

Projekt laboratórium 2017

Sheldon terepasztala

14 – legJava

Konzulens:

Dudás Ádám

Csapattagok

Tóth Attila	B1SS8B	toth.attila9704@gmail.com
Kunkli Richárd Zsolt	H3RUYX	kunkli.ricsi@gmail.com
Zahorán László	UM38EE	zahoranl.p@gmail.com
Zoltay Marcell	YUIRY8	zmarci14@gmail.com

2017.05.11

Tartalomjegyzék

2	Követelmény, projekt, funkcionalitás.....	10
2.1	Bevezetés.....	10
2.1.1	Cél.....	10
2.1.2	Szakterület.....	10
2.1.3	Definíciók, rövidítések.....	10
2.1.4	Hivatkozások.....	10
2.1.5	Összefoglalás.....	11
2.2	Áttekintés.....	11
2.2.1	Általános áttekintés.....	11
2.2.2	Funkciók.....	11
2.2.3	Felhasználók.....	12
2.2.4	Korlátozások.....	12
2.2.5	Feltételezések, kapcsolatok.....	12
2.3	Követelmények.....	12
2.3.1	Funkcionális követelmények.....	12
2.3.2	Erőforrásokkal kapcsolatos követelmények.....	13
2.3.3	Átadással kapcsolatos követelmények.....	13
2.3.4	Egyéb nem funkcionális követelmények.....	13
2.4	Lényeges use-case-ek.....	14
2.4.1	Use-case leírások.....	14
2.4.2	Use-case diagram.....	15
2.5	Szótár.....	15
2.6	Projekt terv.....	16
2.7	Napló.....	17
3	Analízis modell kidolgozása 1.....	18
3.1	Objektum katalógus.....	18
3.2	Model.....	18
3.2.1	Engine.....	18
3.2.2	Car.....	18
3.2.3	Station.....	18
3.2.4	Rail.....	18
3.2.5	Switch.....	18
3.2.6	Tunnel.....	19
3.3	Statikus struktúra diagramok.....	20
3.4	Osztályok leírása.....	21

3.4.1	Car.....	21
3.4.2	Engine	21
3.4.3	Node	21
3.4.4	MapItems	22
3.4.5	Model	22
3.4.6	Rail	22
3.4.7	Station	23
3.4.8	Switch	23
3.4.9	TunnelEntrance	23
3.5	Szekvencia diagram	24
3.6	State-chartok.....	26
3.7	Napló	27
4	Analízis modell kidolgozása 2	28
4.1.	Objektum katalógus	28
4.1.1	Model	28
4.1.2	Engine	28
4.1.3	Car.....	28
4.1.4	Station	28
4.1.5	Rail	28
4.1.6	Switch	28
4.1.7	Tunnel.....	28
4.2	Statikus struktúra diagramok	29
4.3	Osztályok leírása.....	30
4.3.1	Car.....	30
4.3.2	Engine	30
4.3.3	Node	30
4.3.4	MapItem	31
4.3.5	Model	31
4.3.6	Rail	31
4.3.7	Station	32
4.3.8	Switch	32
4.3.9	TunnelEntrance	32
4.3.10	Train.....	33
4.4	Szekvencia diagramok	33
4.4.1	Init.....	33
4.4.2	Add Trains.....	34

4.4.3	Move Trains	35
4.4.4	Passengers Off	36
4.4.5	Change Switch	37
4.4.6	Create Tunnel Entrance	37
4.4.7	Destroy Tunnel Entrance	38
4.5	State-chartok	39
4.5.1	Change Switch	39
4.5.2	Create Tunnel Entrance	40
4.6	Napló	40
5	Szkeleton tervezése	41
5.1	A szkeleton modell valóságos use-case-ei	41
5.1.1	Use-case diagram	41
5.1.2	Use-case leírások	42
5.2	A szkeleton kezelői felületének terve, dialógusok	44
5.3	Szekvencia diagramok a belső működésre	45
5.3.1	Add Trains	45
5.3.2	Change Switch	46
5.3.3	Crash	46
5.3.4	Create First Entrance	47
5.3.5	Create Second Entrance	47
5.3.6	Destroy Second Entrance	48
5.3.7	Destroy First Entrance	48
5.3.8	EndWin	49
5.3.9	LastPass	49
5.3.10	Move	50
5.3.11	PassangersOff	51
5.3.12	Move Train	51
5.4	Kommunikációs diagramok	52
5.4.1	Add Trains	52
5.4.2	Change Switch	53
5.4.3	Crash	54
5.4.4	Create First Entrance	55
5.4.5	Create Second Entrance	56
5.4.6	DestroySecondEntrance	57
5.4.7	Last Pass	58
5.4.8	Move	59

5.4.9	Passangers Off	60
5.4.10	Move Train.....	61
5.5	Napló	62
6	Szkeleton beadás.....	63
6.1	Fordítási és futtatási útmutató.....	63
6.1.1	Fájllista.....	63
6.1.2	Fordítás.....	63
6.1.3	Futtatás.....	64
6.2	Értékelés	64
6.3	Napló	65
7	Prototípus koncepciója.....	66
7.0	Módosítások	66
7.0.1	Funkcionális követelmények	66
7.0.2	Objektum katalógus	66
7.0.3	Use-case diagram	67
7.0.4	Osztálydiagram	68
7.0.5	Osztályok leírása.....	69
7.0.6	Cross	70
7.0.7	Szekvencia diagram	71
7.0.8	Kommunikációs diagram	74
7.1	Prototípus interface-definíciója.....	76
7.1.1	Az interfész általános leírása	76
7.1.2	Bemeneti nyelv.....	76
7.1.3	Kimeneti nyelv	78
7.2	Összes részletes use-case	79
7.3	Tesztelési terv.....	80
7.4	Tesztelést támogató segéd- és fordítóprogramok specifikálása	81
7.5	Napló	82
8	Részletes tervek.....	83
8.1	Osztályok és metódusok tervei	83
8.1.1	Manager	83
8.1.2	Menu	83
8.1.3	Game	83
8.1.4	Pause	84
8.1.5	End.....	84
8.1.6	Model	84

8.1.7	MapItem	85
8.1.8	Train	86
8.1.9	Engine	86
8.1.10	CoalCar	87
8.1.11	Car	87
8.1.12	tNode	88
8.1.13	Station	88
8.1.14	Rail	89
8.1.15	Cross	89
8.1.16	Switch	89
8.1.17	TunnelEntrance	90
8.2	A tesztek részletes tervei, leírásuk a teszt nyelvén	90
8.2.1	Váltó állítása	90
8.2.2	Vonat mozgása kereszteződésben	91
8.2.3	Leszállás	93
8.2.4	Felszállás	94
8.2.5	Ütközés	94
8.2.6	Alagút bejárat építése	94
8.2.7	Alagút bejárat törlése	95
8.3	A tesztelést támogató programok tervei	95
8.4	Napló	96
10	Prototípus beadása	97
10.1	Fordítási és futtatási útmutató	97
10.1.1	project könyvtár tartalma	97
10.1.2	maps könyvtár tartalma	98
10.1.3	ref_outputs könyvtár tartalma	98
10.1.4	Fordítás	99
10.1.5	Futtatás	99
10.2	Tesztek jegyzőkönyvei	100
10.2.1	Váltó állítása	100
10.2.2	Vonat mozgatása kereszteződésben	100
10.2.3	Leszállás	100
10.2.4	Felszállás	100
10.2.5	Ütközés	100
10.2.6	Alagút bejárat építése	100
10.2.7	Alagút bejárat törlése	101

10.3	Értékelés	101
10.4	Napló	101
11	Grafikus felület specifikációja.....	102
11.1	A grafikus interfész	102
11.2	A grafikus rendszer architektúrája	105
11.2.1	A felület működési elve	105
11.2.2	A felület osztály-struktúrája	106
11.3	A grafikus objektumok felsorolása	107
11.3.1	Model	107
11.3.2	View	107
11.3.3	EngineGraphics.....	108
11.3.4	CarGraphics	109
11.3.5	TunnelEntranceGraphics	109
11.3.6	CoalCarGraphics	110
11.3.7	StationGraphics	110
11.3.8	Game	110
11.3.9	Menu	111
11.3.10	Pause	111
11.3.11	End.....	111
11.4	Kapcsolat az alkalmazói rendszerrel.....	113
11.4.1	Init.....	113
11.4.2	Game	114
11.4.3	End.....	115
11.4.4	Draw	116
11.5	Napló	116
13	Grafikus változat beadása	117
13.1	Fordítási és futtatási útmutató.....	117
13.1.1	Fájllista.....	117
13.1.2	Fordítás és telepítés	118
13.2	Futtatás.....	118
13.3	Értékelés	119
13.4	Napló	119
14	Összefoglalás	120
14.1	A projektre fordított összes munkaidő.....	120
14.2	• Projekt összegzés.....	120
14.2.1	Mit tanultak a projektből konkrétan és általában?	120

14.2.2	Mi volt a legnehezebb és a legkönnyebb?	120
14.2.3	Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?.....	121
14.2.4	Ha nem, akkor hol okozott ez nehézséget?.....	121
14.2.5	Milyen változtatási javaslatuk van?.....	121
14.2.6	Milyen feladatot ajánlanának a projektre?	121
14.2.7	Egyéb kritika és javaslat.....	121
1.	ábra Use-case diagram	15
2.	ábra Osztálydiagram.....	20
3.	ábra Init szekvencia diagram	24
4.	ábra Vonat hozzáadása és mozgatása szekvencia diagram	24
6.	ábra Alagút törlése szekvencia diagram.....	25
5.	ábra Alagút létrehozása szekvencia diagram.....	25
7.	ábra Váltó állítás state chart.....	26
8.	ábra Alagút építés state chart	26
9.	ábra Osztálydiagram.....	29
10.	ábra Init szekvencia diagram	33
11.	ábra Vonat hozzáadás szekvencia diagram	34
12.	ábra Vonat mozgatása szekvencia diagram	36
13.	ábra Leszállás szekvencia diagram	36
14.	ábra Váltó állítás szekvencia diagram	37
15.	ábra Első alagút létrehozása szekvencia diagram	37
16.	ábra Második alagút létrehozása szekvencia diagram	38
17.	ábra Első alagút törlése szekvencia diagram.....	38
18.	ábra Második alagút törlése szekvencia diagram.....	39
19.	ábra Váltó állítás state-chart	39
20.	ábra Alagút létrehozás state-chart.....	40
21.	ábra Szkeleton Use-case diagram.....	41
22.	ábra Add Trains szekvencia diagram	45
23.	ábra Change Switch szekvencia diagram.....	46
24.	ábra Crash szekvencia diagram	46
25.	ábra Create First Entrance szekvencia diagram	47
26.	ábra Create Second Entrance szekvencia diagram.....	47
27.	ábra Destroy Second Entrance szekvencia diagram.....	48
28.	ábra Destroy First Entrance szekvencia diagram	48
29.	ábra Játék befejezése győzelemmel szekvencia diagram	49
30.	ábra LastPass szekvencia diagram.....	49
31.	ábra Move szekvencia diagram	50
32.	ábra Passangers leszállás szekvencia diagram	51
33.	ábra Move Trains szekvencia diagram	51
34.	ábra Add Trains kommunikációs diagram	52
35.	ábra Change Switch kommunikációs diagram.....	53
36.	ábra Crash kommunikációs diagram	54
37.	ábra Create First Entrance kommunikációs diagram	55
38.	ábra Create Second Entrance kommunikációs diagram.....	56

39. ábra Destroy Second Entrance kommunikációs diagram.....	57
40. ábra Last Pass kommunikációs diagram.....	58
41. ábra Move kommunikációs diagram	59
42. ábra Passangers Off kommunikációs diagram	60
43. ábra Move kommunikációs diagram	61
44. ábra Fordítás és futtatás példa.....	64
45. ábra Prototípus Use-case diagram	67
46. ábra Prototípus osztálydiagram	68
47. ábra Prototípus crash szekvencia diagram	71
48. ábra Prototípus move szekvencia diagram	72
49. ábra Prototípus GetOn Passengers szekvencia diagram	73
50. ábra Prototípus Crash kommunikációs diagram	74
51. ábra Prototípus Move kommunikációs diagram	75
52. ábra Prototípus GetOn Passengers kommunikációs diagram	75
53. ábra Prototípus fordítás és futtatás	99
54. ábra Grafikus interfész főmenü elképzelés	102
55. ábra Grafikus interfész szünet menü elképzelés.....	102
56. ábra Grafikus interfész veszített játék menü elképzelés.....	103
57. ábra Grafikus interfész nyert játék menü elképzelés	103
58. ábra Grafikus interfész játék elképzelés.....	104
59. ábra MVC grafikus rendszer architektúrája.....	105
60. ábra Grafikus megjelenítés osztály diagram	106
61. ábra Grafikus felület Init szekvencia diagram	113
62. ábra Grafikus felület Game szekvencia diagram	114
63. ábra Grafikus felület End szekvencia diagram.....	115
64. ábra Grafikus felület Draw szekvencia diagram	116

2. Követelmények, projekt, funkcionalitás

2 Követelmény, projekt, funkcionalitás

2.1 Bevezetés

2.1.1 Cél

A dokumentum célja, hogy a megrendelő által kapott feladatnak rögzítsük a követelményeit és funkcionalitásait. Erre azért van szükség, hogy szinkronban legyünk a megrendelővel és azt a szoftvert készítsük el amit ő megrendelt.

2.1.2 Szakterület

A megrendelőnk által kért szoftvert szórakozás céljára lehet majd használni, mivel egy játék programot készítünk.

2.1.3 Definíciók, rövidítések

Vonat: Vasúti közlekedésben vonatnak nevezik egy vagy több vasúti jármű olyan szerelvényét, ami a vasúti pálya vonalán közlekedik. Személy vagy teherszállításra alkalmas. A vonat vontatójárműve a mozdony.

Állomás: Vasúti hálózatban csomóponti szerepet tölt be. Itt történik meg az utasok fel- és leszállása.

Váltó: A vasúti váltó három irányból közelíthető meg. Két esetben csak a harmadik irányba hagyhatja el a vonat a váltót, a harmadik esetben van lehetőség a kimenetének megváltoztatására.

Sín: A vonat közlekedésének lehetőségét biztosítja. A vonat a sínen, annak irányába tud mozogni.

Alagút: A sín két pontját összekötő szakasz, melynek végpontjai közvetlen a sínen találhatóak. Felülírja a vonat alapvető haladási irányát. A vonat e két pont között halad, ha az egyiken bement, a másikon fog kimenni, a beállított irányban.

2.1.4 Hivatkozások

- <https://www.iit.bme.hu/targyak/BMEVIIIAB02> - A tárgy honlapja
- Google Drive - Közös dokumentumszerkesztés
- Git - Verziókezelés

2.1.5 Összefoglalás

A dokumentum további részében egy általános áttekintést adunk a kialakítandó szoftverről néhány blokkra lebontva és a blokkok közötti kommunikációról. Ezt követően a program funkcióinak informatikai kifejezések nélküli leírása következik. Ismertetjük a program funkcionális és egyéb követelményeit. Végül pedig megadjuk a funkcionális követelmények használati eseteit.

2.2 Áttekintés

2.2.1 Általános áttekintés

- **Modell** - A játék elemeit tartalmazza.
A modell alrendszer a játék elemeit tartalmazza. Az irányító a felhasználó parancsai alapján ezt módosítja.
- **Irányító** - A játék vezérlését végzi.
Az irányító alrendszer a felhasználó parancsai alapján módosítja a modellt.
- **Megjelenítés** - A játék megjelenítéséért felel.
A megrendelt szoftver megjelenítését egy grafikus alrendszer végzi, amely tájékoztatja a felhasználót minden hasznos információról, mint például a vonat helyzete, váltó állása, alagút bejáratának helye.

2.2.2 Funkciók

A játékban a felhasználónak vonatokat kell irányítani úgy, hogy a vonaton lévő utasokat elszállítsa az állomásokra megfelelő sorrendben. Egy vonat mozdonyból és néhány kocsiból áll. A kocsinak és állomásoknak színük van és egy adott színű állomáson csak az adott színű kocsiban utazó utasok szállnak le (piros állomáson a piros kocsik utasai). Azonban a leszállásnak sorrendje van, mégpedig a mozdonytól kezdődő sorrendben lehet a kocsikat elhagyni. Ebből adódóan ha egy állomáshoz érünk és a mozdonyhoz legközelebbi utasokat szállító kocsi színe nem egyezik meg az állomáséval, akkor a vonat keresztül halad az állomáson. Természetesen a pályán nem csak egy vonat van, hanem időnként a pálya széléről újabb vonatok indulnak el. A vonatok síneken közlekednek, a sínek elágazásait pedig váltók kötik össze. A pálya bizonyos helyein alagút bejáratait lehet felépíteni és megszüntetni. Egyszerre egy időben csak két ilyen bejárat lehet megépítve a pályán. A két bejáratot egy speciális alagút köti össze ha a vonat bemegy az egyik szájon a másikon jön ki. A vonat az alagútban nem látható.

A felhasználó a játékot el tudja indítani (csak alap beállítások vannak), ezután megjelenik a pálya és irányítani tudja a vonatok útvonalát a váltók állításával, vagy azzal, ha létrehoz illetve eltüntet egy-egy alagút bejáratot. Az újonnan megjelenő vonatok utasait is el kell vinnie a játékosnak a megfelelő színű állomásra a fenti szabályok betartásával. A játék célja, hogy eljuttassuk az összes utast a megfelelő állomásra. Ha ez sikerül, akkor a játék folytatódik a következő pályával, ha viszont a szállítás közben összeütköznek a vonatok a játékos elveszíti a játékot. A játék nehézségét az adja, hogy egyre több és több vonat utasait kell a játékosnak elszállítani.

2.2.3 Felhasználók

A programot az átlagos felhasználók is könnyen fogják tudni kezelni egyszerű kezelőfelülete miatt. A játékot 6 éves kortól bárki használhatja.

2.2.4 Korlátozások

A megrendelt szoftverhez Java futtató környezetre van szükség. Mac OS operációs rendszeren a szoftver nem indul el (nem futtatható).

2.2.5 Feltételezések, kapcsolatok

Tárgy honlapja - A tárgy honlapján található a feladat kiírás, ami alapján meghatároztuk a követelményeket.

Google Drive - A közös dokumentumszerkesztést teszi lehetővé.

Git - A program verziókezelését teszi lehetővé.

GitHub - A kód megosztását segíti

2.3 Követelmények

2.3.1 Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
Func1	A játék elindítható	Bemutató	Alapvető	Feladat	Init	
Func2	A váltók állíthatók	Bemutató	Alapvető	Feladat	ChangeSwitch	
Func3	Az alagút bejárat építhető	Bemutató	Alapvető	Feladat	CreateEntrance	
Func4	Az alagút bejáratának iránya megadható	Bemutató	Fontos	Csapat	CreateEntrance	
Func5	Az alagút bejárat lebontható	Bemutató	Alapvető	Feladat	DestroyEntrance	
Func6	Időnként új vonatok indulnak	Bemutató	Alapvető	Feladat	AddNewTrain	
Func7	A vonatok mozognak	Bemutató	Alapvető	Feladat	MoveTrain	

Func8	A játékos tudja vezérelni a játékot	Bemutató	Alapvető	Csapat		
Func9	Az utasok le tudnak szállni a vonatról	Bemutató	Alapvető	Feladat	MoveTrain	
Func10	A vonatok össze tudnak ütközni	Bemutató	Alapvető	Feladat	MoveTrain	

2.3.2 Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
Resource1	Windows	Bemutató	Fontos	Csapat	
Resource2	Java JRE	Bemutató	Fontos	Csapat	
Resource3	Java JDK	Bemutató	Fontos	Csapat	
Resource4	Git	Bemutató	Fontos	Csapat	
Resource5	GitHub	Bemutató	Fontos	Csapat	

2.3.3 Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
Touse1	Windows	Bemutató	Fontos	Csapat	
Touse2	Java JRE	Bemutató	Fontos	Csapat	

2.3.4 Egyéb nem funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
Nonfunc1	JUnit Tesztek	Kiértékelés	Fontos	Laborvezető	
Nonfunc2	Könnyű kezelés	Kiértékelés	Fontos	Csapat	

2.4 Lényeges use-case-ek

2.4.1 Use-case leírások

Use-case neve	Init
Rövid leírás	Betölti a pályát, elindítja a játékot.
Aktorok	User

Use-case neve	Exit
Rövid leírás	A játékból való kilépés.
Aktorok	User

Use-case neve	Pause
Rövid leírás	A játékmenet ideiglenes megszakítása.
Aktorok	User

Use-case neve	ChangeSwitch
Rövid leírás	A felhasználó változtatja a váltó állását
Aktorok	User
Forgatókönyv	Az aktív jelző, az egyik kimenetről a másikra kerül

Use-case neve	CreateEntrance
Rövid leírás	A felhasználó alagút bejáratot készít
Aktorok	User
Forgatókönyv	Az alagút bejárat a váltók és állomások kivételével, a sín bármely pontjára tehető. Az alagút bejáratának irányát be lehet állítani.

Use-case neve	DestroyEntrance
Rövid leírás	A felhasználó törli az alagút bejáratot
Aktorok	User
Forgatókönyv	Egy már meglévő alagút bejáratot töröl

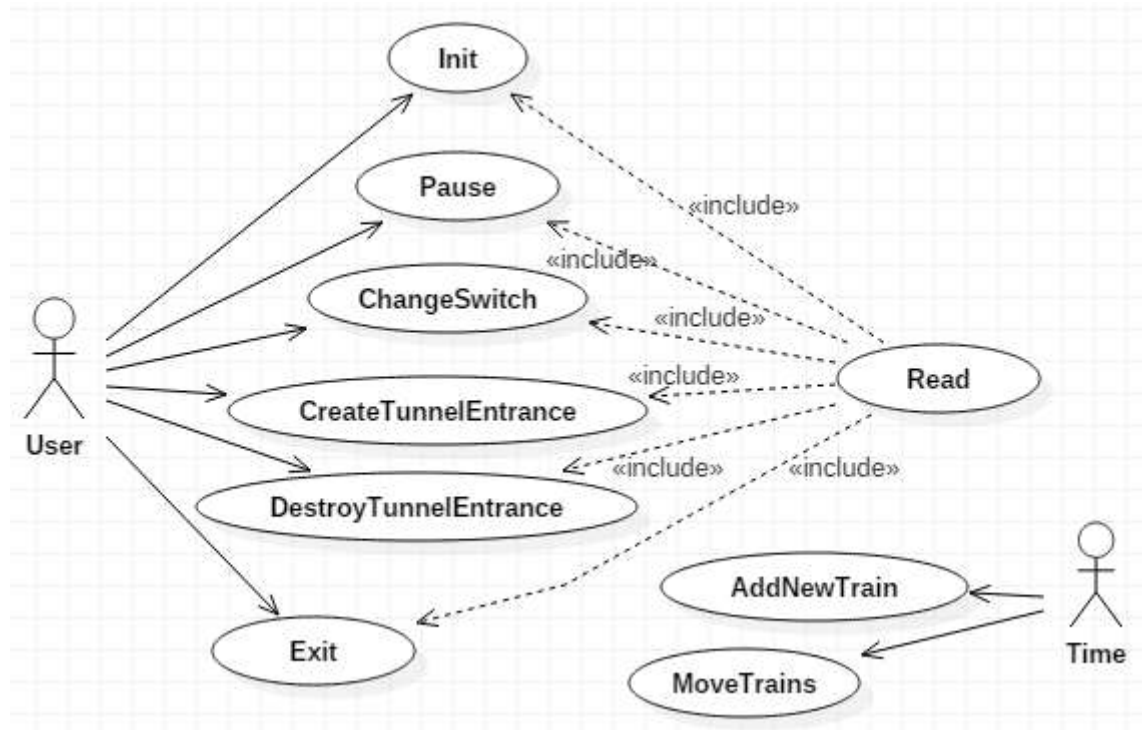
Use-case neve	Read
Rövid leírás	Az utasítások megadása.
Aktorok	User
Forgatókönyv	Az utasítások megadása, mint például alagút bejárat építése, rombolása, váltó állítása.

Use-case neve	AddNewTrain
Rövid leírás	Új vonat hozzáadása
Aktorok	Time
Forgatókönyv	Megfelelő időközönként új vonat indul a pályán

Use-case neve	MoveTrain
Rövid leírás	A vonatok mozgatása

Aktorok	Time
Forgatókönyv	A pályán lévő vonatok megfelelő időközönként előre lépnek

2.4.2 Use-case diagram



1. ábra Use-case diagram

2.5 Szótár

Windows - a Microsoft Corporation gyártotta operációs rendszerek, illetve az ezekbe épített többfeladatos grafikus felhasználói felületek, valamint bizonyos mobiltechnológiák családja. A "Windows" szó és logó a Microsoft cég védjegye.

Java - A Java általános célú, objektumorientált programozási nyelv, amelyet a Sun Microsystems fejlesztett a '90-es évek elejétől kezdve egészen 2009-ig, amikor a céget felvásárolta az Oracle. 2011-ben a Java 1.7-es verzióját az új tulajdonos gondozásában adták ki.

JUnit teszt - Unit egy egységteszt keretrendszer Java programozási nyelvhez. Ezeken egységtesztetek karbantartására, csoportos futtatására szolgál ez a keretrendszer. A JUnit teszteteket gyakran a build folyamat részeként szokták beépíteni. Pl. napi build-ek esetén ezek a tesztek is lefutnak. A release akkor hibátlan, ha az összes teszt hibátlanul lefut.

2.6 Projekt terv

A betartandó ütemterv az alábbi:

Febr. 20.	Követelmény, projekt, funkcionalitás beadása
Febr. 27.	Analízis modell kidolgozása 1. beadása
Márc. 6.	Analízis modell kidolgozása 2. beadása
Márc. 13.	Szkeleton tervének beadása és a dokumentum herculesre való feltöltése
Márc. 20.	Szkeleton beadása és a forráskód hercules-re való feltöltése
Márc. 27.	Prototípus koncepciójának beadása
Ápr. 3.	Részletes tervek beadása
Ápr. 10.	Prototípus készítése, tesztelése
Ápr. 17.	Prototípus beadás és a forráskód, a tesztbemenetek és az elvárt kimenetek hercules-re való feltöltése
Ápr. 24.	Grafikus felület specifikációjának beadása
Máj. 1.	Grafikus változat készítése
Máj. 8.	Grafikus változat beadása és a forráskód hercules-re való feltöltése
Máj. 10	Összefoglalás beadása és feltöltése

A konzultációs alkalmakon kívül (szerda 12 - 14) a csoport heti 1 - 2 alkalommal összeül, hogy átbeszélje a teendőket, hogy kinek milyen szerepe van a feladatok elvégzésében. A következő gyűlés idejét, mindig az előző alkalommal vitatjuk. Előre láthatóan péntekenként és hétfőn- vagy keddenként fog gyűlni a csapat.

A leadandó dokumentum template-eket feltöltöttük a Google Drive-ba, ahol közösen szerkesztjük azt.

A forráskód megosztását GitHub-bal oldjuk meg.

A végleges dokumentumok kinyomtatásáért elsősorban Zahorán László a felelős, mivel neki van otthon nyomtatója. Bárminemű probléma esetén, a Schönherz Zoltán Kollégiumban is tudunk nyomtatni.

A dokumentumok leadásáért Zoltay Marcell a felelős.

2.7 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.02.15 21:30	1 óra	Tóth	Bevezetés
2017.02.17. 14:00	2,5 óra	Tóth Kunkli Zahorán Zoltay	Értekezlet: A játék nagy vonalakban való összefoglalása, dokumentálása.
2017.02.17. 17:00	1 óra	Kunkli	Use-Case leírás
2017.02.17. 21:00	0,5 óra	Zoltay	Use-case diagram
2017.02.17 23:00	1 óra	Tóth	Funkciók
2017.02.18. 13:00	0,5 óra	Kunkli	Projekt terv
2017.02.18. 14:00	0,5 óra	Zoltay	Áttekintés
2017.02.18. 14:30	0,5 óra	Zoltay	Felhasználók, Korlátozások
2017.02.18 17:45	1 óra	Zahorán	Funkcionális követelmények
2017.02.19. 14:00	0,5 óra	Kunkli	Szótár
2017.02.19 18:00	0,5 óra	Zahorán	Átadással, Erőforrásokkal kapcsolatos követelmények
2017.02.19. 18:30	0,5 óra	Zoltay	Feltételezések és kapcsolatok, Dokumentum formázása
2017.02.19 22:00	0,5 óra	Zahorán	Egyéb nem funkcionális követelmények, Végleges formázás, nyomtatás

3. Analízis modell kidolgozása

3 Analízis modell kidolgozása 1

3.1 *Objektum katalógus*

3.2 *Model*

A Model objektum a pálya leírásához és kezeléséhez tartozó elemeket gyűjti össze és végez rajta műveleteket a játék működése közben.

3.2.1 Engine

A mozdonyt megvalósító osztály felelőssége hogy vezesse a vonatot a síneken, vontassa a vagonokat. A mozdony a játékban nem közlekedhet vagon nélkül, és minden esetben előre halad, azaz a vagonokat vontatja. Két mozdony ütközése esetén nem sikerül a pályát teljesíteni.

3.2.2 Car

A Car osztály a vagonokat valósítja meg. A vagonok a vonat elején levő mozdonyt követik. Több vagon is kapcsolódhat egymáshoz. Minden szomszédos vagon színe különbözik egy szerelvényen és utasokat szállít. Az utasok csak a mozdonytól a szerelvény hátulja felé szállhatnak le a megadott helyeken.

3.2.3 Station

Az állomást megvalósító Station osztály jelképezi az utasok célját a játékban, ahová szeretnének eljutni. Az állomások színekkel vannak megkülönböztetve a pályán és minden esetben sín vezet hozzájuk. A megfelelő színű vagonban utazó utasok csak a megfelelő színű állomáson szállhatnak le amikor a vonat elhalad mellette.

3.2.4 Rail

A Rail osztály a síneket valósítja meg a játékban, ábrázolja a vonatok lehetséges útvonalait. Több elem is kapcsolódhat a sínekhez, úgy mint: váltó, állomás, másik sín, alagút. A vonatok csak a síneken képesek az elem között közlekedni.

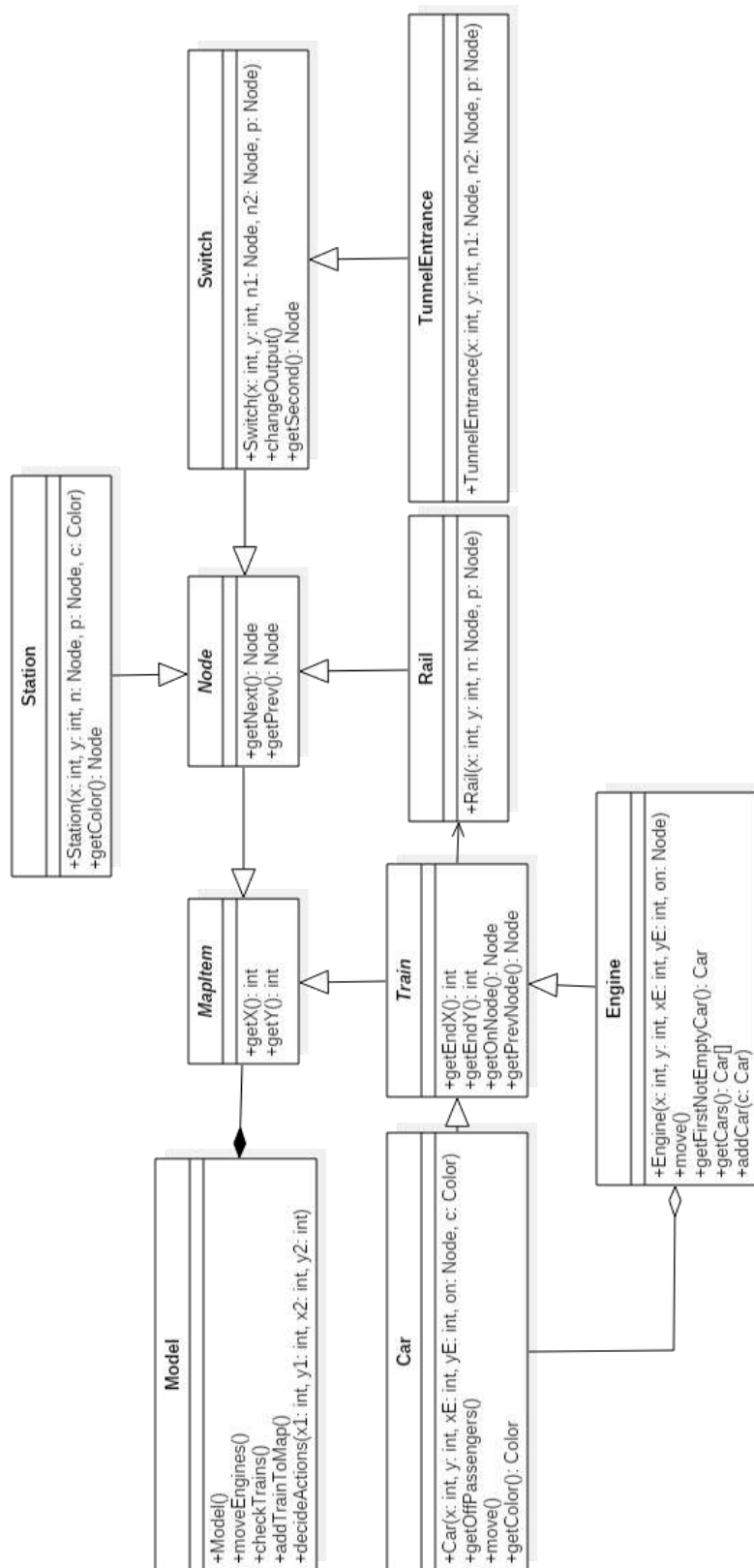
3.2.5 Switch

A Switch osztály a vasúti váltót valósítja meg a játékban, három sín elemet köt össze, két irány esetén csak a harmadikba közlekedhet rajta a vonat, viszont a harmadik irányból érkezve a játékos állíthatja, hogy merre szeretné navigálni a vonatot. A váltók a pálya megkezdésekor mindig valamilyen alaphelyzetbe kerülnek, amelyek aztán megváltoztathatóak, de nincs olyan eset, amikor nincs kiválasztott kimenete.

3.2.6 Tunnel

Az alagutat megvalósító Tunnel osztály lehetőséget ad a játékosnak, hogy adott helyzetből nem feltétlenül elérhető helyre irányítsa a vonatot. Legfeljebb két alagút bejárat lehet elhelyezve a pályán. Mind a két elem használható bejáratként és kijáratként is egyaránt, minden esetben amelyik elemen bement a vonat a másikban jelenik meg. Amennyiben csak egy elem kerül elhelyezésre a pályán a vonat érkezéséig a vonat csak áthalad rajta. Az alagút bejáratot csak sínen lehet elhelyezni.

3.3 Statikus struktúra diagramok



2. ábra Osztálydiagram

3.4 Osztályok leírása

3.4.1 Car

- **Felelősség**

A vagon megvalósító osztály. A vagonnak színe van. A rajta utazó összes utas a vagon színével azonos színű állomáson leszáll, ha az a mozdonyhoz legközelebbi nem üres vagon.

- **Ősosztályok:** MapItems → Train

- **Interfészek:** -

- **Attribútumok:** -

- **Metódusok**

- **Car(x: int, y: int, xE: int, yE: int, on: Node, c: Color):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
- **void getOffPassengers():** Az összes utast leszállítja a vagonból.
- **void move():** A vagon mozgatja.
- **Color getColor():** Visszatér a vagon színével.

3.4.2 Engine

- **Felelősség**

A Engine osztály valósítja meg a mozdonyt. A vagonokat vontatásáért felel.

- **Ősosztályok:** MapItem → Train

- **Interfészek:** -

- **Attribútumok:** -

- **Metódusok**

- **Engine(x: int, y: int, xE: int, yE: int, n: Node):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
- **void move():** Mozgatja a mozdonyt. Meghívja a vagonok move() metódusát.
- **Car getFirstNotEmptyCar():** Visszatér az első nem üres vontatott vagonnal.
- **Car[] getCars():** Visszatér a vontatott vagonok tömbjével.
- **void addCar(Car c):** Hozzáad a vagonok listájához egy vagon.

3.4.3 Node

- **Felelősség**

Absztrakt osztály, a csomópontokat valósítja meg. ezek segítségével lesz bejárható a pálya, tárolja a következő és előző elemet, így a váltók, sínek, állomások, összekapcsolhatóak.

