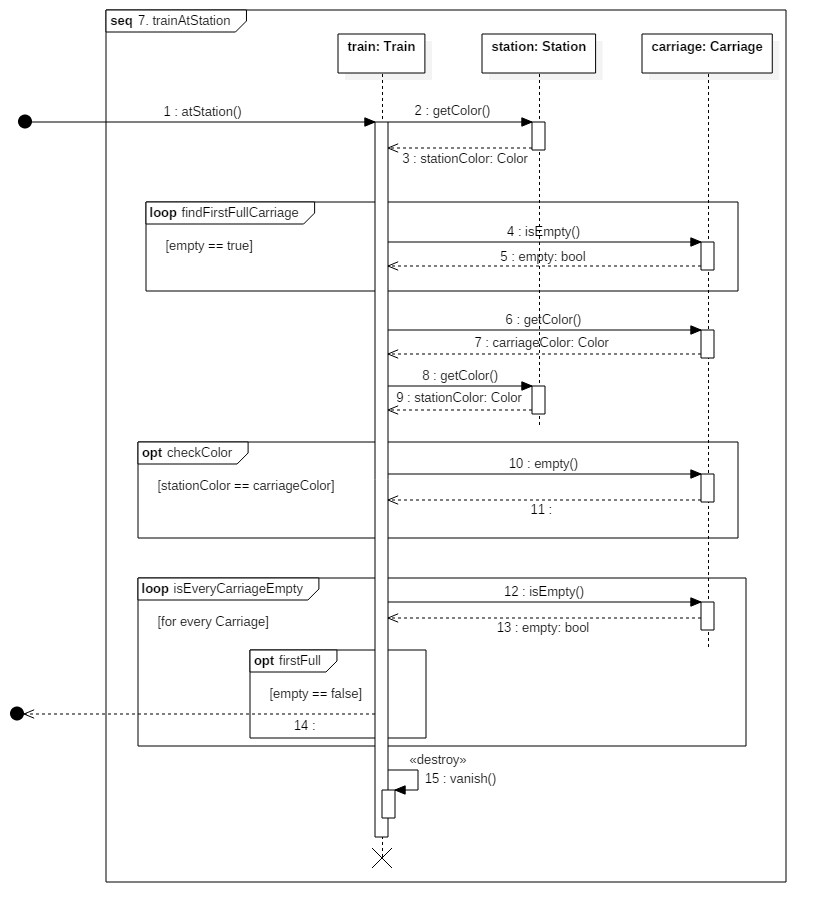
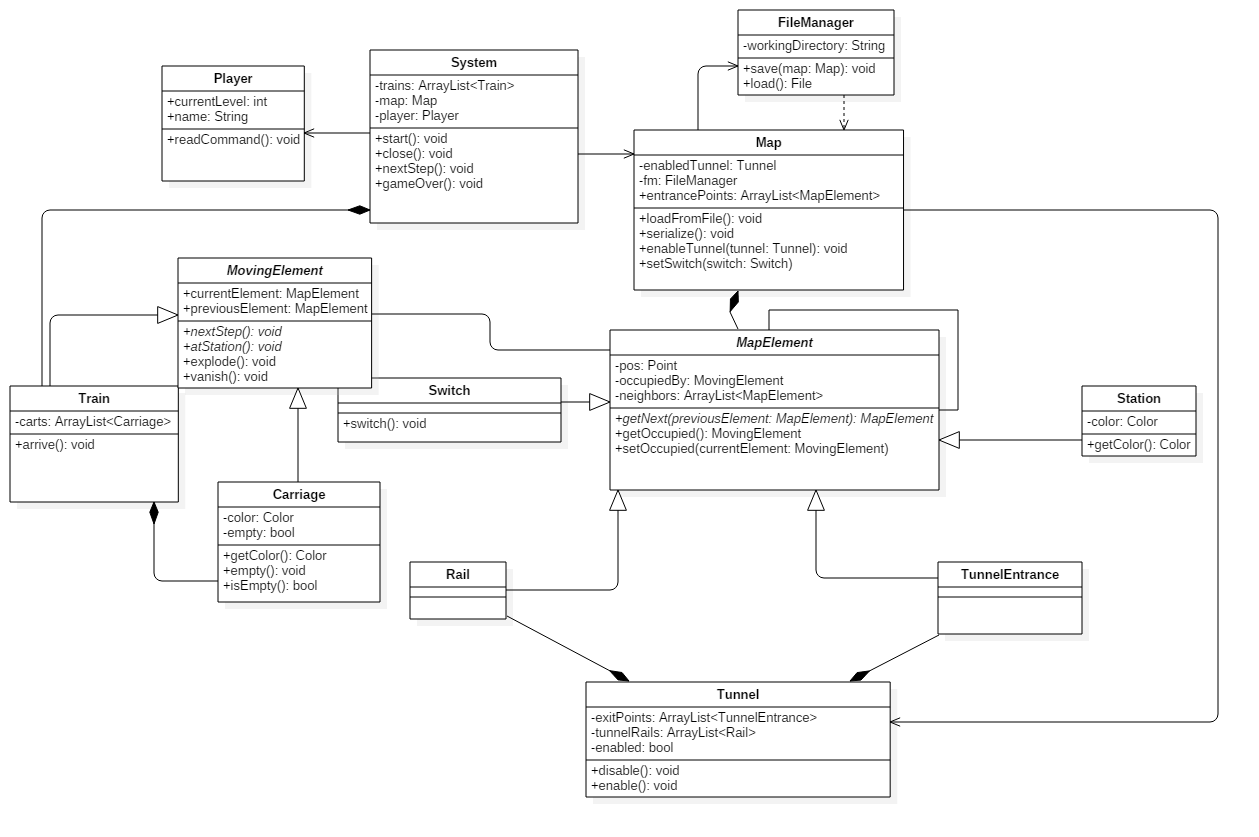
1. **Eddig eszközölt, de nem dokumentált változtatások**
   1. ***TrainAtStation FindFirstFullCarriage alpontjának kijavítása***

