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

A terepasztal felelőssége, hogy tárolja, számontartja a szimulációhoz szükséges játékelemeket, például sín, állomás, vonat. Figyeli a vonatok összeütközéseit (vesztés), és hogy van-e a pályán még teli kocsi (győzelem, ha nincs).

* + 1. **Mozdony**

Síneken közlekedik, fő felelőssége, hogy maga után húzhat egy kocsit. Színe nincsen, és nem szállít utasokat. Soha nem áll meg.

* + 1. **Kocsi**

Síneken közlekedik, maga után húzhat egy kocsit. Színe van és utasokat szállíthat. A benne utazó utasok, ugyan olyan színűek.

* + 1. **Sín**

Számontartja a kapcsolódó síneket. Felelőssége, hogy a ráérkező mozdonyt továbbirányítsa a következő sín egységre.

* + 1. **Váltó**

A váltó egy olyan sín, amely n>=3 darab szomszédot tart számon. Felelőssége, hogy a ráérkező mozdonyt továbbirányítsa a megfelelő sín ágra, továbbá tárolja, hogy éppen melyik kijárat aktív.

* + 1. **Alagút Száj**

A terepasztalon csak speciális helyen helyezhető el. Csak egy másik alagút bejárattal együtt létezhet, ezek párban vannak, képesek egymást számon tartani.

* + 1. **BeSín**

Felelőssége, hogy időnként új mozdonyt és kocsikat generál a pálya szélén. A gyakoriság véletlenszerű, de függ a terepasztalon mozgó eddigi vonatok számától. Emellett figyel arra, hogy ha egy vonat ki megy a terepasztalról, akkor az ütközik.

* + 1. **Állomás**

A sínek mellett helyezkedik el. Felelőssége, hogy tárolja a színét, melyet a ráérkező vonat lekérdezhet. Az utasok pedig eszerint a szín szerint szállnak le a vonatról.

* + 1. **Timer**

A szimuláció időbeli futására szolgál, hatására a mozdonyok továbbhaladnak. Meghívja az újrarajzolást.

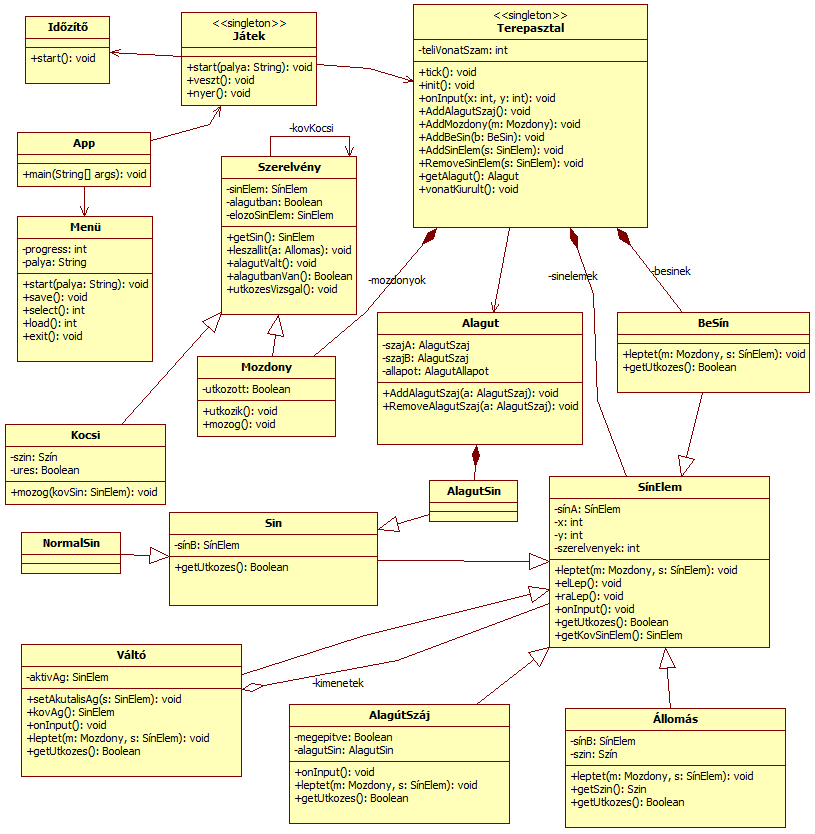
* + 1. **Menü**

A program funkcióinak elérésére szolgál, innen lehet a játékot elindítani, eredményjelző megtekintése, játék betöltése / mentése és kilépni. Ezen funkciók elérésére használt menüpontokat tárolja.

