

## 2. Követelmény, projekt, funkcionálitás

### 2.1 Bevezetés

#### 2.1.1 Cél

A dokumentum célja, hogy ismertesse a projekt követelményeit és funkcionálitását. Ezek segítségével egy kész szoftver előállítása a projekt végső célja.

#### 2.1.2 Szakterület

A szoftver egy szórakoztatás céljából készült videójáték. Vasúthálózatok kiépítésére és fentartására alkalmas szimulátor, mely a felhasználó stratégikus képességeit méretteti meg.

#### 2.1.3 Definíciók, rövidítések

**MVC:** Model-View-Control

**Szkeleton:** váz, megíratlan metódusokkal

**Proto:** kész metódusokkal rendelkezik, de nincs grafikus felület még

**HSZK:** Hallgatói Számítógép Központ

**UML:** Unified Modeling Language

**JRE:** Java Runtime Environment - futtatókörnyezet

**Eclipse:** fejlesztő környezet

**GIT:** verziókezelő

**GitLab:** verziókezelő rendszerre épülő internetes szolgáltatás

**GitHub:** verziókezelő rendszerre épülő internetes szolgáltatás és kliens

**IIT:** Irányítástechnika és Informatika Tanszék

**Skype:** Egy alkalmazás, ami lehetővé teszi az internetes hangalapú kommunikációt

**Verziókezelő:** egy olyan szolgáltatás, aminek segítségével el lehet tárolni a régebbi verzióit a fájloknak, így gond esetén bármikor vissza lehet térti egy-egy régebbi verzióhoz

#### 2.1.4 Hivatkozások

**Java** - <https://www.java.com>

**IIT** - <https://www.iit.bme.hu>

**Lego Loco** - [https://en.wikipedia.org/wiki/Lego\\_Loco](https://en.wikipedia.org/wiki/Lego_Loco)

**Traintown Deluxe** - [https://en.wikipedia.org/wiki/3D\\_Ultra\\_Lionel\\_Traintown](https://en.wikipedia.org/wiki/3D_Ultra_Lionel_Traintown)

#### 2.1.5 Összefoglalás

A dokumentum további részei ismertetni fogják a projekttel kapcsolatos további tervezeteket. Megtalálhatók a különféle funkciók és szoftverkövetelmények, illetve a projektnapló.

## 2.2 Áttekintés

### 2.2.1 Általános áttekintés

A kialakítandó szoftver legmagasabb szintű architekturális képe a Model-View-Control (MVC). E három alrendszerből fog felépülni a szoftver. A Model alkotja a vázat, melyben a legfontosabb eljárások találhatók. A View alrendszer segítségével a program grafikusan fogja megjeleníteni az adatokat, a Control pedig a felhasználói interakciókat kezeli. Ezek képezik a felhasználói kapcsolatok alapját. Hálózati hozzáférést a szoftver nem igényel. Adattárolási szempontból a szoftver képes lesz a játékmenet adott pillanatát fájlba elmenteni, és ezt egy későbbi futás során akár visszatölteni.

### 2.2.2 Funkciók

A **program** egy valós idejű vonatos stratégiai játék. A játék lényege, hogy az előre elkészített játékpályákon, azaz úgynevezett **terepasztalokon** vonatok jelennek meg látszólag véletlen időközönként a **bejövő távolsági sínpárokon**. A vonatokat a játékosnak kell terelnie váltók segítségével a megfelelő sínekre, amikkel el tudja juttatni az állomásokra az utasokat. Ezen kívül használhat úgynevezett alagutakat is.

A **terepasztal** a játék alapja, ezen található minden sín, alagút, vonat, állomás, váltó.

Az **alagutak** arra használhatók, hogy megépítésük után a vonat bármelyik oldalról belemehet, és a megépített alagút túloldalán jön ki belőle, ezzel létrehozva egy névleges új sínt. Az alagutat csak bizonyos helyekre lehet építeni a pályán, azaz csak olyan sínszakaszokra, amik meg vannak jelölve, hogy alagútépítésre alkalmasak. Az alagút építésének folyamata úgy működik, hogy először a játékos elhelyezi az egyik végét, ekkor még csak tervben van az építkezés, azaz az alagút még nem funkcionál, a vonatok elmennek mellette, és nem mennek bele. Amint lerakjuk a másik végét az alagútnak, megépül, és használhatóvá válik a vonatok számára. Az alagút lebontása hasonló elven működik, bármelyik végét lebonthatjuk, ezáltal megszűnik a funkcionálitása, azaz mivel csak egyik alagút bejárat van megépítve, így a vonatok nem mennek az alagútba. A Sugár Állami Vasutak (SÁV) balesetmentes közlekedési tervezetének köszönhetően a balesetek elkerülésének érdekében az alagút nem bontható, amíg vonat tartózkodik benne. Az alagút hossza a bejáratok távolságával egyenesen arányos.

A **váltók** ennél egyszerűbben működnek, itt a játékos egyszerűen átkapcsolhatja a váltót, ezáltal módosítva a lehetséges pályáját a vonatoknak. A váltó nem kapcsolható, ha egy vonat éppen halad át rajta, és a SÁV nem is javasolja, hogy megpróbáljuk. Amennyiben egy vonat a váltó inaktív ágáról érkezik, a váltó automatikusan át fog állni. A **sínek** működésük szempontjából elég egyszerűek, minden végükkel kapcsolódnak vagy egy másik sínhez, vagy a bejövő távolsági sínpárokhoz, illetve kapcsolódhatnak még váltókhöz és alagútbejáratokhoz is. Ha ezek bármelyikén egynél több vonat tartózkodik, akkor azok felrobbannak, és elveszítjük a játékot.

A **bejövő távolsági sínpárok** olyan különleges sínpárok, ahonnan érkezhetnek vonatok. Több is lehet belőlük a terepasztalon, így nehezítve a játékos dolgát, megosztva a figyelmét.

A **vonatok** egy mozdonyból és egy vagy több vasúti kocsiból állnak. Számuk a pálya nehézségtől függ, és látszólag véletlen időközönként érkeznek a bejövő távolsági sínpárokon.

A **mozdony** minden vonat elemi része, a vasúti kocsik elején található minden esetben, azaz nem foglalkozunk tolatómozdonyos vonatokkal. Nem utaznak utasok a mozdonyban, a mozdonyvezető csak az után hagyja el a mozdonyt, hogy megnyertük a játékot és már nem látjuk.

Az **állomások** a sínek mellett találhatóak közvetlenül, és színük is van. Amennyiben egy

ugyanilyen színű vasúti kocsi érkezik az állomáshoz, és a vasúti kocsi előtt található kocsik már kiürültek, akkor ez a kocsi is kiürül, mégpedig a SÁV balesetvédelmi előírásainak megfelelően leugranak az utasok a vonatról.

A játék célja az, hogy a játékos az összes pályán lévő vonatnak a vasúti kocsijait kiürítse. Ezt úgy tudja elérni, hogy a váltókat kapcsolatja és alagutakat épít, ezáltal a megfelelő állomásokhoz juttatva a vonatokat, miközben arra is figyel, hogy ne ütközzenek. Amint az összes vonat összes kocsija üres, a játékos nyert. Amennyiben bármelyik vonat ütközik egy másikkal, a játékos vesztett.

Amennyiben a játékos **megnyerte** az adott pályát, a következő pályára lép és győzelmenek ideje rögzítésre kerül az **eredményjelzőn**, ahol minél gyorsabban sikerült megoldania a pályát, annál jobbnak számít az ideje.

### 2.2.3 Felhasználók

A felhasználó nagyon széles körű spektrumból jöhet, hiszen a videójátékok témaköre sok embert érint. Előismeretek nélkül is könnyen megtanulható a szoftver. Feltételezhető, hogy alapszinten ért a számítógép kezeléséhez, mint például egérmozgatás és kattintás.

### 2.2.4 Korlátozások

Előírás, hogy a szoftvert java nyelven kell írni, és az alapvető könyvtárakon kívül semmilyen más külső könyvtár nem használható.

### 2.2.5 Feltételezések, kapcsolatok

**Java** - <https://www.java.com>

A fejlesztéshez használt programnyelv weboldala.

**IIT** - <https://www.iit.bme.hu>

A megrendelő weboldala, a feladatkiírás itt olvasható.

**Lego Loco** - [https://en.wikipedia.org/wiki/Lego\\_Loco](https://en.wikipedia.org/wiki/Lego_Loco)

98-as videójáték, mely inspirációként szolgál és segít a tervezésben.

**Traintown Deluxe** - [https://en.wikipedia.org/wiki/3D\\_Ultra\\_Lionel\\_Traintown](https://en.wikipedia.org/wiki/3D_Ultra_Lionel_Traintown)

99-es videójáték, szintén ötletmerítésre

## 2.3 Követelmények

### 2.3.1 Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
1.01	a szélekről vonatok indulnak	bemutatás	alapvető	megrendelő		
1.02	a vonatban egy mozdony és kocsik	bemutatás	alapvető	megrendelő		Egy darab mozdony és mögötte a kocsik
1.03	a vonatok sínen közlekednek	bemutatás	alapvető	megrendelő		
1.04	elágazásoknál a váltók állíthatók	bemutatás	alapvető	megrendelő	Váltó állít	rá kell kattintani
1.05	két alagútbejárat építhető és rombolható	bemutatás	alapvető	megrendelő	Alagút épít	vagy mindkettő aktív, vagy egyik sem
1.06	a kocsik sorban ürülnek ki, színek szerint a megállokánál	bemutatás	alapvető	megrendelő	Állomás-hoz ér	győzelem
1.07	két vonat összefüggő	bemutatás	alapvető	megrendelő		veszítés
1.08	pálya teljesítéskor az állás elmentődik	bemutatás	opcionális	csapat		
1.09	váltó átváltódik ha nem abba az irányba áll amerről a vonat jön	bemutatás	alapvető	csapat	Váltó állít	
1.10	jelzés, ha vonat érkezik a pályára	bemutatás	fontos	csapat		ütközések elkerülésére
1.11	vonat visszajön, ha kimegy a pályáról	bemutatás	fontos	csapat		

### 2.3.2 Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
2.01	Git	nincs	alapvető	csapat	verziókezelő
2.02	Gitlab repo	nincs	alapvető	csapat	tárhely
2.03	Eclipse	nincs	opcionális	csapat	fejlesztő környezet
2.04	WhiteStarUML	nincs	opcionális	csapat	UML diagram-szerkesztő
2.05	JRE	bemutatás	alapvető	megrendelő	futtató környezet
2.06	elég teljesítményű hardver	bemutatás	alapvető	megrendelő	elég: HSZK gépei (monitor, egér is)

### 2.3.3 Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
3.01	Szkeleton bemutatása március 22-én	bemutatás	alapvető	megrendelő	
3.02	Proto bemutatása április 19-én	bemutatás	alapvető	megrendelő	
3.03	Véleges bemutatás május 10-én	bemutatás	alapvető	megrendelő	grafikus felülettel

### 2.3.4 Egyéb nem funkcionális követelmények

Nincsenek egyéb nem funkcionális követelmények.

## 2.4 Lényeges use-case-ek

### 2.4.1 Use-case leírások

Use-case neve	Váltó állít
Rövid leírás	A felhasználó állítani tudja a váltókat.
Aktorok	User
Forgatókönyv	A felhasználó rákattint a váltóra, így a sín átáll egy másik irányba, ezzel módosítja a vonat útját.

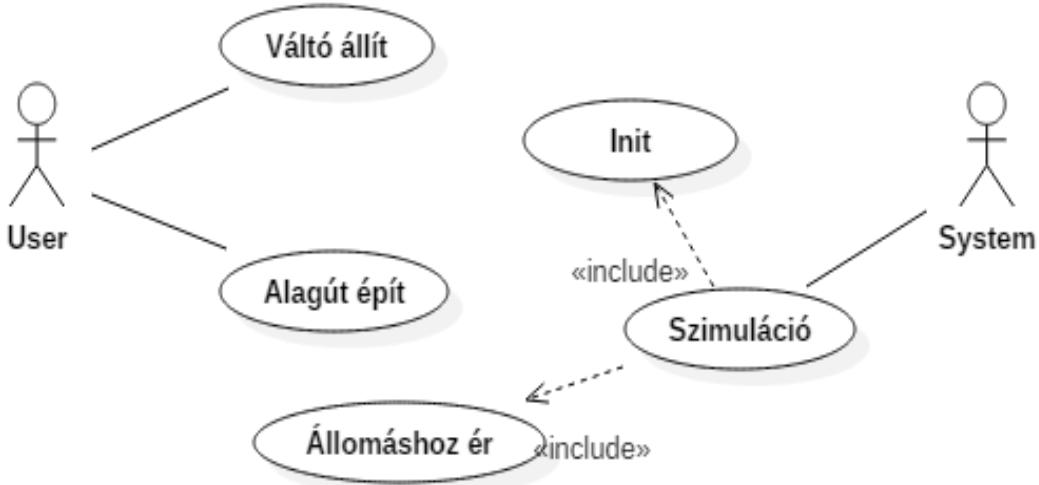
Use-case neve	Alagút épít
Rövid leírás	A felhasználó új alagutakat tud hozzáadni.
Aktorok	User
Forgatókönyv	A felhasználó először az alagút egyik végét, majd a másikat kell, hogy megépítse. Ezután a vonatok képesek az alagút két vége között közlekedni.

<b>Use-case neve</b>	Szimuláció
<b>Rövid leírás</b>	A vasúthálózaton a váltóknak és az alagutaknak megfelelően vonatok közlekednek.
<b>Aktorok</b>	System
<b>Forgatókönyv</b>	A vonatok a síneken és az alagutakban közlekednek. Továbbá az init és az Állomáshoz ér include-okon keresztül valósul meg.

<b>Use-case neve</b>	init
<b>Rövid leírás</b>	A vasúthálózat, váltók, sínek, vonatok létrehozása, elindítása.
<b>Aktorok</b>	System
<b>Forgatókönyv</b>	A vasúthálózatok egy előre megírt fájlból lesznek betölthetőek.

<b>Use-case neve</b>	Állomáshoz ér
<b>Rövid leírás</b>	A vonat elhalad az állomás mellett.
<b>Aktorok</b>	System
<b>Forgatókönyv</b>	A vonatról leugranak az utasok az adott kocsi színének megfelelően.

#### 2.4.2 Use-case diagram



## 2.5 Szótár

**Alagútépítés:** Kattintással építhető a bejárat. Akkor épül meg az alagút, ha már minden bejárata megépült.

**Alagút:** Sínek összekötésére és ütközések elkerülésére szolgáló építmény, egy darab található belőle a terepasztalon.

**Alagútbejárat:** Az alagúthoz két bejárat tartozik, ezeket a játékos építheti meg.

**Állomás:** A terepasztalon a sínek mellett található. Színe alapján szállnak le az utasok.

**Bejövő távolsági sínpár:** A vonat ezen érkezik a terepasztalra.

**Cél:** Az összes vonat kiürítése, az utasok megfelelő állomásra eljuttatása ütközés nélkül.

**Eredményjelző:** minden pályához tartozik egy, ezen sorrendben szerepelnek az adott pályát legeredményesebben teljesítő játékosok.

**Győzelem:** A cél elérése vesztés nélkül.

**Játékos:** Az ember, aki játszani akar a játékkel.

**Kocsi:** Utasokat szállít szín szerint. Kiürül, ha azonos színű állomáson halad át és nincs előtte teli kocsi.

**Mozdony:** A vonat elején helyezkedik el, nincs színe

**Összeütközés:** Két vonat egymással szembe kerül és egymásba rohannak. Ekkor a játékos elveszti a játékot.

**Sín:** Ezeken közlekednek a vonatok.

**Terepasztal:** ezen helyezkednek el az állomások, sínek és vonatok.

**Utas:** Abban a színű kocsiban ül, amelyik színű állomáson le szeretne szállni. Cél, hogy elszállítsunk minden utast.

**Váltó állítás:** Kattintással állítható át a másik sínpárra.

**Váltó:** Vonatok átirányítására szolgál.

**Vesztés:** Két vonat összeütközése esetén.

**Vonat:** egy darab mozdonyból és legalább egy darab kocsiból áll.

## 2.6 Projekt terv

### 2.6.1 Ütemterv

Dátum	Feladat
február 20.	Követelmény, projekt, funkcionalitás - beadás
február 27.	Analízis modell kidolgozása 1. - beadás
március 6.	Analízis modell kidolgozása 2. - beadás
március 13.	Szkeleton tervezése - beadás és a dokumentum herculesre való feltöltése.
március 20.	Szkeleton - beadás és a forráskód herculesre való feltöltése
március 27.	Prototípus koncepciója - beadás
április 3.	Részletes tervezek - beadás
április 10.	Prototípus készítése, tesztelése
április 17.	Prototípus - beadás és a forráskód, a tesztbemenetek és az elvárt kiemeltek herculesre való feltöltése
április 24.	Grafikus felület specifikációja - beadás
május 1.	Grafikus változat készítése
május 8.	Grafikus változat - beadás és a forráskód herculesre való feltöltése
május 10.	Összefoglalás - beadás és feltöltés

## 2.6.2 Csapat

Név	Feladat
Dócs Zoltán	Kód, dokumentáció, naplózás
Krátky Gergő Ádám	Kód, dokumentáció, dokumentáció organizálása
Sillye Márk	Kód, dokumentáció, visual designer
Szili Péter	Kód, dokumentáció, nyomtatás, GitLab organizálása
Varga János	Kód, dokumentáció, UML

## 2.6.3 Kommunikáció

Alapvetően kommunikációt élőben vagy Facebook chat-en folytatjuk, emellett hetente minimum egyszer csapat megbeszélést tartunk.

Csapat megbeszélésekhez Skype konferenciahívást használunk.

Dokumentáció megosztása, írása a Google Docs-ban és az Online Office-ban történik. A dokumentumokhoz mindenkinél teljes hozzáférés van.

Verziókezeléshez GitLab-ot használunk, GitHub klienssel.

## 2.7 Napló

Kezdet	Időtartam	Részvevők	Leírás
2017.02.12. 15:14	1 óra	Dócs Krátky Sillye Szili Varga	Projekt előkészítése, ötletelés, feladat kiírás átnézése
2017. 02. 18. 11:00	2 óra	Dócs Krátky Sillye Szili Varga	Csapat megbeszélés, dokumentáció készítése
2017. 02. 18. 13:30	3,5 óra	Dócs Krátky Sillye Szili Varga	Dokumentáció szerkesztése
2017. 02. 19. 19:45	0,5 óra	Krátky	Dokumentáció átolvasása
2017. 02. 19. 20:50	0,5 óra	Szili	Dokumentáció átolvasása
2017. 02. 19. 20:15	0,5 óra	Varga	Dokumentáció átolvasása
2017. 02. 19. 21:00	0,5 óra	Dócs	Dokumentáció átolvasása
2017. 02. 19. 22:00	0,5 óra	Sillye	Dokumentáció átolvasása

## 3. Analízis modell kidolgozása

### 3.1 Objektum katalógus

#### 3.1.1 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).

#### 3.1.2 Mozdony

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

#### 3.1.3 Kocsi

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

#### 3.1.4 Utas

Az utas abban a színű kocsiban utazik, amelyik színű állomásra el szeretne jutni. Akkor száll le, ha a kocsija előtt lévő kocsi már üres, és megérkezik a kiválasztott állomásra.

#### 3.1.5 Sín

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

#### 3.1.6 Váltó

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

#### 3.1.7 Alagút Száj

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

#### 3.1.8 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 azt kis idő elteltével "visszadorsítja", újra beengedi a játékba.

#### 3.1.9 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.

#### 3.1.10 Timer

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

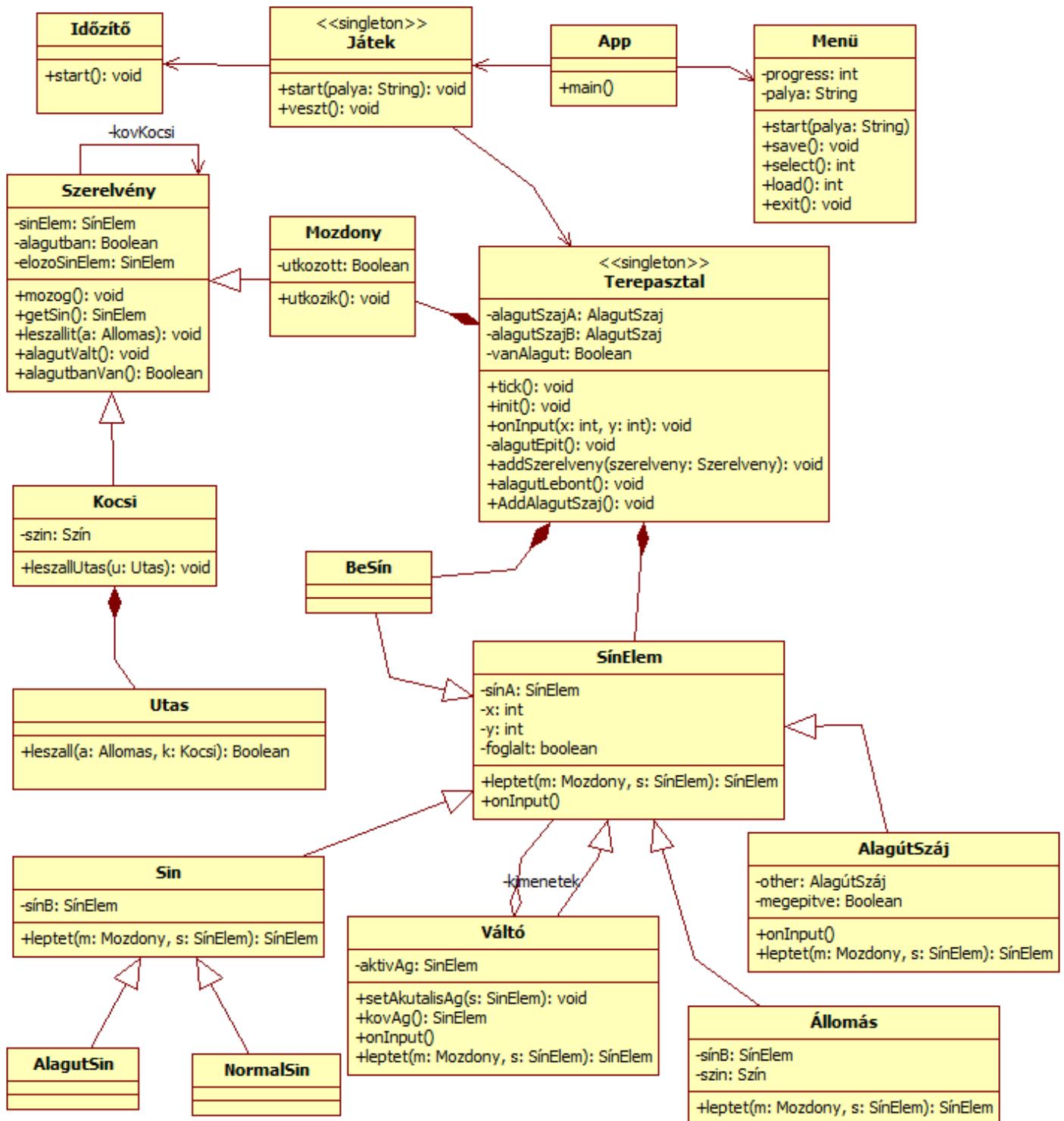
#### 3.1.11 Menü

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

#### 3.1.12 Játék

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

## 3.2 Statikus struktúra diagramok



### 3.3 Osztályok leírása

#### 3.3.1 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.

#### 3.3.2 AlagutSzaj

- **Felelősség**

Számon tartja az alagut másik AlagutSzaj-át, és hogy meg van-e építve.

- **Ősosztályok**

SinElem->AlagutSzaj

- **Interfészek**

Nincs.

- **Attribútumok**

- **other: AlagutSzaj** – A másik AlagutSzaj referenciajára

- **megepitve: boolean** - Meg van-e építve az alagút ezen szája.

- **Metódusok**

- **void onInput()** - A bevitelre reagál az osztály.

#### 3.3.3 Allomas

- **Felelősség**

Tárol egy Színt, melyet a ráérkező vonat lekérdezhet.

- **Ősosztályok**

SinElem -> Allomas

- **Interfészek**

Nincs.

- **Attribútumok**

- **sinB: SinElem** – a következő SinElem

- **szin: Szín** - Az Utas eszerint dönti el, hogy le száll-e

- **Metódusok**

Nincs.

### 3.3.4 App

- **Felelősség**

Felelőssége a view, controll és modell inicializálása.

- **Ősosztályok**

Nincs.

- **Interfészek**

Nincs.

- **Attribútumok**

Nincs.

- **Metódusok**

- **void main()** - létrehozza az Idozito, Menu, Jatek objektumokat.

### 3.3.5 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**

- **SínElem leptet(Mozdony m, SínElem s)** - Ha s=SinA akkor a vonat felrobban metódusát hívja meg, különben bevezeti a vonatot SinA-ra

### 3.3.6 Idozito

- **Felelősség**

Felelőssége a periodikus jelgenerálás. A játék időbeli szimulálásának alapja.

- **Ősosztályok**

Nincs.

- **Interfészek**

Nincs.

- **Attribútumok**

Nincs.

- **Metódusok**

- **void start()** - elindítja a jelgenerálási folyamatot

### 3.3.7 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.

### 3.3.8 Kocsi

- **Felelősség**

Ugyanaz a felelőssége, mint szülőjének, a Szerelvénynek. Ezenkívül tárolja a színét, és szól az utasoknak, hogy szálljanak le, ha akarnak.

- **Ősosztályok**

Szerelveny -> Kocsi

- **Interfészek**

Nincs.

- **Attribútumok**

- **szin: Szín** – A kocsi színe. Ez alapján dönti el az Utas, hogy leszáll-e az Állomáson.
- **utasok: List<Utas>** - A kocsin utazó utasok listája

- **Metódusok**

- **void leszallUtas(u Utas)**

### 3.3.9 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**

- **String palya()** - Visszaadja a kiválasztott pályához tartozó file címét.
- **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.

### 3.3.10 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**

- **kocsi: Kocsi** – Az első kocsi referenciajára
- **utkozott: Boolean** - Alapesetben false, ha ütközik a vonat akkor true-ra állítódik

• **Metódusok**

- **void utkozik()** - Átállítja az utkozott változó értékét

### 3.3.11 NormalSin

• **Felelősség**

Ugyanaz, mint a Sin felelőssége, megjelenik a térképen.

• **Ősosztályok**

SínElem->Sin->NormalSin

• **Interfészek**

Nincs.

• **Attribútumok**

Nincs.

• **Metódusok**

Nincs.

### 3.3.12 Sin

• **Felelősség**

Továbbirányítja a mozdonyt a következő SínElemre.

• **Ősosztályok**

SínElem->Sin

• **Interfészek**

Nincs.

• **Attribútumok**

- **sinB: SínElem** – a másik kapcsolódó SínElem

• **Metódusok**

Nincs.

### 3.3.13 SinElem

- **Felelősség**

Mozgásteret biztosít a vonatok számára: adott SinElemről következik.

- **Ősosztályok**

Nincs.

- **Interfészek**

Nincs.

- **Attribútumok**

- **sinA: SinElem** – Az egyik kapcsolódó SínElem
- **foglalt: boolean** - Igaz, amikor szerelvény halad át rajta.
- **x: int** - az x pozíciója a SinElemnek
- **y: int** - az y pozíciója a SinElemnek

- **Metódusok**

- **SinElem leptet(Mozdony m, SinElem s)** - Megadja a mozdynak a következő SinElemet az előző SinElem függvényében
- **void onInput()** - A bevitelre reagál az osztály.

### 3.3.14 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**

- **sin: SinElem** – Melyik sínen található a szerelvény
- **elozoSin: Sinelem** – Melyik sínen volt utoljára
- **alagutban: boolean** – Alagútban van-e az adott szerelvény vagy sem
- **kovKosci: Szerelveny** - A következő szerelvény referenciája

- **Metódusok**

- **void mozog()**– A következő SínElere lépteti a Szerelveny-t.
- **SinElem getSin()**– visszatér a jelenlegi SinElammal.
- **void leszallit(a: Allomas)** – Megkérdezi az utasokat, hogy leszállnak-e.
- **void AlagutValt()** – Átkapcsolja a szerelvényt, hogy alagútban van-e vagy sem.
- **boolean alagutbanVan()** - visszatér az alagutban attribútum értékével

### 3.3.15 Terepasztal

- **Felelősség**

A terepasztal felelőssége, hogy tárolja a SínElem-eket, BeSín-eket és Mozdony-okat. Figyeli a vonatok összeütközéseit (vesztés), és hogy van-e a pályán még teli kocsi (győzelem ha nincs). Ő építi és bontja le az alagutakat, adja hozzá a szerelvényeket a játékhoz.

- **Ősosztályok**

Nincs.

- **Interfészek**

Nincs.

- **Attribútumok**

- **mozdonyok: List<Mozdony>** - A mozdonyok listája
- **sinelemek: List<SinElem>** - A sínelemek listája
- **besinek: List<BeSin>** - A bemeneti sínpárok listája
- **alagutSzajA: AlagutSzaj** - Az egyik aktív alagútszáj
- **alagutSzajB: AlagutSzaj** - A másik aktív alagútszáj
- **vanAlagut: Boolean** - Van-e éppen megépült alagút

- **Metódusok**

- **void tick()** – Szól a Mozdonyoknak, hogy mozogjanak.
- **void init()** – Feltölti elemekkel a terepasztalt
- **void addSzerelveny(szerelveny: Szerelveny)** - Kiválaszt egy véletlen BeSín-t és lerakja rá a Szerelvénnyt
- **void alagutLebont()** - Lebont egy alagutat.
- **void AddAlagutSzaj()** - Hozzáad egy alagútszájat.
- **void onInput(int x, int y)** - Megnézi mindegyik SinElemre, hogy rakattintottak-e

### 3.3.16 Utas

- **Felelősség**

Ellenőrzi, hogy adott megállónál le akar-e szállni. (Komparálja a Kocsijának és az Állomás színét, de már nem kell vizsgálnia az előtte lévő kocsi ürességét)

- **Ősosztályok**

Nincs.

- **Interfészek**

Nincs.

- **Attribútumok**

Nincs.

- **Metódusok**

- **boolean leszall(Allomas a, Kocsi k)** - a és k színét komparálja, ez alapján igaz/hamis értéked ad vissza a kocsinak, hogy le akar-e szállni.

### 3.3.17 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 ágot.

- **Ősosztályok**

SinElem->Valto

- **Interfészek**

Nincs.