- **Ősosztályok:** MapItems

- **Interfészek:** -

- **Attribútumok:** -

- **Metódusok**

- **Node getNext():** A következő csatlakoztatott elemmel tér vissza.
- **Node getPrev():** A előző csatlakoztatott elemmel tér vissza.

3.4.4 MapItems

- **Felelősség:**

Absztrakt osztály, a pályát alkotó elemek őssztálya, rögzíti azok helyét a pályán.

- **Őssztályok:** -
- **Interfészek:** -
- **Attribútumok:** -
- **Metódusok**
 - **int GetX():** Visszatér az elem x koordinátájának értékével.
 - **int GetY():** Visszatér az elem y koordinátájának értékével.

3.4.5 Model

- **Felelősség**

Összefogja a rendszerben levő egységeket. Meg tudja hívni a függvényeket, vezérli a vonatok mozgását, az új vonatok indítását, és biztosítja az alagutat a játékban.

- **Őssztályok:** -
- **Interfészek:** -
- **Attribútumok:** -
- **Metódusok**
 - **Model():** A konstruktor létrehozza a model objektumot
 - **moveEngines():** Mozgatja a vonatokat.
 - **checkTrains():** Vizsgálja a vonatok helyzeteit (van-e még vonata pályán, ütköztek-e a vonatok, üres-e a vonat, összeveti az állomás színét a legelső nem üres kocsiéval)
 - **addTrainToMap():** Új vonatot indít a pályán.
 - **decideActions(x1: int, y1: int, x2: int, y2:int):** A kapott paraméterek alapján eldönti az elvégzendő műveletet. Ez lehet alagút építés, rombolás, váltó állítása.

3.4.6 Rail

- **Felelősség**

A Rail osztály a síneket valósítja meg. A Node osztály leszármazottja, így a két végéhez további elemek kapcsolhatóak.

- **Őssztályok:** MapItem→Node
- **Interfészek:** -
- **Attribútumok:** -
- **Metódusok**
 - **Rail(x: int, y: int, n: Node, p: Node):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.

3.4.7 Station

- **Felelősség**

A Station osztály a megállót valósítja meg. A megálló színével azonos színű vagonokból az utasok leszállnak itt.

- **Ősosztályok: MapItem→Node**

- **Interfészek: -**

- **Attribútumok: -**

- **Metódusok**

- **Station(x: int, y: int, n: Node, p: Node, c: Color):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
- **Color getColor():** Visszatér az állomás színével

3.4.8 Switch

- **Felelősség**

A Switch osztály a váltót valósítja meg a pályán, Node-ból származik le ezért két elem kapcsolható hozzá, a harmadik

- **Ősosztályok: MapItem→Node**

- **Interfészek: -**

- **Attribútumok: -**

- **Metódusok**

- **Switch(x: int, y: int, n1: Node, n2: Node, p: Node):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
- **void changeOutput():** Megváltoztatja a kiválasztott kimenetet
- **Node getSecond():** Megadja mi áll a váltó másik kimenetén

3.4.9 TunnelEntrance

- **Felelősség**

A TunnelEntrance osztály valósítja meg az alagút bejáratot a pályán.

- **Ősosztályok: MapItem→Node→ Switch**

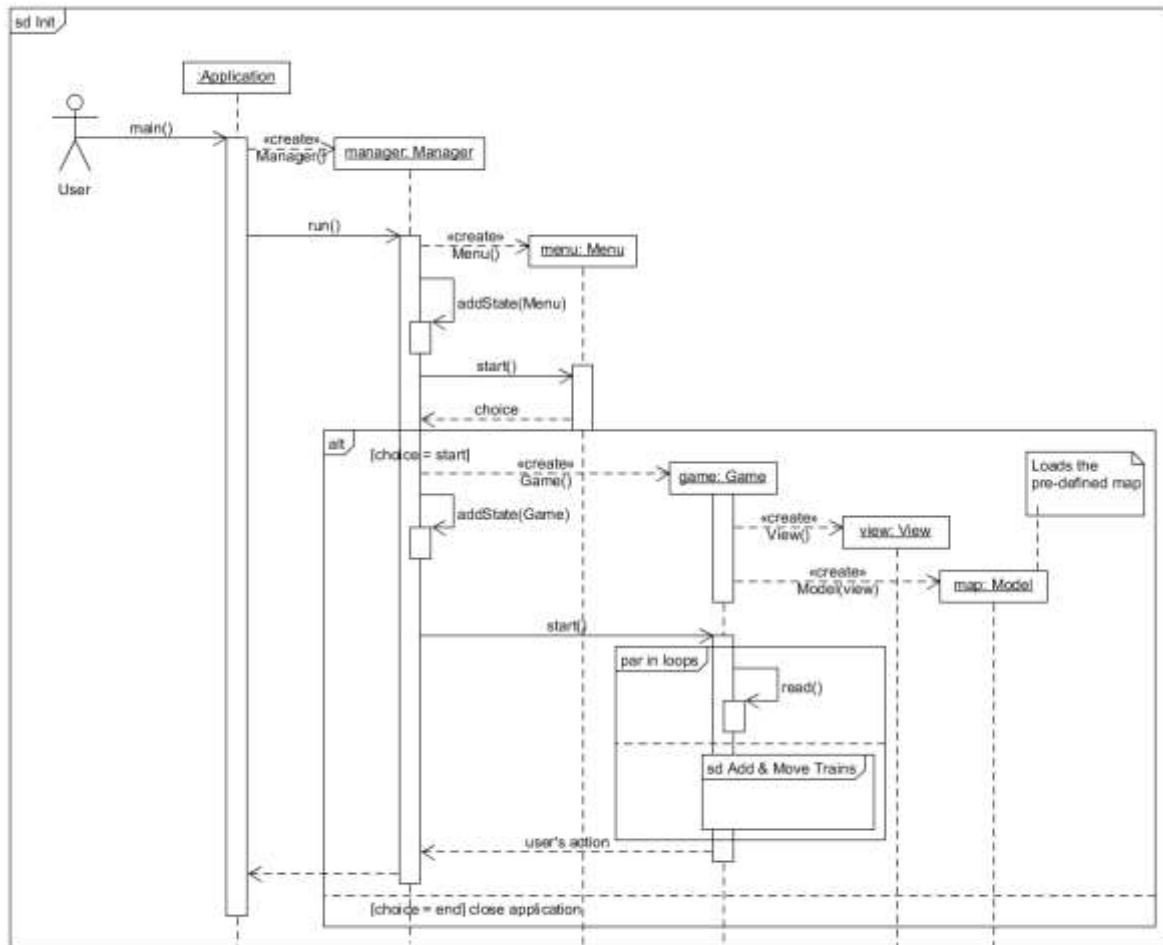
- **Interfészek: -**

- **Attribútumok**

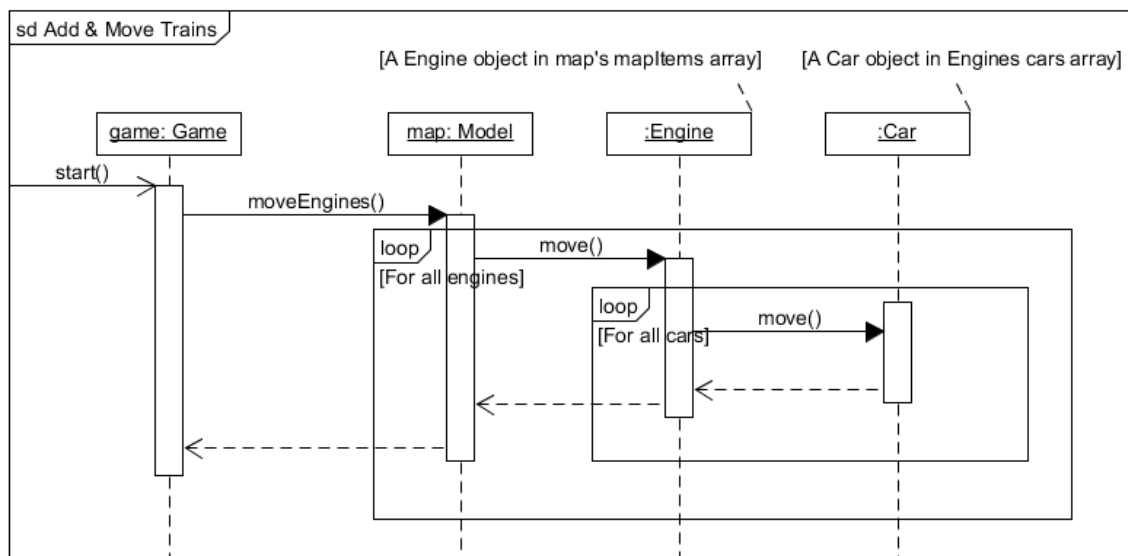
- **Metódusok**

- **TunnelEntrance(x: int, y: int, n1: Node, n2: Node, p: Node):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.

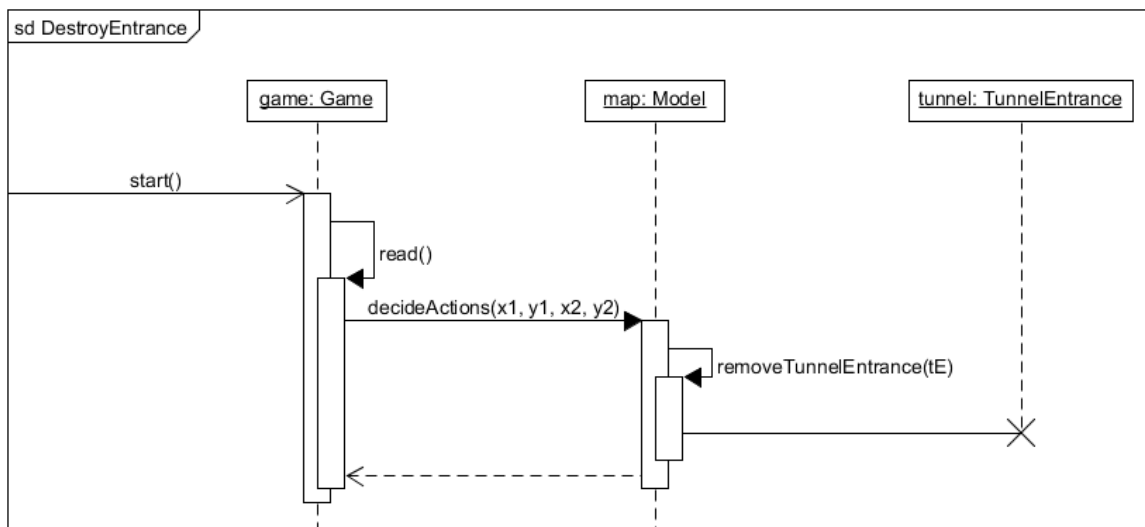
3.5 Szekvencia diagram



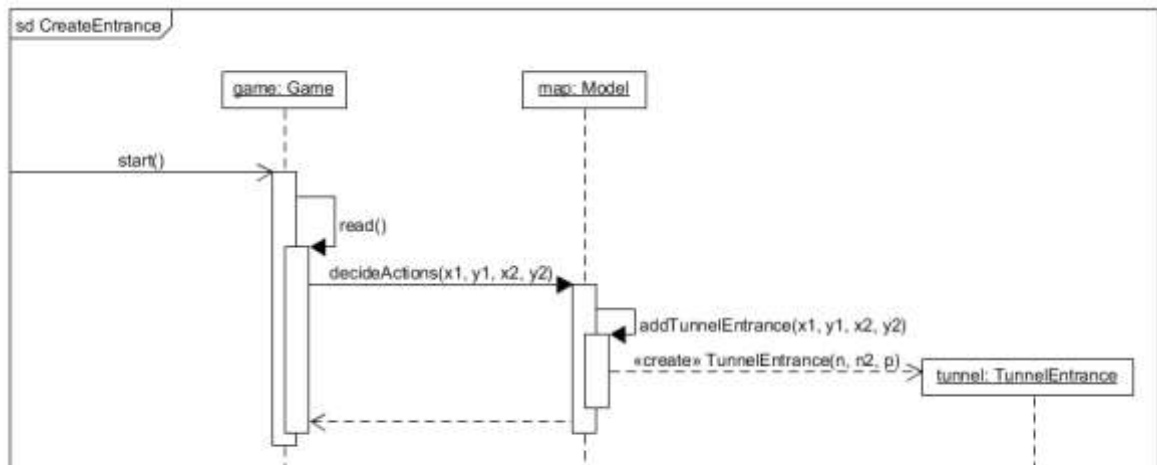
3. ábra Init szekvencia diagram



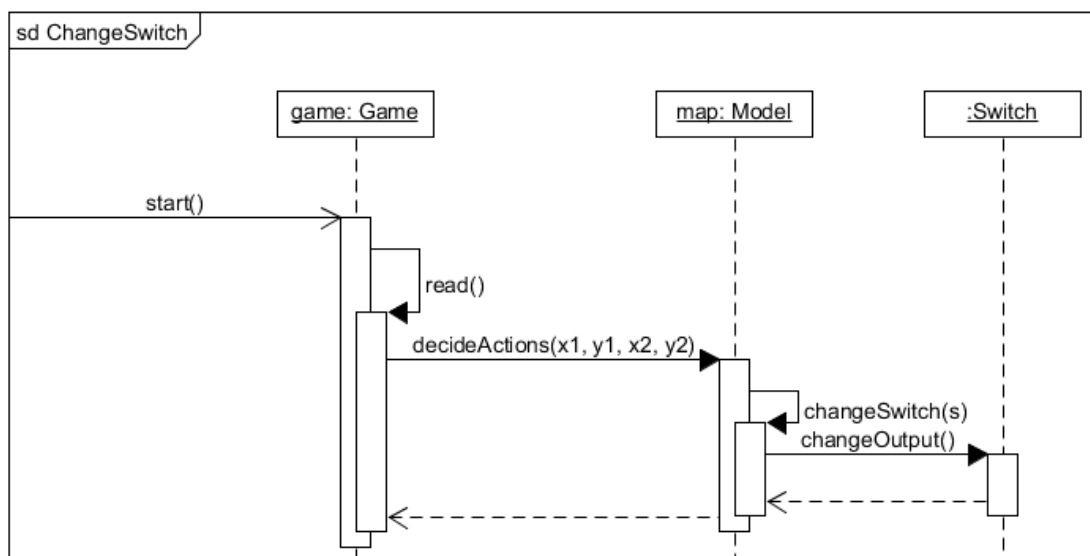
4. ábra Vonat hozzáadása és mozgatása szekvencia diagram



5. ábra Alagút törlése szekvencia diagram

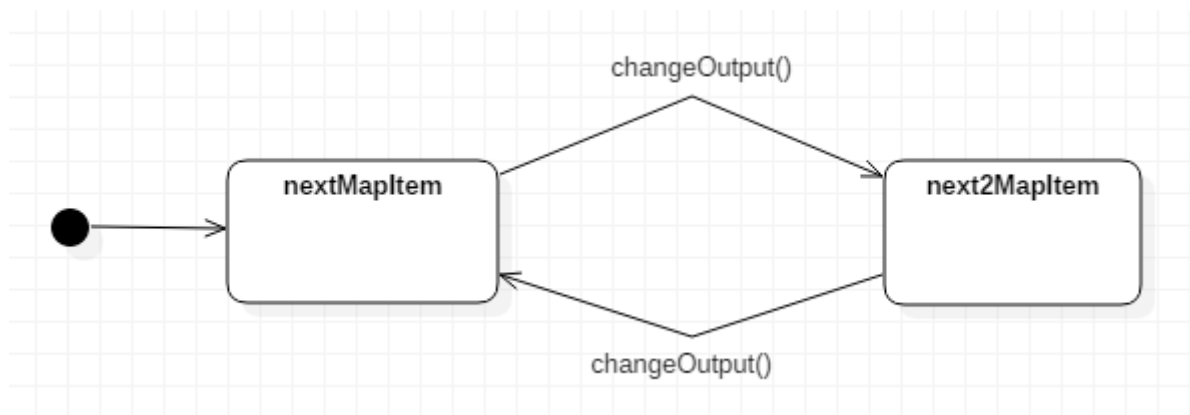


6. ábra Alagút létrehozása szekvencia diagram



7. ábra Váltó állítás szekvencia diagramm

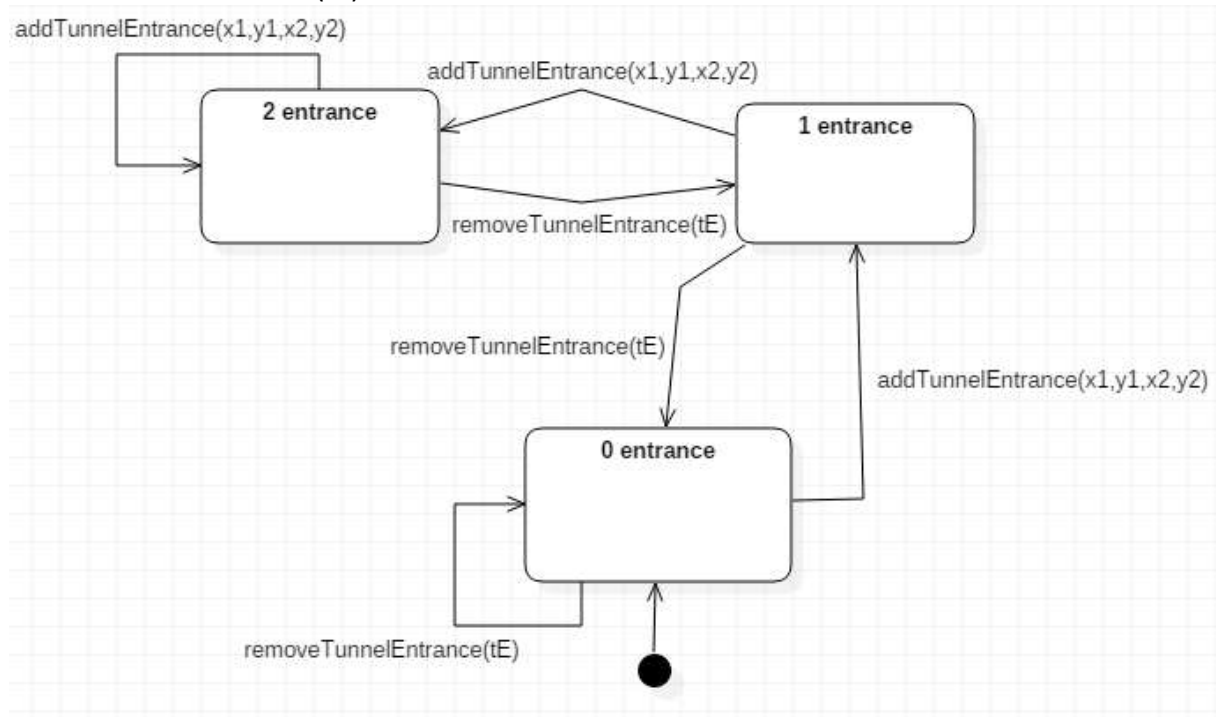
3.6 State-chartok



7. ábra Váltó állítás state chart

A fenti állapot diagram a váltó állapotait mutatja. A váltó két kimenete között lehet váltani a `changeOutput()` függvénnyel. A váltót a játékos tudja állítani.

A lenti állapot diagram az alagút felépítésének állapotait mutatja. Bejáratot az `addTunnelEntrance(x1,y1,x2,y2)` függvénnyel lehet létrehozni, míg lebontani a `removeTunnelEntrance(tE)`-el.



8. ábra Alagút építés state chart

3.7 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.02.20. 13:00	3 óra	Tóth Kunkli Zahorán Zoltay	Értekezlet. Osztályok és kapcsolatuk átbeszélése.
2017.02.23. 14:00	2 óra	Zahorán	Objektum katalógus
2017.02.26. 13:00	2 óra	Kunkli	Szekvencia diagrammok
2017.02.26. 15:00	2 óra	Zoltay	Osztálydiagram
2017.02.26. 17:00	2 óra	Tóth	State-chart, Osztályok leírása

4. Analízis modell kidolgozása

4 Analízis modell kidolgozása 2

4.1. Objektum katalógus

4.1.1 Model

A Model objektum a pálya leírásához és kezeléséhez tartozó elemeket gyűjti össze és végez rajtuk műveleteket a játék működése közben.

4.1.2 Engine

A mozdony felelőssége, hogy vezesse a vonatot a síneken, vontassa a vagonokat. A mozdony a játékban nem közlekedhet vagon nélkül, és minden esetben előre halad, azaz a vagonokat vontatja. Két mozdony ütközése esetén nem sikerül a pályát teljesíteni.

4.1.3 Car

A vagonok a vonat elején levő mozdonyt követik. Több vagon is kapcsolódhat egymáshoz. Minden szomszédos vagon színe különbözik egy szerelvényen és utasokat szállít. Az utasok csak a mozdonytól a szerelvény hátulja felé szállhatnak le a megadott helyeken.

4.1.4 Station

Az állomás jelképezi az utasok célját a játékban, ahová szeretnének eljutni. Az állomások színekkel vannak megkülönböztetve a pályán és minden esetben sín vezet hozzájuk. A megfelelő színű vagonban utazó utasok csak a megfelelő színű állomáson szállhatnak le amikor a vonat elhalad mellette.

4.1.5 Rail

A sín ábrázolja a vonatok lehetséges útvonalait. Több elem is kapcsolódhat a sínekhez, úgy mint: váltó, állomás, másik sín, alagút. A vonatok csak a síneken képesek az elemek között közlekedni.

4.1.6 Switch

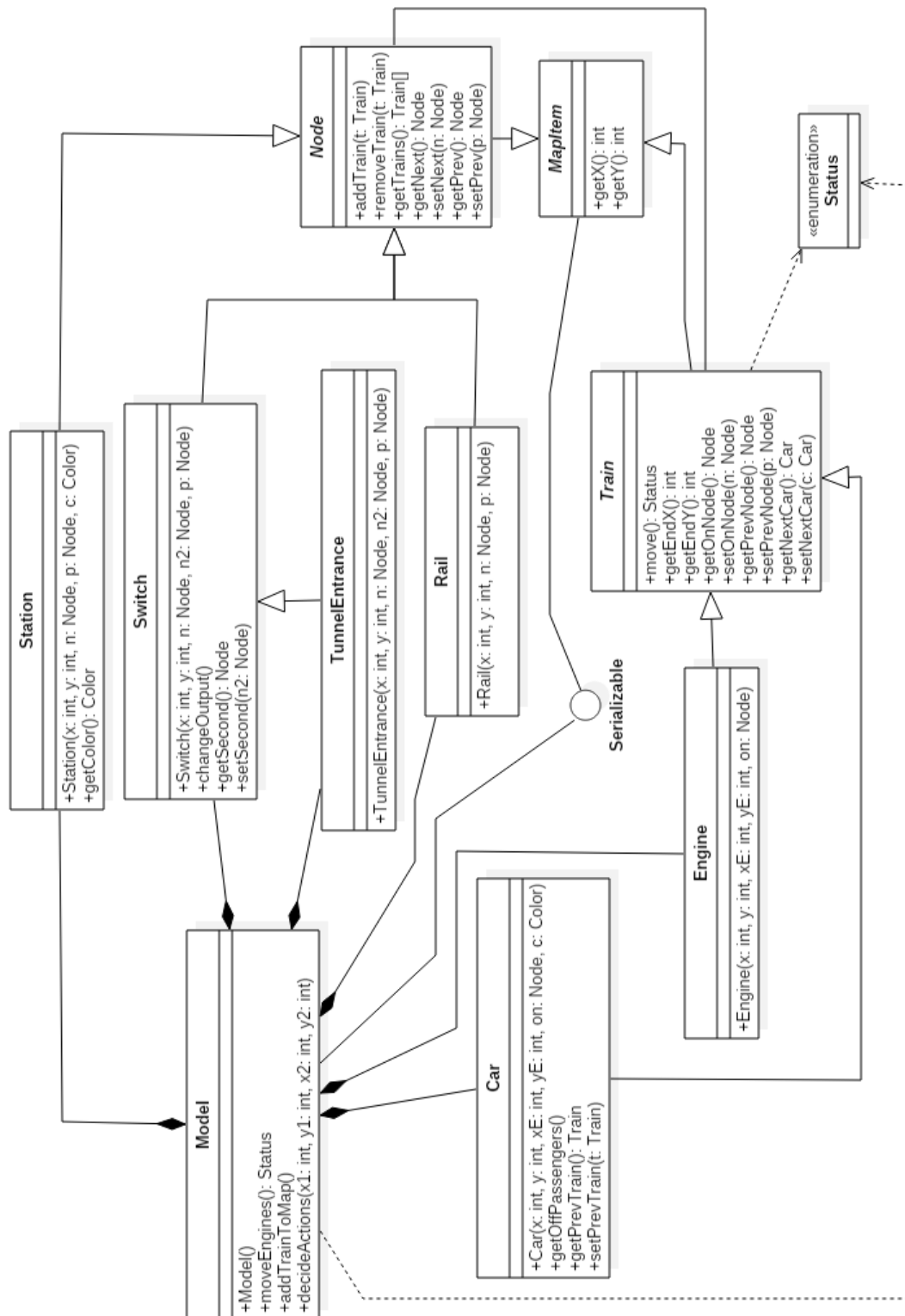
A váltó három sín elemet köt össze, két irány esetén csak a harmadikba közlekedhet rajta a vonat, viszont a harmadik irányból érkező a játékos állíthatja, hogy merre szeretné navigálni a vonatot. A váltók a pálya megkezdésekor mindig valamilyen alaphelyzetbe kerülnek, amelyek aztán megváltoztathatók, de nincs olyan eset, amikor nincs kiválasztott kimenete.

4.1.7 Tunnel

Az alagút lehetőséget ad a játékosnak, hogy adott helyzetből nem feltétlenül elérhető helyre irányítsa a vonatot. Legfeljebb két alagút bejárat lehet elhelyezve a pályán. Mind a két elem használható bejáratként és kijáratként is egyaránt, minden esetben amelyik elemen bement a

vonat a másikon jelenik meg. Amennyiben csak egy elem kerül elhelyezésre a pályán a vonat érkezéséig a vonat csak áthalad rajta. Az alagút bejáratot csak sínen lehet elhelyezni.

4.2 Statikus struktúra diagramok



9. ábra Osztálydiagram

4.3 Osztályok leírása

4.3.1 Car

- **Felelősség**

A vagon megvalósító osztály. A vagonnak színe van. A rajta utazó összes utas a vagon színével azonos színű állomáson leszáll, ha az a mozdonyhoz legközelebbi nem üres vagon.

- **Ősosztályok:** MapItem, Train

- **Interfészek:** -

- **Attribútumok:** -

- **Metódusok**

- **Car(x: int, y: int, xE: int, yE: int, on: Node, c: Color):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
- **void getOffPassengers():** Az összes utast leszállítja a vagonból.
- **Status move():** A vagon mozgatja. Visszatér egy állapot enum-mal.
- **void setPrevTrain(t: Train):** Beállítja az előtte álló kocsit, ha az első kocsi, akkor a mozdonyt.
- **Train getPrevTrain():** Visszatér az előtte álló kocsival, ha az első kocsi, akkor a mozdonyal.

4.3.2 Engine

- **Felelősség**

Az Engine osztály valósítja meg a mozdonyt. A vagonokat vontatásáért felel.

- **Ősosztályok:** MapItem, Train

- **Interfészek:** -

- **Attribútumok:** -

- **Metódusok**

- **Engine(x: int, y: int, xE: int, yE: int, n: Node):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
- **Status move():** Mozgatja a mozdonyt. Meghívja a vagonok move() metódusát. Visszatér egy állapot enum-mal.

4.3.3 Node

- **Felelősség**

Absztrakt osztály, a csomópontokat valósítja meg. Ezek segítségével lesz bejárható a pálya, tárolja a következő és előző elemet, így a váltók, sínek, állomások, összekapcsolhatóak.

- **Ősosztályok:** MapItem

- **Interfészek:** -

- **Attribútumok:** -

- **Metódusok**

- **void addTrain(t: Train):** Hozzáadja a paraméterül kapott mozdonyt vagy kocsit a tömbhöz.

- **void removeTrain(t: Train):** Kitörli a paraméterül kapott mozdonyt vagy kocsit a tömbből.
- **Train[] getTrains():** Visszatér a tömbbel.
- **Node getNext():** A következő csatlakoztatott elemmel tér vissza.
- **void setNext(n: Node):** Beállítja a következő csomópontot.
- **Node getPrev():** Az előző csatlakoztatott elemmel tér vissza.
- **void setPrev(p: Node):** Beállítja az előző csomópontot.

4.3.4 MapItem

- **Felelősség:**

Absztrakt osztály, a pályát alkotó elemek őssztálya, rögzíti azok helyét a pályán.

- **Őssztályok:** -
- **Interfészek:** **Serializable**
- **Attribútumok:** -
- **Metódusok**
 - **int getX():** Visszatér az elem x koordinátájának értékével.
 - **int getY():** Visszatér az elem y koordinátájának értékével.

4.3.5 Model

- **Felelősség**

Összefogja a rendszerben levő egységeket. Meghívja a mozdonyok move() metódusát. Lehetőséget biztosít alagút bejáratok építéséhez, rombolásához, illetve a váltók állításához.

- **Őssztályok:** -
- **Interfészek:** **Serializable**
- **Attribútumok:** -
- **Metódusok**
 - **Model():** A konstruktor létrehozza a model objektumot.
 - **Status moveEngines():** Meghívja a mozdonyok move() metódusát. Visszatér egy állapot enum-mal.
 - **addTrainToMap():** Új vonatot indít a pályán.
 - **decideActions(x1: int, y1: int, x2: int, y2: int):** A kapott paraméterek alapján eldönti az elvégzendő műveletet. Ez lehet alagút építés, rombolás, váltó állítása.

4.3.6 Rail

- **Felelősség**

A Rail osztály a síneket valósítja meg. Meghatározza a vonatok mozgásának irányát.

- **Őssztályok:** **MapItem, Node**
- **Interfészek:** -
- **Attribútumok:** -
- **Metódusok**
 - **Rail(x: int, y: int, n: Node, p: Node):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.

4.3.7 Station

- **Felelősség**

A Station osztály a megállót valósítja meg. A megálló színével azonos színű vagonokból az utasok leszállnak itt.

- **Ősosztályok:** MapItem, Node

- **Interfészek:** -

- **Attribútumok:** -

- **Metódusok**

- **Station(x: int, y: int, n: Node, p: Node, c: Color):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
- **Color getColor():** Visszatér az állomás színével.

4.3.8 Switch

- **Felelősség**

A Switch osztály a váltót valósítja meg a pályán. A két választható kimenete közül a beállítottra irányítja a vonatot. Minden esetben van kiválasztott kimenete, nincs holtponja.

- **Ősosztályok:** MapItem, Node

- **Interfészek:** -

- **Attribútumok:** -

- **Metódusok**

- **Switch(x: int, y: int, n1: Node, n2: Node, p: Node):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
- **void changeOutput():** Megváltoztatja a kiválasztott kimenetet.
- **Node getSecond():** Megadja mi áll a váltó másik kimenetén.
- **void setSecond(n: Node):** Beállítja a másik kimenetet.

4.3.9 TunnelEntrance

- **Felelősség**

A TunnelEntrance osztály valósítja meg az alagút bejáratot a pályán. Két lerakott bejárat esetén alkot alagutat és irányítja át a vonatot az egyik bejáratból a másikba. Amennyiben nincs második bejárat lerakva a vonat áthalad rajta.

- **Ősosztályok:** MapItem, Node, Switch

- **Interfészek:** -

- **Attribútumok**

- **Metódusok**

- **TunnelEntrance(x: int, y: int, n1: Node, n2: Node, p: Node):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.

4.3.10 Train

- **Felelősség:**

Absztrakt osztály, ami a MapItem osztályból származik. A vonatot alkotó elemek ősosztálya. Ebből származnak az Engine, Car és CoalCar osztályok.

- **Ősosztályok:** MapItem

- **Interfészek:** -

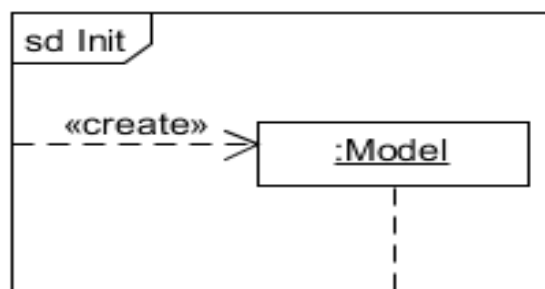
- **Attribútumok:** -

- **Metódusok**

- **Status move():** Mozgatja a vonatot és visszatér egy Status enum-mal.
- **int getEndX():** Visszatér az objektum végének x koordinátájával.
- **int getEndY():** Visszatér az objektum végének y koordinátájával.
- **Node getOnNode():** Visszaadja azt a Node-t, amelyiken a vonat aktuális eleme éppen tartózkodik.
- **void setOnNode(n: Node):** Beállítja a vonat elemének azt a Node-t, amelyiken éppen tartózkodik.
- **Node getPrevNode():** Visszaadja azt a Node-t, amelyikről jött a vonat eleme.
- **void setPrevNode(p: Node):** Beállítja a vonat elemének azt a Node-t, amelyikről jött a vonat eleme.
- **Train getNextCar():** Visszaadja a vonat elemének a következő kocsiját.
- **void setNextCar(t: Train):** Beállítja a paraméterül kapott Train-t a vonat elemének a következő kocsijának.
- **Color getColor():** Visszaadja a vonat egy elemének a színét.

4.4 Szekvencia diagramok

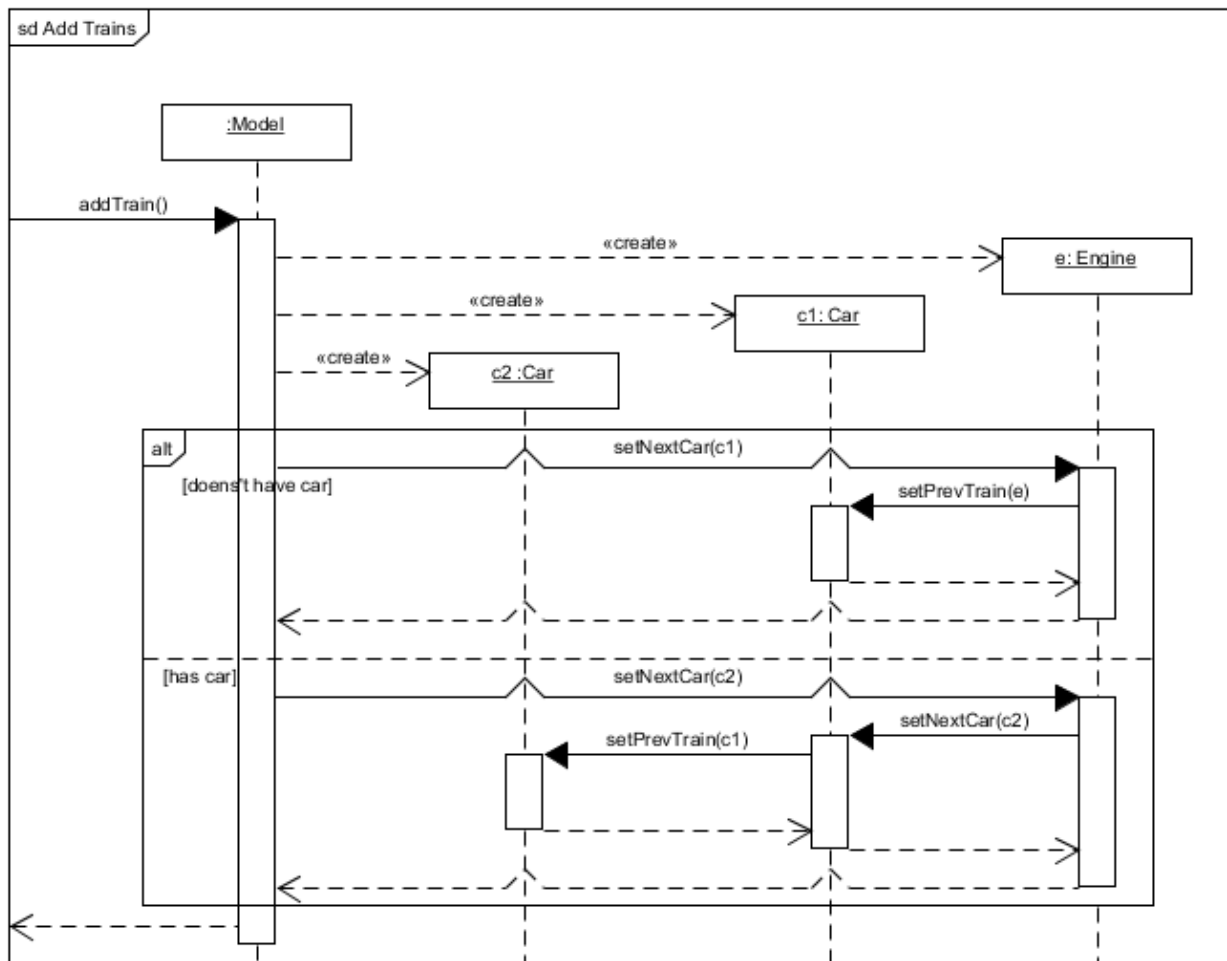
4.4.1 Init



10. ábra Init szekvencia diagram

Az inicializálás során a pályát egy privát metódus fogja beolvasni egy előre létrehozott szerializált fájlból, így csupán a model létrehozása a feladat ebben a lépésben.