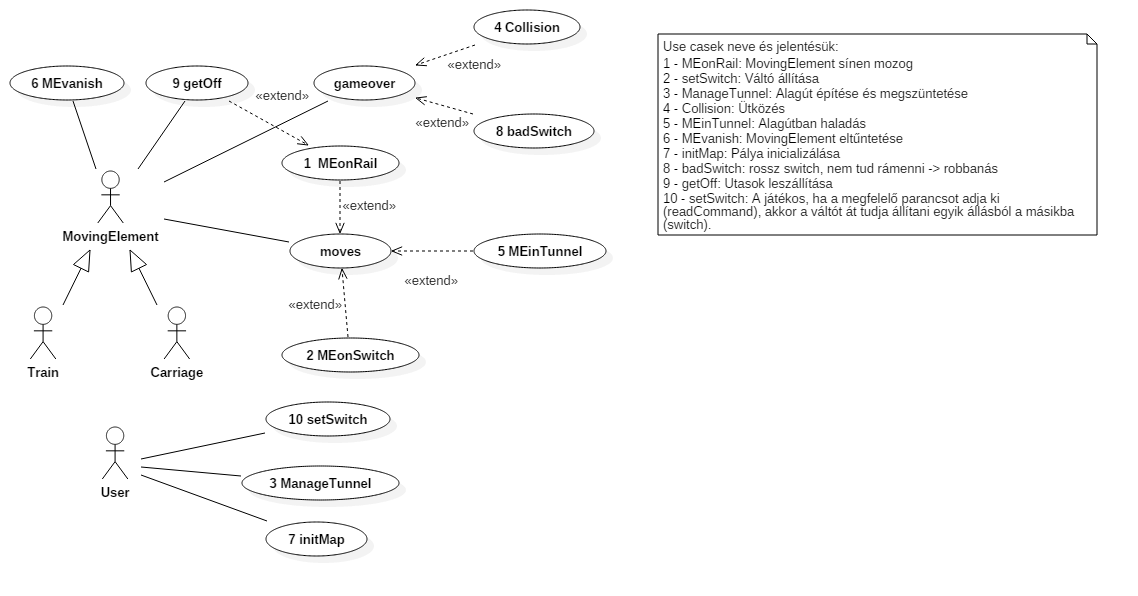
Kijavítottuk az őrfeltételt a fent említett ciklusnál egy amolyan hátultesztelő módon. Addig iterál a carriagek között a train, amíg meg nem találja az első telit, azaz amíg [empty == true]. Ezt követően a logika egyezik a múlt hetivel. Tulajdonképpen az alapvető elképzelés nem változott, csak az ábrázolás módját igyekeztünk pontosítani.



* 1. ***Módosult osztálydiagram***



1. **Szkeleton tervezése**
   1. ***A szkeleton modell valóságos use-case-ei***
      1. **Use-case diagram**



* + 1. **Use-case leírások**

|  |  |
| --- | --- |
| **Use-case neve** | 1 - MEonRail |
| **Rövid leírás** | MovingElement sínen mozog |
| **Aktorok** | MovingElement |
| **Forgatókönyv** | A Train vagy Carriage lépésenként lekérdezi az éppen aktuális síntől, hogy ki a következő. Ezt a getNext függvény segítségével teszi, majd a setOccupied függvénnyel lefoglalja a következő sínt és rálép. |

|  |  |
| --- | --- |
| **Use-case neve** | 2 - MEonSwitch |
| **Rövid leírás** | Mozgás miközben váltóra ér a Train vagy Carriage |
| **Aktorok** | MovingElement |
| **Forgatókönyv** | A Train vagy Carriage egy sínről egy Switchre lép, ahol lekérdezni a következő elemet és a switch állásától függően a megfelelő irányban halad tovább. |

|  |  |
| --- | --- |
| **Use-case neve** | 3 - ManageTunnel |
| **Rövid leírás** | Alagút építése és megszüntetése |
| **Aktorok** | User |
| **Forgatókönyv** | Mikor egy új Tunnelt próbál meg a felhasználó engedélyezni, először végignézzük a TunnelEntranceokat és az alagúton belüli Rail elemeket, hogy üresek-e. Ha mindegyik üres (nem volt rajta Train vagy Carriage) a régi Tunnel bezáródik és az új Tunnel létrejön az enable() függvény segítségével. |

|  |  |
| --- | --- |
| **Use-case neve** | 4 - Collision |
| **Rövid leírás** | Ütközés |
| **Aktorok** | MovingElement |
| **Forgatókönyv** | Egy train olyan Rail-re próbál lépni, ahol már van valami. Mikor megpróbálja elfoglalni (setOccupied) akkor a Rail meghívja az explode függvényét, amit egy gameOver követ. A másik vonatot felrobbantásával nem kell foglazni, hiszen ezen a ponton már úgy is vége van a játéknak. |

|  |  |
| --- | --- |
| **Use-case neve** | 5 - MEinTunnel |
| **Rövid leírás** | Alagútban haladás |
| **Aktorok** | MovingElement |
| **Forgatókönyv** | A Train vagy Carriage először lekérdezi az éppen aktuális síntől, hogy ki a következő TunnelEntrance. Ezt a getNext függvény segítségével teszi, majd a setOccupied függvénnyel lefoglalja és rálép. Az alagúton belül ugyanezen az elven történik a lépés, csak ott Rail-t kérdez le és foglal el. |

|  |  |
| --- | --- |
| **Use-case neve** | 6 - MEvanish |
| **Rövid leírás** | MovingElement eltűntetése, ha már egy vonat összes kocsija kiürült. |
| **Aktorok** | MovingElement |
| **Forgatókönyv** | Meghívódik a vonat vanish függvénye, ami meghívja az összes hozzá tartozó kocsinak ugyan ezen nevű függvényét, amelyek beállítják azon elem foglaltságát, amin tartózkodnak. Ezt követően a vonat is megteszi ezt majd megszűnteti magát. |

|  |  |
| --- | --- |
| **Use-case neve** | 7 - initMap |
| **Rövid leírás** | Pálya inicializálása |
| **Aktorok** | User |
| **Forgatókönyv** | A játék elindításakor a Map betölti a fájlból a pályát, majd létrehozza a fájlban található elemeket. Ebben az esetben 1 darabot mindegyik MapElement-ből |