- **Attribútumok**

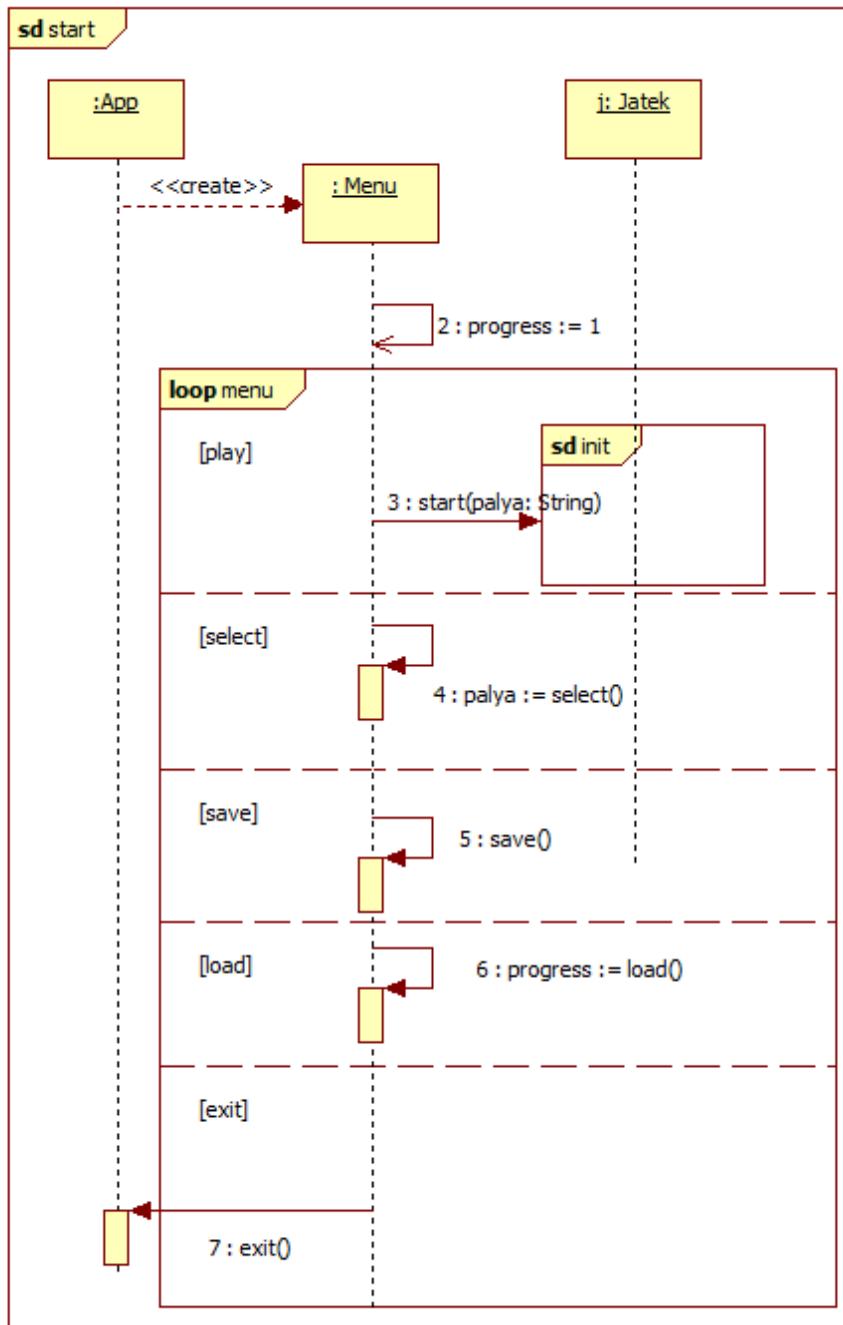
- **kimenetek: List<SinElem>** - az összes kimenő SinElem referenciája
- **aktivAg: SinElem** - melyik ág az aktív

- **Metódusok**

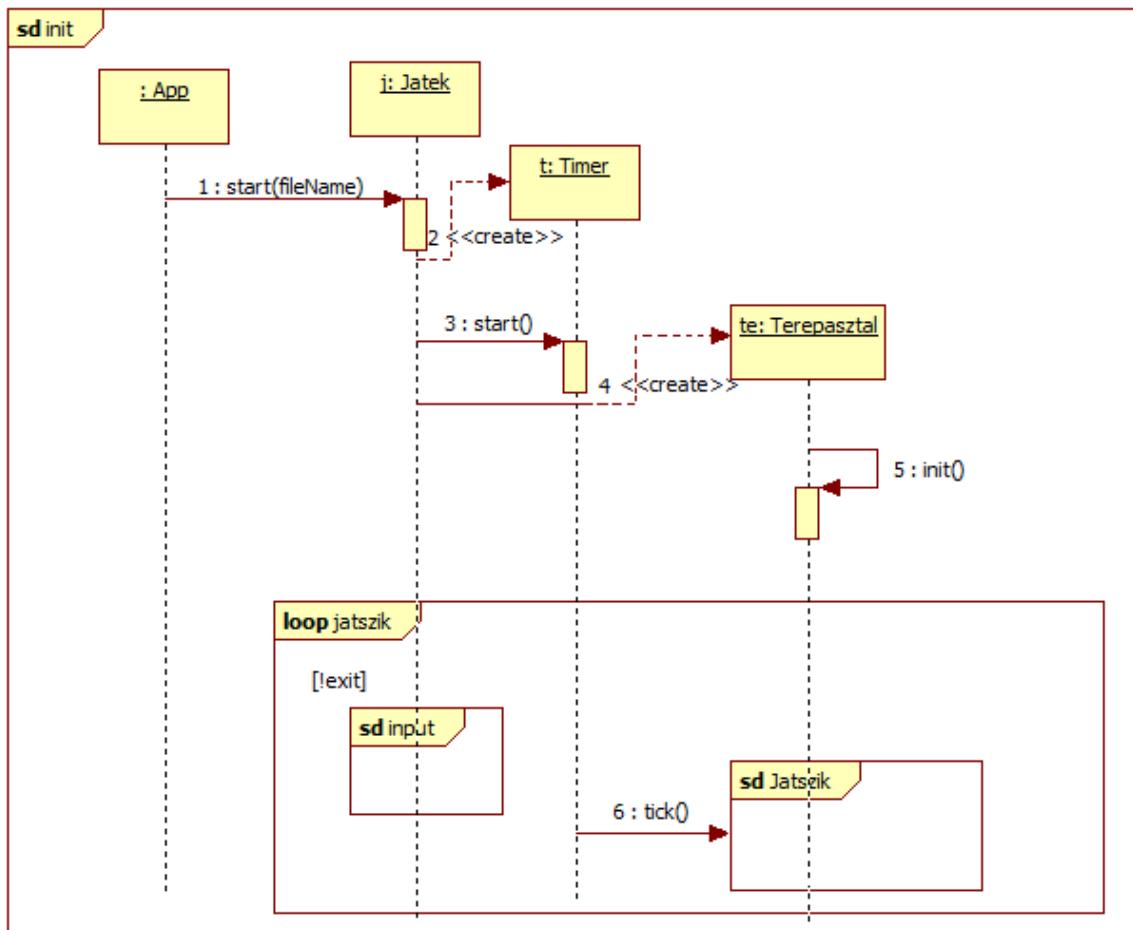
- **void setAktualisAg(Sinelem s)** - aktivAg setter
- **SinElem kovAg()** - sinek listából az aktivAg utáni SinElemet adja vissza
- **void onInput()** - A bevitelre reagál az osztály.

## 3.4 Szekvencia diagramok

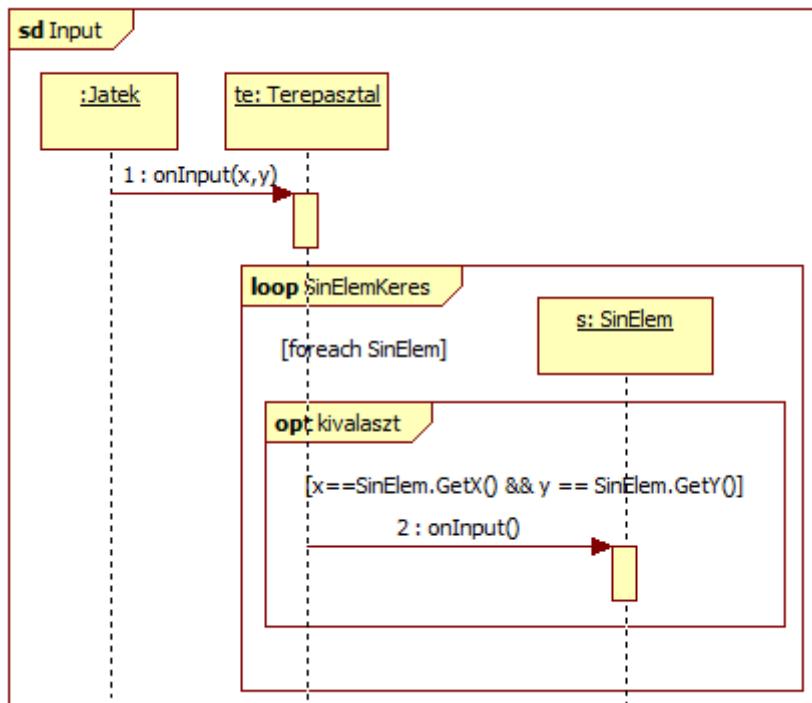
### 3.4.1 Start



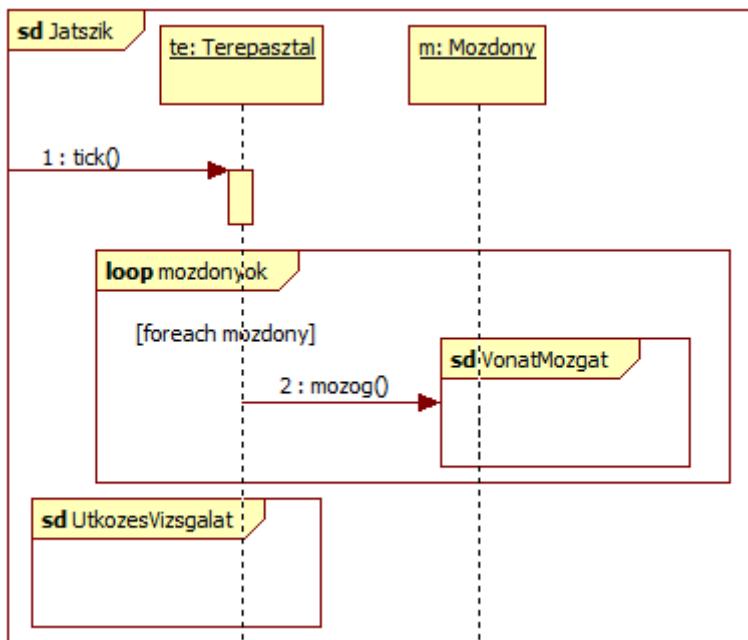
### 3.4.2 InitSzekvencia



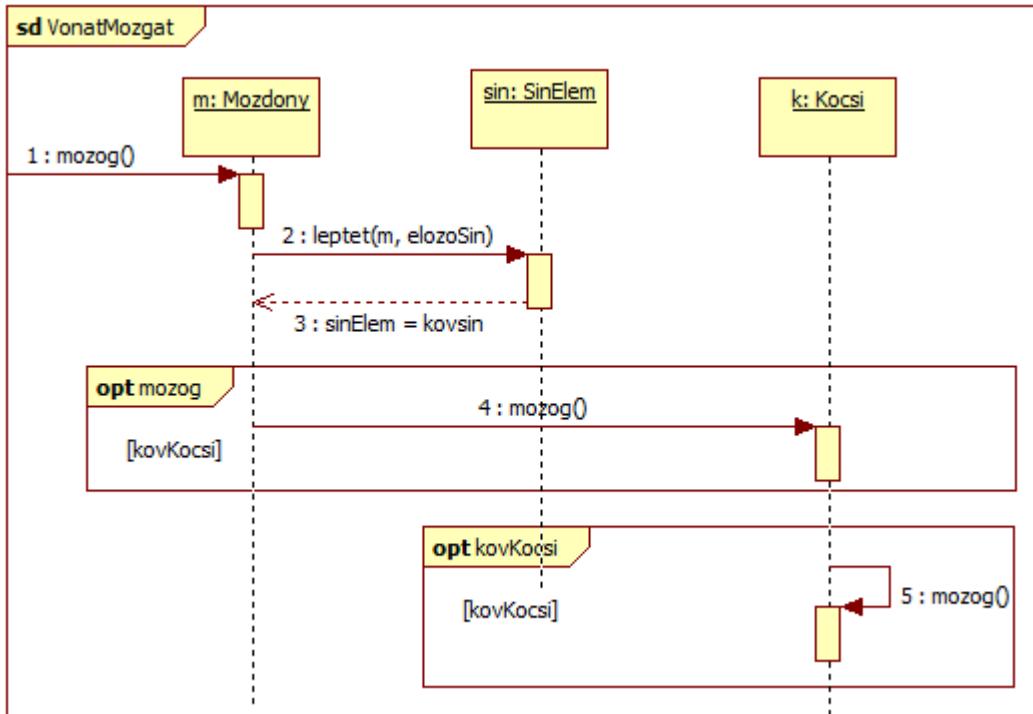
### 3.4.3 Input



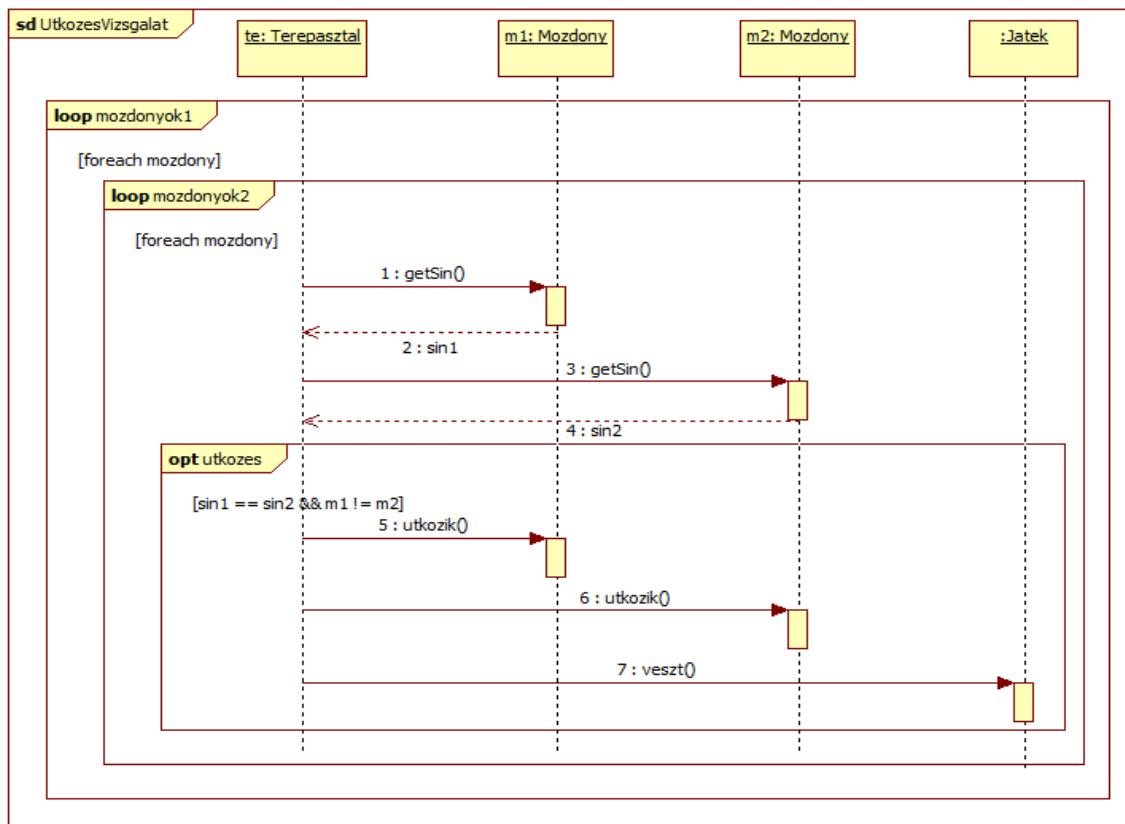
### 3.4.4 Jatszik



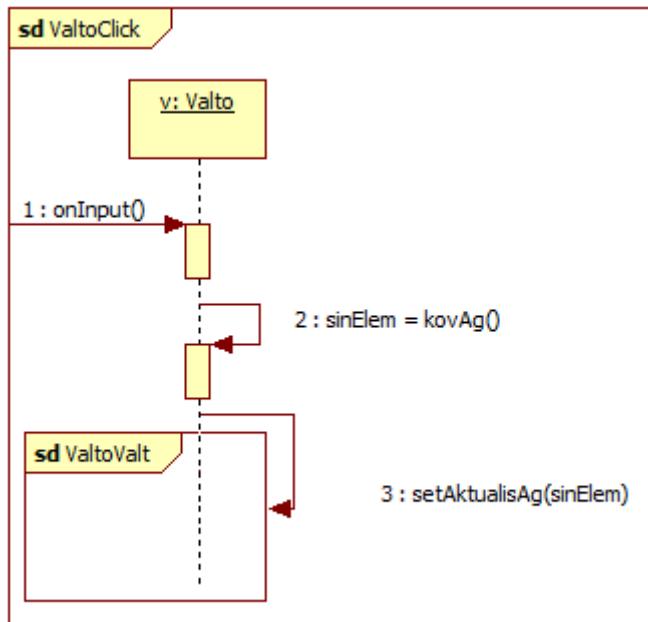
### 3.4.5 VonatMozgat



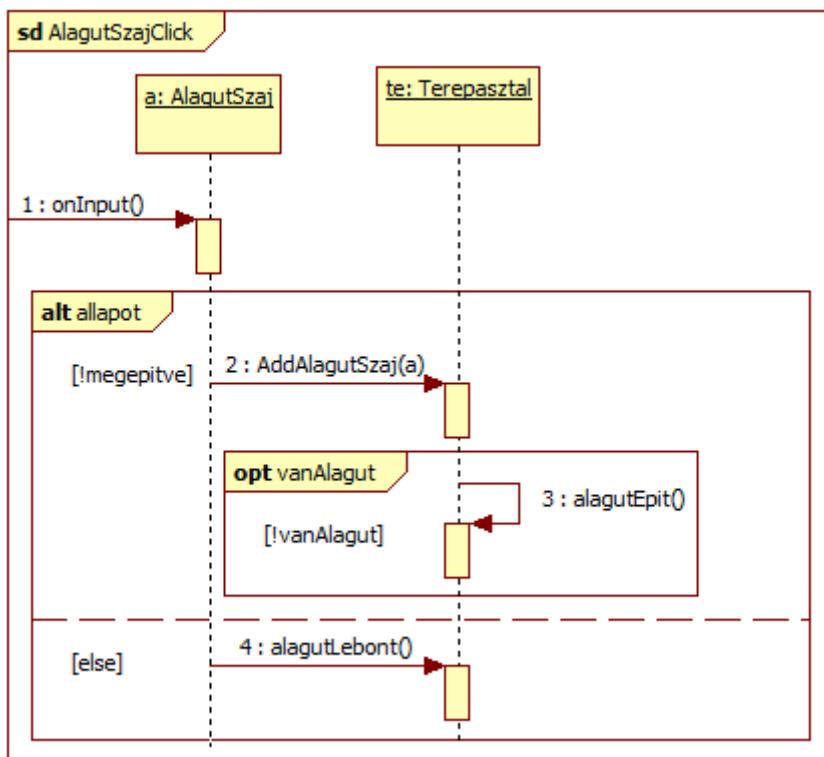
### 3.4.6 UtkozesVizsglat



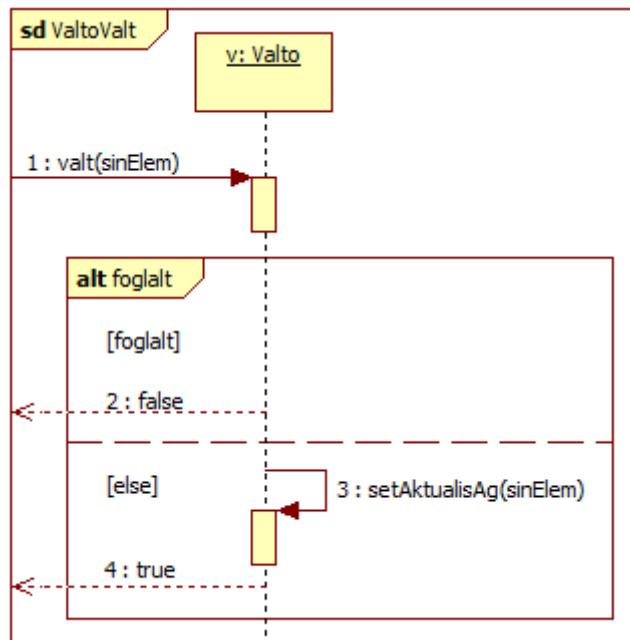
### 3.4.7 ValtoClick



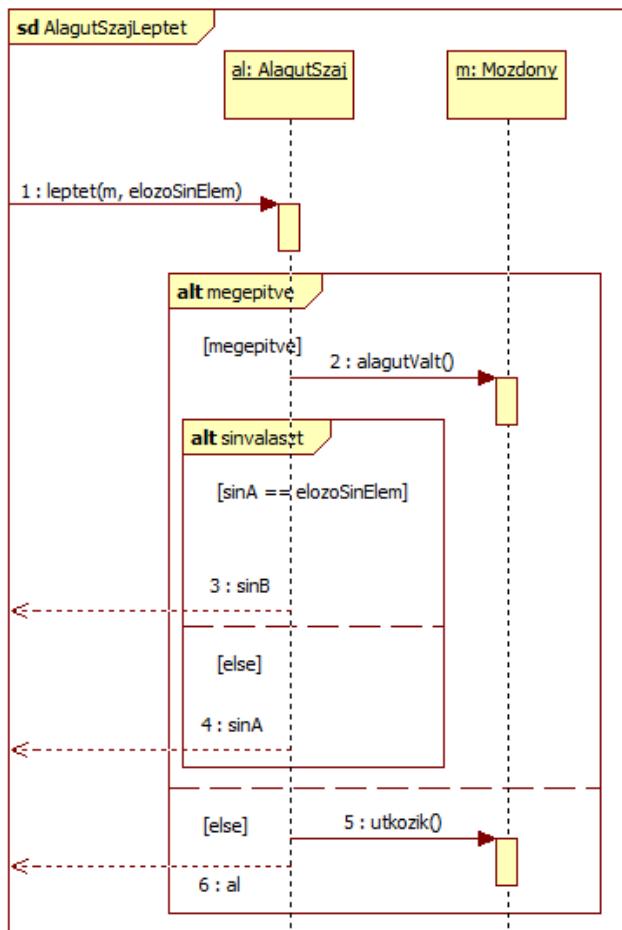
### 3.4.8 AlagutSzajClick



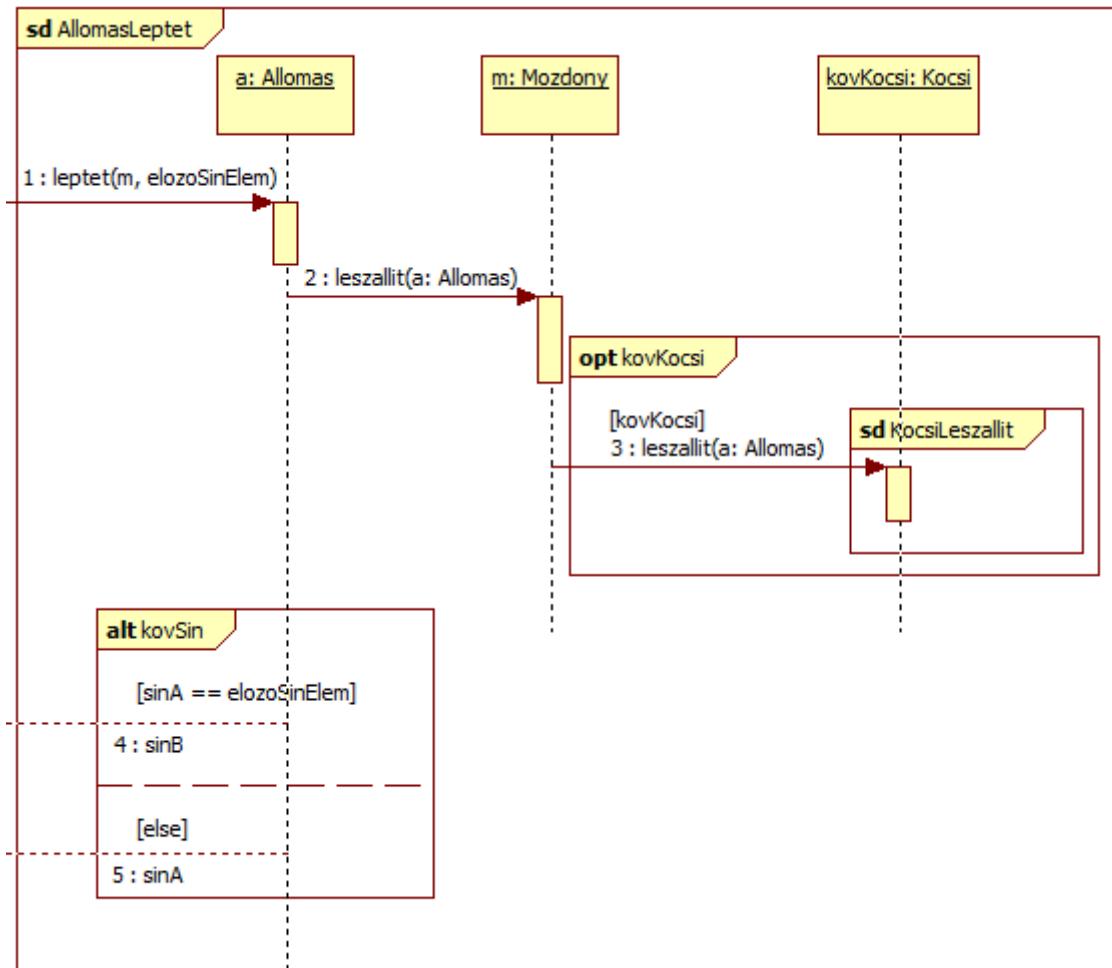
### 3.4.9 ValtoValt



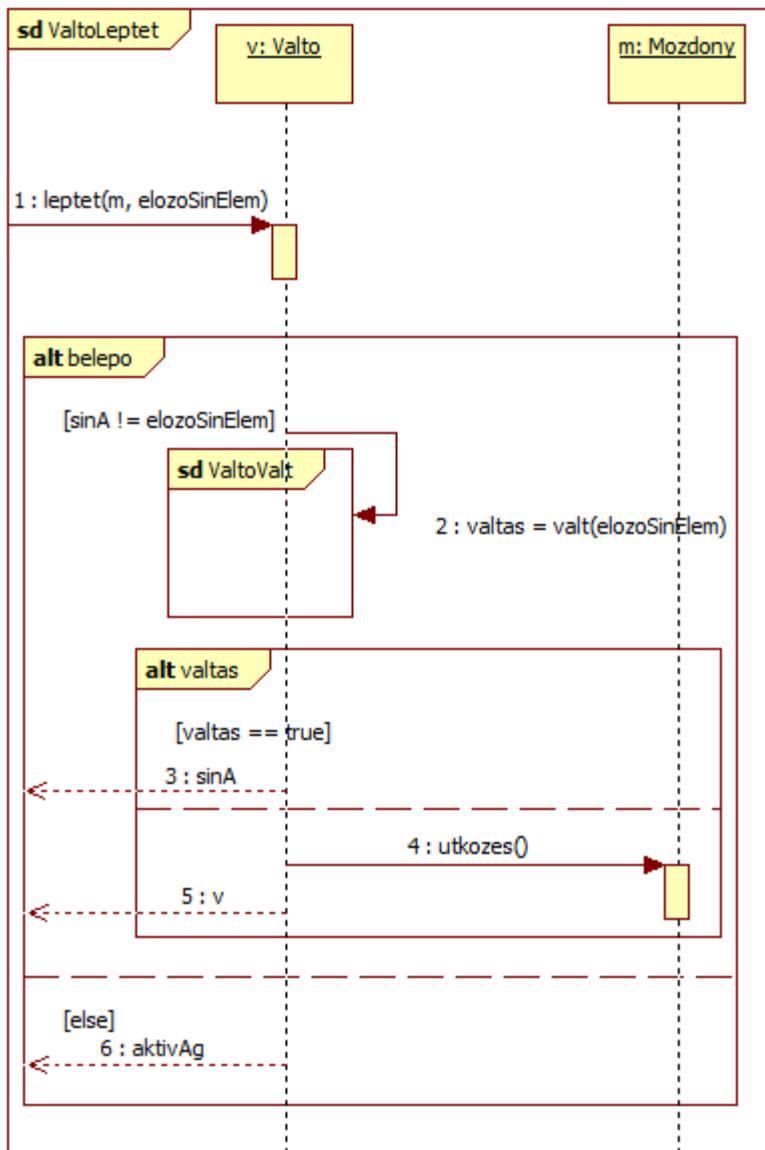
### 3.4.10 AlagutSzajLeptet



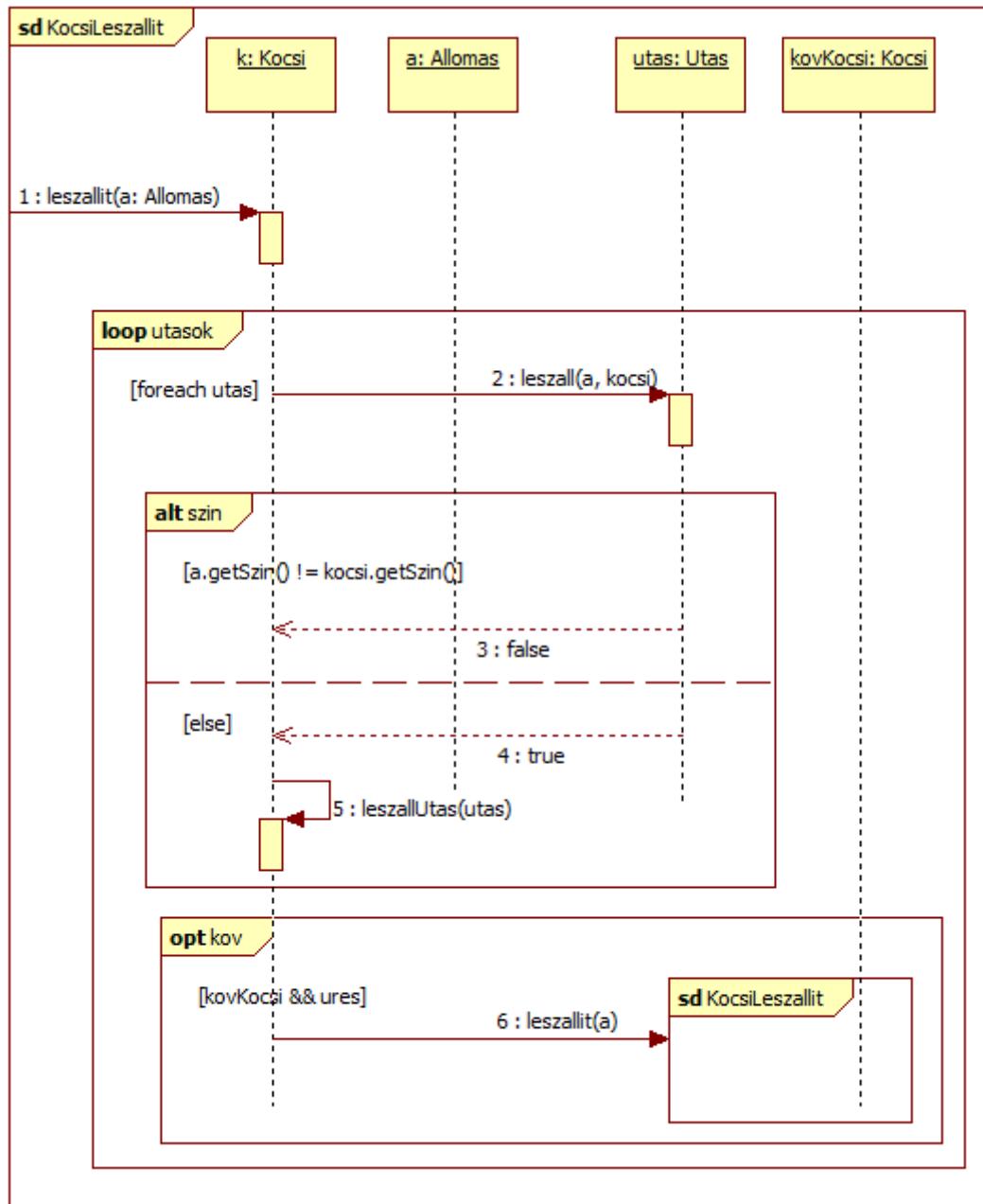
### 3.4.11 AllomasLeptet



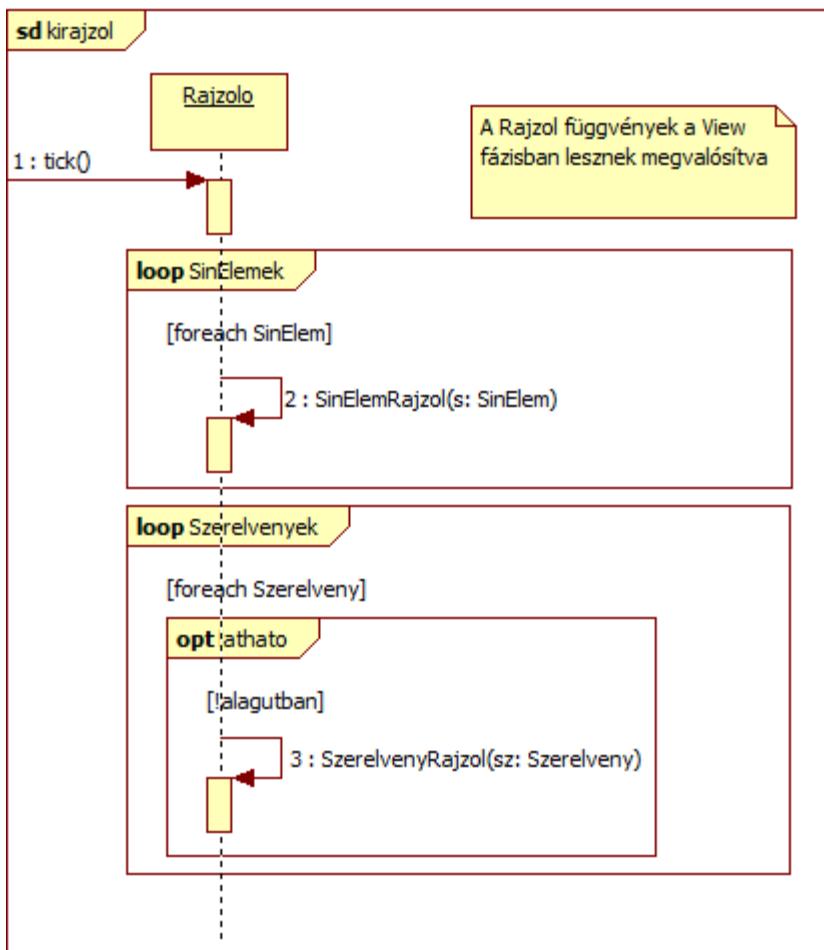
### 3.4.12 ValtoLeptet



### 3.4.13 KocsiLeszallit

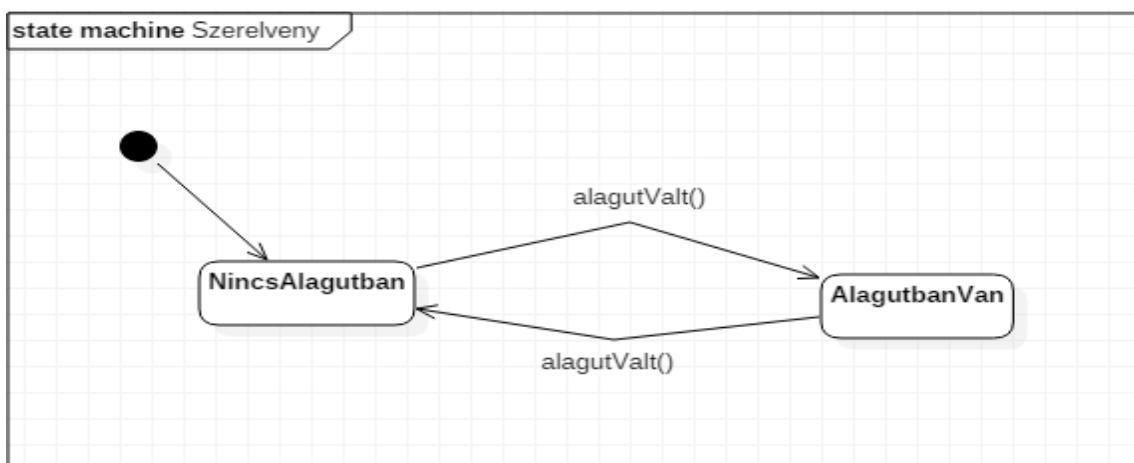


### 3.4.14 Kirajzol



## 3.5 State-chartok

### 3.5.1 Szerelveny



### 3.6 Napló

Kezdet	Időtartam	Résznevők	Leírás
2017.02.24. 16:30	4 óra	Dócs Krátky Varga	Struktúra diagram
2017.02.24. 16:30	2 óra	Szili	Objektum katalógus / osztályok
2017.02.25. 11:00	9 óra	Varga	Osztályok, osztály diagram
2017.02.25 11:00	7 óra	Dócs	Osztálydiagram, Szekvencia diagramok
2017.02.25 11:00	8 óra	Szili Sillye	Osztály- és szekvencia diagramok
2017.02.25 20:00	0.5 óra	Sillye Szili	Állapotdiagram
2017.02.25 16:00	5 óra	Krátky	Osztályok, szekvencia diagram
2017.02.26 11:30	11 óra	Krátky Sillye Szili	Szekvenciadiagramok, utómunkálatok
2017.02.26 11:30	7 óra	Dócs	git verziókezelés elsajátítása, Osztályok leírása
2017.02.26 14:00	8,5 óra	Varga	szekvencia diagram, finomítások
2017.02.26 19:00	1.5 óra	Dócs	Input és Rajzol szekvencia diagramok
2017.02.26 22:00	0,5 óra	Szili	dokumentáció formázása

## 4. Analízis modell kidolgozása 2.

### 4.1 Objektum katalógus

#### 4.1.1 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).

#### 4.1.2 Mozdony

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

#### 4.1.3 Kocsi

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

#### 4.1.4 Sín

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

#### 4.1.5 Váltó

A váltó egy olyan sín, amely  $n \geq 2$  darab szomszédot tart számon. Felelőssége, hogy a ráérkező mozdonyt továbbirányítsa a megfelelő sín ágra, továbbá tárolja, hogy éppen melyik kijárat aktív.