* + 1. **Játék**

Felelőssége a játék felépítése, objektumok létrehozása, vezérlése, terepasztal betöltése. Kezeli az időzítőt.

* 1. **Statikus struktúra diagramok**



* 1. **Osztályok leírása**
     1. **Alagut**
     + **Felelősség**Alagút megépültségének, és az alagút szájak illetve a köztük menő SínElemek számontartása.
     + **Ősosztályok**Nincs
     + **Interfészek**Nincs.
     + **Attribútumok**
       - **szajA: AlagutSzaj –** Az alagút egyik szája
       - **szajB: AlagutSzaj –** Az alagút másik szája
       - **allapot: AlagutAllapot –** NincsAlagutSzaj, EgyAlagutSzaj, VanAlagut
       - **alagutSinek: List<AlagutSin> -** Az alagút belsejében lévő sínek tárolása
     + **Metódusok**
       - **void AddAlagutSzaj(a: AlagutSzaj) –** Alagútszáj megépítését kezeli
       - **void RemoveAlagutSzaj(a: AlagutSzaj) –** Alagútszáj lerombolását kezeli
     1. **AlagutSin**
     + **Felelősség**Ugyanaz, mint a Sin felelőssége, de nem jelenik meg a térképen, mert földalatti.
     + **Ősosztályok**SinElem->Sin->AlagutSin
     + **Interfészek**Nincs.
     + **Attribútumok**Nincs.
     + **Metódusok**Nincs.
     1. **AlagutSzaj**
     + **Felelősség**Számon tartja az alagut másik AlagutSzaj-át, és hogy meg van-e építve.
     + **Ősosztályok**SinElem->AlagutSzaj
     + **Interfészek**Nincs.
     + **Attribútumok**
       - **megepitve: boolean -** Meg van-e építve az alagút ezen szája.
       - **alagutSin: AlagutSin –** A kapcsolódó AlagutSin
     + **Metódusok**
       - **void onInput() -** A bevitelre reagál az osztály.
       - **void leptet(m: Mozdony, s: SinElem) –** Felrobbantja a vonatot, ha nincs megépítve.
     1. **Allomas**
* **Felelősség**Tárol egy Színt, melyet a ráérkező vonat lekérdezhet.
* **Ősosztályok**SinElem -> Allomas
* **Interfészek**Nincs.
* **Attribútumok**
  + **sinB: SinElem –** a következő SinElem
  + **szin: Szín –** A kocsi eszerint dönti el, hogy kiürül-e
    - **Metódusok**
      * **void leptet(m: Mozdony, s: SinElem) –** szól a mozdonynak, hogy szállítsa le a kocsijairól az utasokat.
      * **Szin getSzin() –** visszaadja az állomás színét
    1. **App**
    - **Felelősség**

Felelőssége a view, controll és modell inicializálása.

* + - **Ősosztályok**Nincs.
    - **Interfészek**Nincs.
    - **Attribútumok**Nincs.
    - **Metódusok**
      * **void main() -** létrehozza az Idozito, Menu, Jatek objektumokat.
    1. **BeSin**
    - **Felelősség**

Belépési pontot biztosít az új vonatoknak a Terepasztalra. Nem engedi a vonatot kimenni a terepasztalról, felrobban.

* + - **Ősosztályok**SinElem->BeSin
    - **Interfészek**Nincs.
    - **Attribútumok**

Nincs.

* + - **Metódusok**
    - **void leptet(Mozdony m, SínElem s) -** Ha s=SinA akkor a vonat felrobban metódusát hívja meg
    1. **Idozito**
    - **Felelősség**Felelőssége a periodikus jelgenerálás. A játék időbeli szimulálásának alapja.
    - **Ősosztályok**Nincs.
    - **Interfészek**Nincs.
    - **Attribútumok**Nincs.
    - **Metódusok**
      * **void start() -** elindítja a jelgenerálási folyamatot
    1. **Jatek**
* **Felelősség**

Objektumok létrehozása: Terepasztal és az abban helyet foglaló Sínelemek betöltése fájlból. Időzítő tick továbbítása. Megjeleníti az Eredményjelzőt.

* **Ősosztályok**

Nincs.

* **Interfészek**

Nincs.

* **Attribútumok**

Nincs.