|  |  |
| --- | --- |
| **Use-case neve** | 8 - badSwitch |
| **Rövid leírás** | Olyan Switch állás, amin a vonat nem tud továbbhaladni. |
| **Aktorok** | MovingElement |
| **Forgatókönyv** | Ha a vonat Switchre lép és a tőle lekérdezett következő elem null értékű (ez egy olyan rossz Switch állás, amikor a vonat nem haladhat tovább), akkor a játéknak vége. |

|  |  |
| --- | --- |
| **Use-case neve** | 9 - getOff |
| **Rövid leírás** | Utasok leszállítása |
| **Aktorok** | MovingElement |
| **Forgatókönyv** | Ha a szerelvény egy állomáshoz ér, akkor a Train megkeresi a legelső nem üres kocsit, és annak színét összeveti az állomás színével. Ha a két szín egyezik, az utasok leszállnak a kocsiról.  Ha ezáltal minden kocsi kiürült, akkor a vonat eltűnik.  Továbblépéskor ellenőrzésre kerül az is, hogy nincs-e ütközés (azaz hogy a következő pályaelemen tartózkodik-e már valaki, amikor rálép a vonat). |

|  |  |
| --- | --- |
| **Use-case neve** | 10 - setSwitch |
| **Rövid leírás** | A váltó állítása. |
| **Aktorok** | User |
| **Forgatókönyv** | A játékos, ha a megfelelő parancsot adja ki (readCommand), akkor a váltót át tudja állítani egyik állásból a másikba (switch). |

|  |  |
| --- | --- |
| **Use-case neve** | gameover |
| **Rövid leírás** | A játéknak vége. |
| **Aktorok** | MovingElement |
| **Forgatókönyv** | Előfordulhat többféleképpen is, akár vonat ütközéssel, akár helytelen váltó állással, stb.. Esetünkben ehhez nem tartozik szekvencia, hiszen az összes ok, ami kiválthatja modellezve lett, ezáltal elérve a lefedettségét. |

|  |  |
| --- | --- |
| **Use-case neve** | moves |
| **Rövid leírás** | Egy MovingElement mozog. |
| **Aktorok** | MovingElement |
| **Forgatókönyv** | Az előbbihez hasonlóan ez is specializálva lett az egyes use-caseeken, ezáltal elérve a teljes lefedettséget. |

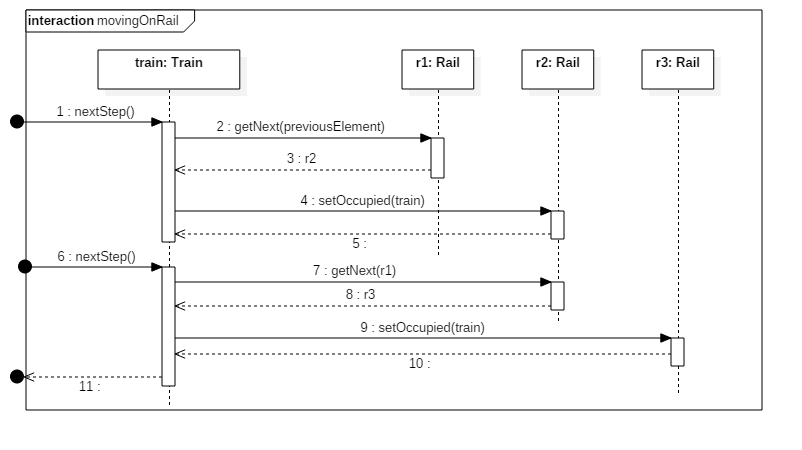
* 1. ***A szkeleton kezelői felületének terve, dialógusok***
* [0] Exit
* [1] MEonRail
* [2] MEonSwitch
* [3] ManageTunnel
* [4] Collision
* [5] MEinTunnel
* [6] MEvanish
* [7] initMap
* [8] badSwitch
* [9] getOff
* [10] setSwitch

A fentebb látható menüpontok megegyeznek a valóságos use-caseekkel.

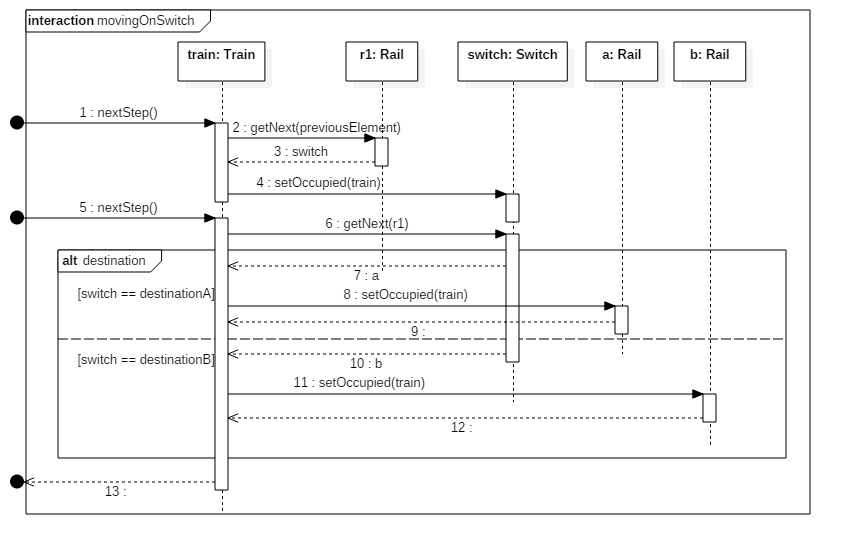
A kívánt menüpont kiválasztásához a standard inputra kell beírni a menüpontnak a sorszámát, például a Collisionhöz az 5-t majd enter lenyomásával elküldeni a programnak. Minden további input hasonlóan működik. Ha nem elfogadott inputot kap a szkeleton újból felteszi a kérdést mindaddig, amíg helyeset nem kap.