#### 4.1.6 Alagút Száj

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

#### 4.1.7 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.

#### 4.1.8 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.

#### 4.1.9 Timer

A szimuláció időbeli futására szolgál, hatására a mozdonyok továbbhaladnak.

Meghívja az újrarendezést.

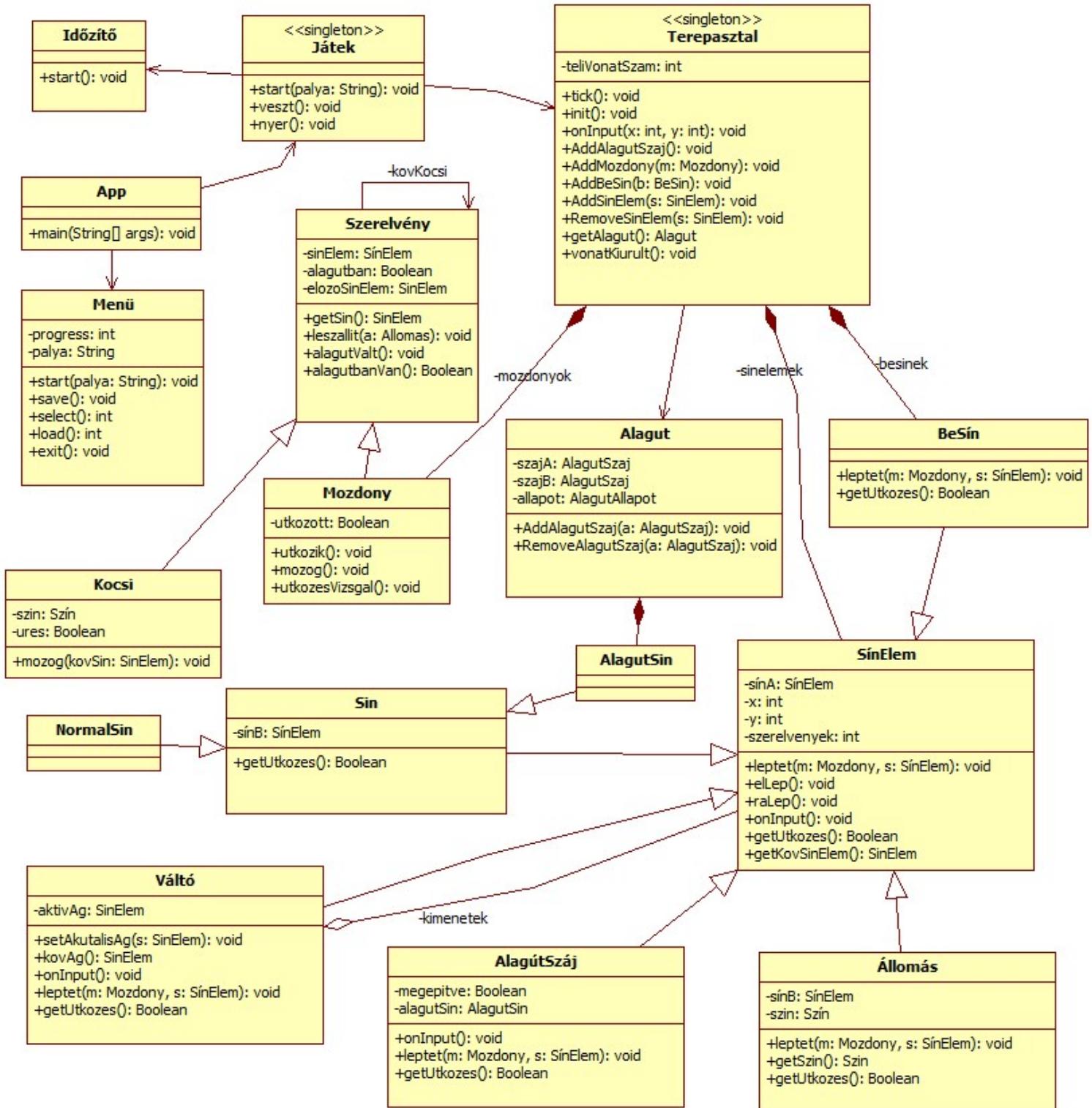
#### 4.1.10 Menü

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

#### 4.1.11 Játék

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

## 4.2 Statikus struktúra diagramok



## 4.3 Osztályok leírása

### 4.3.1 Alagut

- **Felelősség**

Alagút megépültségének, és az alagút szájak illetve a köztük menő SínElemek számítartása.

- **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

### 4.3.2 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.

### 4.3.3 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.

#### 4.3.4 Allomas

- **Felelősség**

Tárol egy Színt, melyet a ráérkező vonat lekérdezhet.

- **Ősosztályok**

SinElem -> Allomas

- **Interfészek**

Nincs.

- **Attribútumok**

- **sinB: SinElem** – a következő SinElem
- **szín: 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

#### 4.3.5 App

- **Felelősség**

Felelőssége a view, controll és modell inicializálása.

- **Ősosztályok**

Nincs.

- **Interfészek**

Nincs.

- **Attribútumok**

Nincs.

- **Metódusok**

- **void main()** - létrehozza az Idozito, Menu, Jatek objektumokat.

#### 4.3.6 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

#### 4.3.7 Idozito

- **Felelősség**

Felelőssége a periodikus jelgenerálás. A játék időbeli szimulálásának alapja.

- **Ősosztályok**

Nincs.

- **Interfészek**

Nincs.

- **Attribútumok**

Nincs.

- **Metódusok**

- **void start()** - elindítja a jelgenerálási folyamatot

#### 4.3.8 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.

#### 4.3.9 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 sínelementet, ahol tartózkodik, és mozgatja a következő kocsit (rekurzív)

**4.3.10 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.

**4.3.11 Mozdony****•Felelősség**

Kérdezgeti az alatta álló SínElementet, 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.
- **void utkozesVizsgal()** – megkérdezi a sinElemet amin áll, hogy van-e még rajta kívül más is.

#### 4.3.12 NormalSin

- **Felelősség**

Ugyanaz, mint a Sin felelőssége, megjelenik a térképen.

- **Ősosztályok**

SinElem->Sin->NormalSin

- **Interfészek**

Nincs.

- **Attribútumok**

Nincs.

- **Metódusok**

Nincs.

#### 4.3.13 Sin

- **Felelősség**

Továbbirányítja a mozdonyt a következő SínElemre.

- **Ősosztályok**

SinElem->Sin

- **Interfészek**

Nincs.

- **Attribútumok**

- **sinB: SinElem** – a másik kapcsolódó SínElem

- **Metódusok**

Nincs.

#### 4.3.14 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 sínElementet.

### 4.3.15 Szerelveny

#### • Felelősség

Számon tartja, az előző és az aktuális SínElementet, 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 SinElementtel.
- **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

### 4.3.16 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 minden egyik 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.

#### 4.3.17 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**

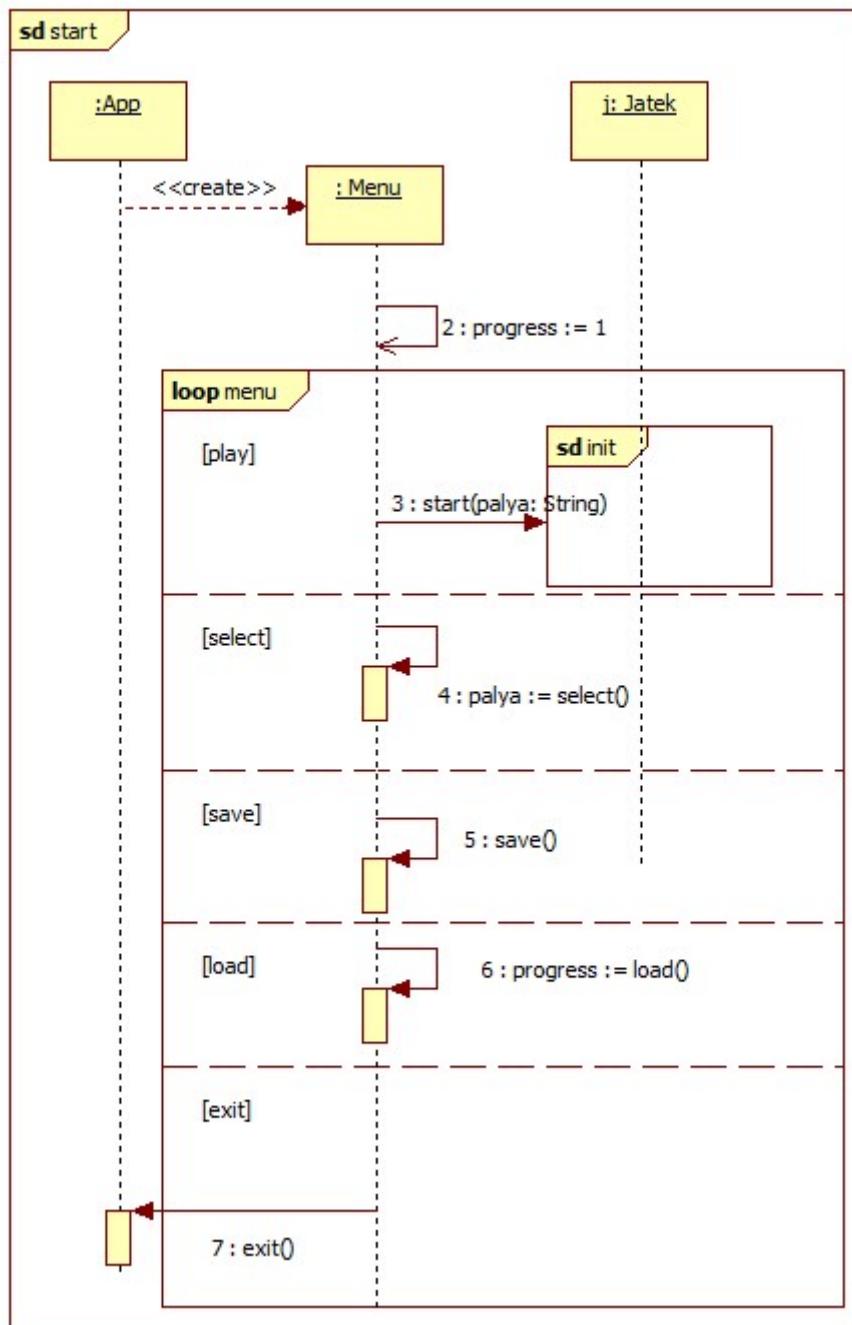
- **kimenetek: List<SinElem>** - az összes kimenő SinElem referenciajára
- **aktivAg: SinElem** - melyik ág az aktív

• **Metódusok**

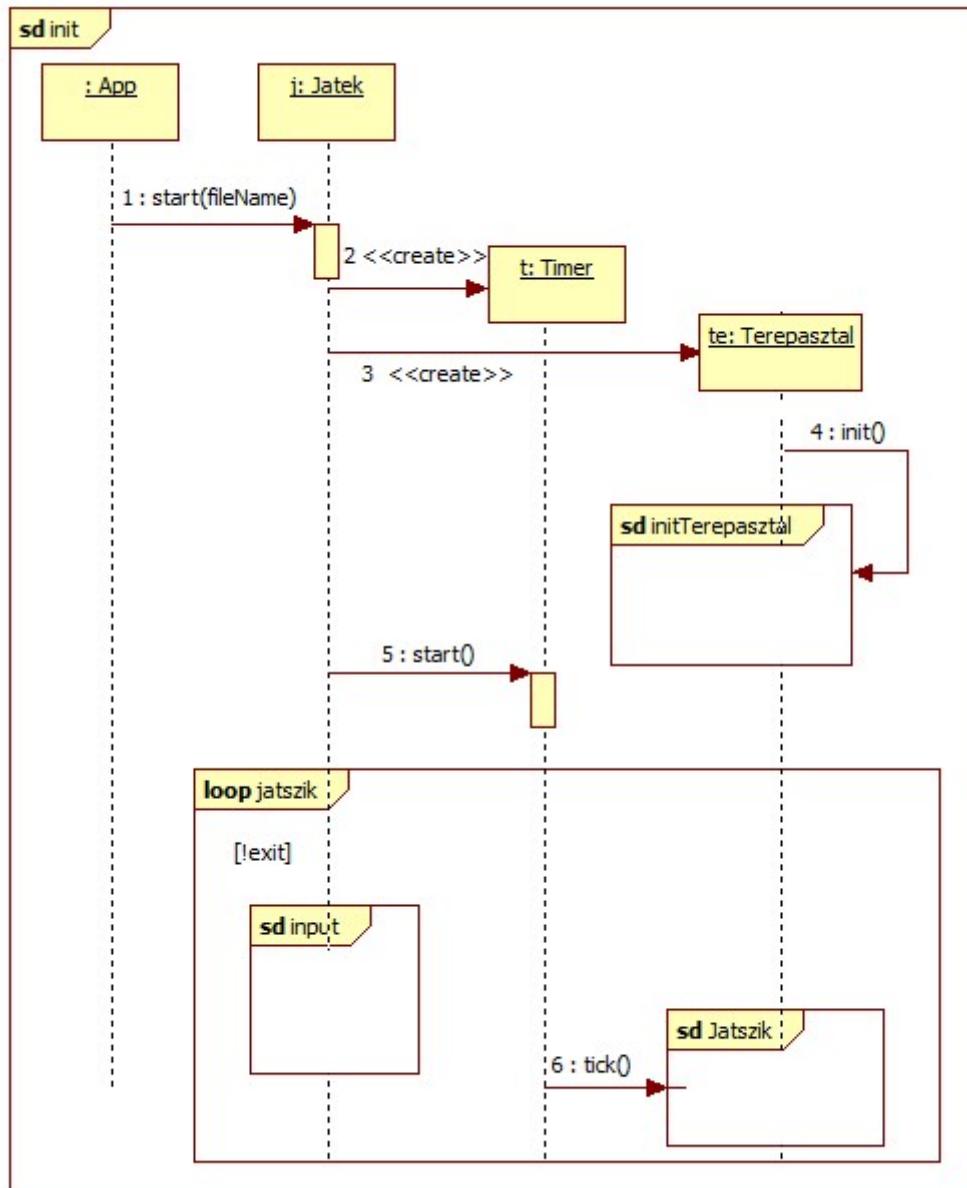
- **void setAktualisAg(Sinelem s)** - aktivAg setter
- **SinElem kovAg()** - sinek listából az aktivAg utáni SinElement adja vissza
- **void onInput()** - A bevitelre reagál az osztály.
- **void leptet(m: Mozdony, s: SinElem)** – átvált, ha nem az aktív Ágból jött a mozdony

