3. Analízis modell kidolgozása

3.1 Objektum katalógus

3.1.1 Player

A játékos játékbeli megnevezése. Ő adja ki a parancsokat a rendszer (System) számára, így irányítva a játék működését. Lehetősége van a váltók állítgatására az ütközések elkerülése érdekében, valamint alagút elhelyezésére az erre kijelölt helyeken.

3.1.2 Carriage

Ez a kocsi, más néven vagon objektumok. Ezek alkotják a vonatokat. Eltérő színűek lehetnek, és a színüktől függően szállhatnak majd le az utasok a megfelelő színű állomásokon, amiken áthaladnak.

3.1.3 Train

Ez nem más mint a vonatunk. Kocsikból áll és alapértelmezetten mozog a pályán, követve a síneket az adott irányba. A játékos képes befolyásolni a vonat irányát a 3.1.1 pontban leírt utasítások segítségével. Egyszerre több vonat is lehet a pályán. A vonatok kizárólag a síneken közlekednek, követik a váltók által beállított irányt, ami a vonat kisiklásához, vagyis felrobbanásához is vezethet, valamint használják az épített alagutat (amennyiben van ilyen).

3.1.4 TunnelEntrance

Egy olyan speciális sín, aminek van két bejárata és amennyiben a vonat útjába ilyen kerül és engedélyezve van, vagyis aktív, akkor alapértelmezetten a vonat azon halad tovább. Amennyiben inaktív, a vonat tovább halad figyelmen kívül hagyva azt.

3.1.5 Rail

A sín, a vonat számára kijelölt útvonalak fő alkotóeleme, mondhatni a pálya gerincét képző objektum. A vonat ezek mentén, vagyis ezeken halad.

3.1.6 Switch

A pályának olyan eleme, amely biztosan sínre illeszkedik. A játékos ennek segítségével irányítja a vonatokat a csomópontokba, ezek segítségével tudja meghatározni a vonat továbbhaladásának az irányát a kritikus pontokban (kereszteződés, elágazás stb.).

3.1.7 Station

Olyan pálya elem, amely a sínek mentén helyezkedik el. Minden állomásnak van valamilyen színe, ami meghatározza, hogy az ott elhaladó vonat mely kocsijáról szállhatnak le utasok annál az állomásnál.

3.1.8 MapElement

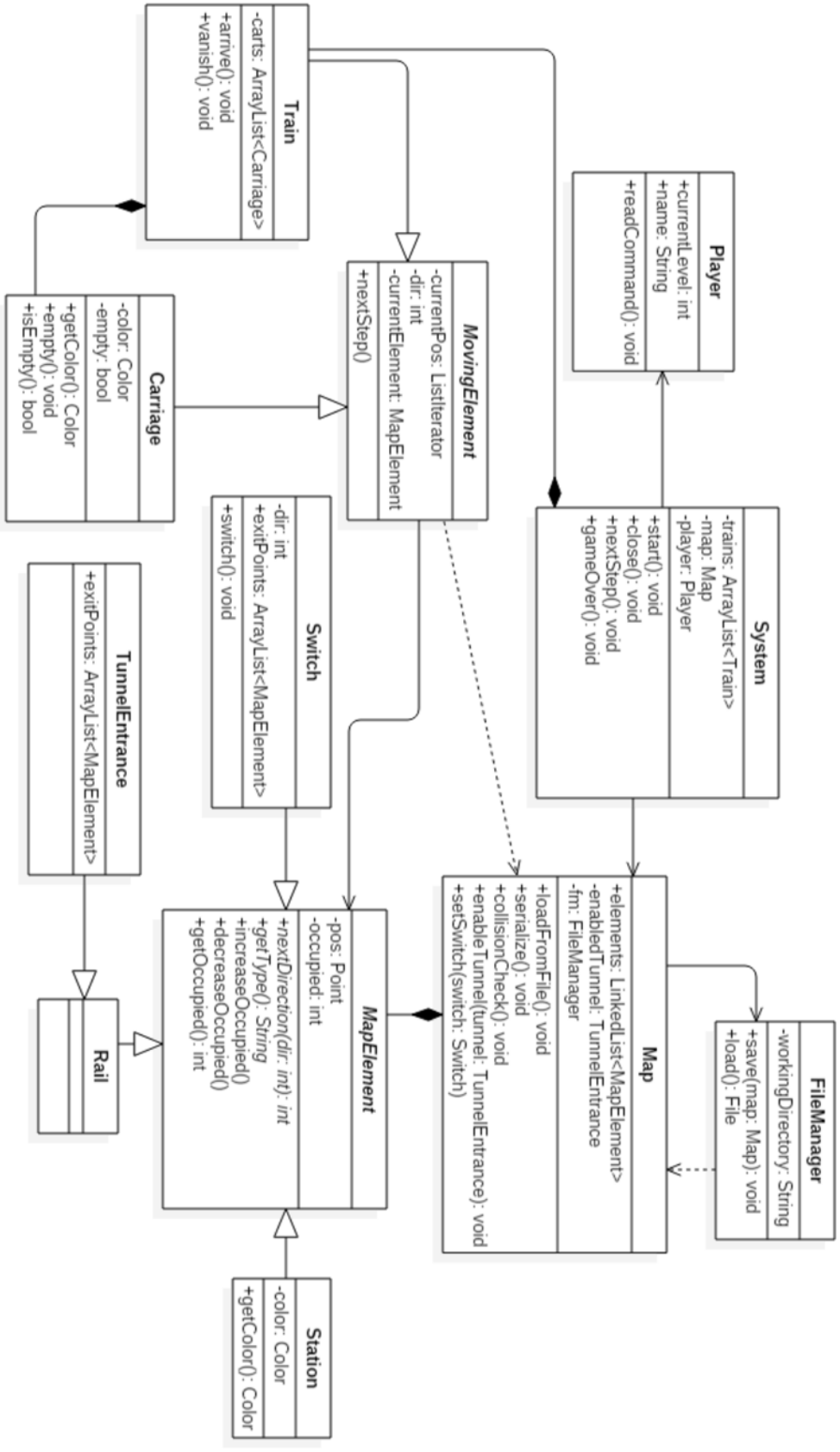
A pályát felépítő elemek összessége. Mondhatni a terep asztalunk (pályánk) építőköveinek gyűjteménye.

3.1.9 Map

A pálya elemeket tartalmazó objektum, amely egy listában tárolja azokat, segítve a vonatok

haladását a pályán. A System osztály ennek az objektumnak a segítségével tudja, hogy a pálya melyik pontjában pontosan milyen pálya elem található.

3.2 Statikus struktúra diagramok



3.3 Osztályok leírása

3.3.1 Carriage

- **Felelősség**

A kocsi osztálya. Mivel a vonat része, tud mozogni a pályán. A követelményeknek megfelelően van színe, illetve ki tud ürülni, ha leszállnak róla az utasok.