Miután kiválasztottuk a menüpontot elkezdi a végrehajtást, de még további kérdésekkel is fordulhat a felhasználó felé. Ezek a kérdések többnyire eldöntendőek és egyértelműek lesznek

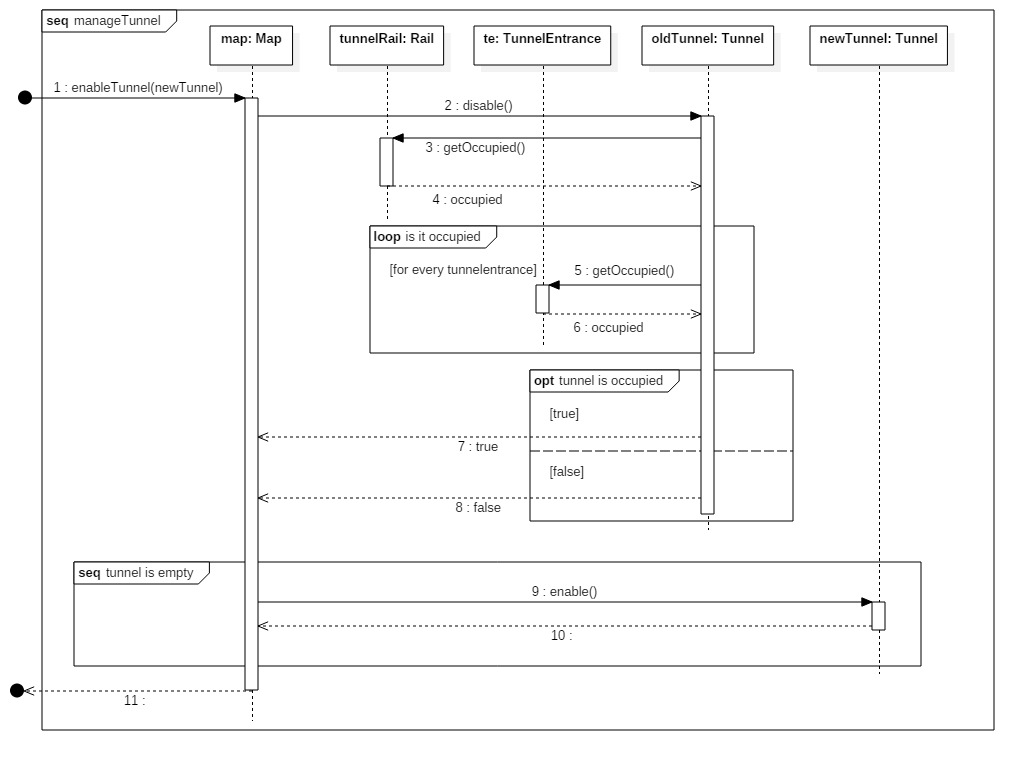
* 1. ***Szekvencia diagramok a belső működésre***
     1. ***movingOnRail (use case: 1. MEonRail)***



* + 1. ***movingOnSwitch (use case: 2. MEonSwitch)***



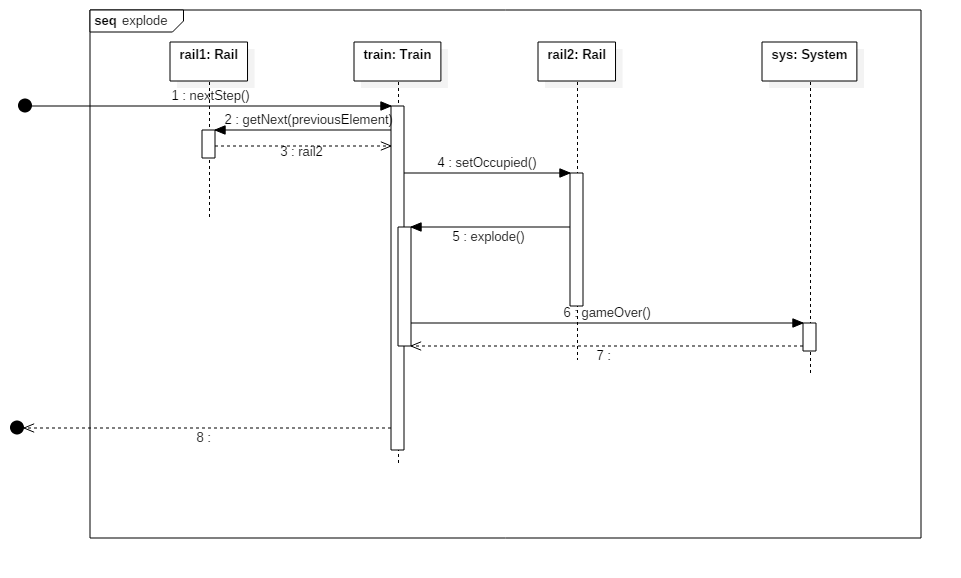
* + 1. ***manage Tunnel (use case: 3. ManageTunnel)***



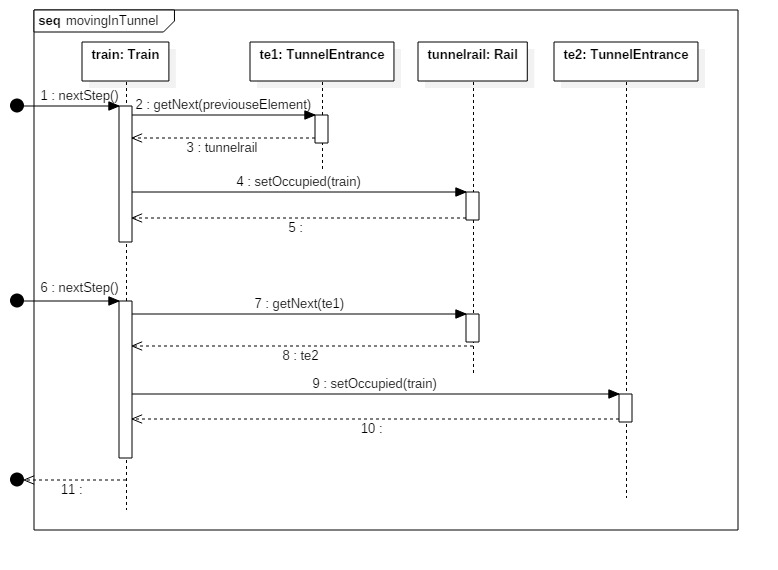
* + 1. ***getOff (use case: 9. getOff)***