* **Metódusok**
  + **void start(palya: String) -** Létrehozza a Terepasztalt, és adott fájlból tölti be rá az elemeket.
  + **void veszt() -** A játékos elveszti a játékot.
  + **void nyer() -** A játékos megnyeri a játékot.
    1. **Kocsi**
    - **Felelősség**Ugyanaz a felelőssége, mint szülőjének, a Szerelvénynek. Ezenkívül tárolja a színét, ami alapján az utasok leszállnak a kocsiból.
    - **Ősosztályok**Szerelveny -> Kocsi
    - **Interfészek**Nincs.
    - **Attribútumok**
      * **szin: Szín –** A kocsi színe. Ez alapján dönti el hogy a kocsi kiürül-e az Állomáson.
      * **ures: Boolean -** Tárolja, hogy utaznak-e a kocsiban
    - **Metódusok**
      * **void mozog(kovSin: SinElem) –** frissíti az aktuális sínelemet, ahol tartózkodik, és mozgatja a következő kocsit (rekurzív)
    1. **Menu**
    - **Felelősség**

A program egyes menüpontjait tárolja.

* + - **Ősosztályok**Nincs.
    - **Interfészek**Nincs**.**
    - **Attribútumok**
      * **palya: String -** A pálya neve.
      * **progress: int** - Meddig jutott el a játékos a pályákon.
    - **Metódusok**
      * **void start(palya: String) -** új játék indítása
      * **void save() -** Elmenti az állást, hogy mennyi pályát nyertünk meg
      * **int select() -** Visszatér a pálya számával, amit kiválasztunk
      * **int load() -** Betölt egy állást, és visszatér azzal, hogy meddig jutottunk el.
      * **void exit() -** Kilép a játékból.
    1. **Mozdony**
    - **Felelősség**Kérdezgeti az alatta álló SínElemet, hogy melyik lesz a következő SínElem (leptet). Szól az első Kocsinak, hogy mozogjon (mozog).
    - **Ősosztályok**Szerelveny -> Mozdony
    - **Interfészek**Nincs.
    - **Attribútumok**
      * **utkozott: Boolean** - Alapesetben false, ha ütközik a vonat akkor true-ra állítódik
    - **Metódusok**
      * **void utkozik() -** Átállítja az utkozott változó értékét
      * **void mozog() –** szól a sínelemeknek, hogy lépett, és mozgatja a mögötte lévő kocsit.
    1. **NormalSin**
    - **Felelősség**Ugyanaz, mint a Sin felelőssége, megjelenik a térképen.
    - **Ősosztályok**SinElem->Sin->NormalSin
    - **Interfészek**Nincs.
    - **Attribútumok**Nincs.
    - **Metódusok**Nincs.
    1. **Sin**
    - **Felelősség**Továbbirányítja a mozdonyt a következő SínElemre.
    - **Ősosztályok**SinElem->Sin
    - **Interfészek**Nincs.
    - **Attribútumok**
      * **sinB: SinElem –** a másik kapcsolódó SínElem
    - **Metódusok**

Nincs.

* + 1. **SinElem**
    - **Felelősség**Mozgásteret biztosít a vonatok számára: adott SinElemről jött Mozdonynak megmondja, hogy melyik SinElem következik.
    - **Ősosztályok**Nincs.
    - **Interfészek**Nincs.
    - **Attribútumok**
      * **sinA: SinElem –** Az egyik kapcsolódó SínElem
      * **x: int -** az x pozíciója a SinElemnek
      * **y: int -** az y pozíciója a SinElemnek
      * **szerelvények: int –** számolja a rajta tartózkodó Szerelvények számát
    - **Metódusok**
      * **void leptet(m: Mozdony, s: SinElem) –** a sín rálépése előtt hívódik meg
      * **void elLep() –** csökkenti a szerelvenyek változót
      * **void raLep() –** növeli a szerelvenyek változót
      * **void onInput() -** A bevitelre reagál az osztály.
      * **Boolean getUtkozes() –** szerelvenyek függvényében dönti el, hogy van-e ütközés
      * **SinElem getKovSinElem() -** Visszaadja a következő sínElemet.
    1. **Szerelveny**
       - **Felelősség**Számon tartja, az előző és az aktuális SínElemet, amin tartózkodik. Tárolja, hogy alagútban van-e. Jelre tovább mozog, és Állomásra érve szól a mögötte lévő szerelvénynek, hogy
       - **Ősosztályok**

Nincs.