- **Ősosztályok**

MovingElement

- **Interfészek**

Nincs.

- **Attribútumok**
  - **Color color**: a kocsi jelenlegi színe.
  - **boolean empty**: megmondja, hogy üres-e a kocsi (azaz hogy leszálltak-e az utasok róla).

- **Metódusok**
  - **Color getColor()**: visszaadja a kocsi jelenlegi színét. Az állomásra érkezés logikájánál lesz haszna, amikor ellenőrizni kell a kocsi és az állomás színét.
  - **void empty()**: kiüríti a kocsit, azaz az empty boolean-t igazra billenti.
  - **void isEmpty()**: visszaadja, hogy üres-e a kocsi.

3.3.2 FileManager

- **Felelősség**

A játékban a fájlkezelést végzi. Kezeli a fájlok tárolásának helyét, jellegét, a Map szerializációs függvényeivel szoros összefüggésben áll, hiszen a szerializált adatot ez az osztály írja ki fájlba.

- **Ősosztályok**

Nincs.

- **Interfészek**

Nincs.

- **Attribútumok**
  - **String workingDirectory:** a lementett fájlok helyét tároló karakterfüzér.

- **Metódusok**
  - **void save(Map)**: a paraméterként kapott állást lementi fájlba.
  - **File load()**: a lementett állást visszatölti, hogy a Map majd tudjon vele dolgozni.

3.3.3 Map

- **Felelősség**

A pályaelemek tárolásáért illetve az ezekkel kapcsolatos műveletek elvégzéséért felelős.

- **Ősosztályok**

Nincs.

- **Interfészek**

Nincs.

- **Attribútumok**
  - **LinkedList<MapElement> elements**: a pálya elemek itt vannak eltárolva. Duplán láncolt lista, mely a könnyű iterálást hivatott segíteni.
  - **TunnelEntrance enabledTunnel**: a jelenleg megépített egyetlen alagút.

- **Metódusok**
  - **void loadFromFile()**: pályaelemek beolvasása a megadott fájlból
  - **void serialize()**: A System által utasított függvény a pálya jelenlegi állásának elmentésére. Meghívja a FileManager fájlba író save() függvényét.
  - **void collisionCheck()**: a System nexStep függvénye hívja meg, végignézi minden, a mapen található elemre, hogy mennyi a rajtuk található elemek száma, és ha ez >1, akkor ütközést detektál. Az alapötlet, hogy pályaelemen egyidőben csak egy másik elem tartózkodhat legfeljebb.
  - **void enableTunnel (tunnel: TunnelEntrance)**: A jelenleg engedélyezett tunnelt lehet vele állítani, felülírja a jelenlegit, ezáltal biztosítva, hogy egyszerre csak egy van engedélyezve.
  - **setSwitch(switch: Switch)**: meghívja a paraméterül kapott Switch switch függvényét.

3.3.4 MapElement

- **Felelősség**

A statikus pálya elemek absztrakt osztálya. Egy Map tartalmazza az összeset. Lehet Switch, Rail, Station, TunnelEntrance. Tudja, hogy hány másik elem van rajta, és egy mozgó elem jelenlegi pozíciójából és irányából ki tudja számolni, hogy mi lesz a következő iránya és pozíciója.

- **Ősosztályok**

Nincs.

- **Interfészek**

Nincs.

- **Attribútumok**
  - **int occupied:** azt tárolja, hogy hány MovingElement van rajta egyszerre.

- **Metódusok**
  - **String getType()**: visszatér a MapElement konkrét típusával (Switch, Rail, Station, TunnelEntrance).
  - **MapElement nextPosition(int)**: egy mozgó elem jelenlegi irányából kiszámítja, hogy mi lesz a következő MapElement, amire lépni fog.
  - **int nextDirection(int):** ha szükség van rá (pl.: switch esetében), egy mozgó elem jelenlegi irányából kiszámolja a következő irányt.
  - **int getOccupied():** visszaadja, hogy hány MovingElement tartózkodik az adott pályaelemen. Hasznos lesz akkor, amikor a ütközéseket akarjuk vizsgálni (hiszen ha egy MapElement-en pl. két Carriage tartózkodik, akkor az a játék végét jelenti)
  - **void increaseOccupied():** növeli az occupied változó értékét eggyel.
  - **void decreaseOccupied():** csökkenti az occupied változó értékét eggyel.

3.3.5 MovingElement

- **Felelősség**

A pálya mozgó elemeinek interfésze. Tudnia kell az időbeli előrehaladást kezelnie, erre szolgál a nextStep függvény.

- **Attribútumok**
  - **ListIterator currentPos:** egy iterátor, mely a lista tetszőleges bejárását teszi lehetővé.
  - **int dir:** irány
  - **MapElement currentElement:** az az elem, amin jelenleg tartózkodik.

- **Metódusok**
  - **void nextStep()**: a léptetést kezelő metódus. Kiszámoltatja a következő pozíciót, a következő irányt. Beállítja a MapElement-ek occupied változóját (csökkenti, ha lelép róla; növeli, ha rálép).

3.3.6 Player

- **Felelősség**

A felhasználó osztálya. A felhasználói parancsok bekérése, illetve az azoknak megfelelő folyamatok elindítása a felelőssége.

- **Ősosztályok**

Nincsenek.

- **Interfészek**

Nincsenek.

- **Attribútumok**
  - **int currentLevel:** A játékos jelenlegi pályájának számát tárolja.
  - **String name:** A játékos neve.

- **Metódusok**
  - **void readCommand()**: játékos által megadott parancsok beolvasása**.**

3.3.7 Rail

- **Felelősség**

A sín modellezésére szolgáló osztály. Alapvetően nincsen semmilyen feladata sem tulajdonsága, mely különbözik a MapElementtől, de mégis szerettük volna logikailag elkülöníteni a többi osztálytól.

- **Ősosztályok**

MapElement.

- **Interfészek**

Nincs.

- **Attribútumok:** nincs
- **Metódusok:** nincs

3.3.8 Station

- **Felelősség**

A feladat kiírásból származó állomás modellezését megvalósító osztály.

- **Ősosztályok**

MapElement.

- **Interfészek**

Nincs.

- **Attribútumok**
  - **Color color:** az állomás színe
- **Metódusok**
  - **Color getColor()**: az állomás színének lekérése.

3.3.9 Switch

- **Felelősség**

A feladatkiírásból származó váltó modellezését megvalósító osztály.

- **Ősosztályok**

MapElement.

- **Interfészek**

Nincs.

- **Attribútumok**
  - **int dir:** a váltó állása. Ebben lesz eltárolva valamilyen konvenció szerint.
  - **ArrayList<MapElement> exitPoints:** a váltó lehetséges kimenetei. Ezek között iterál a dir.

