#### Követelmény, projekt, funkcionalitás

###### Bevezetés

* + 1. Cél

Ez a dokumentum bemutatja a feladatot, a projekt követelményeit, és hogy mi hogyan fog működni.

* + 1. Szakterület

Ez egy többszemélyes számítógépes játék megvalósítása, melyet két csapat játszik egymás ellen.

* + 1. Definíciók, rövidítések

**ill.** = illetve

**stb.**: satöbbi

**JDK**: Java Development Kit

**min.**: minimum

**random**: véletlenszerű

* + 1. Hivatkozások

Szoftver projekt laboratórium tárgyhonlap Szoftvertechnológia órai jegyzet és előadásdiák Programozás alapjai 3 órai jegyzet és előadásdiák https://[www.iit.bme.hu/targyak/BMEVIIIAB02](http://www.iit.bme.hu/targyak/BMEVIIIAB02)

* + 1. Összefoglalás

Továbbiakban bővebben ismertetésre kerülnek a készítendő szoftver sajátosságai, mind a leendő felhasználók szempontjából, mind a mi, fejlesztők szempontjából fontosnak tartott aspektusok. Továbbá ismertetésre kerülnek a szoftver megvalósításával kapcsolatos tervek.

###### Áttekintés

* + 1. Általános áttekintés

A játék szerkezete, alapvetően két nagy részre bontható: pálya elemei és a játékosok. A pálya elemei lehetnek rugalmas csövek, illetve valamiféle aktív elemek.

* napelemes vízemelő pumpa
* forrás
* ciszterna

Legalább 4 játékos lesz, 2 csapatba osztva. A két csapat a szerelők és a nomádok. Mindkét csapat különböző célokat tűz ki a játék során.

* + 1. Funkciók

A drukmákori sivatagon át bonyolult csőrendszer szállítja a vizet a hegyi forrásokból a sivatagon túl elterülő városok ciszternáiba. A víz a forrásoknál ered, ideális esetben áthalad valamilyen útvonalon a csőrendszeren és a ciszternákba töltődik. A csőrendszer egyszerű, elágazás nélküli csövekből és a csövekhez csatlakozó aktív elemekből. Ezek az elemek a forrás, a ciszterna, a napelemmel működő vízátemelő pumpa. A rendszer változtatható és bővíthető csövek mozgatásával, pumpák állításával, vagy ezen elemek újonnan hozzáadásával. Ez a csöveknél és a pumpánál más-más módon történhet.

Egy pumpa több, de véges számú csövet köt össze. Minden pumpán külön-külön állítható, hogy éppen melyik belekötött csőből melyik másik rákötött csőbe továbbítsa a vizet. Ilyenkor minden más rákötött cső el van zárva, oda nem folyik víz.

A pumpák véletlen időközönként el tudnak romlani, ez azt jelenti, hogy az adott pumpa nem tudja a vizet továbbítani, megszűnik a vízáramlás. Mindegyik pumpa rendelkezik egy víztartállyal, amit a víz átemelése közben használ átmeneti tárolóként. Az elromlott pumpáknál a víz ebben a tartályban marad mindaddig, amíg azt meg nem javítják. A pumpa csak akkor tud vizet pumpálni egy csőbe, ha a cső szabad kapacitása ezt lehetővé teszi. Ellenkező esetben egy másik csövet kell választani a feladatra.

A csőhálózat bővíthető, változtatható. A csövek rugalmasságának köszönhetően azokat szabadon le lehet csatlakoztatni a pumpákról, a hálózat egy másik részére átvinni, és ott rácsatlakoztatni egy másik pumpára. Így átalakítva, optimalizálva a hálózatot, vagyis a víz útját a lehető legrövidebbre szűkíteni. A ciszternáknál folyamatosan készülnek az új csövek, amelyek egyik vége a ciszternához kapcsolódik, a másik azonban szabad. A szabad végű csövekből a csőbe betáplált víz kifolyik a hálózatból, a homokba, így nem jut el a városi ciszternákhoz.

A csőhálózatot a szerelők tartják karban. Ők javítják meg az elromlott pumpákat, valamint ők állítják át a pumpákat, hogy mindig a lehető legtöbb víz tudjon áthaladni a hálózaton és jusson el a városi ciszternákig. Ha egy cső kilyukad, az adott lyukas cső megfoltozása is a szerelők feladata. A kilyukadt csövekből a víz kifolyik a homokba, a csövek másik végén lévő pumpához már nem jut belőle. A ciszternáknál lévő szabad csövekkel lehetséges a hálózat kapacitásának növelése a szerelők számára. A ciszternáknál magukhoz tudnak venni új pumpát is, amit egy cső közepén tudnak elhelyezni. A csövet ehhez ketté kell vágni, és a két újonnan keletkezett végét a pumpához kell csatlakoztatni.

A hálózaton élnek a nomád szabotőrök is, akik feladata a vízátvitel szabotálása, megakadályozása. A pumpákat tudják átállítani, hogy azok ne a legoptimálisabb irányba, ne a megfelelő csőbe továbbítsák a vizet. A nomádok másik tevékenysége a csövek kilyukasztása. Ilyenkor a lyukas csöveken a víz elfolyik a homokba.

Mivel a sivatag veszélyes hely, ezért a szerelők és a szabotőrök csak a csőhálózaton haladhatnak, nem léphetnek le a hálózat elemeiről. Ha egy pumpánál találkoznak, akkor ott kikerülhetik egymást, egy pumpán egyszerre többen is állhatnak. Azonban a csöveken már nem tudnak elmenni egymás mellett, egy csövön egyszerre csak egy ember állhat.

A játékot két csapat játssza: a szerelők csapata és a nomádok csapata. Legalább 2 tagnak kell lennie minden csapatban, vagyis összesen legalább 4 fő kell a program használatához. A nomádok (vagy másnéven szabotőrök) dolga, hogy minél több víz folyjon el a lyukakon a homokba, vagyis, hogy ne jusson el a ciszternákig. A szerelők pedig azon dolgoznak, hogy minél több vizet tudjanak átfolyatni a rendszeren, vagyis, hogy minél több jusson el a ciszternákba. Az a csapat nyer, amelyik a játék végére több vizet szerez.

* + 1. Felhasználók

A játékot egyszerre több felhasználó használja. A felhasználóknak a program használatához nincs szükségük különösebb előismeretre.

* + 1. Korlátozások

A játék fejlesztéséhez a Java könyvtárat kell alkalmazni és a felhasználói felületet a swing csomaggal kell megvalósítani.

* + 1. **Feltételezések, kapcsolatok** Szoftver projekt laboratórium feladatkiírás: https://[www.iit.bme.hu/file/11582/feladat](http://www.iit.bme.hu/file/11582/feladat)

###### Követelmények

* + 1. Funkcionális követelmények

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Azonosító** | **Leírás** | **Ellenőrzés** | **Prioritás** | **Forrás** | **Use-**  **case** |
| A-01 | A rendszerben vannak csövek, és a hozzájuk csatlakozó aktív elemek. | Megvizsgáljuk a hálózatot. | Alapvető | Feladatleírás | Mind |
| A-02 | A játékosok a  „Szerelő” és  „Nomád” karaktereket irányítják a vízhálózaton. | Másfajta karaktert nem tudunk irányítani. | Alapvető | Feladatleírás | Mind |
| A-03 | Egy pumpa több csövet is összeköthet, de egyszerre csak egy-egy aktív be- és kimenete lehet. | Megvizsgáljuk, hogy tényleg csak egy irányba folyik-e víz a pumpán át. | Alapvető | Feladatleírás | Mind |
| A-04 | Minden karakter a hálózaton halad, de egy csövön egyszerre csak egy karakter lehet. | Megvizsgáljuk, hogy van-e olyan cső ahol több karakter áll. | Alapvető | Feladatleírás | Lép |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A-05 | Minden karakter átállíthatja a pumpákat. | Megpróbálunk pumpát átállítani mindkét féle karakterrel. | Alapvető | Feladatleírás | Pumpa állít |
| A-06 | A „Szerelő” karakterek át tudják állítani a csöveket. | Átállítunk egy csövet, hogy máshova irányítsuk a vizet. | Alapvető | Feladatleírás | Csőállít ás |
| A-07 | A „Szerelő” karakterek meg tudnak foltozni csöveket. | Megpróbálunk megfoltozni egy lyukas csövet. | Alapvető | Feladatleírás | Csövet foltoz |
| A-08 | A „Nomád” karakterek ki tudják lyukasztani a csöveket. | Megpróbálunk kilyukasztani egy működő csövet. | Alapvető | Feladatleírás | Csövet lyukaszt |
| A-09 | A „Nomád” csapat akkor nyer, ha a játék végén a homokba ment több víz. | Ellenőrizzük a vízállást a játék végén. | Alapvető | Feladatleírás | Mind |
| A-10 | A „Szerelő” csapat akkor nyer, ha a játék végén a ciszternákba ment több víz. | Ellenőrizzük a vízállást a játék végén. | Alapvető | Feladatleírás | Mind |
| A-11 | A lyukas csövekből a víz a homokra folyik, nem jut el a cső végi pumpáig. | Ellenőrizzük, hogy egy lyukas cső utáni pumpában van-e víz. | Alapvető | Feladatleírás | Kör |
| F-01 | A pumpák valamilyen időközönként elromlanak. | Adott időközönként megnézzük, van- e rossz pumpa. | Fontos | Feladatleírás | Pumpa elromlik |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| F-02 | A pumpák a rajtuk lévő víztartályban tárolják ideiglenesen a vizet, ha adott cső már tele van. | Megpróbálunk egy teli csőbe vizet irányítani. | Fontos | Feladatleírás | Pumpa pumpál |
| F-03 | A ciszternáknál folyamatosan készülnek új csövek, melyek egyik vége a ciszternához csatlakozik, másik  vége szabad. | Vizet küldünk át egy ilyen szabad végű csövön, ellenőrizve, hogy a homokba folyik-e. | Fontos | Feladatleírás | Cső létrehoz |
| F-04 | A „Szerelő” karakterek a ciszternákra lépve vehetnek föl új pumpákat is, melyeket csövek közepére rakhatnak  le. | Fölveszünk egy új pumpát egy ciszternánál. | Fontos | Feladatleírás | Pumpa létrehoz |
| F-05 | Pumpákon kikerülhetik egymást a karakterek, azaz egyszerre több is tartózkodhat ott. | Több karakterrel egy pumpára lépünk. | Fontos | Feladatleírás | Lép |
| F-06 | Legalább 2-2 játékossal kell játszani. | Megpróbálunk 1-  1 játékossal elindítani egy játékot. | Fontos | Feladatleírás | Kör |
| O-1 | A vízhálózat bővíthető. | Bármelyik elemből plusz darabokat rakunk a hálózatba. | Opcionális | Feladatleírás | Mind |

* + 1. **Erőforrásokkal kapcsolatos követelmények**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Azonosító** | **Leírás** | **Ellenőrzés** | **Prioritás** | **Forrás** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| R-101 | A program java nyelven készül | bemutatás | Alapvető | Feladatki-írás |
| R-102 | Az elkészült program futásához JDK szükséges | bemutatás | Fontos | Feladatki-írás |
| R-103 | A program futtatásához ajánlott operációs rendszer  Windows 10 | bemutatás | Opcionális |  |
| R-104 | A program fő beviteli eszközei az egér és a billentyűzet. | bemutatás | Fontos | Csapat |

* + 1. Átadással kapcsolatos követelmények

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Azonosító** | **Leírás** | **Ellenőrzés** | **Prioritás** | **Forrás** |
| R-111 | Dokumentáció  átadása | bemutatás | alapvető | Feladatki-írás |
| R-112 | Szkeleton átadás | bemutatás | alapvető | Feladatki-írás |
| R-113 | Prototípus átadás | bemutatás | alapvető | Feladatki-írás |
| R-114 | Teljes program átadása | bemutatás | alapvető | Feladatki-írás |

* + 1. **Egyéb nem funkcionális követelmények**

Nincs ilyen fajta követelmény.

