1. **Analízis modell kidolgozása**
   1. ***Objektum katalógus***
      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.

* + 1. **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.

* + 1. **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).

* + 1. **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.

* + 1. **Rail**

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

* + 1. **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.).

* + 1. **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.

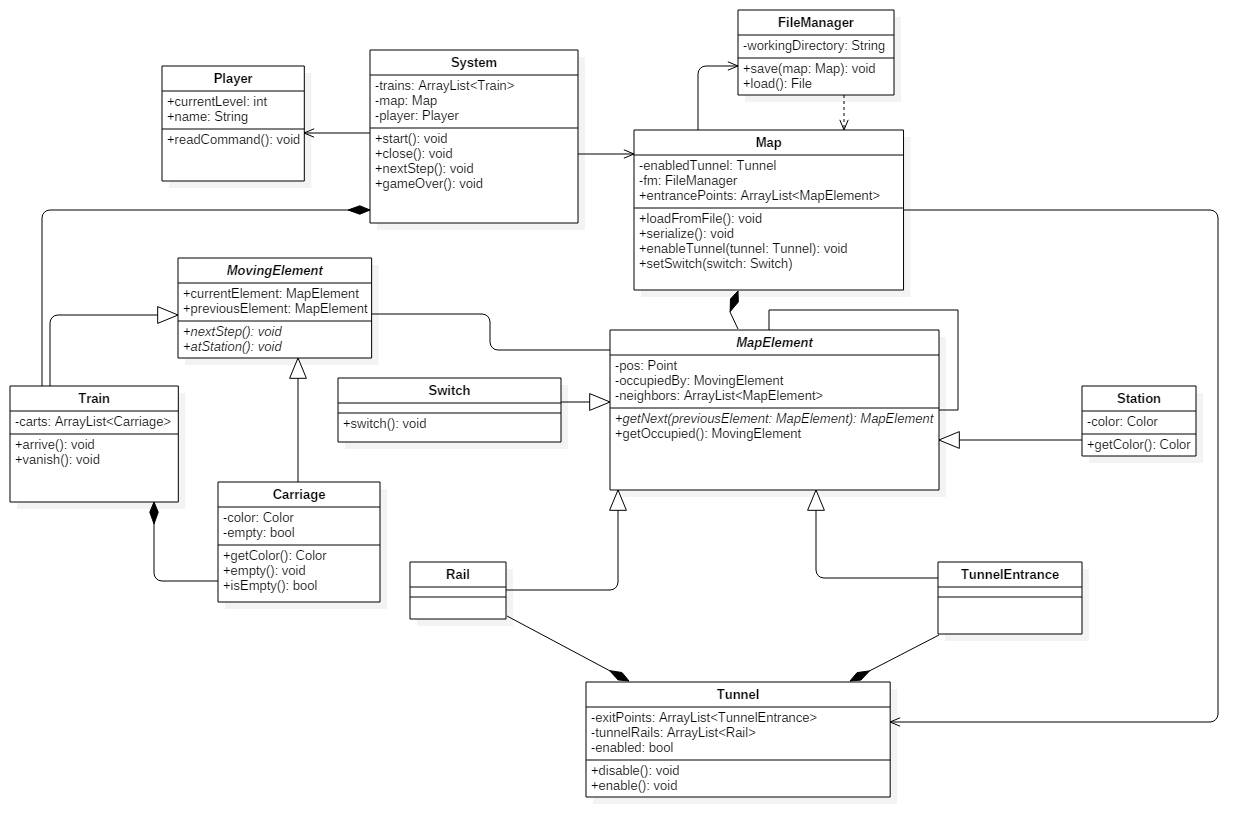
* + 1. **MapElement**

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

* + 1. **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.

* 1. ***Statikus struktúra diagramok***



* 1. ***Osztályok leírása***
     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.
  + 1. **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.
  + 1. **Map**
       - **Felelősség**

A pályaelemek tárolásáért illetve az ezekkel kapcsolatos műveletek elvégzéséért felelős.

* + - * **Ősosztályok**

Nincs.

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**
* **FileManager fm:** a FileManager osztály példánya.
* **ArrayList<MapElement> referencePoints:** lista, ami MapElement típusú elemeket tartalmaz.
* **Tunnel 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 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.
  + 1. **MapElement**
       - **Felelősség**

A statikus pálya elemek absztrakt osztálya. Egy Map tartalmazza az összeset. Lehet Switch, Rail, Station, TunnelEntrance, Tunnel. Tudja, hogy hogyan követik egymást a MapElementek, a láncolás miatt és meghatározza a haladási irányt is.