- **Metódusok**
  - **void switch()**: következő állásba történő állítás. A játékos végzi, kattintással, így az iránya egy megadott szabály szerint fog változni.

3.3.10 System

- **Felelősség**

A játék különböző komponenseit összefogó elem, mely elvégzi az indítást, bezárást, lépések ütemezését és a játékmenetet összefogja.

- **Ősosztályok**

Nincsenek.

- **Interfészek**

Nincsenek

- **Attribútumok**
  - **ArrayList<Train> trains:** pályán lévő vonatokat tároló lista
  - **Player player:** a játékost tárolja el, aki éppen a játékkal játszik
  - **Map map:** az éppen aktuális pálya a játékban

- **Metódusok**
  - **void start()**: a játék inicializálásáért felelős, létrehozza a pályát és feltölti elemekkel.
  - **void close()**: a játékállás elmentését végzi, meghívja a pálya sorosító függvényét.
  - **void nextStep():** az ütközés ellenőrzésének meghívását végzi, valamint a mozgó elemek (train és carriage) pályán történő mozgatását, úgymond az ütemet biztosítja a rendszerben.
  - **void gameOver():**

3.3.11 Train

- **Felelősség**

A vonatok modellezését megvalósító osztály.

- **Ősosztályok**

MovingElement

- **Interfészek**

Nincs.

- **Attribútumok**
  - **ArrayList<Carriage> carts:** a vonathoz csatlakoztatott kocsik vannak benne

eltárolva.

- **Metódusok**
  - **void arrive()**: a vonat állomásra történő megérkezését végzi. Azon követelmények és szabályok beteljesülését vizsgálja amelyeket a funkcionális követelmények a vonatra és állomásra szabnak.
  - **void vanish()**: Ha egy vonat kiürül, akkor eltűnik. Ezt valósítja meg. Megszünteti a jelenlegi vonat objektumot.

3.3.12 TunnelEntrance

- **Felelősség**

Az alagút ki és bejáratát modellezi. Hasonló a switchez, egy alagútnak két bejárata van.

- **Ősosztályok**

MapElement.

- **Interfészek**

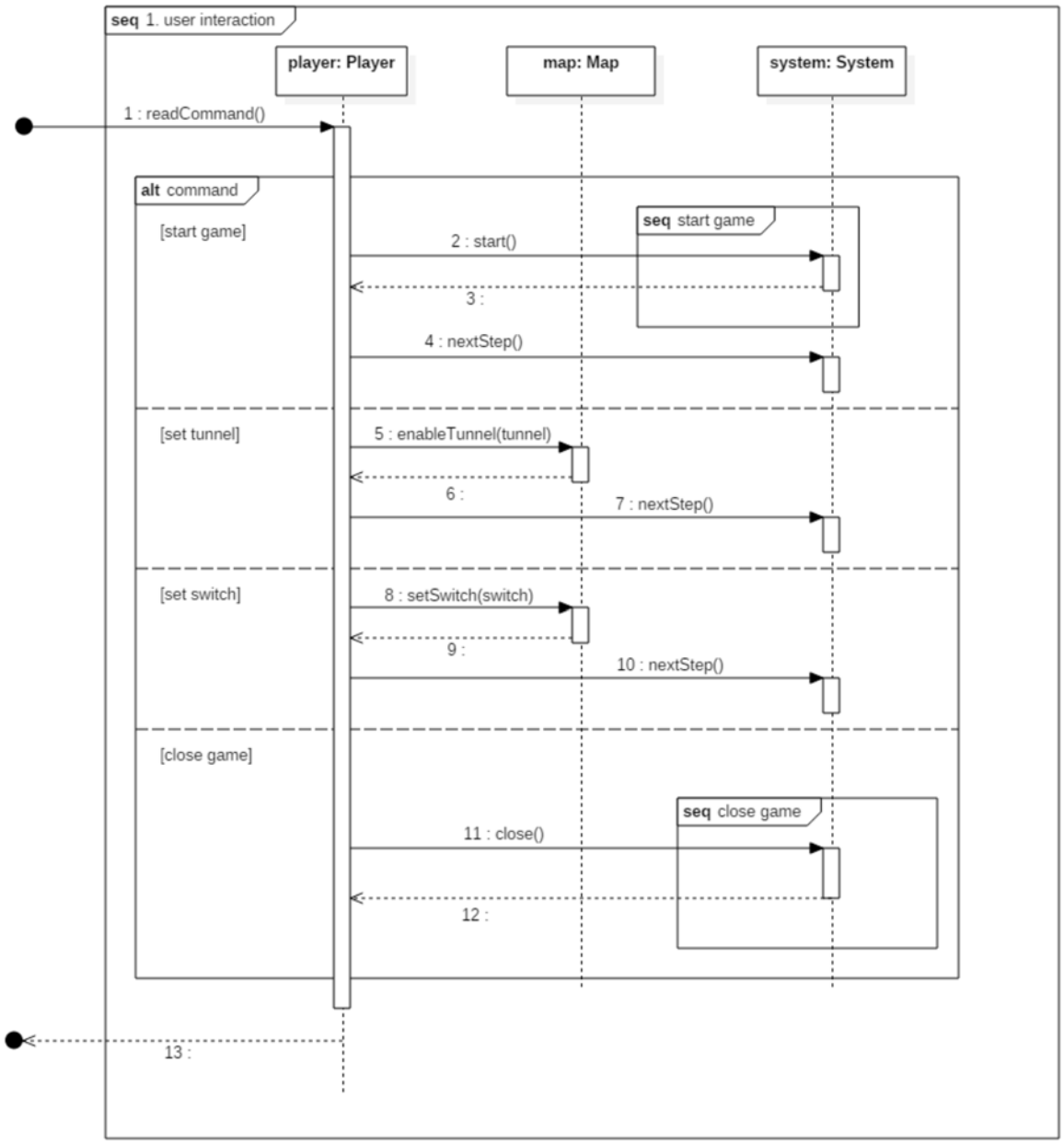
Nincs.

- **Attribútumok**
  - **ArrayList<MapElement> exitPoints:** ki- és bejárata az alagútnak.