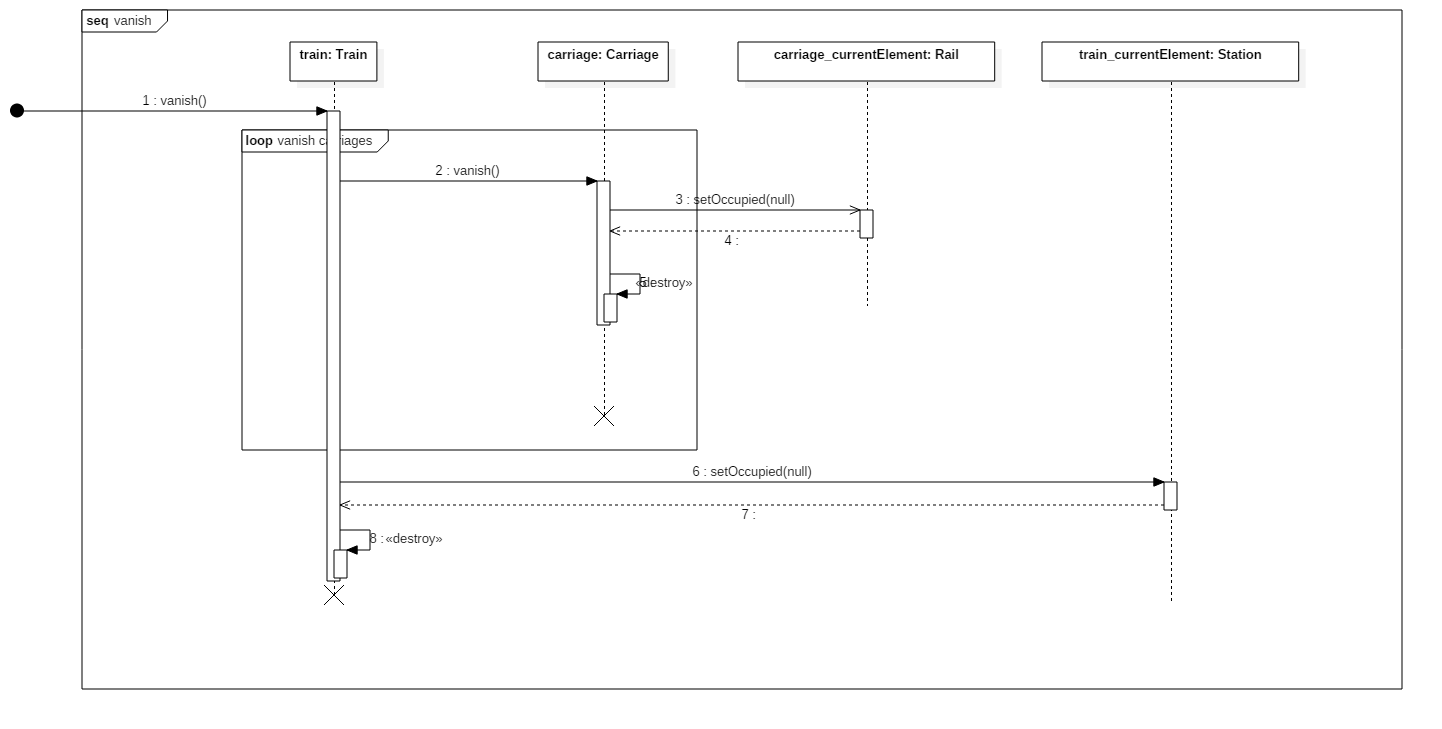
* + 1. ***explode / collision (use case: 4. Collision)***



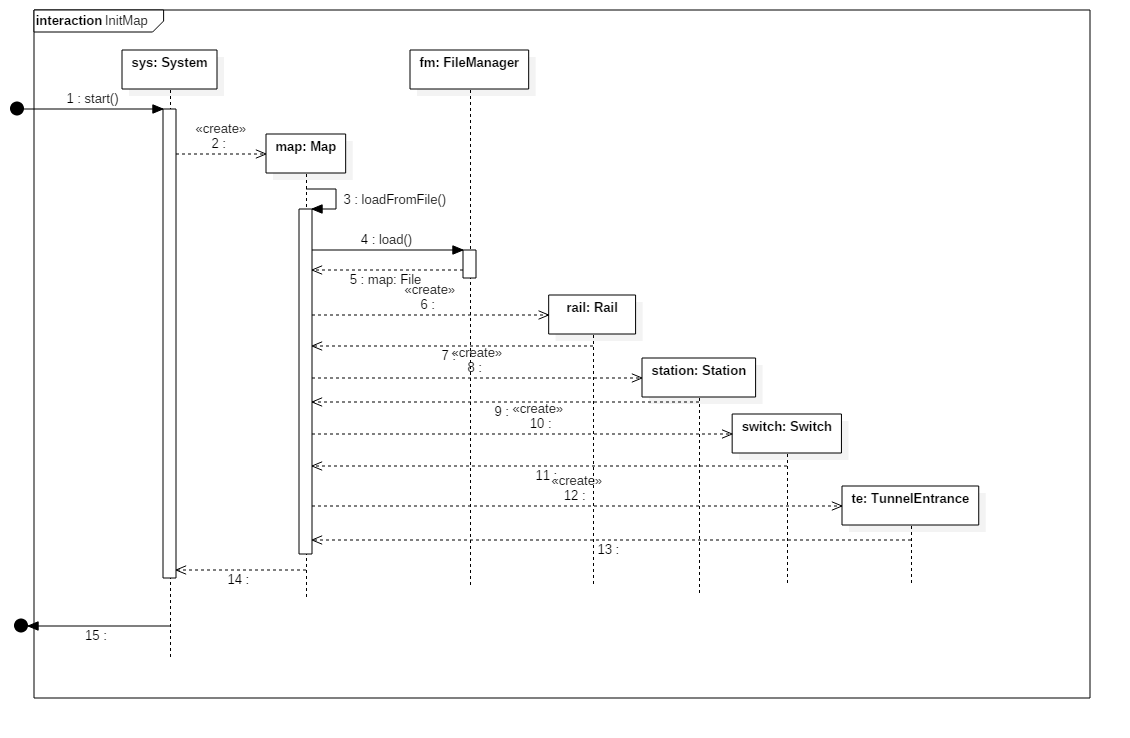
* + 1. ***MovingElement in Tunnel (use case: 5. MEinTunnel)***