* + - * **Ősosztályok**

Nincs.

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**
* **Point pos:** az adott MapElement pozícióját tárolja koordinátákkal.
* **MovingElement occupiedBy:** milyen MovingElement van az adott MapElementen
* **ArrayList<MapElement> neighbors:** MapElementeket tárol, amik az aktuális elemnek a szomszédai.
  + - * **Metódusok**
* **getNext(previousElement: MapElement): MapElement:** visszaadja a következő pályaelemet, az aktuális illetve az előző elem alapján.
* getOccupied(): MovingElement: visszaadja a jelenleg rajta tartózkodó MovingElement-et.
  + 1. **MovingElement**
       - **Felelősség**

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

* + - * **Attribútumok**
* **MapElement previousElement:** az az elem, amin ezelőtt tartózkodott.
* **MapElement currentElement:** az az elem, amin jelenleg tartózkodik.
  + - * **Metódusok**
* **void nextStep()**: a léptetést kezelő metódus. Kiszámoltatja a következő pozíciót, a következő irányt.
* **void atStation():** leszármazott osztályok függvényeit hívja meg, ez végzi a kocsik színének lekérdezését, ürítését, valamint a vonat törlését ha szükséges.
  + 1. **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**.**
  + 1. **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
    1. **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.
  + 1. **Switch**
       - **Felelősség**

A feladatkiírásból származó váltó modellezését megvalósító osztály.

* + - * **Ősosztályok**

MapElement.

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**

Nincs.

* + - * **Metódusok**
* **void switch()**: következő állásba történő állítás. A játékos végzi, kattintással, így az iránya egy megadott szabály szerint fog változni.
  + 1. **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():**
  + 1. **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.
  + 1. **Tunnel**
       - **Felelősség**

A tartalmazott listák segítségével segíti a vonat haladását az alagútban.

* + - * **Ősosztályok**

Nincs.

* + - * **Interfészek**

Nincs.

* + - * **Attribútumok**
* **ArrayList<TunnelEntrance> exitPoints:** Alagút bejáratokat tartalmazó lista.
* **ArrayList<Rail> tunnelRail:** Lista a sínekről amik az alagútban vannak.
* **bool enable:** Eltárolja, hogy engedélyezve van-e a tunnel vagy sem.
  + - * **Metódusok**
* **void enable()**: Engedélyezi a vonat áthaladását az alagúton.
* **void disable()**: Letiltja az alagút használatát.
  + 1. **TunnelEntrance**
       - **Felelősség**

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

* + - * **Ősosztályok**

MapElement.

* + - * **Interfészek**

Nincs.