###### Lényeges use-case-ek

* + 1. Use-case leírások

|  |  |
| --- | --- |
| **Use-case neve** | Lép |
| **Rövid leírás** | Egy karakter lép egyet a hálózaton. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | Egy karakter lép egy csőről egy másikra, egy csőről egy  pumpára, vagy fordítva. |

|  |  |
| --- | --- |
| **Use-case neve** | Csövet foltoz |
| **Rövid leírás** | Egy karakter megjavítja a csövet. |
| **Aktorok** | Szerelő |
| **Forgatókönyv** | Egy lyukas csőre lépve a szerelő megfoltozza azt. |

|  |  |
| --- | --- |
| **Use-case neve** | Pumpát javít |
| **Rövid leírás** | Egy karakter kijavít egy pumpát. |
| **Aktorok** | Szerelő |
| **Forgatókönyv** | Egy rossz pumpára lépve a szerelő kijavítja azt. |

|  |  |
| --- | --- |
| **Use-case neve** | Csövet lyukaszt |
| **Rövid leírás** | Egy karakter kilyukaszt egy csövet. |
| **Aktorok** | Nomád |
| **Forgatókönyv** | Egy nomád egy jó csőre lépve kilyukasztja azt. |

|  |  |
| --- | --- |
| **Use-case neve** | Pumpa elromlik |
| **Rövid leírás** | Egy pumpa elromlik. |
| **Aktorok** | Rendszer |
| **Forgatókönyv** | Adott időközönként véletlenszerűen elromlik egy pumpa. |

|  |  |
| --- | --- |
| **Use-case neve** | Kör |
| **Rövid leírás** | Lejátszódik egy kör a játékból. |
| **Aktorok** | Rendszer |
| **Forgatókönyv** | Véletlenszerűen elronthat egy pumpát, levon egy kört a játékidőből, engedi lépni a játékosokat, létrehoz a ciszternáknál új csöveket és pumpákat, figyeli, hogy véget  ért-e a játék. |

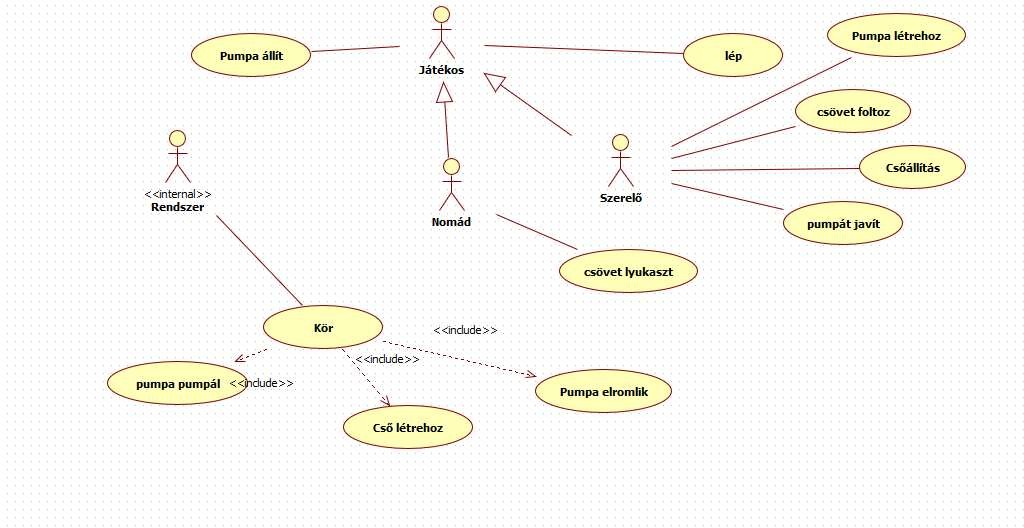
|  |  |
| --- | --- |
| **Use-case neve** | Pumpa pumpál |
| **Rövid leírás** | A csövekbe tesz vizet. |
| **Aktorok** | Rendszer |
| **Forgatókönyv** | A pumpa egyik csőből a másikba, vagy a rajta lévő  víztartályba tesz vizet, avagy fordítva. |

|  |  |
| --- | --- |
| **Use-case neve** | Pumpa állít |
| **Rövid leírás** | Átállít egy pumpát |
| **Aktorok** | Játékos |
| **Forgatókönyv** | Átállítja, hogy a pumpához kötött csövek közül melyiken  folyjon a víz. |

|  |  |
| --- | --- |
| **Use-case neve** | Pumpa létrehoz |
| **Rövid leírás** | Új pumpát hoz létre. |
| **Aktorok** | Szerelő |
| **Forgatókönyv** | A szerelő adott mezőkre lépve kap egy új pumpát. |

|  |  |
| --- | --- |
| **Use-case neve** | Csőállítás |
| **Rövid leírás** | Átállít egy csövet. |
| **Aktorok** | Szerelő |
| **Forgatókönyv** | A szerelő egy csőre lépve átállíthatja, hogy az a cső honnan  hová vezet. |

|  |  |
| --- | --- |
| **Use-case neve** | Cső létrehoz |
| **Rövid leírás** | Új csövet hoz létre. |
| **Aktorok** | Rendszer |
| **Forgatókönyv** | A ciszternáknál a rendszer új csövet hoz létre, amelyeknek  egyik vége szabad. |

* + 1. **Use-case diagram**

###### Szótár

**beállítás**: a szerelők egyik tevékenysége, a pumpa beállított állapotba kerül

**beállított pumpa**: a pumpa a megfelelő csőbe továbbítja a vizet

**ciszterna**: az átfolyatott víz végső állomása, a városok vízgyűjtője

**csapat**: a játékot két csapat játszhatja: a szerelők és a nomádok csapata. Mindkét csapat min. 2 főből állhat

**csatlakoztatás**: a cső egyik végének rácsatlakoztatása vagy lecsatlakoztatása egy pumpáról

**cső**: a játékmező egyik típusa, interaktálható elem

**csőrendszer**: másnéven csőhálózat, vagy hálózat. A játékpálya, ezen belül mozognak a játékosok. Két típusú elemből áll: cső és pumpa

**elállítás**: a nomádok egyik tevékenysége, a pumpák elállítódnak

**elállított puma**: a nomádok pumpa elállításának következménye, az adott pumpa nem tudja a vizet továbbítani

**elromlott pumpa**: random időközönként, random pumpánál következik be, ha elromlott, akkor nem tudja a vizet továbbítani

**elzárt cső**: a pumpáknál csak egy csövön érkezhet a víz és egy csövön távozhat, a többi rákötött cső el van zárva

**hegyi forrás**: innen érkezik a víz a csőrendszerbe

**homok**: a nomádok által kifolyatott víz ide kerül

**játékos**: a program felhasználója, valamelyik csapat tagja: vagy szerelő, vagy nomád

**kapacitás növelés**: a szerelők egyik tevékenysége, a szabad csövekkel és pumpákkal növelni tudják a hálózatot és annak a víz átfolyató képességét

**karbantartás**: a szerelők tevékenységeinek összessége

**kikerülés**: a játékosok mozgás közben a pumpákon el tudnak menni egymás mellett

**lyukas cső**: a nomádok lyukasztásának eredménye, a cső nem tudja továbbítani a vizet, kifolyik a rendszerből

**lyukasztás**: a nomádok egyik tevékenysége, a csövek kilyukasztása **megfoltozás**: a szerelők egyik tevékenysége, a csövek megfoltozása **megfoltozott cső**: a csövön lévő lyuk eltűntetése

**megjavítás**: a szerelők egyik tevékenysége, az elromlott pumpák megjavítása

**mozgás**: a játékosok mozgása, ami csak a csőrendszeren belül történhet. A pumpákon kikerülhetik egymást, a csöveken nem

**nomád**: másnéven szabotőr. Az egyik csapat megnevezése, minimum 2 főből áll. A nomádok a csöveket kilyukasztják és a pumpákat elállítják. Céljuk, hogy minél több víz folyjon el a lyukakon.

**nyertes**: az a csapat nyer, amelyik több vizet tudott megszerezni

**puma**: a játékmező egyik típusa, interaktálható elem

**sivatag**: játéktér, ahol a két csapat versenyzik

**szabad cső**: olyan cső, amivel a hálózat szabadon bővíthető, egyik vége a cisztárnához csatlakozni, a másikat a szerelők tudják bekötni

**szabotálás**: a nomádok tevékenységeinek összessége

**szerelő**: az egyik csapat megnevezése, minimum 2 főből áll. A szerelők feladata az elromlott pumpákat megjavítása és beállítása, valamint a kilyukadt csöveket megfoltozása. Céljuk, hogy minél több vizet juttassanak el a városok ciszternáiba a forrásokból.

**új pumpa**: olyan pumpa, amivel a hálózat szabadon bővíthető, a szerelők kettévágnak egy csövet és a vágásnál keletkezett két végét rácsatlakoztatják

**víz**: a forrásoktól, a csőrendszeren át, a ciszternákig folyik a H2O. A játékosok célja minél nagyobb mennyiségű H2O megszerzése

**víztartály**: a pumpák belsejében található, az átfolyatás közben átmeneti tárolóként szolgál

###### Projekt terv

|  |  |  |
| --- | --- | --- |
| **Hét** | **Feladat** | **Határidő** |
| 2 | Követelmények, projekt,  funkcionalistás | márc.13. |
| 3 | Analízis modell (1.változat) | márc.20. |
| 4 | Analízis modell (2.változat) | márc.27. |
| 5 | Szkeleton tervezése | ápr.3. |
| 6 | Szkeleton elkészítése | ápr.17. |
| 7 |
| 8 | Prototípus koncepciója  Szkeleton bemutatása | ápr.24. |
| 9 | Részletes tervek | máj.03. |
| 10 | Prototípus elkészítése | máj.08. |
| 11 | Grafikus változat tervei  Prototípus bemutatása | máj.15. |
| 12 | Grafikus változat elkészítése | máj.31. |
| 13 |
| 14 | Egyesített dokumentáció  Grafikus változat bemutatása | jún.2. |

|  |  |
| --- | --- |
| **Név** | **feladatkör** |
| Kis | software developer, full stack |
| Nyilas | scrum master, software developer, back end |
| Zsigmond | software developer, back end |
| Farkas | software developer, back end |
| Sőregi | software developer, front end |

Felhasznált eszközök:

* GitHub (Git)
* IntelliJ (IDEA)
* Microsoft Word

**Fejlesztési technika:** agilis megközelítés

Issuekat ír ki a scrum master, és ha lehet akkor az egyeztetett feladatkörök alapján kiosztja a feladatokat. Amennyiben ez nem lehetséges munkaterhelés alapján ossza ki a feladatokat, és a scrum elején ezt megbeszéljük és kiosztjuk. A meghatározott sprint időtartamának végén a feladatok bezárásra kerülnek és a main branchre tesszük a változtatásokat.

#### Analízis modell kidolgozása

###### Objektum katalógus

* + 1. Csőrendszer (Map)

A csőrendszer a sivatagon át szállítja a vizet a hegyi forrástól a városi ciszternákig. Elágazásmentes csövekből és napelemmel működő pumpákból áll össze a hálózat. A rendszer változtatható és bővíthető csövek mozgatásával, pumpák állításával, vagy ezen elemek hozzáadásával.

* + 1. Ciszterna (Cistern)

A ciszternák gyűjtik össze a forrástól a csőhálózaton keresztül szállított vizet. Itt folyamatosan új csövek készülnek, ezek egyik vége egyből a ciszternákhoz csatlakozik, a másik végük szabad. Új pumpák is találhatók itt, ezeket azonban csak a szerelők tudják beépíteni a hálózatba.

* + 1. Cső (Pipe)

A cső a csőrendszer egyik fajta alkotóeleme. Egyszerű, elágazásmentes. A csöveket szabadon le lehet csatlakoztatni a pumpákról, a hálózat egy másik részére átvinni, és ott rácsatlakoztatni egy pumpára. Ha egy cső egyik vége szabad, akkor a rajta átmenő víz a homokba folyik. Ha egy cső kilyukad (vagy kilyukasztják), akkor a víz kifolyik, és a másik végén lévő pumpához már nem tud eljutni.

* + 1. Forrás (WaterSpring)

A víz a forrásból ered. A forráshoz lehet csatlakoztatni a csőrendszer elemeit. A csőrendszeren foglalatoskodó emberek innen indulnak dolgukra.