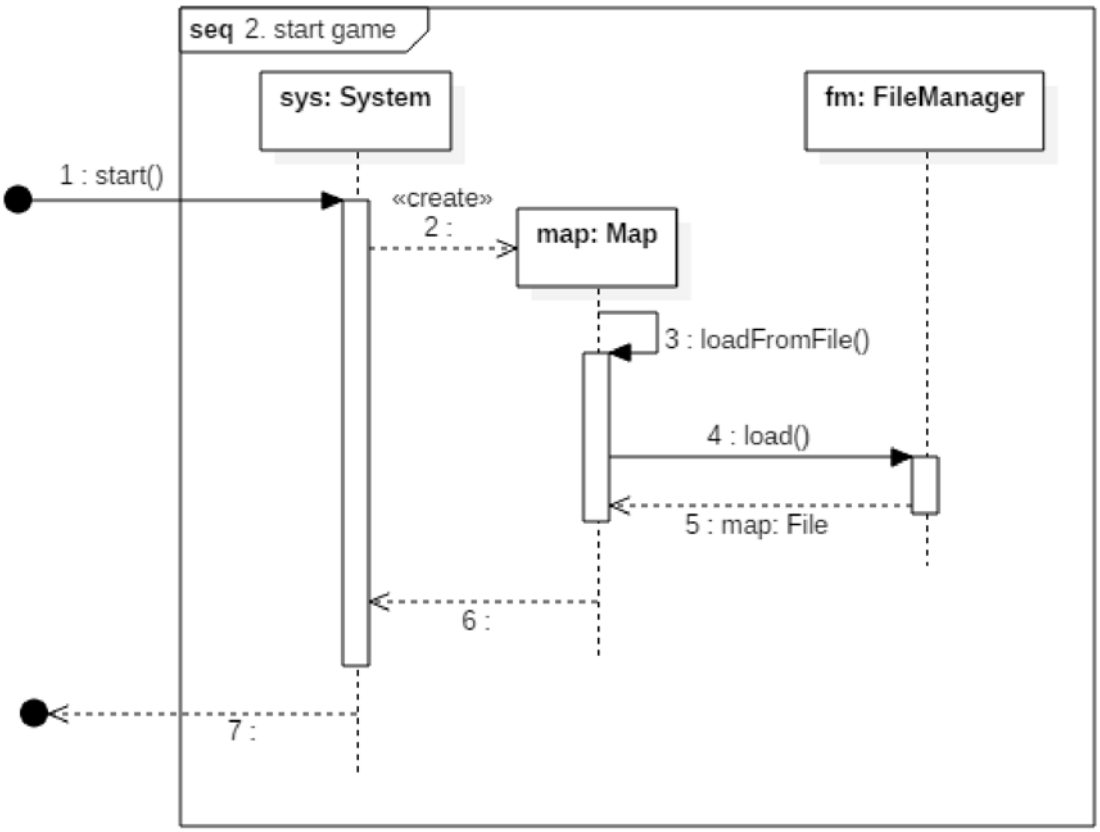
- **Metódusok**: nincsenek

3.4 Szekvencia diagramok

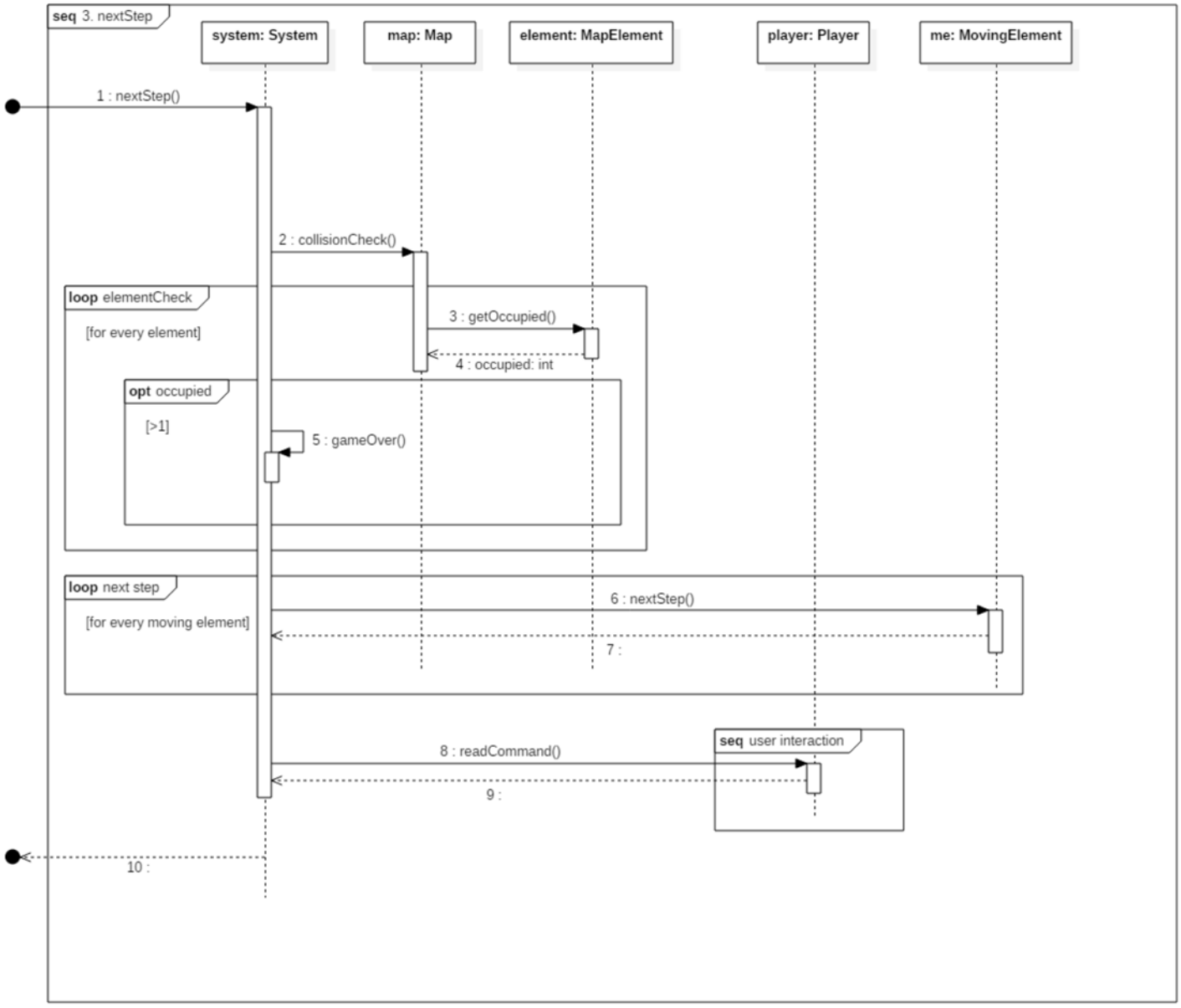
3.4.1 User interaction



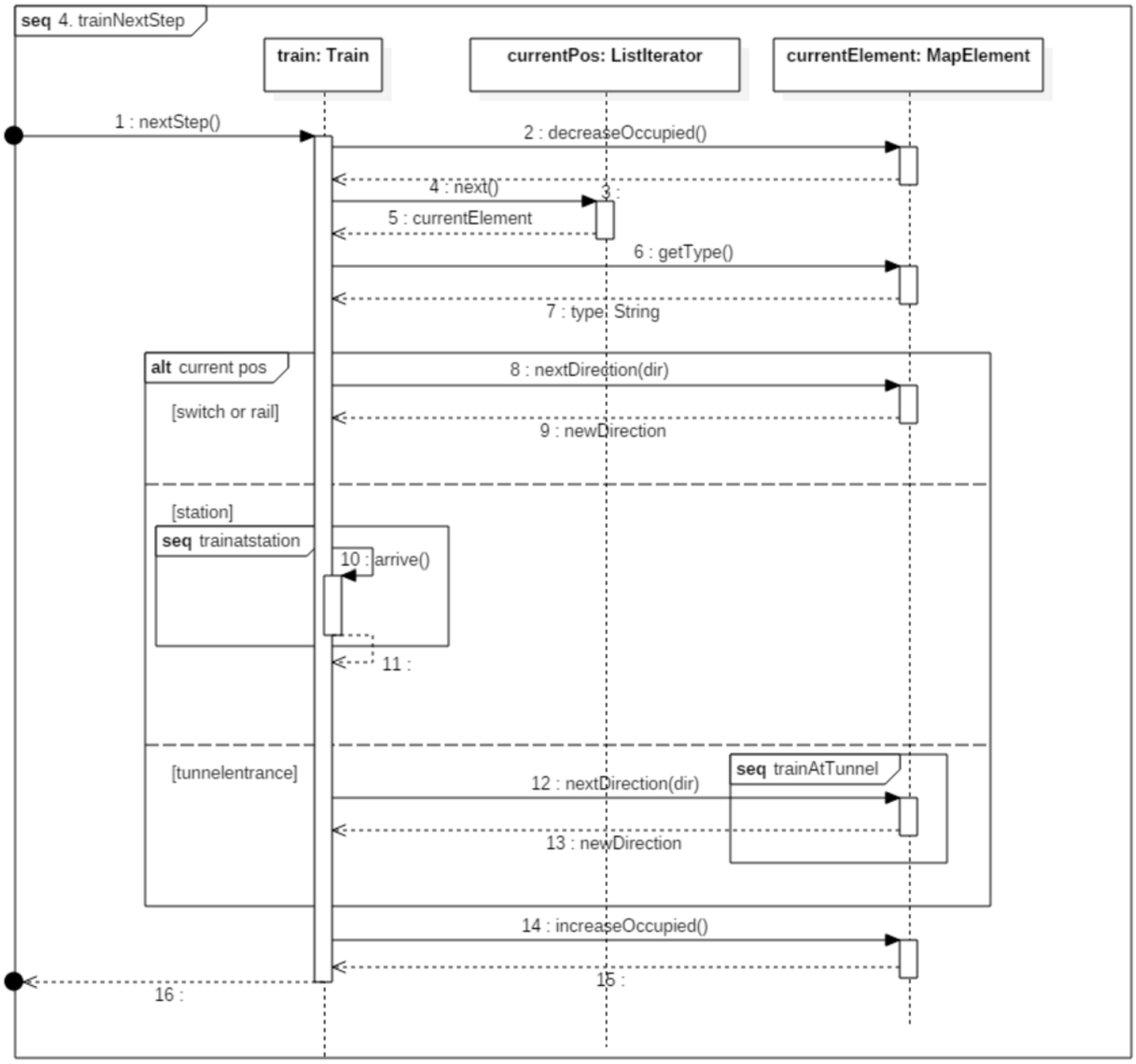
3.4.2 Start game



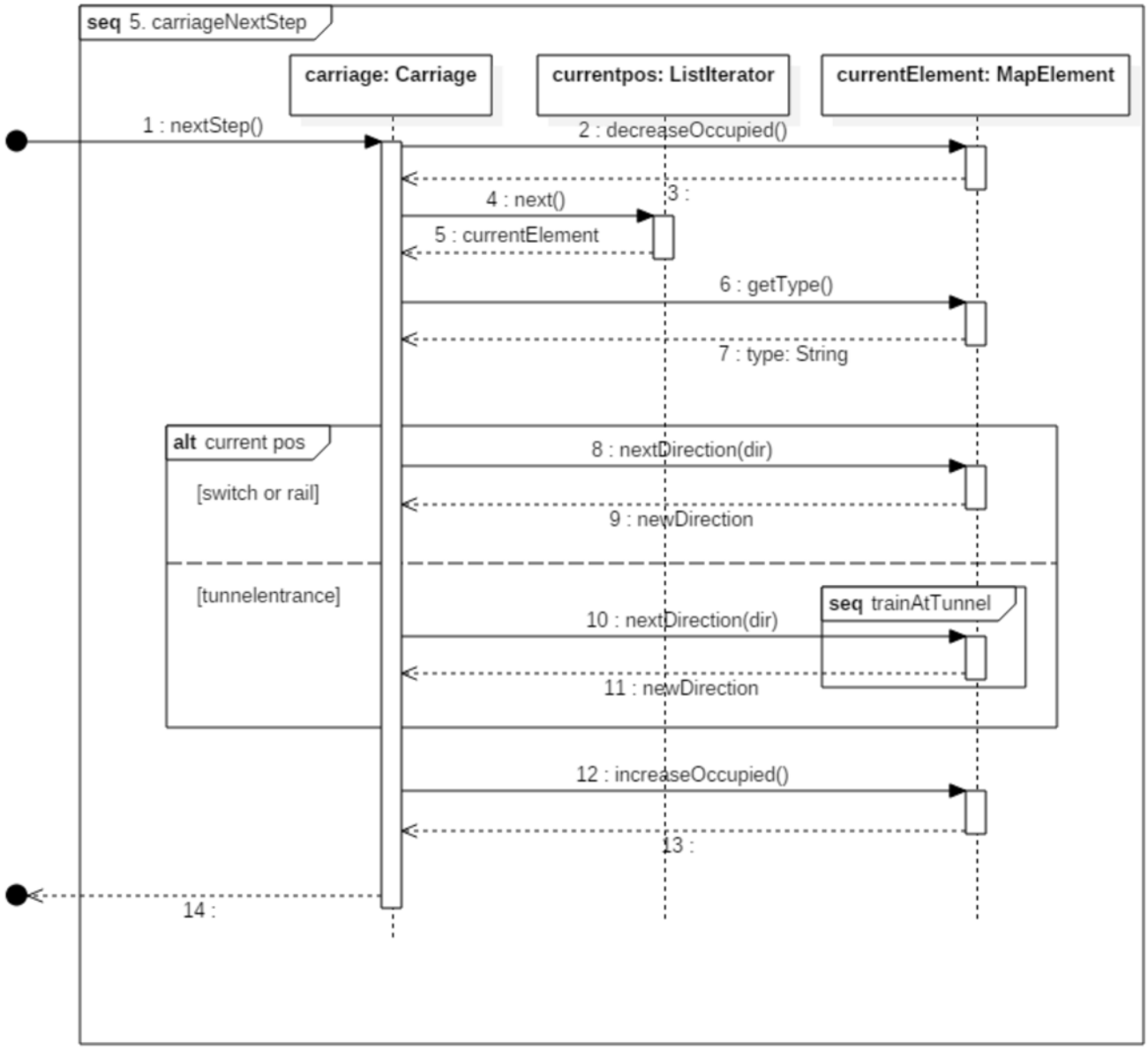
3.4.3 NextStep



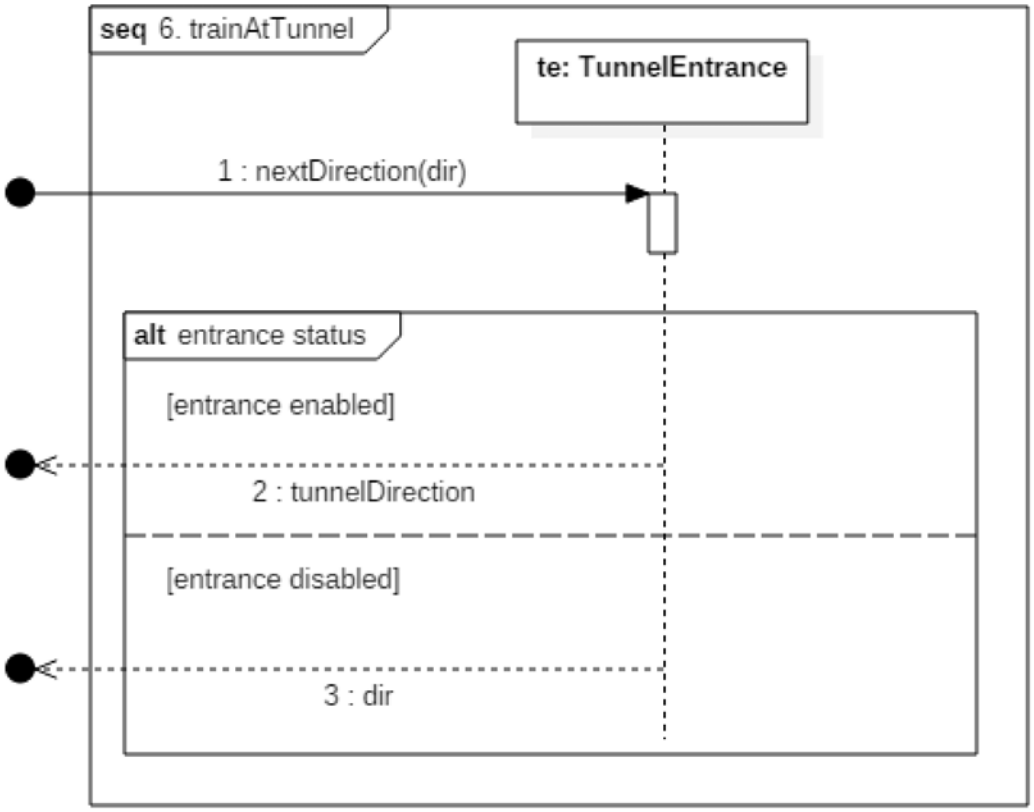
3.4.4 TrainNextStep



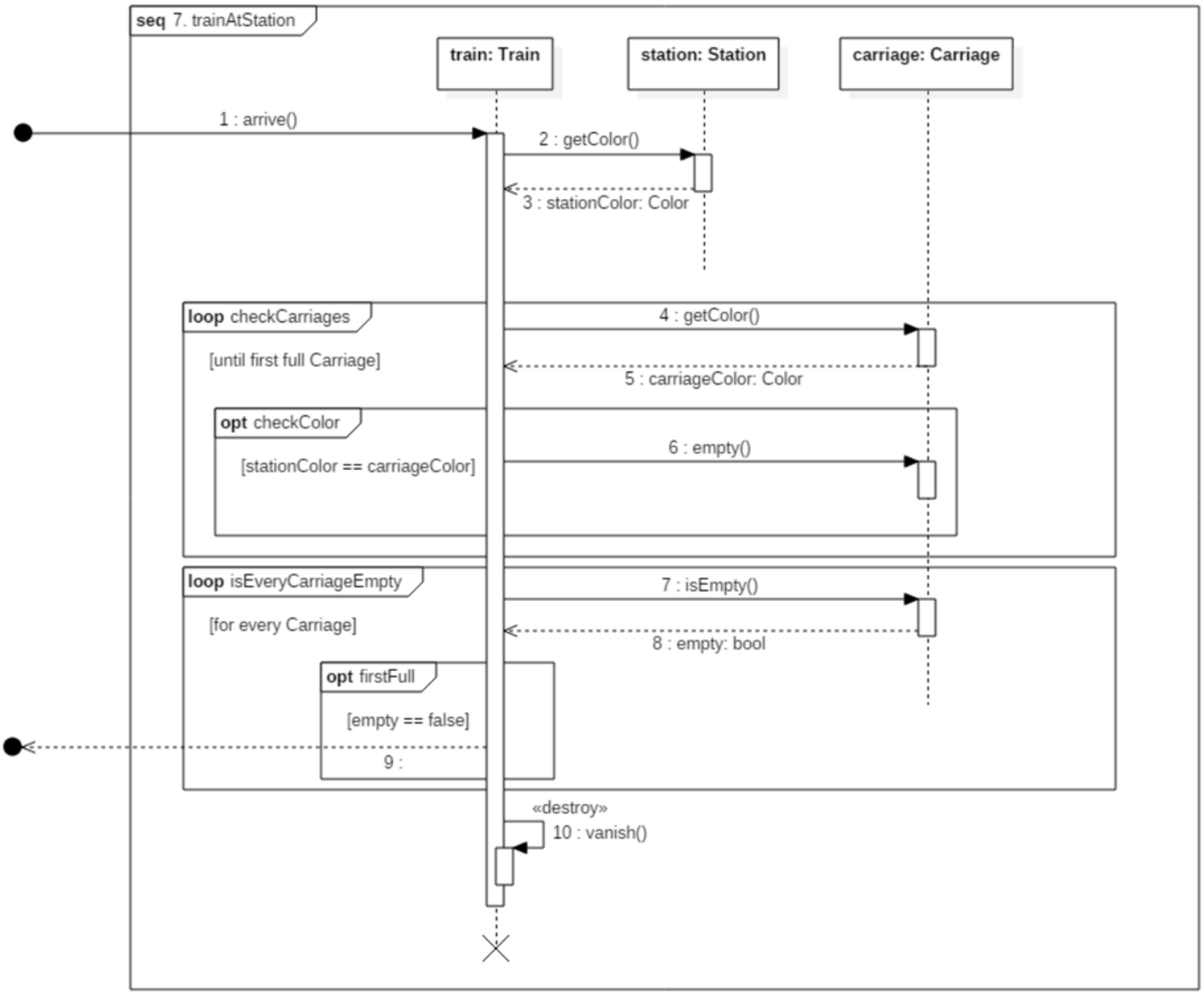
3.4.5 CarriageNextStep



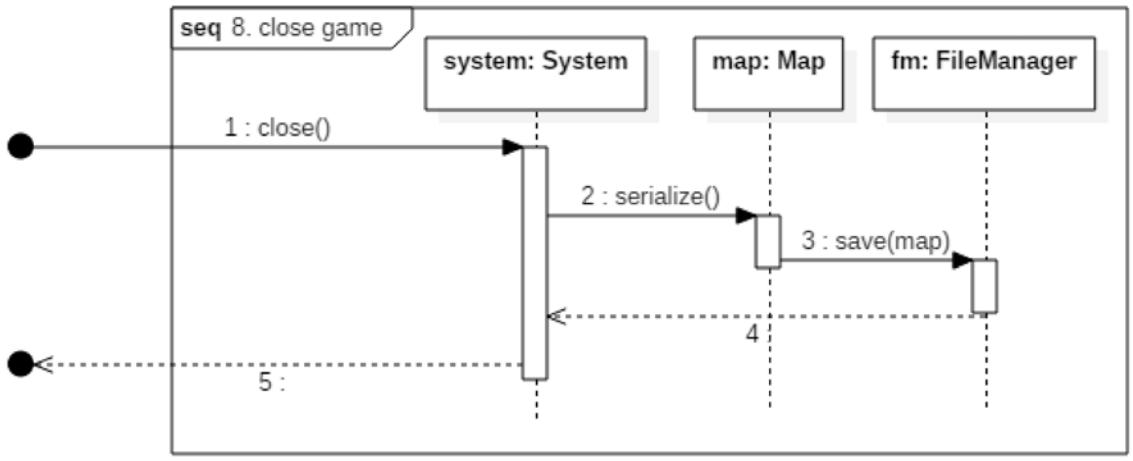
3.4.6 TrainAtTunnel



3.4.7 TrainAtStation

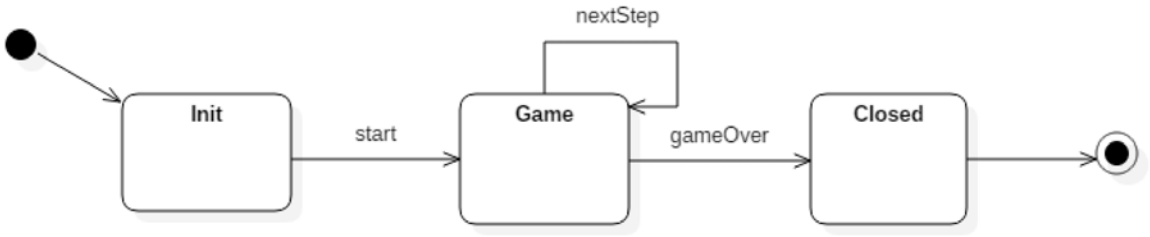


3.4.8 Close game

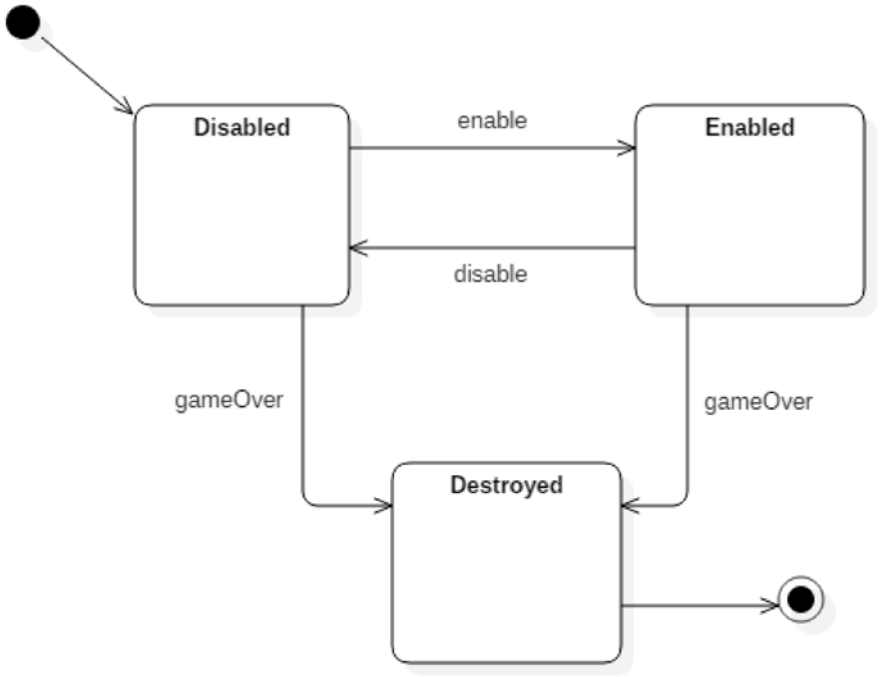


3.5 State-chartok

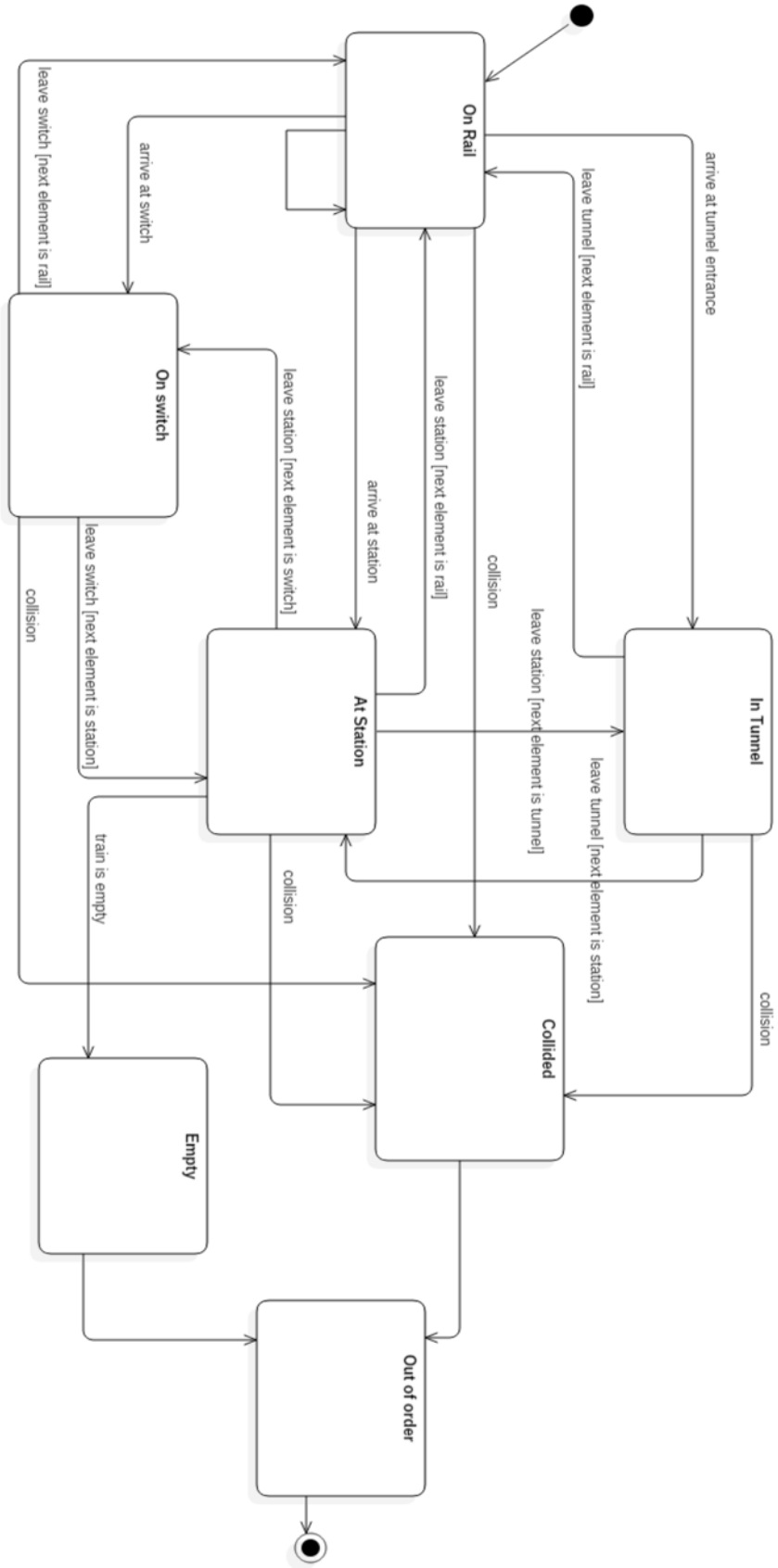
3.5.1 System állapotdiagramja



3.5.2 Tunnel állapotdiagrajma



3.5.3 Train állapotdiagramja