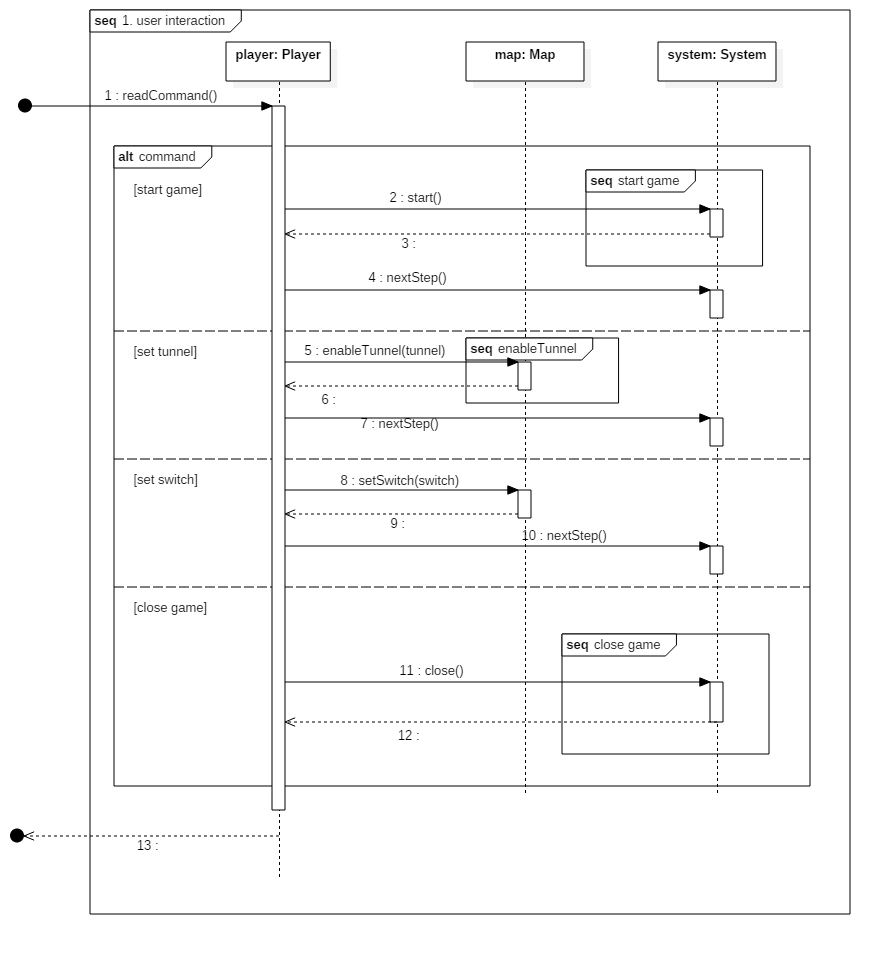
* + - * **Attribútumok**

Nincs.

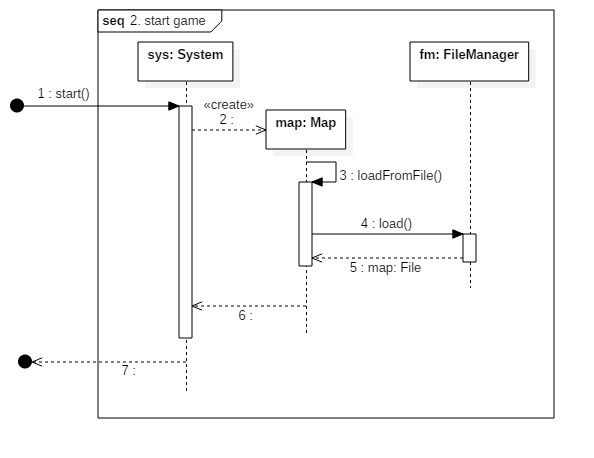
* + - * **Metódusok:**

Nincs.

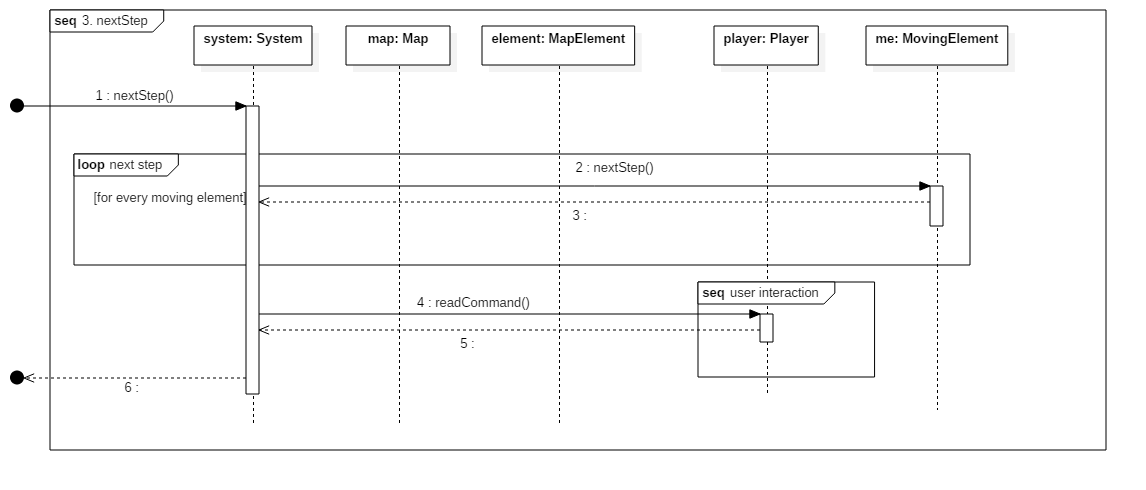
* 1. ***Szekvencia diagramok***
     1. **User interaction**