* + 1. Pumpa (Pump)

A pumpa a csőrendszer másik típusú alkotóeleme. Egy pumpa több csövet is összeköthet, de egyszerre csak egy bemenete és egy kimenete lehet. Egyenként állítható, hogy a rá csatlakoztatott csövek közül melyeken folyjon tovább a víz. A többi bekötött cső ilyenkor elzárt állapotban van. Véletlen időközönként, véletlenszerűen el tudnak romlani a pumpák.

Ha egy pumpa elromlik, akkor megszűnik a rajta áthaladó vízfolyás.

Minden pumpának van egy belső víztartálya, amely az átfolyatott víz átmeneti tárolójaként funkcionál. Csak akkor tudja egy csőbe továbbítani a vizet, ha annak a csőnek a kapacitása ezt engedi.

* + 1. Szerelő (Plumber)

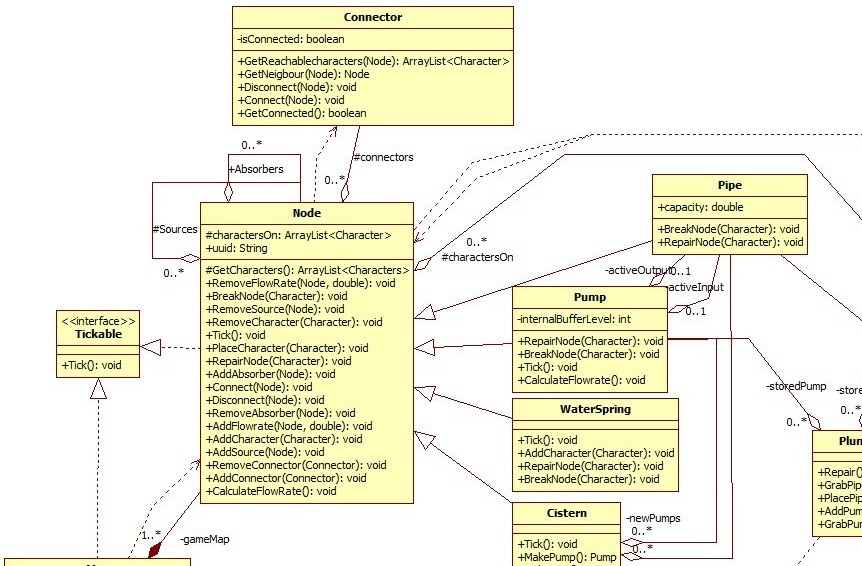
A szerelők feladata a csőhálózat karbantartása. Az elromlott pumpákat megjavítják és a kilyukadt csöveket megfoltozzák. Az ő feladatuk a pumpák beállítása is, amiket úgy állítanak be, hogy mindig a lehető legnagyobb vízmennyiség tudjon átfolyni a rendszeren.

A szabadon lévő csövek segítségével növelni tudják a vízhálózat kapacitását. A ciszternáknál új pumpákhoz juthatnak, amit egy cső közepénél, azt kettévágva tudnak beépíteni a rendszerbe. Csak a csőhálózaton tudnak haladni. A pumpáknál ki tudják kerülni egymást, vagy a nomádokat, de a csöveken már nem. Céljuk, hogy minél több vizet juttassanak el a városok ciszternáiba. Legalább két szerelő tevékenykedik a csőrendszeren.

* + 1. Nomád szabotőr (Nomad)

A nomádok a csöveket kilyukasztják és a pumpákat elállítják. Csak a csőhálózaton tudnak haladni. A pumpáknál ki tudják kerülni egymást, vagy a szerelőket, de a csöveken már nem. Céljuk, hogy minél több víz folyjon el a lyukakon. Legalább két nomád szabotálja a vízhálózatot.

###### Statikus struktúra diagramok



***Osztályok leírása***

* + 1. Character
* Felelősség

A játékban szereplő karaktereket megvalósító osztályok ősosztálya. Tárolja, hogy a pálya mely elemén (Node) tartózkodik, és az összes karakter közös viselkedését megvalósítja.

* Ősosztályok

Nincs ősosztálya.

* Interfészek

Nem implementál interfészt.

* Attribútumok
  + **name**: karakter neve
  + **uuid**: karakter egyedi azonosítója
  + **standingOn**: eltárolja hogy a karakter a Map mely elemén, melyik Node-on áll.
* Metódusok
  + **void placeTo(Node startPoint)**: A karaktert lehelyezi a pálya egy elemére, a játék kezdetekor.
  + **void moveTo(Node moveTo)**: A karakter mozgását valósítja meg, az egyik pályaelemről(Node) a másikra.
  + **void setPump()**: A karakter beállítja a Pumpát, amin áll.
    1. Cistern
* Felelősség

A játékban szereplő ciszternákat megvalósító osztály. Ide gyűjtik a vizet a szerelők, és készülnek itt új csövek is.

* Ősosztályok

Node  Cistern

* Interfészek

Tickable interfészt implementálja.

* Attribútumok

Nincsenek saját attribútumai.

* Metódusok
  + **void tick()**: időközönként számolja a beérkezett vizet.
  + **Pipe makePipe()**: új csöveket készít
  + **Pump makePump()**: új pumpákat készít
    1. Connector
* Felelősség

Ezt az osztályt arra használjuk, hogy két Node objektumot csatlakoztassunk egymáshoz.

* Ősosztályok

Nincsenek ősosztályok.

* Interfészek

Nem implementál interfészt.

* Attribútumok
  + **node1, node2:** a csatlakoztatott Node-ok.
  + **isConnected:** azt mutatja, hogy éppen csatlakoztatva van-e.
* Metódusok
  + **Node getNeighbour(Node self)**: visszaadja a szomszédját annak, amelyiket input-ként adjuk.
  + **ArrayList<Character> getReachableCharacters(Node self)**: visszaad egy listát az erről az objektumról elérhető karakterekről.
  + **void connect(Node node)**: összekapcsolja ezt az objektumot egy másikkal.
  + **void disconnect(Node node)**: szétkapcsolja ezt és egy másik obektumot.
  + **boolean isConnected()**: visszaadja az isConnected változó értékét.
    1. Map
* Felelősség

A játékteret megvalósító osztály. Node-okat, és a játékos karaktereket csoportosít.

* Ősosztályok

Nincs ősosztály

* Interfészek

Tickable interfészt implementálja.

* Attribútumok
  + **waterArrived:** Nyilvántartja, hogy mennyi víz folyt be a ciszternába.
  + **waterLost:** Nyilvántartja, hogy mennyi víz folyt el a lyukakon a sivatagi homokra.
* Metódusok
  + **void FillMap ()**:Map feltöltése elemekkel és játékososkkal.
  + **void addNode (Node node)**: Új pályaelem (Node) hozzáadása a pályához.
  + **void addPlayer(Character c):** Új játékos hozzáadása.
  + **void Tick():** Időközönként (tick) számolja a ciszternába folyt, és a homokba elfolyt víz mennyiséségt.
  + **void PlayerLostHandler(Character c):** Ha egy player alatt eltűnik a node akkor ez a metódus fogja visszarakni egy forrásra.
    1. Node
* Felelősség

A Node osztály egy absztrakt osztály, ezt implementálja minden osztály, amiknek az elemeit a játéktérre tehetjük.

* Ősosztályok

Nincs ősosztály.

* Interfészek

Tickable interfészt implementálja.

* Attribútumok
  + **logger:** egy logger ehhez az osztályhoz.
  + **uuid:** karakter egyedi azonosítója.
  + **charactersOn:** a jelenleg ezen az objektumon álló karakterek listája.
  + **connectors:** a jelenleg ehhez az objektumhoz csatlakoztatott connector-ok.
  + **neighbours:** a jelenleg ezzel az objektummal szomszédos objektumok listája.
  + **sources:** azok az objektumok, ahonnan folyik víz ebbe az objektumba.
  + **absorbers:** azok az objektumok, ahová folyik víz ebből az objektumból.
  + **maxCharacters:** az ezen az objektumon maximálisan elfogadott karakterek száma.
  + **maxConnections:** az ehhez az objektumhoz maximálisan csatlakoztatható elemek száma.
  + **isBroken:** hogy éppen működik-e az objektum.
  + **flowRate:** hogy éppen mennyi víz folyik keresztül az objektumon.
* Metódusok
  + **String getUuid()**: visszaadja az uuid attribútum értékét.
  + **void addAbsorber(Node absorber):** hozzáad az absorber-ekhez egy újat.
  + **void removeAbsorber(Node absorber):** kivesz egyet az absorber-ek közül.
  + **void addSource(Node source):** hozzáad egy source-okhoz egy újat.
  + **void removeSource(Node source):** kivesz a source-ok közül egyet.
  + **void addConnector(Connector connector):** hozzáad a connector-okhoz egy újat.
  + **void removeConnector(Connector connector):** kivesz a connector-ok közül egyet.
  + **ArrayList<Character> getCharactersOn():** visszaadja a charactersOn listát.
  + **void placeCharacter(Character character):** elhelyez egy karaktert az objektumon.
  + **void addCharacter(Character character):** hozzáad egy karaktert a charactersOn listához.
  + **void removeCharacter(Character character):** kivesz egy karaktert a charactersOn listából.
  + **void repairNode(Character character):** megjavítja az objektumot.
  + **void breakNode(Character character):** elrontja az objektumot.
  + **void addFlowRate(Node source, double excededFlow):** hozzáad a flowRate-hez.
  + **void removeFlowRate(Node source, double flowRate):** elvesz a flowRate-ből.
  + **void setFlowRate(double flowRate):** beállítja a flowRate-et.
  + **void tick():** időközönként számolja a flowRate-et.
  + **void connect(Node node):** hozzácsatlakoztat az objektumhoz egy másikat.
    1. Nomad
* Felelősség

A szabotőr nomád-karaktereket megvalósító osztály. A nomád ki tudja lyukasztani a csöveket (Pipe) és el tudja állítani a pumpákat.

* Ősosztályok

Character  Nomad .

* Interfészek

Nem implementál interfészt.

* Attribútumok
  + **name**: karakter neve
  + **uuid**: karakter egyedi azonosítója
  + **standingOn**: eltárolja hogy a karakter a Map mely elemén, melyik Node-on áll.
* Metódusok
  + **void placeTo(Node startPoint)**: A karaktert lehelyezi a pálya egy elemére, a játék kezdetekor.
  + **void moveTo(Node moveTo)**: A karakter mozgását valósítja meg, az egyik pályaelemről(Node) a másikra.
  + **void SetPump()**: A karakter beállítja a Pumpát, amin áll.
  + **void BreakNode()**: A Nomad kilyukasztja a Pumpát.
    1. Pipe
* Felelősség

Ez az osztály implementálja a játékban a csöveket. A csöveket el lehet rontani és meg lehet javítani, továbbá csatlakoztatni lehet őket más objektumokhoz (összesen maximum kettőhöz), és összesen 1 játékos állhat rajtuk.

* Ősosztályok

Node  Pipe

* Interfészek

Nem valósít meg interfészeket.

* Attribútumok
  + **capacity**: azt mutatja mennyi a kapacitása ennek a csőnek (vagyis azt, mennyi víz fér bele).
* Metódusok
  + **double getCapacity()**: visszaadja a capacity értékét, vagyis azt, mennyi víz fér még bele.
  + **void repairNode(Character character)**: megjavítja a csövet.
  + **void breakNode(Character character)**: kilyukasztja, elrontja a csövet.
    1. Plumber
* Felelősség

Ez az osztály a játékban lévő vízszerelők osztálya. A vízszerelők meg tudnak javítani csöveket és pumpákat, le tudnak rakni új csöveket és pumpákat, illetve beállíthatják a pumpákat, hogy a ciszternákba tereljék a vizet.

* Ősosztályok

Character  Plumber

* Interfészek

Nem valósít meg interfészeket.

* Attribútumok

Nincsenek saját attribútumai.

* Metódusok
  + **void repair()**: megjavítja a Node-ot, amin áll.
  + **void placePump()**: elhelyez egy pumpát oda, ahol áll.
  + **void movePipe()**: átállítja a csövet, amin áll.
    1. Pump
* Felelősség

Ez a játékban szereplő pumpák osztálya. Egy pumpa állítható, el tud romlani, és meg lehet javítani. Egy pumpán több játékos is állhat egyszerre.