* + - * **Interfészek**  
        Nincs.
      * **Attribútumok**
* **sinElem**: **SinElem** – Melyik sínen található a szerelvény
* **elozoSinElem: Sinelem** – Melyik sínen volt utoljára
* **alagutban**: **boolean –** Alagútban van-e az adott szerelvény vagy sem
  + - * **Metódusok**
* **SinElem getSin()–** visszatér a jelenlegi SinElemmel.
* **void leszallit(a:** **Allomas) –** A kocsiból kiürülnek az utasok, ha megegyezik az állomás színével és az előtte levő kocsik üresek.
* **void AlagutValt() –** Átkapcsolja a szerelvényt, hogy alagútban van-e vagy sem.
* **Boolean alagutbanVan() -** visszatér az alagutban attribútum értékével
* **void utkozesVizsgal() –** megkérdezi az sínElemet amin áll, hogy van e még rajta kívül más.
  + 1. **Terepasztal**
       - **Felelősség**

A terepasztal felelőssége, hogy tárolja a SínElem-eket, BeSín-eket és Mozdony-okat. Figyeli, hogy van e a pályán még teli kocsi (győzelem ha nincs).

* + - * **Ősosztályok** Nincs.
      * **Interfészek** Nincs.
      * **Attribútumok**
* **mozdonyok: List<Mozdony> -** A mozdonyok listája
* **sinelemek:** **List<SinElem> -** A sínelemek listája
* **besinek: List<BeSin> -** A bemeneti sínpárok listája
* **teliVonatSzam: int -** a pályán utassal rendelkező vonatok száma
  + - * **Metódusok**
* **void tick() –** Szól a Mozdonyoknak, hogy mozogjanak.
* **void init() –** Feltölti elemekkel a terepasztalt
* **void AddAlagutSzaj() -** Hozzáad egy alagútszájat.
* **void onInput(int x, int y) -** Megnézi mindegyik SinElemre, hogy rakattintottak-e
* **void addMozdony(m: Mozdony)** – új mozdonyt ad hozzá a terepasztalhoz
* **void addBeSin(b: beSin) –** új BeSin elemet ad hozzá a terepasztalhoz
* **void addSinElem(s: SinElem) –** új SinElemet ad hozzá a terepasztalhoz
* **void removeSinElem(s: SinElem)** – töröl adott SinElemet
* **Alagut getAlagut() –** visszaadja az éppen megépült alagutat
* **void vonatKiurul() –** csökkenti a teliVonatok számát egyel. Ha 0 a játékos nyert.
  + 1. **Valto**
    - **Felelősség**

SinElem listájából az éppen aktívnak választott ág felé irányítja a mozdonyt. Ha ág felől jön, akkor SinA felé irányítja, és automatikusan vált aktív ágat.

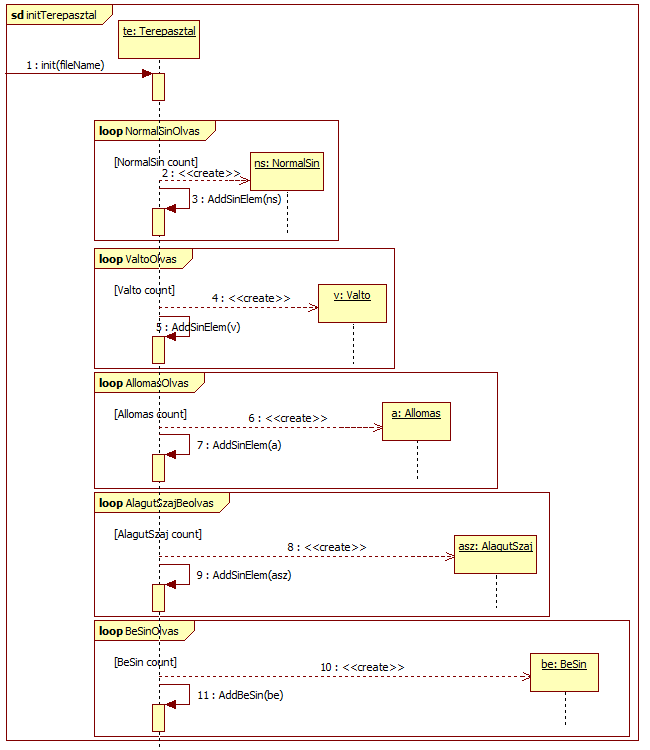
* + - **Ősosztályok**SinElem->Valto
    - **Interfészek**Nincs.
    - **Attribútumok**
      * **kimenetek: List<SinElem> -** az összes kimenő SinElem referenciája
      * **aktivAg: SinElem -** melyik ág az aktív
    - **Metódusok**
      * **void setAktualisAg(Sinelem s) -** aktivAg setter
      * **SinElem kovAg() -** sinek listából az aktivAg utáni SinElemet adja vissza
      * **void onInput() -** A bevitelre reagál az osztály.
      * **void leptet(m: Mozdony, s: SinElem) –** átvált, ha nem az aktívÁgból jött a mozdony
  1. **Szekvencia diagramok**
     1. **Start**