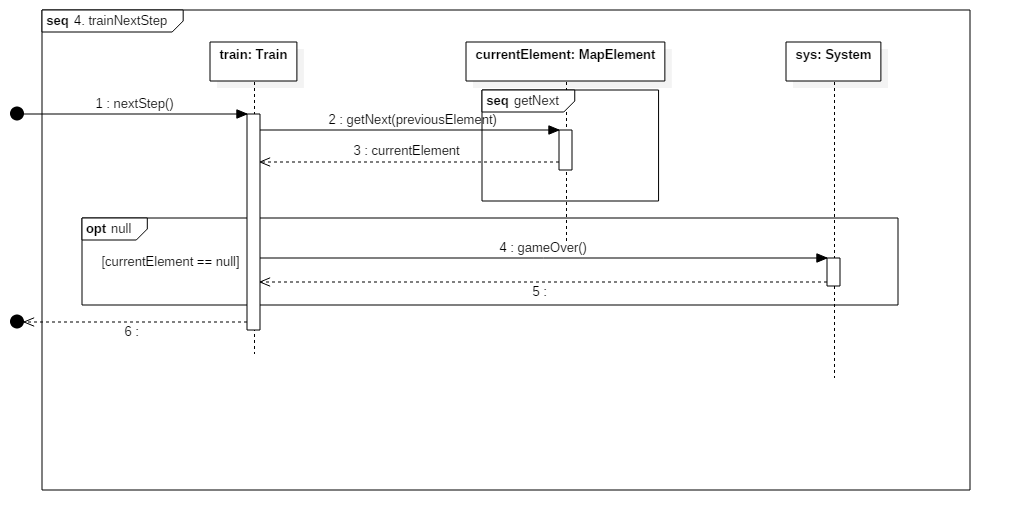
* + 1. **Start game**



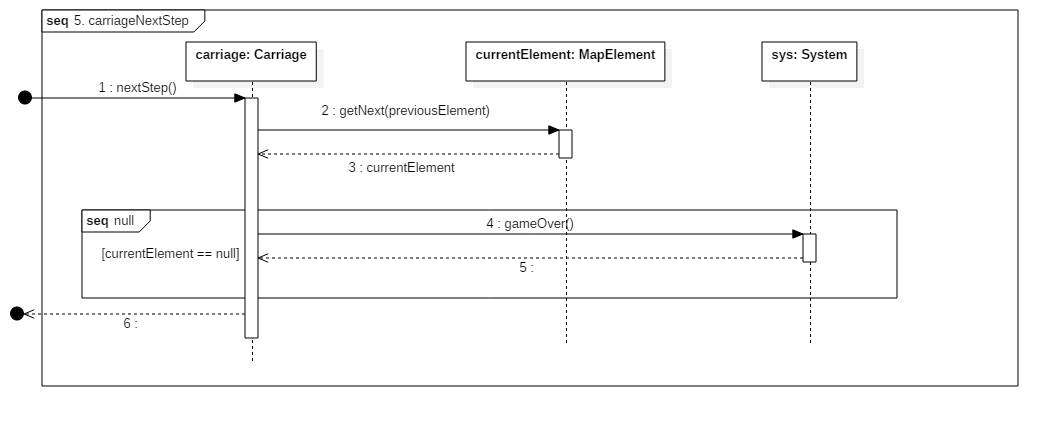
* + 1. **NextStep**



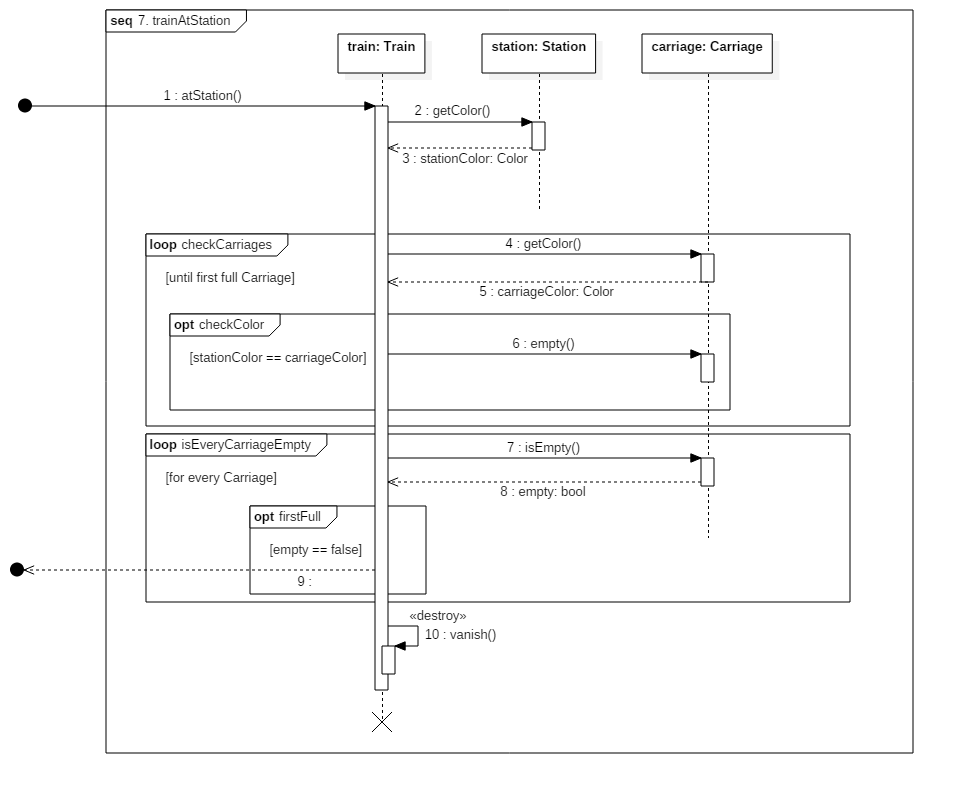
* + 1. **TrainNextStep**



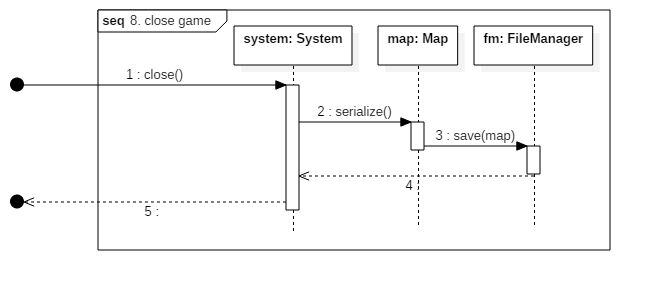
* + 1. **CarriageNextStep**



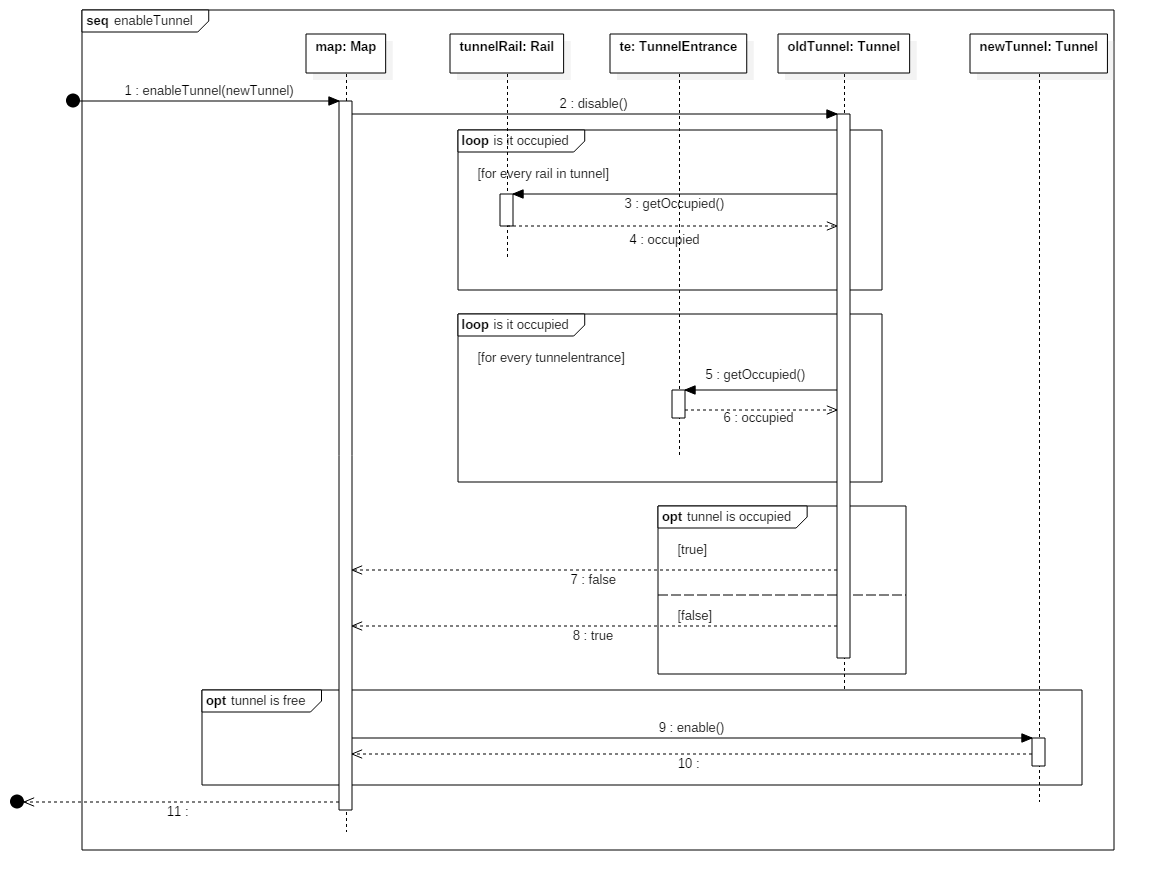
* + 1. **TrainAtStation**



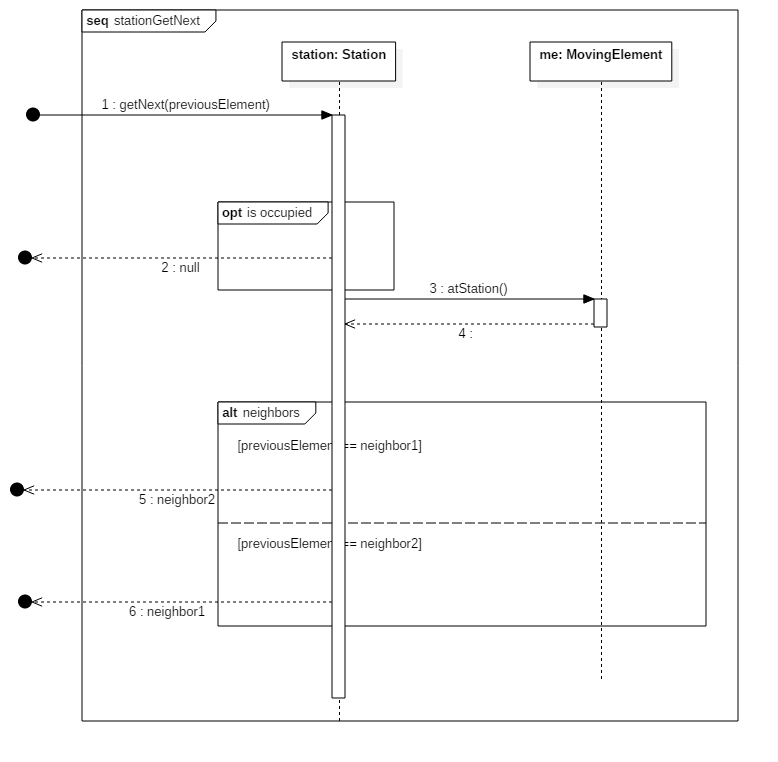
* + 1. **Close game**



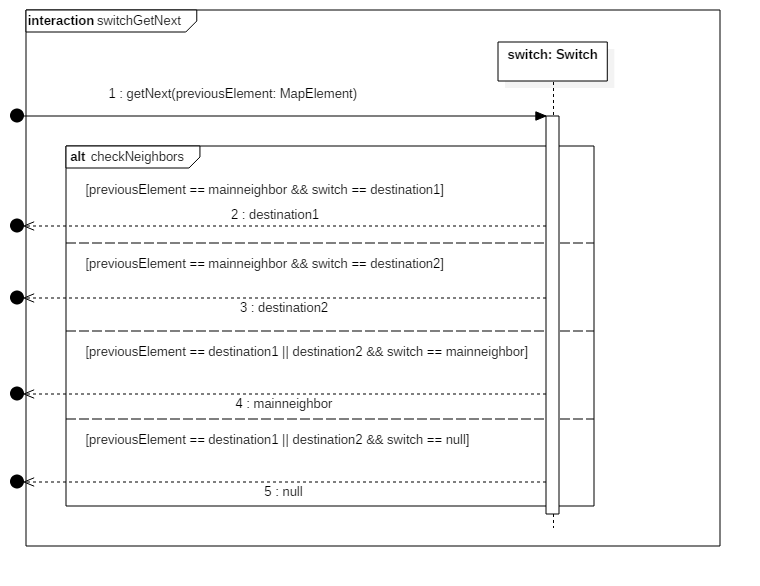