3.6 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017. 02. 20. 19:00	2 óra	Salamon Papp Fenes Vizi Dobó	Értekezlet. Kezdeti osztálydiagram és alapötletek megbeszélése, feladatok kiosztása: ● Salamon, Papp, Vizi szekvenciadiagramokat készíti el ● Dobó az osztálydiagramot ● Vizi az objektumkatalógust ● Fenes és Dobó az állapotgépeket.
2017. 02. 21. 19:00	1 óra	Salamon Papp Dobó Vizi Fenes	Értekezlet. Az előző értekezleten kiosztott feladatokban felmerülő kérdések megbeszélése.
2017. 02. 22. 18:00	2 óra	Fenes Salamon Dobó Vizi Papp	Értekezlet. Laborvezetőnk által felvetett ötletek megvalósításának részletes kidolgozása. Itt döntöttük el, hogy a tárolás duplán láncolt lista jellegű lesz, illetve átbeszéltük a tipikus OO hibákat, melyek a konzultáción elhangzottak. Valamint állandó értekezlet időpontok bevezetésének megtárgyalása, melyek a jövőhéttől esedékesek.
2017. 02. 22. 20:30	2 óra	Papp	Szekvenciadiagramok egy részének elkezdése, ötletelés a működésen. (start game szekvenciadiagram elkészítése, illetve továbbfejlesztésen való elmélkedés.) Ötletek elküldése Salamonnak.
2017. 02. 22 21:00	1,5 óra	Salamon	Szekvenciadiagram elkészítése: close game A további szekvencidiagramok struktúrájának kigondolása, azok Papp ötleteivel való összevetése.
2017. 02. 22. 21:00	1,5 óra	Fenes	A csapat által eddig elkészített anyagok github repositoryba történő szervezése.
2017. 02. 23. 10:30	2,5 óra	Papp	user interaction, next step szekvenciadiagramok befejezése, finomítása, esetleges javítások kigondolása. Salamon által eddig elkészített diagramok megtekintése és javaslatok tétele.
2017. 02. 23. 14:00	2 óra	Salamon	Szekvenciadiagramok készítésének folytatása a korábban egyeztetett ötletek alapján: train at station, moving element at tunnel