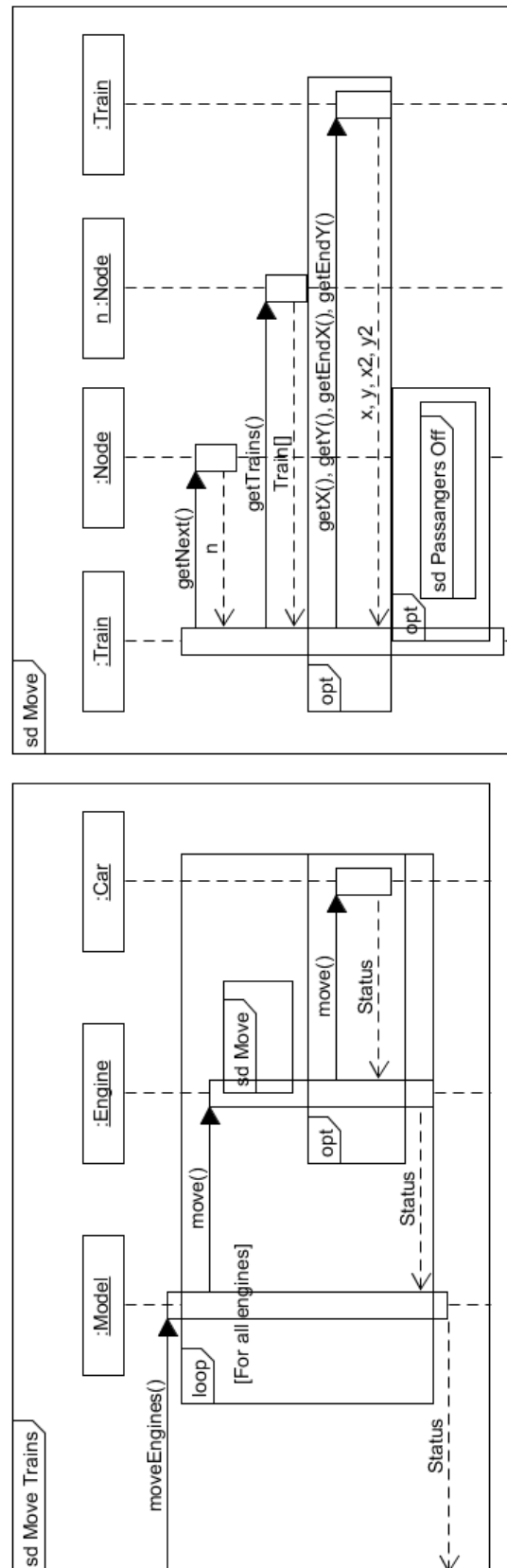
4.4.2 Add Trains



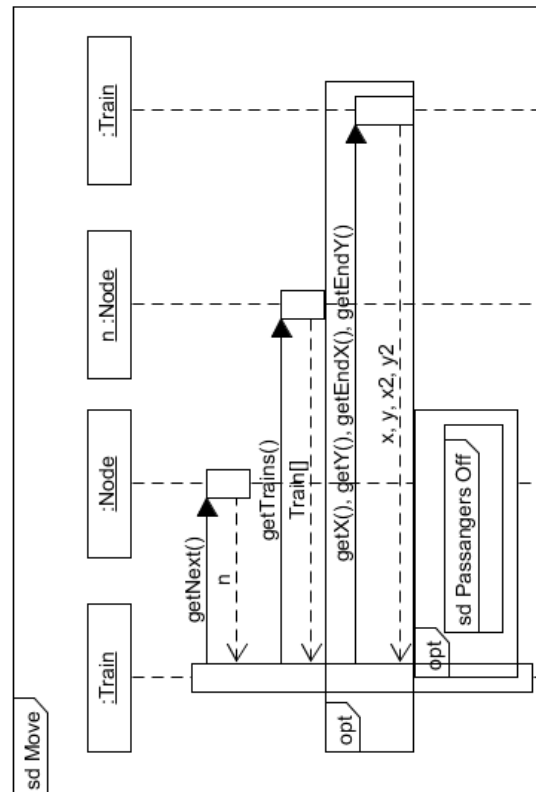
A Model létrehoz valamennyi vagon, és átadja mindet egyesével a korábban konstruált mozdonyoknak. Ha a mozdony már össze van kapcsolva egy elemmel, akkor tovább adja a vagon a hozzá kapcsolt elemeknek, és így tovább addig, amíg el nem érjük a lánc végét. Ha még nincs, akkor összeköti magával.

11. ábra Vonat hozzáadás szekvencia diagram

4.4.3 Move Trains



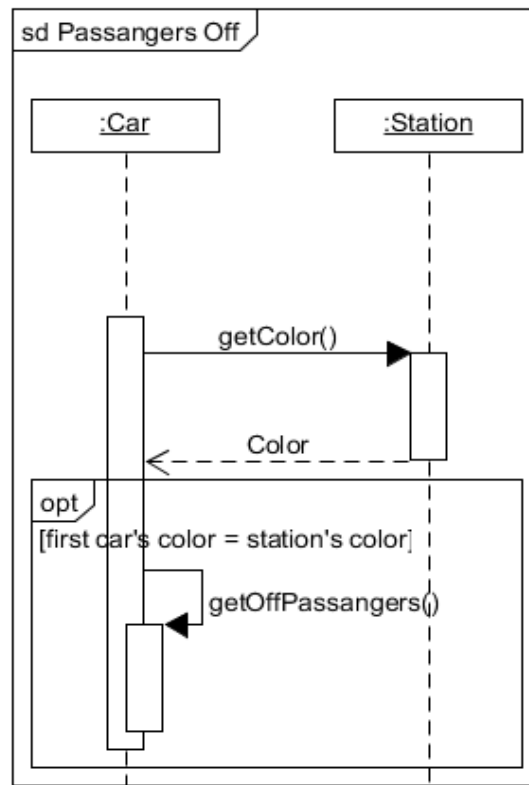
Ha a mozdony ütközést észlel, akkor nem hívja meg a következő hozzá kapcsolt elem mozgató függvényét



A vonat megnézi, hogy annak a csomópontnak, amin van, van-e a közelében olyan csomópont, amint van másik vonat. Ha van, lekéri a koordinátáit és megnézi, hogy ütköztek-e.

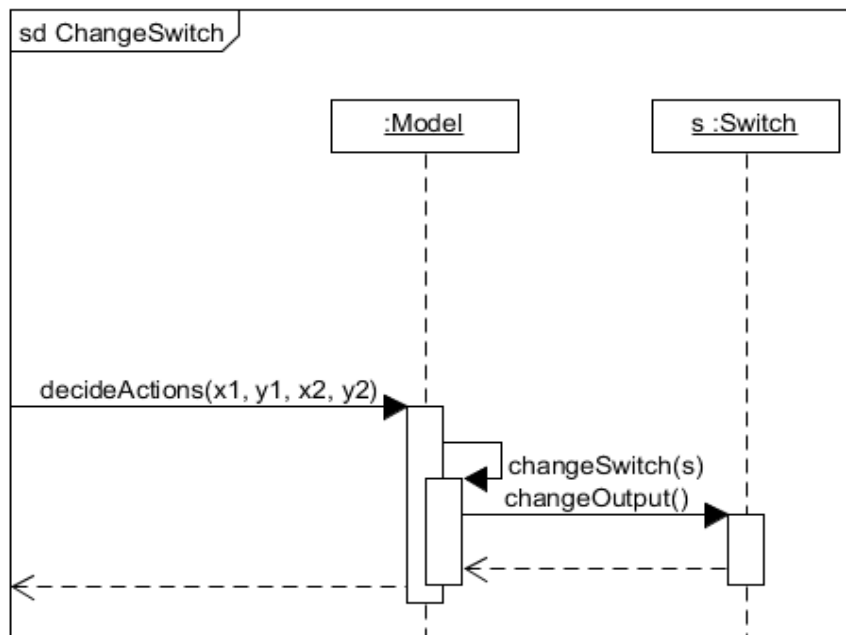
Ha az első nemüres vagon állomáson áll, akkor leszállítja az utasokat.

4.4.4 Passengers Off



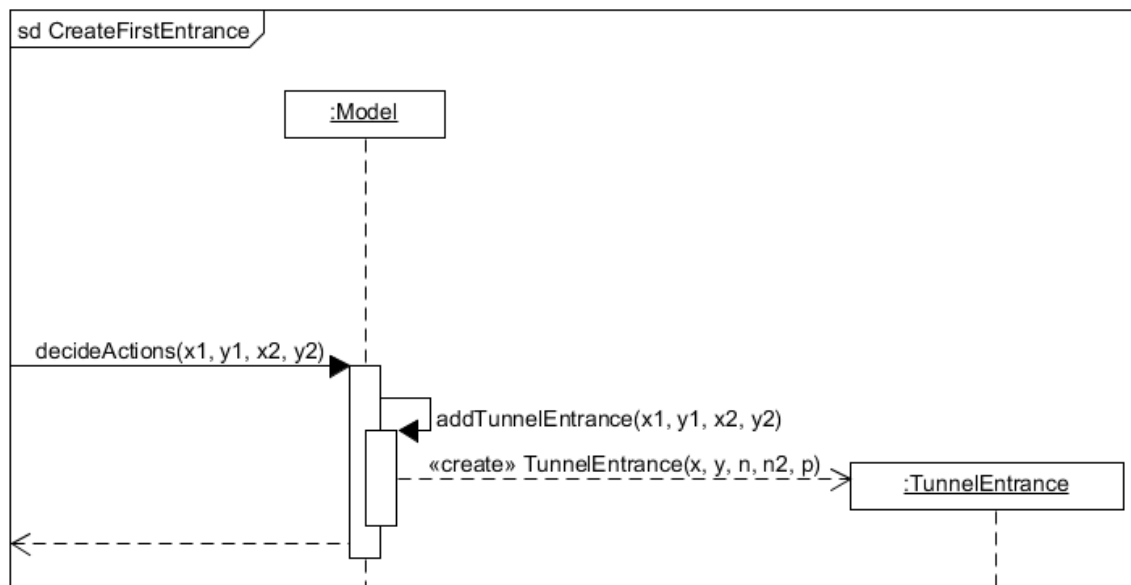
13. ábra Leszállás szekvencia diagram

4.4.5 Change Switch

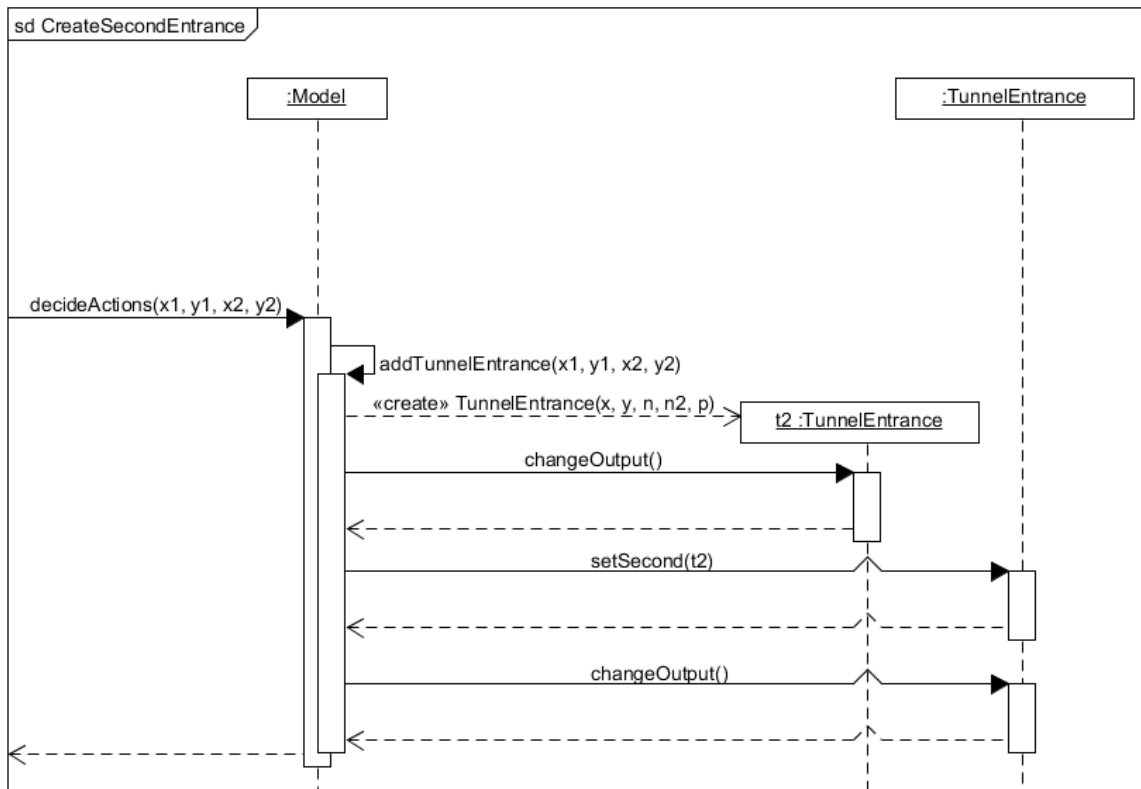


14. ábra Váltó állítás szekvencia diagram

4.4.6 Create Tunnel Entrance

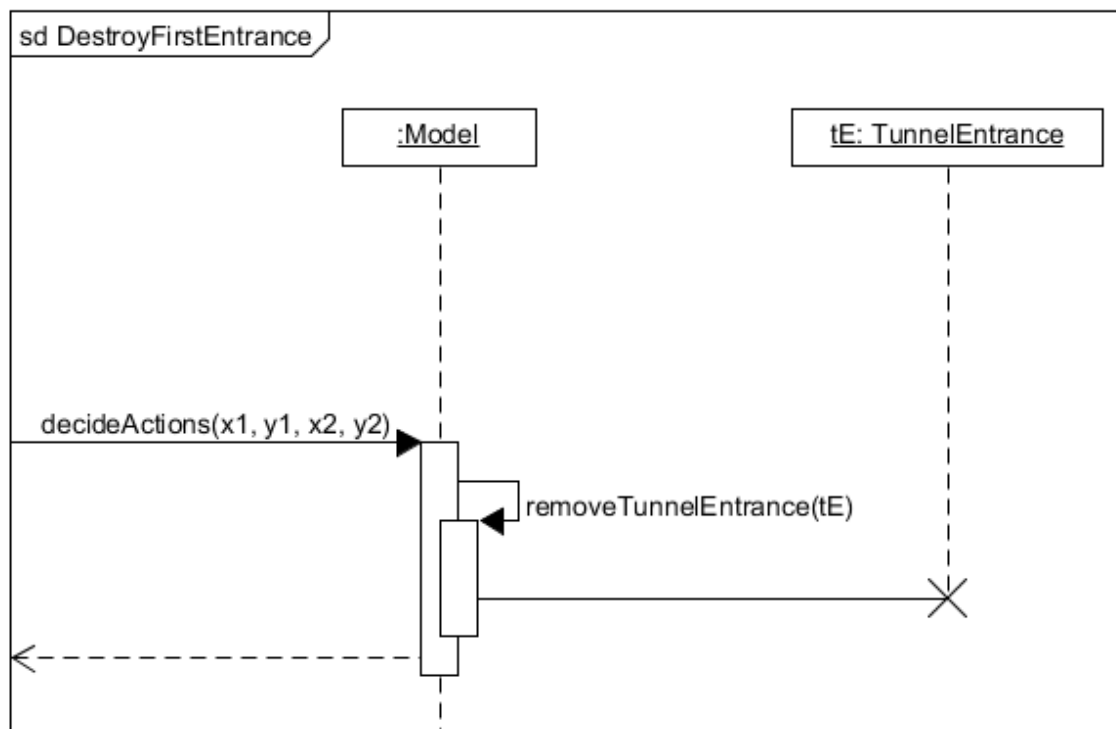


15. ábra Első alagút létrehozása szekvencia diagram

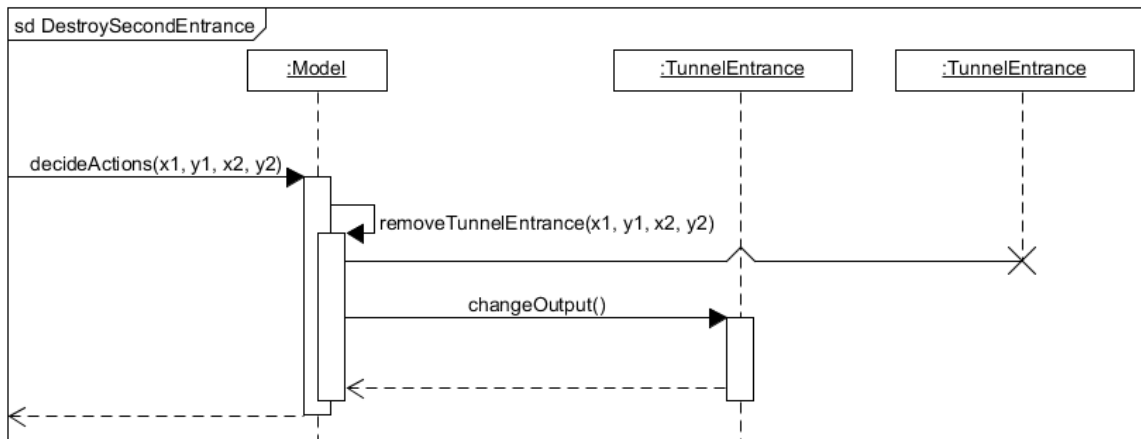


16. ábra Második alagút létrehozása szekvencia diagram

4.4.7 Destroy Tunnel Entrance



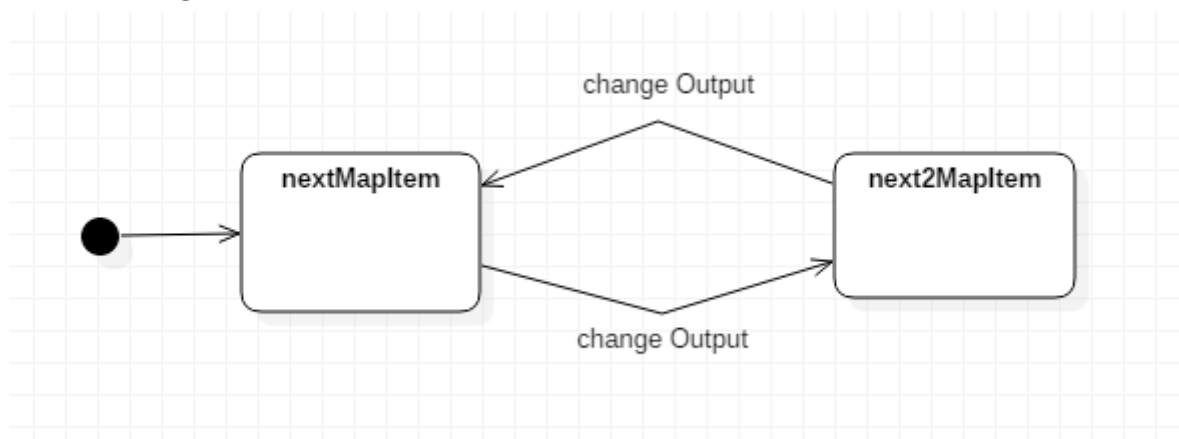
17. ábra Első alagút törlése szekvencia diagram



18. ábra Második alagút törlése szekvencia diagram

4.5 State-chartok

4.5.1 Change Switch

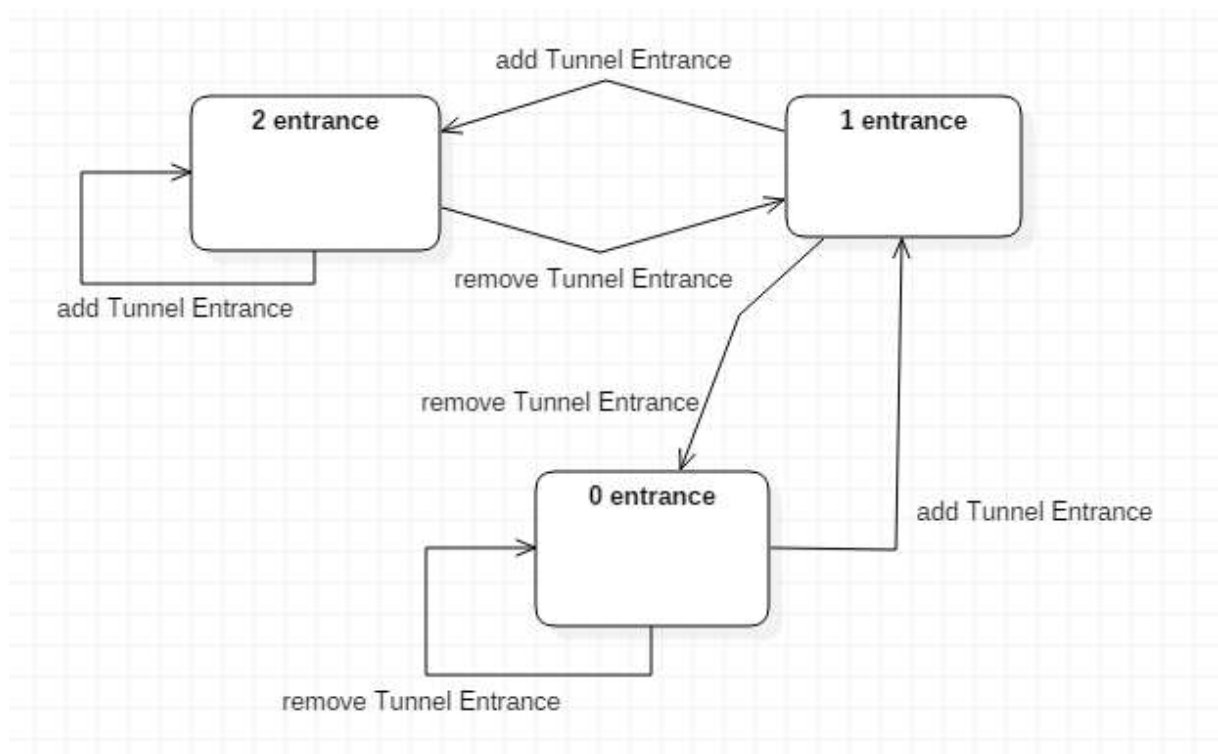


19. ábra Váltó állítás state-chart

A fenti állapot diagram a váltó állapotait mutatja. A váltó két kimenete között lehet váltani a change Output-tal. A váltót a játékos tudja állítani.

4.5.2 Create Tunnel Entrance

A lenti állapot diagram az alagút felépítésének állapotait mutatja. Bejáratot az add Tunnel Entrance-szel lehet létrehozni, míg lebontani a remove Tunnel Entrance-szel.



20. ábra Alagút létrehozás state-chart

4.6 Napló

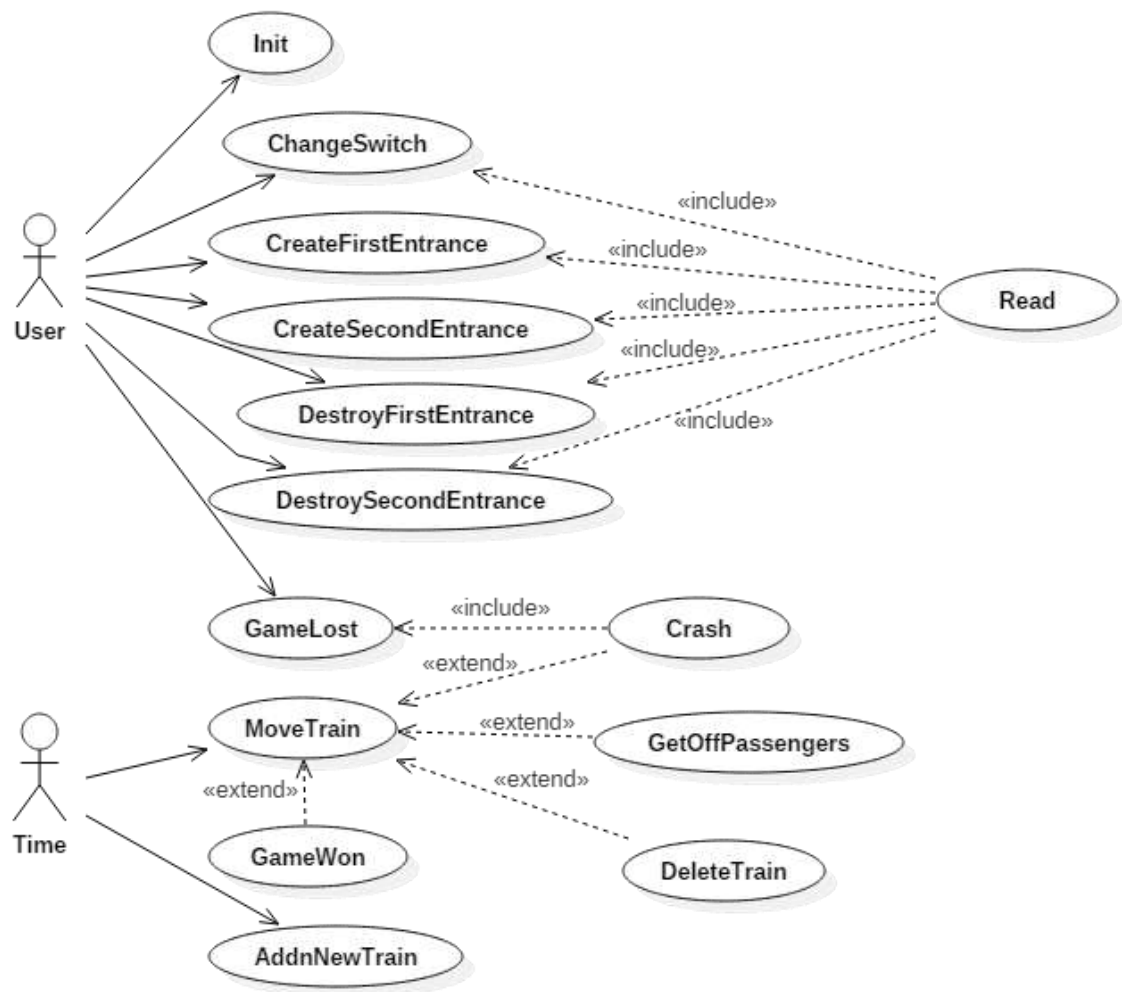
Kezdet	Időtartam	Résztevők	Leírás
2017.03.03. 13:00	4 óra	Tóth Kunkli Zahorán Zoltay	Értekezlet. Osztályok és kapcsolatuk átbeszélése.
2017.03.05. 14:00	2 óra	Zahorán	Objektum katalógus módosítása.
2017.03.05. 14:00	2 óra	Kunkli	Szekvencia diagrammok módosítása és hozzáadása.
2017.03.05. 15:00	2 óra	Zoltay	Osztálydiagram módosítása.
2017.03.05. 17:00	2 óra	Tóth	State-chartok és osztályok leírásának módosítása.

5. Szkeleton tervezése

5 Szkeleton tervezése

5.1 A szkeleton modell valóságos use-case-ei

5.1.1 Use-case diagram



21. ábra Szkeleton Use-case diagram

5.1.2 Use-case leírások

Use-case neve	Init
Rövid leírás	Betölti a pályát, elindítja a játékot.
Aktorok	User

Use-case neve	ChangeSwitch
Rövid leírás	A felhasználó változtatja a váltó állását.
Aktorok	User
Forgatókönyv	Az aktív jelző az egyik kimenetről a másikra kerül.

Use-case neve	CreateFirstEntrance
Rövid leírás	A felhasználó létrehozza az első alagút bejáratot.
Aktorok	User
Forgatókönyv	Az alagút bejárat a váltók és állomások kivételével, a sín bármely pontjára tehető. Az alagút bejáratának irányát be lehet állítani.

Use-case neve	CreateSecondEntrance
Rövid leírás	A felhasználó létrehozza a második alagút bejáratot.
Aktorok	User
Forgatókönyv	Az alagút bejárat a váltók és állomások kivételével, a sín bármely pontjára tehető. Az alagút bejáratának irányát be lehet állítani. Létrejön az összeköttetés a két alagút bejárat között.

Use-case neve	DestroyFirstEntrance
Rövid leírás	A felhasználó törli az első alagút bejáratot.
Aktorok	User
Forgatókönyv	Egy már meglévő alagút bejáratot töröl. Megszakad az összeköttetés a másik alagút bejáráttal.

Use-case neve	DestroySecondEntrance
Rövid leírás	A felhasználó törli a második alagút bejáratot.
Aktorok	User

Use-case neve	Read
Rövid leírás	Az utasítások megadása.
Aktorok	User
Forgatókönyv	Az utasítások megadása, mint például alagút bejárat építése, rombolása, váltó állítása.

Use-case neve	AddNewTrain
Rövid leírás	Új vonat hozzáadása.
Aktorok	Time
Forgatókönyv	Megfelelő időközönként új vonat indul a pályán.

Use-case neve	MoveTrain
Rövid leírás	A vonatok mozgása.
Aktorok	Time
Forgatókönyv	A pályán lévő vonatok megfelelő időközönként előre lépnek.

Use-case neve	GetOffPassengers
Rövid leírás	Leszállnak az utasok az adott kocsiból.
Aktorok	User
Forgatókönyv	A vonat mozgása során elért állomásokon, ha a vonat első nem üres kocsija azonos színű az állomásával, az utasok leszállnak.

Use-case neve	Crash
Rövid leírás	Két vonat ütközése.
Aktorok	User
Forgatókönyv	Két vonat összeütközik a pályán.

Use-case neve	GameLost
Rövid leírás	A játék elvesztése.
Aktorok	User
Forgatókönyv	Ha két vonat ütközik, a játékos veszít.

Use-case neve	DeleteTrain
Rövid leírás	Egy vonat törlése.
Aktorok	User
Forgatókönyv	Ha egy vonat összes kocsijáról leszálltak az utasok, a vonat törlődik a pályáról.

Use-case neve	GameWon
Rövid leírás	A játék megnyerése.
Aktorok	User
Forgatókönyv	Ha a pályáról az összes vonat törlődik, a játékos nyer.

5.2 A szkeleton kezelői felületének terve, dialógusok

Menü felépítése:

1. Váltó állítása
 1. *Szeretné átállítani a váltó kimenetét? I/N
2. Alagút létrehozása
 1. *Hány alagút van a pályán? 0/1/2
 2. *Érvényes helyre akarja rakni? I/N
3. Alagút törlése
 1. *Hány alagút van a pályán? 0/1/2
4. Vonat mozgatása
 1. *Van ütközés? I/N
 2. *Állomásra lép a vagon? I/N
 3. *Milyen színű a vagon? R/Green/B/Y/Gray
 4. *Milyen színű az állomás? R/Green/B/Y/Gray
5. Vonat hozzáadása
 1. *Szeretne vonatot adni a pályához? I/N

A *-gal jelölt menüpontokat nem lehet kiválasztani, ezt a program automatikusan megteszi, ha a felhasználó a parancs szülőjét meghívta.

A kérdések végén láthatóak a lehetséges válaszok.

A következő példa egy olyan esetet mutat be, amikor egy vagon az állomásra ér és az utasok elhagyják a vonatot.

? Adja meg a parancs kódját: 5

- 5. Leszállás

```
>      ->[st:Station].addCar(c1: Car):
>          ->[c1:Car].setOnNode(st: Station):
<          <-[c1:Car].setOnNode():
>          ->[c1:Car].getColor():
?          5.1. Milyen színű a vagon? R/Green/B/Y/Gray : R
<          <-[c1:Car].getColor():
>          ->[st:Station].getColor():
?          5.2. Milyen színű az állomás? R/Green/B/Y/Gray: R
<          <-[st:Station].getColor():
>          ->[c1:Car].getOffPassengers():
>          ->[c2:Car].getColor():
?          5.3. Előző vagon üres? I/N: I
<          <-[c2:Car].getColor()
<          <-[c1:Car].getOffPassengers()
<      <-[st:Station].addCar()
```

? Adja meg a parancs kódját:

Minden sor elején van 1 karakter, ami a sor típusát jelöli:

'?': kérdés, felhasználó beavatkozására vár

'>': metódusba lépés

'<': metódusból visszatérés

'-': megjegyzés

Továbbá minden metódus esetén

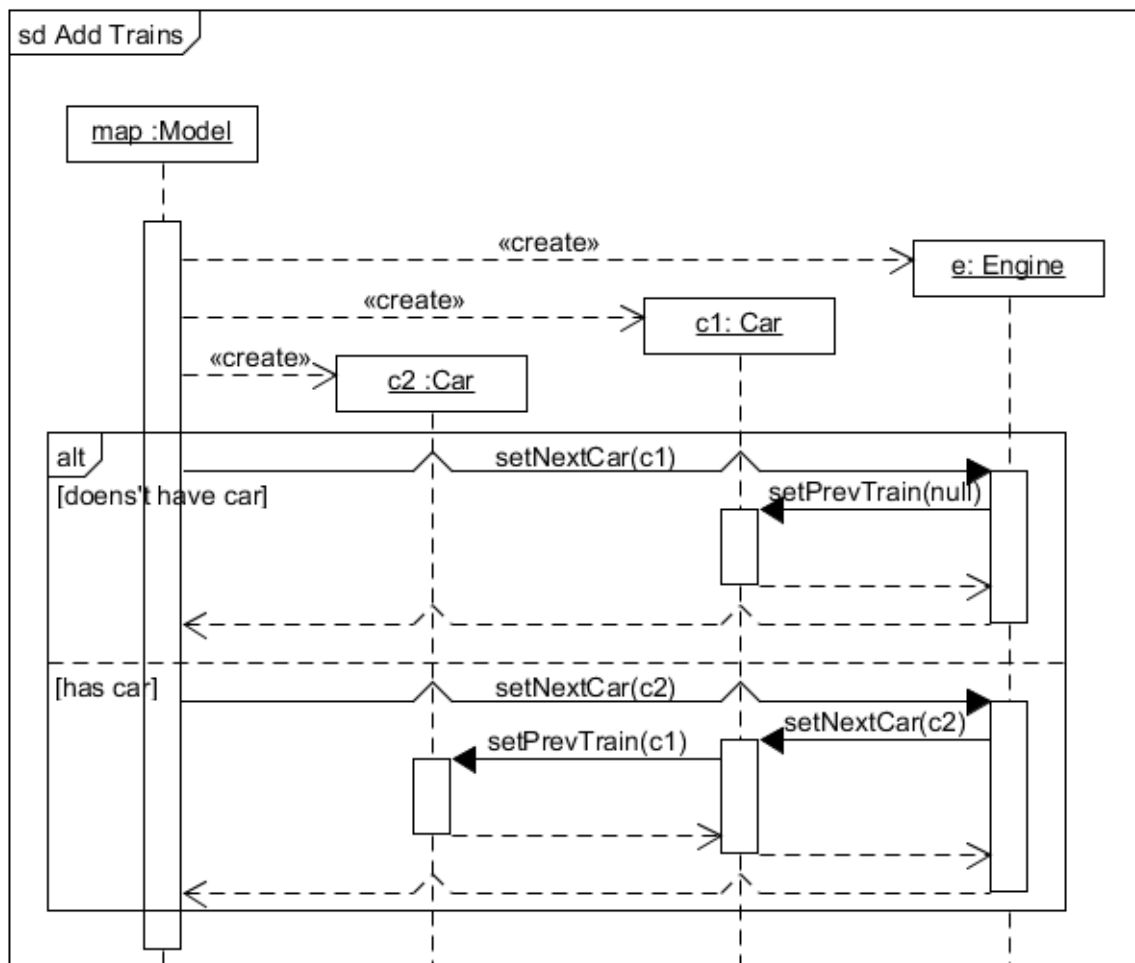
'->[példány :Osztály].metódus()': metódus elindulása

'<-[példány :Osztály].metódus()': metódusból visszatérése

fel van tüntetve.