## 4.4 Szekvencia diagramok

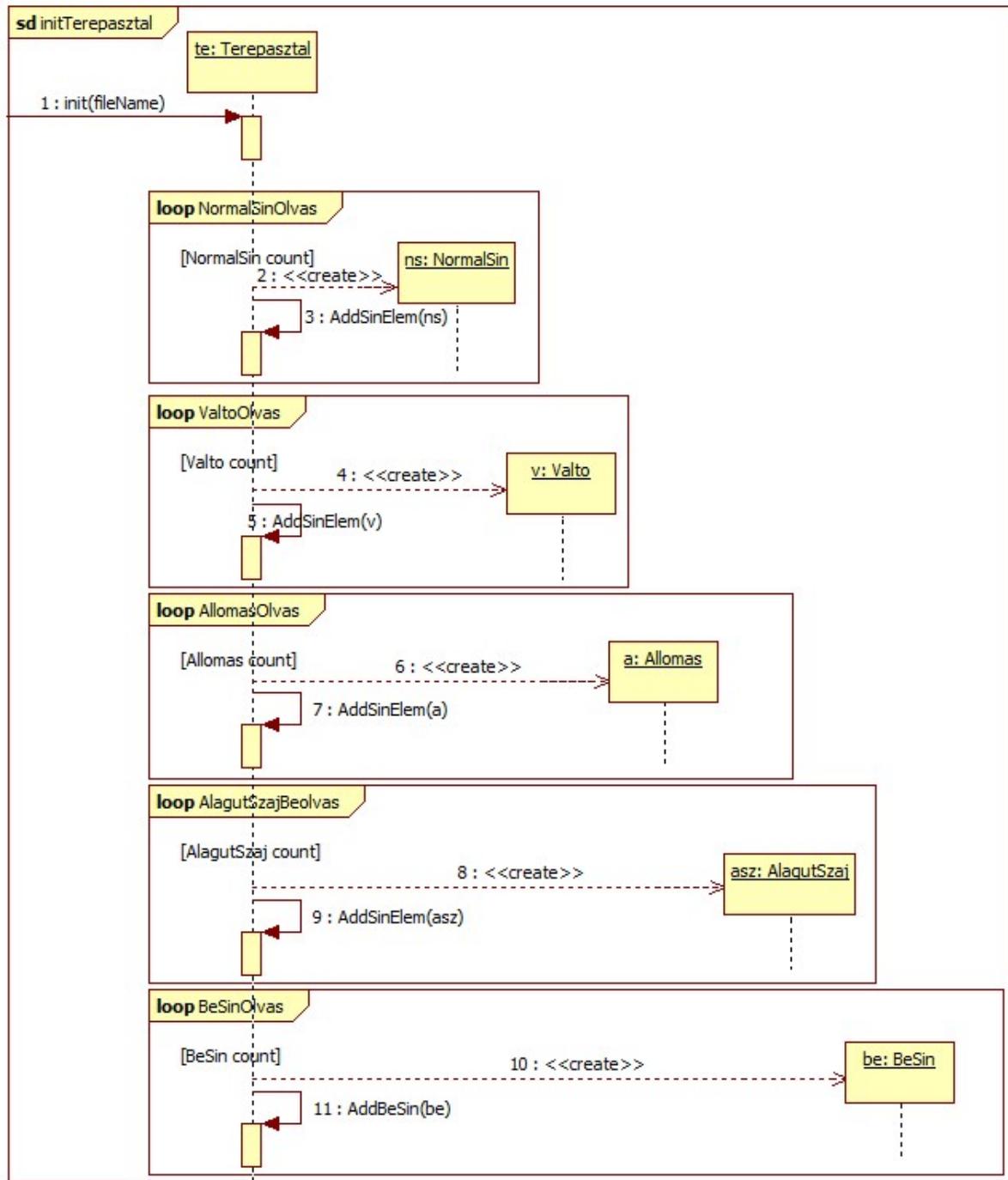
### 4.4.1 Start



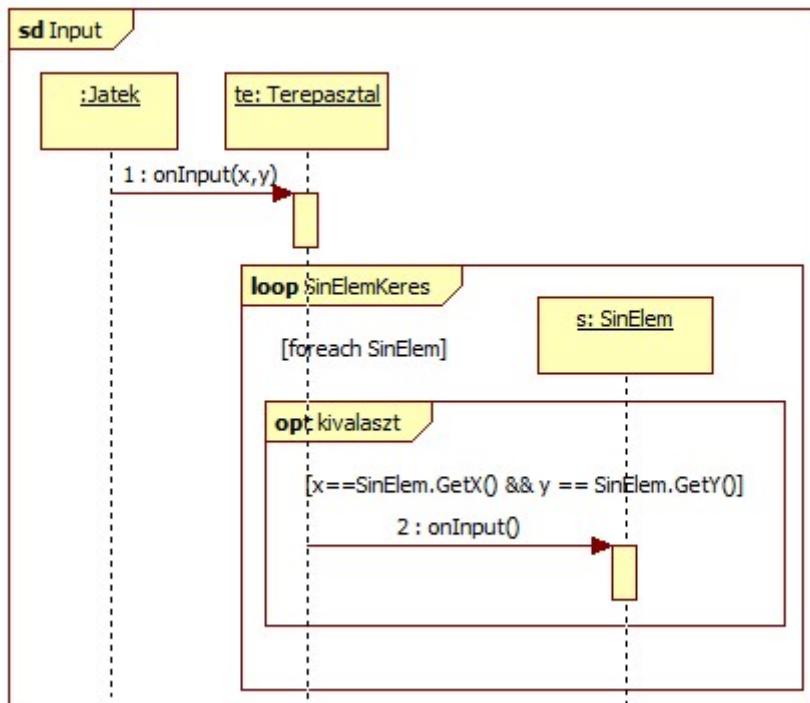
## 4.4.2 InitSzekvencia



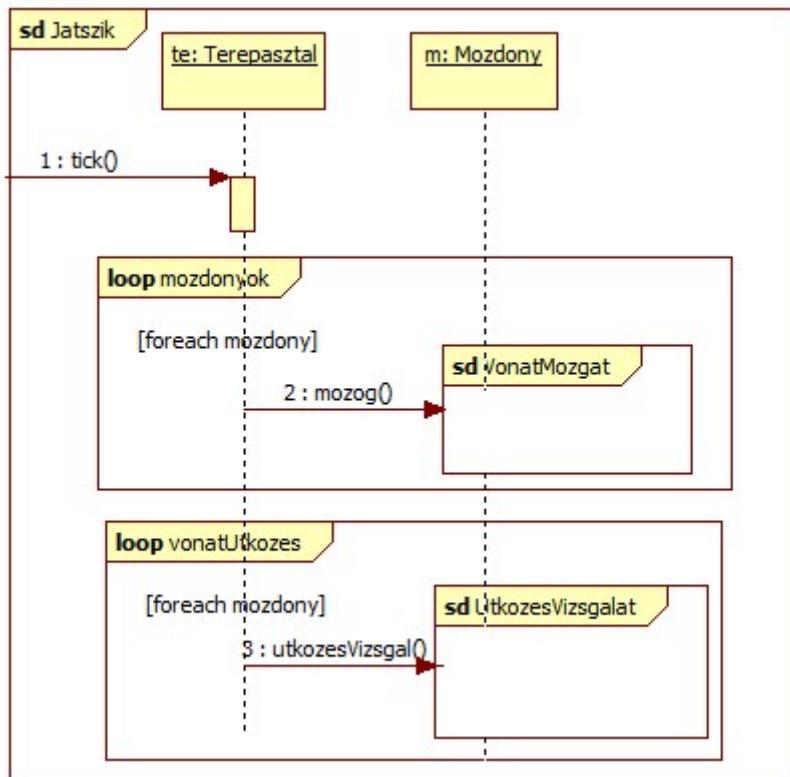
### 4.4.3 initTerepasztal



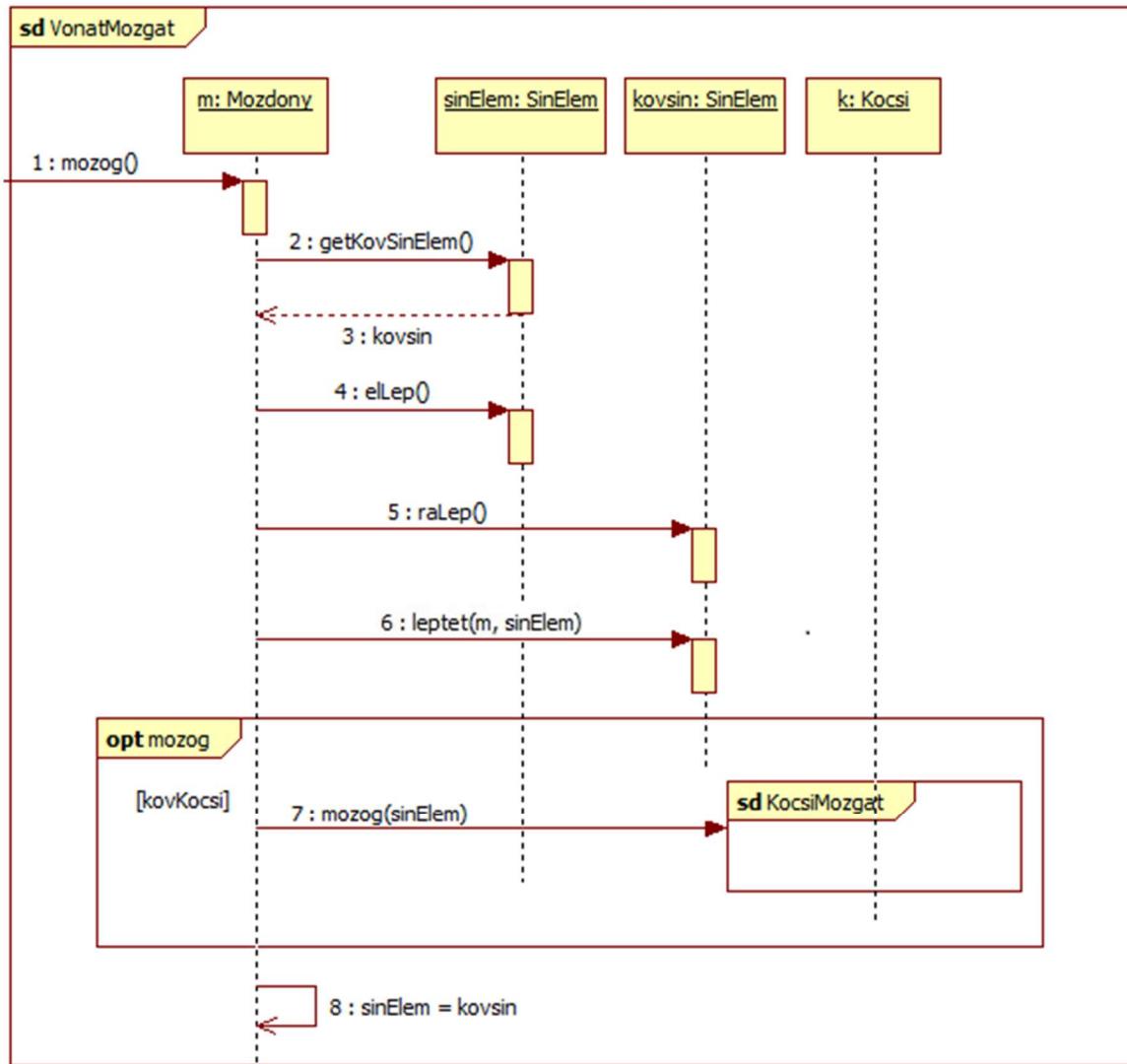
#### 4.4.4 Input



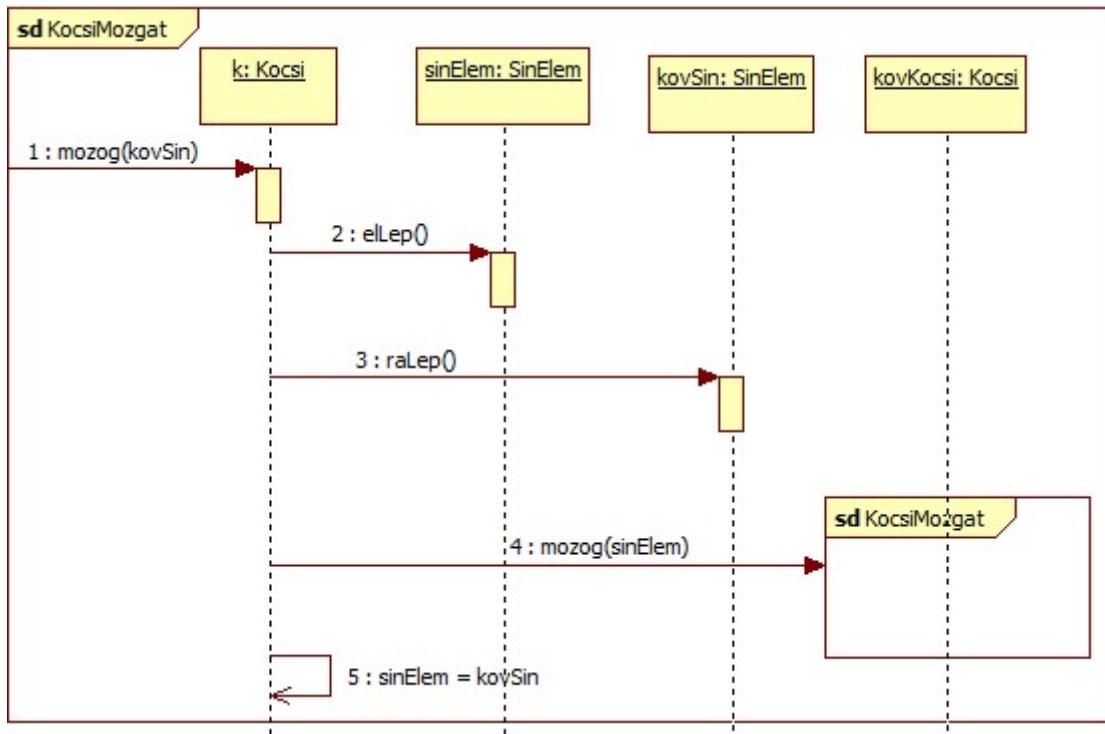
#### 4.4.5 Jatszik



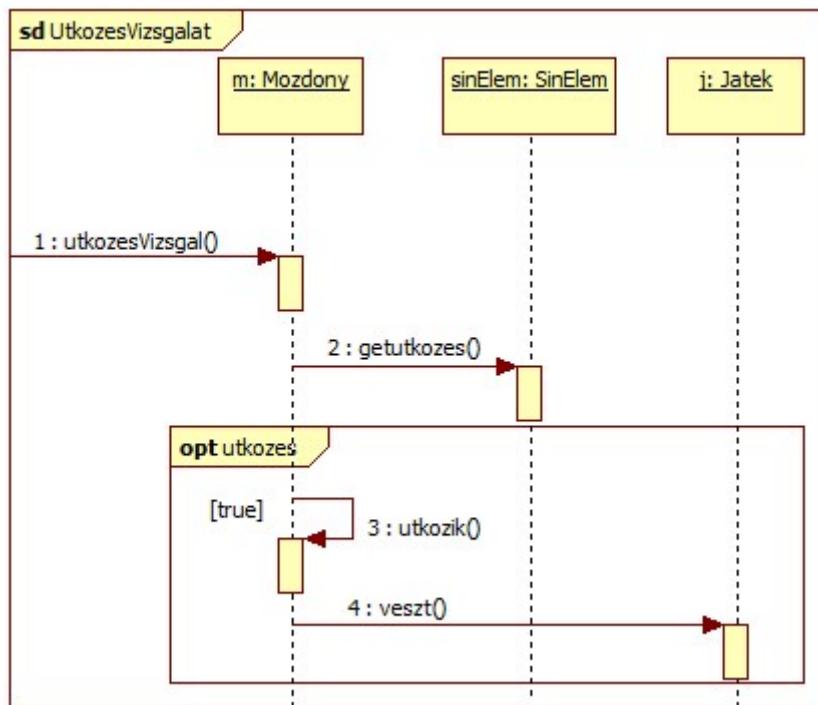
#### 4.4.6 VonatMozgat



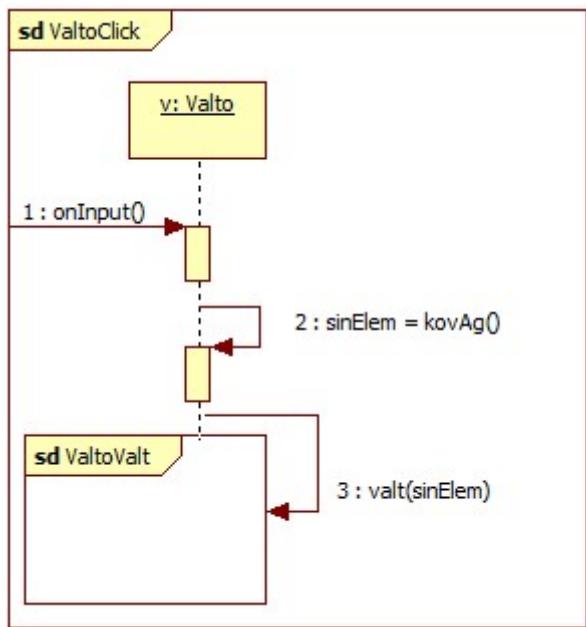
#### 4.4.7 KocsiMozgat



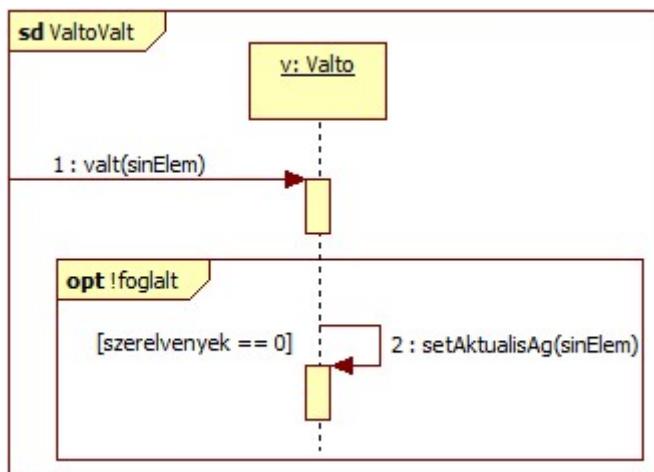
#### 4.4.8 UtkozesVizsglat



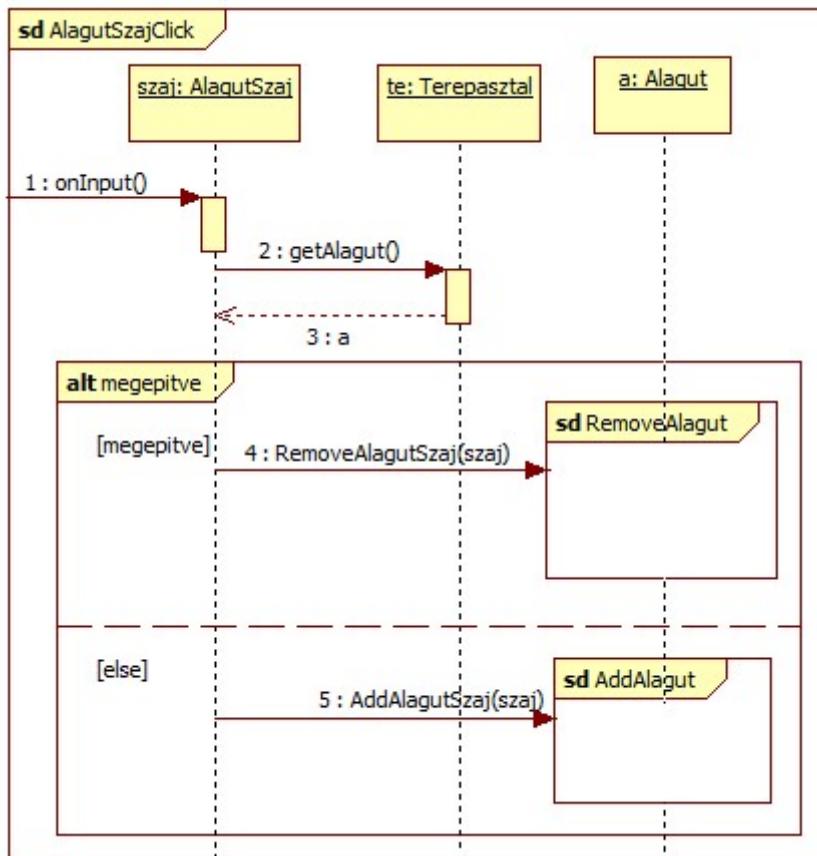
#### 4.4.9 ValtoClick



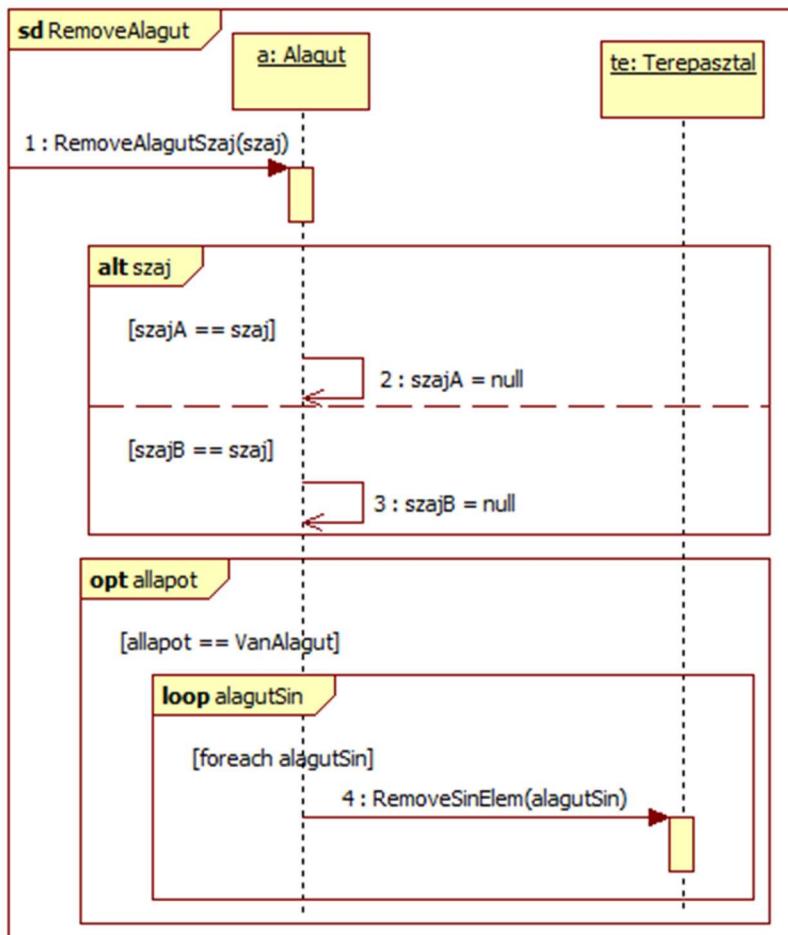
#### 4.4.10 ValtoVolt



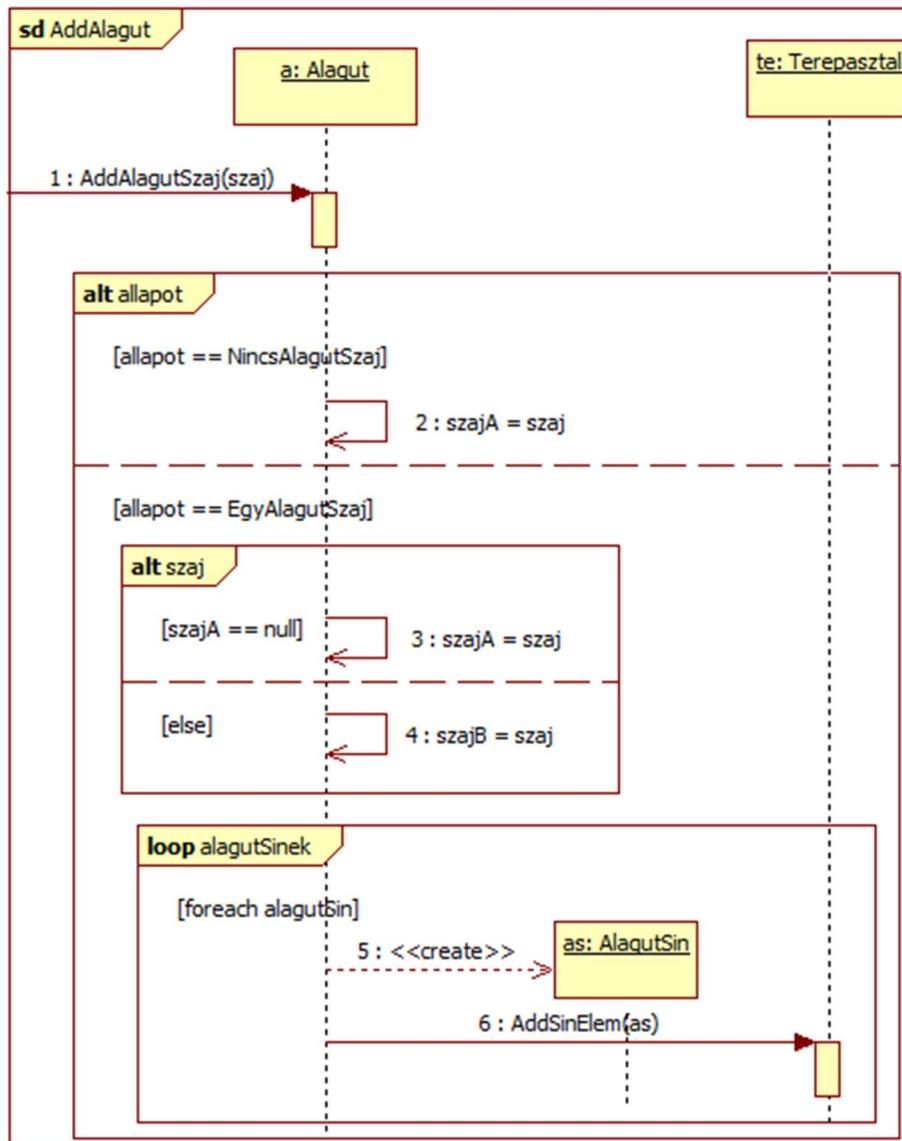
#### 4.4.11 AlagutSzajClick



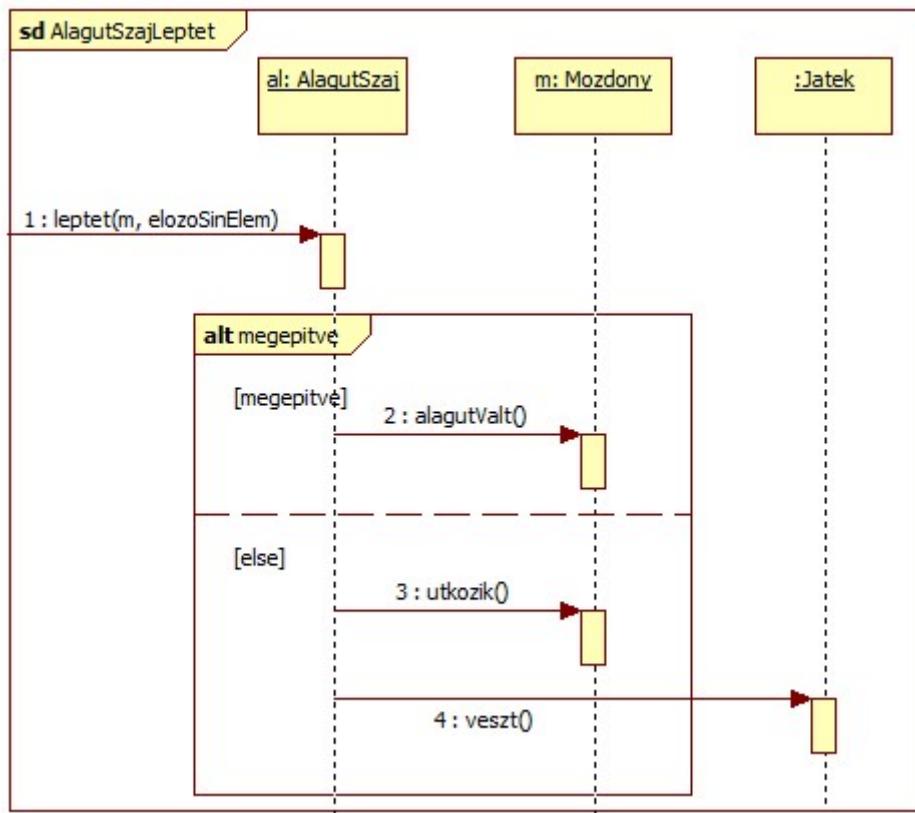
#### 4.4.12 RemoveAlagut



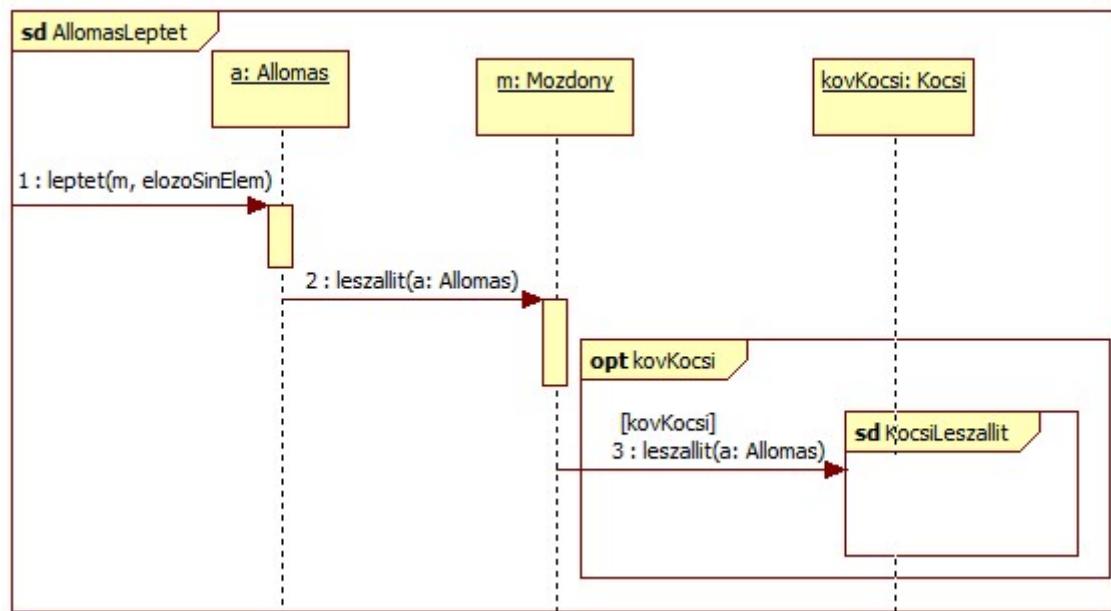
#### 4.4.13 AddAlagut



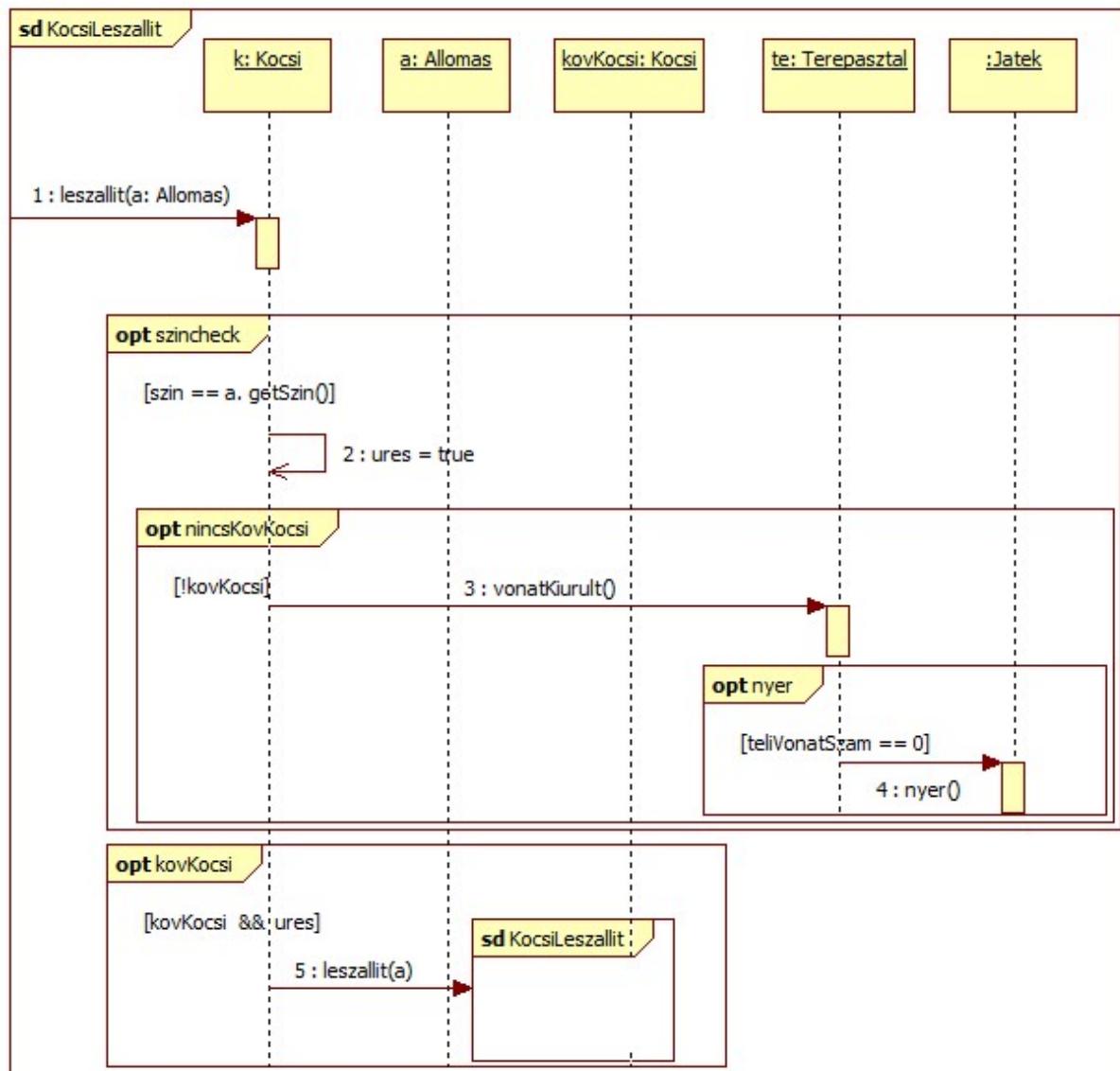
#### 4.4.14 AlagutSzajLeptet



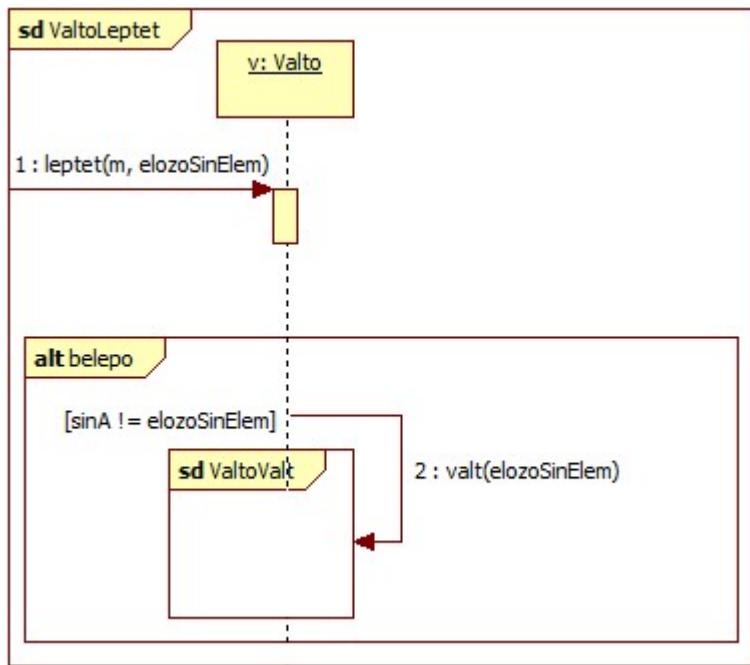
#### 4.4.15 AllomasLeptet



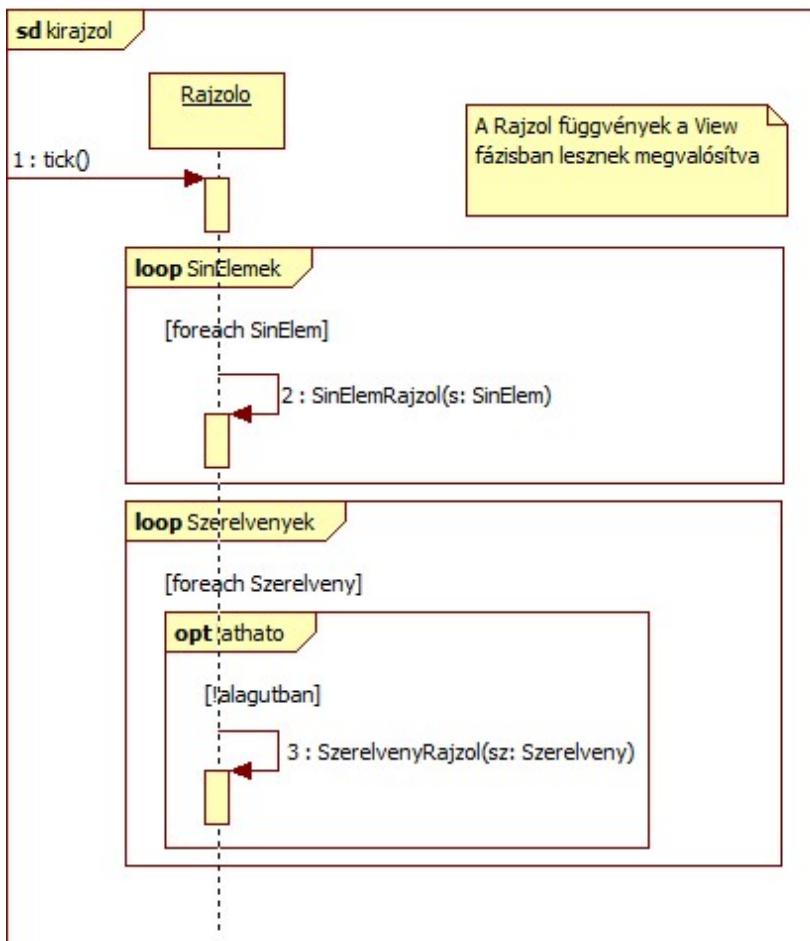
#### 4.4.16 KocsiLeszallit



#### 4.4.17 ValtoLeptet

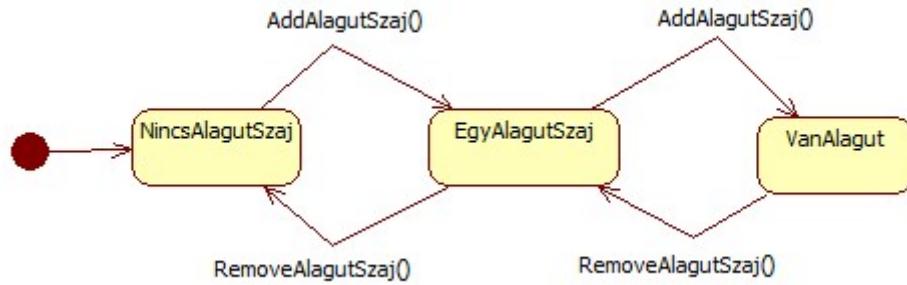


#### 4.4.18 Kirajzol

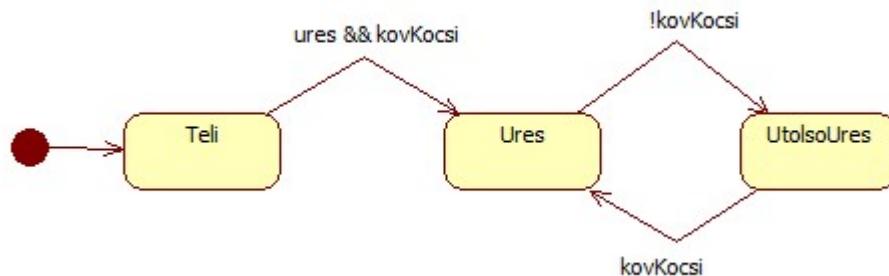


## 4.5 State-chartok

### 4.5.1 AlagutEpit state-chart



### 4.5.2 Kocsi állapot state-chart



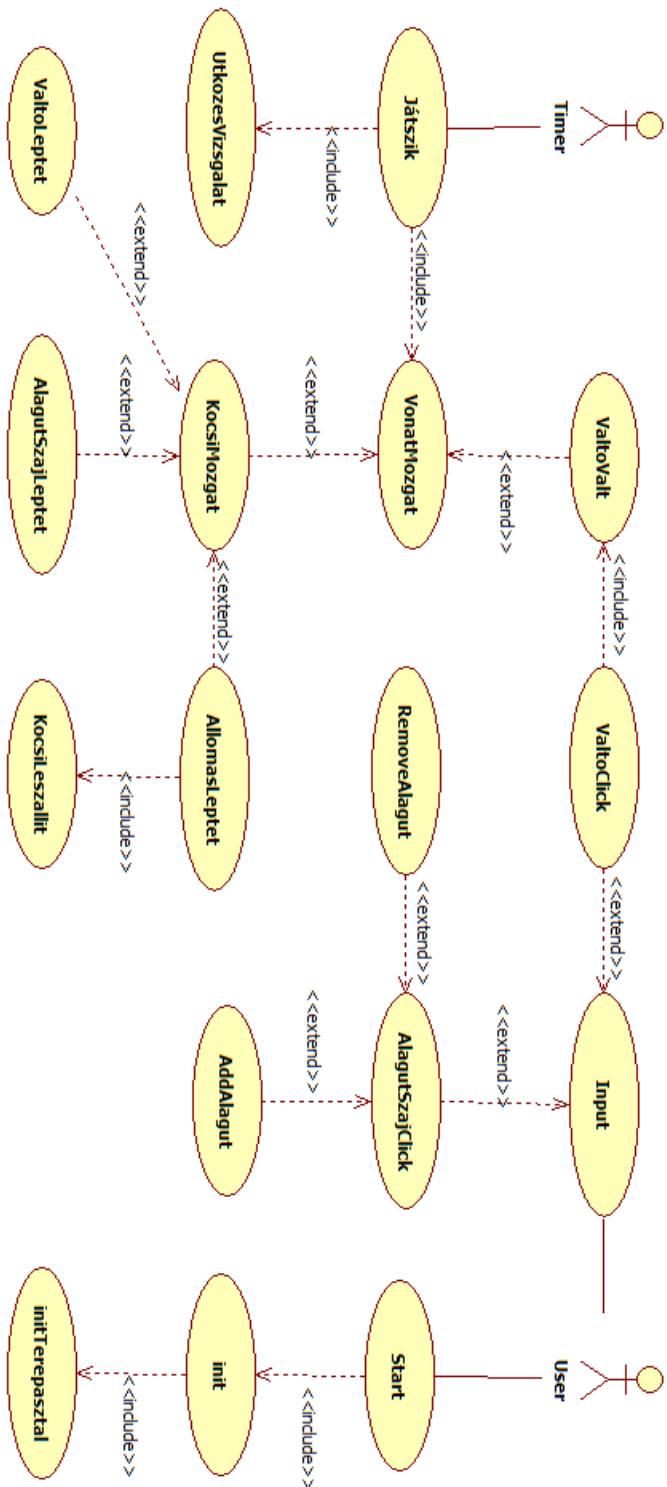
## 4.6 Napló

Kezdet	Időtartam	Részttvevők	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 átgondolása 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
2017.03.06 00:15	0.5 óra	Krátky	doksi áttekintése, módosítások

## 5. Szkeleton tervezése

### 5.1 A szkeleton modell valóságos use-case-ai

#### 5.1.1 Use-case diagram



### 5.1.2 Use-case leírások

<b>Use-case neve</b>	Start
<b>Rövid leírás</b>	Menü inicializálása.
<b>Aktorok</b>	User
<b>Forgatókönyv</b>	A felhasználó elindítja a játékot, kiválaszt egy lehetőséget a menüben.

<b>Use-case neve</b>	init
<b>Rövid leírás</b>	Új játék: Timer és Terepasztal létrehozása
<b>Aktorok</b>	User
<b>Forgatókönyv</b>	A felhasználó kiválasztotta az új játék menüpontot. Létrejön a Trepasztal és elindul a Timer.

<b>Use-case neve</b>	InitTerepasztal
<b>Rövid leírás</b>	Fájlból töltődik fel a Terepasztal SinElemekkel.
<b>Aktorok</b>	User
<b>Forgatókönyv</b>	Kapott fájlnévből egyenként minden egyes SinElem típusát betölt.

<b>Use-case neve</b>	Input
<b>Rövid leírás</b>	A játékos kattint a pályán.
<b>Aktorok</b>	User
<b>Forgatókönyv</b>	Minden kattintás az adott SinElem típusától függően vált ki hatást.

<b>Use-case neve</b>	Jatszik
<b>Rövid leírás</b>	Beüt az időzítő, mozognak és ütköznek a vonatok.
<b>Aktorok</b>	Timer
<b>Forgatókönyv</b>	Minden beütésnél szól a Timer az összes Mozdonynak. Ezután ütközésvizsgálat.

<b>Use-case neve</b>	VonatMozgat
<b>Rövid leírás</b>	Szól a Timer a Mozdynak, hogy mozogjon tovább. Mozdony szól következő Kocsinak, hogy mozogjon.
<b>Aktorok</b>	Timer
<b>Forgatókönyv</b>	Timer beüt, Mozdony lekéri a SinElemétől a következő SinElementet. Ezután a következő Koci is mozog.

<b>Use-case neve</b>	KocsiMozgat
<b>Rövid leírás</b>	A Kocsi következő SinElemre lép, és Kocsi mozgat.
<b>Aktorok</b>	Timer
<b>Forgatókönyv</b>	Megkapja az előző Szerelvénytől a SinElementet, amire rálép, majd szól a következő Kocsinak, hogy mozogjon.

<b>Use-case neve</b>	UtkozesVizsgalat
<b>Rövid leírás</b>	Terepasztal megvizsgálja, hogy volt-e ütközés
<b>Aktorok</b>	Timer
<b>Forgatókönyv</b>	A mozdonyok megnézik, hogy ütköztek-e. Megkérdezik a sinElem-eket, amin állnak, hogy áll-e más rajta, ha igen, ütköznek, és ez alapján jeleznek a programnak.

<b>Use-case neve</b>	ValtoClick
<b>Rövid leírás</b>	Váltóra kattint a Felhasználó.
<b>Aktorok</b>	User
<b>Forgatókönyv</b>	A Felhasználó rákattint egy Váltóra.

<b>Use-case neve</b>	ValtoValt
<b>Rövid leírás</b>	Váltó átvált.
<b>Aktorok</b>	User
<b>Forgatókönyv</b>	Ha nem foglalt, akkor a Váltó átalítja az AktuálisÁg-at.

<b>Use-case neve</b>	AlagutSzajClick
<b>Rövid leírás</b>	AlagutSzajra kattint a Felhasználó, ami erre megépül vagy lerombolódik.
<b>Aktorok</b>	User
<b>Forgatókönyv</b>	A Felhasználó rákattint egy AlagutSzajra. Ha meg volt építve, akkor lerombolódik. Ha üres volt, akkor megépül.

<b>Use-case neve</b>	RemoveAlagut
<b>Rövid leírás</b>	Lerombolódik az AlagutSzaj. Az Alagut felbomlik, a benne lévő SinElemek eltűnnek.
<b>Aktorok</b>	User
<b>Forgatókönyv</b>	A Felhasználó rákattintott egy megépült AlagutSzajra. Ez lerombolódik, és ha van Alagut, akkor az összes benne lévő SinElem is törlődik.

<b>Use-case neve</b>	AddAlagut
<b>Rövid leírás</b>	Megépül az AlagutSzaj. Ha ez a második, akkor megépül az Alagut.
<b>Aktorok</b>	User
<b>Forgatókönyv</b>	Felhasználó rákattint egy nem-megépült AlagutSzajra. Megépül, és ha az Alagut másik szája már meg van épülve, akkor megépül az Alagut az összes SinElemével.

<b>Use-case neve</b>	AlagutSzajLeptet
<b>Rövid leírás</b>	Mozdony AlagutSzajra lépett. Eltűnik szem elől vagy ütközik, Alaguttól függően.
<b>Aktorok</b>	Timer
<b>Forgatókönyv</b>	Egy Mozdony rálép egy AlagutSzajra. Ha meg van építve, akkor bemegy az Alagutba, különben ütközik és a játékos veszt.

<b>Use-case neve</b>	AllomasLeptet
<b>Rövid leírás</b>	Mozdony Allomasra lépett. Szól az első Kocsinak, hogy szállítsa le utasait, ha kell.
<b>Aktorok</b>	Timer
<b>Forgatókönyv</b>	Egy Mozdony rálép egy Allomasra. Mozdony szól a mögötte lévő Kocsinak (ha van), hogy szállítsa le utasait, ha kell.

<b>Use-case neve</b>	KocsiLeszallit
<b>Rövid leírás</b>	Előző Szerelveny szól, hogy Allomason áll. Színek alapján eldől, hogy az utasok leszállnak-e, és jelez a Terepasztalnak. Ha üres, akkor szól a következő Kocsinak is.
<b>Aktorok</b>	Timer
<b>Forgatókönyv</b>	A Kocsi megnézi az Állomás színét, ha az megegyezik a saját színével, akkor a Kocsi kiürül. Ha nincs következő kocsi akkor szól a Terepasztalnak, hogy a Vonat kiürült. Ezután a Terepasztal megnézi hogy van-e még teli kocsi, ha nincs akkor a Játékos nyer. Végül szól a következő Kocsinak is, hogy szállítson le.