* Ősosztályok

Node  Pump

* Interfészek

Tickable interfészt implementálja.

* Attribútumok
  + **internalBufferLevel**: ezt arra használjuk, hogy a flowRate-et számoljuk.
  + **activeInput**: ez a pumpa bemenetei közül tárolja azt, amelyik aktív.
  + **activeOutput**: a pumpa kimenetei közül tárorlja, amelyik aktív.
* Metódusok
  + **void repairNode(Character character)**: megjavítja a pumpát.
  + **void breakNode(Character character)**: elrontja a pumpát.
  + **void setActiveInput(Pipe p)**: beállítja az activeInput-ot.
  + **void setActiveOutput(Pipe p)**: beállítja az activeOutput-ot.
  + **void calculateFlowRate()**: kiszámolja mennyi víz megy át rajta.
  + **void tick()**: időközönként (random) elrontja a pupmpát, és számolja a flowRate-et.
    1. Tickable
* Felelősség

Ez az interfész minden objektumnak, aminek időközönként kell csinálnia valamit.

* Ősosztályok

Nincsenek ősosztályok.

* Metódusok
  + **void tick()**: az adott időközönként végzett tevékenységre való.
    1. WaterSpring
* Felelősség

A WaterSprring egy olyan speciális elem, amivel a ciszternát tudjuk feltölteni. Nem lehet sem elrontani, sem megjavítani, illetve nincs maximum kapacitása.

* Ősosztályok

Node  WaterSpring

* Interfészek

Tickable interfészt implementálja.

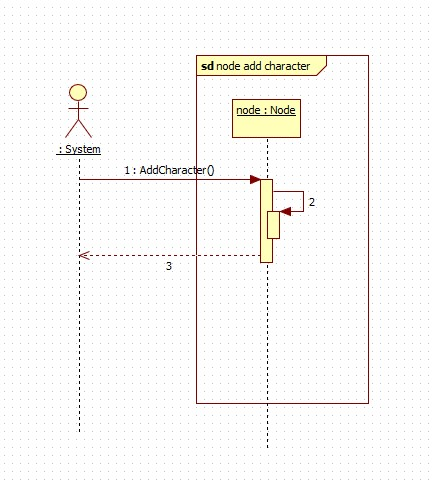
* Attribútumok

Nincsnek attribútumai.

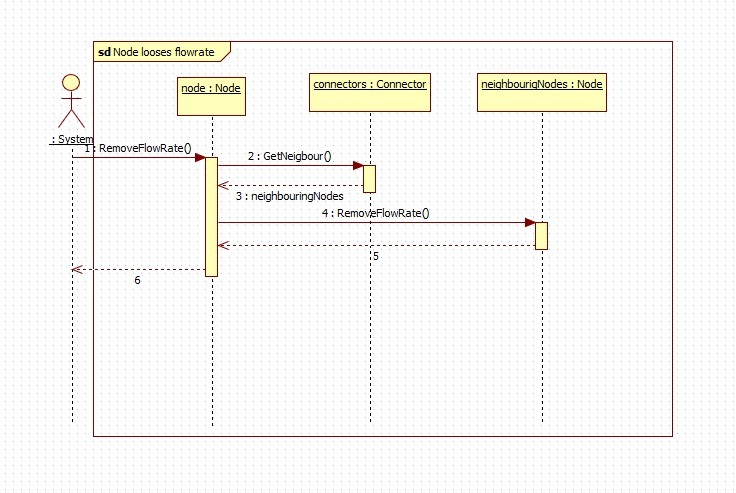
* Metódusok
  + **void addCharacter(Character character)**: hozzáad egy karaktert a charactersOn listához.
  + **void repairNode(Character character)**: beállítjuk, hogy ez nem lehetséges.
  + **void breakNode(Character character)**: beállítjuk, hogy ez nem lehetséges.
  + **void tick()**: időközönként számolja a flowRate-et.

###### Szekvencia diagramok

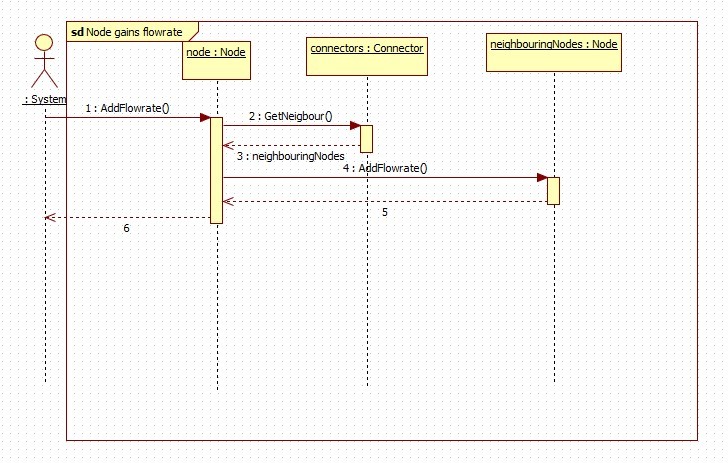
* + 1. Add character to Node



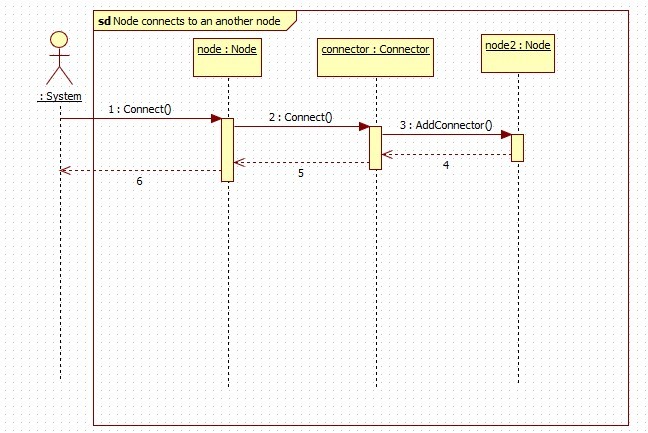
* + 1. **Node looses flowrate:**



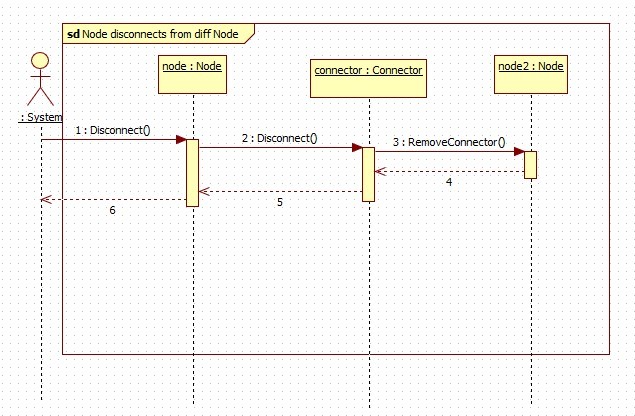
* + 1. Node gains flowrate



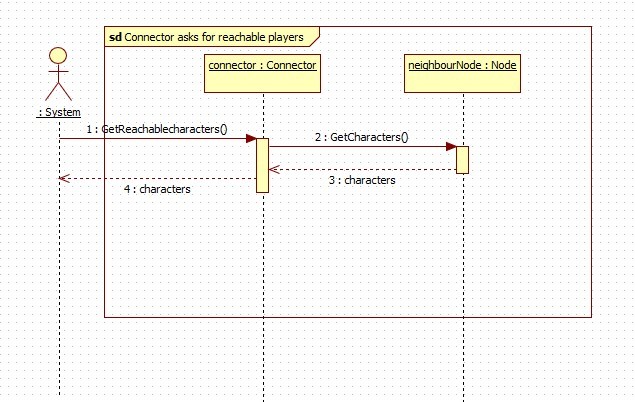
* + 1. **Node connects to diff Node**



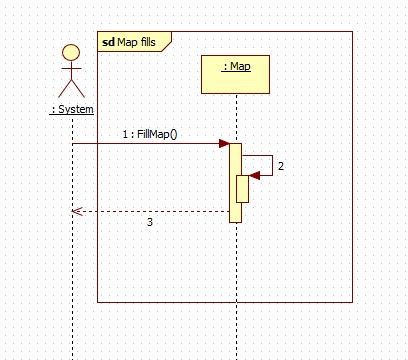
* + 1. Node disconnects from diff Node



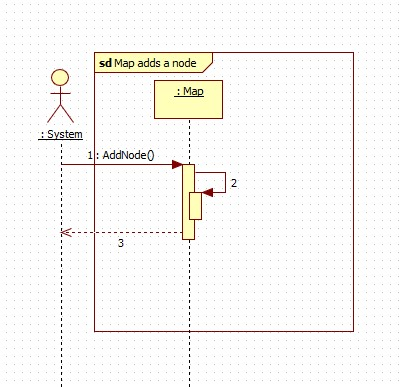
* + 1. **Connector gets neighbouring characters**