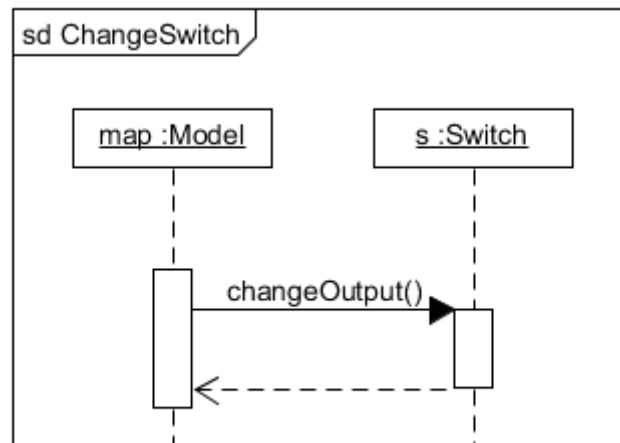
5.3 Szekvencia diagramok a belső működésre

5.3.1 Add Trains



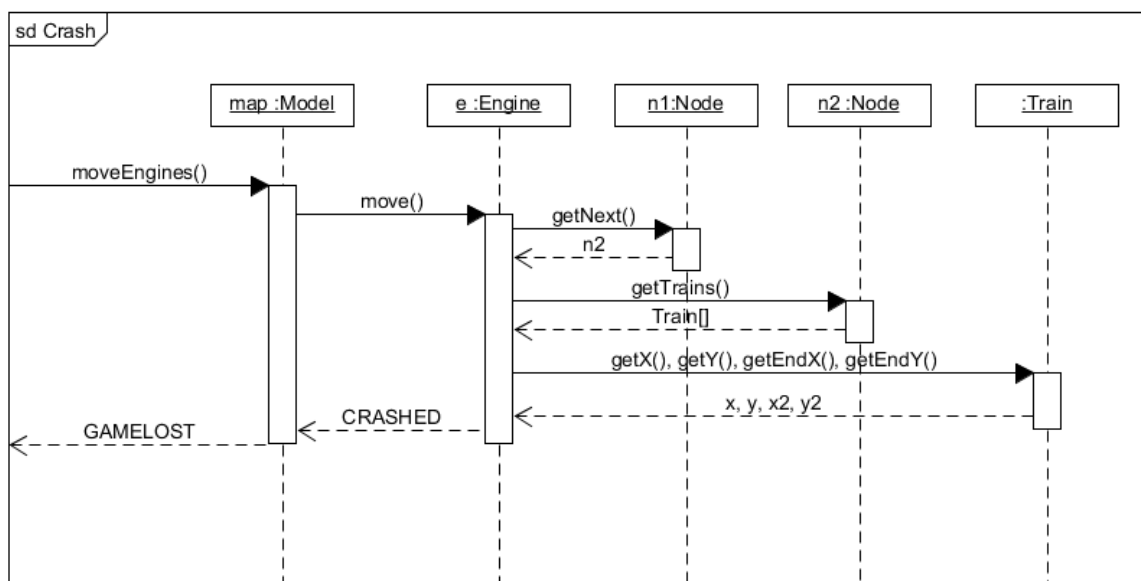
22. ábra Add Trains szekvencia diagram

5.3.2 Change Switch



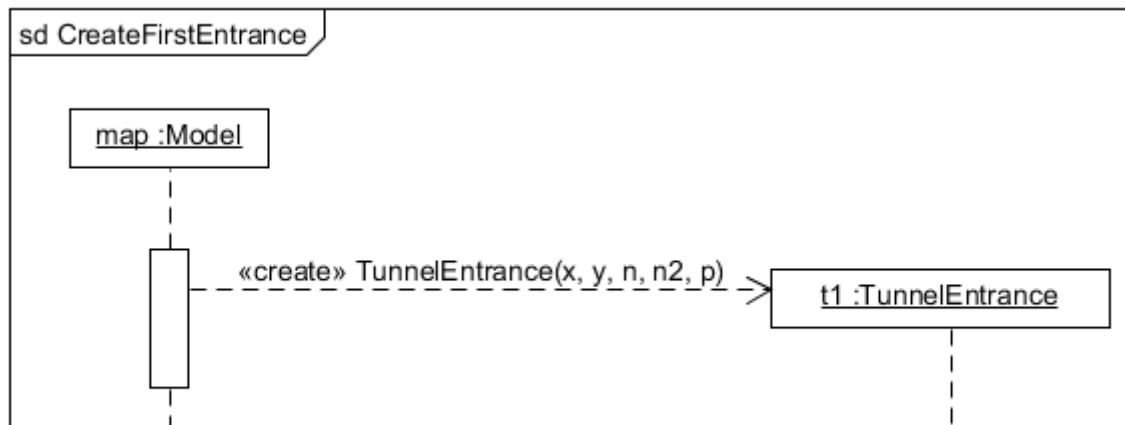
23. ábra Change Switch szekvencia diagram

5.3.3 Crash



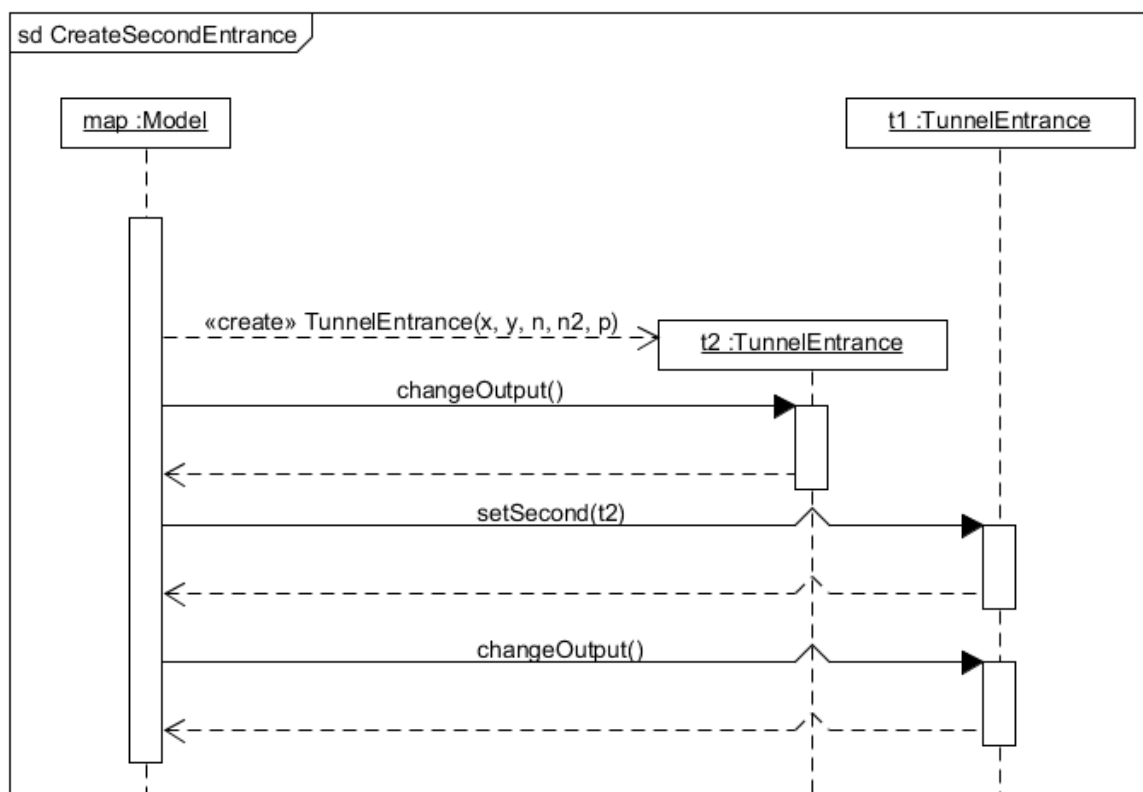
24. ábra Crash szekvencia diagram

5.3.4 Create First Entrance



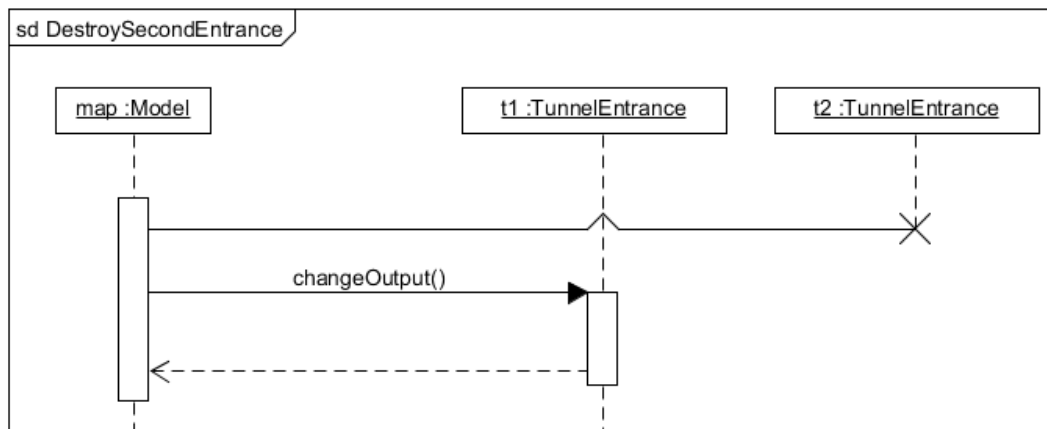
25. ábra Create First Entrance szekvencia diagram

5.3.5 Create Second Entrance



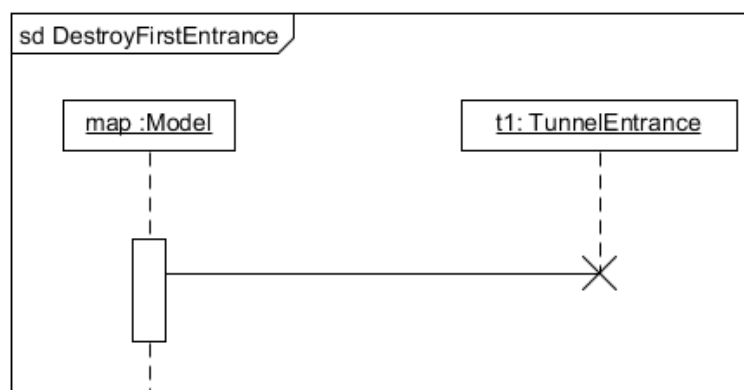
26. ábra Create Second Entrance szekvencia diagram

5.3.6 Destroy Second Entrance



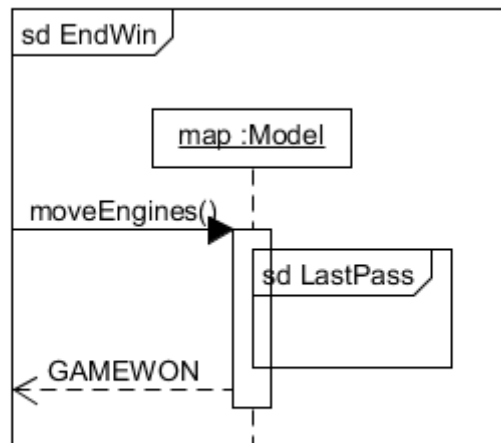
27. ábra Destroy Second Entrance szekvencia diagram

5.3.7 Destroy First Entrance



28. ábra Destroy First Entrance szekvencia diagram

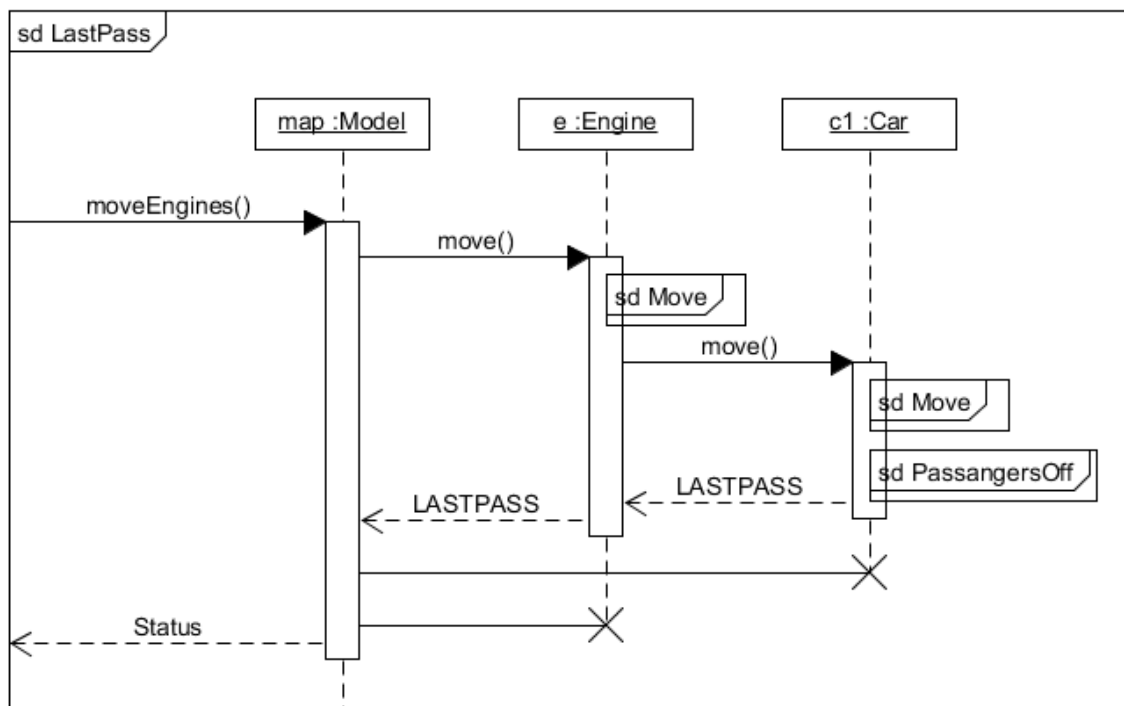
5.3.8 EndWin



29. ábra Játék befejezése győzelemmel szekvencia diagram

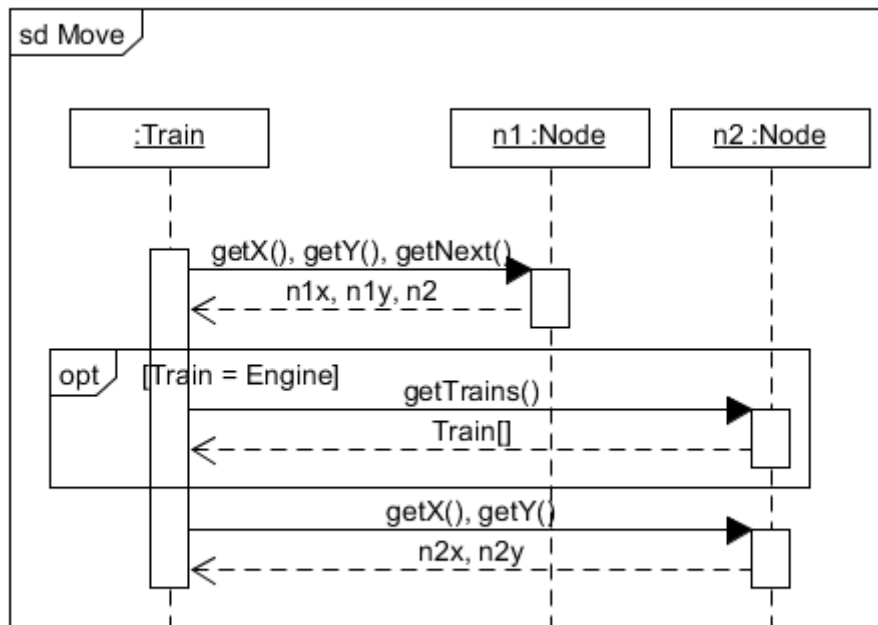
Miután a map, kap egy "LASTPASS" értéket, megnézi, hogy van-e még vonat a pályán, vagy lesz-e még hozzáadva új vonat, ha egyik feltétel sem igaz, akkor a játékos nyert.

5.3.9 LastPass



30. ábra LastPass szekvencia diagram

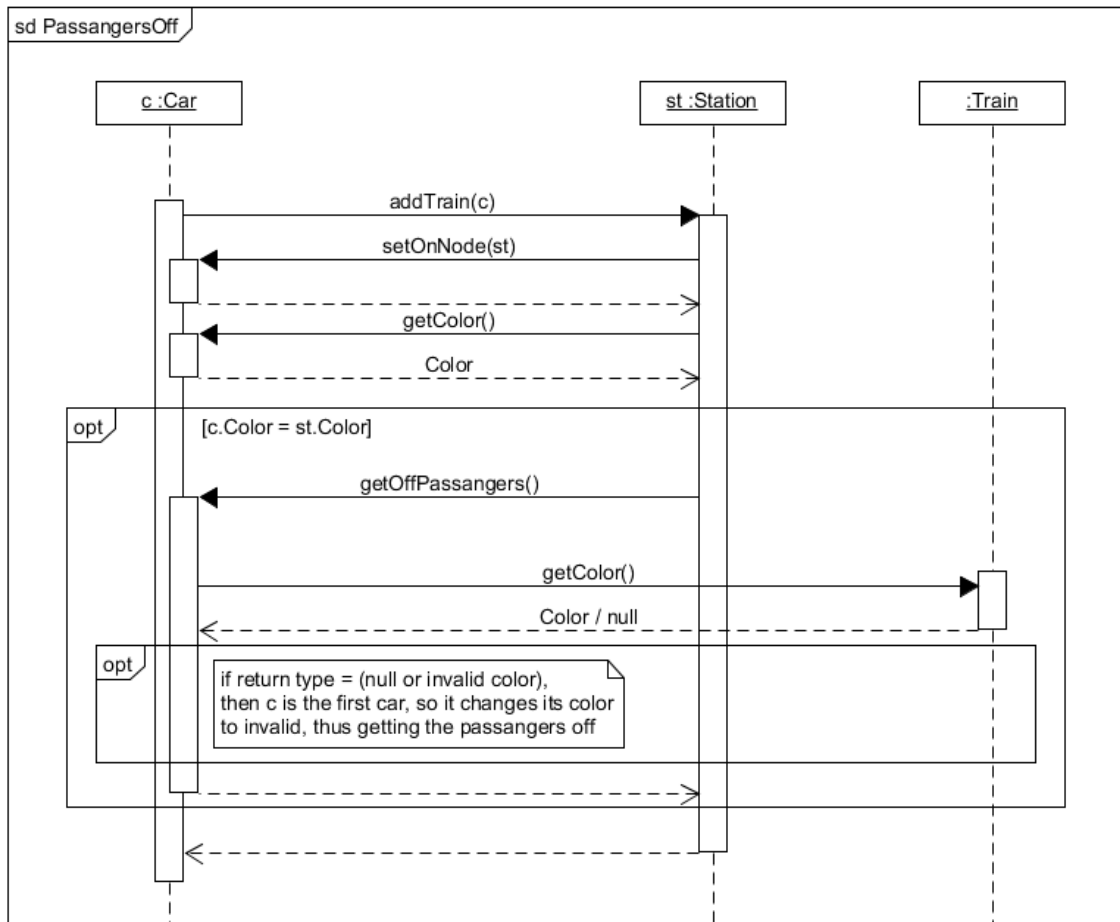
5.3.10 Move



31. ábra Move szekvencia diagram

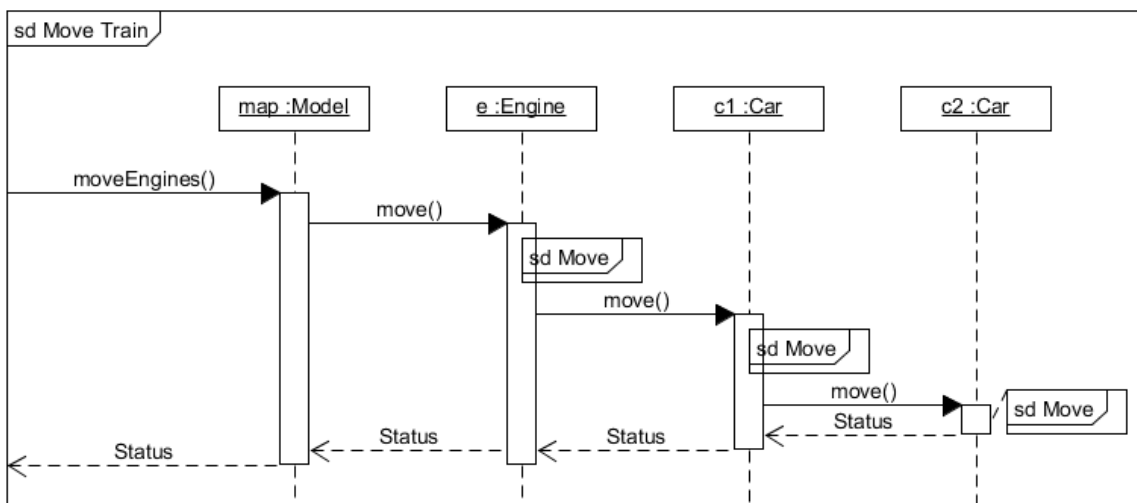
A Car move, és az Engine move függvénye ugyan az. Egyetlen eltérés, hogy az utóbbi lekérdezi a következő csomóponton álló vonatokat is, ahogy az a diagrammon is szerepel.

5.3.11 PassangersOff



32. ábra Passangers leszállás szekvencia diagram

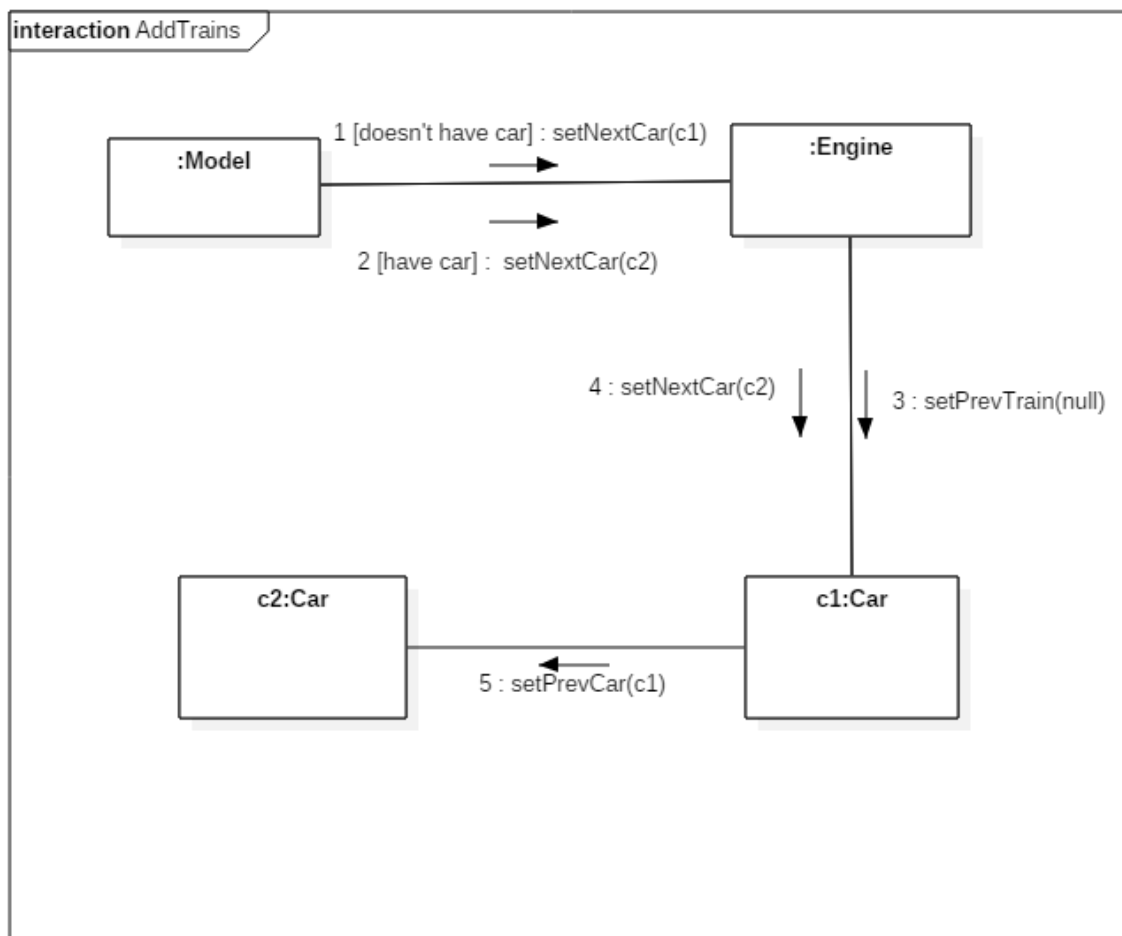
5.3.12 Move Train



33. ábra Move Trains szekvencia diagram

5.4 Kommunikációs diagramok

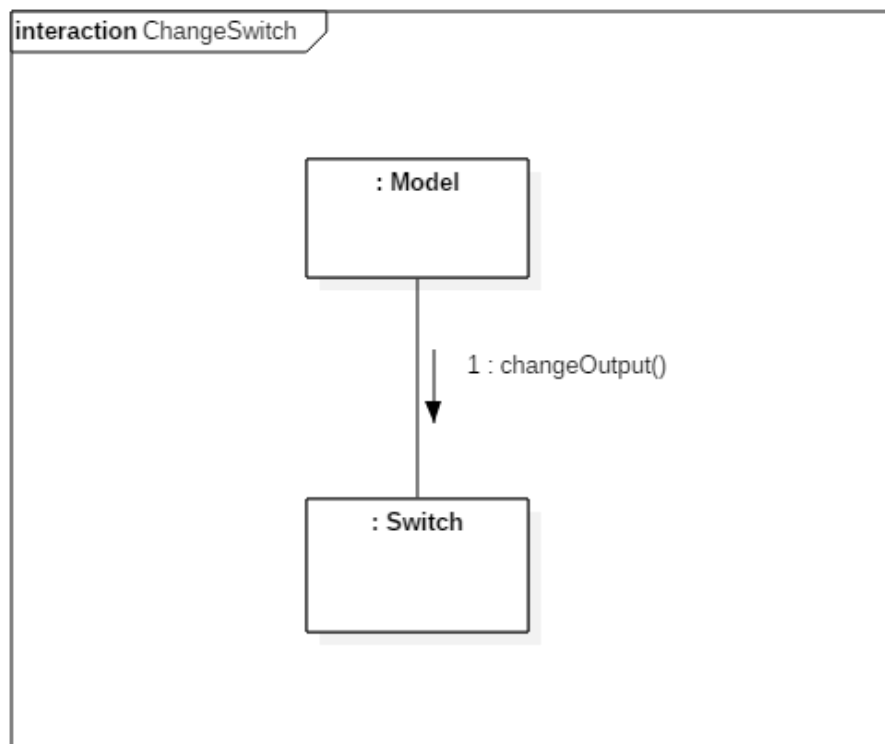
5.4.1 Add Trains



34. ábra Add Trains kommunikációs diagram

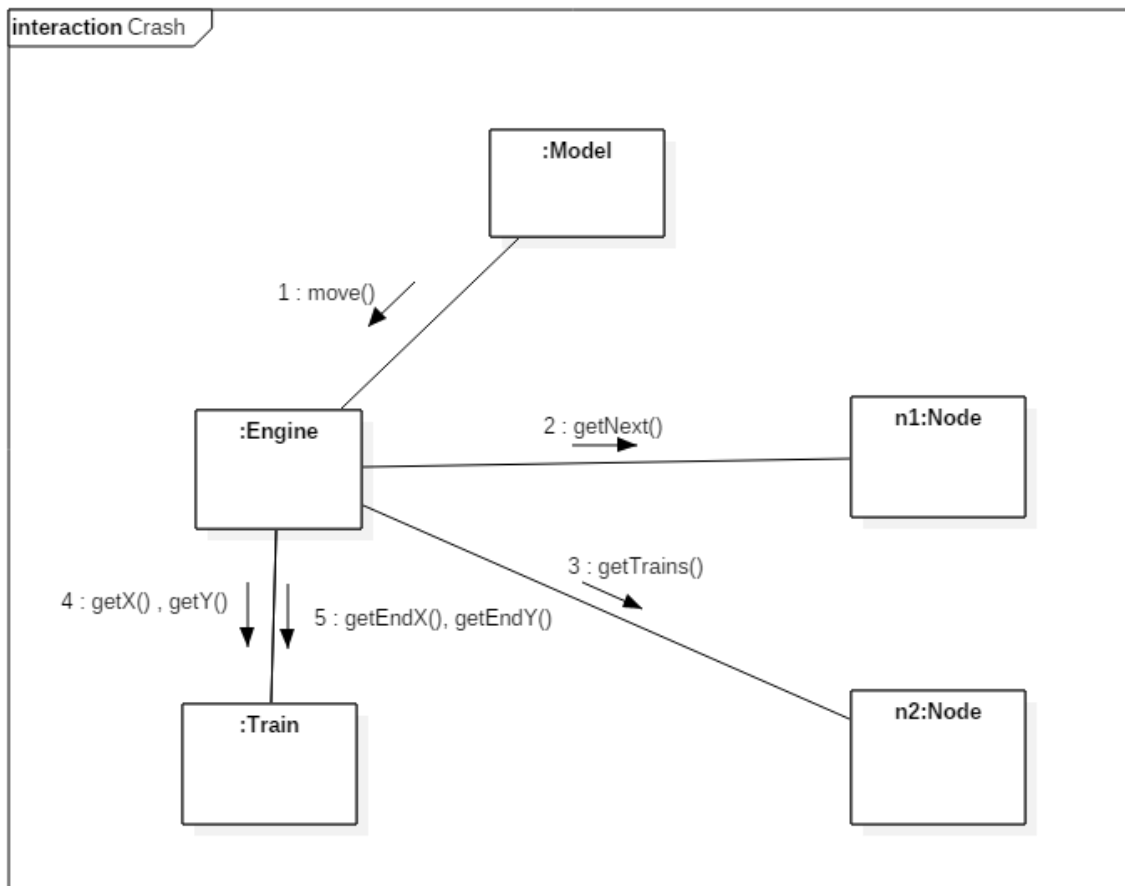
A hierarchikus számozást nem sikerült beállítani, de az összetartozó függvényhívások az 1. és 3. (1.1) ,illetve 2.,4.(2.1),5.(2.2).

5.4.2 Change Switch



35. ábra Change Switch kommunikációs diagram

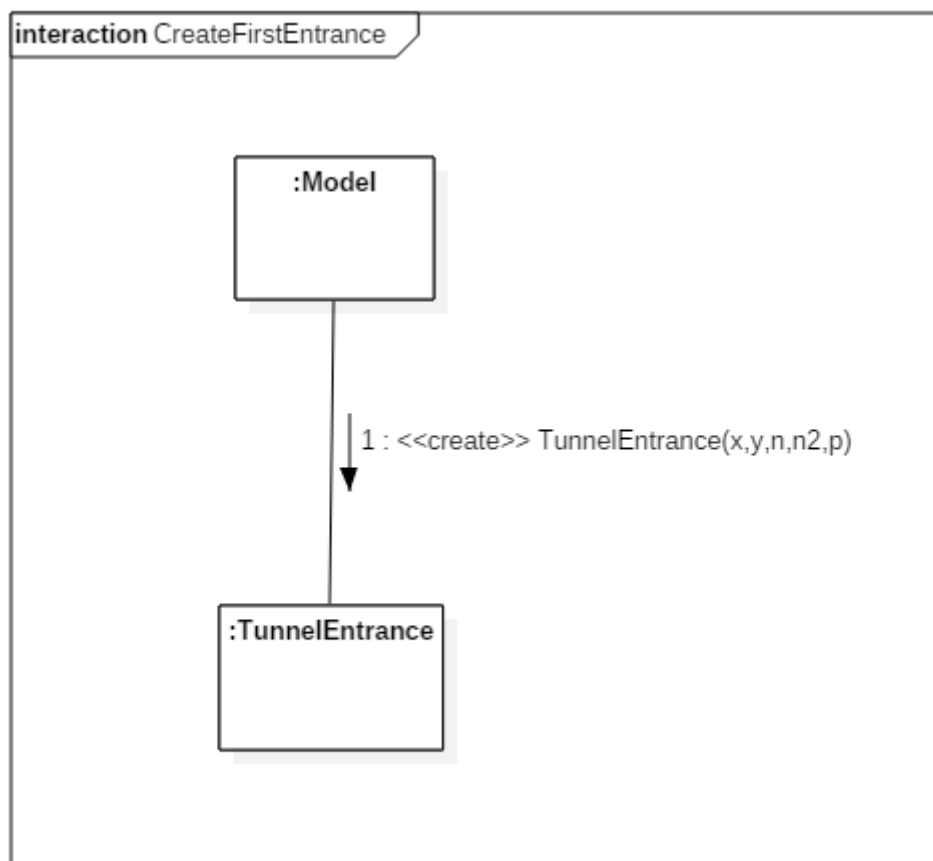
5.4.3 Crash



36. ábra Crash kommunikációs diagram

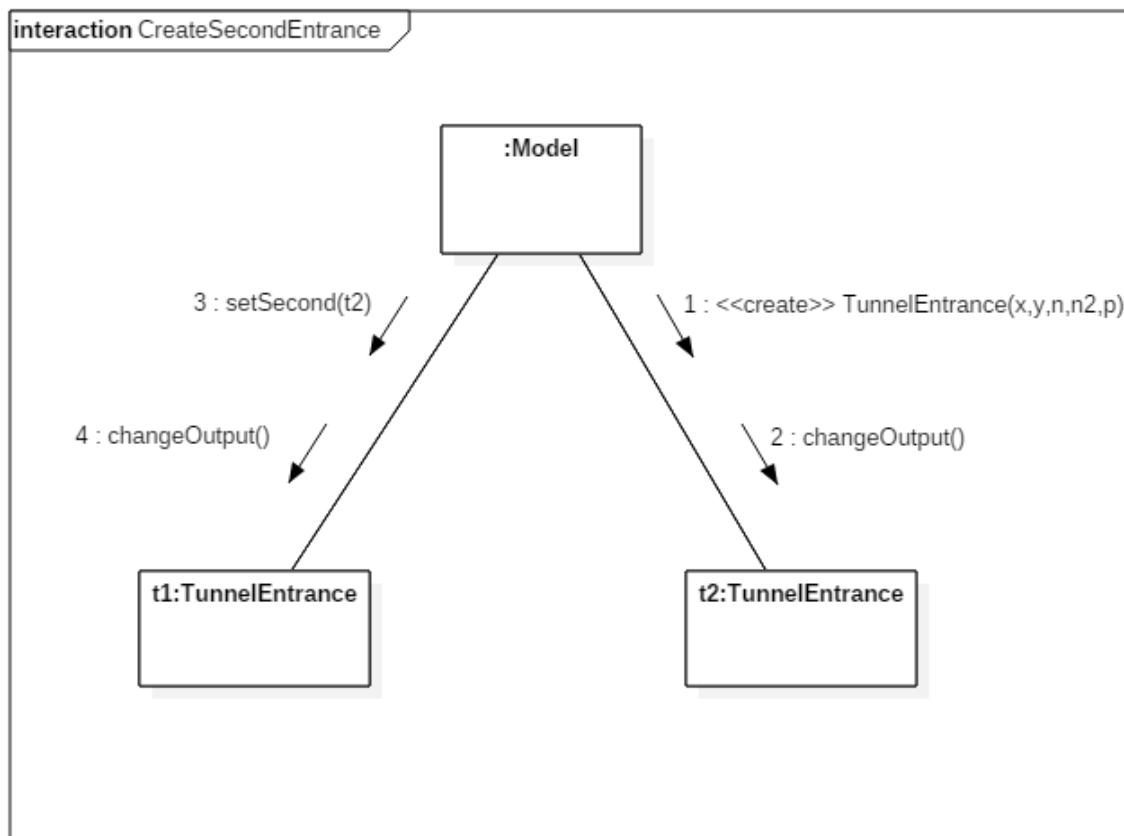
A fenti diagramon a Time actor `moveEngines()` függvénye fogja hívni a `Model`-t azonban a fejlesztőkörnyezet kommunikációs diagramba nem biztosított Actor-t.