<b>Use-case neve</b>	ValtoLeptet
<b>Rövid leírás</b>	Mozdony Valtora lépett. Átvált, ha nem-aktív kimenő ágról jött.
<b>Aktorok</b>	Timer
<b>Forgatókönyv</b>	Rálép a Mozdony egy Valtora, ami megvizsgálja, hogy honnan jött, és átvált automatikusan, ha nem-aktív kimenő ágról jött.

## 5.2 A szkeleton kezelői felületének terve, dialógusok

A szkeletonban konzolos felhasználói felület segítségével fog történni a felhasználói interakció. Különböző utasítások segítségével lesz elérhető a szekvenciák lefutása, melyek paraméterek megadásával igény szerint tovább finomíthatóak. Habár grafikus megjelenítés még nem lesz implementálva, de a konzolon meg fognak jelenni a szekvenciák végrehajtása közben keletkező adatok. Ezek például: A meghívott funkció neve, paraméterei, visszatérési értéke, egyéb információk stb.

Példa:

A Felhasználó az UtkozesVizsgalat szekvencia futását szeretné vizsgálni.

Konzolba begépeli az ehhez tartozó utasítást: **m1.utkozesVizsgal()**

Erre megjelennek a következő adatok:

**getUtkozes() - true**

**utkozik() - void**

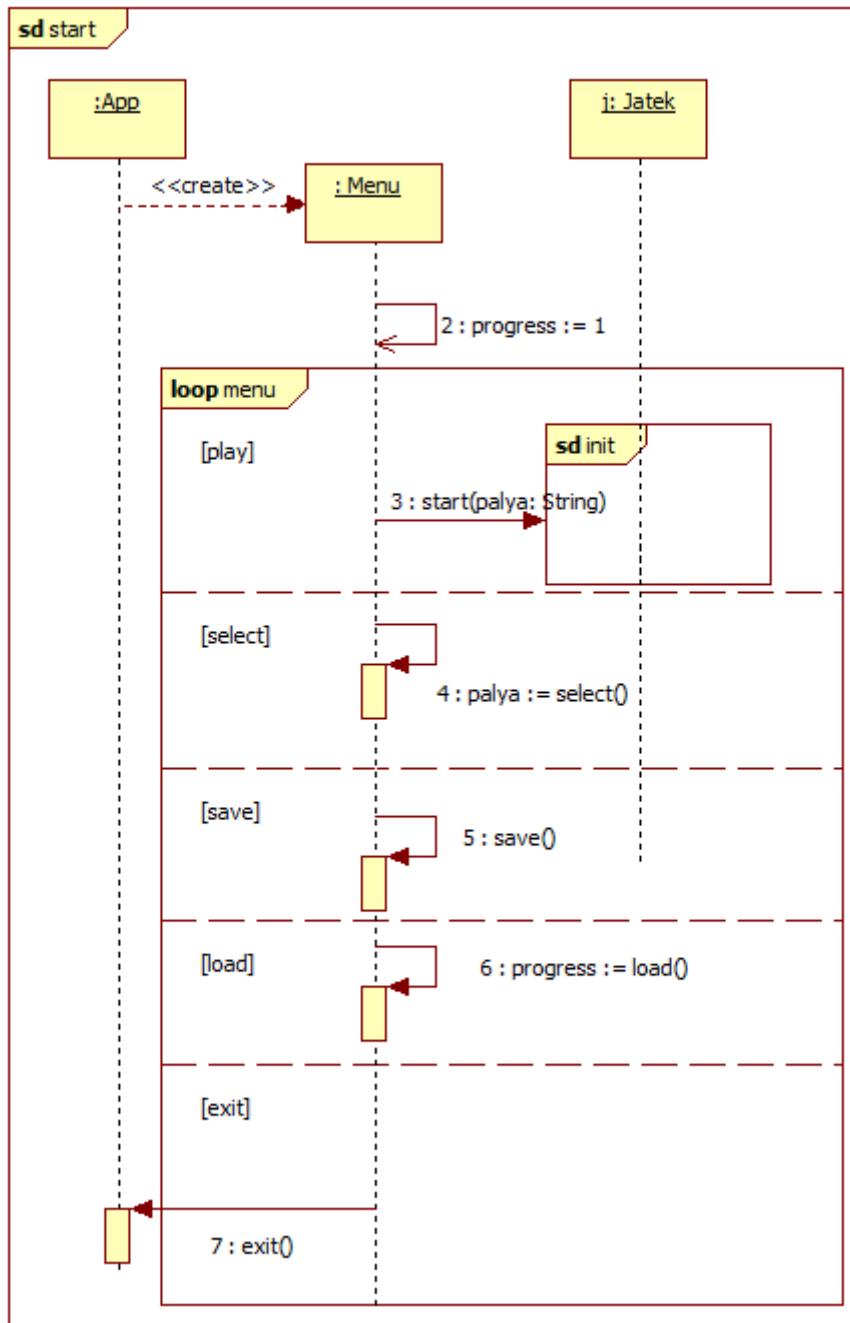
**veszt() - void**

Látható tehát, hogy a szkeletonnal nyomon követhetőek a szekvenciákban résztvevő függvényhívások, és azok visszatérési értékei.

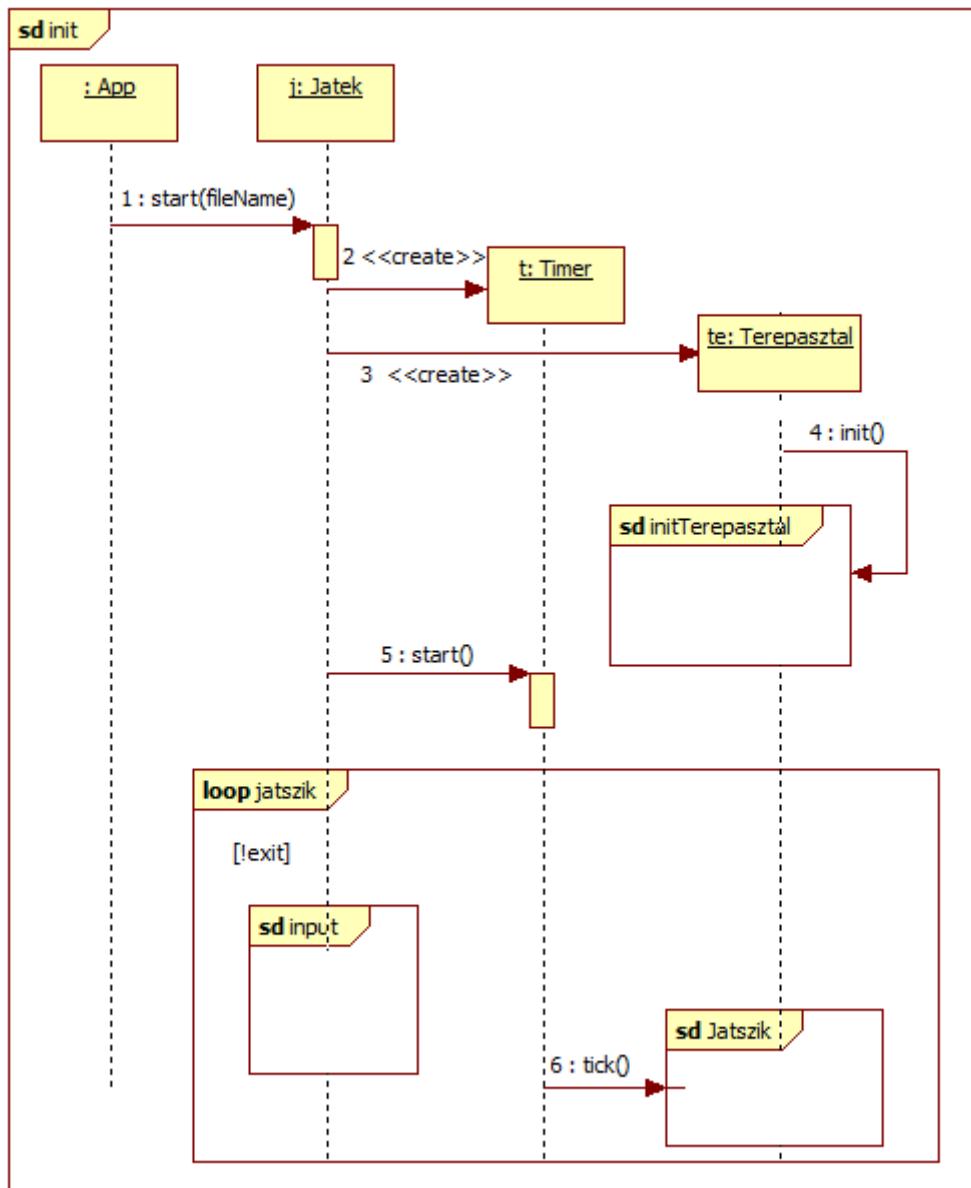
A szkeleton utasításaival kapcsolatos bővebb leírások a 6. pontban találhatók.