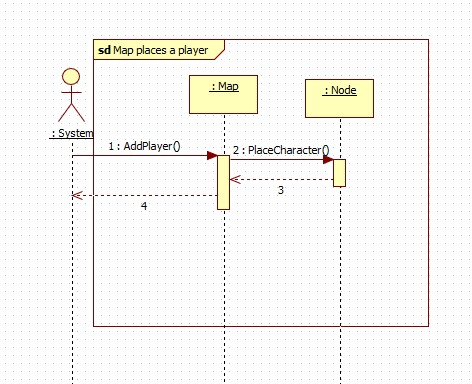
* + 1. Map initializes



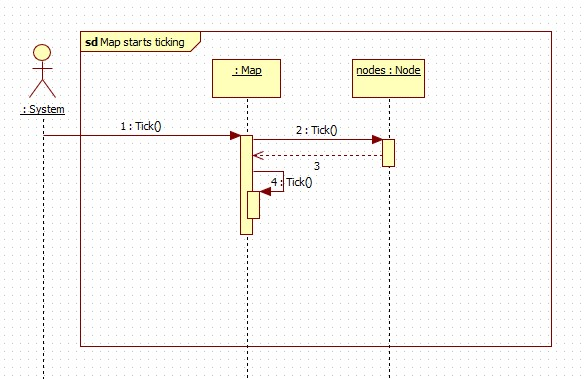
* + 1. **Map adds a node**



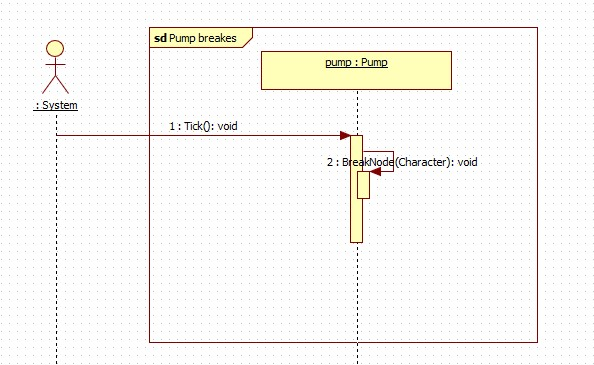
* + 1. Map places a character



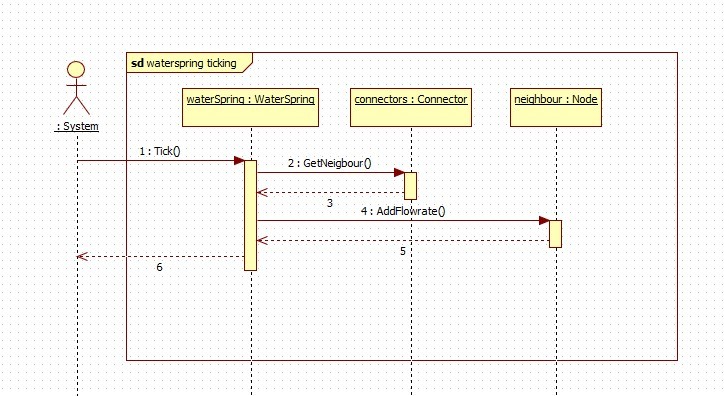
* + 1. **Map starts ticking**



* + 1. Pump breaks



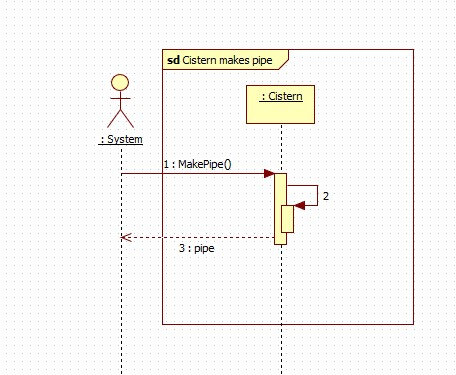
* + 1. **WaterSpring ticking**



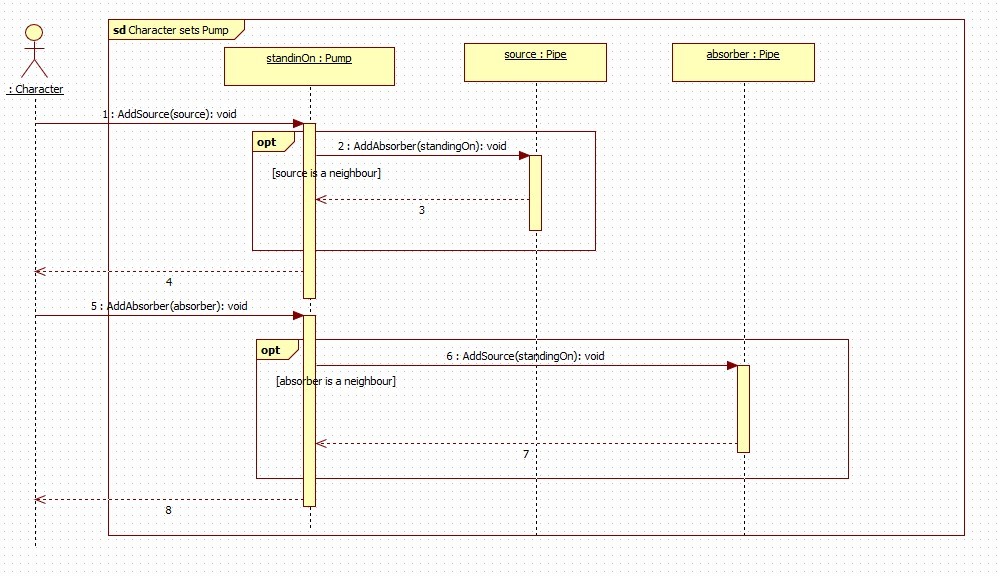
* + 1. Cistern makes Pump



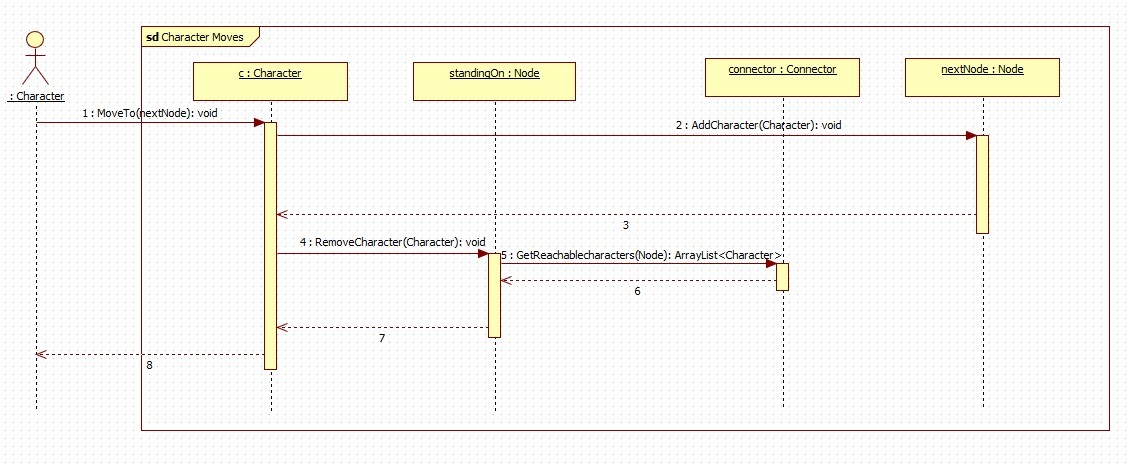
* + 1. **Cistern makes Pipe**



* + 1. Character set Pump



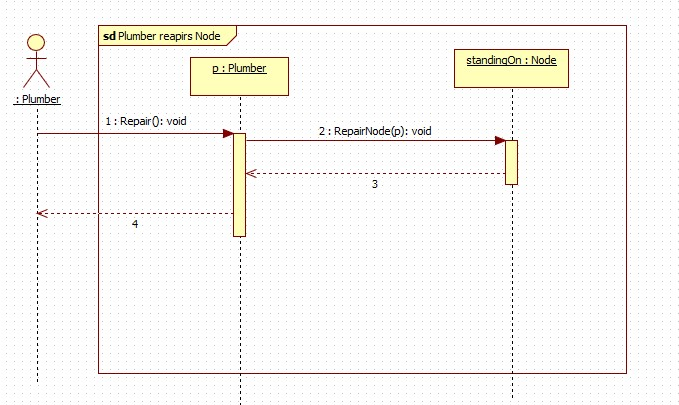
* + 1. **Character Moves**



* + 1. Place Character



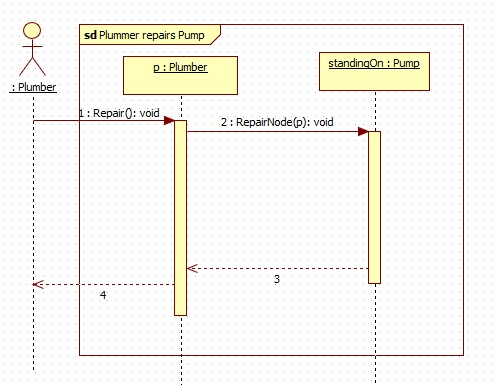
* + 1. **Plumber repairs node**



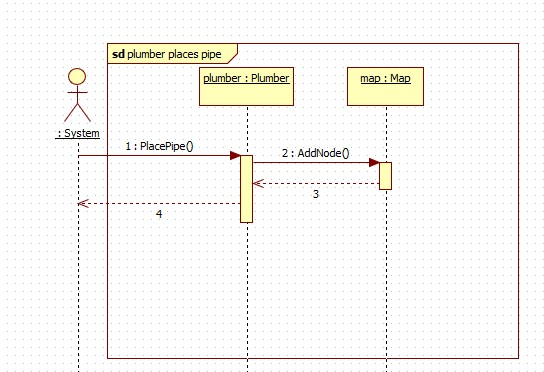
* + 1. Plumber repairs Pipe



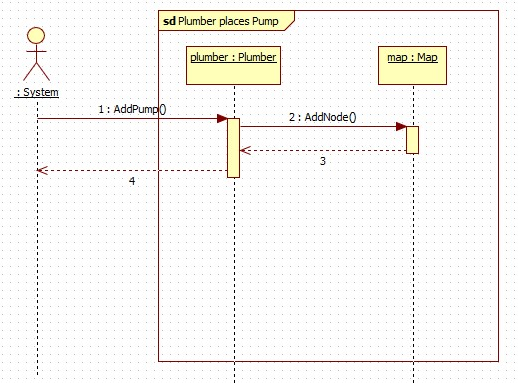
* + 1. **Plumber repairs Pump**



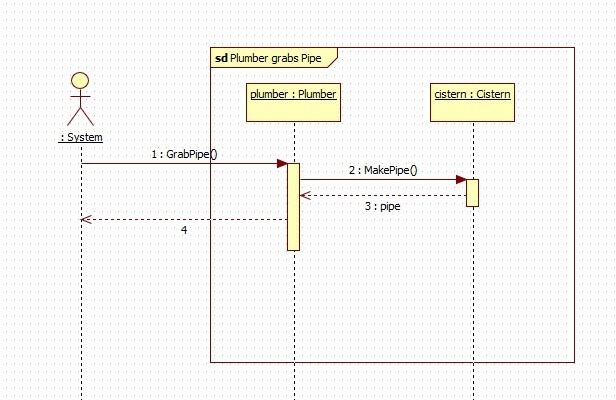
* + 1. Plumber places Pipe



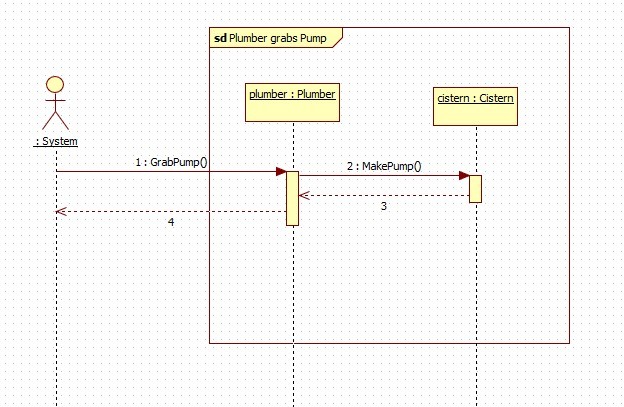
* + 1. **Plumber places Pump**



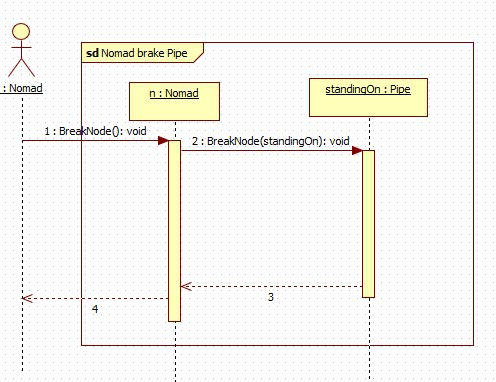
* + 1. Plumber grabs Pipe



* + 1. **Plumber grabs Pump**



* + 1. **Nomad brake Pipe**



1. **Analízis modell kidolgozása**
   1. Objektum katalógus
      1. Csőrendszer (Map)

A csőrendszer a sivatagon át szállítja a vizet a hegyi forrástól a városi ciszternákig. Elágazásmentes csövekből és napelemmel működő pumpákból áll össze a hálózat. A rendszer változtatható és bővíthető csövek mozgatásával, pumpák állításával, vagy ezen elemek hozzáadásával.

* + 1. Ciszterna (Cistern)

A ciszternák gyűjtik össze a forrástól a csőhálózaton keresztül szállított vizet. Itt folyamatosan új csövek készülnek, ezek egyik vége egyből a ciszternákhoz csatlakozik, a másik végük szabad. Új pumpák is találhatók itt, ezeket azonban csak a szerelők tudják beépíteni a hálózatba.

* + 1. Cső (Pipe)

A cső a csőrendszer egyik fajta alkotóeleme. Egyszerű, elágazásmentes. A csöveket szabadon le lehet csatlakoztatni a pumpákról, a hálózat egy másik részére átvinni, és ott rácsatlakoztatni egy pumpára. Ha egy cső egyik vége szabad, akkor a rajta átmenő víz a homokba folyik. Ha egy cső kilyukad (vagy kilyukasztják), akkor a víz kifolyik, és a másik végén lévő pumpához már nem tud eljutni.

* + 1. Forrás (WaterSpring)

A víz a forrásból ered. A forráshoz lehet csatlakoztatni a csőrendszer elemeit. A csőrendszeren foglalatoskodó emberek innen indulnak dolgukra.

* + 1. Pumpa (Pump)

A pumpa a csőrendszer másik típusú alkotóeleme. Egy pumpa több csövet is összeköthet, de egyszerre csak egy bemenete és egy kimenete lehet. Egyenként állítható, hogy a rá csatlakoztatott csövek közül melyeken folyjon tovább a víz. A többi bekötött cső ilyenkor elzárt állapotban van. Véletlen időközönként, véletlenszerűen el tudnak romlani a pumpák.

Ha egy pumpa elromlik, akkor megszűnik a rajta áthaladó vízfolyás.

Minden pumpának van egy belső víztartálya, amely az átfolyatott víz átmeneti tárolójaként funkcionál. Csak akkor tudja egy csőbe továbbítani a vizet, ha annak a csőnek a kapacitása ezt engedi.

