.02.22. 09:30	1 óra	Koncsik Németh	Értekezlet. Döntés: Koncsik tervezi meg a vonathoz kapcsolódó osztályokat, Németh a sínekhez kapcsolódó osztályokat.
2017.02.22. 10:30	4 óra	Németh	Tevékenység: Rail osztályok megtervezése.
2017.02.22. 10:30	4 óra	Koncsik	Tevékenység: Train osztályok megtervezése.
2017.02.22. 19:00	1,5 óra	Németh	Tevékenység: Stage osztály megtervezése
2017.02.22. 21:30	1,5 óra	Koncsik	Tevékenység: Stage osztály folytatása
2017.02.23. 14:00	1,5 óra	Koncsik	Tevékenység: Tunnel osztály megtervezése.
2017.02.23. 14:00	2 óra	Bartók	Tevékenység: Maradék osztályok megtervezése
2017.02.24. 20:00	2 óra	Koncsik	Tevékenység: Class diagram elkészítése.
2017.02.24. 20:00	1 óra	Németh	Tevékenység: Dokumentáció szerkesztése
2017.02.26. 16:00	3 óra	Koncsik	Tevékenység: Szekvenciadiagramok elkészítése
2017.02.26. 18:00	1.5 óra	Németh	Tevékenység: State-Chart diagramok elkészítése

4. Analízis modell kidolgozása 2

Az előző beadáshoz képest változtatni kényszerültünk pár dologon.

- A mozdony léptetésénél eddig „instanceof”-al vizsgáltuk meg, hogy éppen milyen sínen tartózkodik, ez egy rossz ötlet volt. A mostani verzióban azt változtattuk meg, hogy a Rail osztály absztrakttá vált, és immáron ő végzi a mozdonyok léptetését, a moveToNext függvénnyel. Ez egy absztrakt függvény, ami a különböző leszármazott osztályokban különbözőképpen működik. NormalRail esetén csak egyszerűen a következő sínre lép. Váltónál (SwitchRail) átadja a vonatnak a megfelelő irányt (Direction), majd tovább lépteti a váltó állásának megfelelő sínre. Az állomásnál (StationRail), ha követelménynek megfelel, kiüríti a mozdony adott vagonját, majd utána lépteti a következő sínre.
- Továbbá változtatás volt még, ugyanezen az „instanceof”-os probléma megoldására, hogy az alagút lerakásnál már úgy vizsgáljuk meg a síneket, hogy az Rail osztály absztrakt canPlaceTunnel függvénye csak abban az esetben (NormalRail) ad vissza igaz értéket, ha arra a sínre helyezhető alagút. Egyébként (SwitchRail, StationRail) hamis értéket ad vissza.
- Megszüntettük az InputRail osztályt, azzal az indokkal, hogy felesleges, mert megoldható úgy, hogy választunk egy NormalRailt bemeneti sínnek, és a kiválasztott sínnek az előző sínjét null-ra állítjuk. Így a vonatok arra egyszerűen nem tudnak tovább közlekedni.
- Kikerültek az előző beadásban benne szereplő privát tagváltozók.
- Pontosítottuk néhány függvény leírását, illetve sorrendjét a classok leírásában, hogy konzisztens legyen a class diagramban szereplő sorrenddel, ezáltal könnyebben értelmezhető legyen.

4.1 Objektum katalógus

4.1.1 Rail

A sínek, amiken a vonatok (Locomotive, Wagon) közlekednek.

4.1.2 Station

Amikor egy mozdony (Locomotive) elér egy állomást (Station), a szabályok szerint kiüríti a vagon (Wagon).

4.1.3 Switch

A váltó (Switch) összeköt egy sínt (Rail) 2 másik sínrel. Van egy főág, valamint 2 mellékág. A két mellékág közül mindig csak egy aktív. Az aktív mellékágat a váltó kapcsolásának hatására változtathatjuk.

4.1.4 Locomotive

A mozdonyhoz (Locomotive) vannak kötve a vagonok (Wagon). A mozdony mozog a síneken (Rail), a mozdony felelős a hozzá tartozó vagonokért.

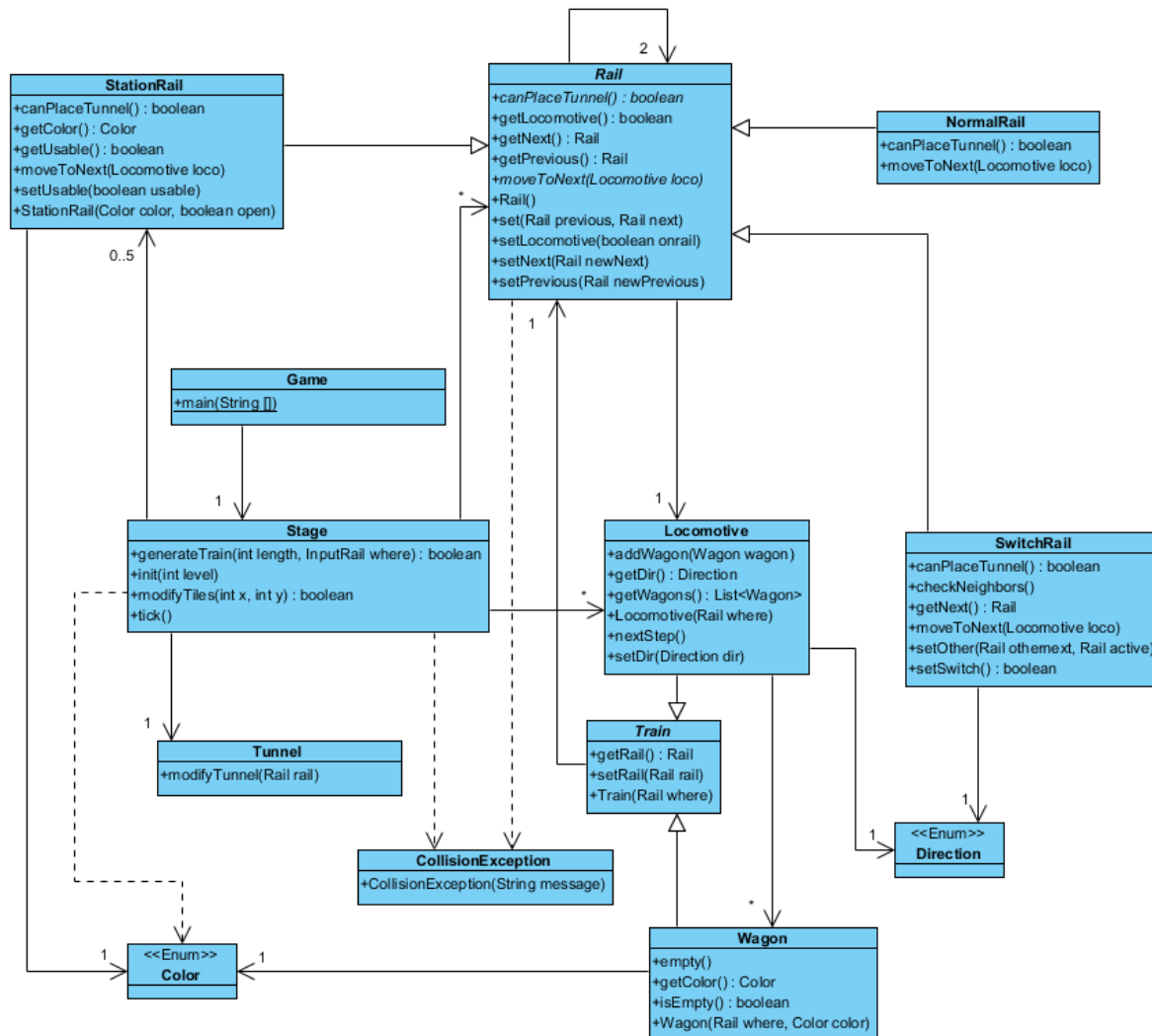
4.1.5 Wagon

Egy vagon (Wagon) egy mozdonyhoz (Locomotive) tartozik. A vagon lehet teli, vagy ha már a szabályok szerint kiürült, üres. A vagon is mozog a síneken (Rail).

4.1.6 Tunnel

A pályán lévő alagút (Tunnel). Egyszerre maximum két bejárata (tehát 1 alagút) lehet a pályán. Állomás (Station), váltó (Switch) fölé helyezni nem szabad.

4.2 Statikus struktúra diagramok



4.1. ábra
Class diagram

4.3 Osztályok leírása

4.3.1 CollisionException

- **Felelősség**

Saját kivétel osztály. Akkor dobódik, amikor egy vonat ütközik egy másik vonattal, felvág egy váltót, vagy kimegy a pályáról. Ilyen kivételt a Rail leszármazottjai dobnak. A Stageben van feldolgozva az összes ilyen kivétel.

- **Össztályok**

Exception

- **Metódusok**

- **CollisionException(String message):** Létrehozza a kivételt a paraméterben kapott stringgel.

4.3.2 Color

- **Felelősség**

Az állomások, vagonok lehetséges színe.

- **Attribútumok**

- **RED**
- **GREEN**
- **BLUE**
- **YELLOW**
- **ORANGE**

4.3.3 Direction

- **Felelősség**

A mozdonyok (Locomotive) váltótól (Switch) váltóig egy adott Direction szerint mozognak, amik az alábbiak lehetnek. Ha a mozdony rálép egy váltóra, a váltó átadja neki az új Directiont. A következő váltóig e szerint a Direction szerint fog mozogni.

- **Attribútumok**

- **NEXT:** A sínek next-jére kell lépjen. (A váltó főágán, vagy az előre kapcsolt mellékágon lépked.)
- **PRE:** Ha váltón áll, akkor még egyszer a next-re kell lépni, de az utána következő sínek previous-án folytatja. (A visszakapcsolt mellékágon lépked.)
- **PREPRE:** A sínek previous-ára kell lépjen. (A váltó főágán visszafele lépked.)

4.3.4 Game

- **Felelősség**

A szimuláció legmagasabb szintje. Itt tud beleavatkozni a játékos a folyamatokba a váltó (Switch) állításával, alagút (Tunnel) építésével.

- **Metódusok**

- **static void main(String[] args):** A játék futtatása.

4.3.5 Locomotive

- **Felelősség**

A mozdony fogja össze a vonatot, hiszen ő kezeli a hozzá tartozó vagonokat. A mozdony csak mozogni tud, azt, hogy hogyan mozog, azt a sín (Rail) intézi.

- **Ősosztályok**

Train

- **Metódusok**

- **void addWagon(Wagon wagon):** Hozzáad egy vagon a mozdony vagonjainak listájának a végéhez.
- **Direction getDir():** Getter függvény, visszaadja a mozdony mozgási irányát.
- **List<Wagon>getWagons():** Getter függvény, visszaadja a mozdony mögé csatolt vagonok listáját.
- **Locomotive(Rail where):** Konstruktor, lehelyezi a mozdonyt az adott sínre.
- **void nextStep():** A mozdony léptetése. Meghívja annak a sínnek a moveToNext függvényét, amelyiken éppen áll.
- **void setDir(Direction dir):** Setter függvény, beállítja a mozdony új mozgási irányát.

4.3.6 NormalRail

- **Felelősség**

A sínnek a legáltalánosabb leszármazottja. A dolga csak a mozdony következő sínre való mozgatása. Ehhez overrideolja az örökölt moveToNext() függvényt. Ezen kívül csak erre a sínre lehet építeni alagutat (Tunnel).

- **Ősosztályok**

Rail

- **Metódusok**

- **boolean canPlaceTunnel():** True-val tér vissza, hiszen a NormalRail-re mindig lehet építeni alagutat (Tunnel).
- **void moveToNext():** Az aktuális sínen álló mozdonyt a következő sínre mozgatja.

4.3.7 Rail

- **Felelősség**

Minden sínnek az ősosztálya. Ez az osztály biztosítja, hogy vonatot (Train) lehessen rá helyezni, valamint biztosít gettert, settert a fontosabb attribútumokhoz. Az osztály absztrakt, hiszen néhány függvénye sinspecifikus. Tárolja továbbá, hogy van-e rajta mozdony (Locomotive), a váltó (SwitchRail) checkNeighbors() függvénye miatt.

- **Metódusok**

- **boolean canPlaceTunnel()**: Megmondja, hogy lehet-e alagutat (Tunnel) építeni a sínre.
- **boolean getLocomotive()**: Getter függvény, visszaadja, hogy tartózkodik-e mozdony a sínen.
- **Rail getNext()**: Getter függvény, visszaadja a következő sínt.
- **Rail getPrevious()**: Getter függvény, visszaadja az előző sínt.
- **abstract void moveToNext(Locomotive loco)**: Az aktuális sínen álló mozdonyt a következő sínre mozgatja, valamint ha a sínnek még van dolga a paraméterben kapott Locomotive-val, azt elvégzi.
- **Rail()**: Paraméter nélküli konstruktor, létrehoz egy sínt.
- **void set(Rail previous, Rail next)**: Setter függvény, beállítja az előző, és a következő sínt.
- **void setLocomotive(boolean onrail)**: Setter függvény, ami az adott sínre beállítja, hogy van-e rajta mozdony (Locomotive).
- **void setNext(Rail next)**: Setter függvény, beállítja a következő sínt.
- **void setPrevious(Rail previous)**: Setter függvény, beállítja az előző sínt.

4.3.8 Stage

- **Felelősség**

Felépíti a pályát, lehetőséget biztosít a szimulálásra a metódusai segítségével. Valamint a pálya játékos általi manipulálása ebben a classban van megvalósítva.

- **Metódusok**

- **generateTrain(int length, Rail where)**: Generál egy vonatot, csak olyan színű vagonokkal, amilyen állomások aktívak, és lerakja a paraméterként kapott sínre. A hossz paraméter a vonat mozdony nélküli hossza (vagyis a vagonok száma).
- **void init()**: Létrehozza, felépíti a pályát.
- **boolean modifyTiles(int x, int y)**: A paraméterben kapott x és y paraméter szerint kikeresi a pályán van-e sín, és ha van, akkor az milyen típusú. Ennek megfelelően, ha Switch, akkor vált, ha sima sín, akkor alagutat épít rá, valamint rombol le. Visszaadja, hogy sikeres volt-e a manipuláció.
- **void tick()**: Az összes Stage-hez tartozó Locomotive-ot lépteti, valamint az összes váltó felméri, hogy a környezetében van-e Locomotive, és aszerint állítja be a saját Direction enumját.

4.3.9 StationRail

- **Felelősség**

Az állomások (StationRail), ahol a vagonok (Wagon) kiürülhetnek. Ez akkor történhet meg, ha az első nemüres vagon színe megegyezik az állomás színével. Ezen kívül lépteti a vonatot, mint a többi sín. Egy állomás lehet nyitva, vagy zárva, ha zárva van, nem lehet kiüríteni a vagonokat.

- **Ősosztályok**

Rail

- **Metódusok**

- **boolean canPlaceTunnel()**: False-szal tér vissza, hiszen StationRail-re sosem lehet alagutat (Tunnel) építeni.
- **Color getColor()**: Getter függvény, visszaadja az állomás színét.
- **boolean getUsable()**: Getter függvény, visszaadja, hogy nyitva van-e az állomás.
- **void moveToNext(Locomotive loco)**: A játékszabályok szerint kiüríti a paraméterben kapott mozdonyhoz tartozó valamelyik vagon. Az aktuális sínen álló mozdonyt a következő sínre mozgatja.
- **void setUsable(boolean usable)**: Setter függvény, beállítja, hogy használható-e az állomás.
- **StationRail(Color color, boolean open)**: Konstruktor, ami beállítja az állomás színét, és hogy használatban van-e.

4.3.10 SwitchRail

- **Felelősség**

A váltó (SwitchRail) tárol egy Direction enumot, melyet a szomszédjaiban lévő mozdonyok (Locomotive) állásából dönt el. Ezt a Directiont adja át a rálépő mozdonyoknak. A mozdony szerint a Direction szerint fog lépni a következő váltóig. A váltó továbbá nem kettő sínt köt össze, mint a többi sín, hanem hármat. Van egy főág-beli sín, valamint kettő mellékág-beli sín. A mellékág-beli sínek közül lehet a váltás funkció segítségével kiválasztani, hogy melyik legyen aktív.

- **Össztályok**

Rail

- **Metódusok**

- **boolean canPlaceTunnel()**: False-szal tér vissza, hiszen SwitchRail-re sosem lehet alagutat (Tunnel) építeni.
- **void checkNeighbors()**: Megnézi, hogy a váltó szomszéd-sínein tartózkodik-e vonat.
- **Rail getNext()**: Visszaadja az aktív sínt.
- **void moveToNext(Locomotive loco)**: Átadja a paraméterben kapott mozdonyt az új Directiont. Az aktuális sínen álló mozdonyt a következő sínre mozgatja.
- **void setOther(Rail othernext, Rail active)**: Beállítja a váltó másik állásának megfelelő sínt, illetve hogy melyik az aktív pillanatnyilag.
- **boolean setSwitch()**: Megvizsgálja, hogy van-e rajta vonat. Ha van rajta, akkor nem lehet váltani, ha nincs rajta, akkor átváltja a másik állásba. Visszatér, hogy sikerült-e a váltás.

4.3.11 Train

- **Felelősség**

A sínen járó eszközöket megvalósító absztrakt osztály.

- **Metódusok**

- **Rail getRail()**: Getter függvény, visszaadja melyik sínen áll.
- **void setRail(Rail rail)**: Setter függvény, beállítja, hogy melyik sínen áll.
- **Train(Rail where)**: Konstruktorként, lehelyezi a vonatot a kapott sínre.

4.3.12 Tunnel

- **Felelősség**

Alagút osztály. Csak NormalRail fölé helyezhető el. Egyszerre csak 1 példány lehet a pályára lehelyezve.

- **Metódusok**

- **List<Rail> modifyTunnel(Rail rail)**: Ha a paraméterben kapott sínre (Rail) már van építve bejárat, akkor azt törli. Ha még egy bejárat sincs lerakva, akkor lerakja a sínre. Ha már van egy bejárat lerakva, megvizsgálja, hogy a lerakott bejárat és a paraméterben kapott sín között lehet-e létrehozni alagutat, ha lehet, létrehozza. Visszatér a manipulált sínek listájával (kirajzolásnál hasznos lesz).

4.3.13 Wagon

- **Felelősség**

A vagonokban (Wagon) utasok helyezkednek el, őket kell eljuttatni az állomásokra (StationRail). A játékszabályok szerint lehet kiüríteni a vagonokat.

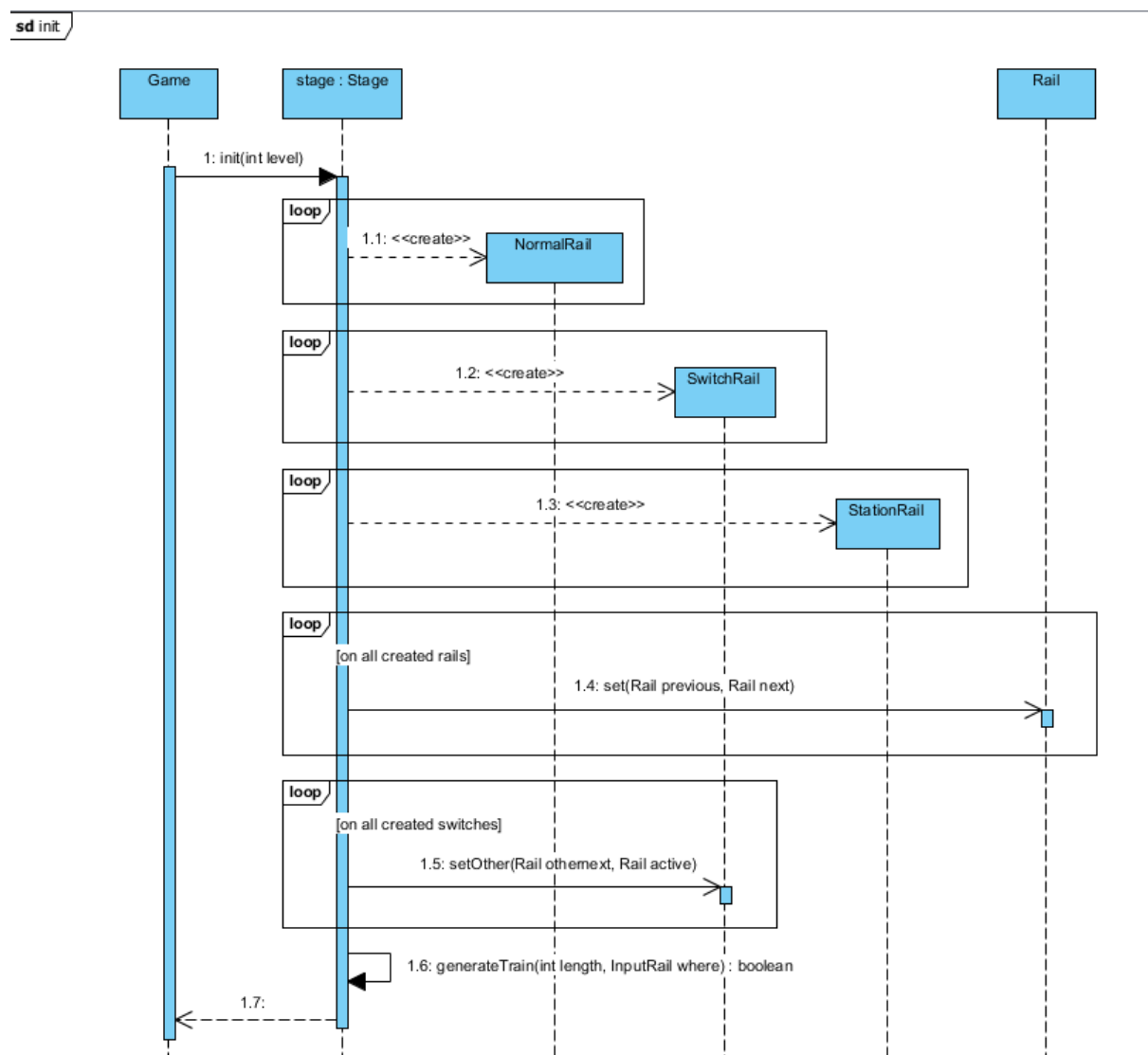
- **Ősosztályok**

Train

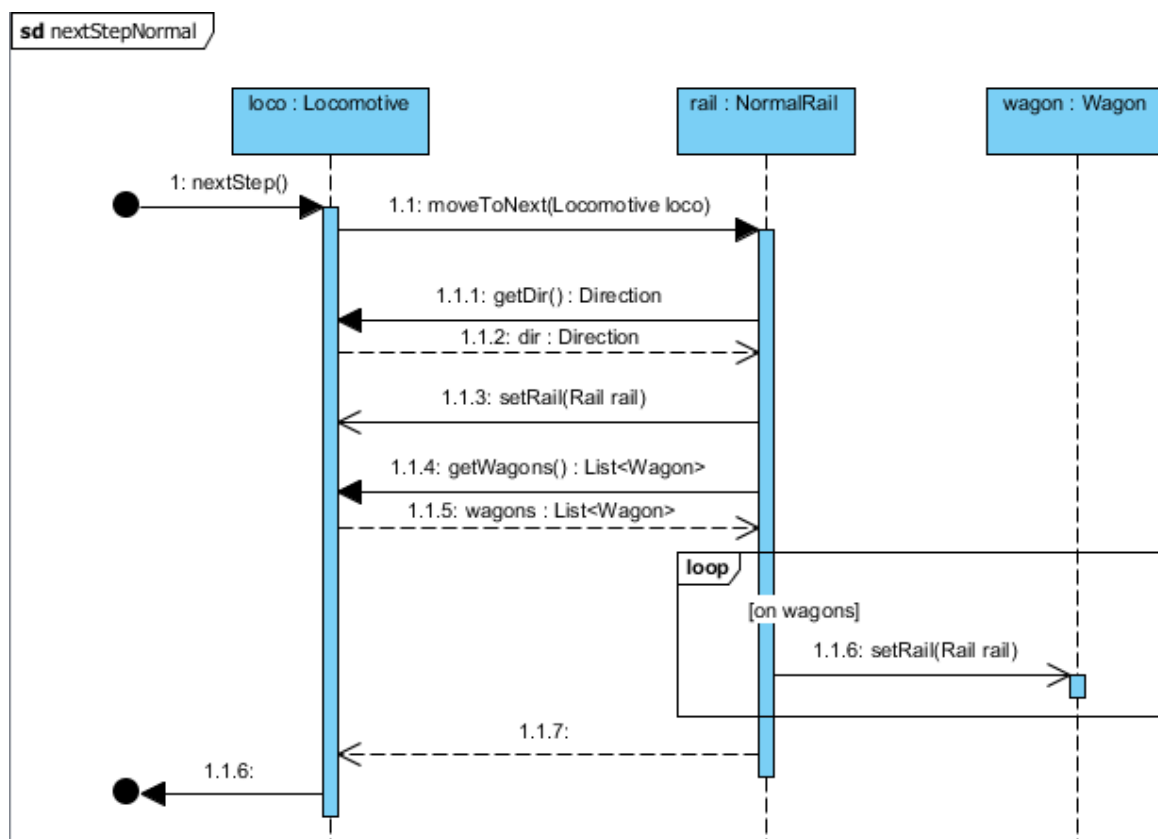
- **Metódusok**

- **void empty()**: Kiüríti a vagon.
- **Color getColor()**: Getter függvény, visszaadja a vagon színét.
- **boolean isEmpty()**: Getter függvény, visszaadja, hogy üres-e a vagon.
- **Wagon(Rail where, Color color)**: Konstruktor, beállítja a vagon pozícióját, illetve színét.