5.4.4 Create First Entrance



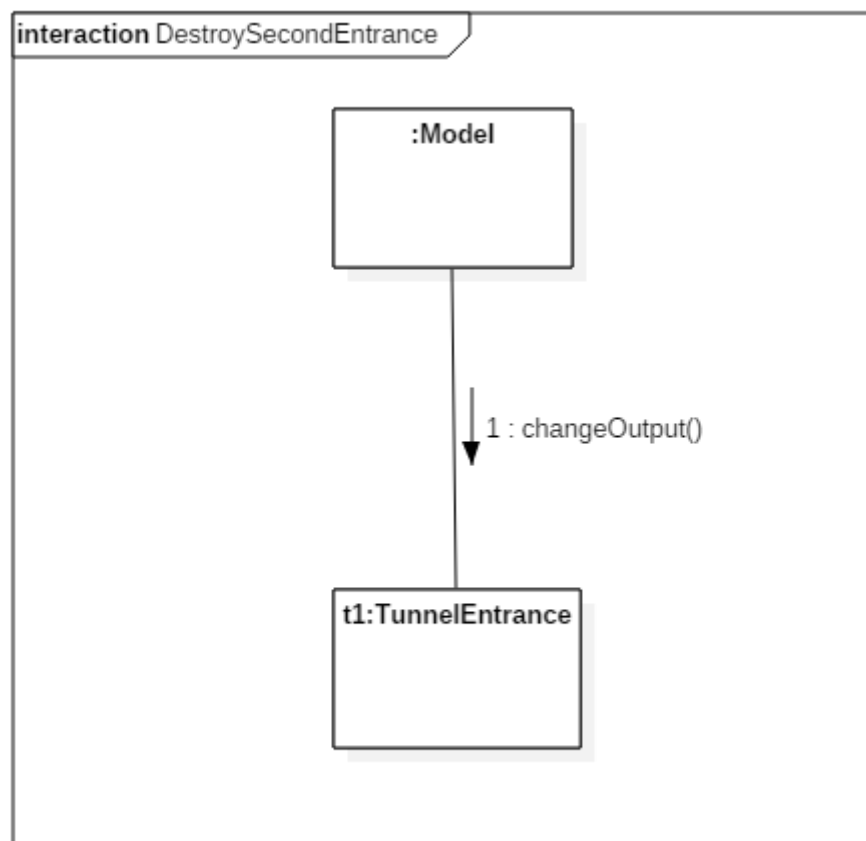
37. ábra Create First Entrance kommunikációs diagram

5.4.5 Create Second Entrance



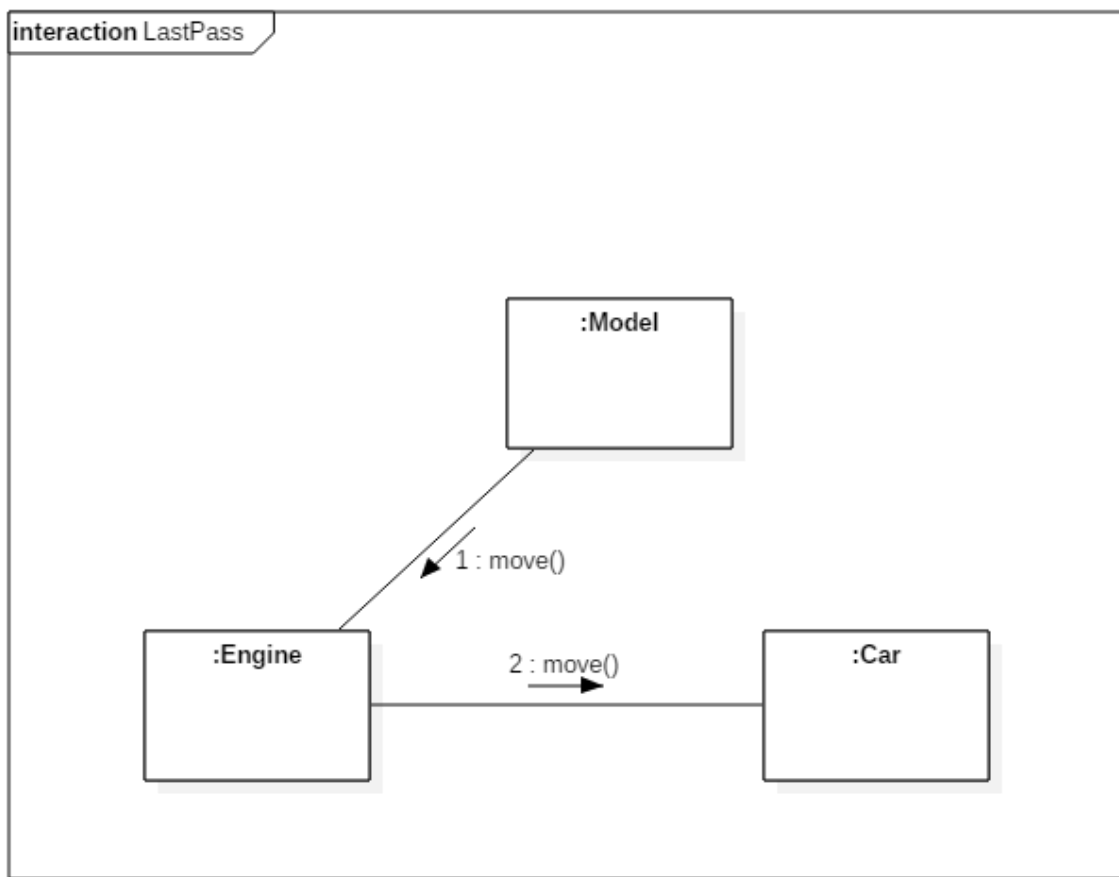
38. ábra Create Second Entrance kommunikációs diagram

5.4.6 DestroySecondEntrance



39. ábra Destroy Second Entrance kommunikációs diagram

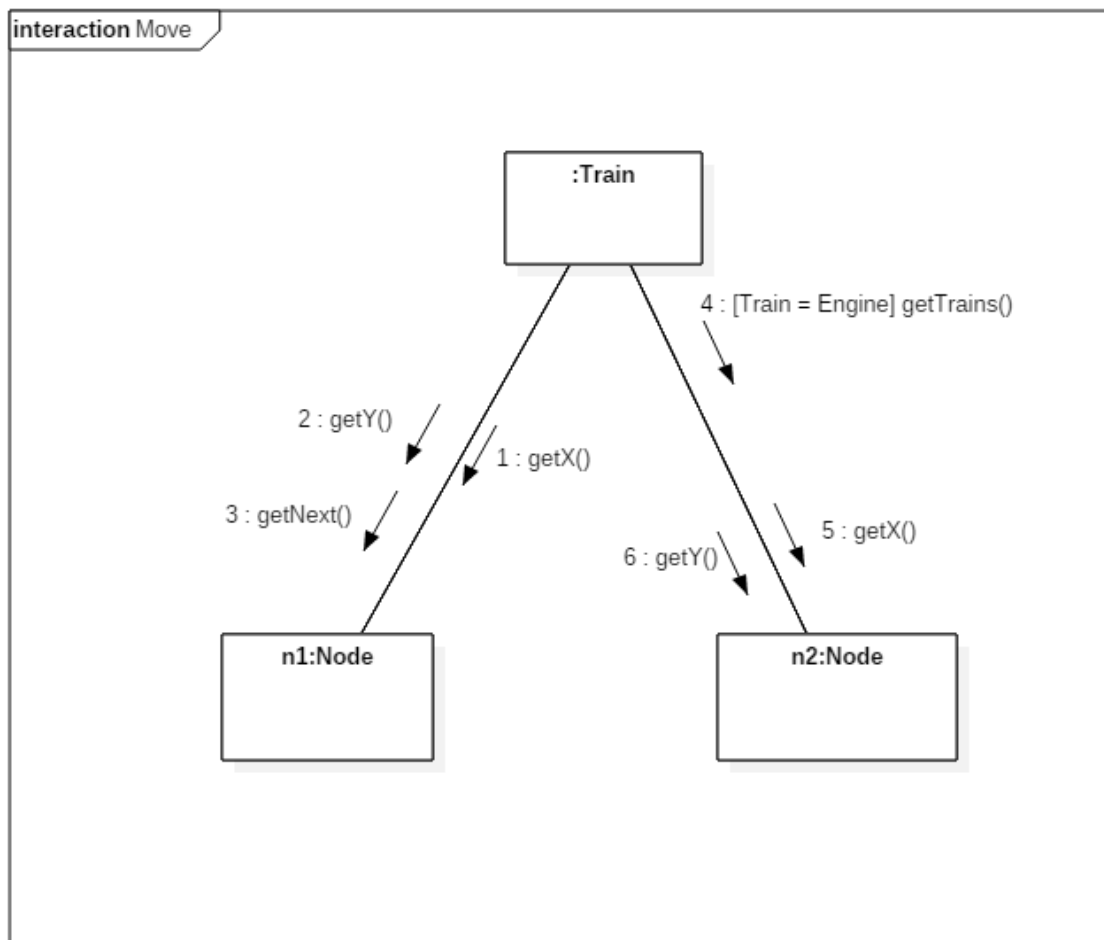
5.4.7 Last Pass



40. ábra Last Pass kommunikációs diagram

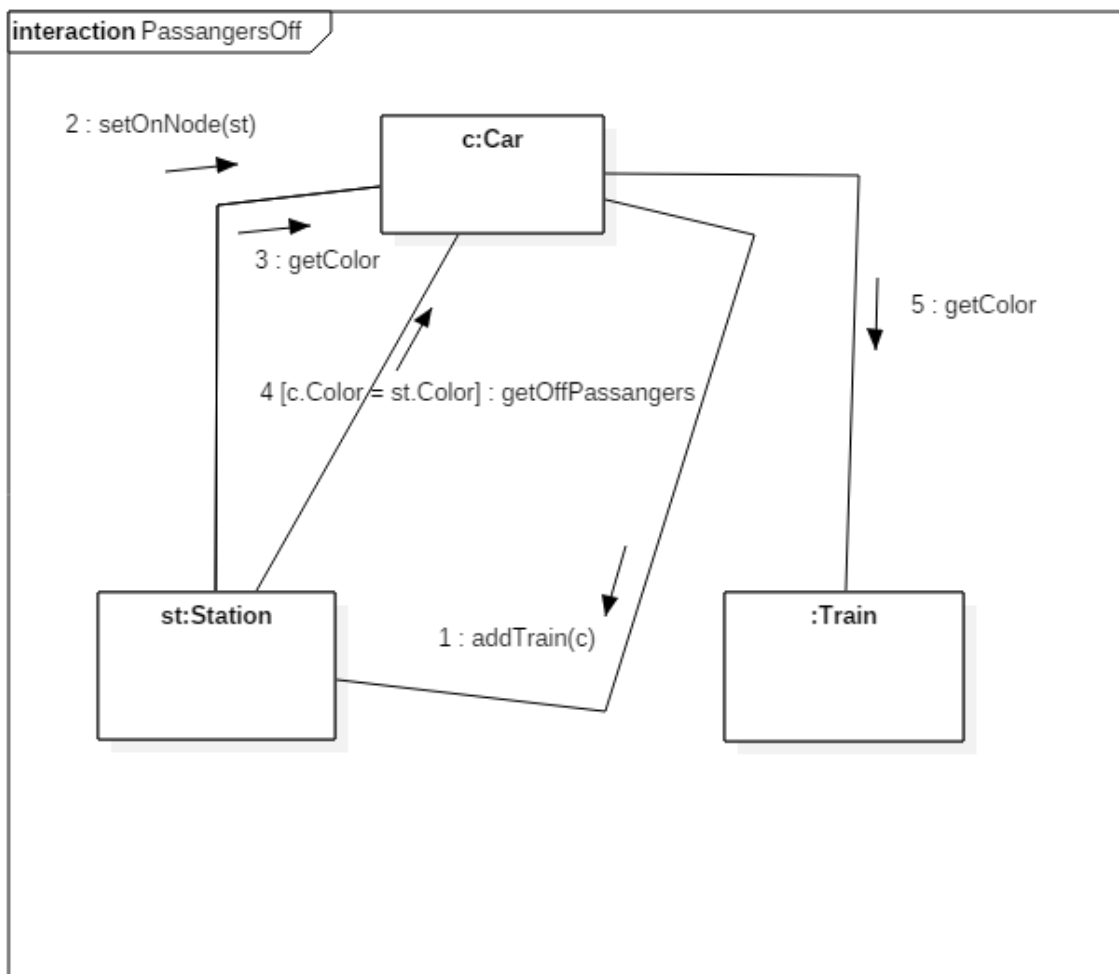
A fenti diagramon a Time actor moveEngines() függvénye fogja hívni a Model-t azonban a fejlesztőkörnyezet kommunikációs diagramba nem biztosított Actor-t.

5.4.8 Move



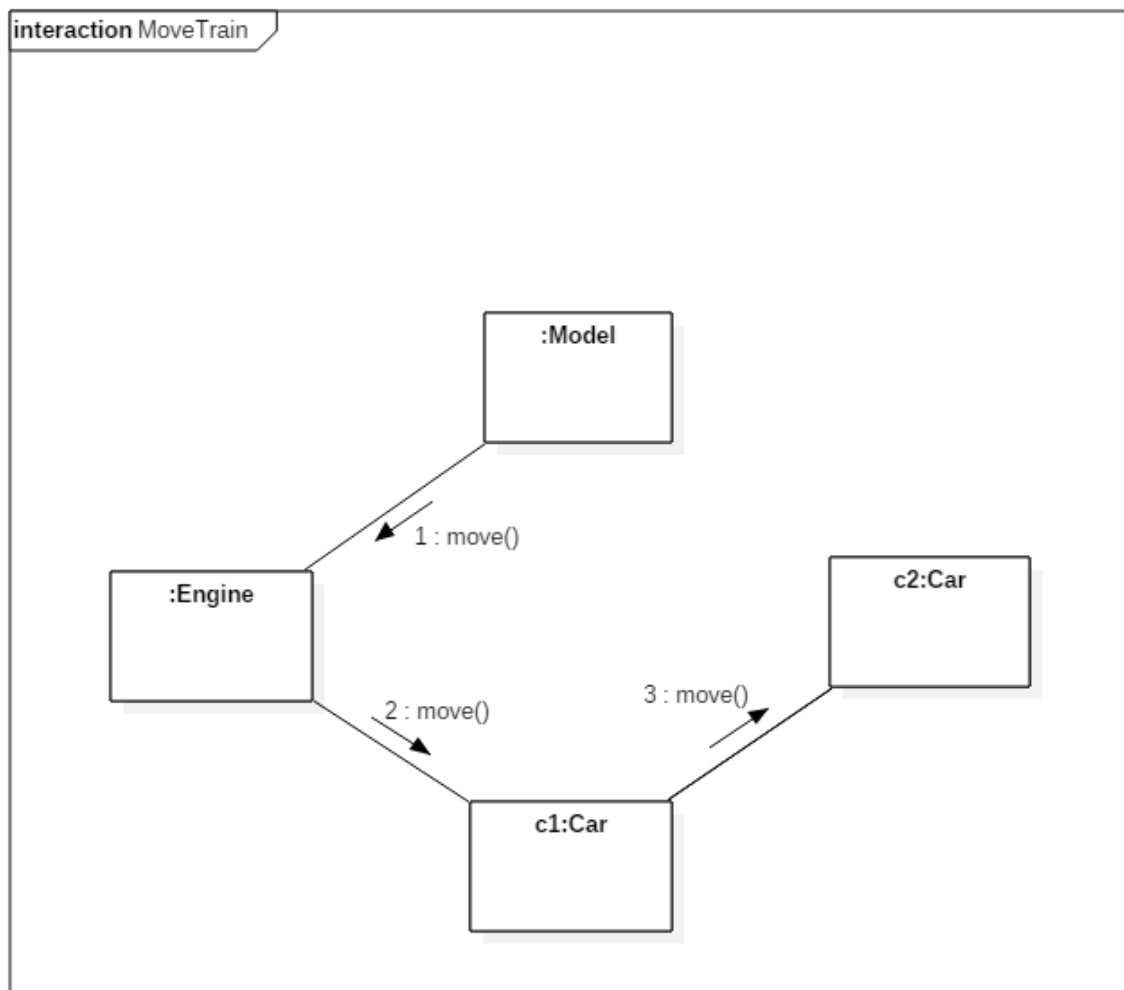
41. ábra Move kommunikációs diagram

5.4.9 Passangers Off



42. ábra Passangers Off kommunikációs diagram

5.4.10 Move Train



43. ábra Move kommunikációs diagram

A fenti diagramon a Time actor moveEngines() függvénye fogja hívni a Model-t azonban a fejlesztőkörnyezet kommunikációs diagramba nem biztosított Actor-t.

5.5 *Napló*

Kezdet	Időtartam	Résztevők	Leírás
2017.03.10. 8:00	2,5 óra	Tóth Kunkli Zoltay Zahorán	Megbeszélés: Use Casek megvitatása, szkeleton kezelői felület elképzelése.
2017.03.11. 11:00 2017.03.11. 22:00	5 óra	Kunkli	Szekvencia diagrammok elkészítése
2017.03.11. 22:00	3,5 óra	Tóth	Kommunikációs diagramok
2017.03.12. 16:00	3 óra	Zoltay	Use-case diagram, use-case leírások
2017.03.12. 18:00	3 óra	Zahorán	Szkeleton kezelői felületének terve, dialógusok. Végleges formázás.

6. Szkeleton beadása

6 Szkeleton beadás

6.1 Fordítási és futtatási útmutató

6.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Car.java	6 Kb	2017.03.19. 18:13	Car osztály.
Engine.java	4 Kb	2017.03.19. 18:13	Engine osztály.
Main.java	6 Kb	2017.03.19. 18:13	Main osztály. A program belépési pontja.
MapItem.java	2 Kb	2017.03.19. 18:13	MapItem absztrakt osztály.
Model.java	3 Kb	2017.03.19. 18:13	Model osztály.
Node.java	3 Kb	2017.03.19. 18:13	Node absztrakt osztály.
Rail.java	1 Kb	2017.03.19. 18:13	Rail osztály.
Station.java	3 Kb	2017.03.19. 18:13	Station osztály.
Status.java	1 Kb	2017.03.19. 18:13	Status enumok.
Switch.java	2 Kb	2017.03.19. 18:13	Switch osztály.
Train.java	4 Kb	2017.03.19. 18:13	Train absztrakt osztály,
TunnelEntrance.java	2 Kb	2017.03.19. 18:13	TunnelEntrance osztály.

6.1.2 Fordítás

A fenti fájlok .java kiterjesztésűek. Java nyelven íródtak, java kódot tartalmaznak, amelyek még nem futtatható állományok. A JDK (Java Development Kit) segítségével operációs rendszerünknek megfelelő futtatható állomány elkészítése a cél.

A javac.exe állomány felel a fordításért, a letöltött JDK mappájában található meg ..\bin\javac.exe helyen. Parancssori futtatásához legegyszerűbb, ha a bin mappában Shift+Jobb gomb lenyomása után a Parancsablak nyitása itt lehetőséget választjuk.

Majd a javac programot elindítjuk a megfelelő paraméterekkel, amelyek a projektben szereplő java fájlok. Több fájl fordítása esetén a tartalmazó mappa elérési útját érdemes megadni *.java kiegészítéssel, és elérni a -encoding UTF8 kapcsolót. Így az összes .java kiterjesztésű fájl lefordul.

Pl: javac -encoding UTF8 C:\project*.java

Amennyiben mindent jól csináltunk megjelenik minden .java fájl mellett egy azonos nevű .class kiterjesztésű fájl.

6.1.3 Futtatás

A futtatáshoz az előzőekhez hasonlóan kell eljárunk, javac.exe helyett a java.exe-t kell megfelelően paramétereznünk az indításhoz.

Pl: java C:\project.Main

```

C:\Windows\system32\cmd.exe - java project.Main

C:\Users\Zahorán László\Desktop> javac -encoding UTF8 project\*.java

C:\Users\Zahorán László\Desktop> java project.Main

Menü:
1. Váltó állítása
2. Alagút létrehozása
3. Alagút törlése
4. Vonat mozgatása
5. Vonat hozzáadása
6. Kilépés

? Adja meg a parancs kódját: 4
4. Vonat mozgatása
->[:Model].moveEngines()
->[:Engine].setNextCar()
->[:Car].setPrevTrain()
<-[:Car].setPrevTrain()
<-[:Engine].setNextCar()
->[:Train].setOnNode()
<-[:Train].setOnNode()
->[:Train].getNextCar()
<-[:Train].getNextCar()
->[:Train].getOnNode()
<-[:Train].getOnNode()
->[:Train].setOnNode()
<-[:Train].setOnNode()
->[:Engine].move()

```

44. ábra Fordítás és futtatás példa

6.2 Értékelés

Tag neve	Munka százalékban	Aláírás
Tóth Attila	25 %	
Kunkli Richárd	25 %	
Zahorán László	25 %	
Zoltay Marcell	25 %	

6.3 *Napló*

Kezdet	Időtartam	Résztevők	Leírás
2017.03.17. 12:00	3 óra	Tóth Kunkli Zoltay Zahorán	Értekezlet. Döntések: Az 5. Szkeleton tervezése c. dokumentum 5.2-es menü pontjában szereplő Menüpontok megvalósítása felosztva: Tóth: Alagút létrehozása, törlése Kunkli: Váltó állítás, Vonat mozgás Zoltay: Vonat hozzáadása Zahorán: Menü, Leszállás
2017.03.18. 14:00	6 óra	Zoltay	Vonat hozzáadása menüponthoz tartozó metódusok, Fájllista, A szkeleton kezelői felület tervének módosítása.
2017.03.18. 15:00	5,5 óra	Tóth	Alagút létrehozása és alagút törlése menüpontokhoz tartozó metódusok.
2017.03.18. 17:00	4 óra	Kunkli	Váltó állítása és Vonat mozgatása menüpontokhoz tartozó metódusok.
2017.03.18.	4 óra	Zahorán	Menü elkészítése, Leszállás folyamatához tartozó metódusok.
2017.03.19.	2 óra	Zahorán	Hiányzó kommentek kiegészítése, 6.1.2/3 Fordítás/Futtatás

7. Prototípus koncepciója

7 Prototípus koncepciója

7.0 Módosítások

7.0.1 Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case
Func11	Az utasok fel tudnak szállni a vagonokra	Bemutató	Alapvető	Feladat módosítása	MoveTrain

7.0.2 Objektum katalógus

7.0.2.1 Car

A vagonok a vonat elején levő mozdonyt követik. Több vagon is kapcsolódhat egymáshoz. Minden szomszédos vagon színe különbözik egy szerelvényen és utasokat szállít. Az utasok csak a mozdonytól a szerelvény hátulja felé szállhatnak le a megadott helyeken. **Bármelyik állomással azonos színű üres kocsiba utasok szállhatnak fel, de legfeljebb egy kocsiba.**

7.0.2.2 CoalCar

A szénkocsik a mozdonyt követően a szerelvény bármely részén elhelyezkedhetnek. Utasok nem utaznak rajta és nem is szállhatnak fel rá.

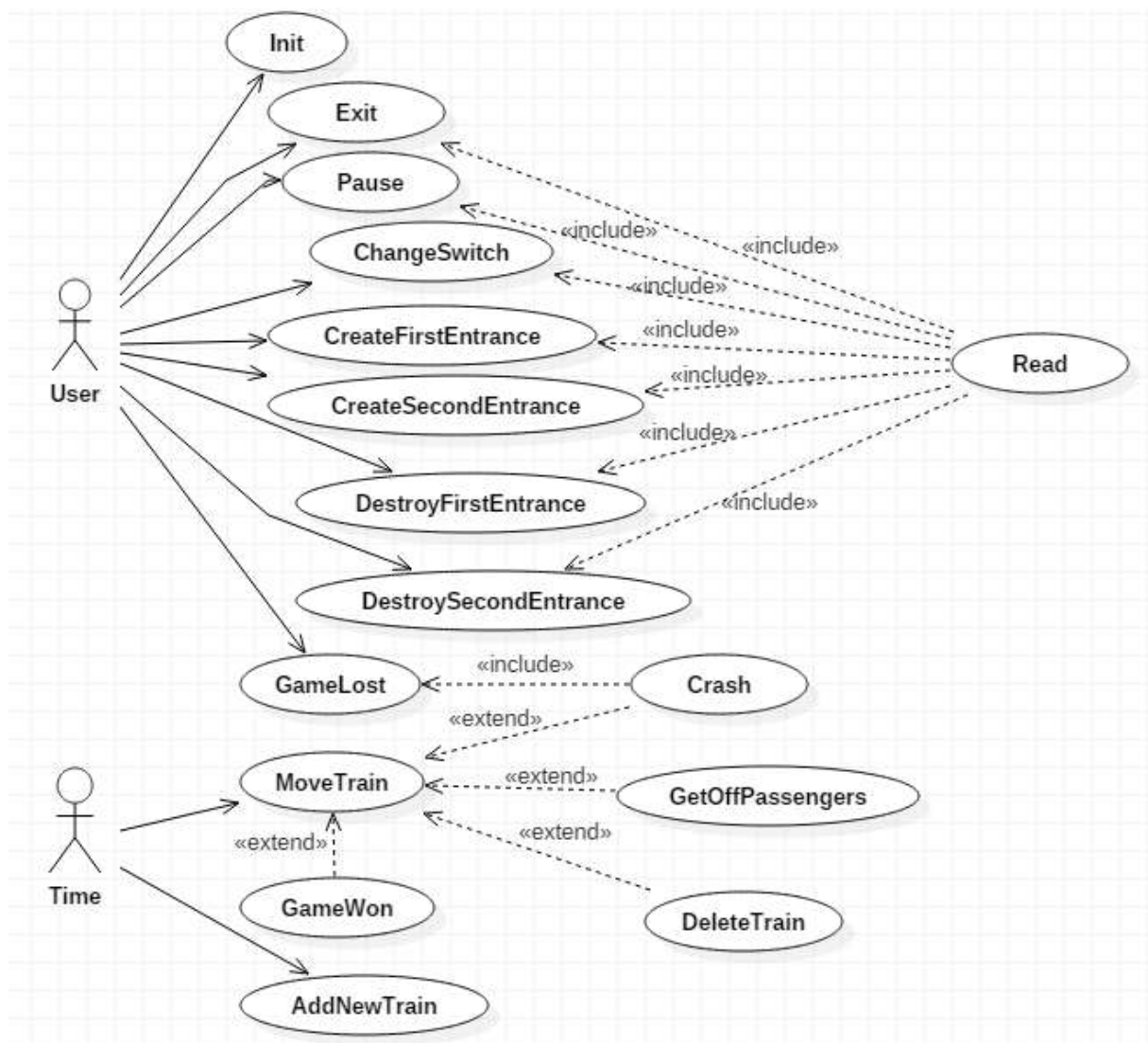
7.0.2.3 Station

Az állomás jelképezi az utasok célját a játékban, ahová szeretnének eljutni. Az állomások színekkel vannak megkülönböztetve a pályán és minden esetben sín vezet hozzájuk. A megfelelő színű vagonban utazó utasok csak a megfelelő színű állomáson szállhatnak le amikor a vonat elhalad mellette. **Az állomáson, azzal megegyező színű, legfeljebb egy üres kocsiba utasok szállhatnak fel.**

7.0.2.4 Cross

A kereszteződés olyan csomópont, melynek négy kimenete van és ezek párba vannak állítva. A négy irány bármelyikéből érkezve csak annak párja irányába mehet tovább a vonat.

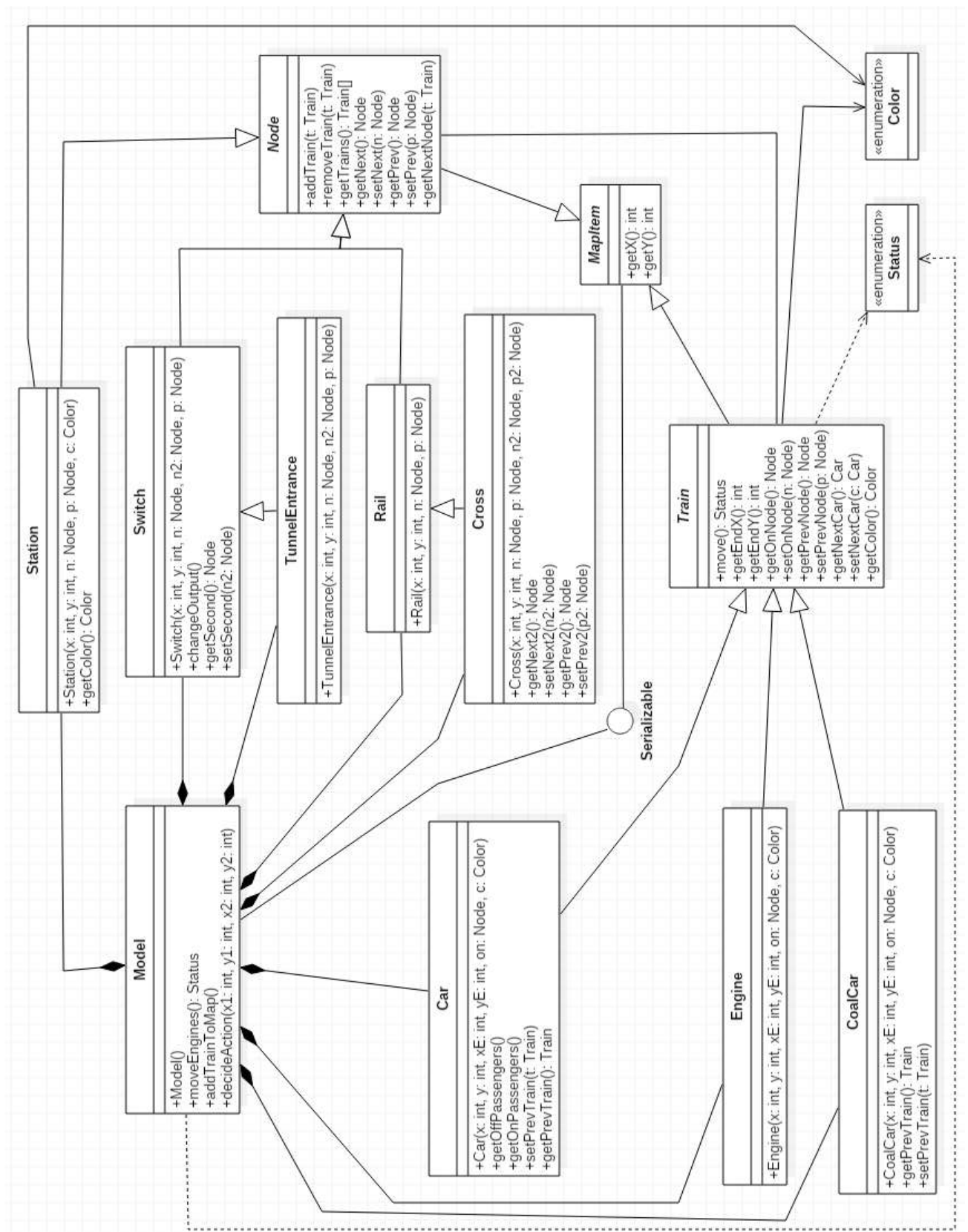
7.0.3 Use-case diagram



45. ábra Prototípus Use-case diagram

Use-case neve	GetOnPassengers
Rövid leírás	Felszállnak az utasok az adott kocsiba.
Aktorok	User
Forgatókönyv	A vonat mozgása során elért állomásokon utasok szállhatnak fel az állomással azonos színű üres kocsikba.

7.0.4 Osztálydiagram



46. ábra Prototípus osztálydiagram

7.0.5 Osztályok leírása

7.0.5.1 Car

- **Felelősség**

A kocsit megvalósító osztály. A vagonnak színe van. A rajta utazó összes utas a vagon színével azonos színű állomáson leszáll, ha az a mozdonyhoz legközelebbi nem üres vagon.

- **Ősosztályok:** MapItem, Train

- **Interfészek:** -

- **Attribútumok:** -

- **Metódusok**

- **Car(x: int, y: int, xE: int, yE: int, on: Node, c: Color):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
- **void getOffPassengers():** Az összes utast leszállítja a vagonból.
- **void getOnPassengers():** Utasokat szállít fel a vonatra.
- **void setPrevTrain(t: Train):** Beállítja az előtte álló kocsit, ha az első kocsi, akkor a mozdonyt.
- **Train getPrevTrain():** Beállítja az előtte álló kocsit, ha az első kocsi, akkor a mozdonyt.

7.0.5.2 CoalCar

- **Felelősség**

A szeneskocsit megvalósító osztály. Nem utazhatnak rajta utasok. A szeneskocsi a mozdonyt követően a szerelvény bármely részén elhelyezkedhet.

- **Ősosztályok:** MapItem, Train

- **Interfészek:** -

- **Attribútumok:** -

- **Metódusok**

- **CoalCar(x: int, y: int, xE: int, yE: int, on: Node, c: Color):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
- **Status move():** A vagon mozgatja. Visszatér egy állapot enum-mal.
- **void setPrevTrain(t: Train):** Beállítja az előtte álló kocsit, ha az első kocsi, akkor a mozdonyt.
- **Train getPrevTrain():** Beállítja az előtte álló kocsit, ha az első kocsi, akkor a mozdonyt.

7.0.5.3 Node

- **Felelősség**

Absztrakt osztály, a csomópontokat valósítja meg. Ezek segítségével lesz bejárható a pálya, tárolja a következő és előző elemet, így a váltók, sínek, állomások, keresztezések összekapcsolhatóak.

- **Ősosztályok:** MapItem

- **Interfészek:** -

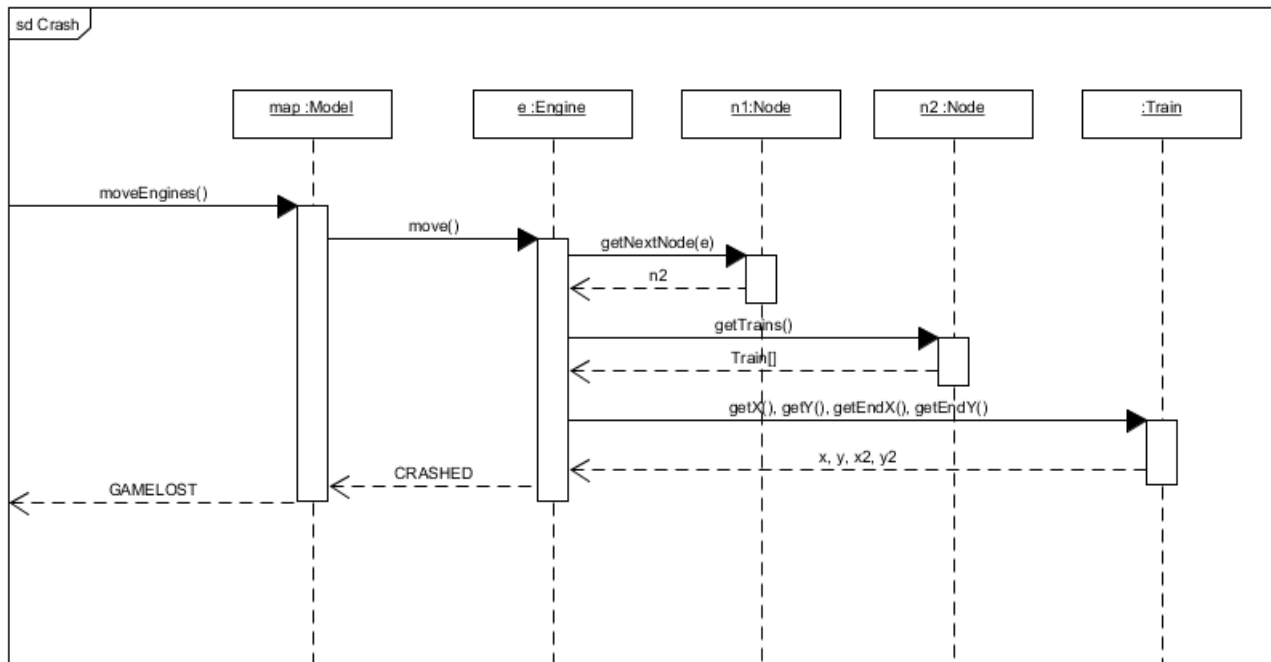
- **Attribútumok:** -
- **Metódusok**
 - **void addTrain(t: Train):** Hozzáadja a paraméterül kapott mozdonyt vagy kocsit a tömbhöz.
 - **void removeTrain(t: Train):** Kitörli a paraméterül kapott mozdonyt vagy kocsit a tömbből.
 - **Train[] getTrains():** Visszatér a tömbbel.
 - **Node getNext():** A következő csatlakoztatott elemmel tér vissza.
 - **void setNext(n: Node):** Beállítja a következő csomópontot.
 - **Node getPrev():** Az előző csatlakoztatott elemmel tér vissza.
 - **void setPrev(p: Node):** Beállítja az előző csomópontot.
 - **Node getNextNode(t: Train):** Visszaadja azt a következő csomópontot, amelynek irányába a paraméterül kapott vonatnak mennie kell.