* + 1. Szerelő (Plumber)

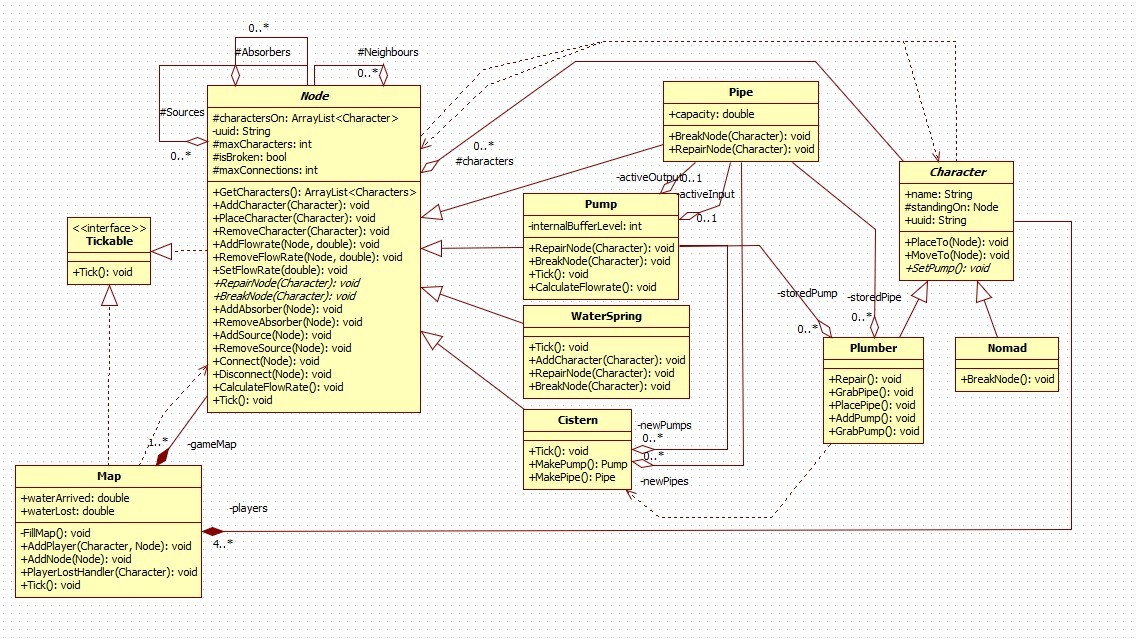
A szerelők feladata a csőhálózat karbantartása. Az elromlott pumpákat megjavítják és a kilyukadt csöveket megfoltozzák. Az ő feladatuk a pumpák beállítása is, amiket úgy állítanak be, hogy mindig a lehető legnagyobb vízmennyiség tudjon átfolyni a rendszeren.

A szabadon lévő csövek segítségével növelni tudják a vízhálózat kapacitását. A ciszternáknál új pumpákhoz juthatnak, amit egy cső közepénél, azt kettévágva tudnak beépíteni a rendszerbe. Csak a csőhálózaton tudnak haladni. A pumpáknál ki tudják kerülni egymást, vagy a nomádokat, de a csöveken már nem. Céljuk, hogy minél több vizet juttassanak el a városok ciszternáiba. Legalább két szerelő tevékenykedik a csőrendszeren.

* + 1. Nomád szabotőr (Nomad)

A nomádok a csöveket kilyukasztják és a pumpákat elállítják. Csak a csőhálózaton tudnak haladni. A pumpáknál ki tudják kerülni egymást, vagy a szerelőket, de a csöveken már nem. Céljuk, hogy minél több víz folyjon el a lyukakon. Legalább két nomád szabotálja a vízhálózatot.

* 1. Statikus struktúra diagramok



***Osztályok leírása***

* + 1. Character
* **Felelősség**

A játékban szereplő karaktereket megvalósító osztályok ősosztálya. Tárolja, hogy a pálya mely elemén (Node) tartózkodik, és az összes karakter közös viselkedését megvalósítja.

* **Ősosztályok**

Nincs ősosztálya.

* **Interfészek**

Nem implementál interfészt.

* **Attribútumok**
  + **name**: karakter neve
  + **uuid**: karakter egyedi azonosítója
  + **standingOn**: eltárolja hogy a karakter a Map mely elemén, melyik Node-on áll.
* **Metódusok**
  + **void placeTo(Node startPoint)**: A karaktert lehelyezi a pálya egy elemére, a játék kezdetekor.
  + **void moveTo(Node moveTo)**: A karakter mozgását valósítja meg, az egyik pályaelemről(Node) a másikra.
  + **void setPump()**: A karakter beállítja a Pumpát, amin áll.
    1. Cistern
* **Felelősség**

A játékban szereplő ciszternákat megvalósító osztály. Ide gyűjtik a vizet a szerelők, és készülnek itt új csövek is.

* **Ősosztályok**

Node  Cistern

* **Interfészek**

Tickable interfészt implementálja.

* **Attribútumok**

Nincsenek saját attribútumai.

* **Metódusok**
  + **void tick()**: időközönként számolja a beérkezett vizet.
  + **Pipe makePipe()**: új csöveket készít
  + **Pump makePump()**: új pumpákat készít
    1. Map
* **Felelősség**

A játékteret megvalósító osztály. Node-okat, és a játékos karaktereket csoportosít.

* **Ősosztályok**

Nincs ősosztály

* **Interfészek**

Tickable interfészt implementálja.

* **Attribútumok**
  + **waterArrived:** Nyilvántartja, hogy mennyi víz folyt be a ciszternába.
  + **waterLost:** Nyilvántartja, hogy mennyi víz folyt el a lyukakon a sivatagi homokra.
* **Metódusok**
  + **void FillMap ()**:Map feltöltése elemekkel és játékososkkal.
  + **void addNode (Node node)**: Új pályaelem (Node) hozzáadása a pályához.
  + **void addPlayer(Character c):** Új játékos hozzáadása.
  + **void Tick():** Időközönként (tick) számolja a ciszternába folyt, és a homokba elfolyt víz mennyiséségt.
  + **void PlayerLostHandler(Character c):** Ha egy player alatt eltűnik a node akkor ez a metódus fogja visszarakni egy forrásra.
    1. Node
* **Felelősség**

A Node osztály egy absztrakt osztály, ezt implementálja minden osztály, amiknek az elemeit a játéktérre tehetjük.

* **Ősosztályok**

Nincs ősosztály.

* **Interfészek**

Tickable interfészt implementálja.

* **Attribútumok**
  + **logger:** egy logger ehhez az osztályhoz.
  + **uuid:** karakter egyedi azonosítója.
  + **charactersOn:** a jelenleg ezen az objektumon álló karakterek listája.
  + **neighbours:** a jelenleg ezzel az objektummal szomszédos objektumok listája.
  + **sources:** azok az objektumok, ahonnan folyik víz ebbe az objektumba.
  + **absorbers:** azok az objektumok, ahová folyik víz ebből az objektumból.
  + **maxCharacters:** az ezen az objektumon maximálisan elfogadott karakterek száma.
  + **maxConnections:** az ehhez az objektumhoz maximálisan csatlakoztatható elemek száma.
  + **isBroken:** hogy éppen működik-e az objektum.
  + **flowRate:** hogy éppen mennyi víz folyik keresztül az objektumon.
* **Metódusok**
  + **String getUuid()**: visszaadja az uuid attribútum értékét.
  + **void addAbsorber(Node absorber):** hozzáad az absorber-ekhez egy újat.
  + **void removeAbsorber(Node absorber):** kivesz egyet az absorber-ek közül.
  + **void addSource(Node source):** hozzáad egy source-okhoz egy újat.
  + **void removeSource(Node source):** kivesz a source-ok közül egyet.
  + **ArrayList<Character> getCharactersOn():** visszaadja a charactersOn listát.
  + **void placeCharacter(Character character):** elhelyez egy karaktert az objektumon.
  + **void addCharacter(Character character):** hozzáad egy karaktert a charactersOn listához.
  + **void removeCharacter(Character character):** kivesz egy karaktert a charactersOn listából.
  + **void repairNode(Character character):** megjavítja az objektumot.
  + **void breakNode(Character character):** elrontja az objektumot.
  + **void addFlowRate(Node source, double excededFlow):** hozzáad a flowRate-hez.
  + **void removeFlowRate(Node source, double flowRate):** elvesz a flowRate-ből.
  + **void setFlowRate(double flowRate):** beállítja a flowRate-et.
  + **void tick():** időközönként számolja a flowRate-et.
  + **void connect(Node node):** hozzácsatlakoztat az objektumhoz egy másikat.
    1. Nomad
* **Felelősség**

A szabotőr nomád-karaktereket megvalósító osztály. A nomád ki tudja lyukasztani a csöveket (Pipe) és el tudja állítani a pumpákat.

* **Ősosztályok**

Character  Nomad .

* **Interfészek**

Nem implementál interfészt.

* **Attribútumok**
  + **name**: karakter neve
  + **uuid**: karakter egyedi azonosítója
  + **standingOn**: eltárolja hogy a karakter a Map mely elemén, melyik Node-on áll.
* **Metódusok**
  + **void placeTo(Node startPoint)**: A karaktert lehelyezi a pálya egy elemére, a játék kezdetekor.
  + **void moveTo(Node moveTo)**: A karakter mozgását valósítja meg, az egyik pályaelemről(Node) a másikra.
  + **void SetPump()**: A karakter beállítja a Pumpát, amin áll.
  + **void BreakNode()**: A Nomad kilyukasztja a Pumpát.
    1. Pipe
* **Felelősség**

Ez az osztály implementálja a játékban a csöveket. A csöveket el lehet rontani és meg lehet javítani, továbbá csatlakoztatni lehet őket más objektumokhoz (összesen maximum kettőhöz), és összesen 1 játékos állhat rajtuk.

* **Ősosztályok**

Node  Pipe

* **Interfészek**

Nem valósít meg interfészeket.

* **Attribútumok**
  + **capacity**: azt mutatja mennyi a kapacitása ennek a csőnek (vagyis azt, mennyi víz fér bele).
* **Metódusok**
  + **double getCapacity()**: visszaadja a capacity értékét, vagyis azt, mennyi víz fér még bele.
  + **void repairNode(Character character)**: megjavítja a csövet.
  + **void breakNode(Character character)**: kilyukasztja, elrontja a csövet.
    1. Plumber
* **Felelősség**

Ez az osztály a játékban lévő vízszerelők osztálya. A vízszerelők meg tudnak javítani csöveket és pumpákat, le tudnak rakni új csöveket és pumpákat, illetve beállíthatják a pumpákat, hogy a ciszternákba tereljék a vizet.

* **Ősosztályok**

Character  Plumber

* **Interfészek**

Nem valósít meg interfészeket.

* **Attribútumok**

Nincsenek saját attribútumai.

* **Metódusok**
  + **void repair()**: megjavítja a Node-ot, amin áll.
  + **void placePump()**: elhelyez egy pumpát oda, ahol áll.
  + **void movePipe()**: átállítja a csövet, amin áll.
    1. Pump
* **Felelősség**

Ez a játékban szereplő pumpák osztálya. Egy pumpa állítható, el tud romlani, és meg lehet javítani. Egy pumpán több játékos is állhat egyszerre.

* **Ősosztályok**

Node  Pump

* **Interfészek**

Tickable interfészt implementálja.

* **Attribútumok**
  + **internalBufferLevel**: ezt arra használjuk, hogy a flowRate-et számoljuk.
  + **activeInput**: ez a pumpa bemenetei közül tárolja azt, amelyik aktív.
  + **activeOutput**: a pumpa kimenetei közül tárorlja, amelyik aktív.
* **Metódusok**
  + **void repairNode(Character character)**: megjavítja a pumpát.
  + **void breakNode(Character character)**: elrontja a pumpát.
  + **void setActiveInput(Pipe p)**: beállítja az activeInput-ot.
  + **void setActiveOutput(Pipe p)**: beállítja az activeOutput-ot.
  + **void calculateFlowRate()**: kiszámolja mennyi víz megy át rajta.
  + **void tick()**: időközönként (random) elrontja a pupmpát, és számolja a flowRate-et.
    1. Tickable
* **Felelősség**

Ez az interfész minden objektumnak, aminek időközönként kell csinálnia valamit.

* **Ősosztályok**

Nincsenek ősosztályok.