## 5.3 Szekvencia diagramok a belső működésre

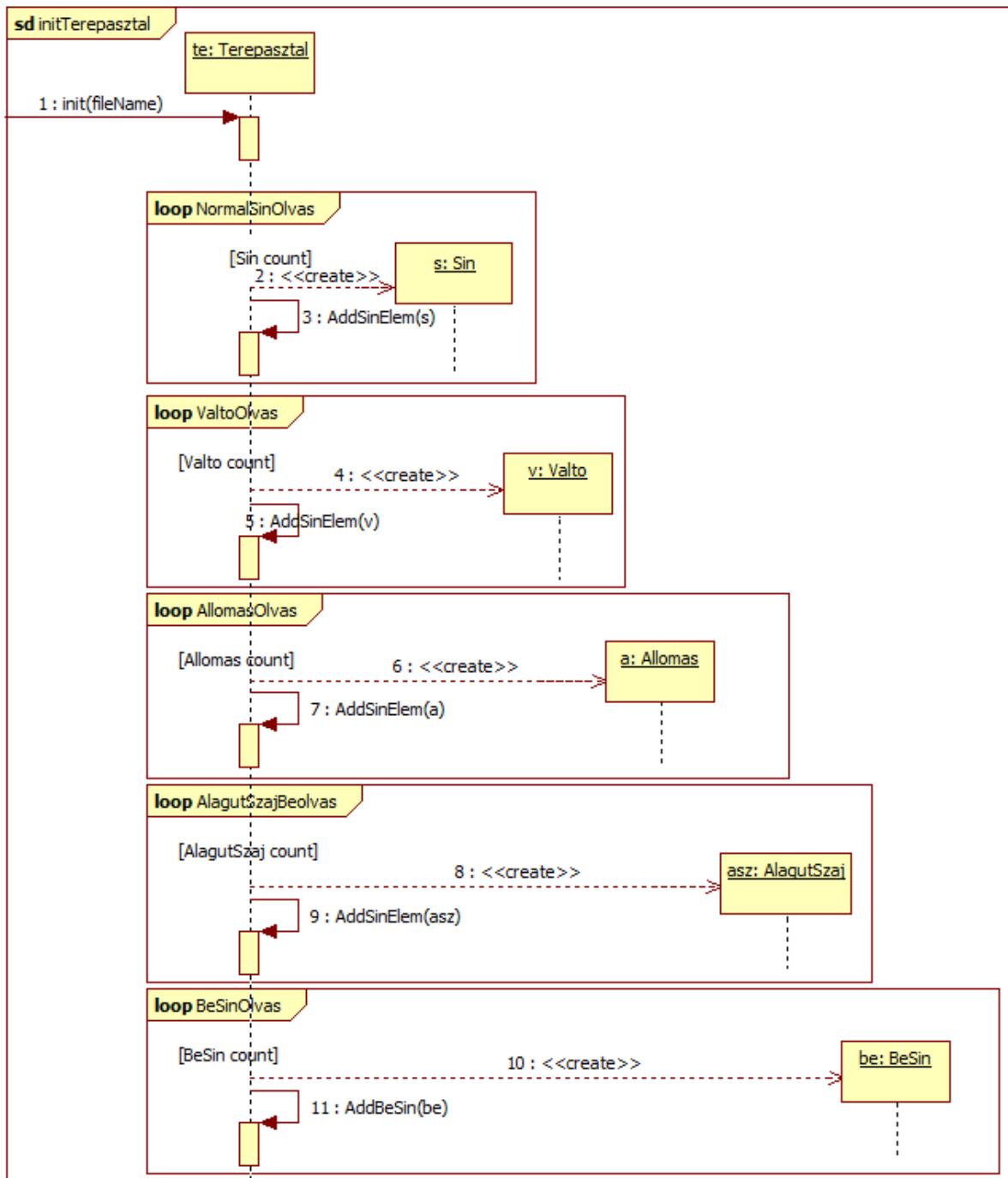
### 5.3.1 Start



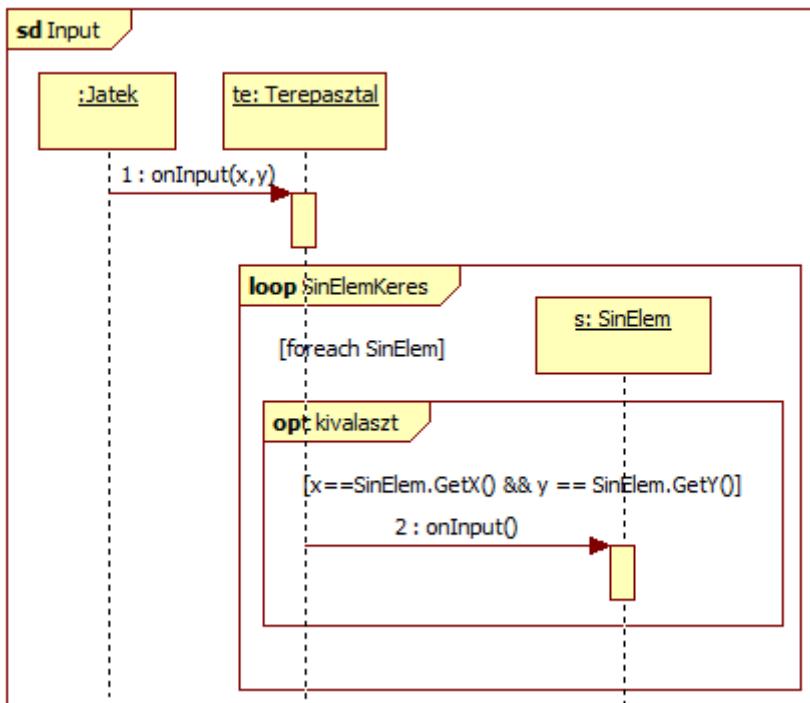
### 5.3.2 InitSzekvencia



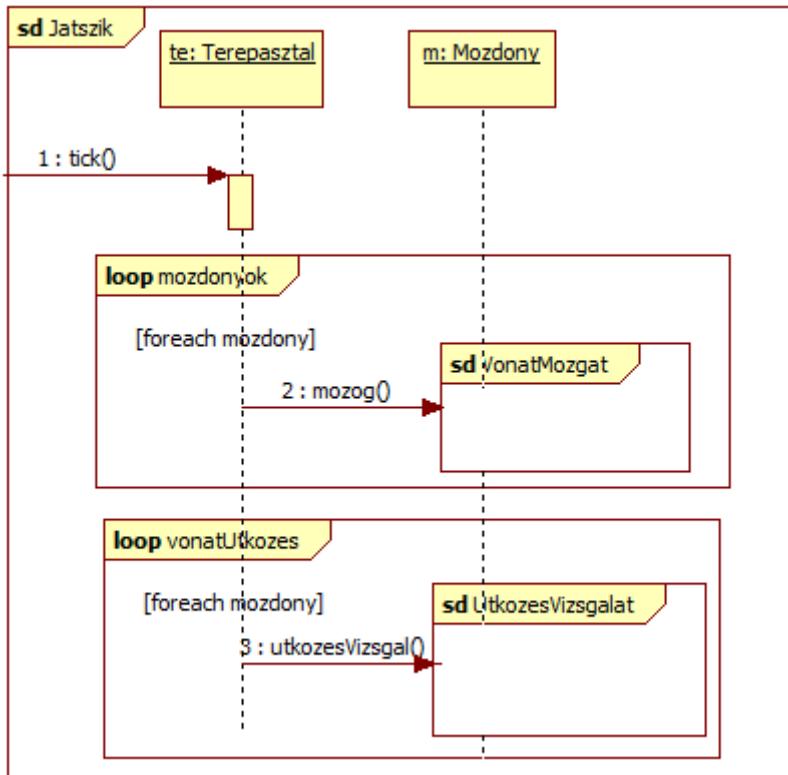
### 5.3.3 initTerepasztal



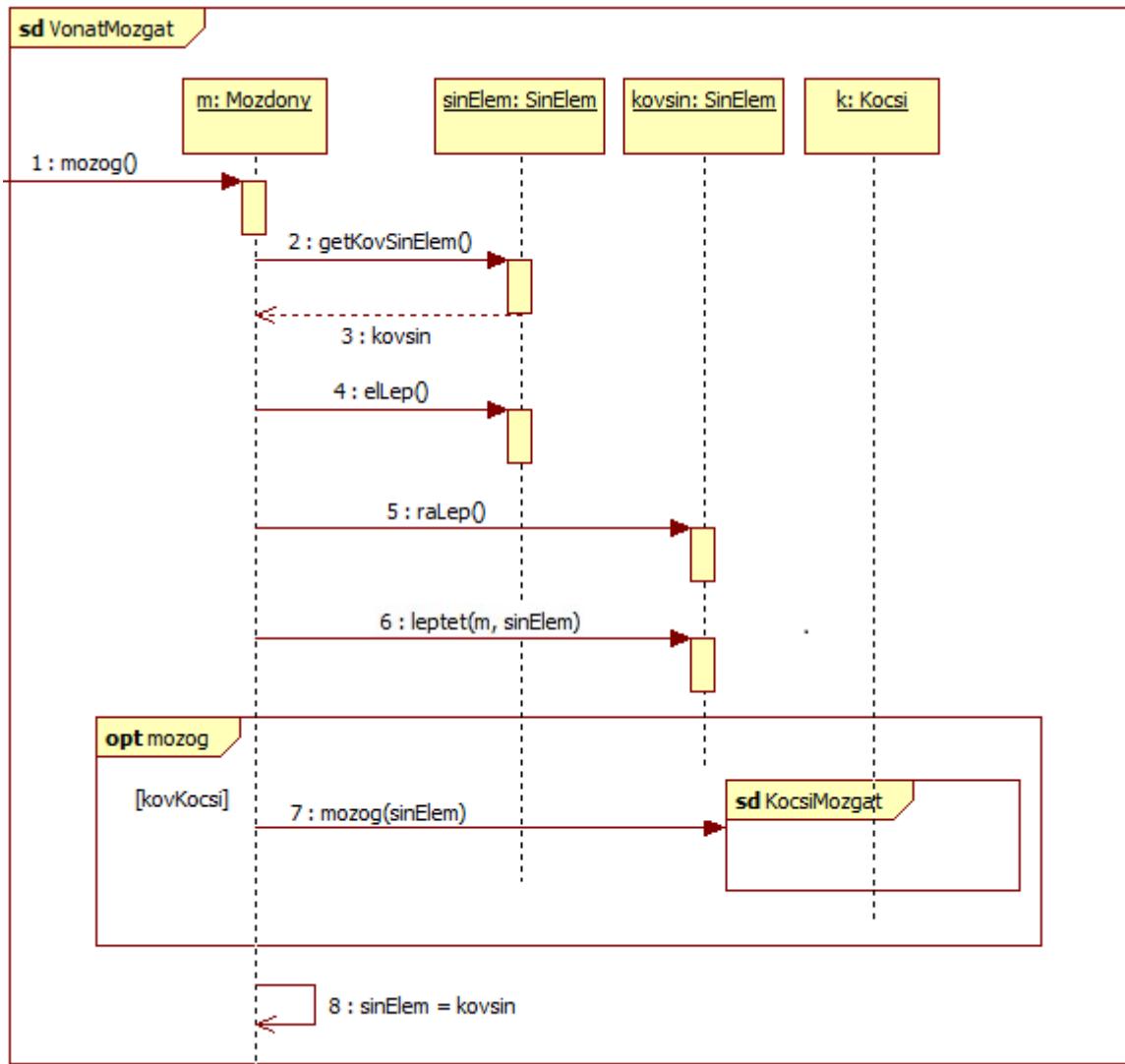
### 5.3.4 Input



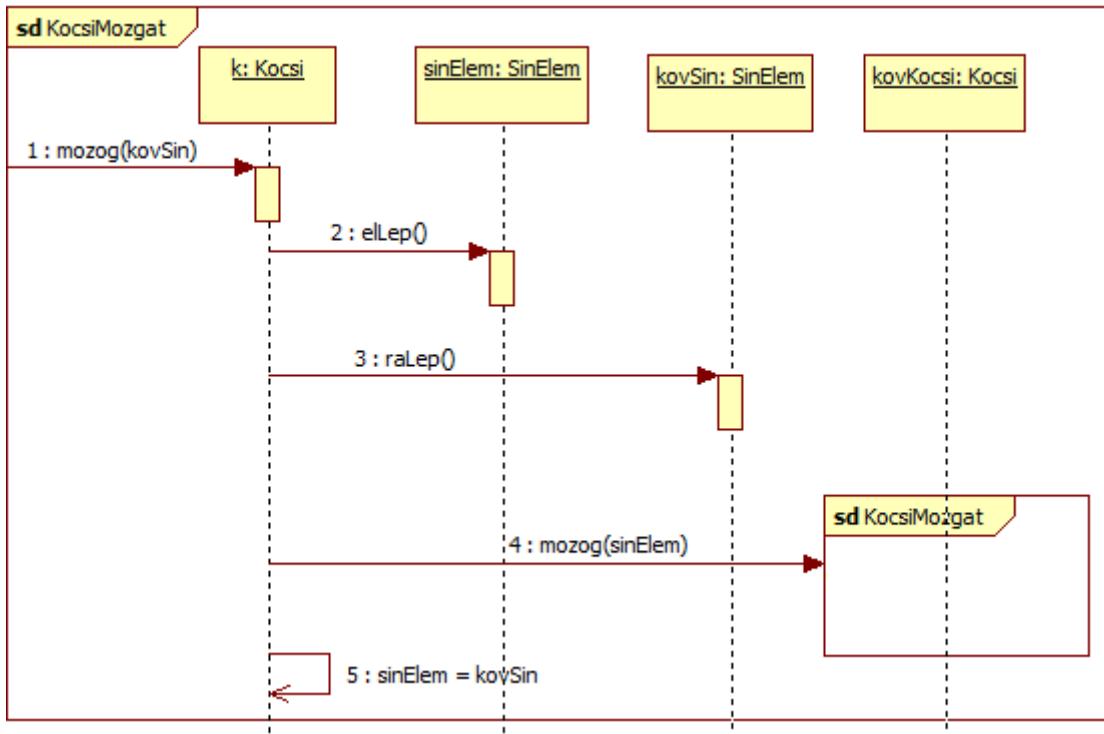
### 5.3.5 Jatszik



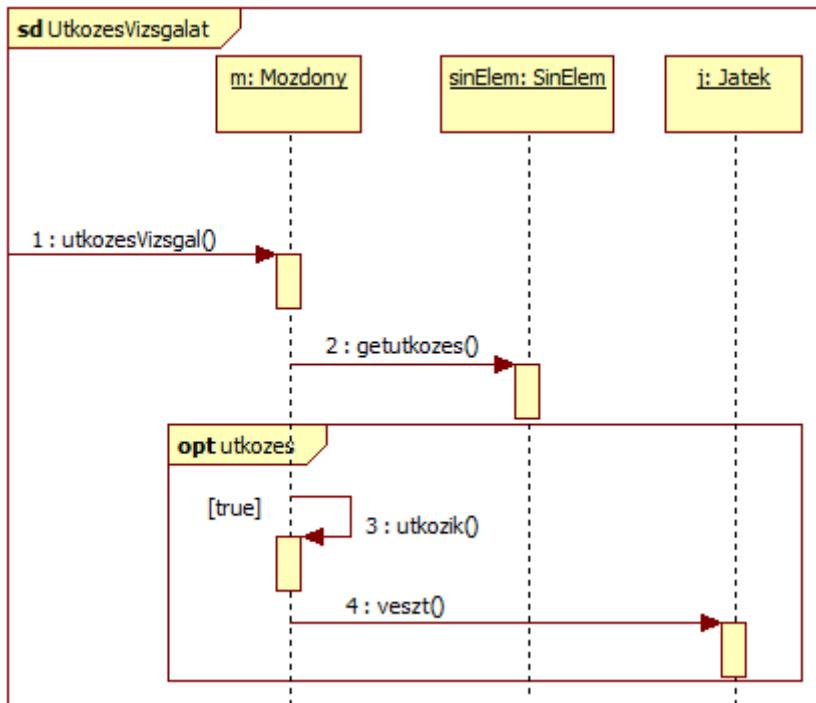
### 5.3.6 VonatMozgat



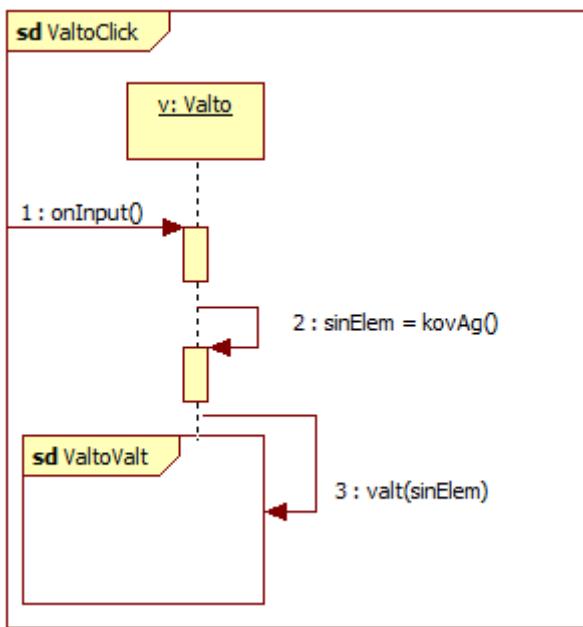
### 5.3.7 KocsiMozgat



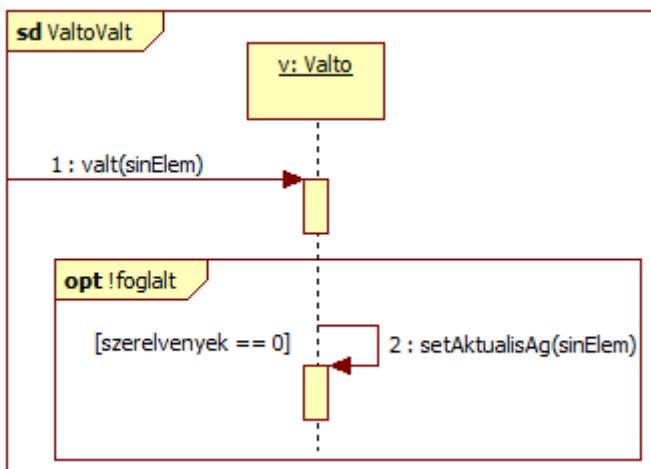
### 5.3.8 UtkozesVizsglat



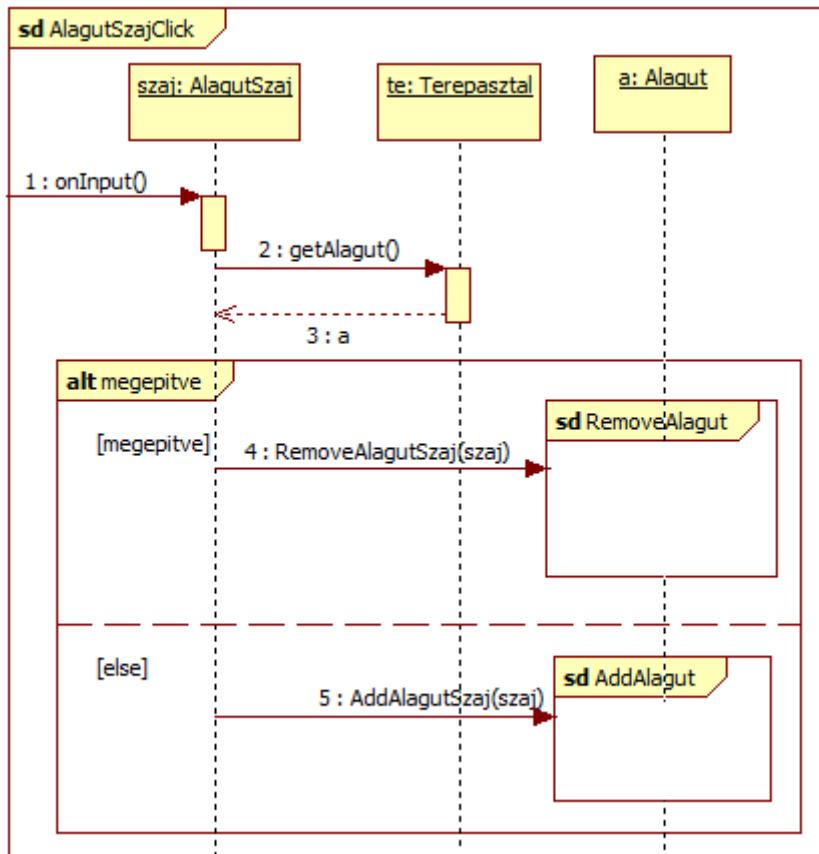
### 5.3.9 ValtoClick



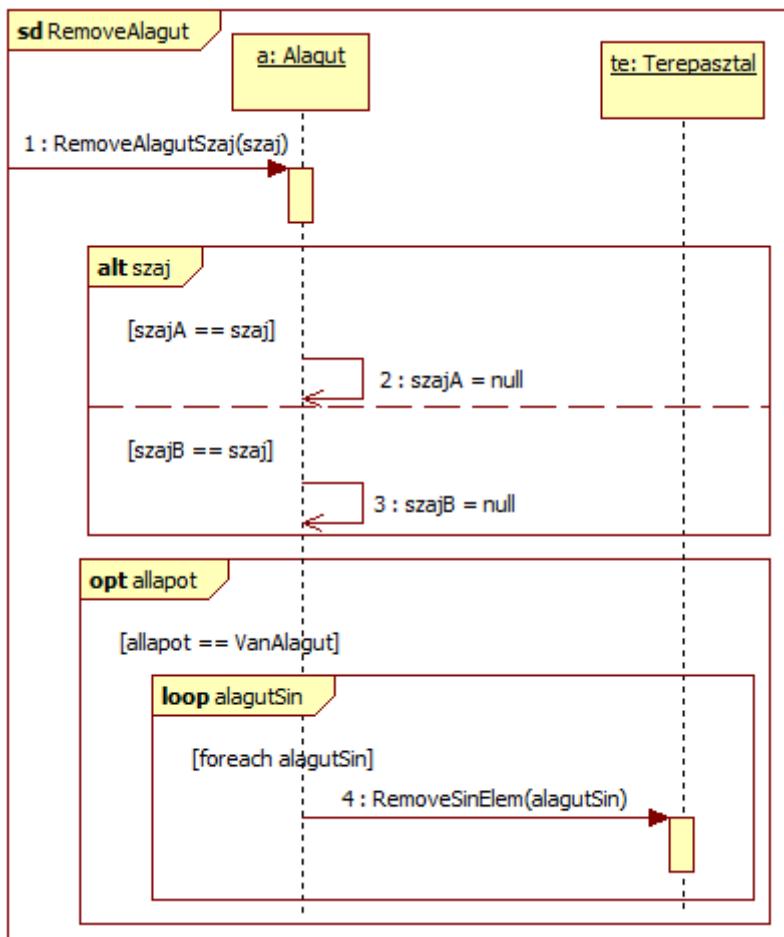
### 5.3.10 ValtoValt



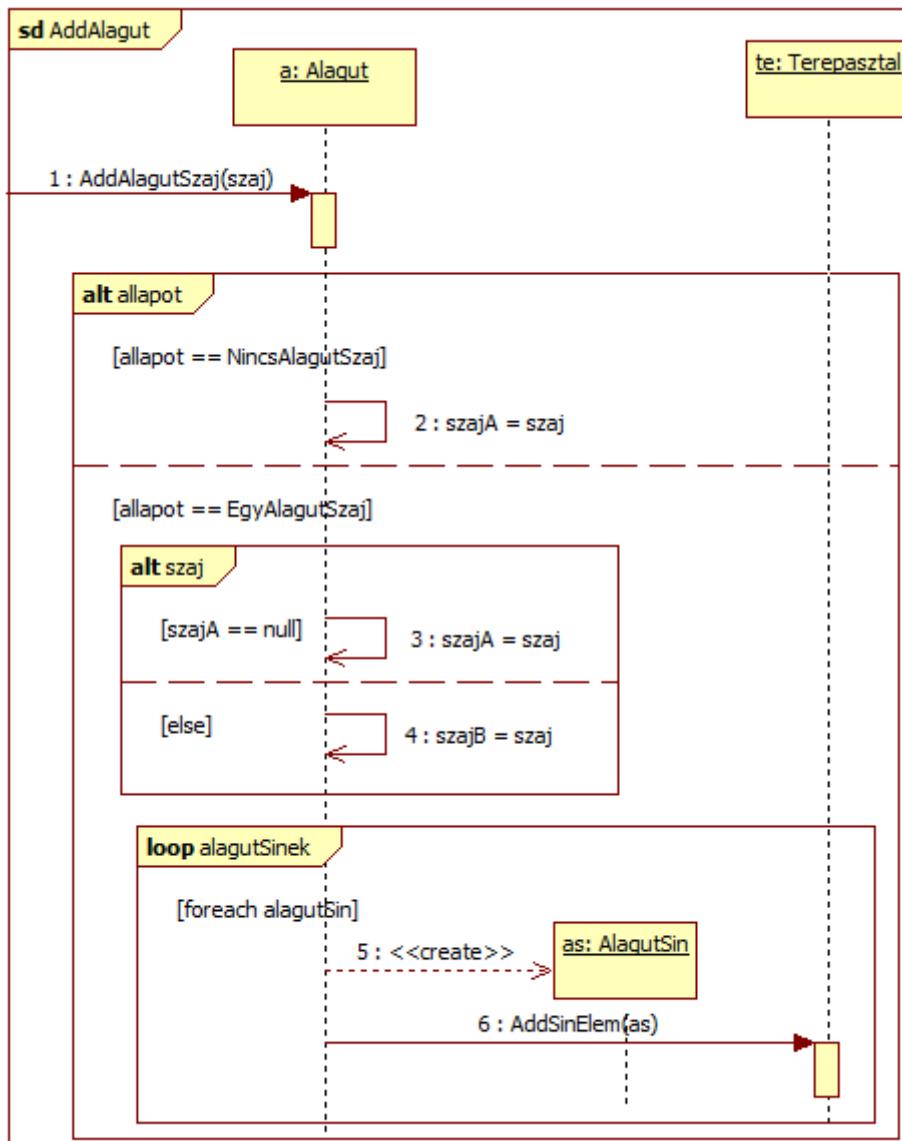
### 5.3.11 AlagutSzajClick



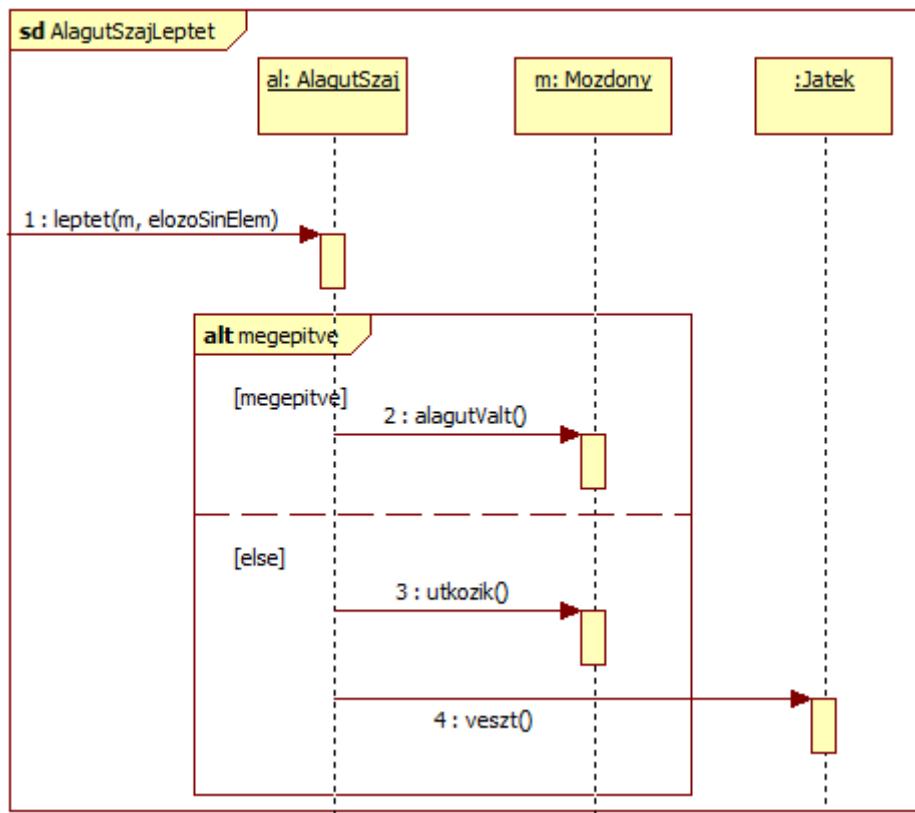
### 5.3.12 RemoveAlagut



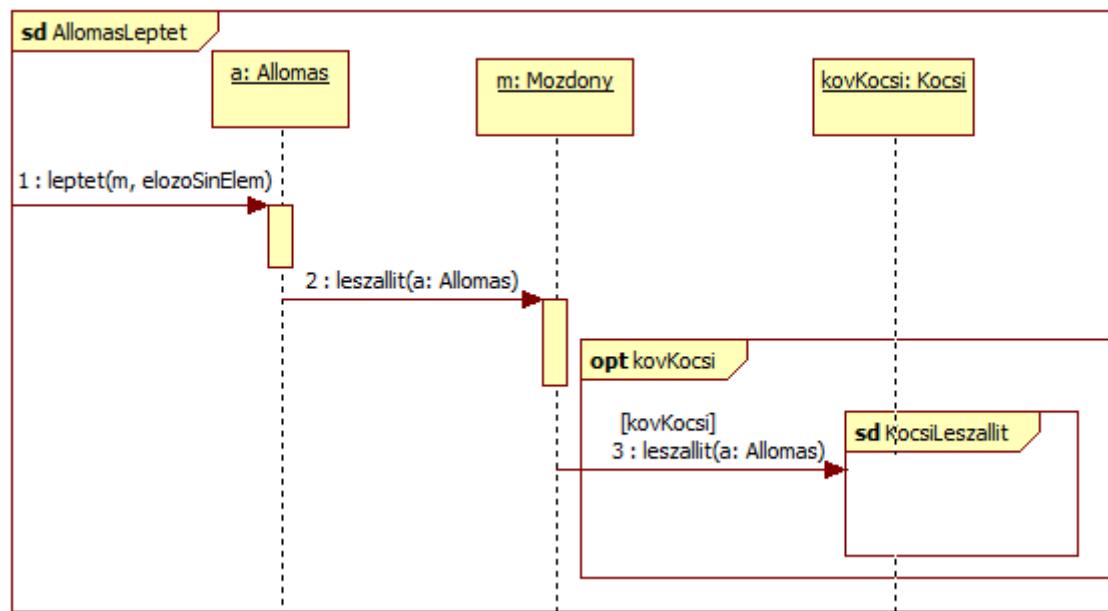
### 5.3.13 AddAlagut



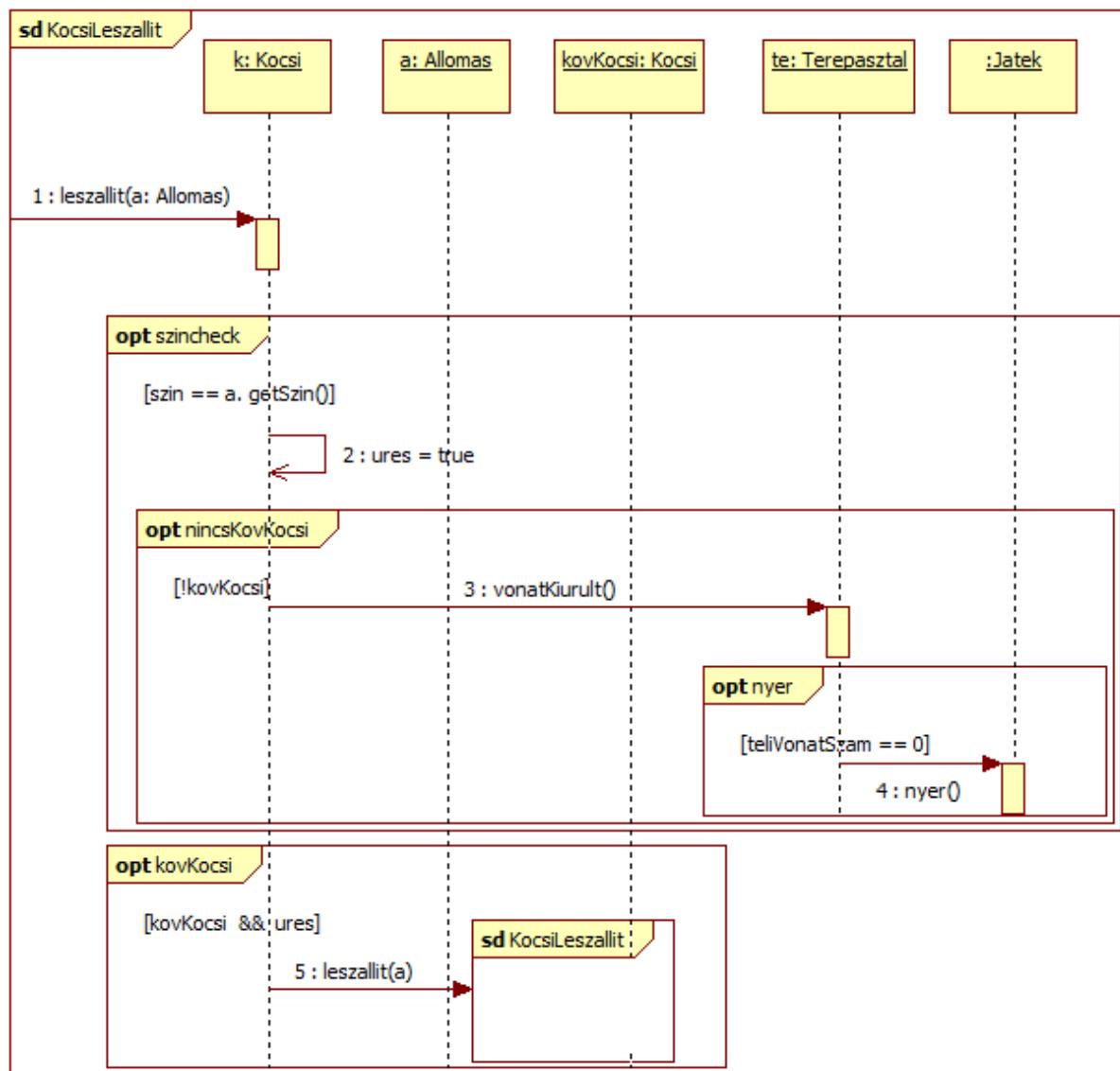
### 5.3.14 AlagutSzajLeptet



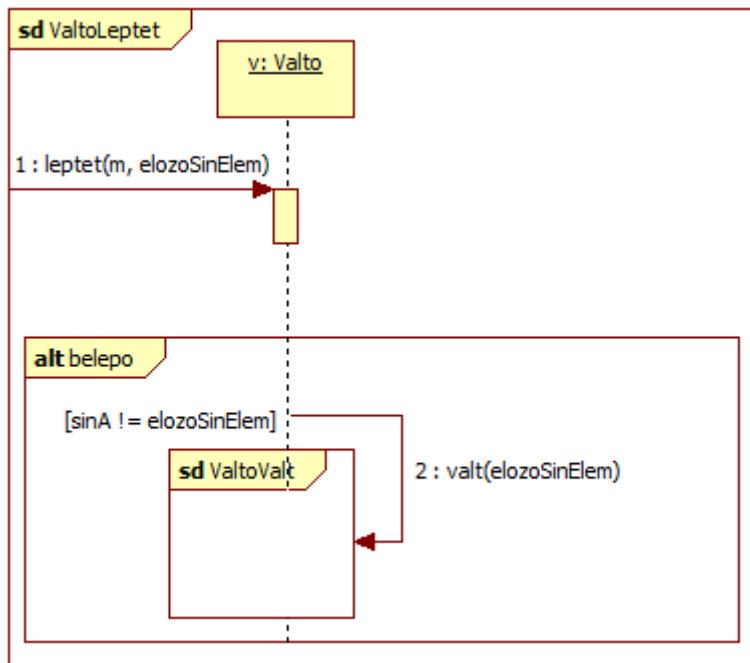
### 5.3.15 AllomasLeptet



### 5.3.16 KocsiLeszallit

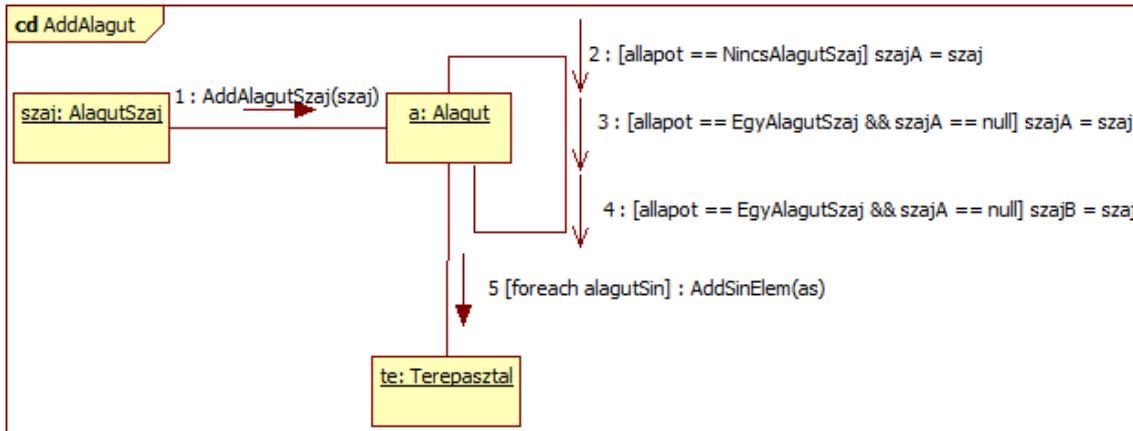


### 5.3.17 ValtoLeptet

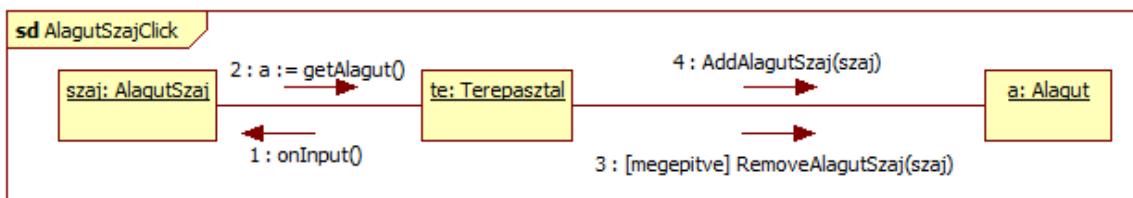


## 5.4 Kommunikációs diagramok

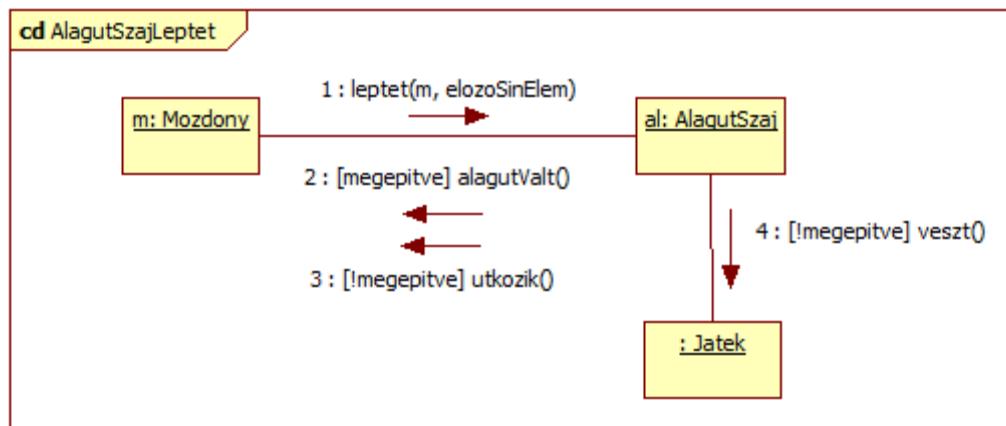
### 5.4.1 AddAlagut



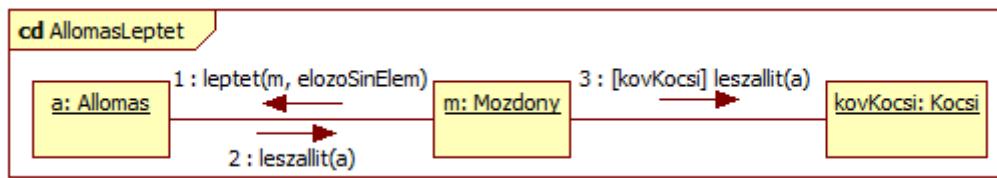
### 5.4.2 AlagutSzajClick



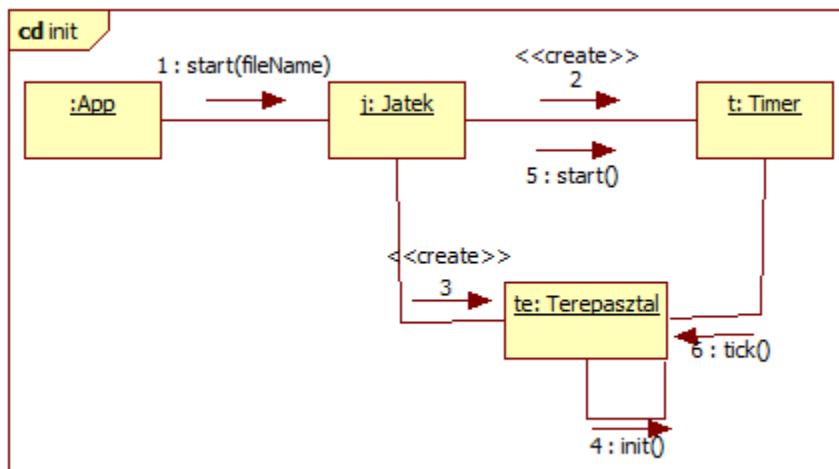
### 5.4.3 AlagutSzajLeptet



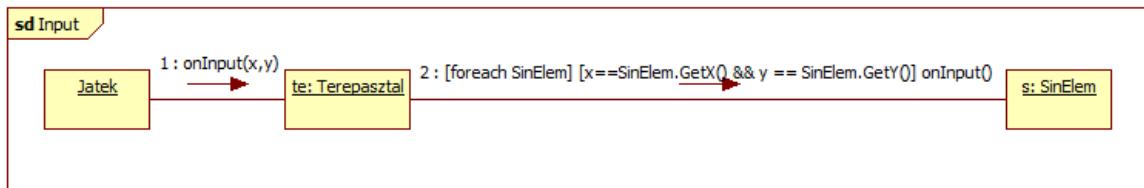
### 5.4.4 AllomasLeptet



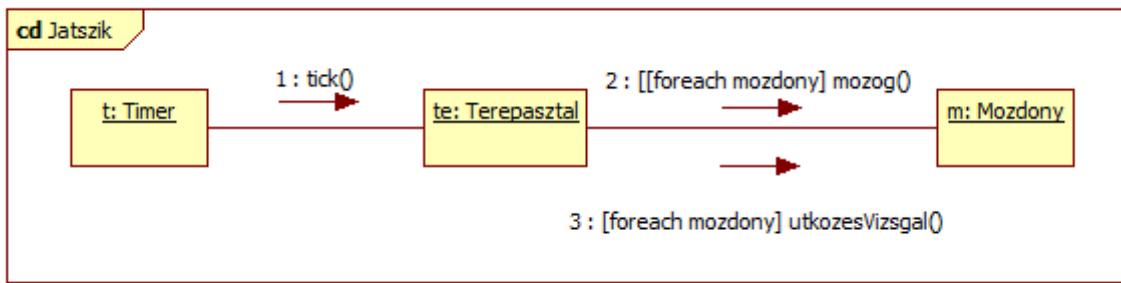
### 5.4.5 Init



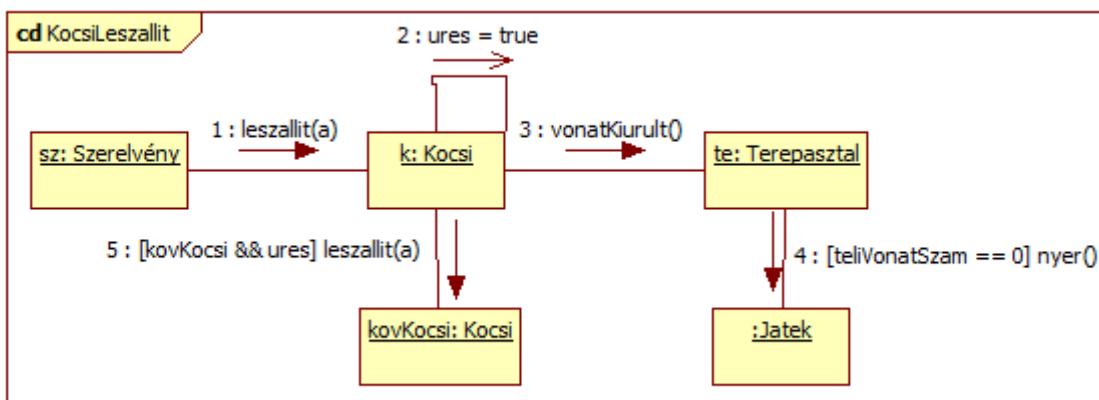
### 5.4.6 Input



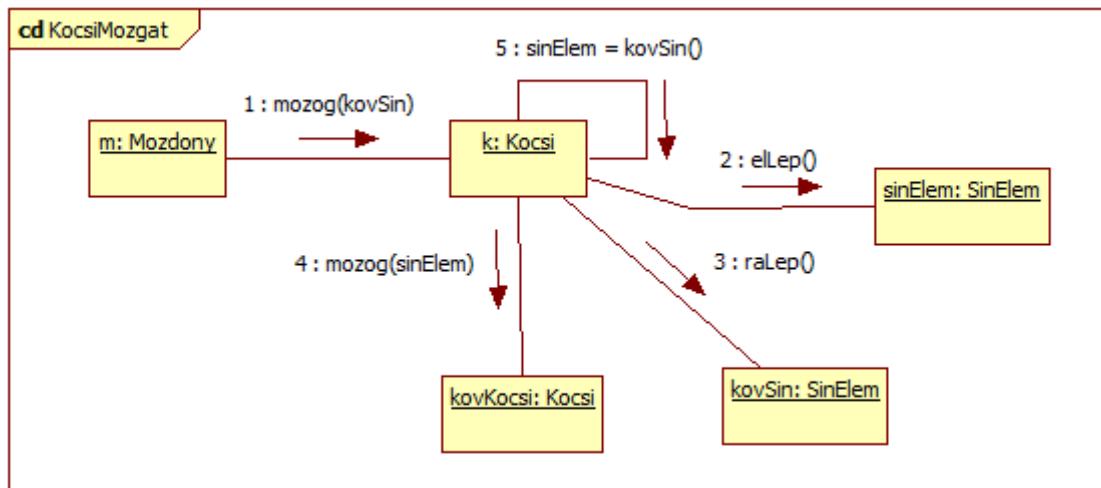
### 5.4.7 Jatszik



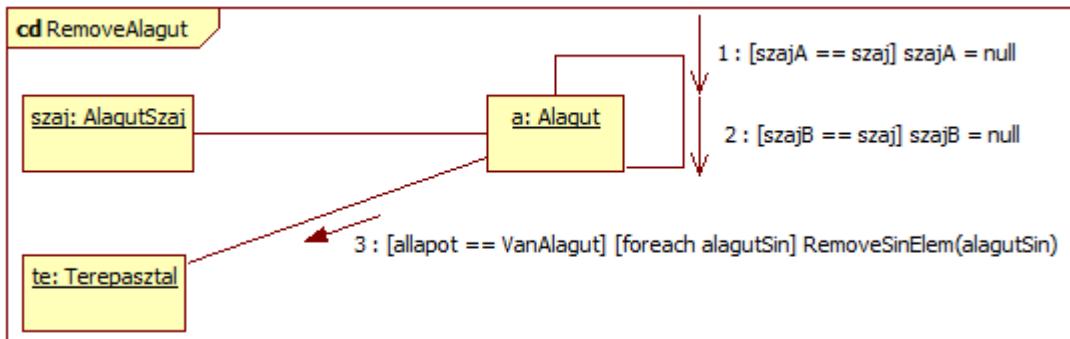
### 5.4.8 KocsiLeszallit



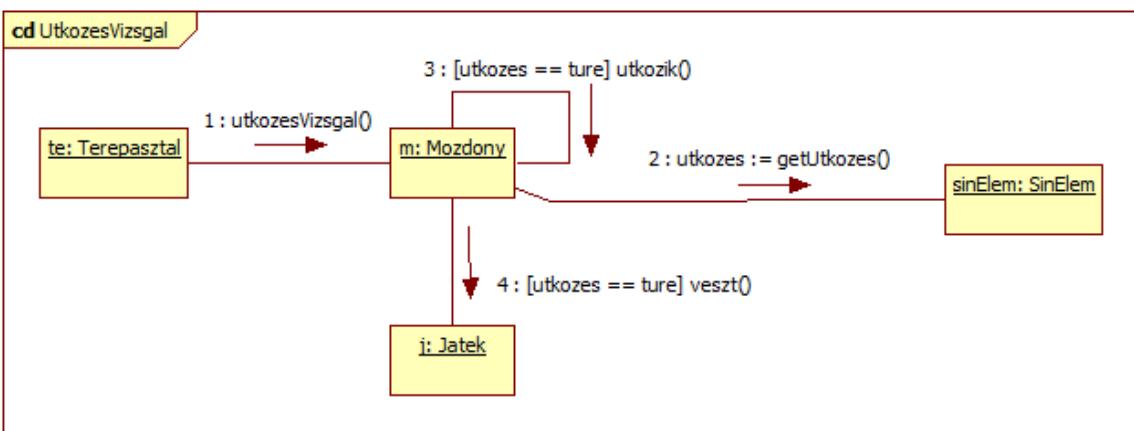
### 5.4.9 KocsiMozgat



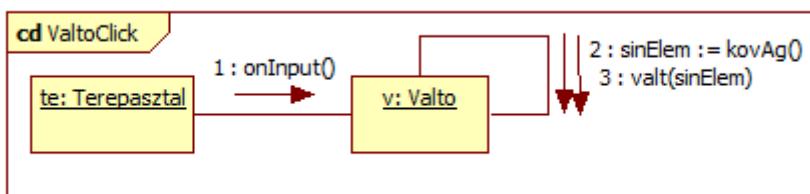
### 5.4.10 RemoveAlagut



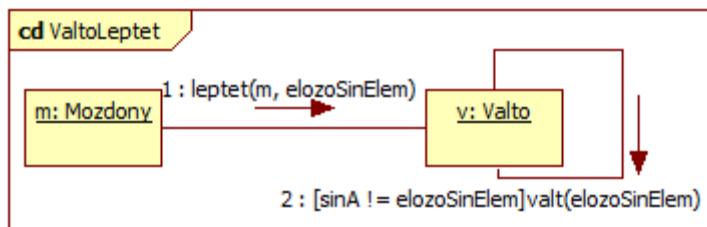
### 5.4.11 UtkozesVizsgalat



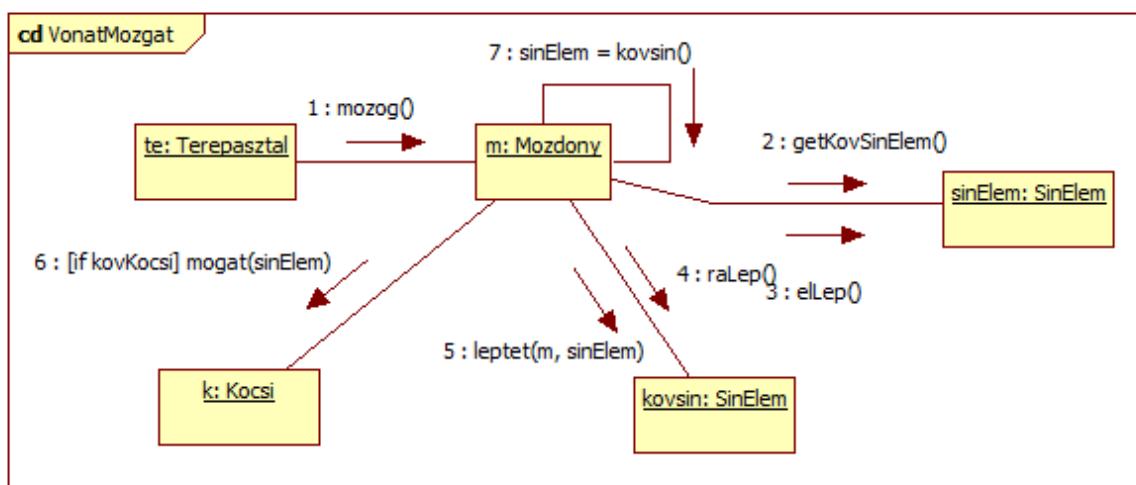
### 5.4.12 ValtoClick



### 5.4.13 ValtoLeptet



### 5.4.14 VonatMozgat



## 5.5 Napló

Kezdet	Időtartam	Részttvevők	Leírás
2017.03.10. 18:45	4 óra	Dócs Szili Varga	Értekezlet. Dócs: 5.1.2 - Use-Case leírások, 5.2 - Szkeleton UI terv Szili,Varga: Diagramok
2017.03.10. 18:45	2 óra	Sillye Krátky	Use-Casek, szekvenciadiagram mok, kommunikációs diagram átbeszélés
2017.03.11 20:00	0,5 óra	Sillye	Átolvasás
2017.03.12 17:45	0,5 óra	Szili	Átvizsgálás, módosítás, dokumentum formázás
2017.03.12 20:00	0,5 óra	Krátky	Dokumentum átnézése
2017.03.12 21:20	0,5 óra	Varga	Dokumentum átnézése, módosítások

## 6. Szkeleton beadás

### 6.1 Fordítási és futtatási útmutató

#### 6.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Alagut.java	1 474 byte	2017.03.18 19:00	Alagut osztály
AlagutAllapot.java	78 byte	2017.03.18 19:00	AlagutAllapot enum
AlagutSzaj.java	2 008 byte	2017.03.18 19:00	AlagutSzaj osztály
Allomas.java	1 566 byte	2017.03.18 19:00	Allomas osztály
App.java	204 byte	2017.03.18 19:00	App osztály
BeSin.java	693 byte	2017.03.18 19:00	BeSin osztály
Jatek.java	908 byte	2017.03.18 19:00	Jatek osztály
Kocsi.java	1 668 byte	2017.03.18 19:00	Kocsi osztály
Menu.java	1 613 byte	2017.03.18 19:00	Menu osztály
Mozdony.java	2 263 byte	2017.03.18 19:00	Mozdony osztály
Sin.java	585 byte	2017.03.18 19:00	Sin osztály
SinElem.java	1 340 byte	2017.03.18 19:00	SinElem osztály
Szerelveny.java	534 byte	2017.03.18 19:00	Szerelvény osztály
Szin.java	126 byte	2017.03.18 19:00	Szin enum
Terepasztal.java	4 131 byte	2017.03.18 19:00	Terepasztal osztály
Timer.java	891 byte	2017.03.18 19:00	Timer osztály
Valto.java	2 370 byte	2017.03.18 19:00	Valto osztály

#### 6.1.2 Fordítás

A fenti listában felsorolt fájlok felhasználásával az alábbi módon lehetséges bináris, futtatható kódot előállítani. Töltsük le, és telepítsük a Java-t a gépre, ha még nincs fent. Ezután töltse le, telepítse és indítsuk el az Eclipse fejlesztői környezetet. Hozzunk létre egy új projektet a File/New/Java Project opcióval. Projektnévnek adjuk meg kívánt szöveget, pl.: projlab. Állítsuk be a projekt helyét kívánt helyre, pl. az alapból felajánlott workspace mappába. A Finish gombra kattintva bezáródik a varázsló és egy üres projektet kapunk. Nyissunk meg egy fájlkezelőt, és keressük meg a workspace mappán belül a projekt nevének mappáját, és ezen belül az üres src mappát. Ebbe helyezzük át az összes fent olvasható .java kiterjesztésű fájlt. Lépjünk vissza Eclipse-be, és jobb egérgombbal kattintsunk a projekt nevére a bal oldali sávban, majd válasszuk a Refresh opciót, hogy betöltsön a forráskód. Az App.java fájlban található a program belépései pontja. Végül a fenti menüsávban található zöld nyíl gombot nyomjuk meg. Ez automatikusan le fogja fordítani a kódot, és el is indítja a programot. Az src mappa mellett pedig megjelennek a binárisok a bin mappában. Lehetőség van ezeket összecsomagolni, egy egységes .jar fájlba. Ezt a File/Export menüpontban tehetjük meg. A listából a java fül alatt a Runnable JAR File elemet válasszuk ki. A következő lépésben válasszunk launch configot (App - projlab) és az exportálás helyét. A Finish gombra kattintva legenerálódik a futtatható .jar fájt.

### 6.1.3 Futtatás

A fentebb leírt módon tehát lehetőség van Eclipse-beli futtatásra is. Azonban a jar fájl futtatása sokkal kényelmesebb, hiszen nincs szükség fejlesztői környezetre. Nyissunk egy command promptot, és navigálunk az előző pontban generált jar fájl mappájához a cd paranccsal. Futtatáshoz írjuk be az alábbi utasítást:

```
java -jar projlab.jar
```

## 6.2 Értékelés

Tag neve	Munka százalékban
Dócs Zoltán	20%
Krátky Gergő Ádám	20%
Sillye Márk	20%
Szili Péter	20%
Varga János	20%

### 6.3 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017.03.18. 18:00	2 óra	Varga	Szerelveny, Mozdony, Kocsi megvalósítása.
2018.03.18 16:00	1 óra	Szili	Terepasztal, Jatek, Timer megvalósítása.
2017.03.18. 23:00	1 óra	Krátky	Alagut, AlagutSzaj, AlagutAllapot megvalósítása.
2017.03.19. 13:00	8 óra	Dócs	Allomas, BeSin, Menu, Szín megvalósítása. Dokumentum: 6.1.2 fordítás és 6.1.3 futtatás.
2017.03.19. 13:00	8 óra	Sillye	Sin, SinElem, Valto, tervezés, tesztelés
2017.03.19. 13:00	6 óra	Varga	Osztályok átnézése, Tesztelés
2017.03.19. 13:00	6 óra	Krátky	Osztályok átnézése, Dokumentum szerkesztése
2017.03.19 13:00	6 óra	Szili	Osztályok átnézése, dokumentum átolvasása, formázása

## 7. Prototípus koncepciója

### 7.0 Változásokkal kapcsolatos módosítások

#### 7.0.1 Szöveges elemzés

**Sheldon új pályaelemet, kereszteződő síneket vásárolt. A kereszteződés egyszintű, a különböző irányokból jövő vonatok a kereszteződésben ütköznek.**

Egy új KeresztSin osztályt vezetünk be, amihez négy SinElem csatlakozik. Így ennek az osztálynak a feladata lesz, hogy az adott irányból érkező vonatot a kereszteződésnek megfelelően továbbítsa.