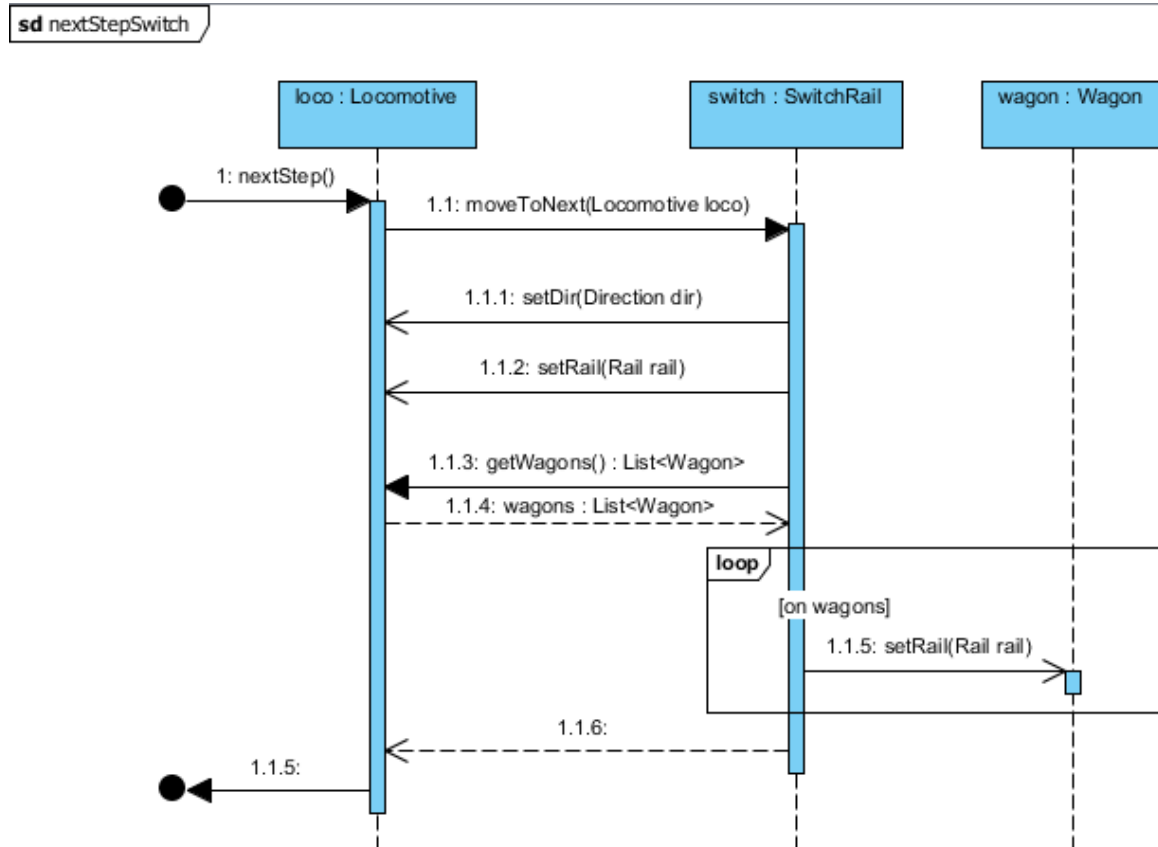
4.4 Szekvencia diagramok



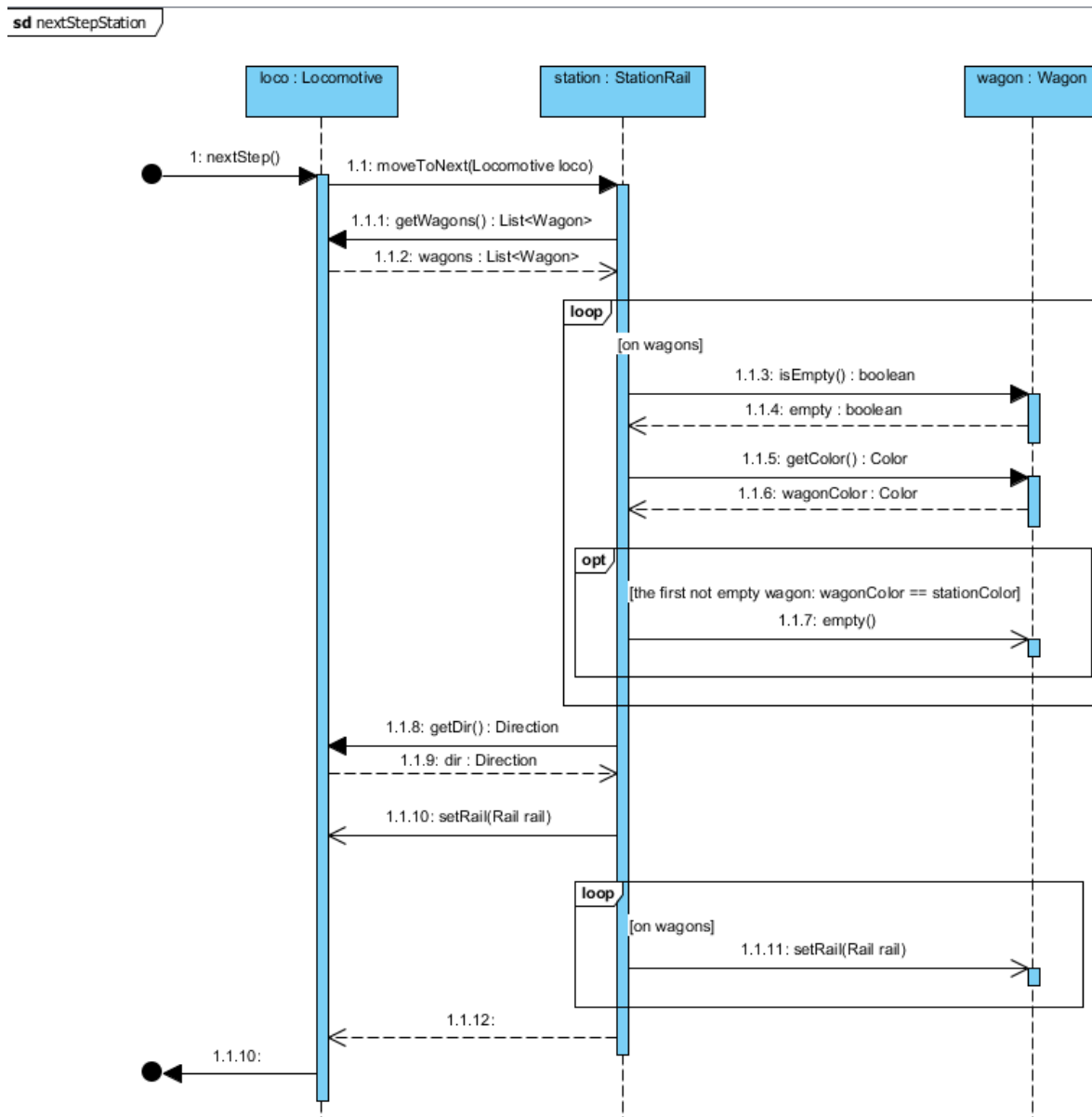
4.2. ábra
Inicializálás szekvenciadiagramja



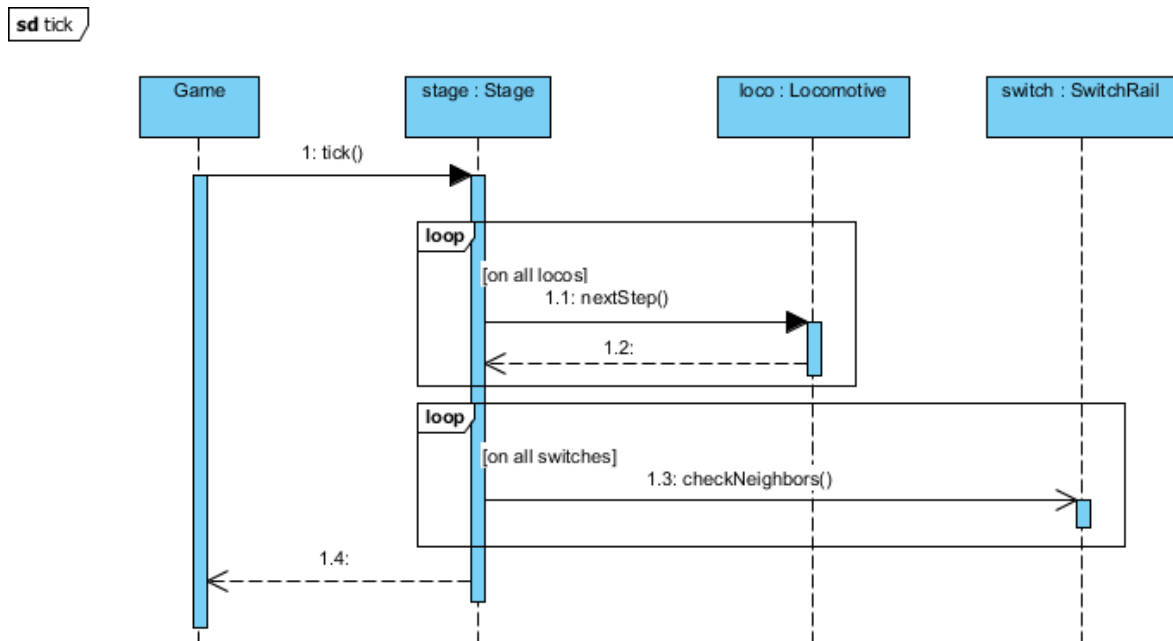
4.3. ábra
NormalRail-en lépés szekvenciadiagramja



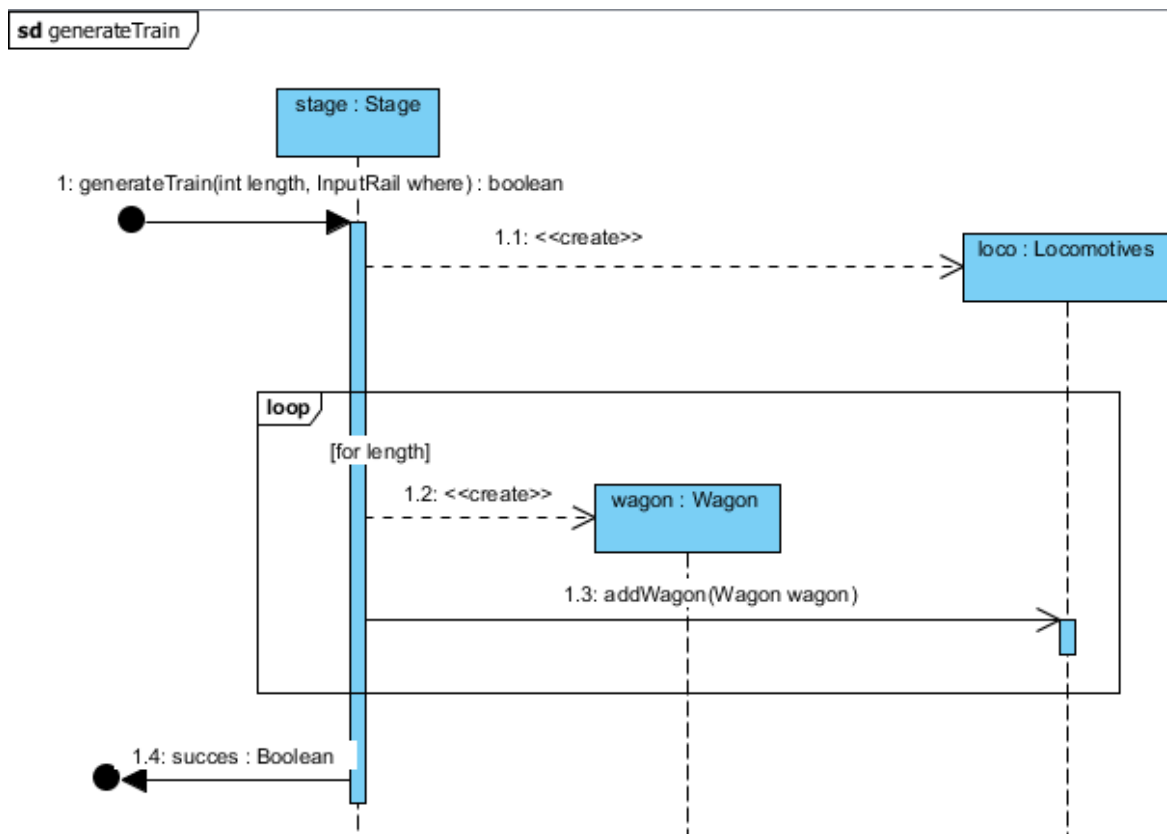
4.4. ábra
Váltóról továbblépés szekvenciadiagramja



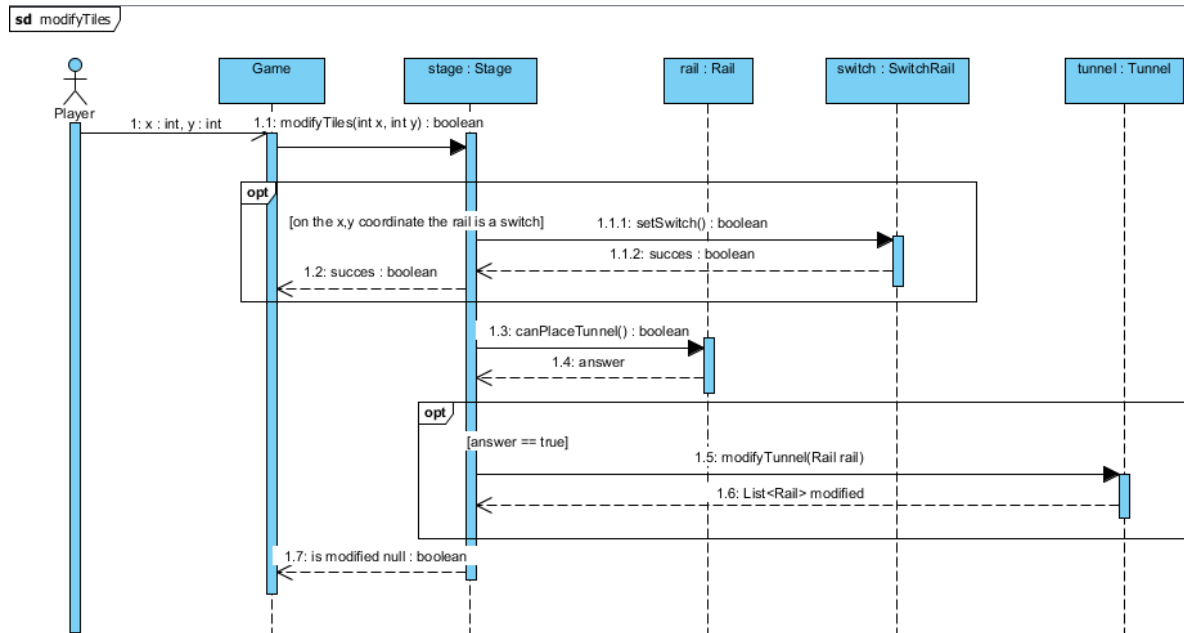
4.5. ábra
Állomásról továbblépés szekvenciadiagramja



4.6. ábra
Következő időpillanatba lépés szekvenciadiagramja

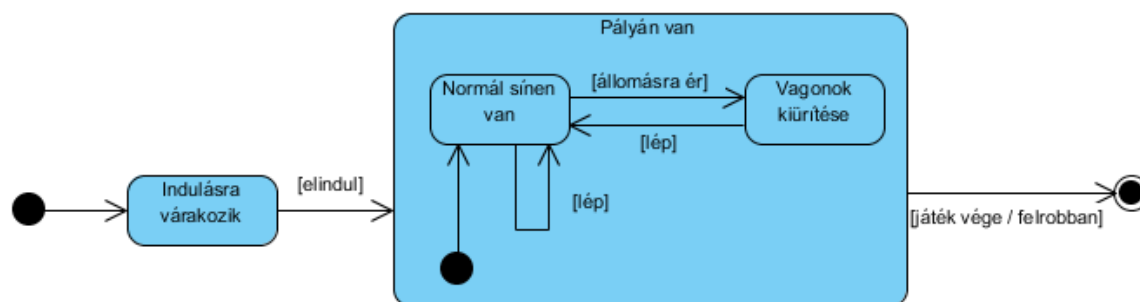


4.7. ábra
Vonatok generálásának szekvenciadiagramja

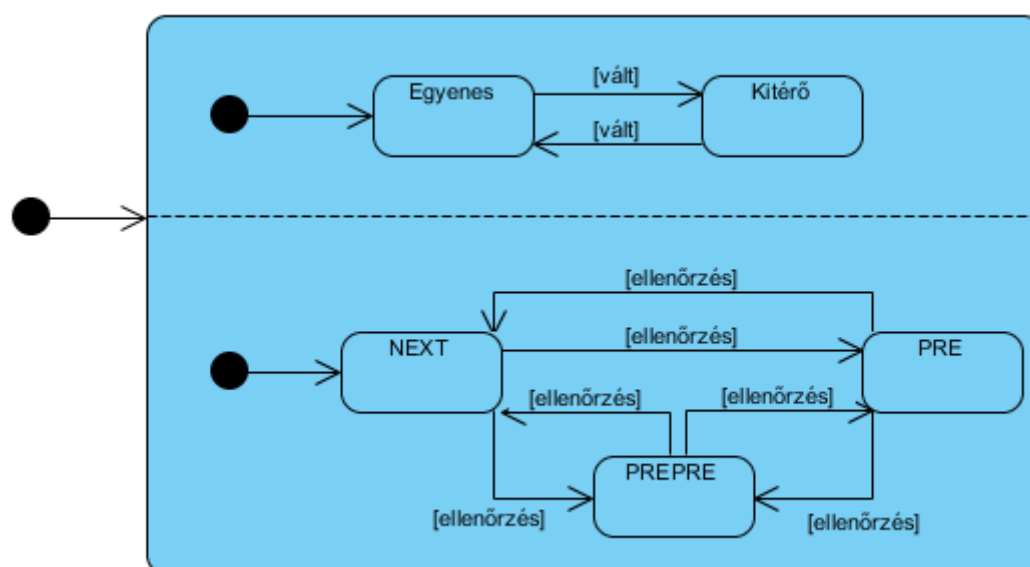


4.8. ábra
Pálya manipulálásának szekvenciadiagramja

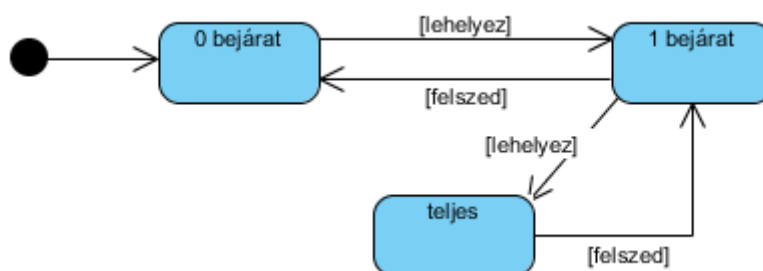
4.5 State-chartok



4.9. ábra
Vonat állapotdiagramja



4.10. ábra
Válto állapotdiagramja (magyarázat a 4.3.3-as pontban)