* **Metódusok**
  + **void tick()**: az adott időközönként végzett tevékenységre való.
    1. WaterSpring
* **Felelősség**

A WaterSprring egy olyan speciális elem, amivel a ciszternát tudjuk feltölteni. Nem lehet sem elrontani, sem megjavítani, illetve nincs maximum kapacitása.

* **Ősosztályok**

Node  WaterSpring

* **Interfészek**

Tickable interfészt implementálja.

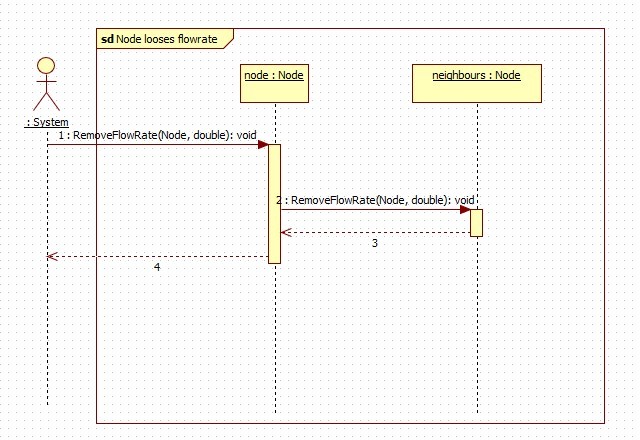
* **Attribútumok**

Nincsnek attribútumai.

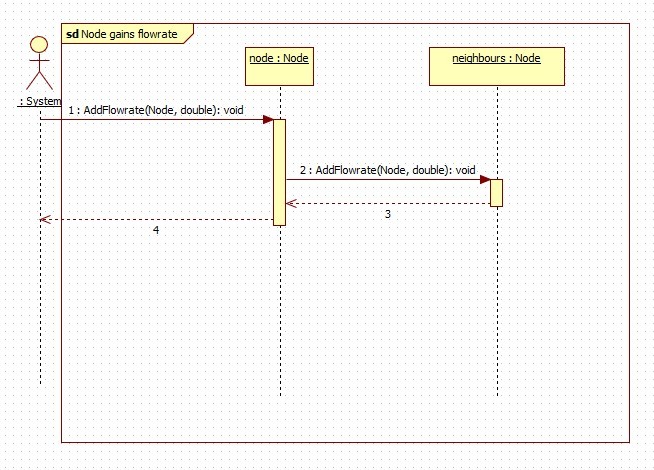
* **Metódusok**
  + **void addCharacter(Character character)**: hozzáad egy karaktert a charactersOn listához.
  + **void repairNode(Character character)**: beállítjuk, hogy ez nem lehetséges.
  + **void breakNode(Character character)**: beállítjuk, hogy ez nem lehetséges.
  + **void tick()**: időközönként számolja a flowRate-et.
  1. Szekvencia diagramok
     1. Add character to Node