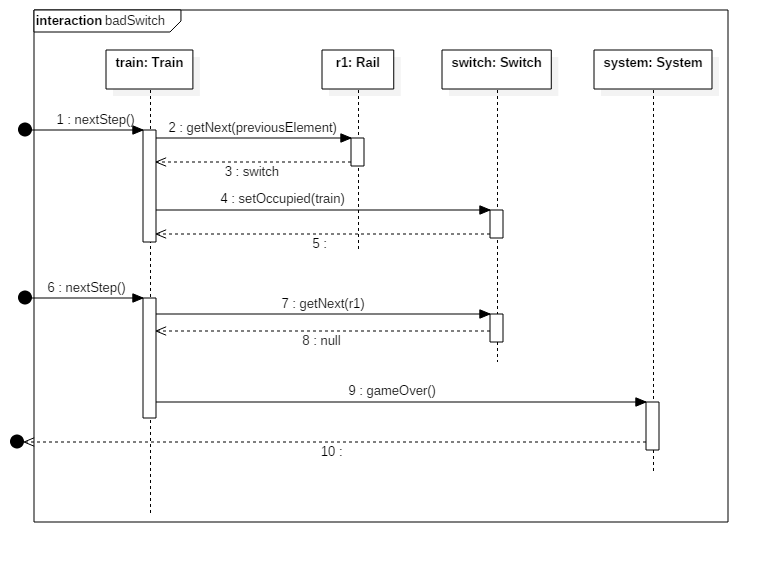
* + 1. ***vanish (use case: 6. MEvanish)***



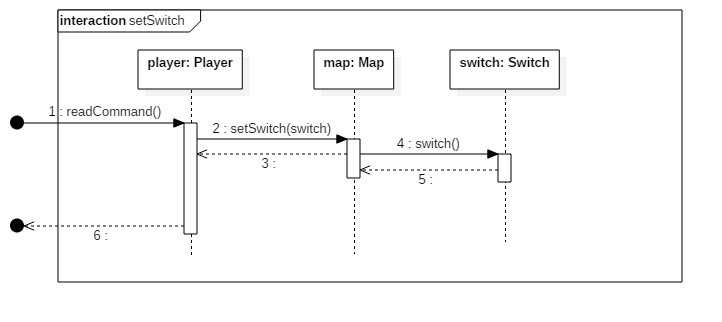
* + 1. ***initialize Map (use case: 7. initMap)***



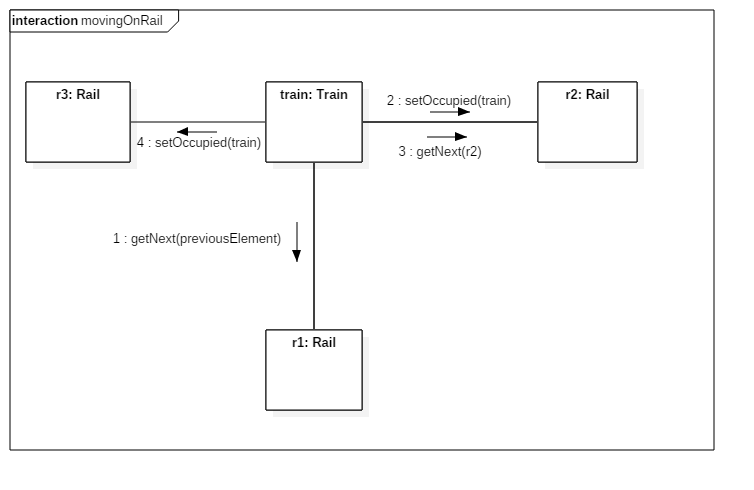
* + 1. ***bad Switch (use case: 8. badSwitch)***



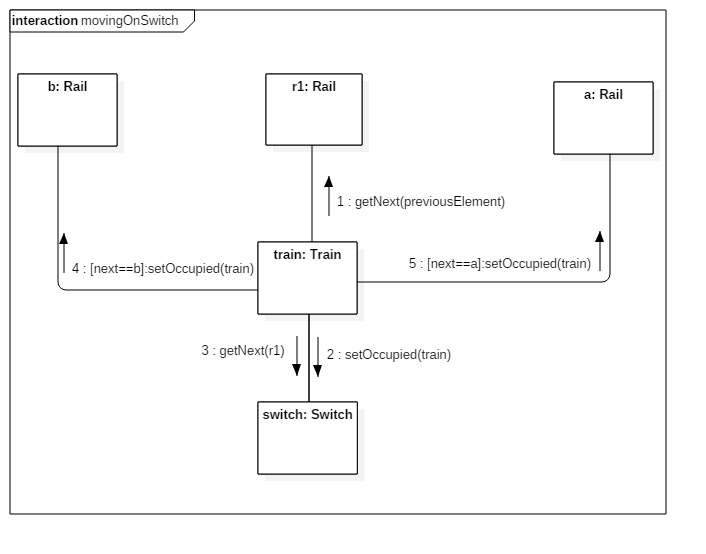
* + 1. ***set Switch (10. use case setSwitch)***



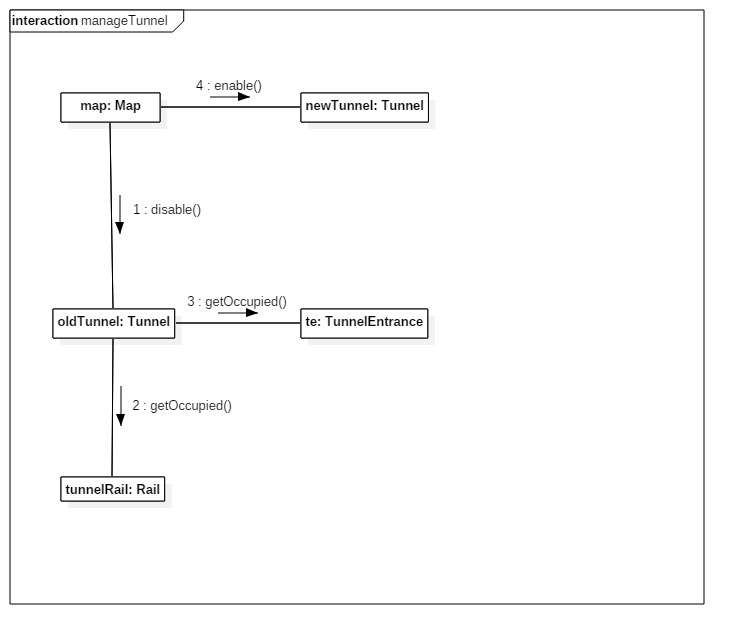
* 1. ***Kommunikációs diagramok***
     1. ***MEonRail***