4.11. ábra
Alagút állapotdiagramja

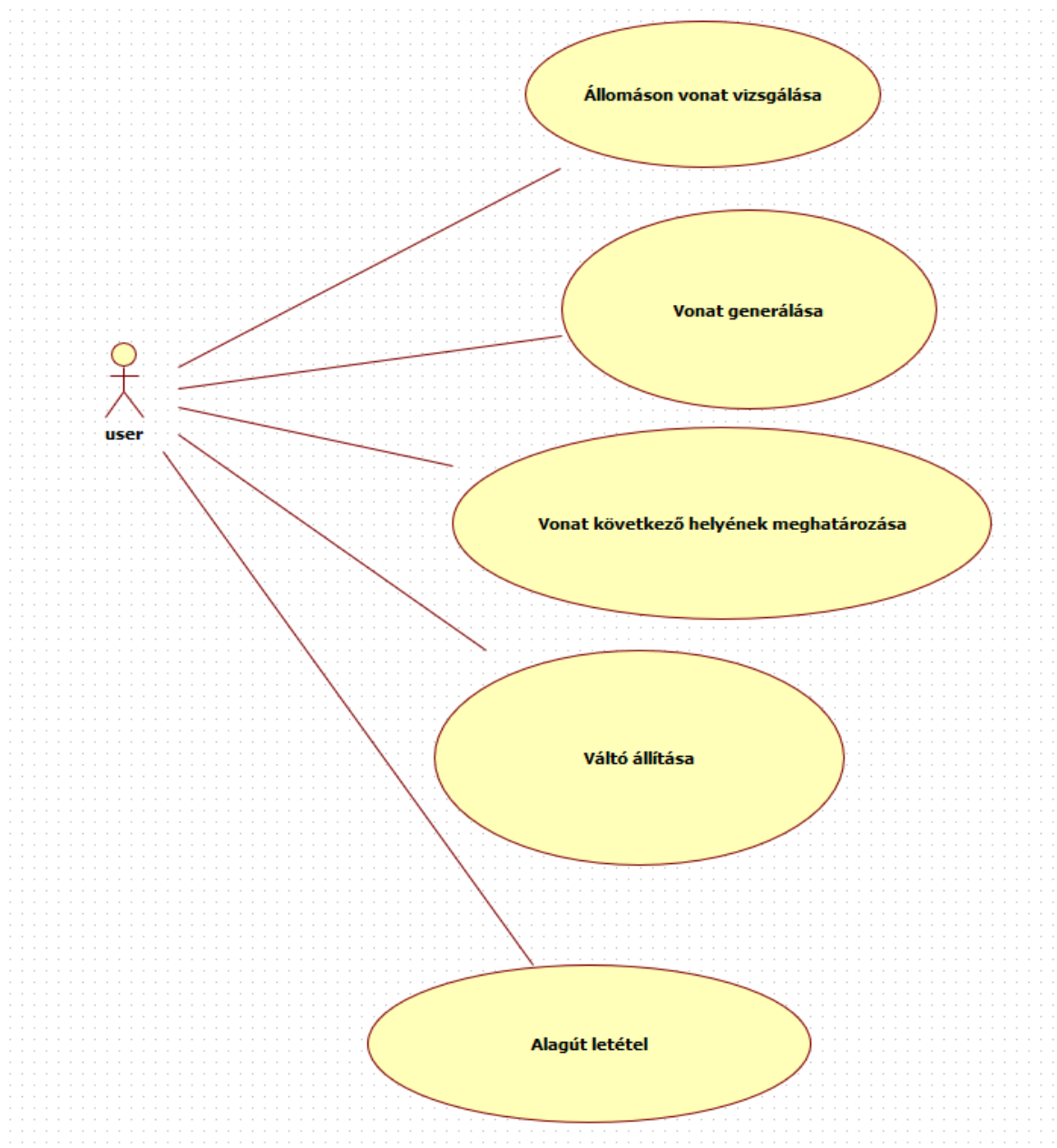
4.6 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.03.03. 19:00	1 óra	Bartók Burom Koncsik Németh Pekk	Tevékenység: Változtatások megbeszélése
2017.03.03. 20:00	30 perc	Bartók	Tevékenység: Objektumkatalógus átdolgozása
2017.03.03. 20:00	2 óra	Burom Koncsik	Tevékenység: Locomotive és Rail osztályok átdolgozása
2017.03.04. 12:00	2 óra	Burom Koncsik	Tevékenység: Szekvenciadiagramok elkészítése
2017.03.04. 14:00	3 óra	Németh Pekk	Tevékenység: Class leírások átdolgozása.
2017.03.04. 14:00	30 perc	Bartók	Tevékenység: Class diagram átdolgozása
2017.03.05. 21:00	30 perc	Németh	Tevékenység: Változtatások dokumentálása (a dokumentum elején lévő megjegyzés)

5. Szkeleton tervezése

5.1 A szkeleton modell valóságos use-case-ei

5.1.1 Use-case diagram



5.1. ábra. Use case diagram

5.1.2 Use-case leírások

Állomáson vonat vizsgálása	Use-case neve
Ellenőrzés	Rövid leírás
user	Aktorok

Az állomáson tisztázható, hogy megegyezett-e a szín és üres volt-e az adott vagon. Ha egyezik a szín és nem üres, ürít. Ha nem, továbbmegy.	Forgatókönyv
---	---------------------

Vonat generálása	Use-case neve
Előállítás	Rövid leírás
user	Aktorok
A felhasználó megadhatja, mekkora legyen a vonat.	Forgatókönyv

Vonat következő helyének meghatározása	Use-case neve
Léptetés	Rövid leírás
user	Aktorok
A felhasználó megadhatja, milyen típusú sín lesz a soron következő és a program az szerint hajtja végre az utasításokat.	Forgatókönyv

Váltó állítása	Use-case neve
Váltó beállítás	Rövid leírás
user	Aktorok
Ha a felhasználó azt adja meg, hogy a manipulálni kívánt sín egy váltó, akkor a váltót átváltja a program.	Forgatókönyv

Alagút letétel	Use-case neve
Alagút készítés	Rövid leírás
user	Aktorok
Ha a felhasználó azt adja meg, hogy a manipulálni kívánt sín egy sima sín, akkor alagút lehelyezés.	Forgatókönyv

5.2 A szkeleton kezelői felületének terve, dialógusok

Minden függvény kiírja a nevét és meghív minden függvényt, amit majd élesben is meg fog. Logikai elágazásnál a konzolon megjelenő kérdésre a felhasználó válaszol.

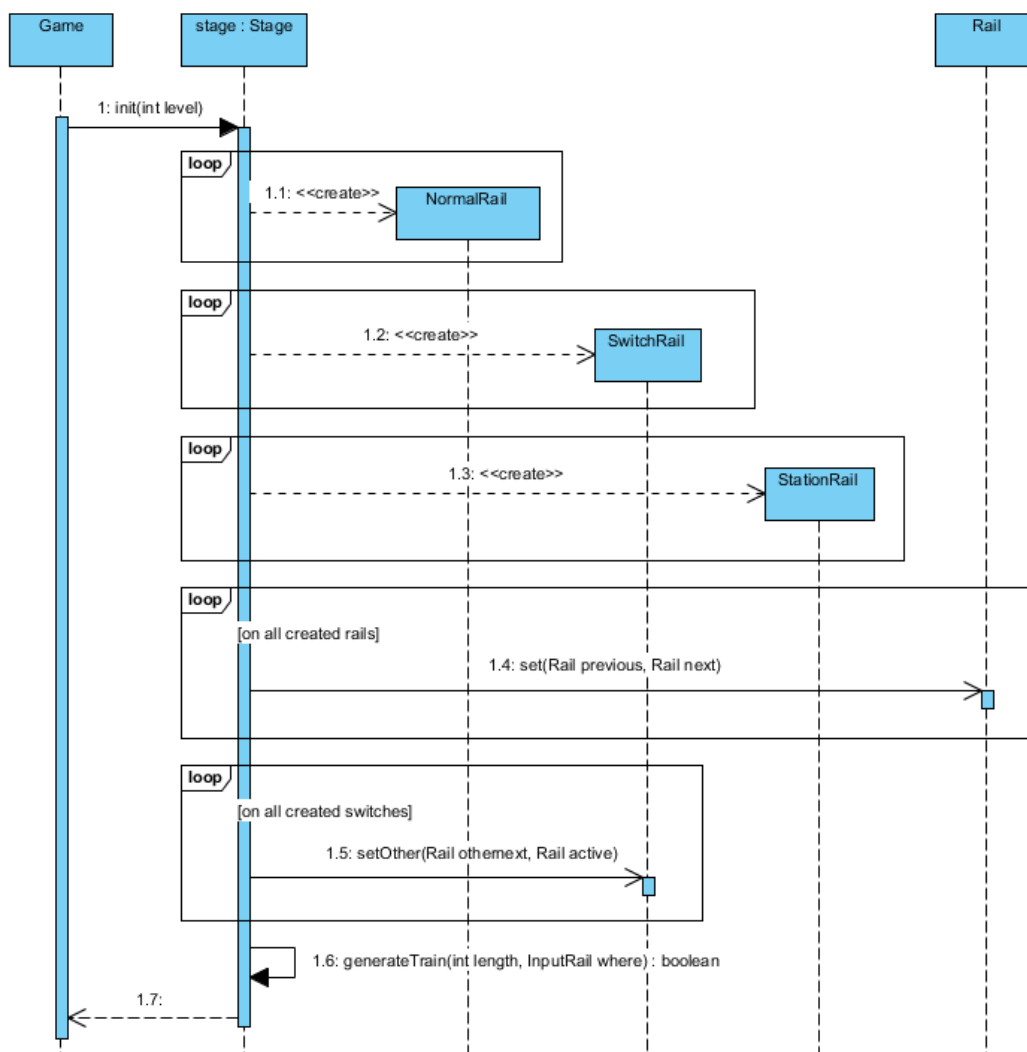
Lehetséges kérdések adott függvények után, amiket a program automatikusan feltesz:

- Üres volt és megegyezik a szín? igen / nem
- Milyen sínen van a mozdony? állomás / sima / váltó (ha nem ezek valamelyikét írjuk be, kiírja a program hogy "Nincs ilyen sín!")
- Hány vagon legyen? (számot fogad el)
- A manipulálni kívánt sín egy váltó? igen / nem
- Milyen féle a sín? állomás / sima

A kérdések után láthatóak a lehetséges válaszlehetőségek.

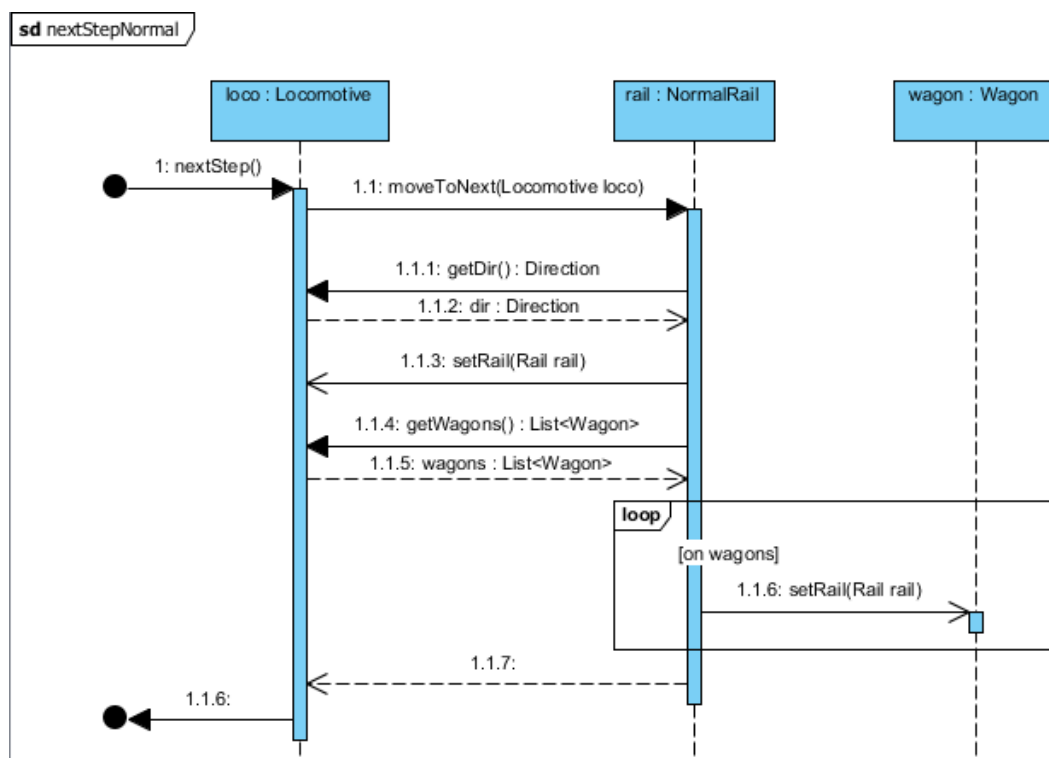
5.3 Szekvencia diagramok a belső működésre

sd init

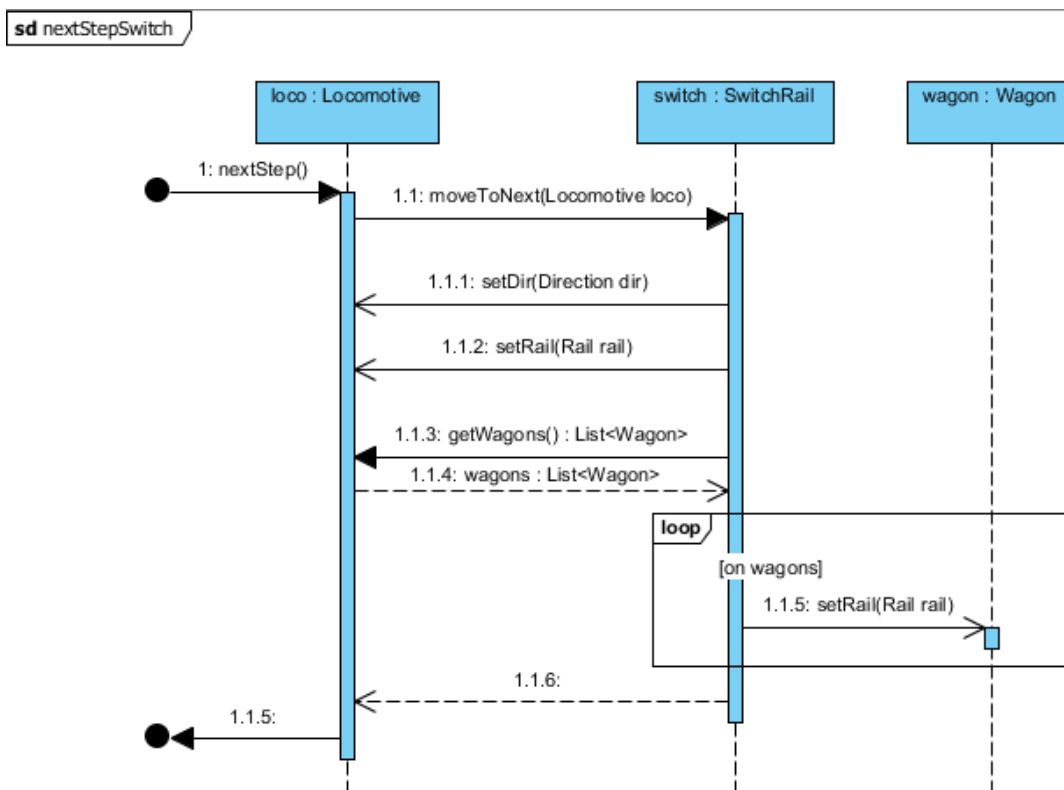


5.2 ábra.

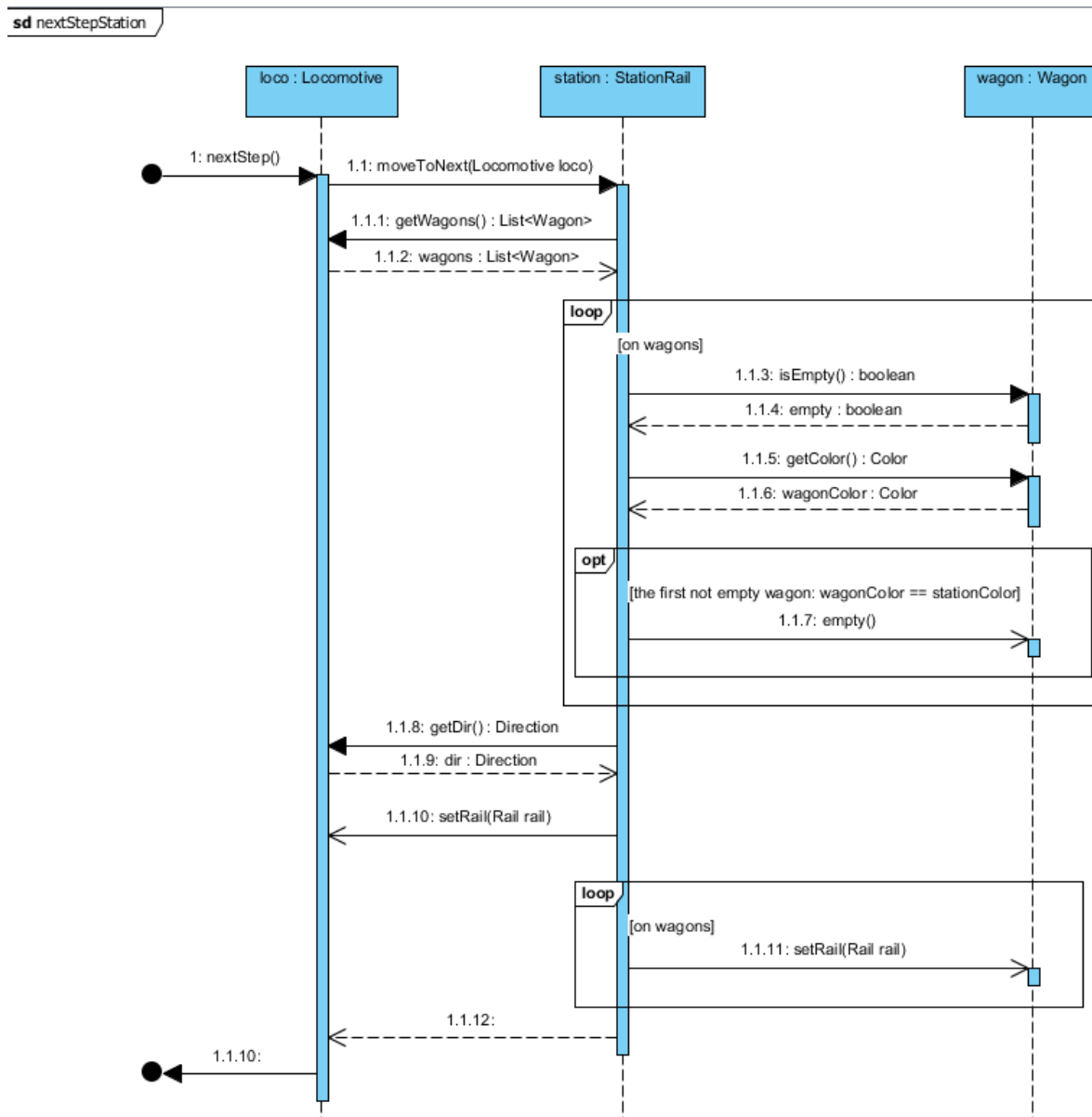
Inicializálás szekvenciadiagramja



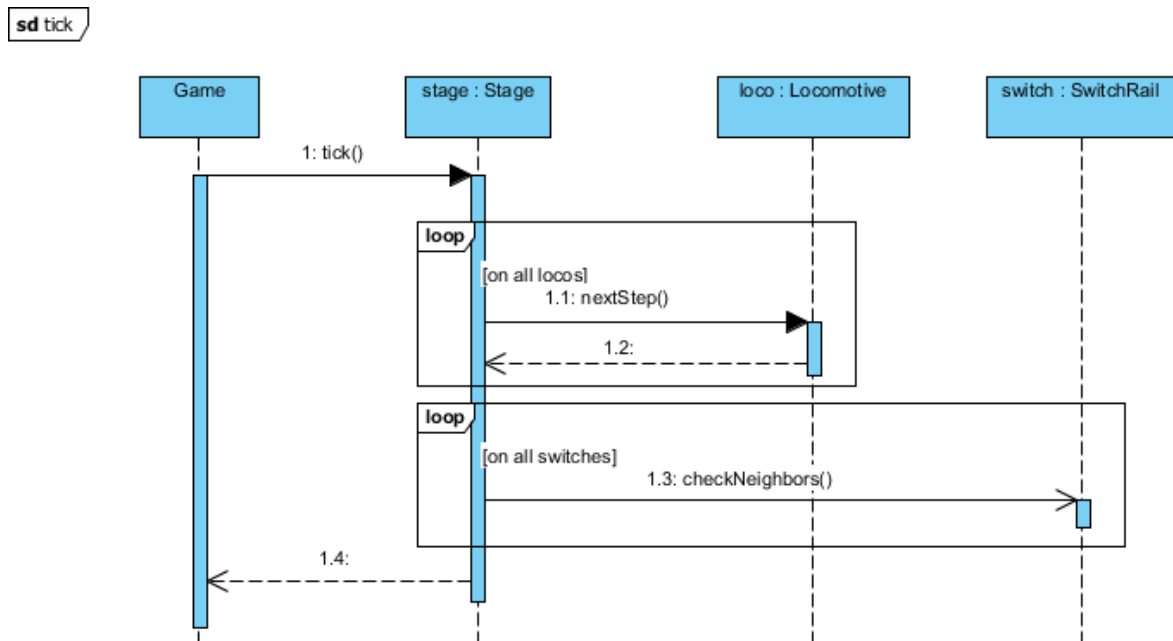
5.3. ábra
NormalRail-en lépés szekvenciadiagramja