* + 1. **InitSzekvencia**



* + 1. **initTerepasztal**



* + 1. **Input**



* + 1. **Jatszik**



* + 1. **VonatMozgat**

****

* + 1. **KocsiMozgat**



* + 1. **UtkozesVizsglat**



* + 1. **ValtoClick**



* + 1. **ValtoValt**



* + 1. **AlagutSzajClick**



* + 1. **RemoveAlagut**

****

* + 1. **AddAlagut**

****

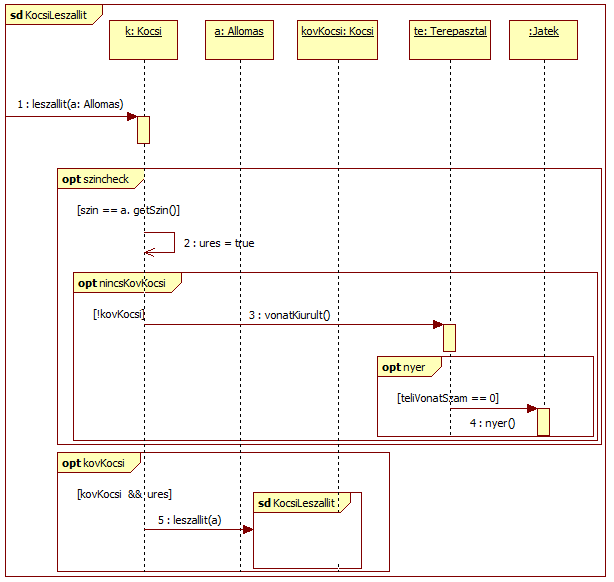
* + 1. **AlagutSzajLeptet**



* + 1. **AllomasLeptet**



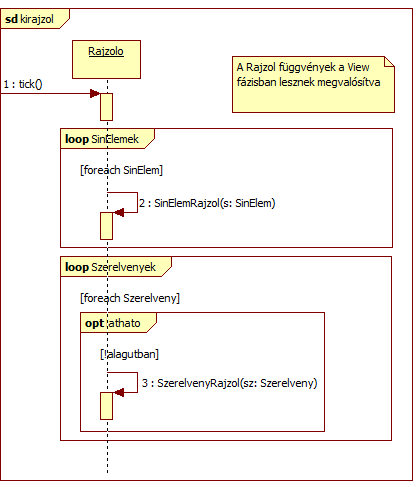
* + 1. **KocsiLeszallit**



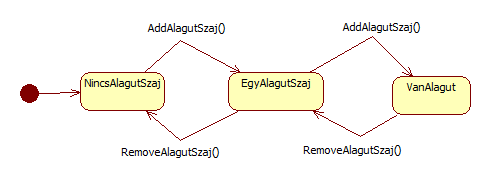
* + 1. **ValtoLeptet**



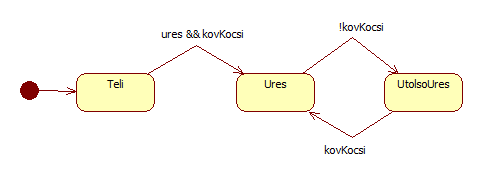
* + 1. **Kirajzol**



* 1. **State-chartok**
     1. **AlagutEpit state-chart**



* + 1. **Kocsi állapot state-chart**

****

* 1. **Napló**

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2017.03.02 19:00 | 2 óra | Szili | Néhány szekvencia diagram javítása, hibák összeírása, class diagram módosítás |
| 2017.03.03 16:30 | 3 óra | Szili | Szekvencia diagrammok javítása |
| 2017.03.03 16:30 | 3.5 óra | Krátky | Szekvencia diagrammok  State-chart |
| 2017.03.03 20:00 | 1.5 óra | Dócs | Szekvencia diagrammok asszisztálása |
| 2017.03.03 16:30 | 5 óra | Varga | Szekvenciák és state-chart diagrammok |
| 2017.03.04 14:00 | 5 óra | Dócs  Krátky  Sillye  Varga | Szekvencia diagrammok átgonddolása módosítása/javítása, state-chart diagram |
| 2017.03.04 17:00 | 2 óra | Szili | Szekvencia diagram, osztály diagram módosítás doksi véglegesítése |
| 2017.03.05 20:00 | 0.5 óra | Varga | doksi áttekintés, módosítások |
| 2017.03.05 20:30 | 0.5 óra | Szili | doksi áttekintése, módosítások |