7.0.6 Cross

- **Felelősség**
Az Railból származó egymást keresztező síneket megvalósító osztály. A vonatok mozgási irányát meghatározza.
- **Ősosztályok:** MapItem -> Node -> Rail
- **Interfészek:** -
- **Attribútumok:** -
- **Metódusok**
 - **Cross(x: int, y: int, n: Node, p: Node, n2: Node, p2: Node):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
 - **Node getNext2():** Visszaadja a kereszteződésből kimenő csomópontot.
 - **void setNext2(n2: Node):** Beállítja a paraméterként kapott Node-ot kimenő csomópontnak.
 - **Node getPrev2():** Visszaadja a kereszteződésbe bemenő csomópontot.
 - **void setPrev2(p2: Node):** Beállítja a paraméterként kapott Node-ot bemenő csomópontnak.

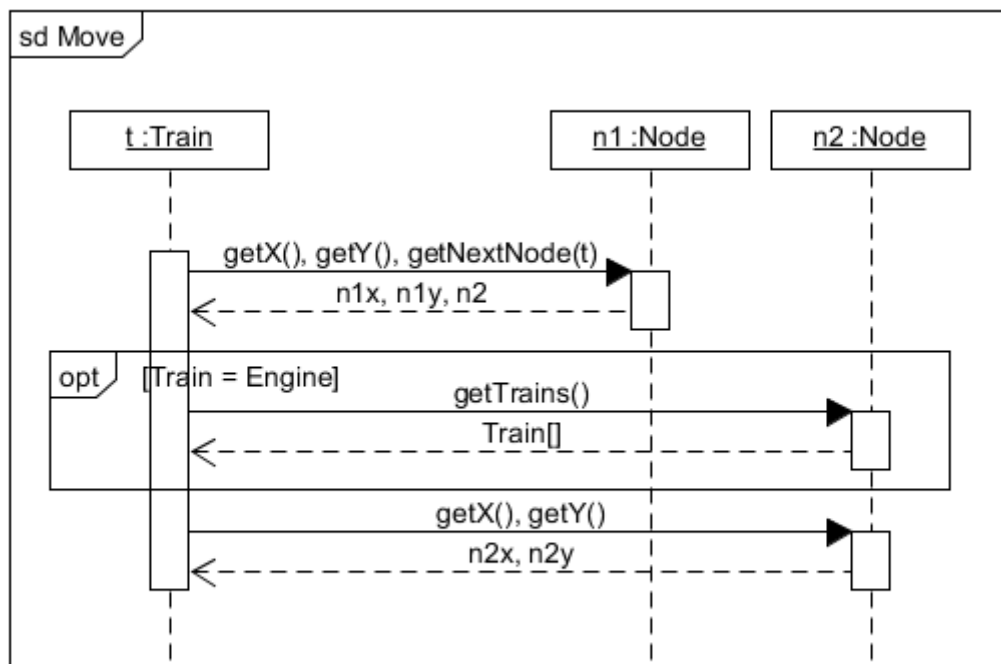
7.0.7 Szekvencia diagram

7.0.7.1 Crash



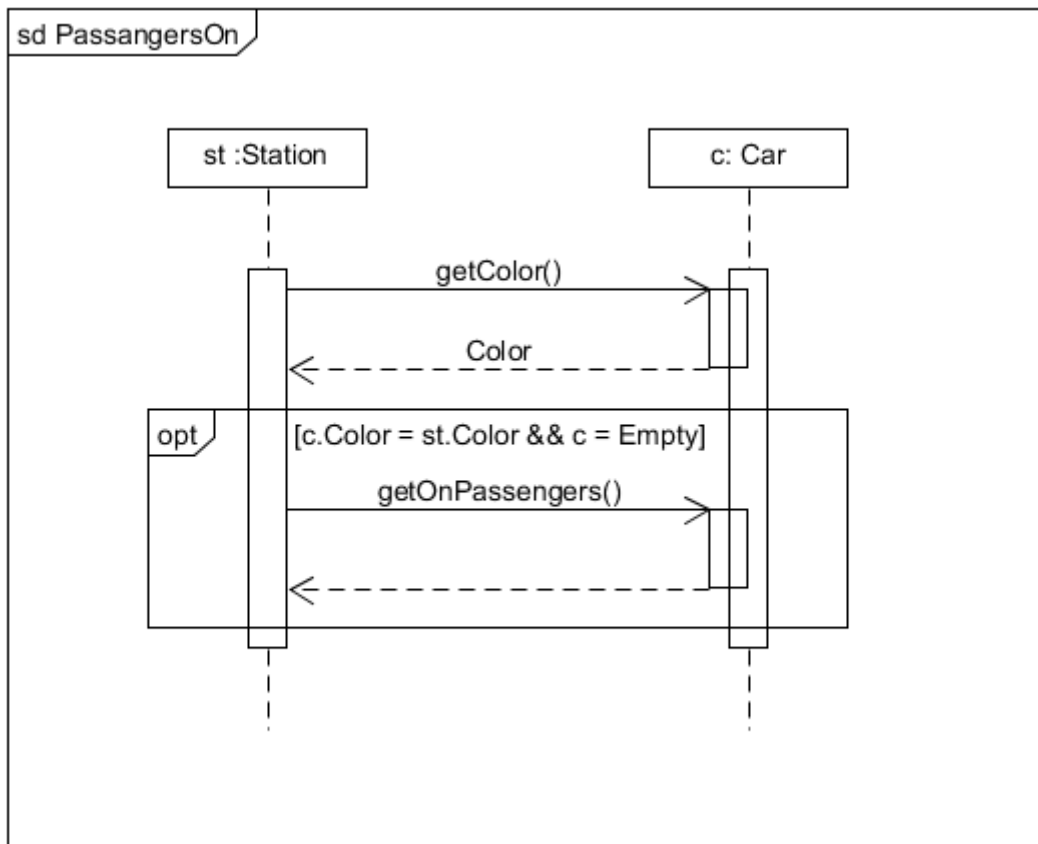
47. ábra Prototípus crash szekvencia diagram

7.0.6.2. Move



48. ábra Prototípus move szekvencia diagram

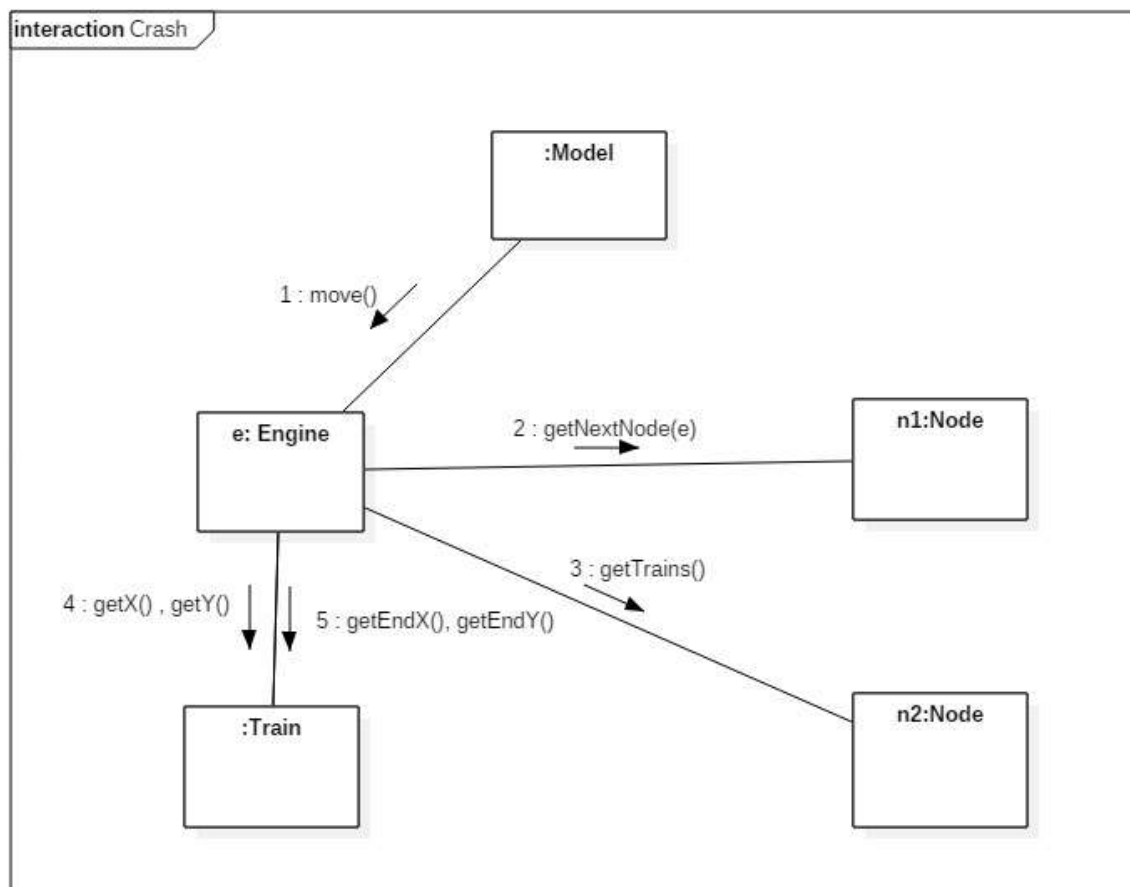
7.0.7.2 GetOnPassengers



49. ábra Prototípus GetOn Passengers szekvencia diagram

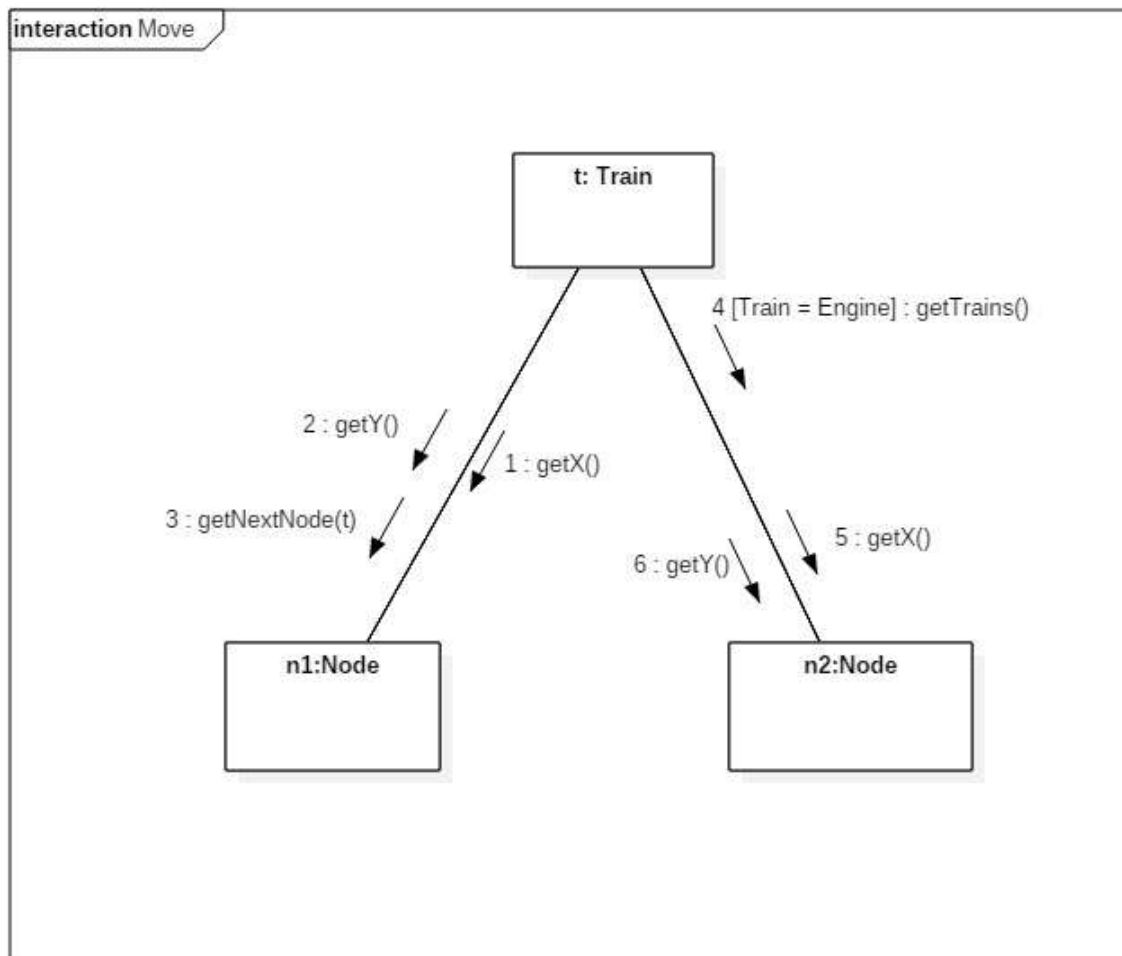
7.0.8 Kommunikációs diagram

7.0.8.1 Crash



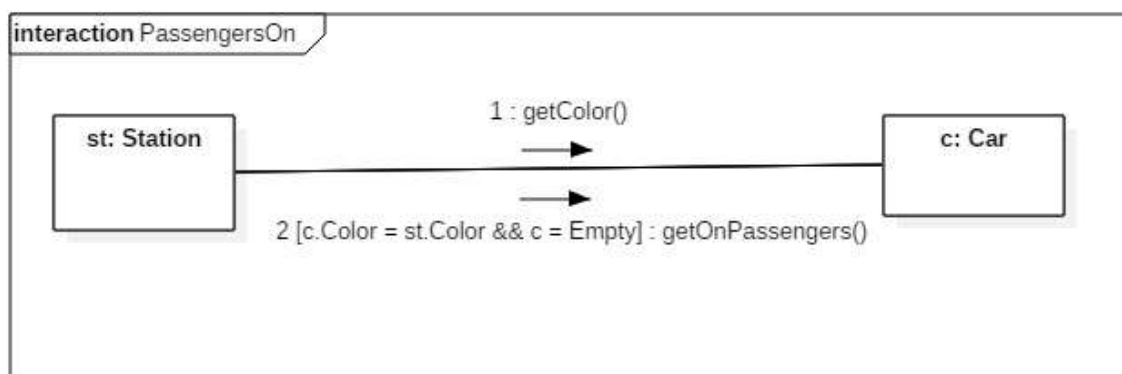
50. ábra Prototípus Crash kommunikációs diagram

7.0.8.2 Move



51. ábra Prototípus Move kommunikációs diagram

7.0.8.3 GetOnPassengers



52. ábra Prototípus GetOn Passengers kommunikációs diagram

7.1 Prototípus interface-definíciója

7.1.1 Az interfész általános leírása

A projekthez tartozó prototípus a grafikus felülethez képest egy egyszerűsített interface, amely segítségével minden funkció és megvalósított működés elérhető. Ez az interfész csak a szabványos bemenetről vezérelhető, és a szabványos kimenetre írja ki az egyes parancsokra adott kimeneteket. A tesztelő segédprogram ennek a típusú működésnek köszönhetően nagyban segíti az automatikus tesztelést. Ehhez a lehető legtöbb lehetőséget lefedő tesztek készülnek, amelyekhez minden esetben meg van adva a bemenet, ami egy meghatározott parancs sorozat, és az ehhez tartozó referencia kimenet. Ezek a be- és kimenetre irányíthatóak, így a valós és az elvárt kimenet összehasonlításával megállapítható a teszt sikeressége.

7.1.2 Bemeneti nyelv

- **node**

Leírás: Több funkciós parancs, mely segítségével létrehozhatunk, törölhetünk csomópontokat, azokat állíthatjuk.

Opciók:

-t TYPE, --type TYPE

Megadja a node típusát pl.: Rail, TunnelEntrance.

-c NAME, --create NAME

Létrehoz egy új elemet, ha az már létezik, azt módosíthatja a parancs többi attribútumával.

-r NAME, --remove NAME

Töröl egy már létező elemet, ha az alagút.

-C COORDS, --coords COORDS

Beírunk 2 egész értéket, melyek a csomópont koordinátái lesznek.

-sN NODE [NODE], --setnext NODE [NODE]

Beállítja a csomópont következő hozzá kapcsolt elemét. A váltóknak és kereszteződésnek kettőt kell beállítani.

-sP NODE [NODE], --setprev NODE [NODE]

Beállítja a csomópont előző hozzá kapcsolt elemét. A kereszteződésnek kettőt kell.

-sC COLOR, --setcolor COLOR

Beállítja a csomópont színét, ha van neki.

-o, --change

Vált egyet a csomópont kimenetén, ami azt adja meg hogy elsődlegesen merre menjen a vonat tovább ettől a csomóponttól, ha annak 2 kimenetele van.

- **move**

Leírás: A pályán lévő szerelvények mozgását végzi.

Opciók:

-s NUM, --steps NUM

Megadhatjuk, hogy hányat lépjenek egyszerre a szerelvények.

- **train**

Leírás: A pályára helyez egy új szerelvényt.

Opciók:

-t TYPE, --type TYPE

Megadja a train típusát pl.: Coal, Engine.

-c NAME, --create NAME

Létrehoz egy új elemet, ha az már létezik azt módosíthatja a parancs többi attribútumával.

-C COORDS, --coords COORDS

Beírunk 4 egész értéket, melyek a vonat koordinátái lesznek. Rendre x1 y1 x2 y2.

-sN TRAIN, --setnext TRAIN

Beállítja a vonat következő hozzá kapcsolt elemét.

-sP TRAIN, --setprev TRAIN

Beállítja a vonat előző hozzá kapcsolt elemét.

-sO NODE, --seton NODE

Beállítja, hogy melyik csomóponton áll a vonat.

-sC COLOR, --setcolor COLOR

Beállítja a vonat színét, ha az vagon.

- **load**

Leírás: Egy előre elkészített pálya tölthető be vele.

Opciók:

-I FILE, --load FILE

Megadja a file nevét és kiterjesztését.

- **ls**

Leírás: Ha nincs megadva opció, kilistázza a történt változásokat. Ellenkező esetben a típusnak megadott elemek adatait listázza ki. Amennyiben nincs mit kilistáznia, nem listáz ki semmit.

Opciók:

-a, --all

Minden elem, minden attribútumát kilistázza, nem csak azokat amelyek változtak.

-t TYPE, --type TYPE

Megadja, hogy milyen típusokat listázzon pl.: Engine, Switch, Train. Több megadása esetén, többet listáz.

Példa működésre:

A következő parancs sorozattal létrehozunk egy sín-váltó-állomás/sín pályát majd változtattuk a váltó állását. A csomópontok összekapcsolását elég 1 oldalról jelezni, a program magától beállítja a hiányzó attribútumokat (lásd: nem hívtuk a setprev-et se a váltóhoz kapcsolt sínre, se az állomásra).

```
node -c r1 -t Rail -C 0 0
node -c r2 -t Rail -C 2 0
node -c st1 -t Station -C 2 1 -sC Red
node -c s1 -t Switch -C 1 0 -sN st1 r2 -sP r1
node -c s1 -o
```

7.1.3 Kimeneti nyelv

Csak az 'ls' parancs rendelkezik kimenettel, mely megadja, hogy mik változtak az egyes elemek attribútumaiban, ha nincs változás, nem ír ki semmit.

Példa működésre:

```
...
node -c s1 -o
move
ls

s1 :Switch
  nextNode: r4 -> s1
  next2Node: s1 -> r4
e1 :Engine
  onNode: r3 -> s1
  prevNode: r2 -> r3
c1 :Car
  onNode: r2 -> r3
  prevNode: r1 -> r2
```

7.2 Összes részletes use-case

Use-case neve	changeSwitch
Rövid leírás	A váltó állapotának változtatása.
Aktorok	Tester
Forgatókönyv	A megadott azonosítóval rendelkező váltó kimenetét változtatja meg, ha nem tartózkodik rajta vonat.

Use-case neve	createTunnelEntrance
Rövid leírás	Alagút bejárat építés.
Aktorok	Tester
Forgatókönyv	Amennyiben kevesebb mint két bejárat van és a megadott koordináták helyesek, alagút bejáratot helyez el a pályán. Ha második bejárat készül, létrejön a kapcsolat a két bejárat között.

Use-case neve	destroyTunnelEntrance
Rövid leírás	Alagút bejárat törlése.
Aktorok	Tester
Forgatókönyv	A megadott azonosítóval rendelkező alagutat lebontja. Amennyiben párja van a pályán, a kapcsolatot megszünteti.

Use-case neve	move
Rövid leírás	A vonat mozgatása.
Aktorok	Tester
Forgatókönyv	A move a pályán található összes mozdonyt, így az összes vagon lépteti a pályán, és ezek helyzet változásából adódó összes esemény kezelésre kerül. Úgy mint: ütközés detektálás, váltón és kereszteződésben útvonal választás, állomásokon le- és felszállás.

Use-case neve	addTrainToMap
Rövid leírás	Hozzáad egy szerelvényt a pályához.
Aktorok	Tester
Forgatókönyv	

Use-case neve	listStations
Rövid leírás	Kilistázza az állomásokat.
Aktorok	Tester
Forgatókönyv	Megjeleníti a kimeneten az összes állomást.

Use-case neve	listSwitches
Rövid leírás	Kilistázza a váltókat.
Aktorok	Tester
Forgatókönyv	Megjeleníti a kimeneten az összes váltót.

Use-case neve	listTunnelEntrances
Rövid leírás	Kilistázza az alagút bejárátokat.
Aktorok	Tester
Forgatókönyv	Megjeleníti a kimeneten az összes alagút bejáratot.

Use-case neve	listTrains
Rövid leírás	Kilistázza a pályán lévő vonatokat.
Aktorok	Tester
Forgatókönyv	Megjeleníti a kimeneten az összes szerelvényt.

Use-case neve	loadMap
Rövid leírás	Betölti a megadott teszthez szükséges pályát.
Aktorok	Tester
Forgatókönyv	A megadott azonosítóval rendelkező előre elkészített kiszerializált pályát betölti.

7.3 Tesztelési terv

Teszt-eset neve	Váltó állítása
Rövid leírás	A váltó kimenetének tesztelése.
Teszt célja	Teszteli, hogy váltó kimenetének állítása után a megfelelő irányba halad-e tovább a vonat.

Teszt-eset neve	Vonat mozgatása alagútban
Rövid leírás	A két alagút bejárat közötti kapcsolat létrejöttének tesztelése.
Teszt célja	Teszteli, hogy a két alagút bejárat között létrejött-e a kapcsolat, vagyis ha az egyik bejáraton bemegy a vonat, akkor a másikon jön-e ki.

Teszt-eset neve	Vonat mozgatása kereszteződésben
Rövid leírás	Vonat kereszteződésbe érkezésének tesztelése.
Teszt célja	Teszteli, hogy a vonat a kereszteződésbe az egyik irányból érkezve a helyes irányba megy-e tovább.

Teszt-eset neve	Leszállás
Rövid leírás	Az utasok leszállásának tesztelése.
Teszt célja	Teszteli, hogy az utasok az állomással azonos színű első nem üres kocsiból leszállnak-e és különböző színű első nem üres kocsiból nem szállnak le.

Teszt-eset neve	Felszállás
Rövid leírás	Az utasok felszállásának tesztelése.
Teszt célja	Teszteli, hogy az utasok felszálltak-e valamelyik állomással azonos színű üres kocsiba.

Teszt-eset neve	Ütközés
Rövid leírás	Két vonat ütközése.
Teszt célja	Teszteli, hogy két vonat összeütközött-e.

Teszt-eset neve	Alagút bejárat építés
Rövid leírás	Alagút bejárat építésének tesztelése.
Teszt célja	Teszteli, hogy az alagút bejárat érvényes helyre épül-e meg és nem lehet megépíteni egy érvénytelen helyre.

Teszt-eset neve	Alagút bejárat törlése
Rövid leírás	Alagút bejárat törlésének tesztelése.
Teszt célja	Teszteli, hogy két meglévő alagút bejárat közül az egyik törlése esetén a másik bejáraton áthalad-e a vonat.

7.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása

A teszteléshez elkészített segédprogram, a tesztelési fázis felgyorsítását segíti, a megtervezett tesztek tematikusan a pálya összes elemét és azok kapcsolatát minél több esetben megpróbálja vizsgálni. Ezekhez elkészült referencia kimenetek, amelyek válaszok megadott bemenetekre könnyen összehasonlíthatóak a valós kimenettel. A tesztelést támogató segédprogram ezt felhasználva végzi majd a kimenetek összehasonlítását, így könnyen levonható a teszt sikeressége, a két kimenet megegyezése esetén sikeres, különбözés esetén sikertelen volt a teszt.

7.5 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.03.23. 12:00	4 óra	Tóth Kunkli Zahorán Zoltay	A projekt módosítások átbeszélése. Szükséges változtatások átgondolása. Prototípus működése nagy vonalakban.
2017.03.23. 18:00	3,5 óra	Zahorán	7.1.2 Bemeneti nyelv, 7.1.3 Kimeneti nyelv 7.1.1 Az interfacek ált. leírása 7.0.1 Funkcionális követelmény
2017.03.24. 16:00	3,5 óra	Zoltay	7.0.3 Use-case módosítások 7.0.4 Osztály diagram 7.0.5 Osztályok leírása 7.2 Use Case-ek
2017.03.26. 14:00	3,5 óra	Kunkli	7.0.2 Objektum katalógus 7.0.6 Szekvencia diagramm 7.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása
2017.03.26. 16:00	3,5 óra	Tóth	7.0.7 Kommunikációs diagramm 7.3 Tesztelési terv

8. Részletes tervek

8 Részletes tervek

8.1 *Osztályok és metódusok tervei*

8.1.1 Manager

- **Felelősség**

Állapotokat tárol, melyek befolyásolják a játék központi működését. Reagál a felhasználó parancsaira, melyek alapján betölti a következő állapotokat.

- **Ősosztályok:** -

- **Interfészek:** -

- **Attribútumok**

- - **State[] states:** Állapotokat tároló tömb.

- **Metódusok**

- + **Manager():** Létrehozza a Manager egy példányát.
- + **void run():** Elindítja a játékot, betölti az első állapotot, a Menu-t és elindítja azt.

8.1.2 Menu

- **Felelősség**

Egy állapot, melyet a manager használ. A menü működtetéséért felel.

- **Ősosztályok:** -

- **Interfészek:** State

- **Attribútumok:** -

- **Metódusok**

- + **Menu():** Létrehozza a Menu egy példányát.
- + **int start():** Elindítja a menüt. A menüben új játékot lehet kezdeni vagy kilépni.

8.1.3 Game

- **Felelősség**

Egy állapot, melyet a manager használ. A játék fő elemének indításáért felelős. Kezeli a felhasználó utasításait.

- **Ősosztályok:** -

- **Interfészek:** State
- **Attribútumok**
 - - **Model map:** Tárolja a pályát.
 - - **int numberOfTrains:** Megadja, hogy hány vonatot fog betölteni a pálya.
 - - **double waitingTime:** Megadja, hogy milyen időközönként töltsön be új vonatot.
- **Metódusok**
 - + **Game():** Létrehozza a Game egy példányát.
 - + **void start():** Elindítja a játékot, meghívja a read() metódusát.
 - - **void read():** Beolvassa a játékos utasításait.

8.1.4 Pause

- **Felelősség**
Egy állapot, melyet a manager használ. A szünet működtetéséért felel.
- **Ősosztályok:** -
- **Interfészek:** State
- **Attribútumok:** -
- **Metódusok**
 - + **Pause():** Létrehozza a Pause egy példányát.
 - + **int start():** Elindítja a szünetet. A szünetben folytatni lehet a játékot vagy kilépni.

8.1.5 End

- **Felelősség**
Egy állapot, melyet a manager használ. A játék befejező képernyőjének futtatásáért felel.
- **Ősosztályok:** -
- **Interfészek:** Status
- **Attribútumok:** -
- **Metódusok**
 - + **End():** Létrehozza az End egy példányát.
 - + **int start():** Elindítja a bezérő képernyőt.

8.1.6 Model

- **Felelősség**
Összefogja a rendszerben levő egységeket. Meghívja a mozdonyok move() metódusát. Lehetőséget biztosít alagút bejáratok építéséhez, rombolásához, illetve a váltók állításához.

- **Ősosztályok:** -
- **Interfészek:** -
- **Attribútumok**
 - - **List<Engine> engines:** A pályán lévő mozdonyok listája.
 - - **List<Car> cars:** A pályán lévő kocsik listája.
 - - **List<CoalCar> coalCars:** A pályán lévő szeneskocsik listája.
 - - **List<Station> stations:** A pályán lévő állomások listája.
 - - **List<Rail> rails:** A pályán lévő sínek listája.
 - - **List<Cross> crosses:** A pályán lévő kereszteződések listája.
 - - **List<Switch> switches:** A pályán lévő váltók listája.
 - - **List<TunnelEntrance> tunnelEntrances:** A pályán lévő alagút bejáratok listája.
- **Metódusok**
 - + **Model():** A konstruktor létrehozza a model objektumot.
 - + **Status moveEngines():** Meghívja a mozdonyok move() metódusát. Visszatér egy állapot enum-mal.
 - + **void addTrainToMap():** Új vonatot indít a pályán.
 - + **void decideActions(x1: int, y1: int, x2: int, y2: int):** A kapott paraméterek alapján eldönti az elvégzendő műveletet. Ez lehet alagút építés, rombolás, váltó állítása.
 - - **void addTunnelEntrance(x1: int, y1: int, x2: int, y2: int):** Új alagút bejáratot hoz létre a paraméterül kapott koordinátákkal, majd hozzáadja az alagút bejáratot tároló listához.
 - - **void removeTunnelEntrance(tE: TunnelEntrance):** Eltávolítja a paraméterül kapott alagút bejáratot a listából.
 - - **void changeSwitch(s: Switch):** Átállítja a paraméterként kapott váltó kimenetét.
 - - **void removeTrain(train: Train):** Kitörli a paraméterül kapott vonatot a pályáról.
 - - **boolean isEmptyMap():** Eldönti, hogy a pályán van-e még vonat és visszatér ennek megfelelően egy logikai változóval.

8.1.7 MapItem

- **Felelősség**

Absztrakt osztály, a pályát alkotó elemek ősosztálya, rögzíti azok helyét a pályán.
- **Ősosztályok:** -
- **Interfészek:** -
- **Attribútumok**
 - # **int x:** A csomópont x koordinátája.
 - # **int y:** A csomópont y koordinátája.
- **Metódusok**
 - + **int getX():** Visszatér az elem x koordinátájának értékével.
 - + **int getY():** Visszatér az elem y koordinátájának értékével.

8.1.8 Train

- **Felelősség**

Absztrakt osztály, ami a MapItem osztályból származik. A vonatot alkotó elemek ősosztálya. Ebből származnak az Engine, Car és CoalCar osztályok.

- **Ősosztályok:** MapItem

- **Interfészek:** -

- **Attribútumok**

- **# int xEnd:** A Train osztályból származó osztályok objektumai végének az x koordinátája.
- **# int yEnd:** A Train osztályból származó osztályok objektumai végének az y koordinátája.
- **# Node onNode:** Azt a csomópontot tárolja amelyiken aktuálisan van a Train osztályból származó osztályok objektumainak valamelyike.
- **# Node prevNode:** Azt a csomópontot tárolja ahonnan jött a Train osztályból származó osztályok objektumainak valamelyike, vagyis az onNode előtti csomópontot.
- **# Train nextCar:** A vonat elem következő kocsija.
- **#Color color:** A vonatelem színe.

- **Metódusok**

- **+ Status move():** Mozgatja a vonatot és visszatér egy Status enum-mal.
- **+ int getEndX():** Visszatér az objektum végének x koordinátájával.
- **+ int getEndY():** Visszatér az objektum végének y koordinátájával.
- **+ Node getOnNode():** Visszaadja azt a Node-t, amelyiken a vonat aktuális eleme éppen tartózkodik.
- **+ void setOnNode(n: Node):** Beállítja a vonat elemének azt a Node-t, amelyiken éppen tartózkodik.
- **+ Node getPrevNode():** Visszaadja azt a Node-t, amelyikről jött a vonat eleme.
- **+ void setPrevNode(p: Node):** Beállítja a vonat elemének azt a Node-t, amelyikről jött a vonat eleme.
- **+ Train getNextCar():** Visszaadja a vonat elemének a következő kocsiját.
- **+ void setNextCar(t: Train):** Beállítja a paraméterül kapott Train-t a vonat elemének a következő kocsijának.
- **+ Color getColor():** Visszaadja a vonat egy elemének a színét.
- **+ void setColor():** Beállítja a vonatelem színét.

8.1.9 Engine

- **Felelősség**

Az Engine osztály valósítja meg a mozdonyt. A vagonokat vontatásáért felel.

- **Ősosztályok:** MapItem -> Train

- **Interfészek:** -

- **Attribútumok:** -
- **Metódusok**
 - **+ Engine(x: int, y: int, xE: int, yE: int, on: Node, c: Color):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
 - **+ Status move():** Mozdítja a mozdonyt. Meghívja a vagonok move() metódusát. Visszatér egy állapot enum-mal.

8.1.10 CoalCar

- **Felelősség**
A szeneskocsit megvalósító osztály. Nem utazhatnak rajta utasok. A szeneskocsi a mozdonyt követően a szerelvény bármely részén elhelyezkedhet.
- **Ősosztályok:** MapItem -> Train
- **Interfészek:** -
- **Attribútumok**
 - **- Train prevTrain:** A tárolt előző mozdony, kocsi vagy szeneskocsi.
- **Metódusok**
 - **+ CoalCar(x: int, y: int, xE: int, yE: int, on: Node, c: Color):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
 - **+ Status move():** A vagon mozgatja. Visszatér egy állapot enum-mal.
 - **+ Train getPrevTrain():** Beállítja az előtte álló mozdonyt, kocsit vagy szeneskocsit.
 - **+ void setPrevTrain(t: Train):** Visszatér az előtte álló vonatrésszel.

8.1.11 Car

- **Felelősség**
A vagon megvalósító osztály. A vagonnak színe van. A rajta utazó összes utas a vagon színével azonos színű állomáson leszáll, ha az a mozdonyhoz legközelebbi nem üres vagon. Utasok szállhatnak fel a vagonba, ha üres és a színe azonos az állomásával.
- **Ősosztályok:** MapItem -> Train
- **Interfészek:** -
- **Attribútumok**
 - **Train prevTrain:** A tárolt előző mozdony, kocsi vagy szeneskocsi.
- **Metódusok**
 - **+ Car(x: int, y: int, xE: int, yE: int, on: Node, c: Color):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
 - **+ Status move():** A vagon mozgatja. Visszatér egy állapot enum-mal.
 - **+ void getOffPassengers():** Az összes utast leszállítja a vagonból.
 - **+ void getOnPassengers():** Utasokat szállít fel a vagonba.

- **+ Train getPrevTrain():** Beállítja az előtte álló mozdonyt, kocsit vagy szeneskocsit.
- **+ void setPrevTrain(t: Train):** Visszatér az előtte álló vonatrésszel.

8.1.12 tNode

- **Felelősség**

Absztrakt osztály, a csomópontokat valósítja meg. Ezek segítségével lesz bejárható a pálya, tárolja a következő és előző elemet, így a váltók, sínek, állomások, összekapcsolhatóak.

- **Ősosztályok:** MapItem

- **Interfészek:** -

- **Attribútumok**

- **# List<Train> trainsOn:** A csomóponton lévő mozdonyok, kocsik és szeneskocsik.
- **# Node nextNode:** Az előző kapcsolódó csomópont.
- **# Node prevNode:** A következő kapcsolódó csomópont.

- **Metódusok**

- **+ void addTrain(t: Train):** Hozzáadja a paraméterül kapott mozdonyt, kocsit vagy szeneskocsit a tömbhöz.
- **+ void removeTrain(t: Train):** Kitörli a paraméterül kapott mozdonyt, kocsit vagy szeneskocsit a tömbből.
- **+ List<Train> getTrains():** Visszatér a csomóponton lévő vonat elemek listájával.
- **+ Node getNext():** A következő csatlakoztatott csomóponttal tér vissza.
- **+ void setNext(n: Node):** Beállítja a következő csomópontot.
- **+ Node getPrev():** Az előző csatlakoztatott elemmel tér vissza.
- **+ void setPrev(p: Node):** Beállítja az előző csomópontot.
- **+ Node getNextNode(t: Train):** Visszaadja azt a következő csomópontot, amelynek irányába a paraméterül kapott vonatnak mennie kell.