5.4. ábra
Váltóról továbblépés szekvenciadiagramja

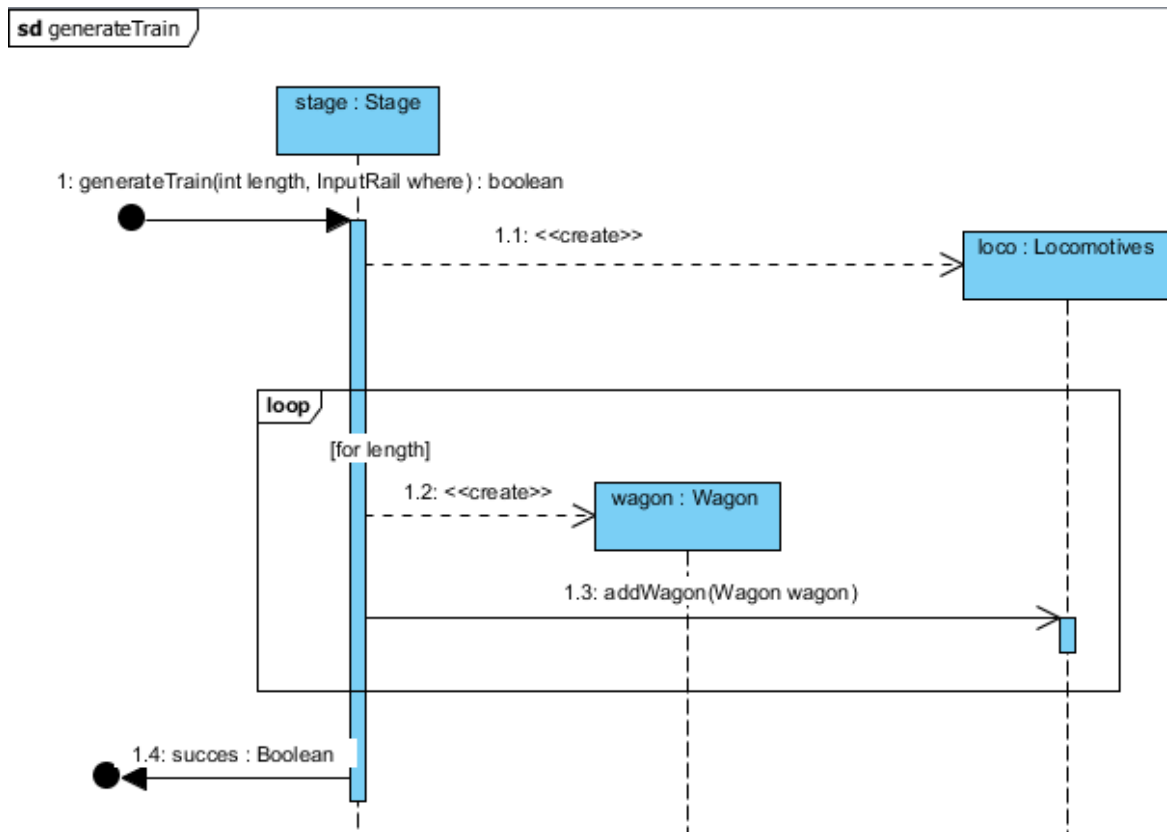


5.5. ábra
Állomásról továbblépés szekvenciadiagramja



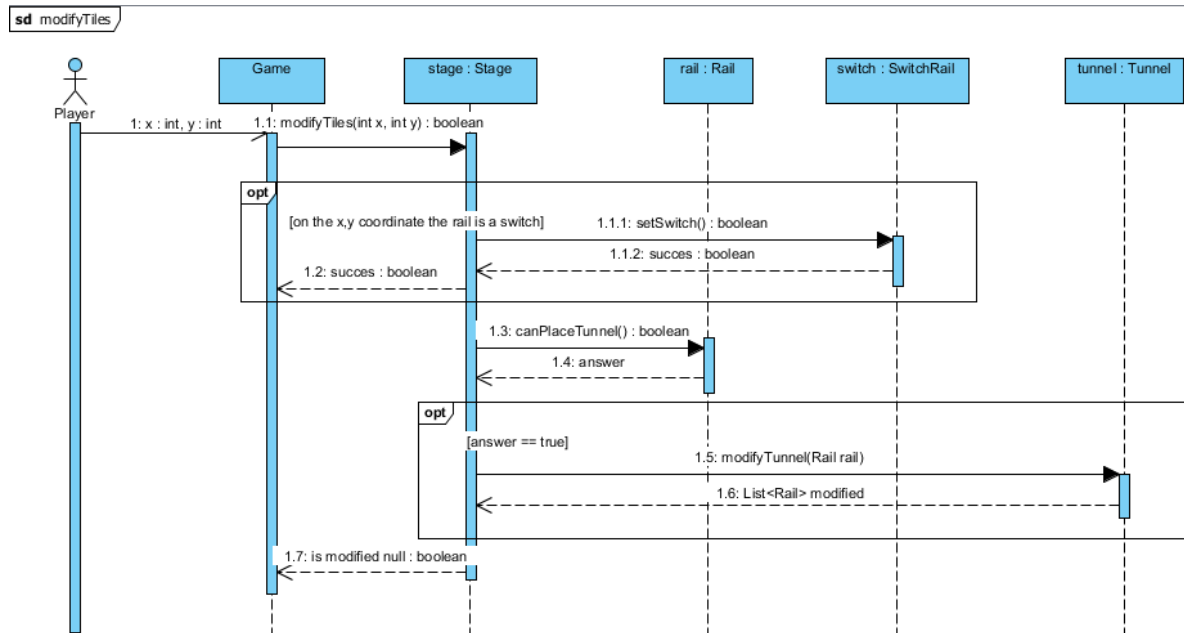
5.6. ábra

Következő időpillanatba lépés szekvenciadiagramja



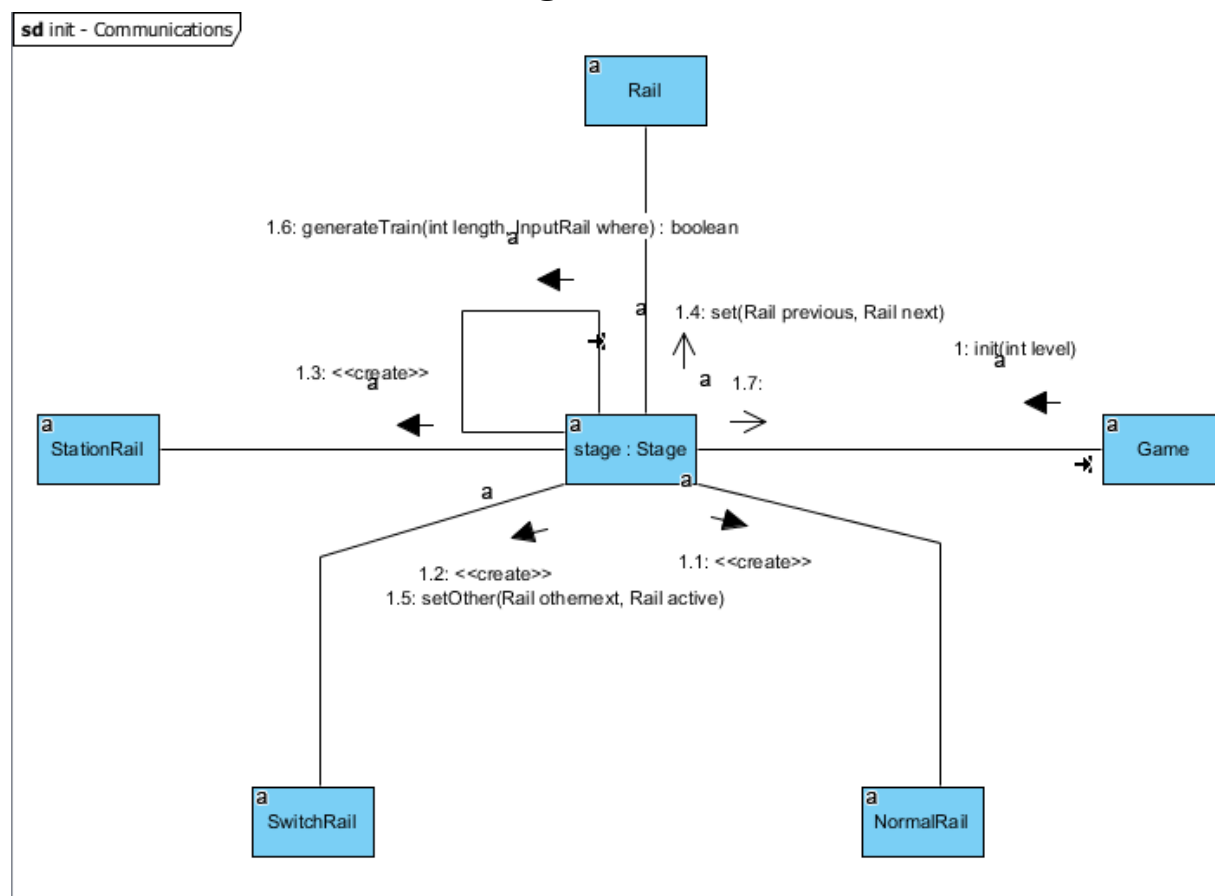
5.7. ábra

Vonatok generálásának szekvenciadiagramja

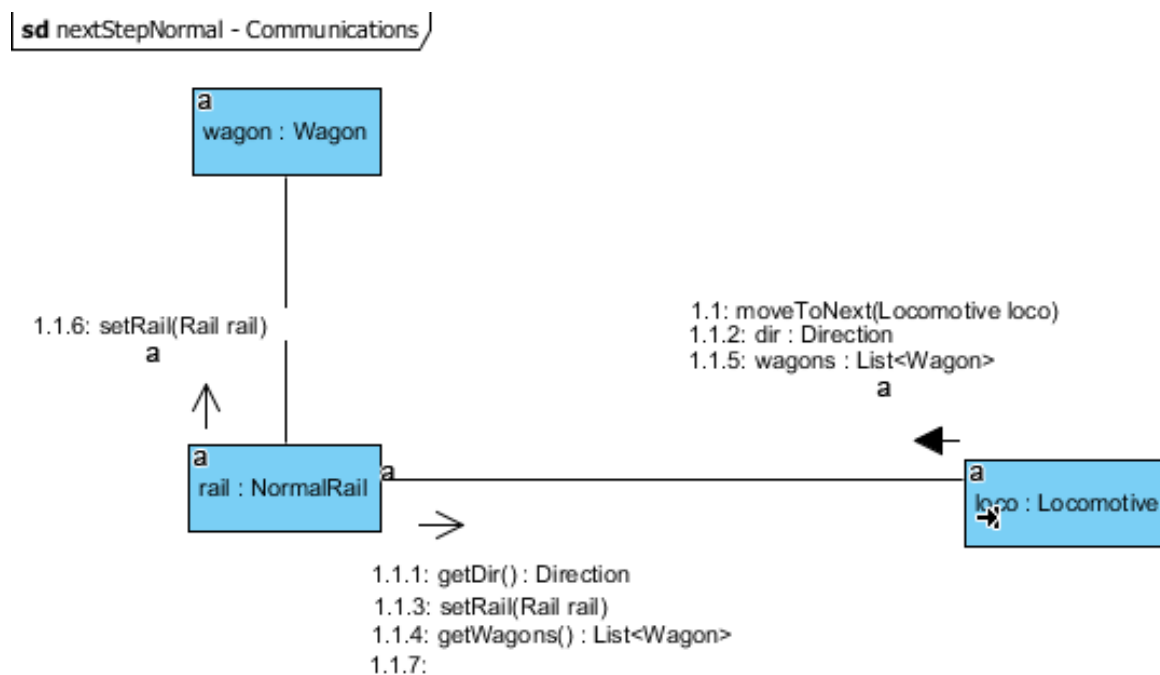


5.8. ábra
Pálya manipulálásának szekvenciadiagramja

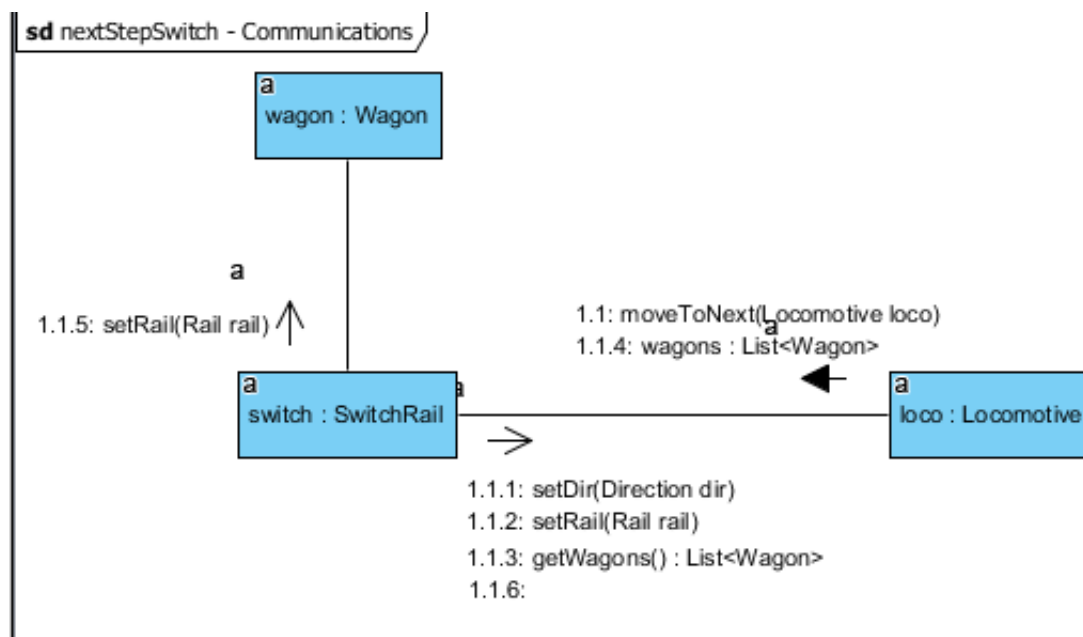
5.4 Kommunikációs diagramok



5.9. ábra.
Inicializálás kommunikációs diagramja

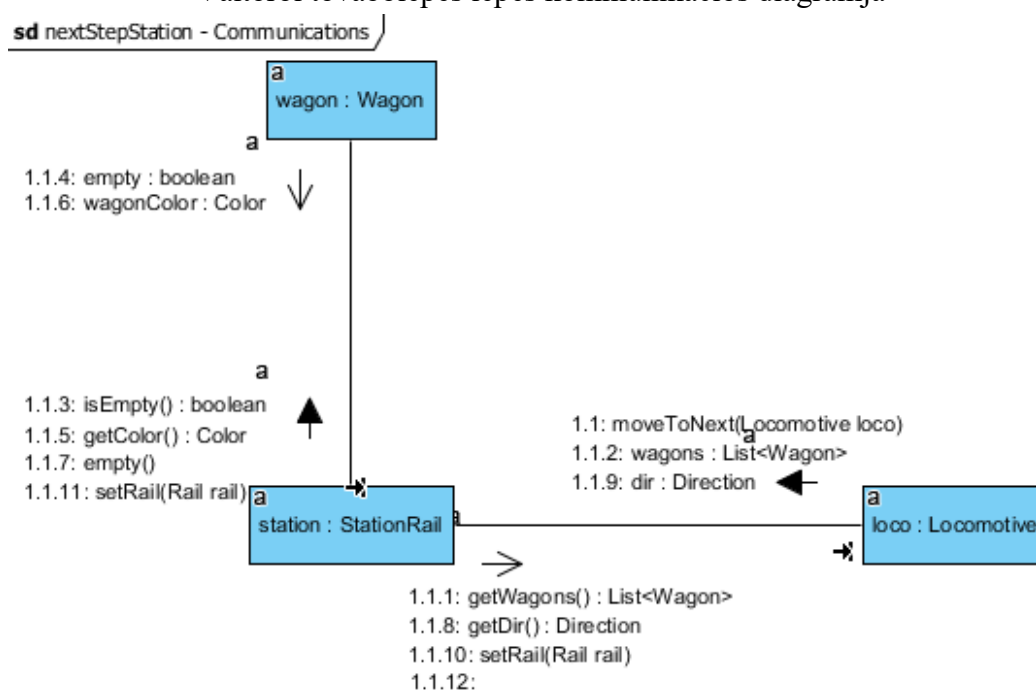


5.10. ábra.
NormalRail-en lépés kommunikációs diagramja



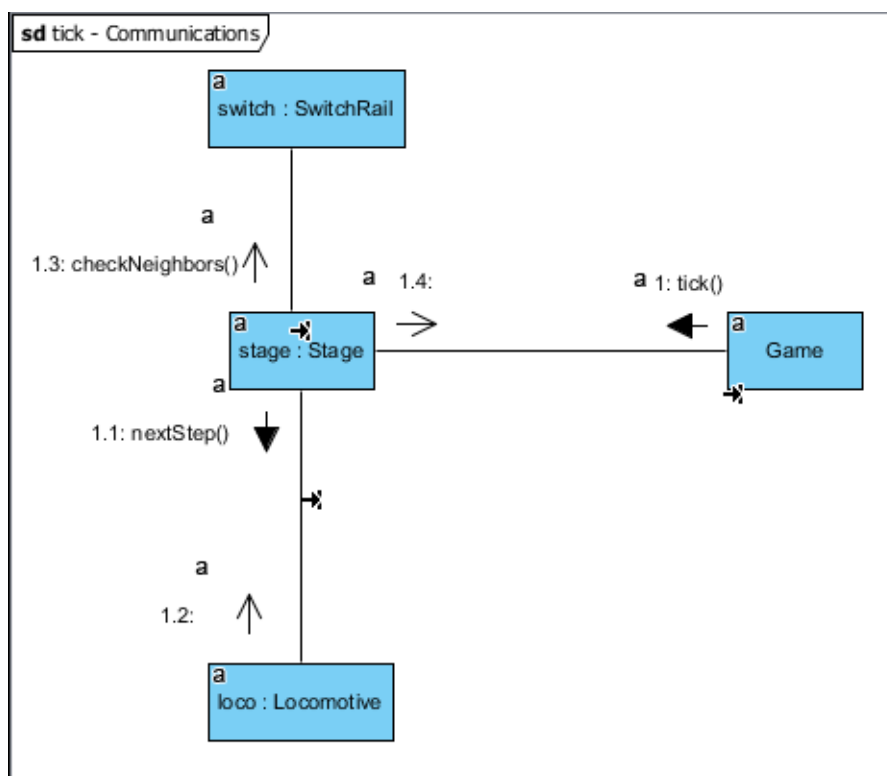
5.11. ábra.

Váltóról továbblépés lépés kommunikációs diagramja



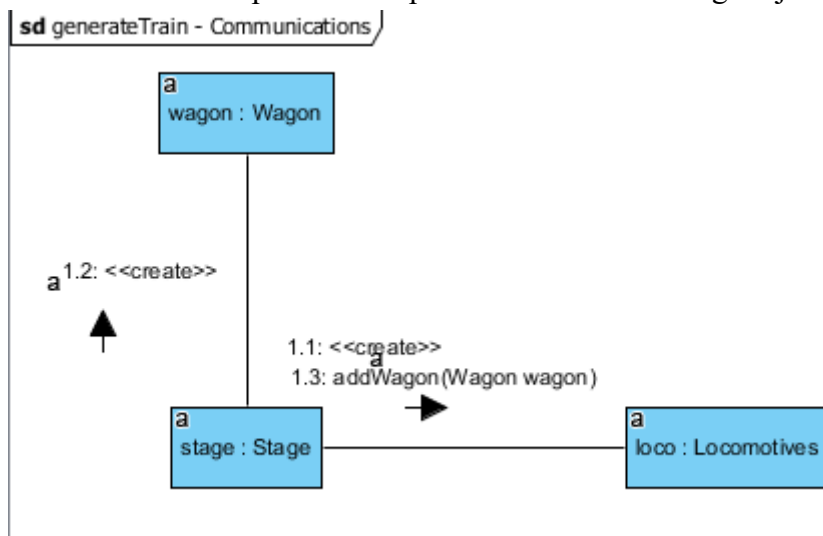
5.12. ábra

Állomásról továbblépés kommunikációs diagramja



5.13. ábra.

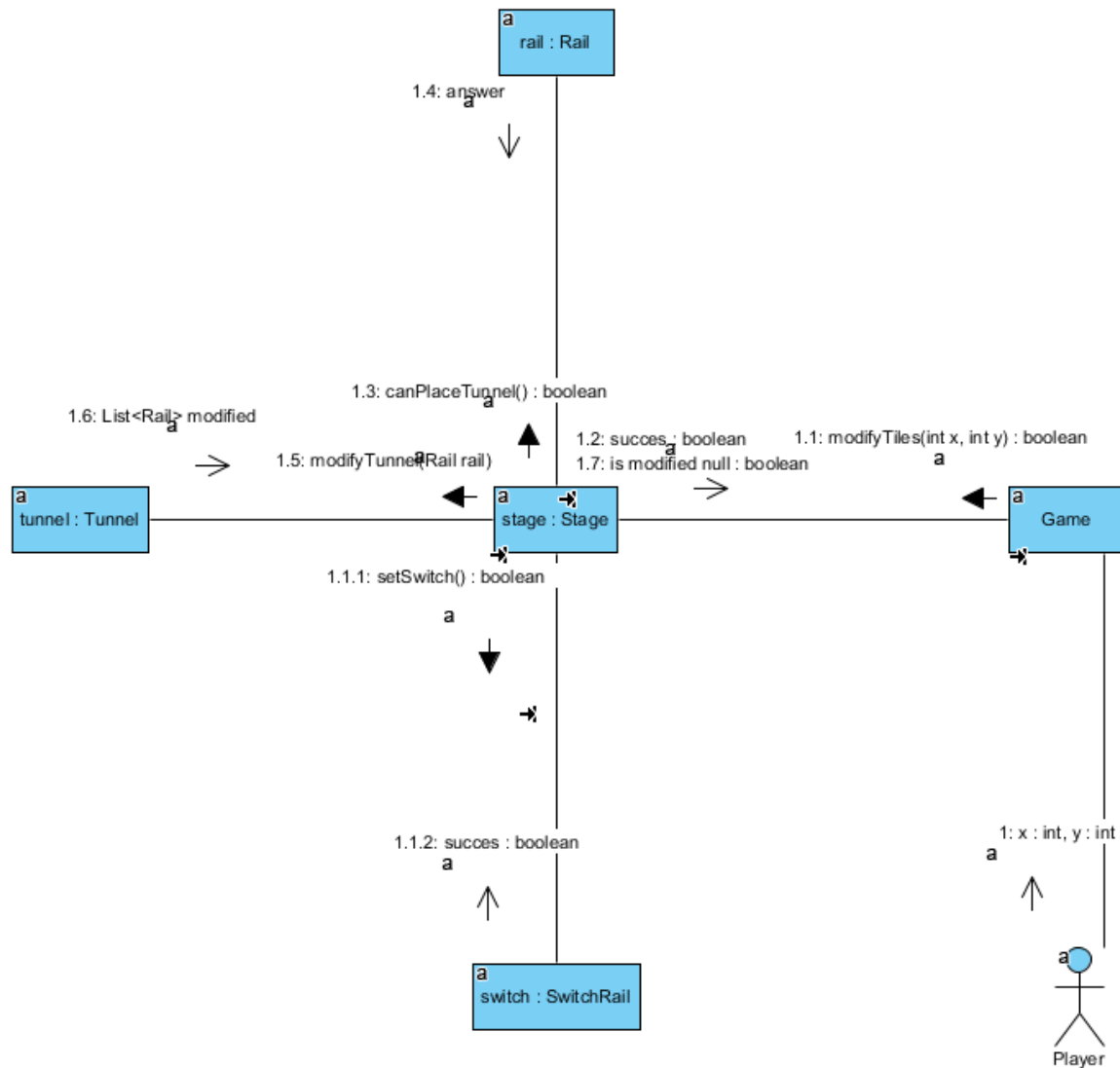
Következő időpillanatba lépés kommunikációs diagramja



5.14. ábra

Vonatok generálásának kommunikációs diagramja

sd modifyTiles - Communications



5.15. ábra

Pálya manipulálásának kommunikációs diagramja

5.5 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.03.12. 16:00	1 óra	Bartók Burom Koncsik Németh Pekk	Értekezlet. Döntés: Németh és Koncsik elkészíti a Use-Case diagrammot, és kitöltik a leírást. A többiek megcsinálják a diagrammokat.
2017.03.12. 18:00	2 óra	Németh Koncsik	Use-Case diagram, leírás
2017.03.12. 20:00	1 óra	Németh Koncsik	A szkeleton kezelői felületének terve, dialógusok
2017.03.13 19:00	3,5 óra	Bartók Burom Pekk	Szekvencia diagramok a belső működésre, kommunikációs diagramok.
2017.03.21. 22:00	1 óra	Koncsik	Szekvenciadiagramok javítása, dokumentáció formázása

6. Szkeleton beadás

6.1 Fordítási és futtatási útmutató

6.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
CollisionException.java	368 bájt	2017. 03. 19.	Saját kivétel osztály.
Color.java	91 bájt	2017. 03. 19.	Az állomások, vagonok lehetséges színe.
Direction.java	78 bájt	2017. 03. 19.	A mozdonyok mozgási iránya.
Game.java	3281 bájt	2017. 03. 19.	A szimuláció legmagasabb szintje. Itt tud beleavatkozni a játékos a folyamatokba a váltó állításával, alagút építésével.
Locomotive.java	2608 bájt	2017. 03. 19.	Mozdony osztály.
NormalRail.java	1410 bájt	2017. 03. 19.	A sínnek a legáltalánosabb leszármazottja. Vonat léptetése valósítja meg.
Rail.java	2523 bájt	2017. 03. 19.	Minden sínnek az ősoosztálya.
Stage.java	4267 bájt	2017. 03. 19.	Felépíti a pályát, lehetőséget biztosít a szimulálásra a metódusai segítségével. Valamint a pálya játékos általi manipulálása ebben a classban van megvalósítva.
StationRail.java	2904 bájt	2017. 03. 19.	Az állomások osztálya, itt szállnak le az utasok a vagonokról.
SwitchRail.java	2558 bájt	2017. 03. 19.	A váltó osztály.
Train.java	756 bájt	2017. 03. 19.	A sínen járó eszközök osztálya.
Tunnel.java	726 bájt	2017. 03. 19.	Alagút osztály.
Wagon.java	1011 bájt	2017. 03. 19.	A vagonokban utasok helyezkednek el, őket kell eljuttatni az állomásokra.