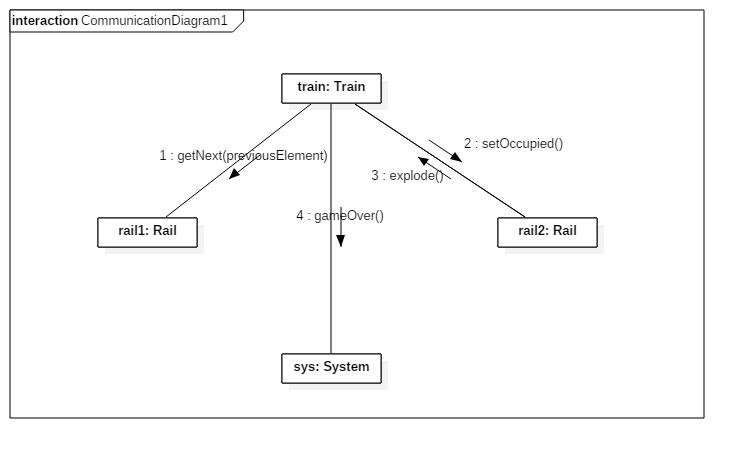
* + 1. ***MEonSwitch***



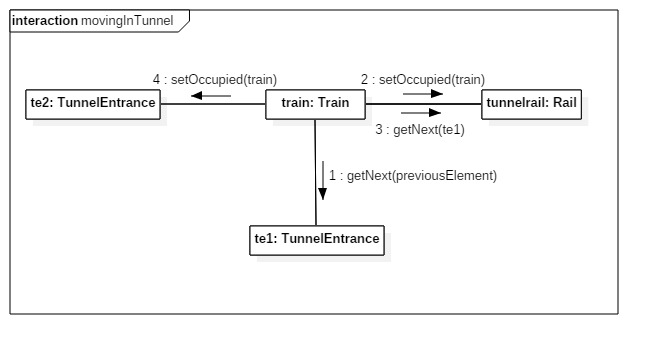
* + 1. ***ManageTunnel***



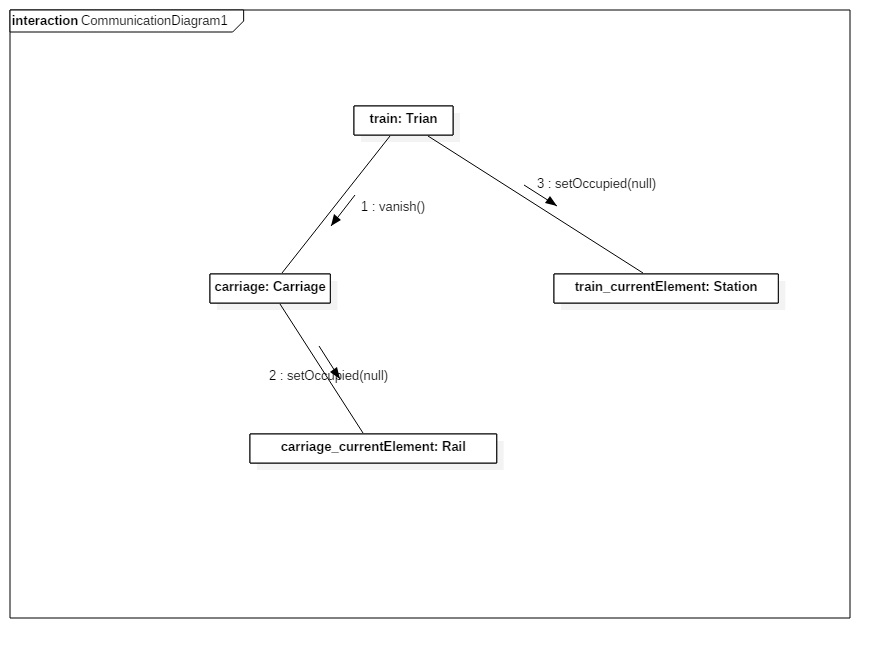
* + 1. ***Collision***



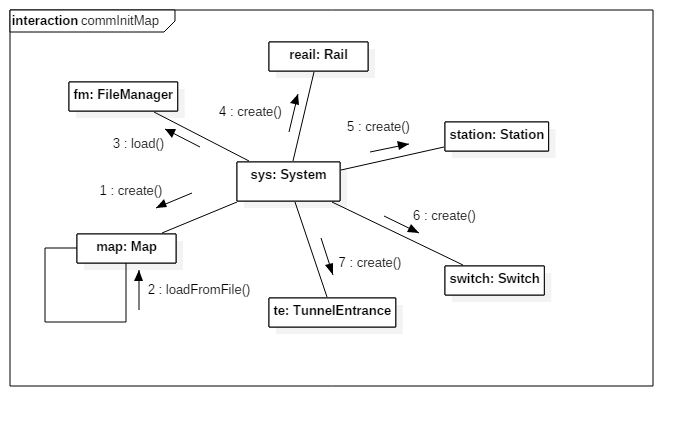
* + 1. ***MEinTunnel***



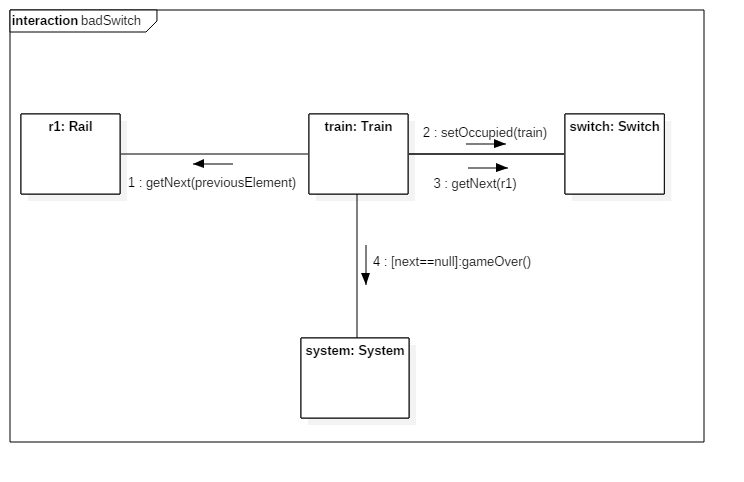
* + 1. ***MEvanish***



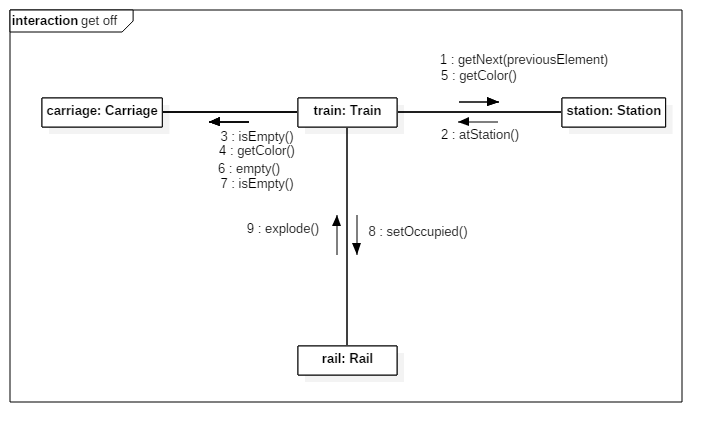
* + 1. ***initMap***



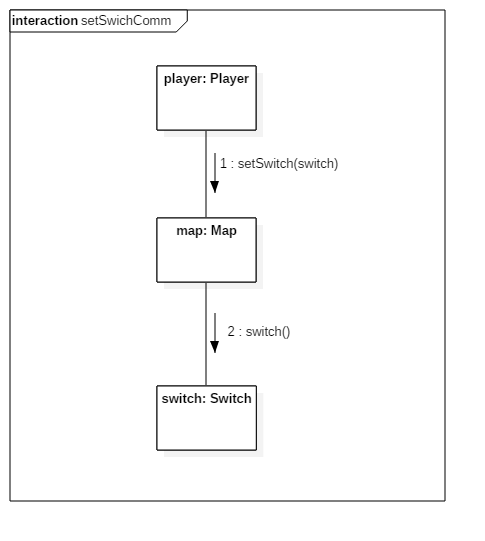
* + 1. ***badSwitch***



* + 1. ***getOff***



* + 1. ***setSwitch***



* 1. ***Napló***

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2017. 03. 06. 19:00 | 1 óra | Salamon  Papp  Fenes  Dobó  Vizi | Értekezlet. Az eheti feladat vázlatos megbeszélése az elkészítés lépéseinek megértése. |
| 2017. 03. 08. 19:00 | 1,5 óra | Salamon  Papp  Fenes  Dobó  Vizi | Felvázoltuk az esetleges usecaseeket és ezeket tagokhoz rendeltük. Mindenki feladata, hogy a neki kiosztott usecasehez tartozó szekvenciákat, kommunikációs diagramokat és leírásokat elkészítse.  Felosztás:  Salamon:   * átírás * 10. u.c.   Dobó:   * 1 u.c * 2 u.c * 9 u.c   Papp:   * 5 u.c * 7 u.c   Fenes:   * 3 u.c * 6 u.c   Vizi:   * 4 u.c * 8 u.c |
| 2017. 03. 08. 21:00 | 1 óra | Papp | Use-case diagram elkészítése az értekezleten felvetett gondolatok alapján, valamint ezen dokumentum inicializálása a közös szerkesztésre. |
| 2017. 03. 10.  08:00 | 1 óra | Salamon | getOff szekvenciadiagram elkészítése. |
| 2017. 03. 10.  12:00 | 1,5 óra | Vizi | MEatStation szekvencia és kommunikációs diagram elkészítése (bele lett illesztve a getOff szekvenciába, de a működése ekkor lett definiálva). |