2017.02.24.	3,5 óra	Dobó	Statikus struktúra diagram elkezdése,

15:00			state-chartok kigondolása
2017. 02. 24. 18:00	2 óra	Salamon	Szekvenciadiagramok újragondolása, néhány változtatás eszközölése. Osztályok leírásának elkezdése.
2017. 02. 25 10:00	3,5 óra	Dobó	State-chart diagramok készítése (Tunnel, System), statikus struktúra diagram finomítása
2017. 02. 25. 11:00	4 óra	Vizi	Eddig meglévő szekvenciadiagramok ellenőrzése. Szekvenciadiagramok készítése: trainNextStep, carriageNextStep
2017.02.25. 12:00	4 óra	Fenes	Szekvenciadiagramok helyességének ellenőrzése, hiányzó szekvenciadiagram elkészítése, többi szekvenciadiagram elrendezésének, kinézetének finomítása. A szekvenciadiagramok dokumentumba illesztése.
2017. 02. 25. 14:15	2,5 óra	Papp	Hiányzó osztályleírások kiegészítése, dokumentum formázásának igazítása.
2017. 02. 25. 16:00	2 óra	Fenes	Train állapotdiagramjának elkészítése, carriageNextStep és trainNextStep currentElement kérdésének megoldása.
2017. 02. 25. 18:00	2 óra	Vizi	Objektum katalógus elkészítése.
2017.02.26 10:30	1 óra	Dobó	Kisebb hibák javítása a statikus struktúra és szekvenciadiagramokon
2017. 02. 26. 11:00	1,5 óra	Salamon	Az occupied kérdés megoldása a szekvenciadigramokon, ennek megfelelően az osztálydiagram javítása.
2017. 02. 26. 12:00	0,5 óra	Fenes	Github repo használata közbeni hiba kijavítása.
2017. 02. 26. 16:00	1 óra	Salamon	Dokumentum formázásának javítása.
2017. 02. 26 17:00	2 óra	Vizi	A dokumentáció átolvasása, elírások, helyesírási hibák javítása, nem egyértelmű megfogalmazások átalakítása, következetesség felülvizsgálata.
2017. 02. 26. 20:00	1 óra	Papp	Formázás átnézése, nyomtatás.