6.1.2 Fordítás

Fordítás menete:

1. A .zip állomány kicsomagolása
2. Parancssor indítása, a kicsomagolt fájlok helyére navigálás (src mappa)
3. `java -version` parancs kiadása, a telepített Java verzió meghatározásához
4. A java és javac programok működéséhez szükséges path beállítása:
`set path=%path%;C:\Program Files\Java\jdk1.8.0_121\bin`
 A fenti parancs egy példa, ha a telepítés helye, vagy a verziószám különbözik, aszerint kell futtatni a parancsot.
5. Fordítás mind az 5 mappában a következő paranccsal:
`javac game/*.java others/*.java rails/*.java stage/*.java trains/*.java`

6.1.3 Futtatás

Futtatni az előző pontban helyesen fordított fájlokat lehetséges. Parancssorban a fordított fájlok gyökérkönyvtárába (src) navigálás után a futtatás a `java game.Game` paranccsal történik.

6.2 Értékelés

Tag neve	Munka százalékban
Koncsik Norbert	26%
Németh Bence	26%
Pekk János Richárd	16%
Burom Bence	16%
Bartók Patrik István	16%

6.3 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017. 03. 18. 20:00	1 óra	Bartók Burom Pekk Koncsik Németh	Értekezlet. Döntés: Németh, Koncsik készíti el a Train és Rail osztályokat. Bartók, Burom, Pekk készíti el a többi osztályt.
2017. 03. 19. 14:00	1 óra	Koncsik, Németh	Train, Rail osztályok implementálása
2017. 03. 19. 14:00	1 óra	Pekk, Bartók, Burom	Stage, Tunnel, Game osztályok implementálása
2017. 03. 19. 18:00	1,5 óra	Koncsik	Fordítás, futtatás leírásának elkészítése

7. Prototípus koncepciója

7.0 Módosítások

A feladatkírásban az alábbi módosítások történtek:

- Sheldon új pályaelemet, kereszteződő síneket vásárolt. A kereszteződés egyszintű, a különböző irányokból jövő vonatok a kereszteződésben ütköznek.
Egy új osztályt hozunk létre, CrossRail néven. Ez is a közös Rail osztályból fog leszármazni, minden metódusával együtt. A különbség annyi lesz, hogy ennek a típusú sínnek 2 előző, és 2 következő sínje lesz. A Rail abstract void moveToNext függvényét úgy fogja felüldefiniálni, hogy megjegyzi, hogy az adott szerelvény melyik előző sínjéről érkezett, és az annak megfelelő következőre fogja tovább léptetni.
- Egyes állomásokon utasok a megfelelő színű üres kocsikba (a kocsi szerelvényben elfoglalt helyzetétől függetlenül) fel tudnak szállni.
A StationRail (állomás) osztály void moveToNext függvénye kap egy új feladatot, mostantól már nem csak üríti a megfelelő színű vagonokat, hanem egy adott valószínűséggel meg is tölti utasokkal azokat (természetesen nem ugyanazt, amelyiket éppen kiürített). Csak akkor fognak felszállni utasok, ha az adott megállón nem szálltak le. Ez úgy fog történni, hogy a megfelelő Wagon (új) fill függvényét hívja meg, ami eldönti, hogy szállnak-e fel utasok a vonatra, vagy sem.
- Sheldon bővítette a vagonkészletét. Vett szeneskocsikat, amiken nem utaznak utasok, nem is tudnak felszállni. Az utasok leszállásánál az ilyen vagonokat nem vesszük figyelembe.
Egy új osztályt hozunk létre CoalWagon néven, ami a Wagon osztály leszármazottja lesz. Ennek fekete színt fogunk adni (ilyen színt nem használunk az állomásokhoz). A Wagon osztály attribútumait és metódusait fogja örökölni. Az isEmpty függvénye mindig true értéket fog visszaadni.

Újonnan létrehozott osztályok:**7.0.1 CoalWagon**

- **Felelősség**

A szeneskocsi (CoalWagon), egy olyan vagon, amiben nem utazhatnak utasok, és a színe fekete. A vonaton bárhol elhelyezkedhet.

- **Ősosztályok**

Train -> Wagon

- **Metódusok**

- **CoalWagon(Rail where):** Konstruktor, meghívja az ősosztály konstruktorát, fekete színnel, illetve a paraméterként kapott pozícióval.
- **void empty():** A szeneskocsival nem csinál semmit, mivel itt nincsenek utasok.
- **void fill():** A szeneskocsival nem csinál semmit, mivel itt nincsenek utasok.
- **Color getColor():** Getter függvény, a vagon színét adja vissza, ami jelen esetben a fekete.
- **boolean isEmpty():** Getter függvény, visszaadja, hogy üres-e a vagon, ami jelen esetben mindig igaz lesz, mivel itt nincsenek utasok.

7.0.2 CrossRail

- **Felelősség**

A keresztirányú síneken (CrossRail) a vonatok azonos szintben keresztezhetik egymás útját. 2 előző és 2 következő sínje van. A dolga a mozdony következő sínre való mozgása. Ehhez overrideolja az örökölt moveToNext() függvényt, úgy hogy megjegyzi, hogy az adott szerelvény melyik előző sínjéről érkezett, és az annak megfelelő következőre fogja tovább léptetni.

- **Ősosztályok**

Rail

- **Metódusok**

- **boolean canPlaceTunnel():** False-szal tér vissza, hiszen a CrossRail-re sosem lehet alagutat (Tunnel) építeni.
- **void checkNeighbors():** Hasonlóképp működik, mint a váltó (SwitchRail) checkNeighbors metódusa. Minden tick végén megnézi a CrossRail, hogy a közvetlen szomszédjában van-e mozdony, és aszerint állítja be az aktív előző és/vagy következő sínjét.
- **void moveToNext():** Az aktuális sínen álló mozdonyt a következő sínre mozgatja.

Módosított osztályok:**7.0.3 StationRail**

- **Felelősség**

Az állomások (StationRail), ahol a vagonokról (Wagon) le- **illetve felszállhatnak** utasok. Leszállás akkor történhet meg, ha az első nemüres vagon színe megegyezik az állomás színével. **Felszállás akkor történhet meg, ha nem történt leszállás, valamint a vagon színe megegyezik az állomás színével.** Ezen kívül lépteti a vonatot, mint a többi sín. Egy állomás lehet nyitva, vagy zárva, ha zárva van, nem lehet kiüríteni a vagonokat.

- **Ősosztályok**

Rail

- **Metódusok**

- **boolean canPlaceTunnel():** False-szal tér vissza, hiszen StationRail-re sosem lehet alagutat (Tunnel) építeni.
- **Color getColor():** Getter függvény, visszaadja az állomás színét.
- **boolean getUsable():** Getter függvény, visszaadja, hogy nyitva van-e az állomás.
- **void moveToNext(Locomotive loco):** A játékszabályok szerint kiüríti a paraméterben kapott mozdonyhoz tartozó valamelyik vagon. Az aktuális sínen álló mozdonyt s következő sínre mozgatja.
- **void setUsable(boolean usable):** Setter függvény, beállítja, hogy használható-e az állomás.
- **StationRail(Color color, boolean open):** Konstruktor, ami beállítja az állomás színét, és hogy használatban van-e.

7.0.4 Wagon

- **Felelősség**

A vagonokban (Wagon) utasok, **vagy szén** helyezkednek el. Az utasokat kell eljuttatni az állomásokra (StationRail). A játékszabályok szerint lehet kiüríteni, **feltölteni** a vagonokat.

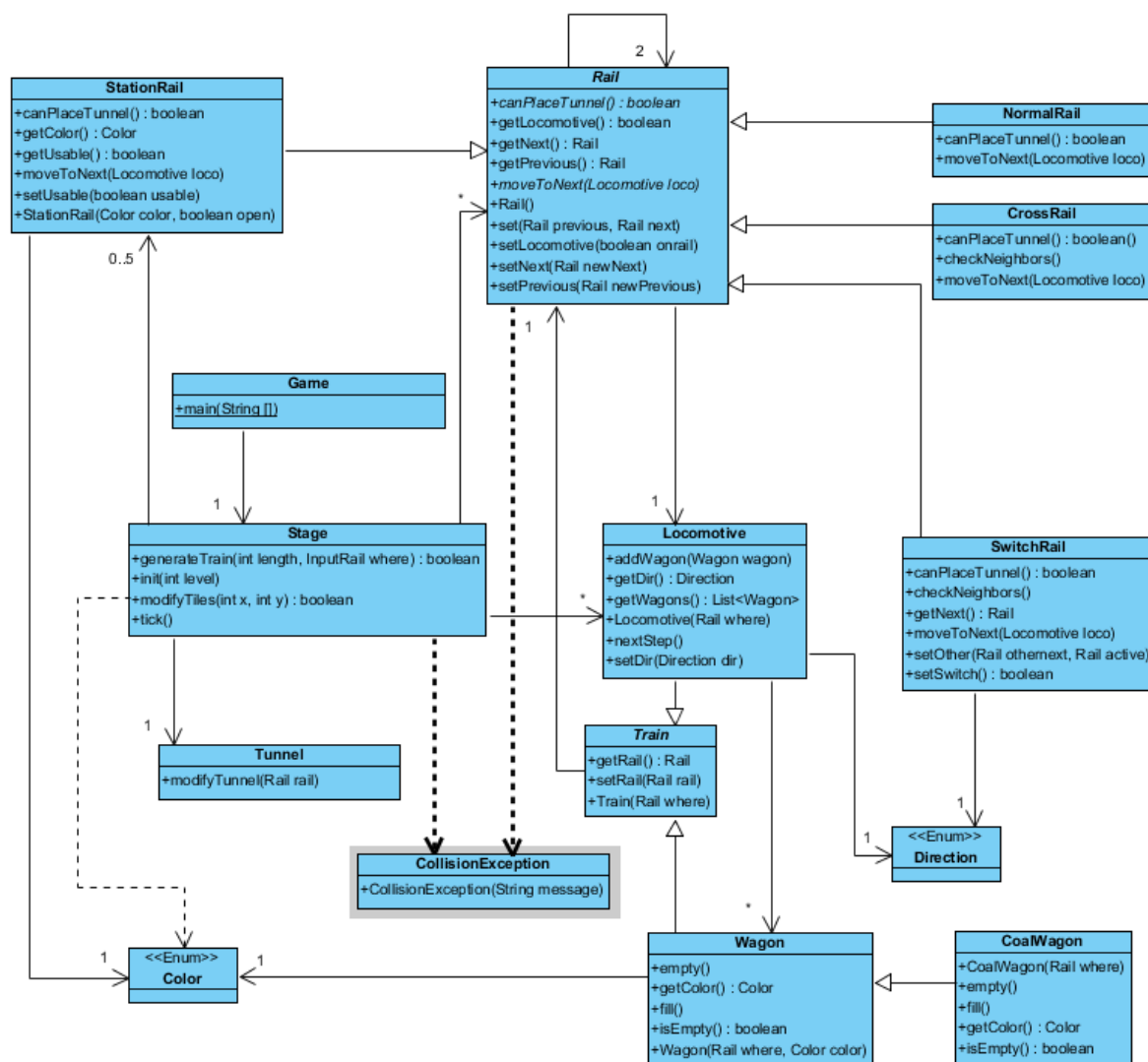
- **Ősosztályok**

Train

- **Metódusok**

- **void empty():** Kiüríti a vagon.
- **void fill():** **0.5-ös valószínűségi tényezővel feltölti a vagon, vagy nem.**
- **Color getColor():** Getter függvény, visszaadja a vagon színét.
- **boolean isEmpty():** Getter függvény, visszaadja, hogy üres-e a vagon.
- **Wagon(Rail where, Color color):** Konstruktor, beállítja a vagon pozícióját, illetve színét.

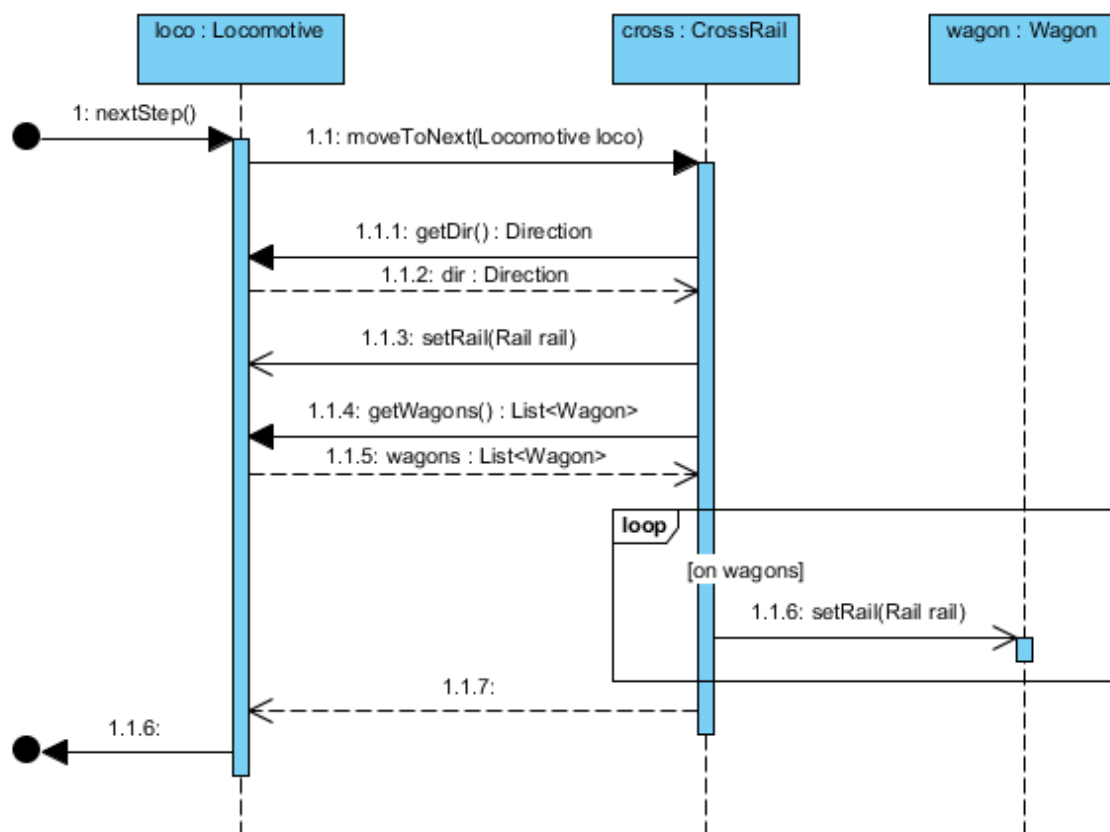
7.0.5 Módosított class diagram



7.2 ábra: Class diagram (módosított)

7.0.5 CrossRail szekvencia diagram

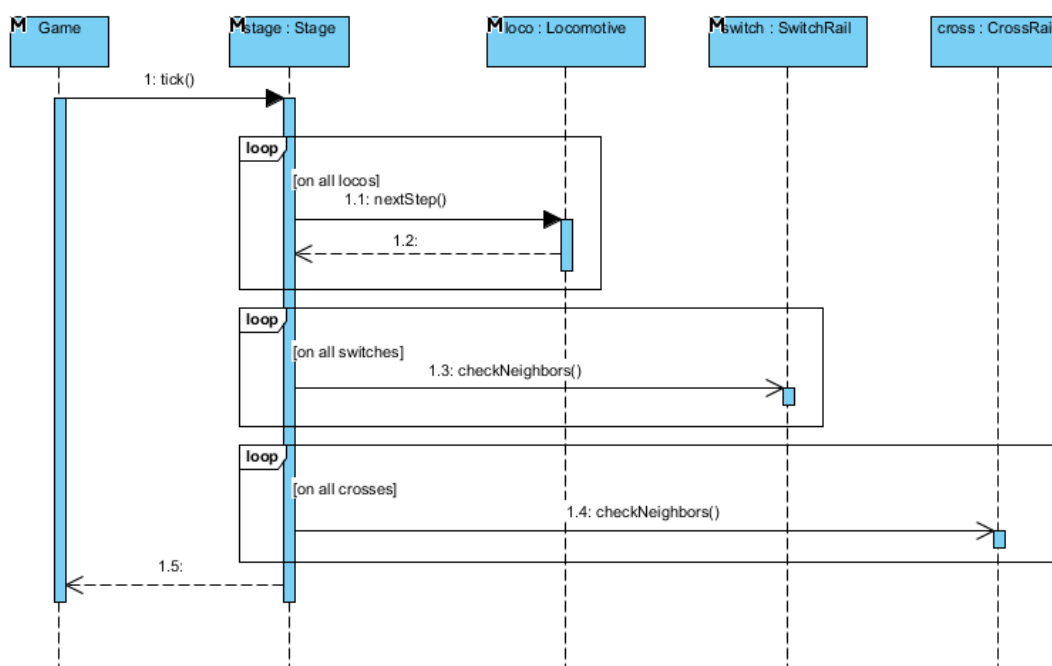
sd nextStepCross



7.3 ábra. CrossRail-en lépés szekvenciadiagramja

7.0.6 Módosított tick szekvencia diagram

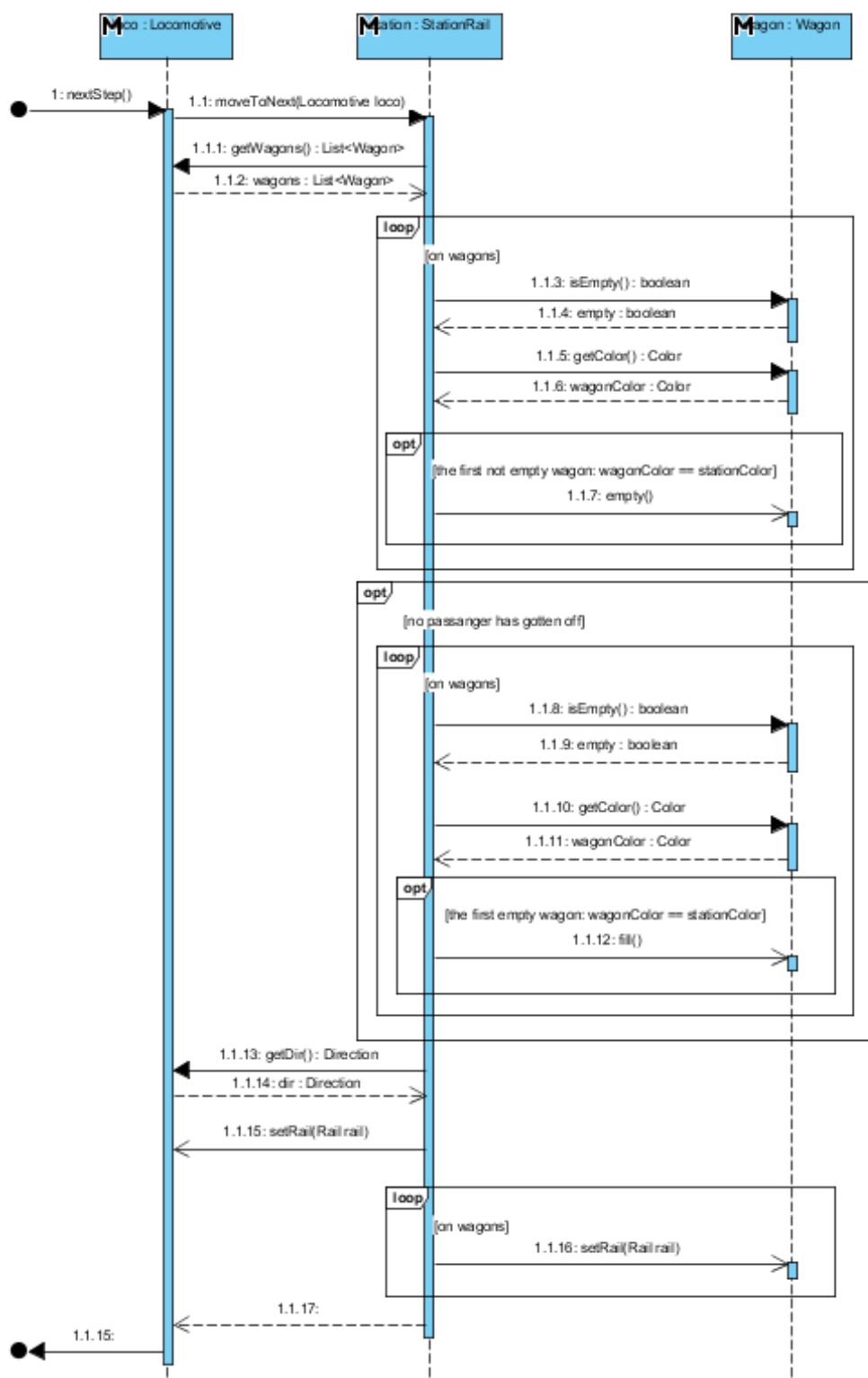
sd tick



7.2 ábra. Következő időpillanatba lépés szekvenciadiagramja (módosított)

7.0.8 Módosított StationRail szekvencia diagram

sd: nextStepStation



7.4 ábra. SwitchRail-en lépés szekvenciadiagramja (módosított)

7.1 Prototípus interface-definíciója

7.1.1 Az interfész általános leírása

Az interfész parancsokat csak a szabványos bemenetről fogad el, a kimeneteket csak a szabványos kimenetre írja ki. A tesztelő program segítségével a tesztesetek fájlokból is beolvashatóak, illetve a teszteredmények (kimenetek) fájlba menthetők. A tesztek akkor lesznek sikeresek, ha a kapott és az elvárt kimenetek megegyeznek.