| 2017. 03. 10.  16:00 | 1,5 óra | Dobó | Szekvencia diagramok elkészítése a rám kiszabott use-case esetekre, ezen diagramok formázása. |
| 2017. 03. 11.  10:00 | 1 óra | Salamon | getOff kommunikációs diagram elkészítése, a hozzá tartozó use-case leírás átgondolása, a dokumentum szerkezetének beállítása. |
| 2017.03. 11.  10:00 | 1,5 óra | Dobó | A szekvenciadiagramokhoz tartozó kommunkációs diagramok és use-case leírások elkészítése. |
| 2017.03.11. 11:00 | 1 óra | Fenes | A MEInTunnel és ManageTunnel use-casekhez tartozó leírás megírása, dokumentumba szerkesztése. |
| 2017. 03. 11. 11:00 | 3 óra | Papp | Collision és MEvanish szekvenciáinak és kommunikációs diagramjainak megírása.  A többi szekvenciadiagram, illetve a dokumentáció rendbetétele |
| 2017. 03. 11.  16:00 | 1 óra | Dobó | setSwitch use-case szekvenciadiagramjának és kommunikációs diagramjának elkészítése |
| 2017.03.11. 18:00 | 2 óra | Fenes | A MEInTunnel és ManageTunnel use-casekhez tartozó szekvenciadiagramok elkészítése, dokumentumba illesztése. |
| 2017. 03. 11.  19:00 | 1,5 óra | Vizi | initMap szekvenciadiagramjának és kommunikációs diagramjának elkészítése. |
| 2017. 03. 11.  22:00 | 1 óra | Fenes | MEInTunnel és ManageTunnel kommunikációs diagramok elkészítése. |
| 2017. 03. 12.  10:00 | 1,5 óra | Salamon | Az előző heti anyag javításának elkészítése, beillesztése a dokumentációba. |
| 2017. 03. 12.  11:00 | 1 óra | Vizi | A kezelői felület kigondolása, leírásának elkészítése. |
| 2017. 03. 12.  16:00 | 0,5 óra | Salamon | Dokumentáció véglegessé tétele (esztétikai javítás és konzisztencia megteremtése), nyomtatás. |