**Egyes állomásokon utasok a megfelelő színű üres kocsikba (a kocsi szerelvényben elfoglalt helyzetétől függetlenül) fel tudnak szállni.**

Az Allomas osztály kapott egy új boolean ures változót. A pálya betöltésénél meg lehet adni, hogy egyes állomásokon legyenek utasok. Az állomás feladata, hogy az elhaladó, üres és megegyező színű kocsikra utasokat szállítsanak fel a kocsik új felszall metódusával.

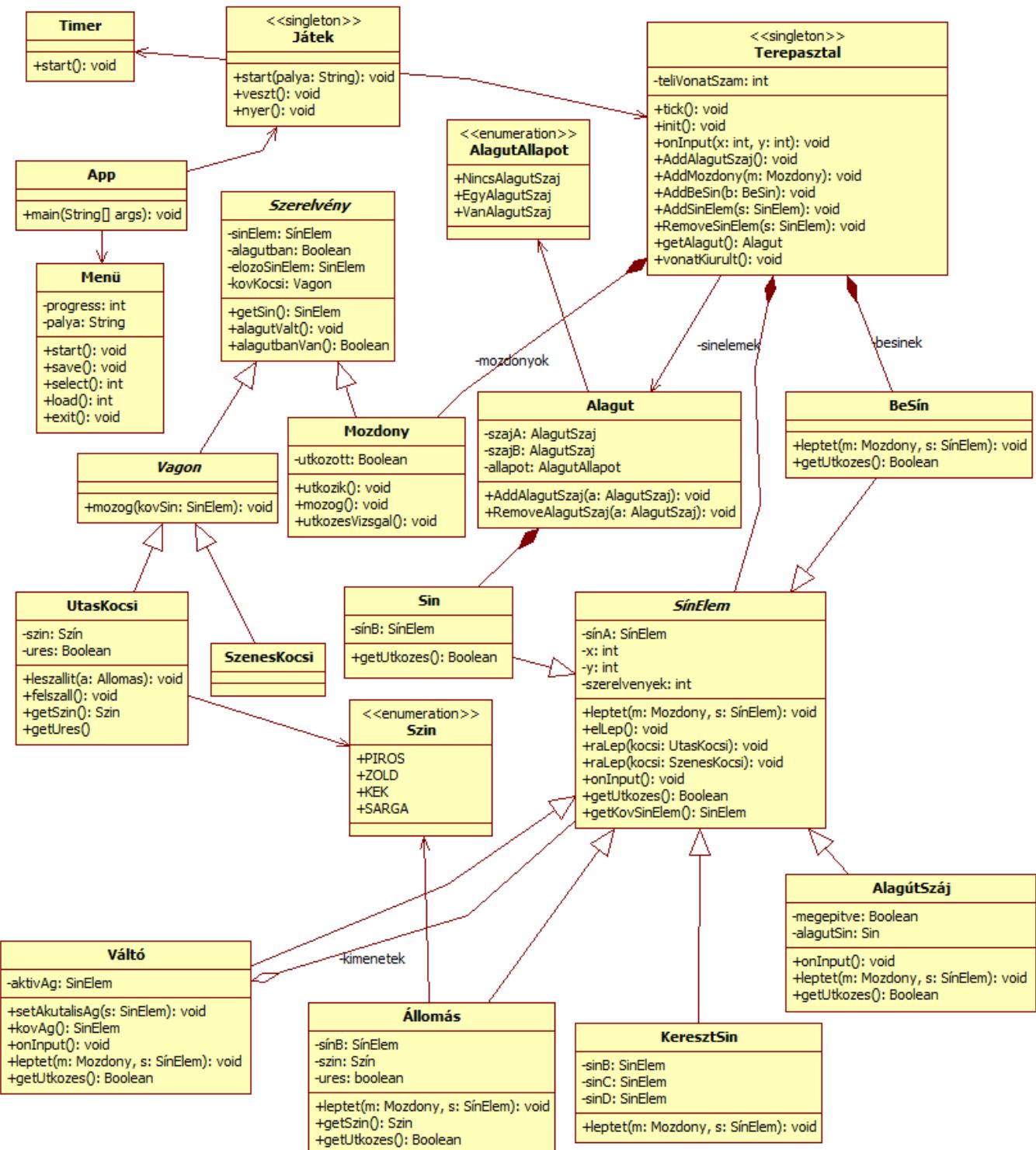
Akkor is felszállnak az állomáson tartózkodó utasok, ha az állomásra érkező vonat akkor ürül ki.

**Sheldon bővítette a vagonkészletét. Vett szeneskocsikat, amiken nem utaznak utasok, nem is tudnak felszállni. Az utasok leszállásánál az ilyen vagonokat nem vesszük figyelembe.**

Bevezettünk egy új absztrakt osztályt, Vagon, ami a Szerelveny osztalyból szarmazik. Az új SzenesKocsi osztaly ebből a Vagon osztalyból származik, és ez valósítja meg a leírásban kért szeneskocsit. A Kocsi osztályt az egyértelműség kedvéért átneveztük UtasKocsi-ra. Az UtasKocsi osztály is a Vagon-ból származik.

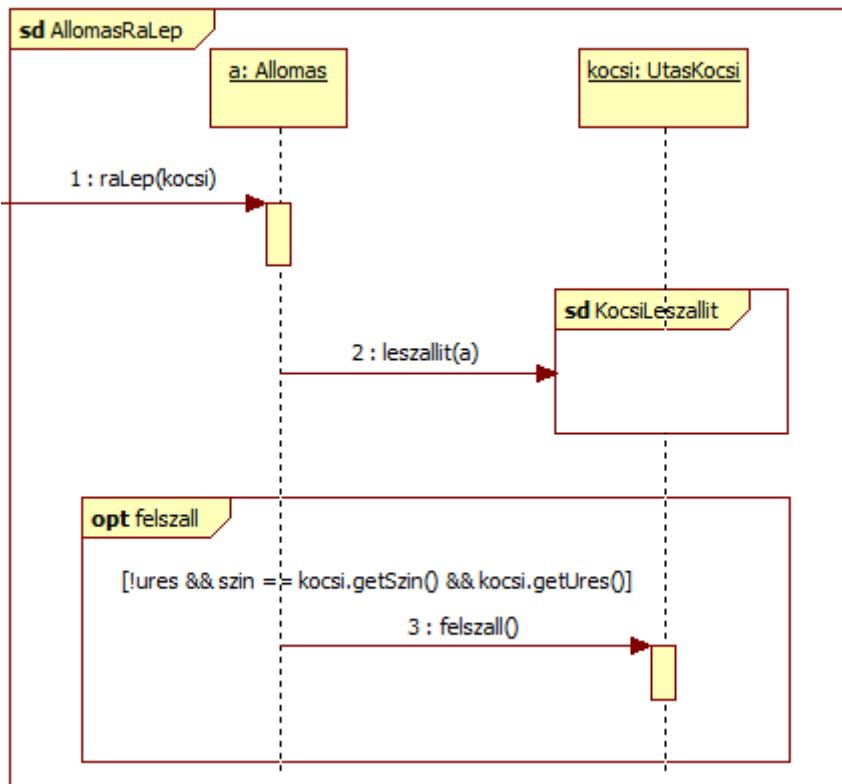
Továbbá a SinElem osztály kapott egy új metódust: raLep(k: SzenesKocsi), így ez fog meghívódni, mikor szeneskocsi lép a sínelemre. Így az állomsnál külön lehet kezelni a szeneskocsi érkezését is.

## 7.0.2 Módosított osztálydiagram



## 7.0.3 Módosított szekvenciadiagrammok

### 7.0.3.1 AllomasLeptet



## 7.1 Prototípus interface-definíciója

### 7.1.1 Az interfész általános leírása

A proto interfésze a szabványos bemenetről olvas be utasításokat a Tesztelőtől, és a szabványos kimenetre írja ki a parancsok kimeneteit. Ezáltal lehetőség nyílik nem csak konzolos szövegbevitelre, hanem fájlból olvasásra is, illetve igény szerint a kimenet fájlba mentésére.

### 7.1.2 Bemeneti nyelv

**loadmap <palyanev>**

Leírás: Pálya betöltése

Opciók: palyanev: Melyik pályát töltse be.

**play**

Leírás: Elindítja a játékot

Opciók: -

**tick <db>**

Leírás: Idő léptetése egységgel.

Opciók: db: Léptetések száma.

**valto <id>**

**Leírás:** Váltó aktív ágának változtatása.

**Opciók:** id: Váltandó váltó azonosítója

**alagutszaj <id>**

**Leírás:** Alagútszájra kattintás.

**Opciók:** id: Adott alagútszaj azonosítója.

**random <enabled>**

**Leírás:** Tesztelés céljából determinisztikus lefutás kikényszerítése.

**Opciók:** enabled: véletlenszerűség engedélyezése.

**reset**

**Leírás:** Állapot alaphelyzetbe állítása.

**Opciók:** -

**info <id> <attributum>**

**Leírás:** Információ lekérdezése az adott elemmel kapcsolatban.

**Opciók:**

argumentum nélkül: minden id kilistázása

id: az adott id-jű elem attribútumainak kilistázása

attributum: az adott attribútum értéknek kiírása

Szükség volt konfigurációs fájlok tervezésére is, melyekben az egyes pályák felépítését tároljuk. Egy ilyen fájl a következő adatokat tárolja: sínelemek és azok tulajdonságai, sínelemek közötti kapcsolatok leírása, vonatok összetétele és indulása (determinisztikus lefutás esetén).

Fontos volt belátni, hogy addig nem kezdhetjük el a kapcsolatokat felépíteni, míg minden sínelementet be nem olvastunk (hiszen pl. kihez kötnénk az első beolvasott elemet?). Így az a tervezői döntés született, hogy minden sínelementet egy azonosítóval jelölünk és a fájlt úgy struktúraljuk, hogy először beolvassuk az összeset, tulajdonságaikkal együtt, viszont kapcsolatok nélkül. Majd mikor már biztosan minden elemről tudomásunk van, elkezdjük egyenként beolvasni a kapcsolatokat az azonosítók alapján.

Végül a kapcsolatok leírása után szükség van egy 3. blokkra, mely a vonatok determinisztikus pályáralépését biztosítja. Itt minden vonatra megadjuk, hogy melyik BeSínről, hányadik időlépésben, milyen típusú vagonokkal indul.

A konkrét struktúra példájaként az alábbi mintafájl szolgál:

```

besin b1           //sínelemek felsorolása:
besin b2           // [Típus ID] formátumban
normalsin n1       //elemek sorrendje nem számít
allomas a1 p 1     //a1 paraméterei: p-piros, 1-utasok
allomas a2 k 0     //a2 paraméterei: k-kék, 0-üres
valto v1
alagutszaj s1
alagutszaj s2
alagutszaj s3
keresztsin k1
.
b1-a n1-a         //kapcsolatok felsorolása: sorrend nem
n1-b a1-a         számít. [ID-ág ID-ág] formátumban
a1-b v1-a         //pl.: n1 azonosítójú normalsin típusú
v1-b s1-a         sínelem sinB ága kapcsolódik a1
v1-c k1-a         azonosítójú allomas típusú sínelem sinA
k1-b s2-a         ágához (és fordítva)
k1-c s3-a
k1-d b2-a
.
b1 2 ppxk         //vonatok felsorolása:[besín idő vagonok]
b2 15 xxxxpx     //pl.: vonat érkezik b1 azonosítójú besin
                  típusú sínelemre a 2. ticknel.
                  Szerelevenyek: Mozdony, piros, piros,
                  szemes, kék

```

A későbbiekben itt tervezzük a sínelemek grafikus tulajdonságainak (pl. koordináták) eltárolását is.

### 7.1.3 Kimeneti nyelv

**loadmap <palyanev>:**

<palyanev>. palya kivalasztva

**play:**

Jatek elindult

**tick <db>:**

<db> tick megtortent

**valto <id>:**

<id>. valto atváltott az <ág\_id>. agba

**alagutszaj <id>:**

<id>. alagutszaj megepitve

<id>. alagutszaj lerombolva

alagut megepitve

alagut lerombolva

**random <enabled>:**

random enabled

random disabled

**reset:**

reset done

**info <id>:**

```
<id>
<Attributum1 neve>: <attributum1 értéke>
<Attributum2 neve>: <attributum2 értéke>
...

```

## 7.2 Összes részletes use-case

<b>Use-case neve</b>	Pálya betöltése
<b>Rövid leírás</b>	A program betölt egy pályát.
<b>Aktorok</b>	Tesztelő
<b>Forgatókönyv</b>	Tesztelő által megadott nevű pálya betöltődik fájlból.

<b>Use-case neve</b>	Játék indítása
<b>Rövid leírás</b>	A betöltött pálya elindítása.
<b>Aktorok</b>	Tesztelő
<b>Forgatókönyv</b>	A játék elindul, az egyes objektumok inicializálódnak.

<b>Use-case neve</b>	Léptetés
<b>Rövid leírás</b>	A játék léptetése.
<b>Aktorok</b>	Tesztelő
<b>Forgatókönyv</b>	A játék továbblép megadott időegységgel.

<b>Use-case neve</b>	Váltó váltás
<b>Rövid leírás</b>	Adott váltó váltása.
<b>Aktorok</b>	Tesztelő
<b>Forgatókönyv</b>	A Tesztelő által kiválasztott váltó aktív ága átvált a következő ágba.

<b>Use-case neve</b>	Alagútszáj módosít
<b>Rövid leírás</b>	Adott alagútszáj módosítása.
<b>Aktorok</b>	Tesztelő
<b>Forgatókönyv</b>	A Tesztelő által kiválasztott alagútszáj megépül vagy lerombolódik.

<b>Use-case neve</b>	Random beállítása
<b>Rövid leírás</b>	Tesztelő engedélyezheti vagy kiiktathatja a determinisztikussági faktort a játékból.
<b>Aktorok</b>	Tesztelő
<b>Forgatókönyv</b>	Be illetve kikapcsolódnak a véletlenszerű elemek a játékból.

<b>Use-case neve</b>	Reset
<b>Rövid leírás</b>	Tesztelő a játék során bármikor visszaállíthatja a játéket a kiindulási állapotba.
<b>Aktorok</b>	Tesztelő
<b>Forgatókönyv</b>	A játék visszaáll az alaphelyzetbe.

<b>Use-case neve</b>	Info
<b>Rövid leírás</b>	Információ lekérése adott azonosítójú elemről.
<b>Aktorok</b>	Tesztelő
<b>Forgatókönyv</b>	Kiíródik az azonosítóval adott elem összes attribútuma.

### 7.3 Tesztelési terv

<b>Teszt-eset neve</b>	Játék fut
<b>Rövid leírás</b>	A tesztelő kiválasztja a megfelelő pályát, amit aztán elindít, ezután a játékban egy vonat kering olyan körülményekkel, hogy a játéknak soha ne legyen vége.
<b>Teszt célja</b>	Tesztelni, hogy a játék megfelelően elindul és működik.

<b>Teszt-eset neve</b>	Besín működik
<b>Rövid leírás</b>	A megfelelő pálya elindítása után, a besín elemeken szerelvények érkeznek a pályára.
<b>Teszt célja</b>	Tesztelni, hogy a pályára beérkeznek a szerelvények.

<b>Teszt-eset neve</b>	Vonat ütközik
<b>Rövid leírás</b>	A megfelelő pálya elindítása után, két vonat rövid időn belül ütközik egymással.
<b>Teszt célja</b>	Tesztelni, hogy két vonat ütközésekor vége a játéknak.

<b>Teszt-eset neve</b>	Játék nyer
<b>Rövid leírás</b>	A megfelelő pálya elindítása után, egy beérkező szerelvény egyetlen kocsijából rövid időn belül leszállnak az utasok, aminek hatására a játékos nyer.
<b>Teszt célja</b>	Tesztelni, hogy az utolsó kocsiból leszállva a játékos megnyeri-e a játékot.

<b>Teszt-eset neve</b>	Állomásnál leszáll
<b>Rövid leírás</b>	A játék elindul, betöltödik egy teszt pálya, aminek a terepasztalán található BeSin, Sin és Állomás. Utána jön a BeSinen egy vonat, ami tartalmaz legalább egy Kocsit, a vonat halad a Sinen, amíg el nem ér egy olyan színű Állomást, ahol le tudnak szállni az utasok. Az utasok leszállnak, és lesz egy üres kocsija a vonatnak.
<b>Teszt célja</b>	Tesztelni, hogy a leszállás rendesen működik-e az Allomas és a Kocsi osztályban, azaz hogy helyesen értékeli-e ki az előtte lévő üres kocsik létét, majd helyesen ürül-e ki a Kocsi.

<b>Teszt-eset neve</b>	Állomásnál nem száll le
<b>Rövid leírás</b>	A játék elindul, betöltődik egy teszt pálya, aminek a terepasztalán található BeSin, Sin és Állomás. Utána jön a BeSinen egy legalább kettő, különböző színű Kocsit tartalmazó vonat, a vonat halad a Sinen, amíg el nem ér egy Állomást, aminek a Szin-e nem egyezik meg az első nemüres Kocsiével, de megegyezik bármelyik más Kocsi színével. Az utasok nem szállnak le, és megy tovább a vonat.
<b>Teszt célja</b>	Tesztni, hogy a leszállás rendesen működik-e az Allomas és a Kocsi osztályban, azaz hogy helyesen értékeli-e ki az előtte lévő üres kocsik létét, majd helyesen kiürülés nélkül továbbmegy.

<b>Teszt-eset neve</b>	Állomásnál felszáll
<b>Rövid leírás</b>	A játék elindul, betöltődik egy teszt pálya, aminek a terepasztalán található BeSin, Sin és Állomás. Utána jön a BeSinen egy vonat, ami tartalmaz legalább egy Kocsit. A vonat halad a Sinen, amíg el nem ér egy Állomás-t, ahol leszállnak az utasok az első kocsiból. Utána halad a Sinen, és elér egy olyan Állomást, ahol fel akarnak szállni. Az utasok felszállnak, és megy tovább a vonat.
<b>Teszt célja</b>	Tesztni, hogy a felszállás rendesen működik-e, azaz hogy felszállnak az utasok az arra alkalmas Állomáson.

<b>Teszt-eset neve</b>	Állomásnál nem száll fel
<b>Rövid leírás</b>	A játék elindul, betöltődik egy teszt pálya, aminek a terepasztalán található BeSin, Sin és Állomás. Utána jön a BeSinen egy vonat, ami tartalmaz legalább egy Kocsit. A vonat halad a Sinen, amíg el nem ér egy Állomás-t, ahol leszállnak az utasok az első kocsiból. Utána halad a Sinen, és elér egy olyan Állomást, ahol nem akarnak felszállni. Az utasok nem szállnak fel, és megy tovább a vonat.
<b>Teszt célja</b>	Tesztni, hogy a felszállás rendesen működik-e, azaz hogy nem szállnak fel az utasok az arra alkalmatlan Állomáson.

<b>Teszt-eset neve</b>	Váltó vált
<b>Rövid leírás</b>	A megfelelő pálya elindítása után, a tesztelő a megfelelő ágba váltja a váltót, amit ezután a szerelvény rövid időn belül elér és megfelelően továbbhalad rajta.
<b>Teszt célja</b>	Tesztni, hogy a váltó megfelelően vált-e, a szerelvények helyesen átmennék-e rajta

<b>Teszt-eset neve</b>	Váltó nem tud váltani
<b>Rövid leírás</b>	A megfelelő pálya elindítása után egy szerelvény rövid időn belül elér egy váltót, ezután a tesztelő megpróbálja átkapcsolni az adott váltót, ami nem sikerül neki.
<b>Teszt célja</b>	Tesztelni, hogy ha vonat halad át a váltón, akkor a váltó nem kapcsolható.

<b>Teszt-eset neve</b>	Alagútszájnál felrobban
<b>Rövid leírás</b>	A játék elindul, betöltődik egy teszt pálya, aminek a terepasztalán található BeSin, Sin és AlagutSzaj. Utána jön a BeSinen egy vonat. A vonat halad a Sinen, eléri az AlagutSzajat, és felrobban. A játékos veszít.
<b>Teszt célja</b>	Tesztelni, hogy a vonat meg nem épített AlagutSzaj-ra lépve felrobban és a játékos veszít.

<b>Teszt-eset neve</b>	Alagútszáj módosít
<b>Rövid leírás</b>	A játék elindul, betöltődik egy teszt pálya, aminek a terepasztalán található BeSin, Sin és legalább kettő AlagutSzaj. Input érkezik az egyik AlagutSzajra, majd a másikra, mire megépül az alagút. Utána input érkezik az első AlagutSzaj-ra, mire ha az lerombolódik, mert az alagút üres.
<b>Teszt célja</b>	Tesztelni, hogy az AlagutSzaj építése illetve lebontása működik-e, az Alagut megépül és megszűnik-e rendesen.

<b>Teszt-eset neve</b>	Alagút épít
<b>Rövid leírás</b>	A játék elindul, betöltődik egy teszt pálya, aminek a terepasztalán található BeSin, Sin és legalább kettő AlagutSzaj. Input érkezik az egyik AlagutSzajra, majd a másikra, mire megépül az alagút.
<b>Teszt célja</b>	Tesztelni, hogy az AlagútSzájra kattintva helyesen működik az építés, azaz hogyha megépül mind a kettő, akkor létrejön egy alagút.

<b>Teszt-eset neve</b>	Alagúton átmegy
<b>Rövid leírás</b>	A játék elindul, betöltődik egy teszt pálya, aminek a terepasztalán található BeSin, Sin és legalább kettő AlagutSzaj. A két AlagutSzaj-ra input érkezik, ennek hatására megépül az alagút. Utána elindul egy vonat a BeSin-en, halad a Sin-en, elér egy AlagutSzaj-ra, belemegy az alagútba, majd a túloldalon kijön belőle.
<b>Teszt célja</b>	Tesztelni, hogy az alagút rendesen működik, azaz lehet benne haladni.

## 7.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása

A teszteléshez készíteni fogunk egy segédprogramot, amely tárolja az egyes teszt esetekhez tartozó beviteli parancsokat, és az elvárt kimeneteket. Majd a parancsokat lefuttatva ellenőrzi a program kimenetelét, és ez alapján képes kiértékelni, hogy a teszteset sikeres-e vagy sem.

## 7.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017.03.24. 20:00	3.5 óra	Dócs Krátky Varga	Ötletelés és dokumentum szerkesztése
2017.03.24. 21:30	2 óra	Szili	
2017.03.24. 23:00	0.5 óra	Sillye	Dokumentum ellenőrzése
2017.03.25. 15:30	1 óra	Krátky	Ötletelés, tesztelési terv
2017.03.25. 15:30	3 óra	Sillye	Ötletelés, tesztelési terv vázlata, megvalósítása
2017.03.25. 20:30	1,5 óra	Varga	Módosítások leírása, UML diagrammok módosítása
2017.03.26. 14:30	2 óra	Dócs	Módosítások, átolvasás
2017.03.26. 23:00	1 óra	Szili	Módosítások, átolvasás, formázás
2017.03.26. 23:00	0,5 óra	Varga	Dokumentum átnézése

## 8. Részletes tervezés

### 8.1 Osztályok és metódusok tervezése.

#### 8.1.1 Alagut

- **Felelősség**

Alagút megépültségének, és az alagút szájak illetve a köztük menő SínElemek számítartása.

- **Ősosztályok**

Nincs

- **Interfészek**

Nincs

- **Attribútumok**

- **-szajA: AlagutSzaj:** Az alagút egyik szája
- **-szajB: AlagutSzaj:** Az alagút másik szája
- **-allapot: AlagutAllapot:** · NincsAlagutSzaj, EgyAlagutSzaj, VanAlagut
- **-alagutSinek: ArrayList<Sin>:** · Az alagút belsejében lévő sínek tárolása

- **Metódusok**

- **+AddAlagutSzaj(AlagutSzaj a): void:** Alagút száj megépítése, ha kettő alagútszáj van akkor megépül az alagút, a két alagút száj között sinek jönnek létre. Elösször kiszámítódik egy út a két alagútszáj között. Ez a két alagút szájat összekötő egyenes lesz. Majd ezen út mentén sin-ek kerülenk az alagutSinek listába, majd ezek bekerülnek a Terepasztalba is.
- **+RemoveAlagutSzaj(AlagutSzaj a): void:** A paraméterül kapott AlagutSzaj kivétele az attribútumok közül. Ha két alagút száj volt megépítve, akkor az alagutat lebontja. Végigmegy az alagutSinek lista elemein, és kiszedi ezeket a Terepasztalból.

### 8.1.2 AlagutSzaj

- **Felelősség**

Számon tartja, hogy meg van-e építve. Kezeli az alagút szájra történő kattintást.

- **Ősosztályok**

SinElem->AlagutSzaj

- **Interfészek**

Nincs

- **Attribútumok**

- **-megepitve: Boolean:** Meg van-e építve az alagút ezen szája
- **-alagutSin: SinElem:** A kapcsolódó Sin

- **Metódusok**

- **+onInput(): void:** A kattintásra reagál az osztály. Ha meg van építve lebontja, ha nincs, akkor pedig megépíti az adott alagút szájat.
- **+leptet(Mozdony m, SinElem s): void:** Lépteti a mozdonyt, illetve felrobbantja a vonatot, ha nincs megépítve.
- **+getKovSinElem(elozo: SinElem): SinElem:** A kapott paramétertől függően visszaadja, hogy merre kell továbbhaladnia a mozdonynak.

### 8.1.3 Allomas

- **Felelősség**

Tárol egy Színt, melyet a ráérkező vonat lekérdezhet. Bizonyos állomásokon utasok is felszállhatnak. Az állomás feladata az utasok felszállítása a kocsikra a megfelelő feltételek esetén.

- **Ősosztályok**

SinElem->Allomas

- **Interfészek**

Nincs

- **Attribútumok**

- **-sinB: SinElem:** A következő SinElem
- **-szin: Szin:** A vagon eszerint dönti el, hogy kiürül-e
- **-ures: Boolean:** Jelzi, hogy a vagon üres-e.

- **Metódusok**

- **+leptet(Mozdony m, Sinelem s): void:** Lépteti a mozdonyt.
- **+getSzin(): Szin:** Visszaadja az állomás színét.
- **+getKovSinElem(elozo: SinElem): SinElem:** Visszaadja a kapott paramétertől függően, hogy a mozdonynak merre kell tovább haladnia.

### 8.1.4 App

- **Felelősség**

Felelőssége a view, controll és modell inicializálása.

- **Ősosztályok**

Nincs

- **Interfészek**

Nincs

- **Attribútumok**

- **Metódusok**

- **+main(String[] args): void:** létrehozza a Jatek, Menu, Timer osztályokat

### 8.1.5 BeSin

- **Felelősség**

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

- **Ősosztályok**

SinElem->BeSin

- **Interfészek**

Nincs

- **Attribútumok**

- **-vonatok: int**: A pályára beadandó vonatok száma.

- **Metódusok**

- **+leptet(Mozdony m, Sinelem s): void**: Lépteti a mozdonyt.

Fellrobbantja a vonatot, ha ki akar menni a pályáról.

- **+VonatBead(): void**: Véletlen időközönként vonatokat ad be a pályára.

Számolja, hogy mennyi idő telt el a legutóbbi beadott vonat beadása óta, és ha ez elér egy előre megadott időt, akkor új vonatot ad be a pályára. A Vonat vagonjai véletlenszerűen lesznek szeneskocsik, vagy utaskocsik.

### 8.1.6 Jatek

- **Felelősség**

Objektumok létrehozása: Terepasztal és az abban helyet foglaló Sínelemek betöltése fájlból.

Feladata a játék elvesztésének és megnyerésének a kezelése.

- **Ősosztályok**

Nincs

- **Interfészek**

Nincs

- **Attribútumok**

Nincs

- **Metódusok**

- **+start(String palya): void**: Létrehozza a Terepasztalt, és adott fájlból tölti be rá az elemeket.

- **+veszt(): void**: A játékos elveszti a játékot.

- **+nyer(): void**: A játékos megnyeri a játékot.

### 8.1.7 KeresztSin

- **Felelősség**

Pálya teljesítésének nehezítésére szolgál, több dologra kell figyelni a játékosnak mert a vonatok összeütközhetnek itt.

- **Ősosztályok**

SinElem -> KeresztSin

- **Interfészek**

Nincs

- **Attribútumok**

- **-sinB: SinElem**: Keresztsín egyik ága

- **-sinC: SinElem**: Keresztsín egyik ága

- **-sinD: SinElem:** Keresztsín egyik ága
- **Metódusok**
  - **+leptet(Mozdony m, SinElem s): void:** Lépteti a paraméterként kapott mozdonyt a jó irányba (A-ról C-re, B-ről D-re).
  - **+getKovSinElem(SinElem elozo): SinElem:** Visszaadja a következő SinElemet, az előző SinElem függvényében (A-ról C-re, B-ről D-re).

### 8.1.8 Menü

- **Felelősség**

A program egyes menüpontjait tárolja. Kezeli az egyes menüpontok kiválasztása esetén bekövetkező funkciókat.

- **Ősosztályok**

Nincs

- **Interfészek**

Nincs

- **Attribútumok**

- **-palya: String:** A kiválasztott pálya neve
- **-progress: int:** Tárolja meddig jutott el a játékos a pályákon

- **Metódusok**

- **+start(String palya): void:** Új játék indítása
- **+save(): void:** Elmenti az állást, hogy mennyi pályát nyertünk meg
- **+select(): int:** Visszatér a pálya számával, amit kiválasztunk
- **+load(): int:** Betölt egy állást, és visszatér azzal, hogy meddig jutottunk el.
- **+exit(): void:** Kilép a játékból

### 8.1.9 Mozdony

- **Felelősség**

Kérdezgeti az alatta álló SínElemet, hogy melyik lesz a következő SínElem (leptet).

Szól az első Kocsinak, hogy mozogjon (mozog). Megvizsgálja, hogy történt-e ütközés.

- **Ősosztályok**

Szerelveny->Mozdony

- **Interfészek**

Nincs

- **Attribútumok**

- **-utkozott: Boolean:** Alapesetben false, ha ütközik a vonat akkor true-ra állítódik

- **Metódusok**

- **+utkozik(): void:** Átállítja az utkozott változó értékét true-ra
- **+mozog(): void:** Rálép a következő SinElem-re, és mozgatja a mögötte lévő kocsit.

- **+utkozesVizsgal(): void:** Megnézi, hogy volt-e ütközés a SinEllemen amin áll

### 8.1.10 Sin

- **Felelősség**

Továbbirányítja a mozdonyt a következő SínElemre.

- **Ősosztályok**

SínElem->Sin

- **Interfészek**

Nincs

- **Attribútumok**

- **-sinB: SinElem:** A másik kapcsolódó SinElem.

- **Metódusok**

- **+getKovSinElem(elozo: SinElem): SinElem:** Eldönti a paraméterként kapott SínElemből, hogy melyik ágon jön a mozdony, és aszerint tér vissza a következő SínElem-el

### 8.1.11 SínElem

- **Felelősség**

Mozgásteret biztosít a vonatok számára: adott SínElemről jött Mozdonynak megmondja, hogy melyik SínElem következik. Számon tartja, hogy hány Szerelvénnyel tartózkodik rajta.

- **Ősosztályok**

Nincs

- **Interfészek**

Nincs

- **Attribútumok**

- **-sinA: SinElem:** Az egyik kapcsolódó SínElem
- **-szerelvenyek: int:** Számolja a rajta tartózkodó Szerelvénnyek számát

- **Metódusok**

- **+leptet(Mozdony m, SinElem s):** Lépteti a paraméterként kapott mozdonyt a jó irányba.

- **+elLep(): void:** Csökkentni a Szerelvénnyek számát

- **+raLep(UtasKocsi kocsi): void:** Növeli a Szerelvénnyek számát. Akkor hívódik meg, ha UtasKocsi lép a SínElem-re.

- **+raLep(SzenesKocsi kocsi): void:** Növeli a Szerelvénnyek számát.

Akkor hívódik meg, a SzenesKocsi lép a SínElem-re.

- **+onInput(): void:** A SínElem-re történő kattintást kezeli.

- **+getUtkozes(): Boolean:** Megnézi, hogy történt-e ütközés a SínElemen, vagyis több Szerelvénnyel tartózkodik-e itt, mint 1.

- **+getKovSinElem(elozo: SínElem): SínElem:** Visszaadja a következő SínElemet, a kapott paraméter alapján. Az egyes leszármazottak implementálják.

### 8.1.12 SzenesKocsi

- **Felelősség**

Ugyan az a felelőssége, mint az Ősosztályának, a Vagonnak. Az állomásnál nem csinál semmit.

- **Ősosztályok**

Szerelveny -> Vagon -> SzenesKocsi

- **Interfészek**

Nincs

- **Attribútumok**

Nincs

- **Metódusok**

Nincs

### 8.1.13 Szerelveny

- **Felelősség**

Számon tartja, az előző és az aktuális SínElemet, amin tartózkodik. Tárolja, hogy alagútban van-e. Jelre tovább mozog.