7.1.2 Bemeneti nyelv

A program prototípusa a következő bemenetek fogadja el.

- **load <mapname>**
 - Leírás: A kívánt pálya betöltése.
 - Paraméter: A mapname lehet: normal, cross, station, switch. A pályák a 8. dokumentációban lesznek ismertetve.
- **next**
 - Leírás: Az összes vonat léptetése.
- **click <id>**
 - Leírás: Sínekre kattintás.
 - Paraméter: A módosítani kívánt sín sorszáma. A pályákon az összes sínnek lesz sorszáma, melyek a 8. dokumentációban lesznek ismertetve.
- **gentrain <X color color color color...>**
 - Leírás: A parancsban megadott hosszú- és színű vonat generálása a pálya bemenetére.
 - Paraméterek: X a vagonok hossza. A vagonok számának megadása után X darab colort is meg kell adni. A color lehet: b, g, o, r, y, n.
b = blue, g = green,, o = orange, r = red, y = yellow, n = black (szeneszkocsi)

7.1.3 Kimeneti nyelv

A bemeneti parancsok a következő kimeneteket adják:

- **loadMap**
Kiírja, hogy betöltődött a kiválasztott pálya.
- **nextStep**
Az összes vonat kiírja egymás alá, hogy hova lépett, illetve ha történt fel- vagy leszállás, akkor azt is. Kiírja, ha ütközött valamely vonat, vagy leszállította az összes utast.
- **click**
Abban az esetben, ha váltóra „kattintottunk”, akkor kiírja, hogy hányas sorszámú sínen (váltón) történt meg az átállítás.
Ha sima sínre (NormalRail) „kattintottunk”, akkor a következő kimenetek lehetségesek:
 - Ha az első alagútbejárat került lerakásra, akkor kiírja annak a sínnek a sorszámát, amelyikre a bejárat került.

- Ha második bejárat került lerakásra, akkor visszaadja, hogy melyik sínekre jött létre az alagút.
 - Ha már lent van 2 alagútbejárat, de szeretnénk egy harmadikat is lerakni, akkor kiírja, hogy már van lent alagút.
 - Ha már lent van 2 alagútbejárat, és valamelyikre „rákattintunk”, akkor kiírja, hogy amelyiket felszedtük, melyik sorszámú sínen volt.
 - Ha már lent van 1 alagútbejárat, és erre „rákattintunk”, akkor kiírja, hogy már nincs lent a pályán alagútbejárat, illetve hogy hányas sorszámú sínről szedtük fel.
- **generateTrain**
Kiírja, hogy milyen hosszú és színű vagonok generálódtak.

7.2 Összes részletes use-case

Use-case neve	setInputFile
Rövid leírás	A bemeneti file elérésének meghatározása.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó megadja, hogy fileből vagy konzolból akar utasításokat adni. Ha fileből, megadja az elérési útvonalat.

Use-case neve	setOutputFile
Rövid leírás	A kimeneti file elérésének meghatározása.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó meghatározza, hogy fileba, vagy konzolra akarja a kimenetet.

Use-case neve	loadMap
Rövid leírás	A kívánt pálya betöltése.
Aktorok	Felhasználó
Forgatókönyv	A program a kívánt pályát felépíti.

Use-case neve	nextStep
Rövid leírás	Az összes vonat léptetése.
Aktorok	Felhasználó
Forgatókönyv	A pályán lévő összes vonat következő sínre mozgatása. Ütközésetektálás vizsgálata. Pálya befejezéséhez való kritérium teljesítésének vizsgálata.

Use-case neve	click
Rövid leírás	Sínekre kattintás.
Aktorok	Felhasználó
Forgatókönyv	A pályán egy sorszámozott sínre való kattintás. Ha a sín sima sín (NormalRail), alagút épülhet rajta (vagy rombolódhat), ha váltó (SwitchRail), a váltó a másik állásba áll.

Use-case neve	generateTrain
Rövid leírás	Vonat(ok) generálása.
Aktorok	Felhasználó
Forgatókönyv	A parancsban megadott hosszú, és színű vonat generálása a pálya bemenetére.

7.3 Tesztelési terv

Teszt-eset neve	Switch test
Rövid leírás	Váltók tesztelése.
Teszt célja	A váltók egyesével való állításának tesztelése.

Teszt-eset neve	Station test1
Rövid leírás	Állomások tesztelése.
Teszt célja	Utasok állomáson leszállításának tesztelése.

Teszt-eset neve	Station test2
Rövid leírás	Állomások tesztelése.
Teszt célja	Utasok állomáson felszállításának tesztelése.

Teszt-eset neve	Coal Wagon test1
Rövid leírás	Szenesvagon tesztelése.
Teszt célja	Utasok állomáson leszállításának tesztelése, úgy, hogy szeneskocsi is van a szerelvényen.

Teszt-eset neve	Coal Wagon test2
Rövid leírás	Szenesvagon tesztelése.
Teszt célja	Utasok állomáson felszállításának tesztelése, úgy, hogy szeneskocsi is van a szerelvényen.

Teszt-eset neve	Cross test
Rövid leírás	Keresztirányú sínek tesztelése.
Teszt célja	Keresztirányú sínen való mozgás tesztelése.

Teszt-eset neve	Tunnel test
Rövid leírás	Alagút lerakásnál, felvevésének tesztelése.
Teszt célja	A pályán különböző sínekre „kattintva” próbál lerakni alagutat, majd felvenni.

Teszt-eset neve	Train collision test
Rövid leírás	Két vonat pályán való összeütközésének tesztelése.
Teszt célja	Amikor két vonat a pályán összeütközik, azok felrobbannak.

Teszt-eset neve	Train out of map test
Rövid leírás	Vonat fallal ütközésének tesztelése.
Teszt célja	Amikor egy vonat kimegy a pályáról, felrobban.

Teszt-eset neve	Win stage test
Rövid leírás	A szint megnyerésének a tesztelése.
Teszt célja	Amikor az összes vonat leszállította az összes utasát, a játékos megnyerte a játékos.

Teszt-eset neve	Tick test
Rövid leírás	A szimuláció tesztelése.
Teszt célja	Egy pályán automatikusan a játékos beleszólása nélkül végigmegy a vonat, nem kell next-et írni.

7.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása

A tesztelés kiértékelését a proton belüli kis program fogja elvégezni. Ez a program tárolni fogja a be- és kimeneteket összepárosítva. Lefuttatja a tesztet az adott bementre, majd összehasonlítja az elvárt kimenettel. Az összehasonlítás szövegesen történik. Ha a kapott eredmény egyezik az elvárt kimenettel, akkor a teszt sikerült, ha eltérés van, akkor a teszt sikertelen volt.

7.5 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.03.27. 10:00	2 óra	Bartók Burom Koncsik Németh Pekk	Tevékenység: Változtatások megbeszélése, feladatok kiosztása.
2017.03.27. 12:00	1 óra	Németh	Tevékenység: Új osztályok leírása
2017.03.27. 13:00	1 óra	Németh	Tevékenység: Módosított osztályok leírása
2017.03.27. 12:00	30 perc	Koncsik	Tevékenység: Class diagram módosítása
2017.03.27. 12:30	1.5 óra	Koncsik	Tevékenység: Szekvencia diagramok módosítása
2017.03.27. 12:00	1 óra	Pekk	Tevékenység: Kimeneti nyelv specifikációja
2017.03.27. 12:00	1 óra	Bartók	Tevékenység: Use- case-ek leírása
2017.03.27. 18:00	2 óra	Bartók Burom Koncsik Németh Pekk	Tevékenység: Tesztelési terv leírása.

8. Részletes tervek

8.1 Osztályok és metódusok tervei.

8.1.1 CollisionException

- **Felelősség**

Saját kivétel osztály. Akkor dobódik, amikor egy vonat ütközik egy másik vonattal, felvág egy váltót, vagy kimegy a pályáról. Ilyen kivételt a Rail leszármazottjai dobhatnak. A Stageben van feldolgozva az összes ilyen kivétel.

- **Ősosztályok**

Exception

- **Metódusok**

- **+CollisionException(String message):** Létrehozza a kivételt a paraméterben kapott stringgel.

8.1.2 CoalWagon

- **Felelősség**

A szeneskocsi (CoalWagon), egy olyan vagon, amiben nem utazhatnak utasok, és a színe fekete. A vonaton bárhol elhelyezkedhet.

- **Ősosztályok**

Train → Wagon

- **Metódusok**

- **+CoalWagon(Rail where):** Konstruktor, meghívja az őssosztály konstruktorát, fekete színnel, illetve a paraméterként kapott pozícióval.
- **+void empty():** A szeneskocsival nem csinál semmit, mivel itt nincsenek utasok.
- **+void fill():** A szeneskocsival nem csinál semmit, mivel itt nincsenek utasok.
- **+Color getColor():** Getter függvény, a vagon színét adja vissza, ami jelen esetben a fekete.
- **+boolean isEmpty():** Getter függvény, visszaadja, hogy üres-e a vagon, ami jelen esetben mindig igaz lesz, mivel itt nincsenek utasok.

8.1.3 Color

- **Felelősség**

Az állomások, vagonok lehetséges színe.

- **Attribútumok**

- **+RED**
- **+GREEN**
- **+BLUE**
- **+YELLOW**
- **+ORANGE**

8.1.4 CrossRail

- **Felelősség**

A keresztirányú síneken (CrossRail) a vonatok azonos szintben keresztezhetik egymás útját. 2 előző és 2 következő sínje van. A dolga a mozdony következő sínre való mozgatása. Ehhez overrideolja az örökölt moveToNext() függvényt, úgy hogy megjegyzi, hogy az adott szerelvény melyik előző sínjéről érkezett, és az annak megfelelő következőre fogja tovább léptetni.

- **Össztályok**

Rail

- **Attribútumok**

- **-Rail otherNext:** [Az alternatív következő sín.](#)
- **-Rail otherPrev:** Az alternatív előző sín.

Metódusok

- **+boolean canPlaceTunnel():** False-szal tér vissza, hiszen a CrossRail-re sosem lehet alagutat (Tunnel) építeni.
- **+void checkNeighbors():** Hasonlóképp működik, mint a váltó (SwitchRail) checkNeighbors metódusa. Minden tick végén megnézi a CrossRail, hogy a közvetlen szomszédjában van-e mozdony, és aszerint állítja be az aktív előző és/vagy következő sínjét.
- **+void moveToNext():** Az aktuális sínen álló mozdonyt a következő sínre mozgatja.

8.1.5 Direction

- **Felelősség**

A mozdonyok (Locomotive) váltótól (Switch) váltóig egy adott Direction szerint mozognak, amik az alábbiak lehetnek. Ha a mozdony rálép egy váltóra, a váltó átadja neki az új Directiont. A következő váltóig e szerint a Direction szerint fog mozogni.

- **Attribútumok**

- **+NEXT:** A sínek next-jére kell lépjen. (A váltó főágán, vagy az előre kapcsolt mellékágon lépked.)
- **+PRE:** Ha váltón áll, akkor még egyszer a next-re kell lépni, de az utána következő sínek previous-án folytatja. (A visszakapcsolt mellékágon lépked.)
- **+PREPRE:** A sínek previous-ára kell lépjen. (A váltó főágán visszafele lépked.)

8.1.6 Game

- **Felelősség**

A szimuláció legmagasabb szintje. Itt tud beleavatkozni a játékos a folyamatokba a váltó (Switch) állításával, alagút (Tunnel) építésével.

- **Metódusok**

- **+static void main(String[] args):** A játék futtatása.

8.1.7 Locomotive

- **Felelősség**

A mozdony fogja össze a vonatot, hiszen ő kezeli a hozzá tartozó vagonokat. A mozdony csak mozogni tud, azt, hogy hogyan mozog, azt a sín (Rail) intézi.

- **Ősosztályok**

Train

- **Attribútumok**

- **-Direction dir**: A mozdony haladási iránya, ezt a sínektől kapja.
- **-List<Wagon> wagons**: A mozdony mögé csatolt vagonok (Wagon) listája.

- **Metódusok**

- **+void addWagon(Wagon wagon)**: Hozzáad egy vagon a mozdony vagonjainak listájának a végéhez.
- **+Direction getDir()**: Getter függvény, visszaadja a mozdony mozgási irányát.
- **+List<Wagon>getWagons()**: Getter függvény, visszaadja a mozdony mögé csatolt vagonok listáját.
- **+Locomotive(Rail where)**: Konstruktor, lehelyezi a mozdonyt az adott sínre.
- **+void nextStep()**: A mozdony léptetése. Meghívja annak a sínnek a moveToNext függvényét, amelyiken éppen áll.
- **+void setDir(Direction dir)**: Setter függvény, beállítja a mozdony új mozgási irányát.

8.1.8 NormalRail

- **Felelősség**

A sínnek a legáltalánosabb leszármazottja. A dolga csak a mozdony következő sínre való mozgatása. Ehhez overrideolja az örökölt moveToNext() függvényt. Ezen kívül csak erre a sínre lehet építeni alagutat (Tunnel).

- **Ősosztályok**

Rail

- **Metódusok**

- **+boolean canPlaceTunnel()**: True-val tér vissza, hiszen a NormalRail-re mindig lehet építeni alagutat (Tunnel).
- **+void moveToNext()**: Az aktuális sínen álló mozdonyt a következő sínre mozgatja.

8.1.9 Rail

- **Felelősség**

Minden sínnek az ősosztálya. Ez az osztály biztosítja, hogy vonatot (Train) lehessen rá helyezni, valamint biztosít gettert, settert a fontosabb attribútumokhoz. Az osztály absztrakt, hiszen néhány függvénye sínspecifikus. Tárolja továbbá, hogy van-e rajta mozdony (Locomotive), a váltó (SwichRail) checkNeighbors() függvénye miatt.

- **Attribútumok**

- **#boolean onrail**: True, ha tartózkodik az adott időpillanatban a sínen mozdony, false ha nem.
- **#Rail next**: A következő sín.
- **#Rail prev**: Az előző sín.

- **Metódusok**

- **+abstract boolean canPlaceTunnel()**: Megmondja, hogy lehet-e alagutat (Tunnel) építeni a sínre.
- **#void collisionDetection(Locomotive loco)**: Ütközés detektálás. Miután megtörtént a lépés, megnézi, hogy ahova lépett, ott van-e mozdony. CollisionException-t dob ha „ütközés” történt.
- **+boolean getLocomotive()**: Getter függvény, visszaadja, hogy tartózkodik-e mozdony a sínen.
- **+Rail getNext()**: Getter függvény, visszaadja a következő sínt.
- **+Rail getPrevious()**: Getter függvény, visszaadja az előző sínt.
- **#void moveLocomotive(Locomotive loco)**: Mozdonyok léptetése. A paraméterben kapott mozdonyt a következő sínre lépteti. CollisionException-t dob ha „ütközés” történt.
- **+abstract void moveToNext(Locomotive loco)**: Az aktuális sínen álló mozdonyt a következő sínre mozgatja, valamint ha a sínnek még van dolga a paraméterben kapott Locomotive-val, azt elvégzi.
- **#void moveWagons(Locomotive loco)**: Vagonok léptetése. A paraméterben kapott mozdony mögé csatolt vagonok következő sínre léptetése. A moveLocomotive függvénnyel egy helyen, egymás után kerül meghívásra.
- **+Rail()**: Paraméter nélküli konstruktor, létrehoz egy sínt.
- **+void set(Rail previous, Rail next)**: Setter függvény, beállítja az előző, és a következő sínt.
- **+void setLocomotive(boolean onrail)**: Setter függvény, ami az adott sínre beállítja, hogy van-e rajta mozdony (Locomotive).
- **+void setNext(Rail next)**: Setter függvény, beállítja a következő sínt.
- **+void setPrevious(Rail previous)**: Setter függvény, beállítja az előző sínt.

8.1.10 Stage

- **Felelősség**

Felépíti a pályát, lehetőséget biztosít a szimulálásra a metódusai segítségével. Valamint a pálya játékos általi manipulálása ebben a classban van megvalósítva.

- **Attribútumok**

- **-List<Locomotive> locomotives:** A pályán lévő vonatok.
- **-List<Rail> rails:** A pályán lévő sínek.
- **-List<Station> stations:** A pályán lévő állomások.
- **-List<Switch> switches:** A pályán lévő váltók.
- **-Rail[][] tiles:** A pályán lévő elemek/sínek. A rails listből áll, csak más elrendezésben a játékos kényelme érdekében.
- **-Tunnel tunnel:** A pályán lévő alagút.

- **Metódusok**

- **+generateTrain(int length, Rail where):** Generál egy vonatot, csak olyan színű vagonokkal, amilyen állomások aktívak, és lerakja a paraméterként kapott sínre. A hossz paraméter a vonat mozdony nélküli hossza (vagyis a vagonok száma).
- **-Color generateWagonColor():** A vagonok színét generálja. Csak olyan színt generálhat, amilyen színű aktív állomás van.
- **+void init():** Létrehozza, felépíti a pályát.
- **+boolean modifyTiles(int x, int y):** A paraméterben kapott x és y paraméter szerint kikeresi a pályán van-e sín, és ha van, akkor az milyen típusú. Ennek megfelelően, ha Switch, akkor vált, ha sima sín, akkor alagutat épít rá, valamint rombol le. Visszaadja, hogy sikeres volt-e a manipuláció.
- **+void tick():** Az összes Stage-hez tartozó Locomotive-ot lépteti, valamint az összes váltó felméri, hogy a környezetében van-e Locomotive, és aszerint állítja be a saját Direction enumját.

8.1.11 StationRail

- **Felelősség**

Az állomások (StationRail), ahol a vagonokról (Wagon) le- illetve felszállhatnak utasok. Leszállás akkor történhet meg, ha az első nemüres vagon színe megegyezik az állomás színével. Felszállás akkor történhet meg, ha nem történt leszállás, valamint a vagon színe megegyezik az állomás színével. Ezen kívül lépteti a vonatot, mint a többi sín. Egy állomás lehet nyitva, vagy zárva, ha zárva van, nem lehet kiüríteni a vagonokat.

- **Össztályok**

Rail

- **Attribútumok**

- **-Color color:** Az állomás színe.
- **-boolean open:** Az állomás állapota (nyitott, zárt).

- **Metódusok**

- **+boolean canPlaceTunnel():** False-szal tér vissza, hiszen StationRail-re sosem lehet alagutat (Tunnel) építeni.
- **+Color getColor():** Getter függvény, visszaadja az állomás színét.
- **+boolean getUsable():** Getter függvény, visszaadja, hogy nyitva van-e az állomás.
- **+void moveToNext(Locomotive loco):** A játékszabályok szerint kiüríti a paraméterben kapott mozdonyhoz tartozó valamelyik vagon. Az aktuális sínen álló mozdonyt a következő sínre mozgatja.
- **#void passengerDelivery(Locomotive loco):** Az utasok leszállítása a paraméterben kapott vagonról.
- **+void setUsable(boolean usable):** Setter függvény, beállítja, hogy használható-e az állomás.
- **+StationRail(Color color, boolean open):** Konstruktor, ami beállítja az állomás színét, és hogy használatban van-e.

8.1.12 SwitchRail

- **Felelősség**

A váltó (SwitchRail) tárol egy Direction enumot, melyet a szomszédjaiban lévő mozdonyok (Locomotive) állásából dönt el. Ezt a Directiont adja át a rálépő mozdonynak. A mozdony eszerint a Direction szerint fog lépni a következő váltóig. A váltó továbbá nem kettő sínt köt össze, mint a többi sín, hanem hármat. Van egy főág-beli sín, valamint kettő mellékág-beli sín. A mellékág-beli sínek közül lehet a váltás funkció segítségével kiválasztani, hogy melyik legyen aktív.

- **Össztályok**

Rail

- **Attribútumok**

- **-Rail active:** A váltó aktív állása, amerre a vonatok tovább tudnak közlekedni.
- **-Direction dir:** A mozgási irány amit a sín átad a rajta levő mozdonynak.
- **-Rail othernext:** A váltó másik állása, az adott időpillanatban ebbe az irányba nem tudnak továbbhaladni, illetve az erről érkező vonatok kisiklanak.

- **Metódusok**

- **+boolean canPlaceTunnel():** False-szal tér vissza, hiszen SwitchRail-re sosem lehet alagutat (Tunnel) építeni.
- **+void checkNeighbors():** Megnézi, hogy a váltó szomszéd-sínein tartózkodik-e vonat.
- **+Rail getNext():** Visszaadja az aktív sínt.
- **#void moveLocomotive(Locomotive loco):** A paraméterként kapott mozdony következő sínre léptetése.
- **+void moveToNext(Locomotive loco):** Átadja a paraméterben kapott mozdonynak az új Directiont. Az aktuális sínen álló mozdonyt a következő sínre mozgatja.
- **+void setOther(Rail othernext, Rail active):** Beállítja a váltó másik állásának megfelelő sínt, illetve hogy melyik az aktív pillanatnyilag.
- **+boolean setSwitch():** Megvizsgálja, hogy van-e rajta vonat. Ha van rajta, akkor nem lehet váltani, ha nincs rajta, akkor átváltja a másik állásba. Visszatér, hogy sikerült-e a váltás.

8.1.13 Train

- **Felelősség**

A sínen járó eszközöket megvalósító absztrakt osztály.

- **Attribútumok**

- **#Rail rail:** A sínen járó eszköz tartózkodási helye.

- **Metódusok**

- **+Rail getRail():** Getter függvény, visszaadja melyik sínen áll.
- **+void setRail(Rail rail):** Setter függvény, beállítja, hogy melyik sínen áll.
- **+Train(Rail where):** Konstruktor, lehelyezi a vonatot a kapott sínre.

8.1.14 Tunnel

- **Felelősség**

Alagút osztály. Csak NormalRail fölé helyezhető el. Egyszerre csak 1 példány lehet a pályára lehelyezve.

- **Attribútumok**

- **-NormalRail entrance:** Az alagút bejárata.
- **-NormalRail exit:** Az alagút kijárata.

- **Metódusok**

- **-List<Rail> createEntrance(Rail rail):** Megvizsgálja, hogy tehet-e le bejáratot az adott helyre. Ha tehet, lerakja a bejáratot és visszaadja a sínek listáját, melyen alagút van.
- **-List<Rail> createExit(Rail rail):** Megvizsgálja, hogy tehet-e le kijáratot az adott helyre. Ha tehet, lerakja a kijáratot és visszaadja a sínek listáját, melyen alagút van.
- **-List<Rail> deleteTunnel(Rail rail):** Kitörli a paraméterül kapott ki/bejáratot, visszaadja az alagútban lévő maradék síneket.
- **+List<Rail> modifyTunnel(Rail rail):** Ha a paraméterben kapott sínre (Rail) már van építve bejárat, akkor azt törli. Ha még egy bejárat sincs lerakva, akkor lerakja a sínre. Ha már van egy bejárat lerakva, megvizsgálja, hogy a lerakott bejárat és a paraméterben kapott sín között lehet-e létrehozni alagutat, ha lehet, létrehozza. Visszatér a manipulált sínek listájával (kirajzoláshoz hasznos lesz).