8.1.13 Station

- **Felelősség**

A Station osztály a megállót valósítja meg. A megálló színével azonos színű vagonokból az utasok leszállnak itt. Azonos színű üres vagonba utasok szállhatnak fel.

- **Ősosztályok:** MapItem -> Node

- **Interfészek:** -

- **Attribútumok**

- **- Color color:** az állomás színe.

- **Metódusok**

- **+ Station(x: int, y: int, n: Node, p: Node, c: Color):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
- **+ Color getColor():** Visszatér az állomás színével.

8.1.14 Rail

- **Felelősség**
A Rail osztály a síneket valósítja meg. Meghatározza a vonatok mozgásának irányát.
- **Ősosztályok:** MapItem -> Node
- **Interfészek:** -
- **Attribútumok:** -
- **Metódusok**
 - **+ Rail(x: int, y: int, n: Node, p: Node):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.

8.1.15 Cross

- **Felelősség**
Az Railből származó egymást keresztező síneket megvalósító osztály. A vonatok mozgási irányát meghatározza.
- **Ősosztályok:** MapItem -> Node -> Rail
- **Interfészek:** -
- **Attribútumok**
 - **- Node next2:** A kereszteződésből kimenő másik csomópont.
 - **- Node prev2:** A kereszteződésbe bemenő másik csomópont.
- **Metódusok**
 - **+ Cross(x: int, y: int, n: Node, p: Node, n2: Node, p2: Node):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
 - **+ Node getNext2():** Visszaadja a kereszteződésből kimenő csomópontot.
 - **+ void setNext2(n2: Node):** Beállítja a paraméterként kapott Node-ot kimenő csomópontnak.
 - **+ Node getPrev2():** Visszaadja a kereszteződésbe bemenő csomópontot.
 - **+ void setPrev2(p2: Node):** Beállítja a paraméterként kapott Node-ot bemenő csomópontnak.

8.1.16 Switch

- **Felelősség**

A Swich osztály a váltót valósítja meg a pályán. A két választható kimenete közül a beállítottá irányítja a vonatot. Minden esetben van kiválasztott kimenete, nincs holtpontja.

- **Ősosztályok:** MapItem -> Node
- **Interfészek:** -
- **Attribútumok**
 - **# Node next2Node:** A váltó másik kimenete.
- **Metódusok**
 - **+ Switch(x: int, y: int, n1: Node, n2: Node, p: Node):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.
 - **+ void changeOutput():** Megváltoztatja a kiválasztott kimenetet.
 - **+ Node getSecond():** Megadja mi áll a váltó másik kimenetén.
 - **+ void setSecond(n2: Node):** Beállítja a másik kimenetet.

8.1.17 TunnelEntrance

- **Felelősség**

A TunnelEntrance osztály valósítja meg az alagút bejáratot a pályán. Két lerakott bejárat esetén alkot alagutat és irányítja át a vonatot az egyik bejáratból a másikba. Amennyiben nincs második bejárat lerakva a vonat áthalad rajta.
- **Ősosztályok:** MapItem -> Node -> Switch
- **Interfészek:** -
- **Attribútumok:** -
- **Metódusok**
 - **+ TunnelEntrance(x: int, y: int, n1: Node, n2: Node, p: Node):** A konstruktor a paraméterként kapott értékekkel inicializálja a létrehozott objektumot.

8.2 A tesztek részletes tervei, leírásuk a teszt nyelvén

Megjegyzés: a “...”-al jegyzett részek, csak a dokumentum áttekinthetőségét szolgálják. Így látszik, hogy mik is igazából a figyelni kívánt attribútumok értékei. A végleges kimenetben nem lesznek “...”-al leírt részek, a példány összes attribútuma a kimeneten lesz.

8.2.1 Váltó állítása

- **Leírás**

A váltó kimenetének tesztelése. Teszteli, hogy váltó kimenetének állítása után a megfelelő irányba mutat a váltó.
- **Ellenőrzött funkcionalitás, várható hibahelyek**

A váltó kimenet megváltozásának a hatását vizsgálja. A lehetséges hiba helyek, hogy a váltó kimenete nem változik meg.

- **Bemenet**
load -l changeSwitchMap.map
ls -t Switch
node -c s1 -o
ls -t Switch
- **Elvárt kimenet**
s1
 coordinates: 15, 10
 nextNode: r2
 next2Node: r3
 prevNode: r1
 trains:
s1
 coordinates: 15, 10
 nextNode: r3
 next2Node: r2
 prevNode: r1
 trains:

8.2.2 Vonat mozgása kereszteződésben

- **Leírás**
Vonat kereszteződésbe érkezésének tesztelése. Teszteli, hogy a vonat a kereszteződésbe az egyik irányból érkezve a helyes irányba megy-e tovább.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
A vonat kereszteződésben való mozgását teszteli. Várható hiba helyek, hogy a vonat nem a megfelelő irányba halad tovább a kereszteződésből.
- **Bemenet**
load -l moveTrainInCross.map
ls -t Train
move
ls -t Train
move
ls -t Train
- **Elvárt kimenet**
e1
coordinates: 0, 0, 0, 0
onNode: r1
prevNode: r1
nextCar: c1
c1
coordinates: 0, 0, 0, 0
color: RED
onNode: r1
prevNode: r1
nextTrain: c2
prevTrain: e1
c2

coordinates: 0, 0, 0, 0
color: RED
onNode: r1
prevNode: r1
nextTrain: null
prevTrain: c1

e1
coordinates: 0, 0, 0, 0
onNode: cr1
prevNode: r1
nextCar: c1

c1
coordinates: 0, 0, 0, 0
color: RED
onNode: cr1
prevNode: r1
nextTrain: c2
prevTrain: e1

c2
coordinates: 0, 0, 0, 0
color: RED
onNode: cr1
prevNode: r1
nextTrain: null
prevTrain: c1

e1
coordinates: 0, 0, 0, 0
onNode: r2
prevNode: cr1
nextCar: c1

c1
coordinates: 0, 0, 0, 0
color: RED
onNode: r2
prevNode: cr1
nextTrain: c2
prevTrain: e1

c2
coordinates: 0, 0, 0, 0
color: RED
onNode: r2
prevNode: cr1
nextTrain: null
prevTrain: c1

8.2.3 Leszállás

- **Leírás**

Az utasok leszállásának tesztelése. Teszteli, hogy az utasok az állomással azonos színű első nem üres kocsiból leszállnak-e és különböző színű első nem üres kocsiból nem szállnak le.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Az állomásra beérkező vonatokból az utasok leszállításának hatását teszteli. A várható hiba helyek, hogy az utasok a nem megfelelő színű állomáson is leszállnak vagy éppen ott ahol kellene nem szállnak le, esetleg nem az első nem üres kocsiból történik meg a leszállás.

- **Bemenet**

```
load -l getOffPassengers.map
```

```
ls -t Car
```

```
move
```

```
ls -t Car
```

- **Elvárt kimenet**

c1

coordinates: 0, 0, 0, 0

color: GREEN

onNode: r1

prevNode: r1

nextTrain: c2

prevTrain: e1

c2

coordinates: 0, 0, 0, 0

color: RED

onNode: r1

prevNode: r1

nextTrain: null

prevTrain: c1

c1

coordinates: 0, 0, 0, 0

color: GREEN_GRAY

onNode: st1

prevNode: r1

nextTrain: c2

prevTrain: e1

c2

coordinates: 0, 0, 0, 0

color: RED

onNode: st1

prevNode: r1

nextTrain: null

prevTrain: c1

8.2.4 Felszállás

- **Leírás**

Az utasok felszállásának tesztelése. Ugyan működik, mint a leszállás, csak pont fordítva, nem lehet befolyásolni, mert az idő határozza meg mikor történik. FONTOS, hogy ez a tesztelés során okozhat némi eltérést, mivel az eset bekövetkezésére 20% esély van. Így például lehetséges, hogy a utasok leszállását követően, azok rögtön felszállnak, és így a leszállás teszt eredménye hibásnak fog tűnni, holott jól működik. Ebben az esetben a teszt ismétlése javasolt, melyben nagy valószínűséggel nem fog újra előfordulni a látszólagos 'hiba'.

8.2.5 Ütközés

- **Leírás**

Két vonat ütközése. Teszteli, hogy két vonat összeütközött-e.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A move függvény ütközés érzékelését teszteli, várható hiba, hogy ha nem érzékeli, hogy ütköztek.

- **Bemenet**

```
load -l crash.map  
move
```

- **Elvárt kimenet**

```
> Trains crashed! You lost
```

8.2.6 Alagút bejárat építése

- **Leírás**

Alagút bejárat építésének tesztelése. Teszteli, hogy az alagút kimenete megváltozik-e, ha megépül a második alagút.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Az alagút megépülését teszteli. Várható hiba helyek, hogy az alagút következő csomópont mutatója, nem a másik alagútra mutat.

- **Bemenet**

```
load -l createTunnelEntrance.map  
node -c te1 -t TunnelEntrance -sN r2 -sP r1  
ls -t TunnelEntrance  
node -c te2 -t TunnelEntrance -sN r4 te1 -sP r3  
ls -t TunnelEntrance
```

- **Elvárt kimenet**

```
te1  
  coordinates: 0, 0  
  nextNode: r2  
  prevNode: r1  
  trains:
```

```
te1  
  coordinates: 0, 0  
  nextNode: te2  
  prevNode: r1  
  trains:
```

```
te2
  coordinates: 0, 0
  nextNode: te1
  prevNode: r3
  trains:
```

8.2.7 Alagút bejárat törlése

- **Leírás**
Alagút bejárat törlésének tesztelése. Teszteli, hogy két meglévő bejárat közül, az egyik törlése esetén megváltoznak-e a másik bejárat attribútumai.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
Az alagút törlését teszteli. Várható hiba helyek, hogy a másik alagút következő csomópontra mutatója, nem változik meg.
- **Bemenet**
load -l removeTunnelEntrance.map
ls -t TunnelEntrance
node -r te1
ls -t TunnelEntrance
node -r te2
ls -t TunnelEntrance
- **Elvárt kimenet**
te1
 coordinates: 0, 1
 nextNode: te2
 prevNode: r1
 trains:

te2
 coordinates: 2, 1
 nextNode: te1
 prevNode: r3
 trains:

te2
 coordinates: 2, 1
 nextNode: r4
 prevNode: r3
 trains:

8.3 A tesztelést támogató programok tervei

A csapat úgy gondolta, hogy az előző dokumentumban ismertetett bemeneti nyelv, nem volt tökéletes, ezért egy új nyelv került specifikálásra, melyet alább ismertetünk. Az egységek számozása, az ezt megelőző dokumentumnak felelnek meg, aszerint érvényesek. Így a tesztadatok előállítására, az 'ls' parancs szolgál, a tesztek eredményének kiértékelését a tesztelő végzi, összeveti az előbb ismertetett parancs kimenetét az előre elkészített elvárt kimeneti fájlal.

8.4 *Napló*

Kezdet	Időtartam	Résztevők	Leírás
2017.03.29. 18:00	4 óra	Tóth Kunkli Zahorán Zoltay	Megbeszélés: Tóth: Osztályleírások kiegészítése Kunkli: Bemeneti/Kimeneti nyelv javítása
2017.04.01. 13:00	3 óra	Tóth	8.2 A tesztek részletes tervei
2017.04.01. 15:00	3 óra	Zahorán	8.1 Osztályok terveinek első fele.
2017.04.02. 12:00	3 óra	Zoltay	8.1 Osztályok terveinek második fele.

10. Prototípus beadása

10 Prototípus beadása

10.1 Fordítási és futtatási útmutató

10.1.1 project könyvtár tartalma

Fájl neve	Méret	Keletkezés ideje	Tartalom
Car.java	4 Kb	2017.04.18 6:01	A kocsit megvalósító osztály.
CoalCar.java	3 Kb	2017.04.18 5:10	A szeneskocsit megvalósító osztály.
Color.java	2 Kb	2017.04.18 2:27	A színeket megvalósító enum.
Cross.java	3 Kb	2017.04.18 2:27	A kereszteződést megvalósító osztály.
End.java	1 Kb	2017.04.08 6:29	A játék végét megvalósító menü
Engine.java	2 Kb	2017.04.18 5:20	A mozdonyt megvalósító osztály.
Game.java	3 Kb	2017.04.17 11:42	A játékot megvalósító osztály
Main.java	1 Kb	2017.04.18 6:53	Main osztály. A program belépési pontja.
Manager.java	2 Kb	2017.04.08 6:29	A menüket/állapotokat kezelő osztály
MapItem.java	2 Kb	2017.04.18 6:24	A pálya elemeket megvalósító absztrakt osztály.
Menu.java	1 Kb	2017.04.08 6:29	A játék indításáért felelős osztály
Model.java	32 Kb	2017.04.18 9:50	A parancsok feldolgozásáért, értelmezéséért felel
Node.java	3 Kb	2017.04.18 5:55	A csomópontokat megvalósító absztrakt osztály.
Pause.java	1 Kb	2017.04.08 6:29	A szünetet megvalósító osztály
Rail.java	1 Kb	2017.04.18 2:27	A sínt megvalósító osztály.
State.java	1 Kb	2017.04.08 6:29	Az állapotok őse, interfésze
Station.java	3 Kb	2017.04.18 5:59	Az állomást megvalósító osztály.

Status.java	1 Kb	2017.04.18 2:27	Az állapotokat megvalósító enum.
Switch.java	3 Kb	2017.04.18 2:27	A váltót megvalósító osztály.
Train.java	5 Kb	2017.04.18 5:55	A vonat elemeket megvalósító absztrakt osztály.
TunnelEntrance.java	1 Kb	2017.04.18 6:24	Az alagút bejáratot megvalósító osztály.

10.1.2 maps könyvtár tartalma

Fájl neve	Méret	Keletkezés ideje	Tartalom
changeSwitch.map	1 Kb	2017.04.18 9:44	A váltó állítása teszteset pályája.
crash.map	1 Kb	2017.04.18 3:41	Az ütközés teszteset pályája.
createTunnelEntrance.map	1 Kb	2017.04.17 10:30	Az alagút bejárat készítés teszteset pályája.
getOffPassengers.map	1 Kb	2017.04.18 3:41	Az leszállás teszteset pályája.
moveTrainInCross.map	1 Kb	2017.04.18 3:41	A vonat mozgatása kereszteződésben teszteset pályája.
removeTunnelEntrance.map	1 Kb	2017.04.17 6:17	Az alagút bejárat törlése teszteset pályája.

10.1.3 ref_outputs könyvtár tartalma

Fájl neve	Méret	Keletkezés ideje	Tartalom
changeSwitch_ref.txt	1 Kb	2017.04.18 2:27	A váltó állítása teszteset referencia kimenete.
crash_ref.txt	1 Kb	2017.04.18 3:41	Az ütközés teszteset referencia kimenete.
createTunnelEntrance_ref.txt	1 Kb	2017.04.18 6:25	Az alagút bejárat készítése teszteset referencia kimenete.
getOffPassengers_ref.txt	1 Kb	2017.04.18 6:09	A leszállás teszteset referencia kimenete.
moveTrainInCross_ref.txt	2 Kb	2017.04.18 9:46	A vonat mozgatása kereszteződésben teszteset referencia kimenete.
removeTunnelEntrance_ref.txt	1 Kb	2017.04.18 6:25	Az alagút bejárat törlése teszteset referencia kimenete.

10.1.4 Fordítás

A fenti fájlok .java kiterjesztésűek. Java nyelven íródtak, java kódot tartalmaznak, amelyek még nem futtatható állományok. A JDK (Java Development Kit) segítségével operációs rendszerünknek megfelelő futtatható állomány elkészítése a cél.

A javac.exe állomány felel a fordításért, a letöltött JDK mappájában található meg ..\bin\javac.exe helyen. Parancssori futtatásához legegyszerűbb, ha a bin mappában Shift+Jobb gomb lenyomása után a Parancsablak nyitása itt lehetőséget választjuk.

Majd a javac programot elindítjuk a megfelelő paraméterekkel, amelyek a projektben szereplő java fájlok. Több fájl fordítása esetén a tartalmazó mappa elérési útját érdemes megadni *.java kiegészítéssel, és elérni a -encoding UTF8 kapcsolót. Így az összes .java kiterjesztésű fájl lefordul. Az egyszerűség érdekében másoljuk ki az src mappát a C főkönyvtárba.

Pl: javac -encoding UTF8 C:\src\project*.java

Amennyiben mindent jól csináltunk megjelenik minden .java fájl mellett egy azonos nevű .class kiterjesztésű fájl.

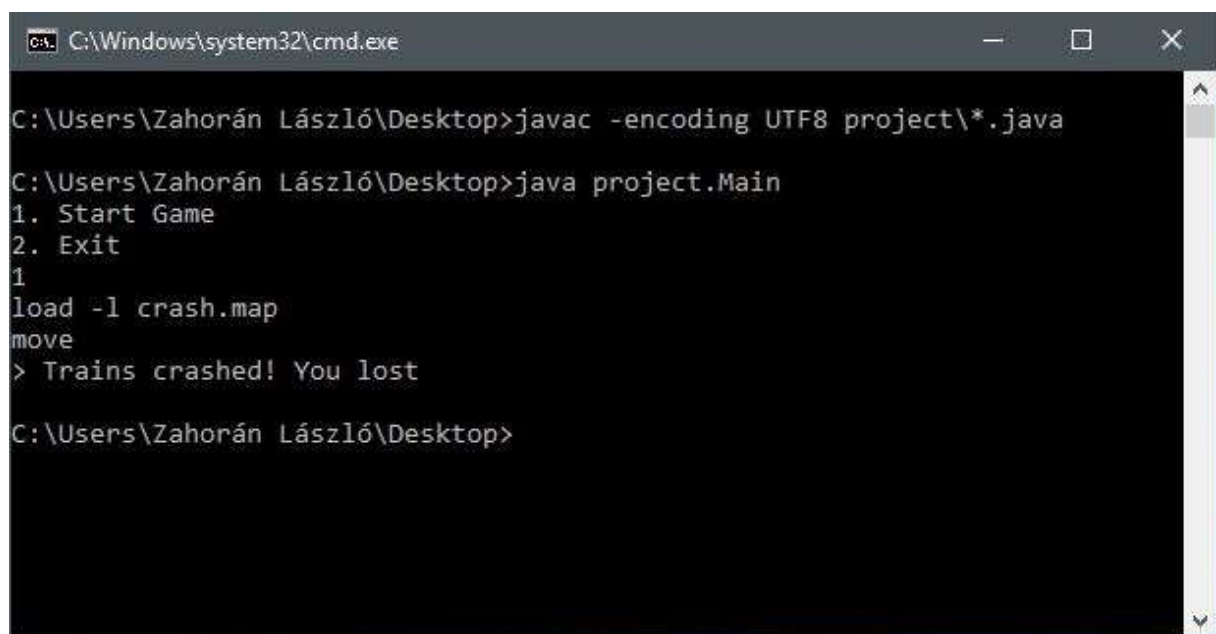
10.1.5 Futtatás

A futtatáshoz az előzőekhez hasonlóan kell eljárunk, javac.exe helyett a java.exe-t kell megfelelően paramétereznünk az indításhoz. A fordítás után lépünk be az 'src' mappába, majd indítsuk el a programot a következő parancsal:

java project.Main

Ha a letöltött JDK mappából futtatjuk a programot akkor a parancs kiegészül egy paraméterrel, mely megadja a lefordított fájl elérési útját.

Pl.: java -cp C:\src project.Main



```
C:\Windows\system32\cmd.exe

C:\Users\Zahorán László\Desktop>javac -encoding UTF8 project\*.java

C:\Users\Zahorán László\Desktop>java project.Main
1. Start Game
2. Exit
1
load -l crash.map
move
> Trains crashed! You lost

C:\Users\Zahorán László\Desktop>
```

53. ábra Prototípus fordítás és futtatás

10.2 Tesztek jegyzőkönyvei

10.2.1 Váltó állítása

Tesztelő neve	Zoltay Marcell
Teszt időpontja	2017.04.17.

10.2.2 Vonat mozgása kereszteződésben

Tesztelő neve	Tóth Attila
Teszt időpontja	2017.04.17.

Tesztelő neve	Tóth Attila
Teszt időpontja	2017.04.16
Teszt eredménye	Fail
Lehetséges hibaok	A szerelvények nem megfelelő úton haladtak tovább.
Változtatások	Cross osztály újragondolása.

10.2.3 Leszállás

Tesztelő neve	Tóth Attila
Teszt időpontja	2017.04.17.

10.2.4 Felszállás

Tesztelő neve	Zahorán László
Teszt időpontja	2017.04.17.

10.2.5 Ütközés

Tesztelő neve	Zahorán László
Teszt időpontja	2017.04.18. 2:00

Tesztelő neve	Zahorán László
Teszt időpontja	2017.04.17
Teszt eredménye	Fail
Lehetséges hibaok	A szerelvények nem észlelték egymást áthaladtak egymáson.
Változtatások	Engine és Car osztály move() metódusának átalakítása

10.2.6 Alagút bejárat építése

Tesztelő neve	Kunkli Richárd
Teszt időpontja	2017.04.17.
Teszt eredménye	Fail

Lehetséges hibaok	Nem változott meg az alagút 2 attribútuma miután mind a két alagút megépült
Változtatások	Az attribútumok megváltoztatása

10.2.7 Alagút bejárat törlése

Tesztelő neve	Kunkli Richárd
Teszt időpontja	2017.04.17.
Teszt eredménye	Fail
Lehetséges hibaok	Hasonló hiba mint az építésnél, mikor az egyik alagút törlődik a másikkal ki kell cserélni az attribútumait
Változtatások	Az attribútumok cserélése

10.3 Értékelés

Tag neve	Munka százalékban	Aláírás
Tóth Attila	25%	
Kunkli Richárd	25%	
Zahorán László	25%	
Zoltay Marcell	25%	

10.4 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.04.06. 12:30	2 óra	Tóth Kunkli Zahorán Zoltay	Feladatok kiosztása
2017.04.09. 2017.04.17.	5 óra 5 óra	Tóth	Osztályok: TunnelEntrance, Switch, Station, Node, MapItem Tesztek: Vonat mozgatása kerszteződésben, Leszállás
2017.04.04. 2017.04.17.	8 óra 2 óra	Kunkli	Osztályok: Manager, Menu, Pause, End, Game, State, Parancs értelmező Tesztek: Alagút bejárat építése, Alagút bejárat törlése
2017.04.13. 2017.04.17.	4 óra 6 óra	Zahorán	Osztályok: Train, Car, CoalCar, Color Tesztek: Felszállás, Ütközés
2017.04.14. 2017.04.17.	6 óra 4 óra	Zoltay	Osztályok: Rail, Cross, Engine, Model Tesztek: Váltó

11. Grafikus felület specifikációja

11 Grafikus felület specifikációja

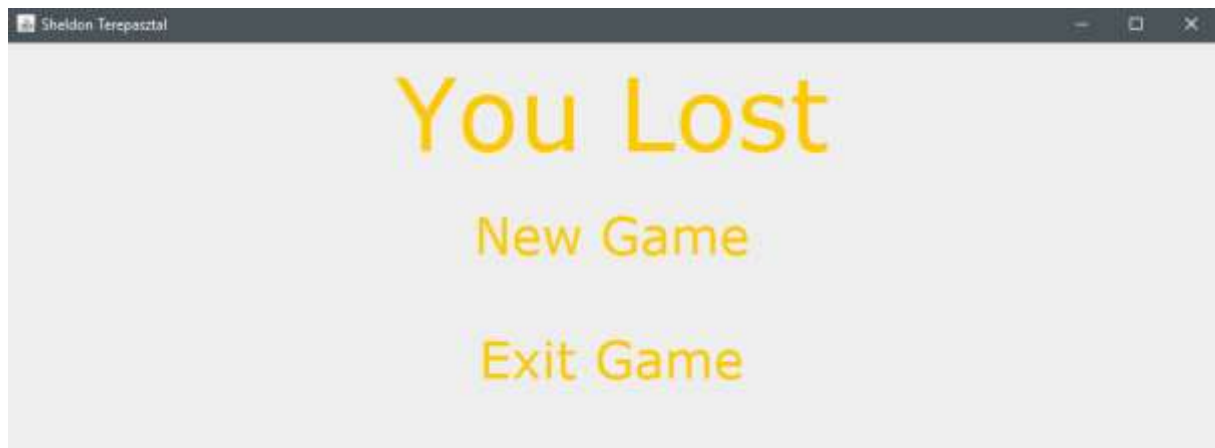
11.1 A grafikus interfész



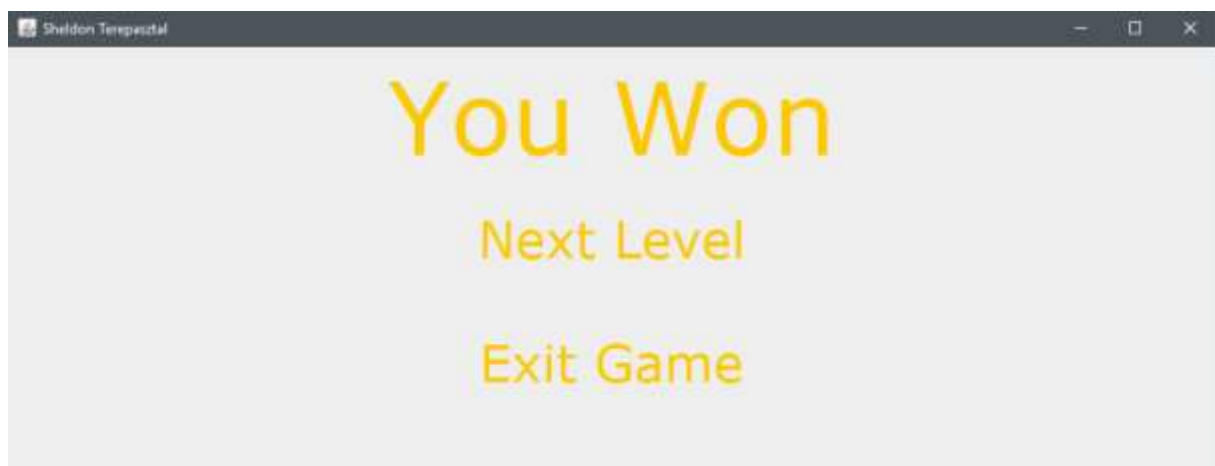
54. ábra Grafikus interfész főmenü elképzelés



55. ábra Grafikus interfész szünet menü elképzelés



56. ábra Grafikus interfész veszített játék menü elképzelés



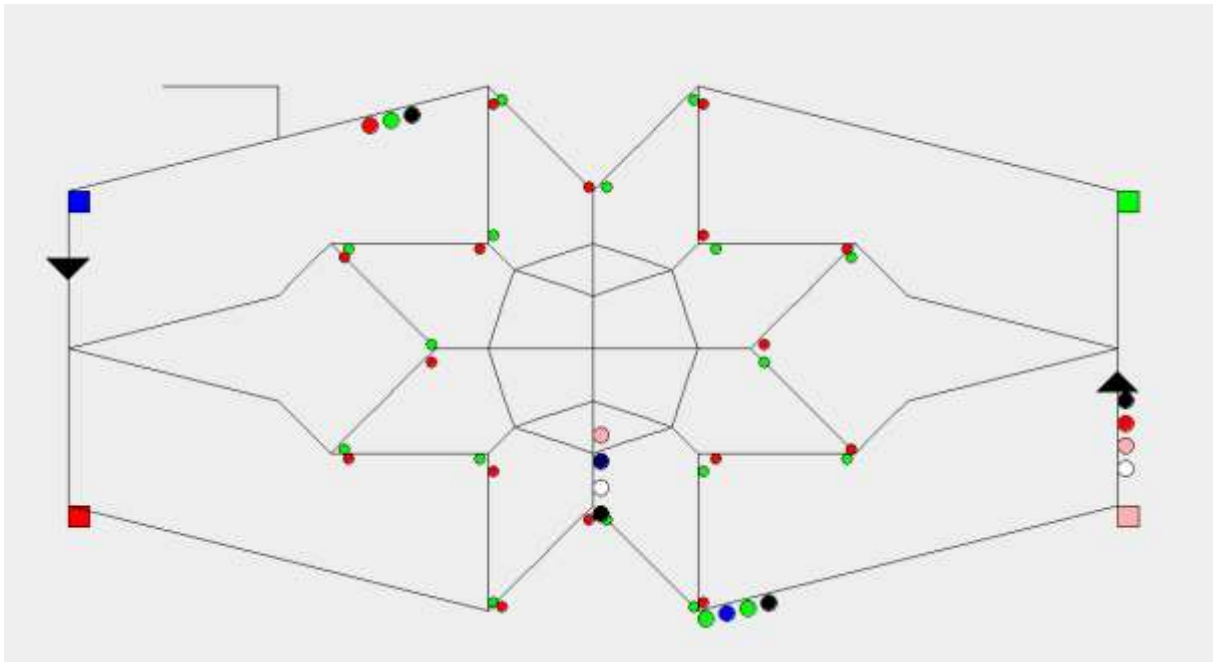
57. ábra Grafikus interfész nyert játék menü elképzelés

A szoftvert elindítva megjelenik egy grafikus ablak az 54. ábrán látható tartalommal. A két gombból választhat a felhasználó mit szeretne csinálni, az Exit feliratú gombbal ki tud lépni a játékból, míg a Start Game feliratú gombot megnyomva játékot tud indítani.

Új játékot kezdve, ha az Escape gombot megnyomjuk, akkor az 55. ábrán látható szünet menü jelenik meg.

A Continue gombra kattintva a játék folytatható, az Exit gombra kattintva kiléphetünk a játékból.

Az 56. és 57. ábra a játék kimenetelét mutatja. Nyert játék esetén a következő pályára léphetünk, veszített játék esetén új játékot kezdhetünk.



58. ábra Grafikus interfész játék elképzelés

A fenti játéktér jeleníti meg a játék pillanatnyi állapotát. A fekete vonalak jelölik a síneket, amiken a vonatok közlekednek. Az egyes kocsikat a különböző színű körök szimbolizálják, a kocsik legelején a kocsiktól különböző, fekete színű mozdony látható.

A sínek mellett színes négyzetekkel jelölt, adott színű állomások helyezkednek el, ahol az utasok fel- és leszállhatnak a kocsikról a játék szabályai szerint. A fekete háromszög jelöli az alagút bejáratokat. A váltóknál a zöld kör jelzi az aktív kimenetet, vagyis a vonatok abba az irányba haladhatnak tovább, a piros a passzív kimenetet.

11.2 A grafikus rendszer architektúrája

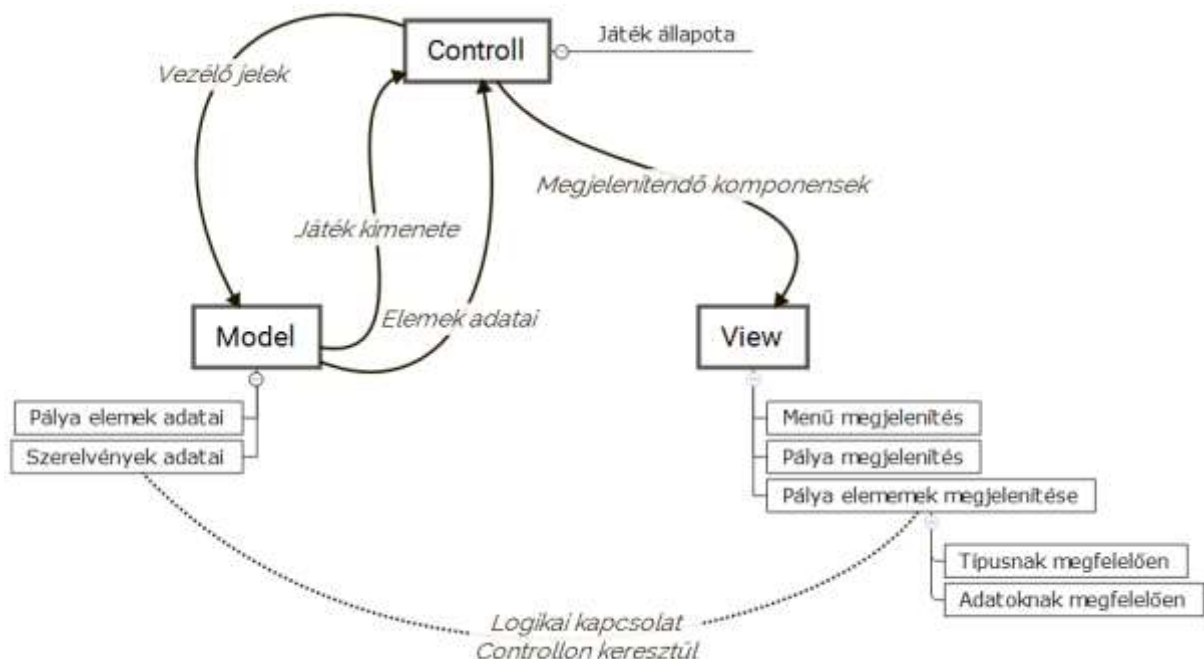
11.2.1 A felület működési elve

A felület működése alapvetően MVC (Model-View-Controller) architektúra alapján kerül megvalósításra kevert alapelv szerint, tehát push és pull is alkalmazásra kerül. Az eddigiekben elkészítésre került a Model és a Controller, a következőkben a View-val kapcsolatos tervek kerülnek dokumentálásra.

A Modellünk által a pályán szereplő elemek leképezhetők önálló objektumokra, ezeket kollektciókba gyűjtve tároljuk típusonként a Model osztályban. Így egységesen elérhető az összes objektum, az összes információval, amely segítségével mindez megjeleníthető a View által és a Controller segítségével a megjelenítésen változást indukálhat.

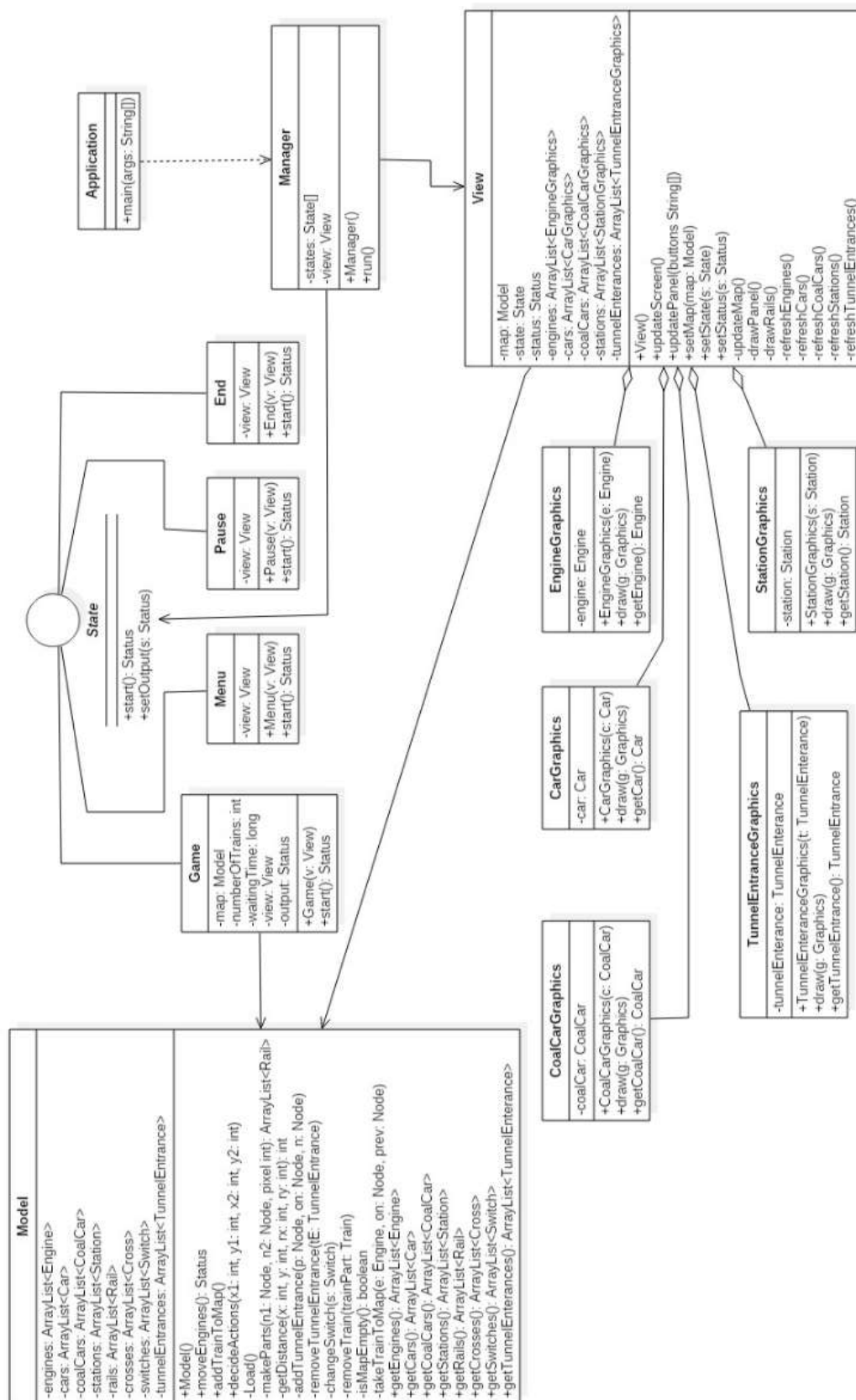
A Controller architektúráis elemnek a megfelelője a Manager osztály. Ennek legfőbb leírója egy State állapotváltozó, ami a különböző játék állapotokat változtatja, mint menü, játék, játék vége. Ez a változás létrejöhet a játékos által (Menüelem választás), vagy a Modellben a mozgás által (Ütközés, Játék megnyerése), így befolyásolva a megjelenítést.