* + 1. **enableTunnel**



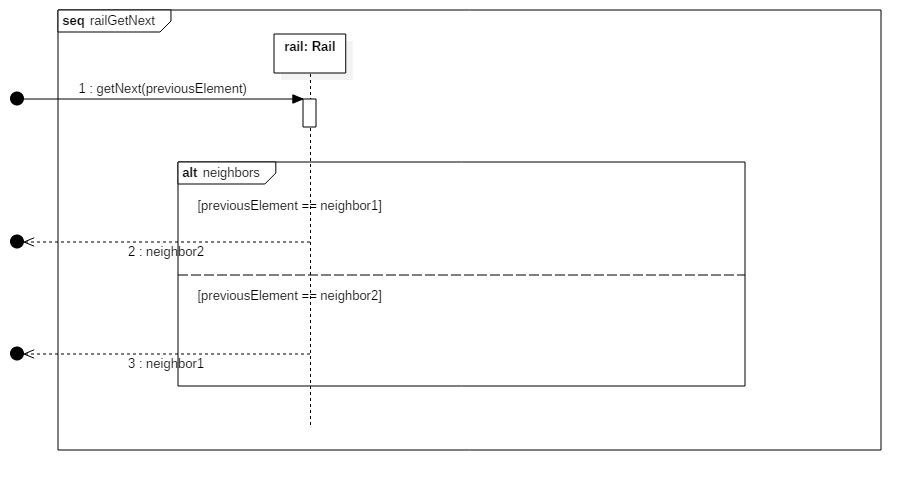
* + 1. **stationGetNext**



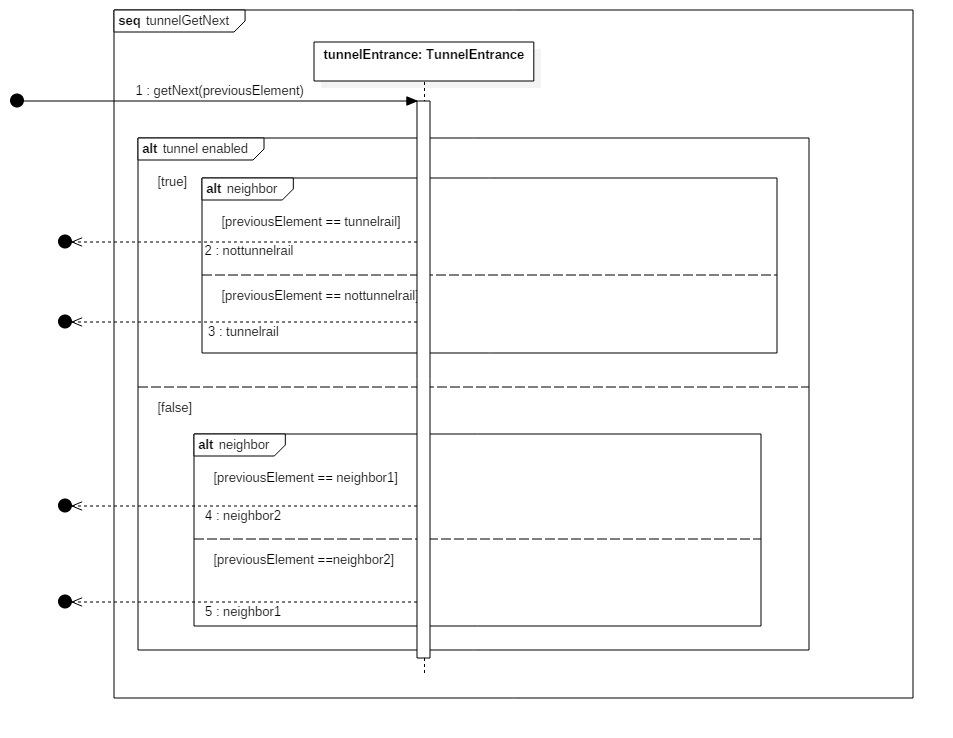
* + 1. **switchGetNext**



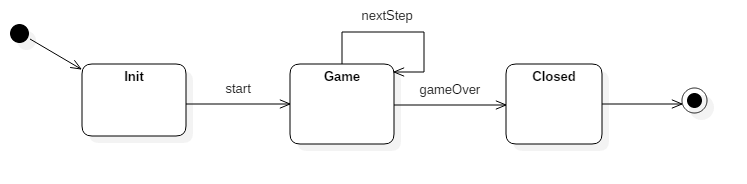
* + 1. **railGetNext**



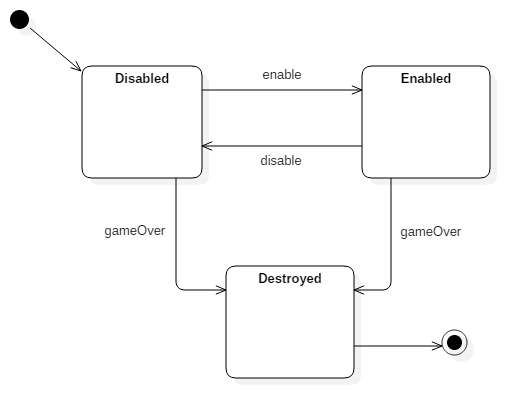
* + 1. **tunnelGetNext**



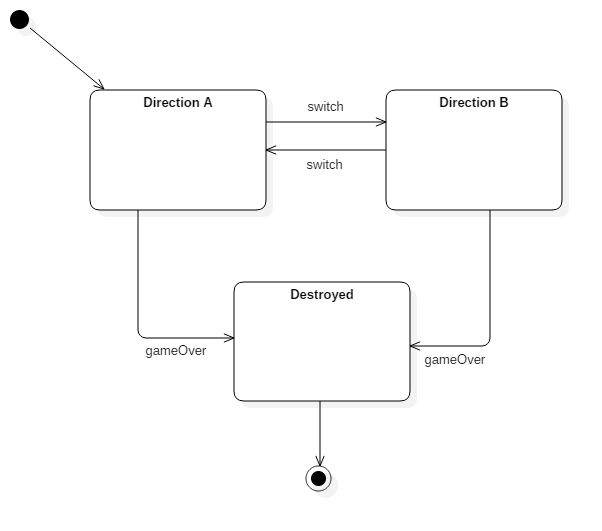
* 1. ***State-chartok***
     1. **System állapotdiagramja**