8.1.15 Wagon

- **Felelősség**

A vagonokban (Wagon) utasok, vagy szén helyezkednek el. Az utasokat kell eljuttatni az állomásokra (StationRail). A játékszabályok szerint lehet kiüríteni, feltölteni a vagonokat.

- **Össztályok**

Train

- **Attribútumok**

- **-Color color:** A vagon színe.
- **-boolean isEmpty:** Van-e a vagonban utas.

- **Metódusok**






- **+void empty():** Kiüríti a vagon.
- **+void fill():** 0.5-ös valószínűségi tényezővel feltölti a vagon, vagy nem.
- **+Color getColor():** Getter függvény, visszaadja a vagon színét.
- **+boolean isEmpty():** Getter függvény, visszaadja, hogy üres-e a vagon.
- **+Wagon(Rail where, Color color):** Konstruktor, beállítja a vagon pozícióját, illetve színét.

8.2 A tesztek részletes tervei, leírásuk a teszt nyelvén

8.2.1 Tesztpályák

Minden négyzet egy sín darabot jelöl. Ezek a sín darabok a játékban résztvevő: normál sín, kereszt-sín, váltó, állomás lehetnek. Minden sínnek van egy azonosítója, mely egy egész szám, így lehet hivatkozni rájuk a bemeneti nyelv utasításaiban. A váltóknak mindig a legkisebb sorszámu sín a főáguk, valamint a második legkisebb sorszámu sín fele állnak alapértelmezetten. Tehát a Swtich pályánál a 3as váltó főága (ami a fix pont) a 2-es sín, és alapértelmezetten a 4-es sín fele áll. Gombnyomásra tudjuk a 4-es és a 9-es sín között váltani.

Kis változtatás történt az 7. dokumentum bemeneti nyelv részében: Ha a load utasítás paramétere station, meg kell adni sorban a két állomás színét is. Pl: load station b y.

	Bemenet (ide generálódnak a vonatok)
	Normál sín (Normal rail)
	Kereszt-sín (Cross rail)
	Váltó (Switch rail)
	Állomás (Station rail)

8.2.1.1 Normal

	0	1	2	3	4
---	---	---	---	---	---

8.2.1.2 Cross

			9	8	7
			10		6
0	1	2	3	4	5
			11		
			12		

8.2.1.3 Switch

			8		
			7		
			6	14	13
			5		12
			4		11
0	1	2	3	9	10

8.2.1.4 Station

	0	1	2	3		4	5		6	7	8
---	---	---	---	---	---	---	---	---	---	---	---

8.2.2 Switch test

- **Leírás**

Váltók tesztelése. A váltók egyesével való állításának tesztelése. Kattintásra vált-e a váltó.

- **Ellenőrzött funkcionális, várható hibahelyek**

Váltók váltása

- **Bemenet**
 - load switch
 - click 3
 - click 6
- **Elvárt kimenet**
 - map switch loaded
 - switch 3 switched from 4 to 9
 - switch 6 switched from 7 to 14

8.2.3 Station test1

- **Leírás**

Állomások tesztelése. Utasok állomáson leszállításának tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Utasok leszállítása. Állomáson történő lépés. Lehetséges hiba, hogy nem szállnak le az utasok az első állomásnál, vagy leszállnak a másodikonál is.

- **Bemenet**
 - load station b r
 - gentrain 2 b y
 - next
 - next
 - next
 - next
 - next
 - next
 - next
- **Elvárt kimenet**
 - map station loaded
 - train generated name train1 length 2 b y
 - train1 moved from 0 to 1
 - train1 moved from 1 to 2
 - train1 moved from 2 to 3
 - train1 moved from 3 to 4
 - train1 moved from 4 to 5 blue passengers got off
 - train1 moved from 5 to 6
 - train1 moved from 6 to 7

8.2.4 Station test2

- **Leírás**

Állomások tesztelése. Utasok állomáson felszállításának tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Utasok leszállítása, majd felszállítása. Állomáson történő lépés. Lehetséges hiba, hogy nem szállnak vissza a vonatra a második állomásnál, amikor már üres a vagon.

- **Bemenet**

```
load station b b
gentrain 2 b y
next
next
next
next
next
next
next
next
```

- **Elvárt kimenet**

```
map station loaded
train generated name train1 length 2 b y
train1 moved from 0 to 1
train1 moved from 1 to 2
train1 moved from 2 to 3
train1 moved from 3 to 4
train1 moved from 4 to 5 blue passengers got off
train1 moved from 5 to 6
train1 moved from 6 to 7 blue passengers got on
```


8.2.5 Coal Wagon test1

- **Leírás**

Szenesvagon tesztelése. Utasok állomáson leszállításának tesztelése, úgy, hogy szeneskocsi is van a szerelvényen.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Utasok leszállítása úgy, hogy az első vagon szeneskocsi. Lehetséges hiba, hogy nem szállnak le az utasok, mert kilép az iteráció a szeneskocsi miatt.

- **Bemenet**

```
load station b r  
gentrain 3 n b y  
next  
next  
next  
next  
next
```

- **Elvárt kimenet**

```
map station loaded  
train generated name train1 length 3 n b y  
train1 moved from 0 to 1  
train1 moved from 1 to 2  
train1 moved from 2 to 3  
train1 moved from 3 to 4  
train1 moved from 4 to 5 blue passengers got off
```

8.2.6 Coal Wagon test2

- **Leírás**

Szenesvagon tesztelése. Utasok állomáson felszállításának tesztelése, úgy, hogy szeneskocsi is van a szerelvényen. Lehetséges hiba, hogy nem szállnak fel az utasok, mert kilép az iteráció a szeneskocsi miatt.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Utasok le- majd felszállítása úgy, hogy az első vagon szeneskocsi.

- **Bemenet**

```
load station b b  
gentrain 3 n b y  
next  
next  
next  
next  
next  
next  
next  
next
```

- **Elvárt kimenet**

```
map station loaded  
train generated name train1 length 3 n b y  
train1 moved from 0 to 1  
train1 moved from 1 to 2  
train1 moved from 2 to 3  
train1 moved from 3 to 4  
train1 moved from 4 to 5 blue passengers got off  
train1 moved from 5 to 6  
train1 moved from 6 to 7 blue passengers got on
```

8.2.7 Cross test

- **Leírás**

Keresztirányú sínek, rajtuk való mozgás tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A kereszt irányú sínen mindkét irányba történő áthaladás. Lehetséges hiba, hogy a vonat nem egyenesen halad tovább, hanem elkanyarodik.

- **Bemenet**

```
load cross
gentrain 1 b
next
next
next
next
next
next
next
next
next
next
next
next
next
next
next
```

- **Elvárt kimenet**

```
map cross loaded
train generated name train1 length 1 b
train1 moved from 0 to 1
train1 moved from 1 to 2
train1 moved from 2 to 3
train1 moved from 3 to 4
train1 moved from 4 to 5
train1 moved from 5 to 6
train1 moved from 6 to 7
train1 moved from 7 to 8
train1 moved from 8 to 9
train1 moved from 9 to 10
train1 moved from 10 to 3
train1 moved from 3 to 11
```

8.2.8 Tunnel test

- **Leírás**

Alagút lerakásnál, felvevésének tesztelése. A pályán különböző sínekre „kattintva” próbál lerakni alagutat, majd felvenni.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Alagút bejárat lerakása, felvétele. Lehetséges hiba, hogy nem engedi lerakni olyan helyre az alagutat, ahova egyébként lehetne, vagy több alagutat is le tudunk rakni.

- **Bemenet**

- load normal

- click 1

- click 3

- click 4

- click 3

- click 1

- **Elvárt kimenet**

- first tunnel entrance put down to rail 1

- second tunnel entrance put down to rail 3, tunnel created

- cannot put down more tunnel entrances

- second tunnel entrance destroyed on rail 3, tunnel destroyed

- first tunnel entrance destroyed on rail 1

8.2.9 Train collision test

- **Leírás**

Két vonat pályán való összeütközésének tesztelése. Amikor két vonat a pályán összeütközik, azok felrobbannak.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ütközés tesztelése. Lehetséges hiba, hogy a két vonat átmegy egymáson.

- **Bemenet**

```
load cross
gentrain 1 b
next
next
next
next
next
next
next
next
next
next
gentrain 1 y
next
next
next
```

- **Elvárt kimenet**

```
map cross loaded
train generated name train1 length 1 b
train1 moved from 0 to 1
train1 moved from 1 to 2
train1 moved from 2 to 3
train1 moved from 3 to 4
train1 moved from 4 to 5
train1 moved from 5 to 6
train1 moved from 6 to 7
train1 moved from 7 to 8
train generated name train2 length 1 y
train1 moved from 8 to 9
train2 moved from 0 to 1
train1 moved from 9 to 10
train2 moved from 1 to 2
train1 moved from 10 to 3
train2 moved from 2 to 3 crashed with other train
```

8.2.10 Train out of map test

- **Leírás**

Vonat fallal ütközésének tesztelése. Amikor egy vonat kimegy a pályáról, felrobban.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ütközés tesztelése fallal. Lehetséges hiba, hogy a exceptiont kapunk, mert a pálya szélén a következő sín null.

- **Bemenet**

```
load normal
gentrain 1 b
next
next
next
next
next
next
```

- **Elvárt kimenet**

```
map normal loaded
train generated name train1 length 1 b
train1 moved from 0 to 1
train1 moved from 1 to 2
train1 moved from 2 to 3
train1 moved from 3 to 4
train1 moved from 4 to null crashed on map edge
```

8.2.11 Win stage test

- **Leírás**

A szint megnyerésének a tesztelése. Amikor az összes vonat leszállította az összes utasát, a játékos megnyerte a játékot.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Játék megnyerésének lehetősége. Lehetséges hiba, hogy nem lesz vége a játéknak.

- **Bemenet**

```
load station
gentrain 1 b
next
next
next
next
next
next
```

- **Elvárt kimenet**

```
map station loaded
train generated name train1 length 1 b
train1 moved from 0 to 1
train1 moved from 1 to 2
train1 moved from 2 to 3
train1 moved from 3 to 4
train1 moved from 4 to 5 blue passengers got off win
```

8.2.12 Tick test

- **Leírás**

A szimuláció tesztelése. Egy pályán automatikusan a játékos beleszólása nélkül végigmegy a vonat, nem kell next-et írni.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Lehetséges hiba, hogy beakad a szimuláció.

- **Bemenet**

- load normal
 - gentrain 1 b

- **Elvárt kimenet**

- map normal loaded
 - train generated name train1 length 1 b
 - train1 moved from 0 to 1
 - train1 moved from 1 to 2
 - train1 moved from 2 to 3
 - train1 moved from 3 to 4
 - train1 moved from 4 to null crashed on map edge

8.3 A tesztelést támogató programok tervei

A specifikált teszteseteket ellátjuk elvárt kimenetekkel. A tesztelést végző segédprogram megmondja, hogy a specifikált tesztesetek sikeresen zajlottak-e le. Tesztelni lehet nem előre specifikált tesztesetekkel is, hanem a bemeneti nyelv utasításait felhasználva, ekkor viszont nem tudunk információt szolgáltatni arról, hogy a teszteset sikeres volt-e. A specifikált tesztesetek összehasonlítása karakterről karakterre történik, ha egy karakter eltér, a teszteset elbukott, ha megegyezik minden karakter, a teszteset sikeres volt.

8.4 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017. 04. 03. 16:30	30 perc	Bartók Burom Koncsik Németh Pekk	Értekezlet. Döntés: Bartók, Burom, Pekk készítik el a tesztesetek részletes leírását. Koncsik, Németh készítik el az osztályok részletes leírását és a tesztelést támogató program tervét.
2017. 04. 03. 17:00	30 perc	Koncsik	Tevékenység: Tesztpályák létrehozása
2017. 04. 03. 17:30	2 óra	Bartók Burom Pekk	Tevékenység: Bartók, Burom, Pekk elkészíti a tesztesetek részletes leírását.
2017. 04. 03. 17:30	2 óra	Németh	Tevékenység: Németh elkészíti az osztályok részletes leírását.
2017. 04. 03. 21:00	1,5 óra	Koncsik	Tevékenység: Koncsik kiegészíti az osztályok leírását és elkészíti a tesztelést támogató program tervét.
2017. 04. 03. 22:30	30 perc	Koncsik	Tevékenység: Koncsik kiegészíti a teszteseteket a lehetséges hibákkal.

10. Prototípus beadása

10.1 Fordítási és futtatási útmutató

10.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
src			
logs.log	0 byte	2017.04.16.	Alapértelmezett kimeneti fájl.
maps			
cross.dat	934 byte	2017.04.16.	Cross pálya
normal.dat	675 byte	2017.04.16.	Normal pálya
station.dat	900 byte	2017.04.16.	Station pálya
switch.dat	968 byte	2017.04.16.	Switch pálya
tests			
coal_wagon_test_1.test	294 byte	2017.04.16.	Szenes vagon 1. teszt
coal_wagon_test_2.test	381 byte	2017.04.16.	Szenes vagon 2. teszt
collision_test.test	605 byte	2017.04.16.	Ütközés teszt
cross_test.test	481 byte	2017.04.16.	CrossRail teszt
out_of_map_test.test	277 byte	2017.04.16.	Pálya elhagyása teszt
station_test_1.test	354 byte	2017.04.16.	Állomás 1. teszt
station_test_2.test	375 byte	2017.04.16.	Állomás 2. teszt
switch_test.test	121 byte	2017.04.16.	Váltó teszt
tick_test.test	258 byte	2017.04.16.	Következő időpillanat teszt
tunnel_test.test	202 byte	2017.04.16.	Alagút teszt
win_test.test	290 byte	2017.04.16.	Pálya teljesítése teszt
game			
Game.java	18650 byte	2017.04.17.	A szimuláció legmagasabb szintje. Itt tud beleavatkozni a játékos a folyamatokba a váltó állításával, alagút építésével.
others			
CollisionException.java	550 byte	2017.04.17.	Saját kivétel osztály.
Color.java	147 byte	2017.04.17.	Az állomások, vagonok lehetséges színe.
Direction.java	365 byte	2017.04.17.	A mozdonyok mozgási iránya.
rails			
CrossRail.java	3182 byte	2017.04.17.	Keresztirányú sín osztály.
NormalRail.java	1274 byte	2017.04.17.	A sínnek a legáltalánosabb leszármazottja. Vonat léptetése valósítja meg.
Rail.java	7022 byte	2017.04.17.	Minden sínnek az ősoosztálya.
StationRail.java	4246 byte	2017.04.17.	Az állomások osztálya, itt szállnak le az utasok a vagonokról.
SwitchRail.java	5343 byte	2017.04.17.	A váltó osztály.

stage			
Stage.java	14425 byte	2017.04.17.	Felépíti a pályát, lehetőséget biztosít a szimulálásra a metódusai segítségével. Valamint a pálya játékos általi manipulálása ebben a classban van megvalósítva.
Tunnel.java	6395 byte	2017.04.17.	Alagút osztály.
trains			
CoalWagon.java	810 byte	2017.04.17.	Szeneskovács osztály.
Locomotive.java	2550 byte	2017.04.17.	Mozdony osztály.
Train.java	843 byte	2017.04.17.	A sínen járó eszközök osztálya.
Wagon.java	1334 byte	2017.04.17.	A vagonokban utasok helyezkednek el, őket kell eljuttatni az állomásokra.

10.1.2 Fordítás

Fordítás menete:

6. A .zip állomány kicsomagolása
7. Parancssor indítása, a kicsomagolt fájlok helyére navigálás (src mappa)
8. `java -version` parancs kiadása, a telepített Java verzió meghatározásához
9. A java és javac programok működéséhez szükséges path beállítása:
`set path=%path%;C:\Program Files\Java\jdk1.8.0_121\bin`
A fenti parancs egy példa, ha a telepítés helye, vagy a verziószám különbözik, aszerint kell futtatni a parancsot.
10. Fordítás mind az 5 mappában a következő paranccsal:
`javac game/*.java others/*.java rails/*.java stage/*.java trains/*.java`

10.1.3 Futtatás

Futtatni az előző pontban helyesen fordított fájlokat lehetséges. Parancssorban a fordított fájlok gyökérkönyvtárába (src) navigálás után a futtatás a `java game.Game` paranccsal történik.

10.2 Tesztek jegyzőkönyvei

10.2.1 coal_wagon_test_1

Tesztelő neve	Koncsik Norbert
Teszt időpontja	2017.04.17.

10.2.2 coal_wagon_test_2

Tesztelő neve	Koncsik Norbert
Teszt időpontja	2017.04.17.

10.2.3 collision_test

Tesztelő neve	Koncsik Norbert
Teszt időpontja	2017.04.17.

10.2.4 cross_test

Tesztelő neve	Koncsik Norbert
Teszt időpontja	2017.04.17.

10.2.5 out_of_map_test

Tesztelő neve	Koncsik Norbert
Teszt időpontja	2017.04.17.

10.2.6 station_test_1

Tesztelő neve	Koncsik Norbert
Teszt időpontja	2017.04.17.

10.2.7 station_test_2

Tesztelő neve	Koncsik Norbert
Teszt időpontja	2017.04.17.

10.2.8 switch_test

Tesztelő neve	Koncsik Norbert
Teszt időpontja	2017.04.17.

10.2.9 tick_test

Tesztelő neve	Koncsik Norbert
Teszt időpontja	2017.04.17.

10.2.10 tunnel_test

Tesztelő neve	Koncsik Norbert
Teszt időpontja	2017.04.17.

10.2.11 win_test

Tesztelő neve	Koncsik Norbert
Teszt időpontja	2017.04.17.

10.3 Értékelés

Tag neve	Munka százalékban
Koncsik Norbert	33%
Németh Bence	25%
Bartók Patrik István	14%
Burom Bence	14%
Pekk János Richárd	14%

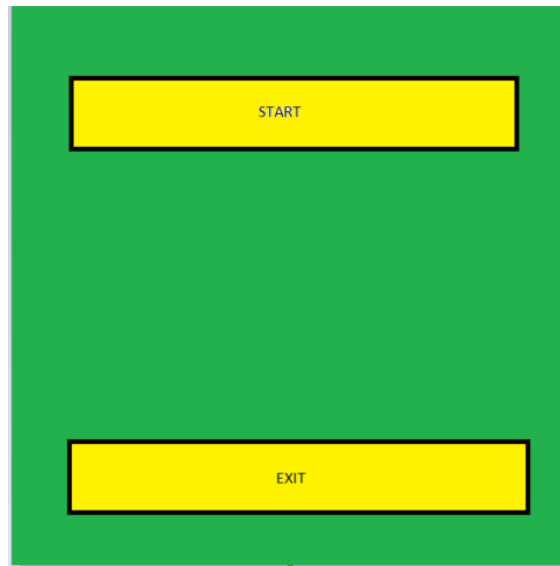
10.4 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.04.15. 15:00	10 óra	Koncsik	Tesztek, pályák elkészítése. Bemeneti parser elkészítése (Game osztály).
2017.04.16. 15:00	15 óra	Koncsik	Kimenetek formázása, függvények kiegészítése a proto működéséhez (Stage, Rail osztályok).
2017.04.17. 14:00	4 óra	Németh	Függvények kiegészítése a proto működéséhez (Train osztályok). Kód formázása, kommentek kiegészítése, tesztek újbóli elvégzése.
2017.04.17. 18:00	2 óra	Németh	Dokumentáció elkészítése

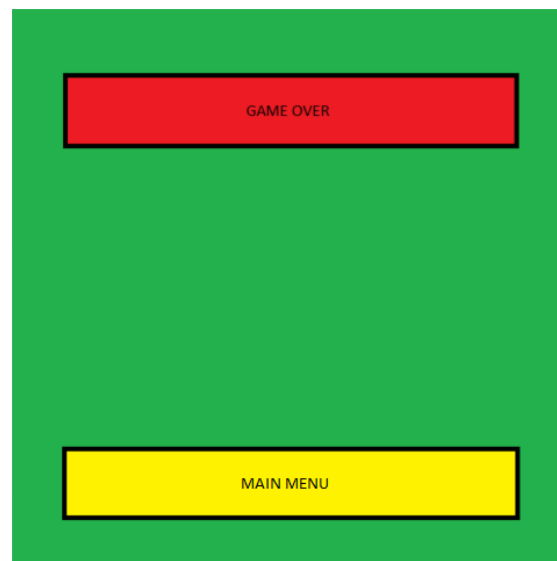
11. Grafikus felület specifikációja

11.1 A grafikus interfész

Főmenü:



Játék vége menü:



Vagonok:

vagonf



vagonk



vagonn



vagonp



vagonz



vagony



vagonü

Locomotív:

loco

Sín:**Váltó:****Állomások:**

station_green.png



station_kek.png



station_orange.png



station_red.png

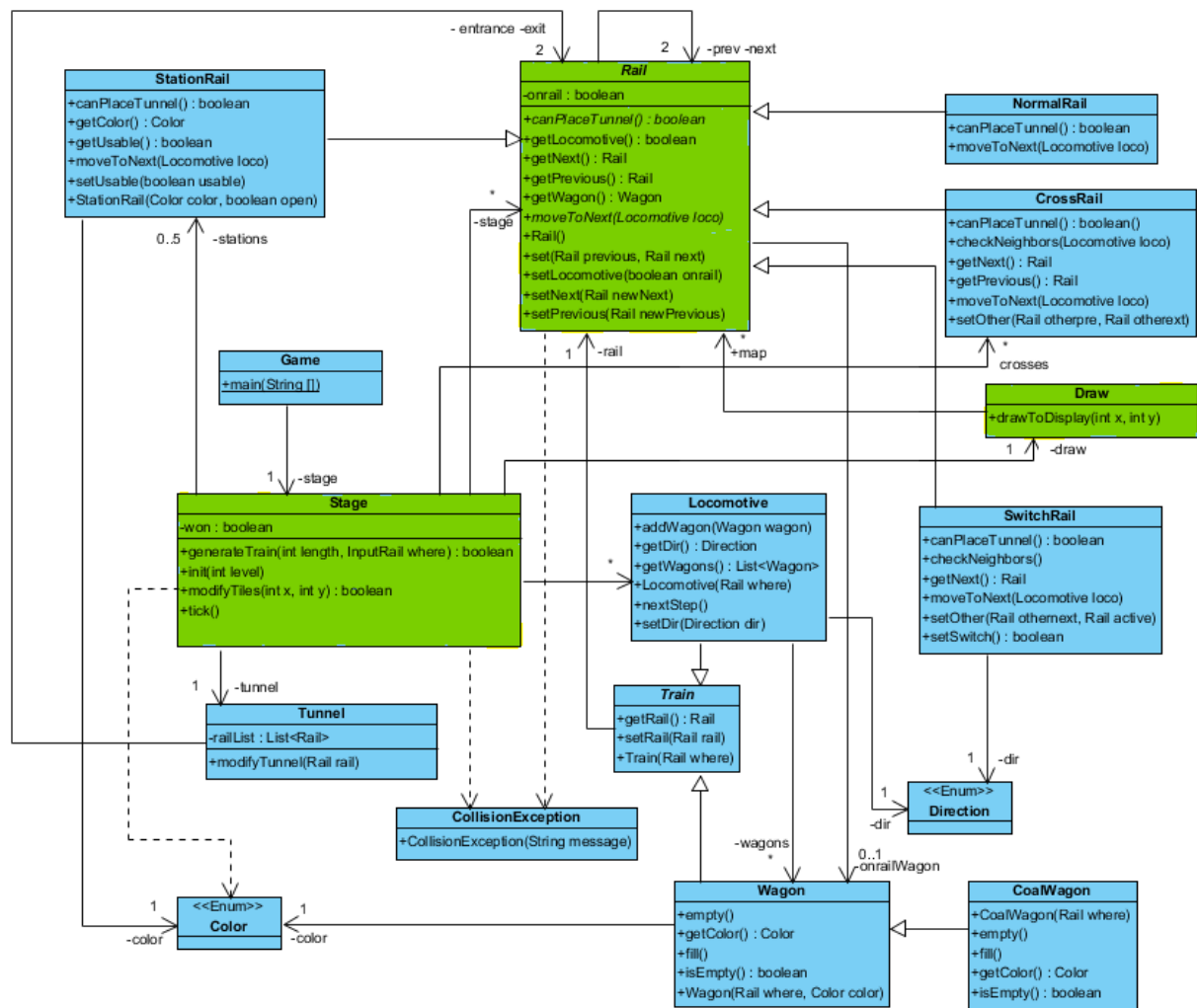


station_yellow.png

Alagút:**11.2 A grafikus rendszer architektúrája****11.2.1 A felület működési elve**

Egy darab új osztályunk lesz, ami felelős lesz a teljes grafikus megjelenítésért. Ez lesz a Draw. Képes lesz egy megfelelő képet a 6x6-os pályán a megfelelő XY koordinátában megjeleníteni. Alapelvünk a “push” lesz, mivel a modell, ha bekövetkezett egy esemény, újrarajzoltatja az egész pályát, vagyis minden lépésben újra ki kell értékelni, hogy melyik pozícióban, milyen képnek kell megjelenennie. Az újrarajzolás módszerével semmilyen rétegező módszert nem alkalmazunk, minden pozícióba azt a képet jelenítjük meg, amit a modell megkövetel, ez minden pozícióban egy darab kép megjelenítését jelenti. A grafikus felület nem lesz képes jelezni semmit a modell felé.