A View modell elemnek megfeleltethető a View osztály, ami a megjelenítésért felel, a Controller állapotának megfelelő ablakot jeleníti meg a kijelzőn. Így amennyiben menüt, vesztett vagy nyert játékot határoz meg a Controller az kerül kirajzolásra. Ugyanígy a Controlleren keresztül kaphatja meg azokat az pályaelemeket, amelyeknek meg kell jelenniük a pályán. A View osztály JForm őssztályának köszönhetően az alapvető ablakozási feladatokat is ellátja, így a pályaelemeket JComponent-ként kirajzolja.



59. ábra MVC grafikus rendszer architektúrája

11.2.2 A felület osztály-struktúrája



60. ábra Grafikus megjelenítés osztály diagram

11.3 A grafikus objektumok felsorolása

11.3.1 Model

- **Felelősség**

Összefogja a rendszerben levő egységeket. Meghívja a mozdonyok `move()` metódusát. Lehetőséget biztosít alagút bejáratok építéséhez, rombolásához, illetve a váltók állításához.

- **Ősosztályok:** -

- **Interfészek:** -

- **Attribútumok:**

- **Metódusok**

- - **Load():** Felépíti a pályát, létrehozza a kapcsolatokat.
- - **ArrayList<Rail> makeParts(n1 Node, n2 Node):** A két átadott pont között Rail Node-okat szűr be.
- - **int getDistance(x int, y int, x2 int, y2 int):** Két pont távolságát adja meg.
- - **addTunnelEntrance(p Node, on Node, n Node):** On node-ot cseréli le alagútbejáratra.
- - **takeTraintoMap():** A pályához ad hozzá véletlenszerűen összeállított szerelvényt.
- - **List<Rail> getRails():** Visszaadja a síneket.
- - **List<Cross> getCrosses():** Visszaadja a kereszteződéseket.
- - **List<Switch> getSwitches():** Visszaadja a váltókat.
- - **List<TunnelEntrance> getTunnelEntrance:** Visszaadja az alagútbejáratokat.
- - **List<Station> getStations():** Visszaadja az állomásokat.
- - **List<Engine> getEngines():** Visszaadja a mozdonyokat.
- - **List<Car> getCars():** Visszaadja a kocsikat.
- - **List<CoalCar> getCoalCars():** Visszaadja a szeneskocsikat.

11.3.2 View

- **Felelősség**

A View osztály az MVC modell fontos eleme, a View-t valósítja meg. Felel a pálya és a menük megjelenítéséért. A Controller által megadott elemeket rajzolja ki, a változásokat újra rajzolja.

- **Ősosztályok:** swing.JFrame

- **Interfészek:** -

- **Attribútumok**

- - **EngineGraphics[] engine:** A Manager-től átvett Engine objektum grafikus változatának (EngineGraphics) gyűjteménye, azok kirajzolásához.

- - **CarGraphics[] cars**: A Manager-től átvett Car objektum grafikus változatának (CarGraphics) gyűjteménye, azok kirajzolásához.
- - **CoalCarGraphics[] coalCars**: A Manager-től átvett CoalCar objektum grafikus változatának (CoalCarGraphics) gyűjteménye, azok kirajzolásához.
- - **RailGraphics[] rails**: A Manager-től átvett Rail objektum grafikus változatának (RailGraphics) gyűjteménye, azok kirajzolásához.
- - **TunnelEntranceGraphics[] tunnelEntrances**: A Manager-től átvett TunnelEntrance objektum grafikus változatának (TunnelEntranceGraphics) gyűjteménye, azok kirajzolásához.
- - **StationGraphics[] stations**: A Manager-től átvett Station objektum grafikus változatának (StationGraphics) gyűjteménye, azok kirajzolásához.
- - **State state**: Egy vezérlő, amely a játék aktuális állapotát reprezentálja. Game állapot esetén ezen keresztül kapja meg a kirajzolandó elemeket.
- - **Model map**: Pálya
- - **Status status**: A játék állapota.
- **Metódusok**
 - + **View()**: Konstruktor. Létrehozza a főablakot, inicializálja a tárolókat.
 - + **View()**: Konstruktor. Létrehozza a főablakot, inicializálja a tárolókat.
 - + **updateScreen()**: Frissíti az ablakot.
 - + **updatePanel(buttons: String[])**: Frissíti a gombokat, amelyeket meg kell jeleníteni.
 - + **setState(s: State)**: Beállítja az aktuális játékállapotot.
 - + **setMap(m: Model)**: Beállítja a pályát.
 - + **setStatus(s: Status)**: A játékállapothoz tartozó enum beállítása.
 - - **updateMap()**: Frissíti a pálya elemek listáját, kirajzolja azokat.
 - - **drawPanel()**: Kirajzolja a panelt az aktuális menühöz.
 - - **drawRails()**: Kirajzolja a síneket.
 - - **refreshEngines()**: Frissíti a megjelenítendő mozdonyok listáját. Ha már valamelyik nincs a pályán törli az ahhoz tartozó grafikus objektumot, újak esetén újakat hoz létre.
 - - **refreshCars()**: Frissíti a megjelenítendő kocsik listáját. Ha már valamelyik nincs a pályán törli az ahhoz tartozó grafikus objektumot, újak esetén újakat hoz létre.
 - - **refreshCoalCar()**: Frissíti a megjelenítendő szeneskocsik listáját. Ha már valamelyik nincs a pályán törli az ahhoz tartozó grafikus objektumot, újak esetén újakat hoz létre.
 - - **refreshStations()**: Frissíti a megjelenítendő állomások listáját. Ha már valamelyik nincs a pályán törli az ahhoz tartozó grafikus objektumot, újak esetén újakat hoz létre.
 - - **refreshTunnelEntrances()**: Frissíti a megjelenítendő alagút bejáratok listáját. Ha már valamelyik nincs a pályán törli az ahhoz tartozó grafikus objektumot, újak esetén újakat hoz létre.

11.3.3 EngineGraphics

- **Felelősség**

A mozdony kirajzolása.

- **Ősosztályok**: -

- **Interfészek:** -
- **Attribútumok**
 - - **engine: Engine:** Az Engine osztály valósítja meg a mozdonyt. A vagonokat vontatásáért felel. Osztály leírása: Részletes tervek 8.1.9
- **Metódusok**
 - + **EngineGraphics(e: Engine):** Beállítja az attribútumát a paraméterül kapott változóval.
 - +**void draw(g: Graphics):** Az attribútumban szereplő pályaelemet rajzolja ki.
 - +**Engine getEngine():** Engine getter metódus.

11.3.4 CarGraphics

- **Felelősség:** A kocsik kirajzolása.
- **Ősosztályok:** -
- **Interfészek:** -
- **Attribútumok**
 - - **car: Car:** A vagon megvalósító osztály. A vagonnak színe van. A rajta utazó összes utas a vagon színével azonos színű állomáson leszáll, ha az a mozdonyhoz legközelebbi nem üres vagon. Utasok szállhatnak fel a vagonba, ha üres és a színe azonos az állomásával. Osztály leírása: Részletes tervek 8.1.11
- **Metódusok**
 - + **CarGraphics(c: Car):** Beállítja az attribútumát a paraméterül kapott változóval.
 - +**void draw(g: Graphics):** Az attribútumban szereplő pályaelemet rajzolja ki.
 - +**Car getCar():** Car getter metódus.

11.3.5 TunnelEntranceGraphics

- **Felelősség:** Az alagút bejáratok kirajzolása.
- **Ősosztályok:** -
- **Interfészek:** -
- **Attribútumok**
 - -**tunnelEntrance: TunnelEntrance:** A TunnelEntrance osztály valósítja meg az alagút bejáratot a pályán. Két lerakott bejárat esetén alkot alagutat és irányítja át a vonatot az egyik bejáratból a másikba. Amennyiben nincs második bejárat lerakva a vonat áthalad rajta. Osztály leírása: Részletes tervek 8.1.17
- **Metódusok**
 - + **TunnelEntranceGraphics(t: TunnelEntrance):** Beállítja az attribútumát a paraméterül kapott változóval.

- **+void draw(g: Graphics):** Az attribútumban szereplő pályaelemet rajzolja ki.
- **+TunnelEnterance getTunnelEnterance():** TunnelEnterance getter metódus.

11.3.6 CoalCarGraphics

- **Felelősség:** A szeneskocsik kirajzolása.
- **Ősosztályok:** -
- **Interfészek:** -
- **Attribútumok**
 - - **coalCar: CoalCar:** A szeneskocsit megvalósító osztály. Nem utazhatnak rajta utasok. A szeneskocsi a mozdonyt követően a szerelvény bármely részén elhelyezkedhet. Osztály leírása: Részletes tervek 8.1.10
- **Metódusok**
 - **+ CoalCarGraphics(c: CoalCar):** Beállítja az attribútumát a paraméterül kapott változóval.
 - **+void draw(g: Graphics):** Az attribútumban szereplő pályaelemet rajzolja ki.
 - **+CoalCar getCoalCar():** CoalCar getter metódus.

11.3.7 StationGraphics

- **Felelősség:** Az állomások kirajzolása.
- **Ősosztályok:** -
- **Interfészek:** -
- **Attribútumok**
 - - **station: Station:** A Station osztály a megállót valósítja meg. A megálló színével azonos színű vagonokból az utasok leszállnak itt. Azonos színű üres vagonba utasok szállhatnak fel. Osztály leírása: Részletes tervek 8.1.13
- **Metódusok**
 - **+ StationGraphics(s: Station):** Beállítja az attribútumát a paraméterül kapott változóval.
 - **+void draw(g: Graphics):** Az attribútumban szereplő pályaelemet rajzolja ki.
 - **+Station getStation():** Station getter metódus.

11.3.8 Game

- **Felelősség:**
- **Ősosztályok:** -
- **Interfészek:** State

- **Attribútumok**
 - **-map: Model:** A pálya és a hozzátartozó elemek.
 - **-numberOfTrains: int:** Az indítandó vonatok száma
 - **-waitingTime: long:** Két vonat indítása közötti várakozási idő.
 - **-view: View:** A megjelenítésért felelős objektum.
 - **-output: Status:** A játék aktuális állapota.
- **Metódusok**
 - **+void setOutput(s: Status):** Beállítja a játék aktuális állapotát.

11.3.9 Menu

- **Felelősség:** -
- **Ősosztályok:** -
- **Interfészek:** State
- **Attribútumok**
 - **-view: View:** A megjelenítésért felelős objektum.
- **Metódusok**
 - **+void setOutput(s: Status):** Beállítja a játék aktuális állapotát.

11.3.10 Pause

- **Felelősség:**
- **Ősosztályok:** -
- **Interfészek:** State
- **Attribútumok**
 - **- view: View:** A megjelenítésért felelős objektum.
- **Metódusok**
 - **+void setOutput(s: Status):** Beállítja a játék aktuális állapotát.

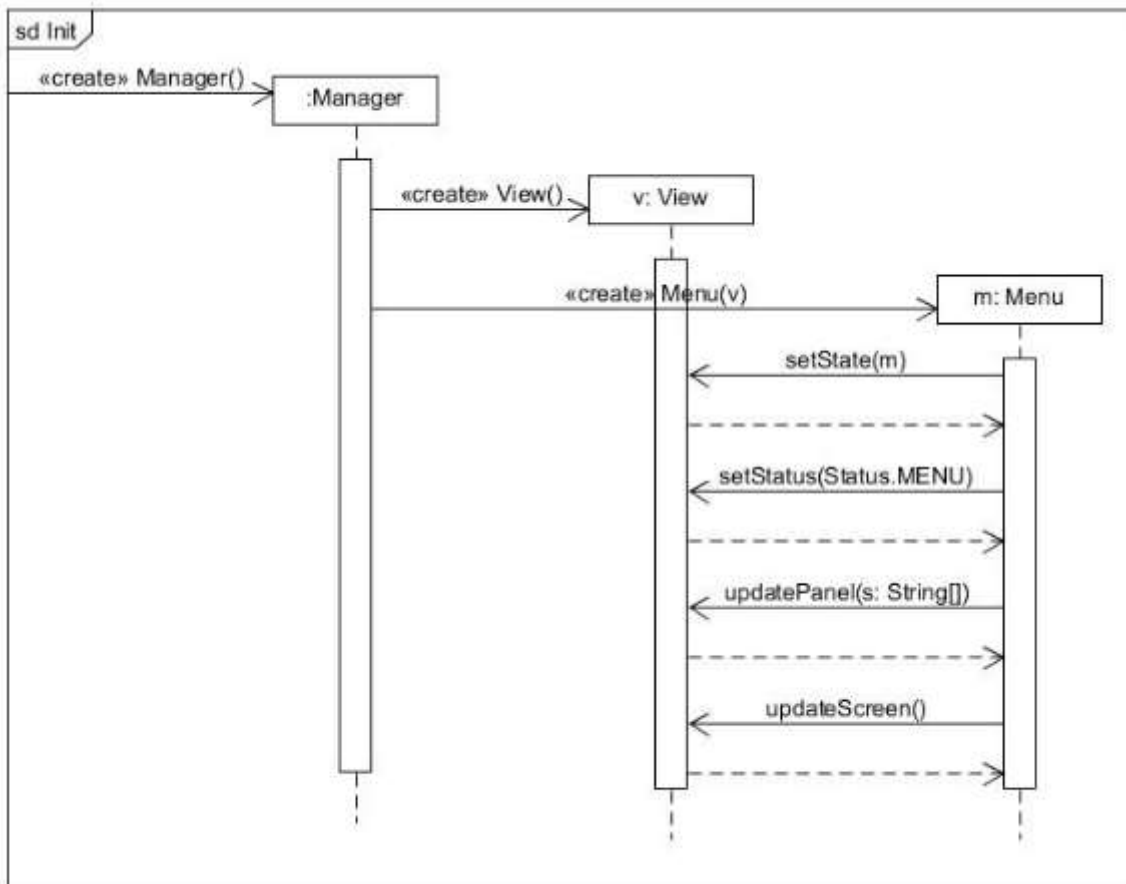
11.3.11 End

- **Felelősség:**
- **Ősosztályok:** -
- **Interfészek:** State
- **Attribútumok**
 - **-view: View:** A megjelenítésért felelős objektum.

- **Metódusok**
 - **+void setOutput(s: Status):** Beállítja a játék aktuális állapotát.

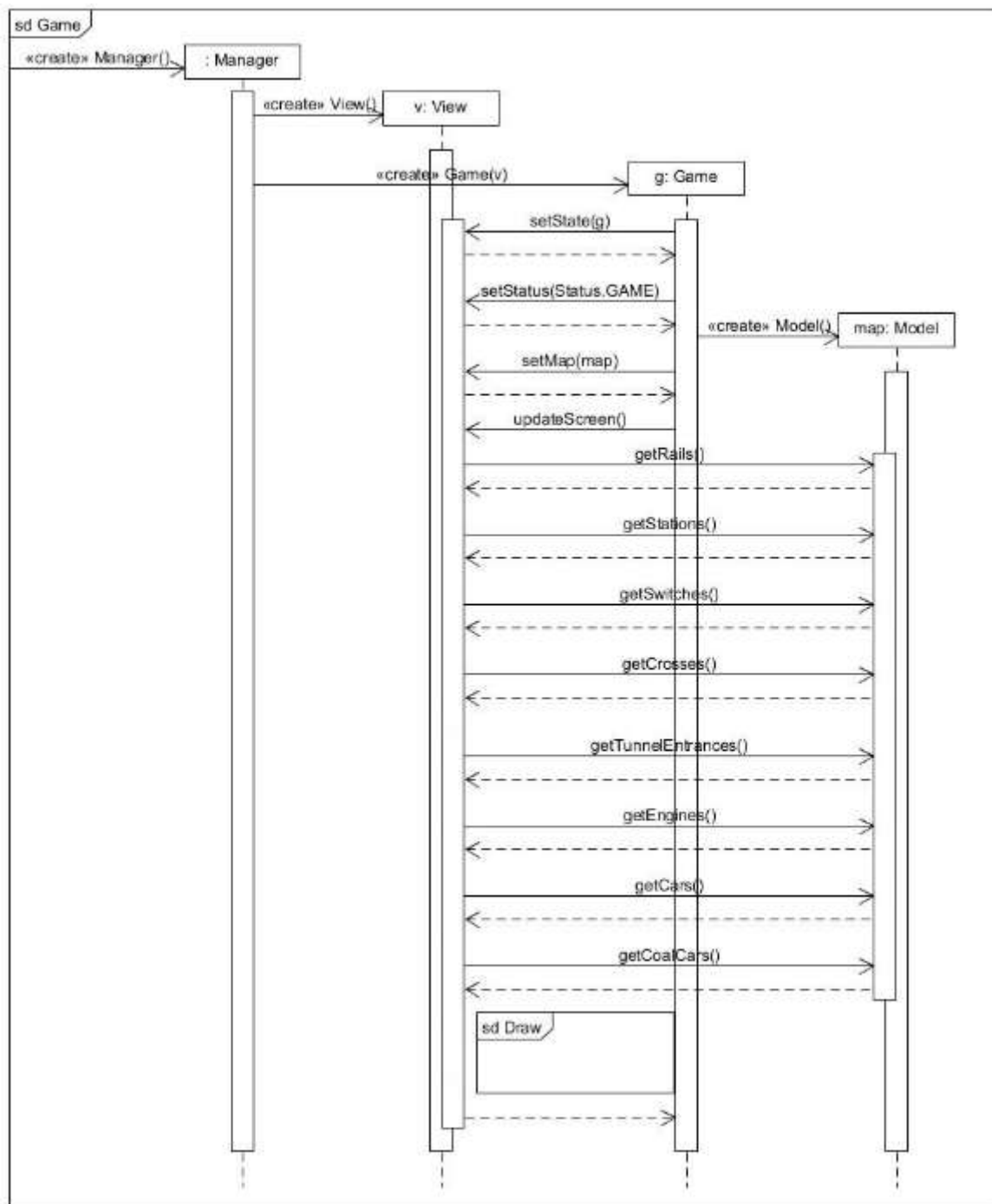
11.4 Kapcsolat az alkalmazói rendszerrel

11.4.1 Init



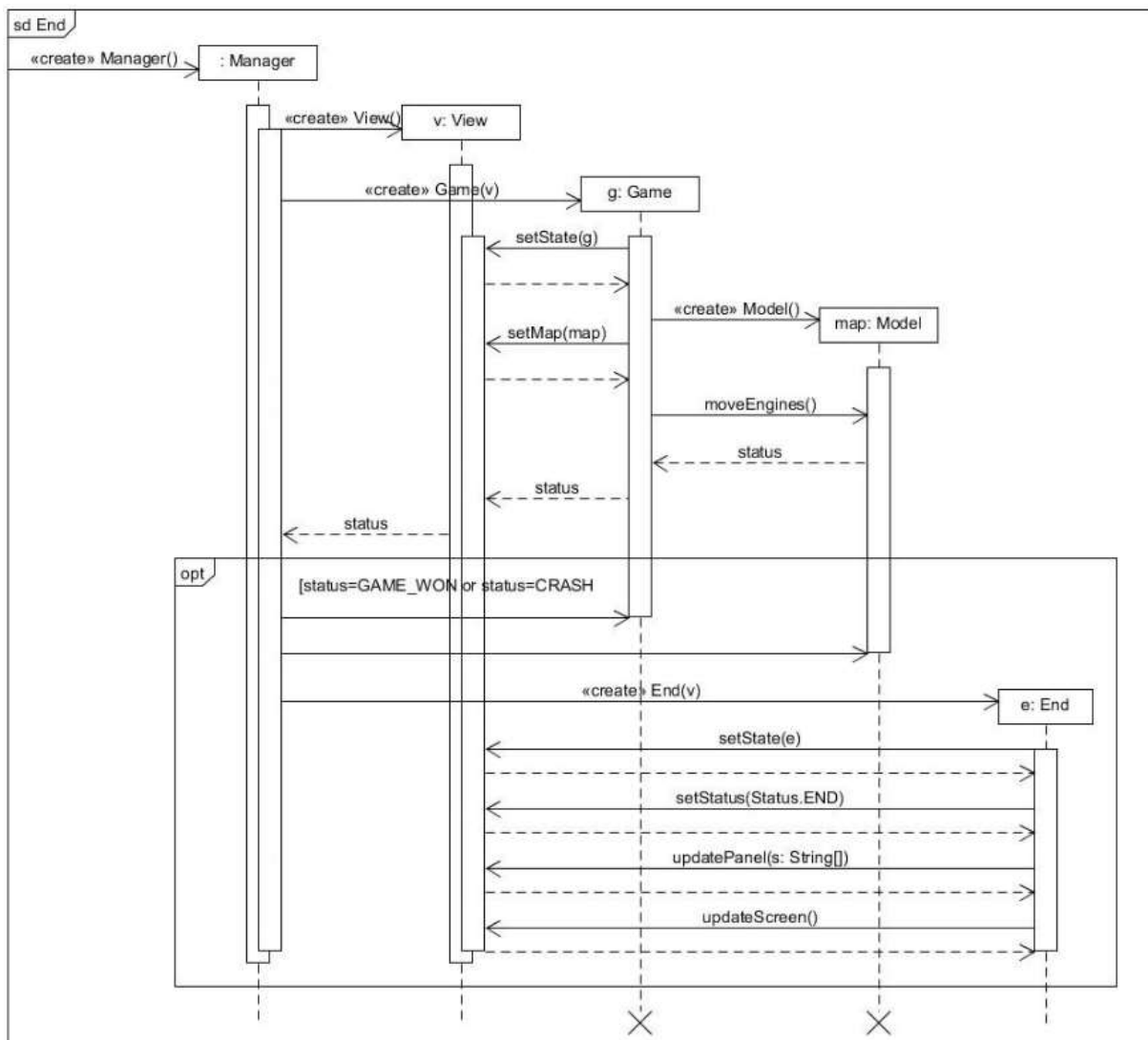
61. ábra Grafikus felület Init szekvencia diagram

11.4.2 Game



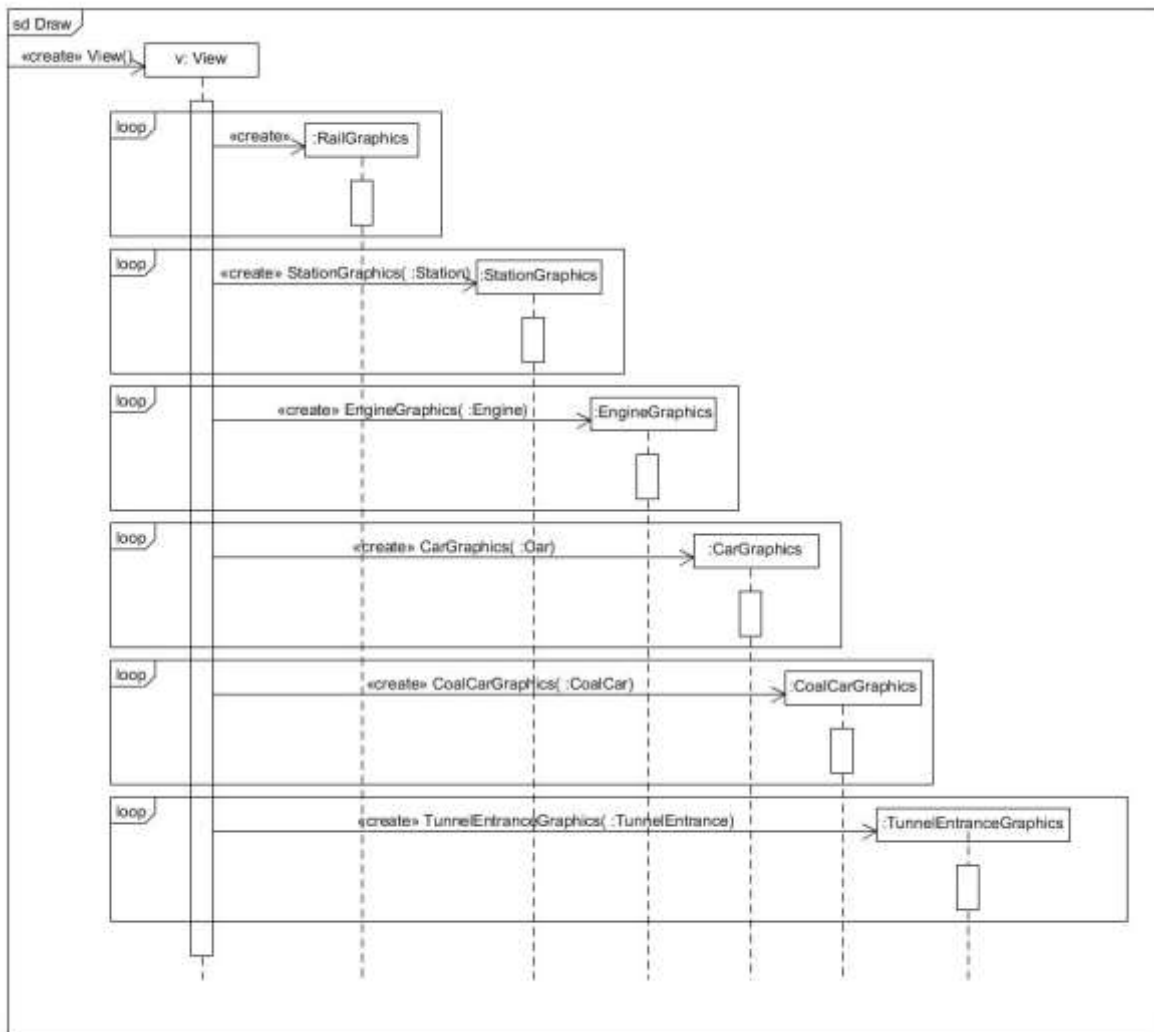
62. ábra Grafikus felület Game szekvencia diagram

11.4.3 End



63. ábra Grafikus felület End szekvencia diagram

11.4.4 Draw



64. ábra Grafikus felület Draw szekvencia diagram

11.5 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.04.20. 12:00	4 óra	Tóth Kunkli Zoltay Zahorán	Értekezlet: Feladatok kiosztása az alábbiak szerint. A grafikához használt eszközök, objektumok átbeszélése.
2017.04.23. 18:00	5 óra	Tóth	11.1 Grafikus Interfész
2017.04.21. 20:00	5 óra	Kunkli	11.2.2 Osztálydiagram
2017.04.22. 16:00 2017.04.23. 18:00	2,5 óra 2,5 óra	Zoltay	11.3 Grafikus objektumok felsorolása 11.4 Kapcsolat az alkalmazói rendszerrel
2017.04.23. 12:00	5 óra	Zahorán	11.2.1 Felület működési elve

13. Grafikus változat beadása

13 Grafikus változat beadása

13.1 Fordítási és futtatási útmutató

13.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Car.java	4 Kb	2017/05/10	A kocsit megvalósító osztály.
CarGraphics.java	2 Kb	2017/05/10	A kocsik kirajzolásáért felelős osztály.
CoalCar.java	3 Kb	2017/05/10	A szeneskocsit megvalósító osztály.
CoalCarGraphics.java	1 Kb	2017/05/10	A szeneskocsik kirajzolásáért felelős osztály.
Color.java	2 Kb	2017/05/10	A színeket megvalósító enum.
Cross.java	4 Kb	2017/05/10	A kereszteződést megvalósító osztály.
End.java	2 Kb	2017/05/10	A játék végét megvalósító menü
Engine.java	2 Kb	2017/05/10	A mozdonyt megvalósító osztály.
EngineGraphics.java	1 Kb	2017/05/10	A mozdonyok kirajzolásáért felelős osztály.
Game.java	3 Kb	2017/05/10	A játékot megvalósító osztály.
Main.java	1 Kb	2017/05/10	Main osztály. A program belépési pontja.
Manager.java	2 Kb	2017/05/10	A menüket/állapotokat kezelő osztály.
MapItem.java	2 Kb	2017/05/09	A pálya elemeket megvalósító absztrakt osztály.
Menu.java	1 Kb	2017/05/10	A játék indításáért felelős osztály.
Model.java	24 Kb	2017/05/10	A parancsok feldolgozásáért, értelmezéséért felel.
Node.java	3 Kb	2017/05/10	A csomópontokat megvalósító absztrakt osztály.
Pause.java	1 Kb	2017/05/10	A szünetet megvalósító osztály.
Rail.java	1 Kb	2017/05/10	A sínt megvalósító osztály.
RailGraphics.java	1 Kb	2017/05/10	A sinek kirajzolásáért felelős osztály.
State.java	1 Kb	2017/05/10	Az állapotok őse, interfésze
Station.java	3 Kb	2017/05/10	Az állomást megvalósító osztály.

StationGraphics.java	1 Kb	2017/05/10	Az állomások kirajzolásáért felelős osztály.
Status.java	1 Kb	2017/05/09	Az állapotokat megvalósító enum.
Switch.java	3 Kb	2017/05/10	A váltót megvalósító osztály.
Train.java	5 Kb	2017/05/09	A vonat elemeket megvalósító absztrakt osztály.
TunnelEntrance.java	1 Kb	2017/05/10	Az alagút bejáratot megvalósító osztály.
TunnelEntranceGraphics.java	2 Kb	2017/05/10	Az alagut kirajzolásáért felelős osztály.
View.java	16 Kb	2017/05/10	A megjelenítésért felelős osztály.

13.1.2 Fordítás és telepítés

A fenti fájlok .java kiterjesztésűek. Java nyelven íródtak, java kódot tartalmaznak, amelyek még nem futtatható állományok. A JDK (Java Development Kit) segítségével operációs rendszerünknek megfelelő futtatható állomány elkészítése a cél.

A javac.exe állomány felel a fordításért, a letöltött JDK mappájában található meg `..\bin\javac.exe` helyen. Parancssori futtatásához legegyszerűbb, ha a bin mappában Shift+Jobb gomb lenyomása után a Parancsablak nyitása itt lehetőséget választjuk.

Majd a javac programot elindítjuk a megfelelő paraméterekkel, amelyek a projektben szereplő java fájlok. Több fájl fordítása esetén a tartalmazó mappa elérési útját érdemes megadni `*.java` kiegészítéssel, és elérni a `-encoding UTF8` kapcsolót. Így az összes .java kiterjesztésű fájl lefordul. Az egyszerűség érdekében másoljuk ki az src mappát a C főkönyvtárba.

Pl: `javac -encoding UTF8 C:\src\project*.java`

Amennyiben mindent jól csináltunk megjelenik minden .java fájl mellett egy azonos nevű .class kiterjesztésű fájl.

13.2 Futtatás

A futtatáshoz az előzőekhez hasonlóan kell eljárunk, javac.exe helyett a java.exe-t kell megfelelően paramétereznünk az indításhoz.

A fordítás után lépünk be az 'src' mappába, majd indítsuk el a programot a következő paranccsal: `java project.Main`

Ha a letöltött JDK mappából futtatjuk a programot akkor a parancs kiegészül egy paraméterrel, mely megadja a lefordított fájlok elérési útját.

Pl.: `java -cp C:\src\project.Main`

13.3 Értékelés

Tag neve	Munka százalékban	Aláírás
Tóth Attila	25%	
Kunkli Richárd Zsolt	25%	
Zahorán László	25%	
Zoltay Marcell	25%	

13.4 Napló

Kezdet	Időtartam	Résztevők	Leírás
2017.05.08. 15:00	4,5 óra	Tóth Kunkli Zoltay Zahorán	Értekezlet: Problémás részek, vitatása, Rájöttünk, hogy ezt csak közösen lehet
2017.05.09. 20:00	8 óra	Tóth Kunkli Zoltay Zahorán	Debugolás: Hibák közös vitatása, javítása
2017.05.09. 12:00	8 óra	Tóth	Node-ok átírása
2017.05.09. 12:00	8 óra	Kunkli	View megvalósítása,
2017.05.09. 12:00	8 óra	Zoltay	Graphics osztályok elkészítése, kirajzolás megvalósítása
2017.05.09. 12:00	8 óra	Zahorán	Pálya elkészítése Pálya feldarabolása

14. Összefoglalás

14 Összefoglalás

14.1 A projektre fordított összes munkaidő

Tag neve	Munkaidő (óra)
Tóth Attila	79
Kunkli Richárd	79
Zahorán László	79
Zoltay Marcell	79
Összesen	316

- **A feltöltött programok forrássorainak száma**

Fázis	Kódsorok száma
Szkeleton	1216
Prototípus	2044
Grafikus változat	2891
Összesen	6151

14.2 • Projekt összegzés

14.2.1 Mit tanultak a projektből konkrétan és általában?

A projekt során lehetőségünk volt az egyéni feladatok után csapatban feladatot megoldani, ezzel kapcsolatban szükség volt, eddig keveset használt kommunikációs képességekre. Soha nem a saját ötletünk volt az egyetlen, így rendszeresen a több közül kellett választani, ehhez mindenkinek ki kellett tudni fejtenie ötletét, érvelni mellette vagy ellene, majd döntéseket meghozni.

Konkrét új szakmai ismereteket az MVC tervezési minta gyakorlati alkalmazása nyújtott egy ilyen nagyobb projekten és a Java nyelv grafikus eszközeinek használata.

14.2.2 Mi volt a legnehezebb és a legkönnyebb?

A legnehezebb feladat a csapat számára az idő megfelelő beosztása volt, hogy a részfeladatokat mindenki megoldja időre, nem elhanyagolva a saját egyéb teendőit és a csapat érdekét.

A legkönnyebb feladat a már megtervezett, dokumentált részek megvalósítása volt, de sajnos volt rá példa, hogy a tervezésnél nem gondoltunk minden szempontra, így azon változtatni kellett.

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

Igen a projekt jelentős részében, talán a grafikus rész jelentett ezalól számunkra kivételt.

14.2.4 Ha nem, akkor hol okozott ez nehézséget?

A grafikus interfész tervezése, megvalósítása az eddigi tanulmányok során kötelezően nem került elő csak felhasználói felületek szintjén, ami kevés ráfordítással pótolható de a 13-14. héten nehéz beütemezni annak megismerését, és előre megtervezni az interfészt, hogy az tényleg működőképes legyen és megfeleljen minden elvárásnak.

14.2.5 Milyen változtatási javaslatuk van?

Szerintünk a grafikus tervezés és beadás kaphatna nagyobb időszületet, ebbe beleértve a prototípus hibáinak javítását, mivel a teljes működést tesztek formájába nehéz volt ellenőrizni így több modell szintű hiba is felbukkant a grafikai rész során.

14.2.6 Milyen feladatot ajánlanának a projektre?

A javaslatunk egy „okos falu” megvalósítása, amely a jövő önszervező nagyvárosainak kicsinyített változata. A falunak rendelkeznie kell úthálózattal jelzőlámpákkal, és különböző járművekkel, buszjáratokkal amelyek ugyanazt az utvonalat járkák be egész nap. Otthonoknak és munkahelyeknek, amelyek között autók és biciklik közlekednek. A városban az intelligens jelzőlámpa rendszernek biztosítania kell a folyamatos haladást szemelőt tartva, hogy a tömegközlekedés és a kerékpár használat jó hatással van a falu életére így ők a lehető leggyorsabban haladjanak. A rendszernek kész kell lennie egy kereszteződésben bekövetkezett baleset kezelésére is. Ami magától nem történhet meg így a felhasználó szabályozhatja. Ekkor a környező keresztezések kommunikálva a járművekkel, megoldják, hogy ne menjenek arra és ő se enged oda több járművet.

Szerintünk ezzel a témával kapcsolatos feladat érdekes lenne, könnyen nehezíthető új elemekkel vagy könnyíthető.

14.2.7 Egyéb kritika és javaslat