- **Ősosztályok**

Nincs

- **Interfészek**

Nincs

- **Attribútumok**

- **-sinElem: SinElem:** Melyik sínen található a szerelvény
- **-alagutban: Boolean:** Alagútban van-e az adott szerelvény vagy sem
- **-elozoSinElem: SinElem:** Melyik sínen volt utoljára
- **kovKocsi: Vagon:** Következő Vagon referenciája

- **Metódusok**

- **+getSin(): SinElem:** Visszatér a SinElemmel amin áll éppen a szerelvény
- **+alagutValt(): void:** Átkapcsolja a szerelvényt, hogy alagútban van-e vagy sem.
- **+alagutbanVan(): Boolean:** Visszatér az alagutban attribútum értékével

### 8.1.14 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 sínelemekek listája
- **-besinek: List<BeSin>:** A bemeneti sínpárok listája
- **-telivonatSzam: int:** a pályán utassal rendelkező vonatok száma

- **Metódusok**

- **+tick(): void:** Szól a Mozdonyoknak, hogy mozogjanak.
- **+init(): void:** Feltölti elemekkel a terepasztalt.
- **+onInput(Int x, Int y): void:** Inputra reagál.
- **+addAlagutSzaj(): void:** sinelemek listához ad egy AlagutSzaj-at
- **+addMozdony(Mozdony m): void:** mozdonyok listához ad egy Mozdonyt.
- **+addBeSin(BeSin b): void:** besinek listához ad egy BeSint.
- **+addSinElem(SinElem s): void:** sinelemek listához ad egy SinElemet.
- **+removeSinElem(SinElem s): void:** sinelemek listából eltávolít egy SinElemet
- **+getAlagut(): Alagut:** Visszaadja az alagutat.
- **+vonatKiurult(): void:** teliVonatSzam-ot csökkenti 1-el.

### 8.1.15 Timer

- **Felelősség**

Felelőssége a periodikus jelgenerálás. A játék időbeli szimulálásának alapja.

- **Ősosztályok**

Nincs

- **Interfészek**

Nincs

- **Attribútumok**

Nincs

- **Metódusok**

- **+start(): void:** Elindítja a jelgenerálási folyamatot.

### 8.1.16 UtasKocsi

- **Felelősség**

Ugyanaz a felelőssége, mint szülőjének, a Vagonnak. Ezenkívül tárolja a színét, ami alapján az utasok leszállnak a kocsiból, illetve tárolja, hogy vannak-e utasok a kocsin.

- **Ősosztályok**

Szerelveny -> Vagon -> UtasKocsi

- **Interfészek**

Nincs

- **Attribútumok**

- **-szín: Szín:** A kocsi színe. Ez alapján dönti el hogy a kocsi kiürül-e az Állomáson.
- **-ures: Boolean:** Tárolja, hogy utaznak-e a kocsiban

- **Metódusok**

- **+leszallit(Állomas a): void:** Állomáshoz érve hívódik meg a függvény. A kocsi eldönti, hogy az utasok leszállnak-e, vagyis hogy az előtte lévő kocsi üres-e, és a paraméterként kapott állomás színe megegyezik-e a kocsiéval.
  - **+felszall(): void:** Az állomásnál új utasok szállnak a kocsira.
  - **+getSzin(): Szín:** Visszaadja a kocsi színét.
  - **+getUres(): Boolean:** Visszaadja, hogy üres-e a kocsi, vagy sem.

### 8.1.17 Vagon

- **Felelősség**

Absztrakt osztály amely a mozdony után álló vagonokat testesíti meg. Feladata, hogy a következő SinElemre mozogjon, illetve értesítse a vonatban a következő vagont, hogy melyik SinElemre kell lépnie.

- **Ősosztályok**

Szerelveny -> Vagon

- **Interfészek**

Nincs

- **Attribútumok**

Nincs

- **Metódusok**

- **+mozog(SinElem kovSin): void**: frissíti az aktuális sínelementet, ahol tartózkodik, és mozgatja a következő vagont (rekurzív)

### 8.1.18 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**

- **-aktivAg: SinElem**: Aktív ág referenciája
- **-kimenetek: ArrayList<SinElem>**: Az összes kimenő SinElem referenciája

- **Metódusok**

- **+setAktualisAg(SinElem s): void**: aktivAg-at átállítja s-re
- **+kovAg(): SinElem**: A kimenetektől visszaadja az aktivAg utáni SinElementet.
- **+onInput(): void**: A bevitelre az aktivAg átáll a kovAg()-ra
- **+leptet(Mozdony m, SinElem s): void**: Tovább küldi a mozdonyt a jó irányba
- **+getKovSinElem(elozo: SinElem): SinElem**: Visszatér a váltó éppen aktuális ágával, vagy a bejövő ágával, a kapott pramétertől függően.

## 8.2 A tesztek részletes tervei, leírásuk a teszt nyelvén

### 8.2.1 BeSin működik

- Leírás

A Teszt célja ellenőrizni, hogy a BeSin működik, és jönnek rajta vonatok. Ehhez egy egyszerű pályát építünk, és a 2. tickben egy vonatot hozunk be a pályára.

- Ellenőrzött funkcionalitás, várható hibahelyek

Timer tick, Terepasztal vonatbead, BeSin vonatbead függény, várható hibahely a vonatbead függvényekben

- Bemenet

```
besinpalya.txt
besin b1
besin b2
normalsin n1
.
b1-a n1-a
n1-b b2-a
.
b1 2 m1 p
```

```
loadmap besinpalya.txt
play
tick 3
info m1
```

- Elvárt kimenet

```
besinpalya.txt palya kivalasztva
jatek elindult
3 tick megtortent
m1:
sinElem: n1
alagutban: false
elozoSinElem: b1
kovKocsi: k1,
utkozott: false
```

## 8.2.2 AlagútSzáj módosít

- **Leírás**

Ellenőrzük, hogy az alagút felépülése után ha az egyik alagutat elvesszük, akkor megszűnik-e az alagút

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Terepasztal onInputja, Alagut AddAlagutSzaj, RemoveAlagutSzaj függvénye, AlagutSzaj onInput függvénye.

Várható hibahelyek: Nem épül meg rendesen az alagút, nem szűnik meg rendesen az alagút amikor lebontjuk

- **Bemenet**

```
alaguttteszt.txt
besin b1
besin b2
normalsin n1
normalsin n2
alagutszaj a1
alagutszaj a2
.
b1-a n1-a
n1-b a1-a
n2-a a2-a
n2-b b2-a
.
b1 2 m1 p
```

```
loadmap alagutttesztt.txt
play
alagutszaj a1
alagutszaj a2
alagutszaj a1
info t1 alagut
```

- **Elvárt kimenet**

```
alaguttteszt.txt palya kivalasztva
jatek elindult
a1 alagutszaj megepitve
a2 alagutszaj megepitve
alagut megepitve
a1 alagutszaj lerombolva
alagut lerombolva
t1:
szajA: null
szajB: a2
allapot: EgyAlagutSzaj
```

### 8.2.3 Alagút épít

- Leírás

Megnézzük, hogy két alagútszaj megépítése után felépül-e az alagút rendesen.

- Ellenőrzött funkcionalitás, várható hibahelyek

Terepasztal onInputja, Alagut AddAlagutSzaj függvénye, AlagutSzaj oninputja.

Várható hibahely: nem jók a referenciák, nem épül meg az alagút

- Bemenet

```
alaguttteszt.txt
besin b1
besin b2
normalsin n1
normalsin n2
alagutszaj a1
alagutszaj a2
.
b1-a n1-a
n1-b a1-a
n2-a a2-a
n2-b b2-a
.
b1 2 m1 p
```

```
loadmap alaguttteszt.txt
play
alagutszaj a1
alagutszaj a2
info t1 alagut
```

- Elvárt kimenet

```
alaguttteszt.txt palya kivalasztva
jatek elindult
a1 alagutszaj megepitve
a2 alagutszaj megepitve
alagut megepitve
t1:
szajA: a1
szajB: a2
allapot: VanAlagut
```

### 8.2.4 Alagúton átmegy

- Leírás

Ellenőrizzük, hogy a vonat át tud-e menni az alagúton, ami meg van építve.

- Ellenőrzött funkcionalitás, várható hibahelyek

Mozdony mozog függvénye, AlagútSzáj léptet függvénye

Várható hibahelyek: Az alagútszájnál a referencia beállítása nem történik meg rendesen

- Bemenet

```
alagutteszt.txt
besin b1
besin b2
normalsin n1
normalsin n2
alagutszaj a1
alagutszaj a2
.
b1-a n1-a
n1-b a1-a
n2-a a2-a
n2-b b2-a
.
b1 2 m1 p
```

```
loadmap alagutteszt.txt
play
alagutszaj a1
alagutszaj a2
tick 5
info m1
```

- Elvárt kimenet

```
alagutteszt.txt palya kivalasztva
jatek elindult
a1 alagutszaj megepitve
a2 alagutszaj megepitve
alagut megepitve
5 tick megtortent
m1:
sinElem: as1
alagutban: true
elozoSinElem: a1
kovKocsi: k1,
utkozott: false
```

## 8.2.5 Vonat ütközik

- Leírás

A megfelelő pálya elindítása után, két vonat rövid időn belül ütközik egymással. A vonatok egymással szemben haladnak a pályán.

- Ellenőrzött funkcionalitás, várható hibahelyek

A két vonat valóban ütközik.

Várható hibahelyek: az *utkozesVizsgal* függvényben

- Bemenet

```
utkozes.txt
besin b1
besin b2
normalsin n1
normalsin n2
normalsin n3
normalsin n4
normalsin n5

.
b1-a n1-a
n1-b n2-a
n2-b n3-a
n3-b n4-a
n4-b n5-a
b2-a n5-b

.
b1 1 m1 p
b2 1 m2 p
```

```
loadmap utkozes.txt
play
tick 2
info m1 sinelem
info m2 sinelem
tick 1
```

- Elvárt kimenet

```
utkozes.txt palya kivalasztva
Jatek elindult
2 tick megtortent
m1:
sinElem: n2
m2:
sinElem n4
1 tick megtortent
Utkozes tortent
```

## 8.2.6 Állomásnál leszáll

- Leírás

A játék elindul, betöltődik egy teszt pálya, aminek a terepasztalán található BeSin, Sin és Állomás. Utána jön a BeSinen egy vonat, ami tartalmaz legalább egy Kocsit, a vonat halad a Sinen, amíg el nem ér egy olyan színű Állomást, ahol le tudnak szállni az utasok. Az utasok leszállnak, és lesz egy üres kocsija a vonatnak

- Ellenőrzött funkcionalitás, várható hibahelyek

Leszállás rendesen működik-e az Allomas és a Kocsi osztályban, azaz hogy helyesen értékeli-e ki az előtte lévő üres kocsik létét, majd helyesen ürül-e ki a Kocsi.

Várható hibahelyek: a *leszallit* függvényben

● **Bemenet**

```
allomas1.txt
besin b1
besin b2
normalsin n1
normalsin n2
normalsin n3
normalsin n4
normalsin n5
allomas a1 p 0
.
b1-a n1-a
n1-b n2-a
n2-b a1-a
a1-b n3-a
n3-b n4-a
n4-b n5-a
b2-a n5-b
.
b1 1 m1 p
```

```
loadmap allomas1.txt
play
tick 3
info m1-u1 ures
tick 1
info m1-u1 sinelem
info m1-u1 ures
```

● **Elvárt kimenet**

```
allomas1.txt palya kivalasztva
Jatek elindult
3 tick megtortent
m1-u1:
ures: false
1 tick megtortent
m1-u1:
sinElem: a1
m1-u1:
ures: true
```

### 8.2.7 Állomásnál nem száll le

- Leírás

A játék elindul, betöltődik egy teszt pálya, aminek a terepasztalán található BeSin, Sin és Állomás. Utána jön a BeSinen egy legalább kettő, különböző színű Kocsit tartalmazó vonat, a vonat halad a Sinen, amíg el nem ér egy Állomást, aminek a Szin-e nem egyezik meg az első nemüres Kocsiéval, de megegyezik bármelyik más Kocsi színével. Az utasok nem szállnak le, és megy tovább a vonat.

- Ellenőrzött funkcionalitás, várható hibahelyek

Leszállás rendesen működik-e az Allomas és a Kocsi osztályban, azaz hogy helyesen értékeli ki az előtte lévő üres kocsik létét, majd helyesen kiürülés nélkül továbbmegy.

Várhtó hibahelyek: a *leszallit* függvényben.

- Bemenet

```
allomas2.txt
besin b1
besin b2
normalsin n1
normalsin n2
normalsin n3
normalsin n4
normalsin n5
allomas a1 p 0
.
b1-a n1-a
n1-b n2-a
n2-b a1-a
a1-b n3-a
n3-b n4-a
n4-b n5-a
b2-a n5-b
.
b1 1 m1 kp
```

```
loadmap allomas2.txt
play
tick 6
info m1-u2 ures
info m1-u2 sinelem
tick 1
info m1-u2 ures
```

- Elvárt kimenet

```
allomas2.txt palya kivalasztva
Jatek elindult
4 tick megtortent
m1-u2:
ures: false
m1-u2:
sinElem: a1
1 tick megtortent
m1-u2:
ures: false
```

### 8.2.8 Állomásnál felszáll

- Leírás

A játék elindul, betöltődik egy teszt pálya, aminek a terepasztalán található BeSin, Sin és Állomás. Utána jön a BeSinen egy vonat, ami tartalmaz legalább egy Kocsit. A vonat halad a Sinen, amíg el nem ér egy Állomás-t, ahol leszállnak az utasok az első kocsiból. Utána halad a Sinen, és elér egy olyan Állomást, ahol fel akarnak szállni. Az utasok felszállnak, és megy tovább a vonat.

- Ellenőrzött funkcionalitás, várható hibahelyek

A felszállás rendesen működik-e, azaz hogy felszállnak az utasok az arra alkalmas Állomáson.

Várható hibahelyek: a *felszall* függvényben.

- Bemenet

```
allomas3.txt
besin b1
besin b2
normalsin n1
normalsin n2
normalsin n3
normalsin n4
allomas a1 p 0
allomas a2 p 1
.
b1-a n1-a
n1-b n2-a
n2-b a1-a
a1-b n3-a
n3-b a2-a
a2-b n4-a
b2-a n4-b
.
b1 1 m1 p
```

```
loadmap allomas3.txt
play
tick 5
info m1-u1 ures
tick 2
info m1-u1 ures
info m1-u1 sinelem
```

- Elvárt kimenet

```
allomas3.txt palya kivalasztva
Jatek elindult
4 tick megtortent
m1-u1:
ures: true
2 tick megtortent
m1-u1:
ures: false
m1-u1:
sinElem: a2
```

### 8.2.9 Állomásnál nem száll fel

- **Leírás**

A játék elindul, betöltődik egy teszt pálya, aminek a terepasztalán található BeSin, Sin és Állomás. Utána jön a BeSinen egy vonat, ami tartalmaz legalább egy Kocsit. A vonat halad a Sinen, amíg el nem ér egy Állomás-t, ahol leszállnak az utasok az első kocsiból. Utána halad a Sinen, és elér egy olyan Állomást, ahol nem akarnak felszállni. Az utasok nem szállnak fel, és megy tovább a vonat.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A felszállás rendesen működik-e, azaz hogy nem szállnak fel az utasok az arra alkalmatlan Állomáson.

Várható hibahelyek: a *felszáll* függvényben.

- **Bemenet**

```
allomas4.txt
besin b1
besin b2
normalsin n1
normalsin n2
normalsin n3
normalsin n4
allomas a1 p 0
allomas a2 k 1
.
b1-a n1-a
n1-b n2-a
n2-b a1-a
a1-b n3-a
n3-b a2-a
a2-b n4-a
b2-a n4-b
.
b1 1 m1 p
```

```
loadmap allomas4.txt
play
tick 5
info m1-u1 ures
tick 2
info m1-u1 ures
info m1-u1 sinelem
```

- **Elvárt kimenet**

```
allomas3.txt palya kivalasztva
Jatek elindult
4 tick megtortent
m1-u1:
ures: true
2 tick megtortent
m1-u1:
ures: true
m1-u1:
sinElem: a2
```

## 8.2.10 Alagútszájnál felrobban

- **Leírás**

A játék elindul, betöltődik a tesztpálya. Besinről a vonat rálép egy meg nem épült alagútszájra és megvizsgáljuk, hogy felrobban-e a vonat, illetve hogy elvesztjük-e a játékot.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Annak az esetnek a helyes működését ellenőrizzük, amikor a vonat meg nem épült alagútszájra lép. Hibának számít, ha nem kapunk visszajelzést a játék elvesztéséről.

- **Bemenet**

```
besin b1
alagutszaj s1
.
b1-a s1-a
.
b1 1 m1 p

loadmap szajnalrobban.txt
play
tick 2
```

- **Elvárt kimenet**

```
szajnalrobban.txt kivalasztva
jatek elindult
2 tick megtortent
vege vesztett
```

## 8.2.11 Váltó nem tud váltani

- **Leírás**

Játék indulás után tick, vonat rálép besinről a váltóra. A tesztelő próbálja átváltani azt, és megvizsgáljuk, hogy sikerült-e.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszt célja ellenőrizni, hogy a váltót valóban nem lehet e átváltani, ha éppen vonat áll rajta. Hibának számít, ha arról kapunk visszajelzést, hogy a váltó átváltott.

- **Bemenet**

```
besin b1
valto v1
normalsin n1
normalsin n2
besin b2
besin b3
.
b1-a v1-a
v1-b n1-a
n1-b b2-a
v1-c n2-a
n2-b b3-a
.
b1 1 m1 p
```

```
loadmap valto.txt
play
tick 2
valto v1
```

- **Elvárt kimenet**

```
valto.txt kivalasztva
jatek elindult
2 tick megtortent
```

### 8.2.12 Váltó vált

- Leírás

Játék indulása után a tesztelő átállítja a váltót, majd besinről jön a vonat, tick és megnézzük, hogy a várt ágra érkezett e a vonat.

- Ellenőrzött funkcionalitás, várható hibahelyek

A váltó átváltásának helyességét teszteljük. Hibának számít, ha vonatunk nem a várt sínrre érkezik.

- Bemenet

```
loadmap valto.txt
```

```
play
```

```
valto v1
```

```
tick 3
```

```
info m1 sinElem
```

- Elvárt kimenet

```
valto.txt kivalasztva
```

```
jatek elindult
```

```
v1 valto atváltott a c agba
```

```
3 tick megtortent
```

```
m1:
```

```
sinElem: n2
```

### 8.2.13 Játék nyer

- Leírás

Egyetlen teli utaskocsiból álló vonatot engedünk besinről egy azonos színű állomásra, és megnézzük, hogy mi történik amikor kiürül.

- Ellenőrzött funkcionalitás, várható hibahelyek

A teszt célja a nyerés észleléssének helyessége. Hiba, ha a kocsi kiürülése után helytelen visszajelzést kapunk a játék állapotával kapcsolatban.

- Bemenet

```
besin b1
```

```
allomas a1 p
```

```
normalsin n1
```

```
besin b2
```

```
.
```

```
b1-a a1-a
```

```
a1-b n1-a
```

```
n1-b b2-a
```

```
.
```

```
b1 1 m1 p
```

```
loadmap jateknyer.txt
```

```
play
```

```
tick 3
```

- Elvárt kimenet

```
jateknyer.txt kivalasztva
```

```
jatek elindult
```

```
3 tick megtörtént
```

```
vege nyert
```

### 8.2.14 Játék fut

- **Leírás**

A tesztelő kiválaszt egy pályát és a játék elindul. A teszt az első tick erejéig fut, és ha erről sikeres visszajelzés jön, akkor sikerült a teszt.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ellenőrizzük a pálya betöltésének, játék elindításának helyességét. Hibának számít, ha nem jön visszajelzés a betöltés sikerességéről, vagy a játék állapotáról (elindult). Továbbá az is probléma, ha az első tickről nem érkezik nyugta.

- **Bemenet**

```
besin b1
valto v1
normalsin n1
.
b1-a v1-b
v1-a n1-a
n1-b v1-c
.
b1 1 m1 p
```

```
load jatekfut.txt
play
tick
```

- **Elvárt kimenet**

```
jatekfut.txt kivalasztva
jatek elindult
1 tick megtortent
```

## 8.3 A tesztelést támogató programok tervez

A tesztelést támogató illetve kiértékelést segítő program a különböző tesztesetekre a bemeneti nyelv használatával a programnak megadja a bemeneteket, majd leellenőrzi a kimeneti nyelven érkezett adatokat, hogy megfelelnek-e az elvártaknak.

A pálya fájljait nem a program generálja, azok a tesztesetek alapján lesznek megírva, illetve a tesztelést segítő program nem nyitja meg külön a pályákat, csak utasítást ad a főprogramnak a loadmap parancsal.

## 8.4 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017.04.01. 14:00	3.5 óra	Sillye	Tesztesetek: Besín működik, Alagútszáj módosít, Alagút épít, Alagúton átmegy
2017.04. 01. 15:40	2 óra	Varga	Tesztesetek: Vonat ütközik, Állomásnál leszáll, Állomásnál nem száll le, Állomásnál felszáll, Állomásnál nem száll fel
2017.04.01. 16:30	2.5 óra	Dócs	Tesztesetek: Játék fut, Játék nyer, Váltó vált, Váltó nem tud váltani, Alagútszájnál felrobban
2017.04.01. 19:00	3 óra	Szili	Osztályleírások: Alagut, AlagutSzaj, Allomas, Mozdony, Sin, SinElem, Szerelveny, Valto
2017.04.01. 22:00	2.5 óra	Krátky	Osztályleírások: app, besin, jatek, menu, szeneskocsi, terepasztal, timer, utaskocsi, vagon
2017.04.02 01:00	0.5 óra	Varga	Osztályleírások átnézése
2017.04.02 23:00	0.5 óra	Szili	Dokumentum formázás

## 10. Prototípus beadása

### 10.1 Fordítási és futtatási útmutató

#### 10.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Alagut.java	4570 byte	2017.04.18 18:50	Alagut osztály
AlagutAllapot.java	120 byte	2017.04.17 01:46	AlagutAllapot enum
AlagutSzaj.java	3779 byte	2017.04.18 00:52	AlagutSzaj osztály
alagutteszt.txt	137 byte	2017.04.16 17:28	alagutteszt pálya
Allomas.java	2950 byte	2017.04.16 17:28	Allomas osztály
allomas1.txt	199 byte	2017.04.18 00:52	allomas1 pálya
allomas2.txt	199 byte	2017.04.16 17:28	allomas2 pálya
allomas3.txt	201 byte	2017.04.18 00:52	allomas3 pálya
allomas4.txt	201 byte	2017.04.18 00:52	allomas4 pálya
App.java	3307 byte	2017.04.18 00:52	App osztály
BeSin.java	3126 byte	2017.04.17 01:46	BeSin osztály
besinpalya.txt	71 byte	2017.04.18 00:52	besinpalya pálya
input.txt	1012 byte	2017.04.18 00:52	Input file a teszthez
Jatek.java	1281byte	2017.04.17 16:31	Jatek osztály
jatekfut.txt	82 byte	2017.04.17 01:46	jatekfut pálya
jateknyer.txt	96 byte	2017.04.17 01:46	jateknyer pálya
Keresztsin.java	2021 byte	2017.04.17 01:46	Kocsi osztály
Menu.java	2306 byte	2017.04.17 01:46	Menu osztály
Mozdony.java	2696 byte	2017.04.17 01:46	Mozdony osztály
output.txt	1955 byte	2017.04.18 00:52	Output file a teszthez
Sin.java	1843 byte	2017.04.16 18:21	Sin osztály
SinElem.java	3209 byte	2017.04.17 01:46	SinElem osztály
szajnalrobban.txt	51 byte	2017.04.16 17:28	szajnalrobban pálya
SzenesKocsi.java	1729 byte	2017.04.17 01:46	SzenesKocsi osztály
Szerelveny.java	3002 byte	2017.04.17 01:46	Szerelvény osztály
Szin.java	198 byte	2017.04.16 17:28	Szin enum
Terepasztal.java	6971 byte	2017.04.17 16:26	Terepasztal osztály
Tester.java	4697 byte	2017.04.18 00:52	Tester osztály
Timer.java	138 byte	2017.04.16 17:28	Timer osztály
UtasKocsi.java	3715 byte	2017.04.17 01:46	UtasKocsi osztály
utkozes.txt	182 byte	2017.04.18 00:52	utkozes pálya
Vagon.java	1017 byte	2017.04.16 17:28	Vagon osztály
Valto.java	4310 byte	2017.04.17 01:46	Valto osztály
valto.txt	138 byte	2017.04.17 01:46	valto palya

### 10.1.2 Fordítás

A fenti listában felsorolt fájlok felhasználásával az alábbi módon lehetséges bináris, futtatható kódot előállítani. Töltsük le, és telepítsük a Java Dev Kit-et a gépre, ha még nincs fent. Csomagoljuk ki az összes fájlt egy mappába. Nyissuk meg a command line-t, és a cd parancssal navigálunk ebbe a mappába: /TesterIO/src/. Az alábbi utasítással fordítható le a kód:

`javac *.java`

Ekkor a mappában megjelennek a fájlok binárisai(.class).

A tesztelést segítő program lefordításához hasonlóan járunk el a /TesterIO/ mappában.

(Probléma esetén: figyeljünk arra, hogy az OS környezeti változói között jelen legyen a javáé.)

### 10.1.3 Futtatás

Az előző pontban lefordított fájlok futtatásához navigálunk a command line-ban a /TesterIO/src/ mappába a cd parancssal, majd a `java App` parancsot használjuk.

A tesztelést segítő program futtatásához hasonlóan járunk el: `java Tester` a /TesterIO/ mappában.

## 10.2 Tesztek jegyzőkönyvei

### 10.2.1 BeSin működik

Tesztelő neve	Szili Péter
Teszt időpontja	2017.04.17 21:55

Tesztelő neve	Varga János, Krátky Gergő, Dócs Zoltán
Teszt időpontja	2017.04.15. 00:33
Teszt eredménye	NullPointerException, kocsit nem írja ki
Lehetséges hibaok	A BeSin nek nincs elozoSinEleme, az utolsó előtti kocsiig számolunk
Változtatások	Megvizsgáljuk, hogy az elozoSinElem nem null-e. Az utolsó kocsiig számolunk.

### 10.2.2 AlagútSzáj módosít

Tesztelő neve	Szili Péter
Teszt időpontja	2017.04.17 18:43

Tesztelő neve	Varga János, Krátky Gergő, Dócs Zoltán, Sillye Márk
Teszt időpontja	2017.04.15. 01:15
Teszt eredménye	NullPointerException
Lehetséges hibaok	Alagút állapota inicializálatlan.
Változtatások	Alagút default állapota: NincsAlagutSzaj

<b>Tesztelő neve</b>	Varga János, Krátky Gergő, Dócs Zoltán, Sillye Márk
<b>Teszt időpontja</b>	2017.04.15. 01:20
<b>Teszt eredménye</b>	NullPointerException
<b>Lehetséges hibaok</b>	Alagút állapotát nem állítjuk át sehol.
<b>Változtatások</b>	Átállítjuk az állapotot a megfelelő helyeken.

<b>Tesztelő neve</b>	Sillye Márk
<b>Teszt időpontja</b>	2017.04.16. 14:55
<b>Teszt eredménye</b>	Az alagút lerombolható, miközben vonat van benne, amint mozogni próbálna ütközik.
<b>Lehetséges hibaok</b>	Nem ellenőrzi az alagút foglaltságát az AlagútSzáj a lebontásnál.
<b>Változtatások</b>	Ellenőrizhetjük, hogy le lehet-e bontani.

### 10.2.3 Alagút épít

<b>Tesztelő neve</b>	Szili Péter
<b>Teszt időpontja</b>	2017.04.17 15:59

### 10.2.4 Alagúton átmegy

<b>Tesztelő neve</b>	Szili Péter
<b>Teszt időpontja</b>	2017.04.17:18:36

<b>Tesztelő neve</b>	Sillye Márk
<b>Teszt időpontja</b>	2017.04.16. 15:00
<b>Teszt eredménye</b>	A teszt sikeres, de a vonat több tick beadása esetén nem az elvárt módon viselkedik, megfordul az alagútban.
<b>Lehetséges hibaok</b>	Rossz felé mozog a vonat az alagútszájban.
<b>Változtatások</b>	Kijavítva a rossz irány.

### 10.2.5 Vonat ütközik

<b>Tesztelő neve</b>	Szili Péter
<b>Teszt időpontja</b>	2017.04.17 21:57

### 10.2.6 Állomásnál leszáll

<b>Tesztelő neve</b>	Szili Péter
<b>Teszt időpontja</b>	2017.04.17 21:59

### 10.2.7 Állomásnál nem száll le

<b>Tesztelő neve</b>	Szili Péter
<b>Teszt időpontja</b>	2017.04.17 22:09

### 10.2.8 Állomásnál felszáll

<b>Tesztelő neve</b>	Szili Péter
<b>Teszt időpontja</b>	2017.04.17 22:10

**10.2.9 Állomásnál nem száll fel**

<b>Tesztelő neve</b>	Szili Péter
<b>Teszt időpontja</b>	2017.04.17 22:12

**10.2.10 Alagútszájnál felrobban**

<b>Tesztelő neve</b>	Szili Péter
<b>Teszt időpontja</b>	2017.04.17 22:14

**10.2.11 Váltó nem tud váltani**

<b>Tesztelő neve</b>	Dócs Zoltán
<b>Teszt időpontja</b>	2017.04.16. 19:48

**10.2.12 Váltó vált**

<b>Tesztelő neve</b>	Dócs Zoltán
<b>Teszt időpontja</b>	2017.04.16. 20:58

<b>Tesztelő neve</b>	Dócs Zoltán
<b>Teszt időpontja</b>	2017.04.16. 20:35
<b>Teszt eredménye</b>	v1 valto atváltott a -97 agba
<b>Lehetséges hibaok</b>	ágak hibás kiírása (pl. int->char konverzió kimaradt, stb.)
<b>Változtatások</b>	Helyes konverzió.

<b>Tesztelő neve</b>	Dócs Zoltán
<b>Teszt időpontja</b>	2017.04.16. 20:50
<b>Teszt eredménye</b>	nem ad információt a mozdony
<b>Lehetséges hibaok</b>	kis- és nagybetűs utasítások inkonzisztens használata
<b>Változtatások</b>	ToLower() segítségével megoldva.

**10.2.13 Játék nyer**

<b>Tesztelő neve</b>	Dócs Zoltán
<b>Teszt időpontja</b>	2017.04.16. 19:33

<b>Tesztelő neve</b>	Dócs Zoltán
<b>Teszt időpontja</b>	2017.04.16. 19:25
<b>Teszt eredménye</b>	OutOfBounds
<b>Lehetséges hibaok</b>	Állomás tulajdonságainak beolvassásakor hiba, ha az Üres érték nincs megadva
<b>Változtatások</b>	Kezeljük a hibát: default értéke üres

**10.2.14 Játék fut**

<b>Tesztelő neve</b>	Dócs Zoltán
<b>Teszt időpontja</b>	2017.04.16. 19:23

<b>Tesztelő neve</b>	Dócs Zoltán
<b>Teszt időpontja</b>	2017.04.16. 19:06
<b>Teszt eredménye</b>	NullPointerException
<b>Lehetséges hibaok</b>	Valto új ágának hozzáadása helyett módosítani próbálunk.
<b>Változtatások</b>	Módosítás helyett hozzáadjuk az új ágat.

**10.3 Értékelés**

<b>Tag neve</b>	<b>Munka százalékban</b>
Dócs Zoltán	20%
Krátky Gergő	20%
Sillye Márk	20%
Szili Péter	20%
Varga János	20%

## 10.4 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017.04.08. 15:00	2 óra	Sillye	Sin SinElem Valto
2017.04.08. 17:00	2 óra	Dócs	Allomas BeSin Szin, Menü
2017.04.08. 19:00	2 óra	Varga	Szerelvény, Mozdony, UtasKocsi, SzenesKocsi
2017.04.08. 14:00	1 óra	Sillye	Sin SinElem Valto javítások
2017.04.10. 19:00	2 óra	Krátky	AlagutAllapot Alagut AlagutSzaj
2017.04.13. 00:00	2 óra	Szili	Terepasztal Jatek Timer
2017.04.14. 20:00	6.5 óra	Krátky Varga	Bemeneti file beolvasása, Input kezelése, Tesztelés
2017.04.14. 20:00	5.5 óra	Dócs	Bemeneti file beolvasása, Input kezelése, Tesztelés
2017.04.15. 01:00	1 óra	Sillye	Tesztelés
2017.04.16. 12:00	3 óra	Sillye	Tesztelés, Sin SinElem Valto osztályok javítása
2017.04.16. 16:00	2 óra	Krátky	Alagut, AlagutSzaj osztályok
2017.04.16. 19:20	1.5 óra	Dócs	Tesztelés: utolsó 4 teszeset
2017.04.16 20:00	1 óra	Szili	Tesztelés
2017.04.16. 23:30	0.5 óra	Krátky	Kommentezés, apró javítások
2017.04.17. 16:00	2 óra	Szili	Tesztelés, dokumentáció
2017.04.17. 19:00	6 óra	Sillye, Szili	Tesztprogram
2017.04.17. 20:00	5 óra	Varga	Tesztprogram
2017.04.17. 23:00	2 óra	Krátky	Tesztprogram, dokumentum
2017.04.17 23:00		Dócs	Fordítás, Futtatás alpontok