11.2.2 A felület osztály-struktúrája



Zöld színnel van megjelölve az új rész a diagramban. A Rail és a Stage kapcsolódik szorosan az új Draw osztályhoz, viszont a prototípus implementálása közben változások történtek. Ezek a változások a kódban jelölve vannak. Az osztálydiagram tartalmazza a változtatásokat.

11.3 A grafikus objektumok felsorolása

11.3.1 Draw

- **Felelősség**

Grafikus megjelenítésért felelős, tárolja a pályamátrixot, ami alapján megtudja határozni, mit kell kirajzolni az adott pozícióba.

- **Attribútumok**

- **Public Rail map[6][6]:** A sínek geometriai elrendezését tárolja.

- **Metódusok**

- **Public void drawToDisplay(int x, int y):**

Az x-y koordinátába kirajzolja a megfelelő képet. Prioritás szerint rajzol az eltárolt map függvényében. Legelső prioritás a barlang, ha valamelyik pozícióban az van, akkor a barlangot kell mindenképpen megjeleníteni. Következő a prioritási sorban a Rail. A legutolsó a fű, maga a háttér, ha nincsen az adott pozícióban sem barlang, sem rail, akkor fű lesz kirajzolva. A Rail tudja magáról, hogy milyen objektum helyezkedik el rajta, valamint, hogy annak az objektumnak milyen színe van, így a metódus ezeket vizsgálva képes lesz a megfelelő képet kiválasztani és megjeleníteni.

11.3.2 Rail

- **Változás:**

Referenciát tárolnak arról a Wagon objektumról, ami éppen rajta van, eddig csak azt tárolta, hogy van-e rajta, vagy nincs.

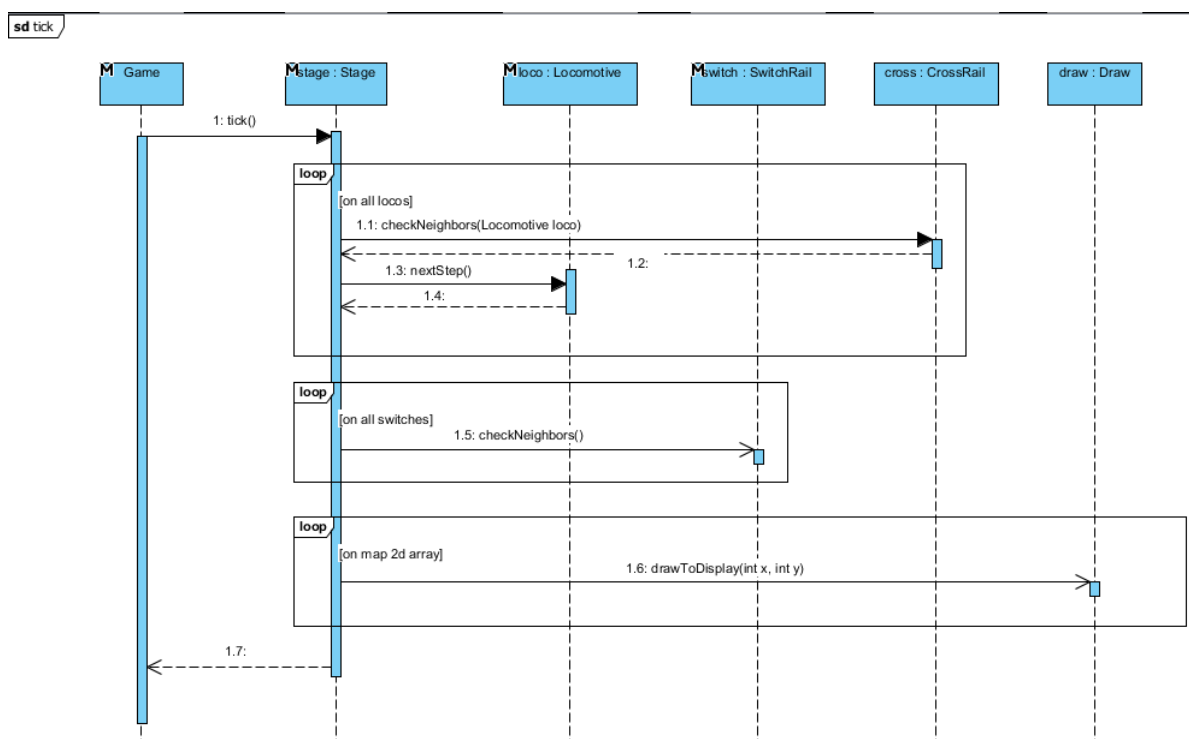
11.3.3 Stage

- **Változás:**

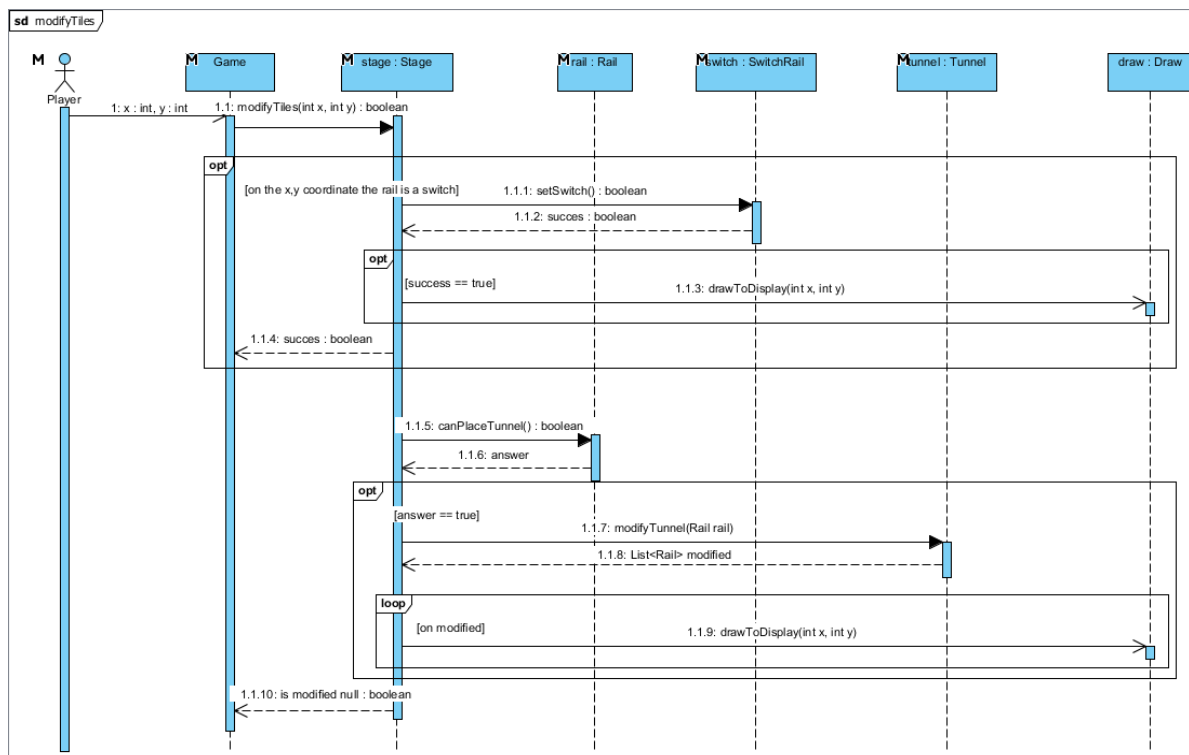
Tartalmaz egy darab privát Draw példányt, Draw draw néven.

11.4 Kapcsolat az alkalmazói rendszerrel

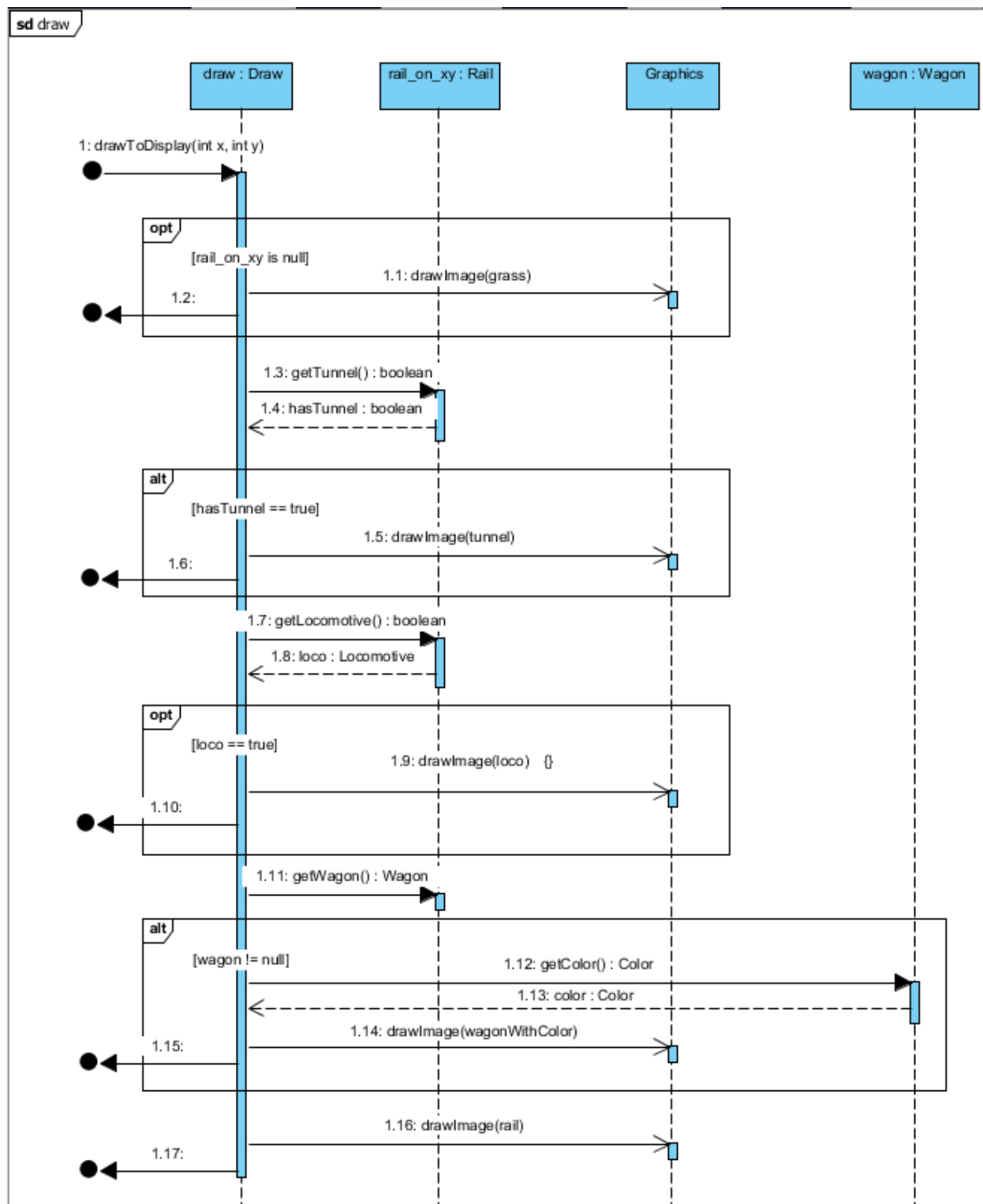
11.4.1 Tick



11.4.2 modifyTiles



11.4.3 draw



11.5 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.04.23. 18:00	1 óra	Bartók Németh Burom Koncsik Pekk	Értekezlet. Döntés: Burom,Bartók,Pekk megrajzolják a pálya építő elemeket, megtervezik az új osztályt, Németh,Koncsik kiegészítik a meglévő diagrammokat az új osztállyal.
2017.04.23. 19:00	2 óra	Bartók Burom Pekk	Tevékenység: Grafikus elemek megrajzolása,Draw osztály megtervezése
2017.04.23. 19:00	2 óra	Koncsik Németh	Tevékenység: Osztálydiagram,Szeq venciadiagram kiegészítése.

13. Grafikus változat beadása

13.1 Fordítási és futtatási útmutató

13.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
src			
logs.log	0 bájt		Alapértelmezett kimeneti fájl.
maps			
cross.dat	908 byte	2017.04.16.	Cross pálya
normal.dat	632 byte	2017.04.16.	Normal pálya
station.dat	831 byte	2017.04.16.	Station pálya
switch.dat	929 byte	2017.04.16.	Switch pálya
graphics			
crossrail.png	397 byte	2017.05.07.	Kereszt irányú sín képe.
grass.png	367 byte	2017.05.07.	Fű képe.
horizontalrail.png	377 byte	2017.05.07.	Vízszintes sín képe.
leftbottomrail.png	400 byte	2017.05.07.	Balalsó sín képe.
lefttoprail.png	396 byte	2017.05.07.	Balfelső sín képe.
locomotive.png	419 byte	2017.05.07.	Mozdony képe.
normalrail.png	365 byte	2017.05.07.	Sín képe.
rightbottomrail.png	407 byte	2017.05.07.	Jobbalsó sín képe.
righttoprail.png	405 byte	2017.05.07.	Jobbfelső sín képe.
stationblue.png	594 byte	2017.05.07.	Kék állomás képe.
stationgreen.png	594 byte	2017.05.07.	Zöld állomás képe.
stationorange.png	591 byte	2017.05.07.	Narancsságra állomás képe.
stationred.png	595 byte	2017.05.07.	Piros állomás képe.
stationyellow.png	588 byte	2017.05.07.	Sárga állomás képe.
tunnel.png	364 byte	2017.05.07.	Alagút képe.
verticalrail.png	384 byte	2017.05.07.	Függőleges sín képe.
wagonblack.png	391 byte	2017.05.07.	Szenesvagon képe.
wagonblue.png	391 byte	2017.05.07.	Kék vagon képe.
wagonempty.png	391 byte	2017.05.07.	Üres vagon képe.
wagongreen.png	396 byte	2017.05.07.	Zöld vagon képe.
wagonorange.png	394 byte	2017.05.07.	Narancsságra vagon képe.
wagonred.png	397 byte	2017.05.07.	Piros vagon képe.
wagonyellow.png	393 byte	2017.05.07.	Sárga vagon képe.
game			
Game.java	460 byte	2017.04.17.	A szimuláció legmagasabb szintje. Itt tud beleavatkozni a játékos a folyamatokba a váltó állításával, alagút építésével.
GameOverMenu.java	2436 byte	2017.05.07.	Játék vége ablak. A játék végeztével ez az ablak jön elő.

MapSelectMenu.java	4320 byte	2017.05.07.	Térképválasztó ablak. Ha kezdünk egy új játékot, ez az ablak jön elő.
StartMenu.java	2580 byte	2017.05.07.	Start menü ablak. A játék indításakor ez az ablak jön elő.
others			
CollisionException.java	550 byte	2017.04.17.	Saját kivétel osztály.
Color.java	147 byte	2017.04.17.	Az állomások, vagonok lehetséges színe.
Direction.java	365 byte	2017.04.17.	A mozdonyok mozgási iránya.
rails			
CrossRail.java	3182 byte	2017.04.17.	Keresztirányú sín osztály.
NormalRail.java	1274 byte	2017.04.17.	A sínnek a legáltalánosabb leszármazottja. Vonat léptetése valósítja meg.
Rail.java	7022 byte	2017.04.17.	Minden sínnek az ősoosztálya.
StationRail.java	4246 byte	2017.04.17.	Az állomások osztálya, itt szállnak le az utasok a vagonokról.
SwitchRail.java	5343 byte	2017.04.17.	A váltó osztály.
stage			
Draw.java	15540 byte	2017.05.07.	Grafikus megjelenítésért felelős, tárolja a pályamátrixot, ami alapján megtudja határozni, mit kell kirajzolni az adott pozícióba.
Stage.java	10325 byte	2017.04.17.	Felépíti a pályát, lehetőséget biztosít a szimulálásra a metódusai segítségével. Valamint a pálya játékos általi manipulálása ebben a classban van megvalósítva.
Tunnel.java	6787 byte	2017.04.17.	Alagút osztály.
trains			
CoalWagon.java	810 byte	2017.04.17.	Szeneskocsi osztály.
Locomotive.java	2549 byte	2017.04.17.	Mozdony osztály.
Train.java	843 byte	2017.04.17.	A sínen járó eszközök osztálya.
Wagon.java	1334 byte	2017.04.17.	A vagonokban utasok helyezkednek el, őket kell eljuttatni az állomásokra.

13.1.2 Fordítás és telepítés

Fordítás menete:

1. A .zip állomány kicsomagolása
2. Parancssor indítása, a kicsomagolt fájlok helyére navigálás (src mappa)
3. `java -version` parancs kiadása, a telepített Java verzió meghatározásához
4. A java és javac programok működéséhez szükséges path beállítása:
`set path=%path%;C:\Program Files\Java\jdk1.8.0_121\bin`
A fenti parancs egy példa, ha a telepítés helye, vagy a verziószám különbözik, aszerint kell futtatni a parancsot.
5. Fordítás mind az 5 mappában a következő paranccsal:
`javac game/*.java others/*.java rails/*.java stage/*.java trains/*.java`

13.1.3 Futtatás

Futtatni az előző pontban helyesen fordított fájlokat lehetséges. Parancssorban a fordított fájlok gyökérkönyvtárába (src) navigálás után a futtatás a `java game.Game` paranccsal történik.

13.2 Értékelés

Tag neve	Munka százalékban
Koncsik Norbert	30%
Németh Bence	25%
Bartók Patrik István	15%
Burom Bence	15%
Pekk János Richárd	15%

13.3 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.05.07. 16:00	1 óra	Bartók Burom Koncsik Németh Pekk	Értekezlet: Bartók, Burom, Pekk elkészítik a menüket, Koncsik, Németh elkészíti a Draw és módosítja a Stage osztályt.
2017.05.07. 17:00	2 óra	Bartók Burom Pekk	Tevékenység: Menük elkészítése.
2017.05.07. 17:00	4 óra	Németh	Tevékenység: Draw osztály készítése.
2017.05.07. 17:00	5 óra	Koncsik	Tevékenység: Draw, Stage osztály készítése.
2017.05.08. 18:30	30 perc	Németh	Tevékenység: A kód átnézése és kommentek elhelyezése.
2017.05.08. 19:00	1 óra	Németh	Tevékenység: Dokumentáció elkészítése.

14. Összefoglalás

14.1A projektre fordított összes munkaidő

Tag neve	Munkaidő (óra)
Koncsik Norbert	72 óra
Németh Bence	46 óra
Bartók Patrik István	26.5 óra
Burom Bence	26.5 óra
Pekk János Richárd	26.5 óra
Összesen	197.5 óra

• A feltöltött programok forrássorainak száma

Fázis	Kódsorok száma
Szkeleton	782 sor
Prototípus	1813 sor
Grafikus változat	2021 sor
Összesen	4615 sor

14.2•Projekt összegzés

14.2.1 Mit tanultak a projektből konkrétan és általában?

Általánosságban azt tanultuk meg belőle, hogy egy projektmunkát nehéz úgy elvégezni, hogy nincsen kijelölve időpont, amikor az egész csapat dolgozik. Az előleges tervezés fontos, viszont nincs értelme szigorúan követni, mert előre nem lehet látni minden nehézséget.

14.2.2 Mi volt a legnehezebb és a legkönnyebb?

A legnehezebb része az volt, hogy mindenki ráérjen egyszerre, valamint a terv követése. Minden mást könnyűnek lehet nevezni, maximum időigényes.

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

Igen, bár inkább a dokumentáció mennyiségével állt összhangban.

14.2.4 Ha nem, akkor hol okozott ez nehézséget?

14.2.5 Milyen változtatási javaslatuk van?

Több kreditet kellene adni a tárgyért (még a 3 is kevés lesz), ezzel együtt több idő is lenne rá.

14.2.6 Milyen feladatot ajánlanának a projektre?

14.2.7 Egyéb kritika és javaslat

A pontszámokat adminisztrálni lehetne valamely webes felületen.