* + 1. **Tunnel állapotdiagrajma**



* + 1. **Switch állapotdiagramja**



* 1. ***Napló***

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2017.03.01. 19:00 | 3 óra | Dobó  Fenes  Papp  Salamon Vizi | Gyűlés. A laborvezető felvetéseinek megbeszélése. A hasznos ötletek átbeszélése és beépítése az eddigi munkánkba. Feladatok kiosztása:   * Fenes: railGetNext szekvencia, trainAtStation kiigazítása * Dobó: switchGetNext szekvencia, switch állapotdiagram * Salamon: tunnelGetNext szekvencia * Vizi: trainNextStep, carriageNextStep átírása * Papp: enableTunnel szekvencia |
| 2017.03.03. 19:00 | 1 óra | Dobó  Fenes  Papp Salamon  Vizi | Feladatok megoldása során felmerülő problémák megbeszélése. |
| 2017.03.03. 20:00 | 1 óra | Salamon | tunnelGetNext szekvencia |
| 2017.03.03. 20:15 | 2 óra | Papp | enableTunnel szekvencia, dokumentáció formai követelményeinek ellenőrzése |
| 2017.03.04. 13:00 | 2 óra | Vizi | carriegeNextStep, trainNextStep szekvenciák módosítása a megbeszéltek alapján |
| 2017.03.04. 17:20 | 2 óra | Dobó | switch állapotdiagram elkészítése valamint a switch getNext szekvencia átszerkesztése |
| 2017.03.05. 10:00 | 2 óra | Fenes | Osztályok leírásának korrigálása |
| 2017.03.05. 16:00 | 1 óra | Salamon | Dokumentáció formázásának rendbeszedése, kinyomtatása |