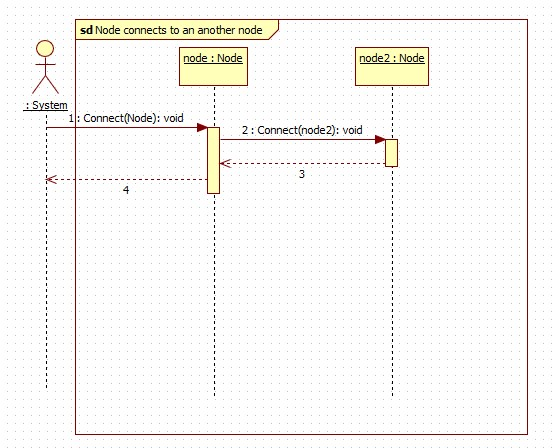
* + 1. **Node loses flowrate:**



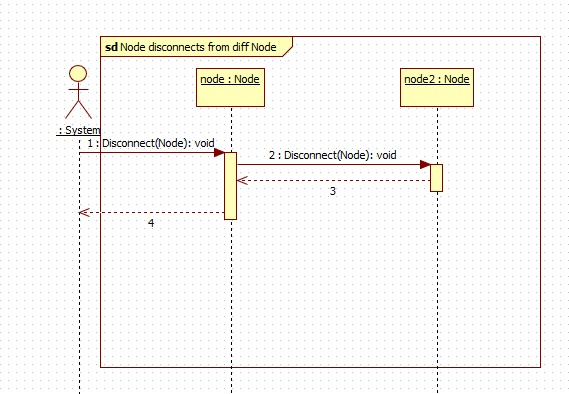
* + 1. Node gains flowrate



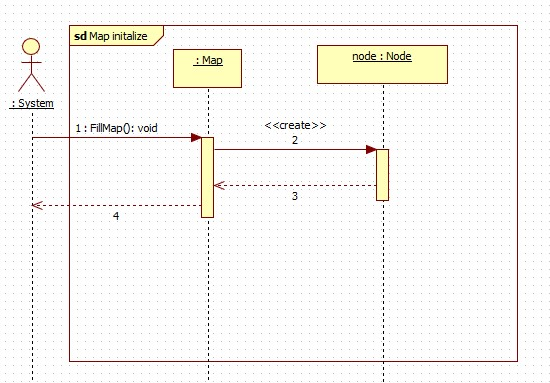
* + 1. **Node connects to diff Node**



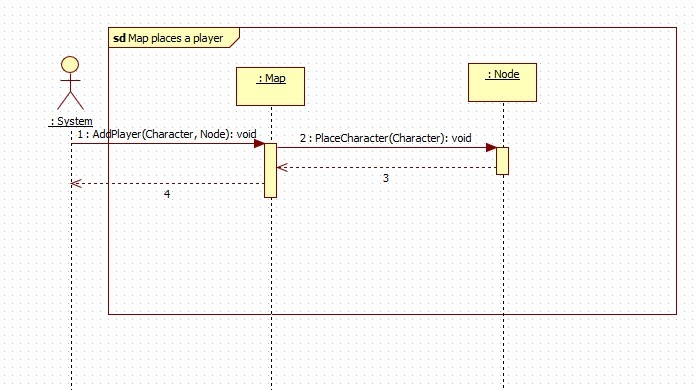
* + 1. Node disconnects from diff Node



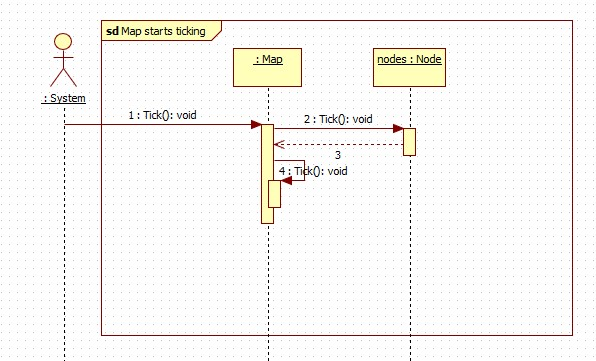
* + 1. **Map initializes**



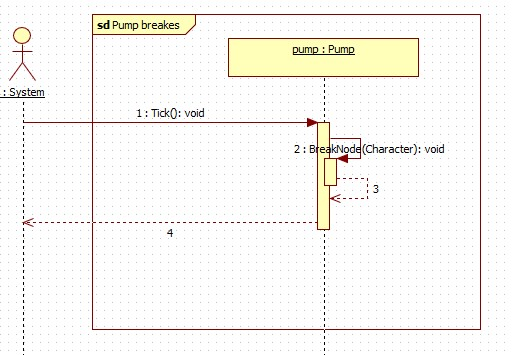
* + 1. Map places a character



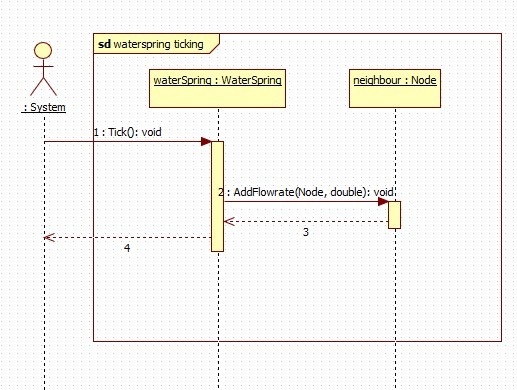
* + 1. **Map starts ticking**



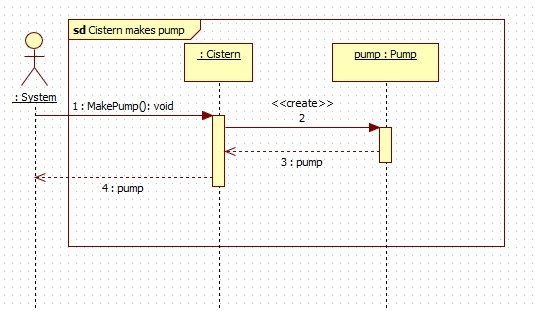
* + 1. Pump breaks



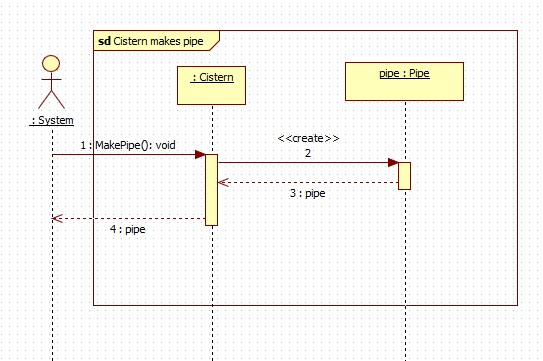
* + 1. **WaterSpring ticking**



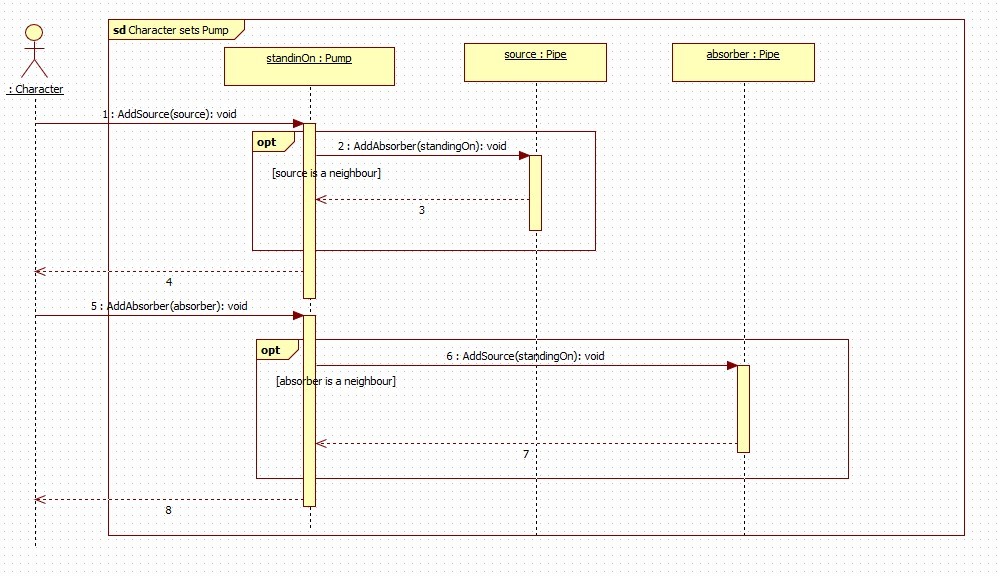
* + 1. Cistern makes Pump



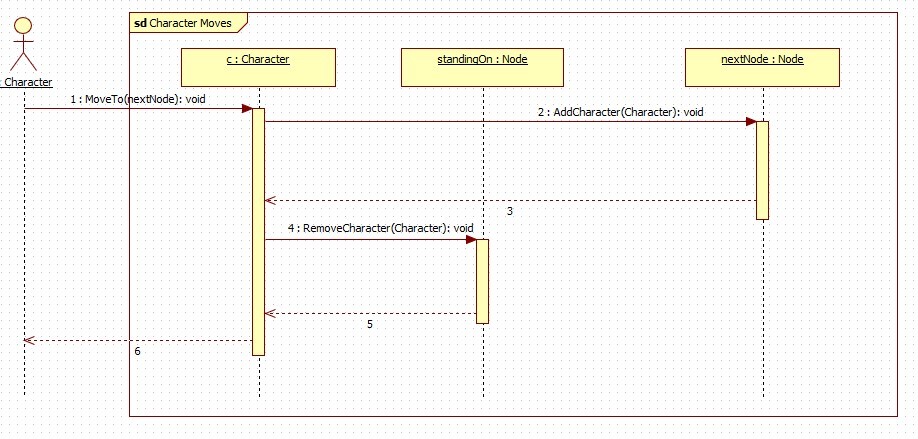
* + 1. **Cistern makes Pipe**



* + 1. Character set Pump



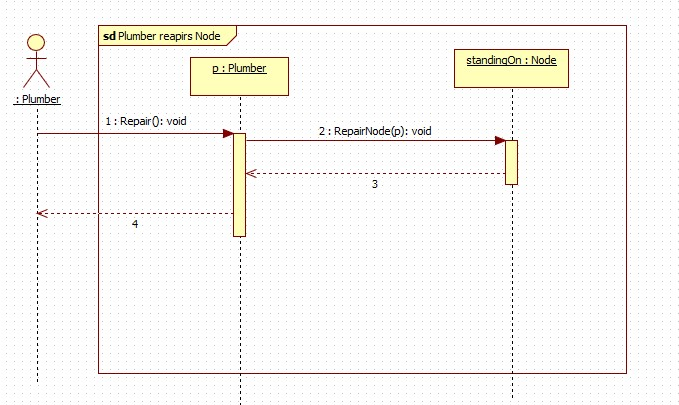
* + 1. **Character Moves**



* + 1. Place Character



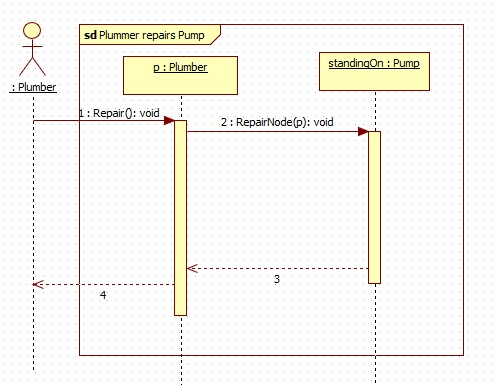
* + 1. **Plumber repairs node**



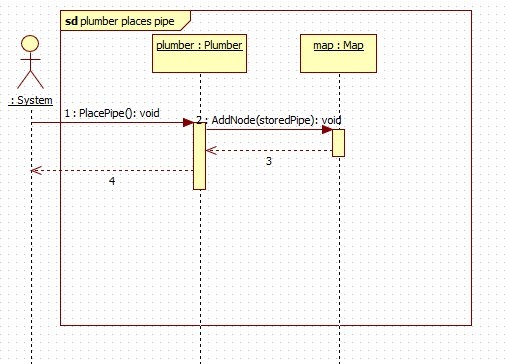
* + 1. Plumber repairs Pipe



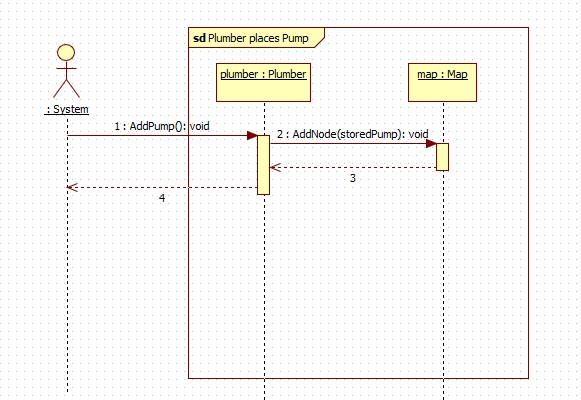
* + 1. **Plumber repairs Pump**



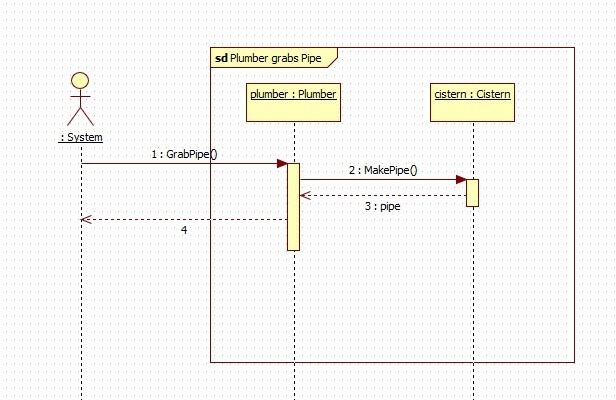
* + 1. Plumber places Pipe



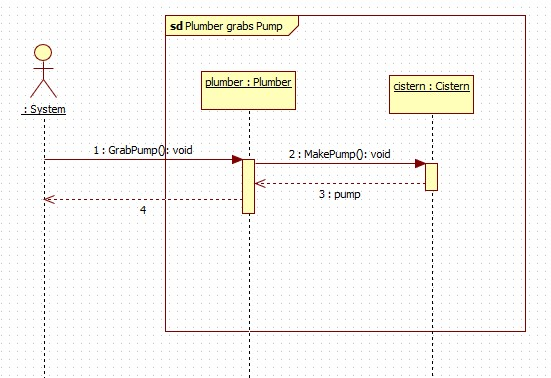
* + 1. **Plumber places Pump**



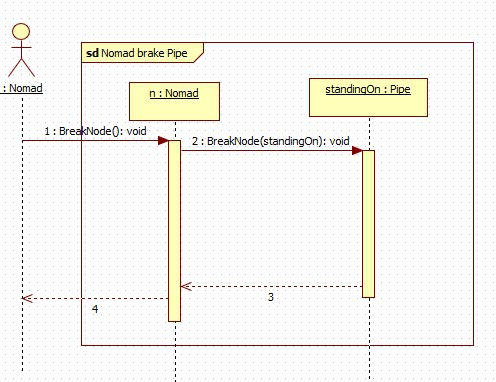
* + 1. Plumber grabs Pipe



* + 1. **Plumber grabs Pump**

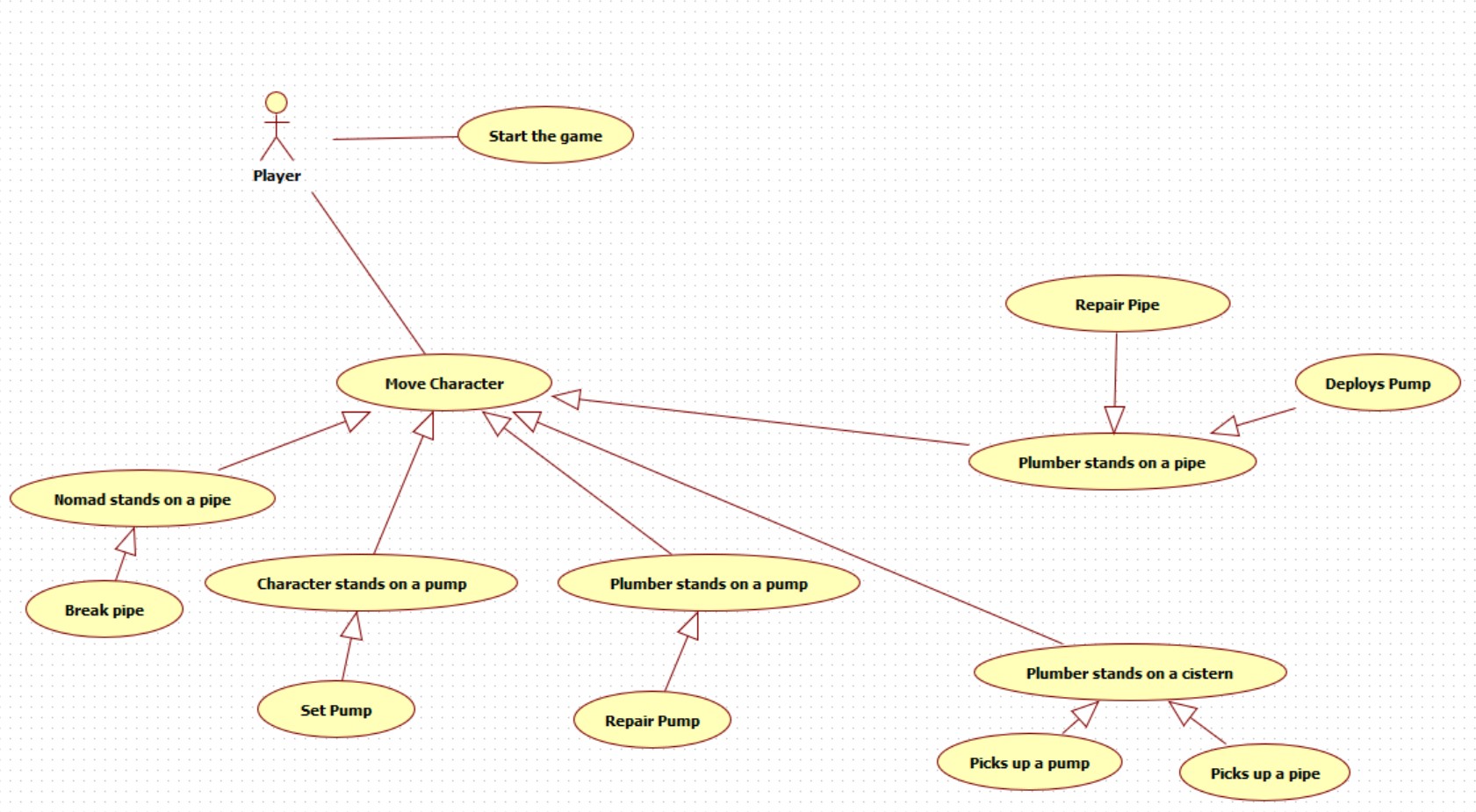


* + 1. **Nomad brake Pipe**



1. **Szkeleton tervezése**

###### A szkeleton modell valóságos use-case-ei

* + 1. Use-case diagram (Test-case)
    2. **Use-case leírások**

|  |  |
| --- | --- |
| **Use-case neve** | **Start the game** |
| **Rövid leírás** | A játékos elindítja a játékot |
| **Aktorok** | Player |
| **Forgatókönyv** | A játék betöltődik |

|  |  |
| --- | --- |
| **Use-case neve** | **Move Character** |
| **Rövid leírás** | A játékos mozgatja a karakterét |
| **Aktorok** | Player |
| **Forgatókönyv** | A játékos egy Plumbert (szerelőt) egy csőre helyez |
| **Forgatókönyv** | A játékos egy Plumbert (szerelőt) egy pumpára helyez |
| **Forgatókönyv** | A játékos egy Nomádot (szabotőrt) egy csőre helyez |
| **Forgatókönyv** | A játékos egy Nomádot (szabotőrt) egy pumpára helyez |

|  |  |
| --- | --- |
| **Use-case neve** | **Nomad stands on a pipe** |
| **Rövid leírás** | A játékos által irányított nomád egy cső típusú elemre lépett |
| **Aktorok** | Player |
| **Forgatókönyv** | A nomád kilyukasztja a csövet (break pipe) |

|  |  |
| --- | --- |
| **Use-case neve** | **Character stands on a pump** |
| **Rövid leírás** | A játékos által irányított karakter egy pumpa típusú elemre |

|  |  |
| --- | --- |
|  | lépett |
| **Aktorok** | Player |
| **Forgatókönyv** | A karakter állít a pumpán |

|  |  |
| --- | --- |
| **Use-case neve** | **Plumber stands on a pump** |
| **Rövid leírás** | A játékos által irányított szerelő egy pumpa típusú elemre lépett |
| **Aktorok** | Player |
| **Forgatókönyv** | Ha a pumpa, amire a szerelő lépett rossz, akkor megjavítja |

|  |  |
| --- | --- |
| **Use-case neve** | **Plumber stands on a pipe** |
| **Rövid leírás** | A játékos által irányított szerelő egy cső típusú elemre lépett |
| **Aktorok** | Player |
| **Forgatókönyv** | Ha a cső, amire a szerelő lépett lyukas, akkor megjavítja |
| **Forgatókönyv** | Ha van nála új pumpa, akkor azt telepítheti az adott csőre |

|  |  |
| --- | --- |
| **Use-case neve** | **Plumber stands on cistern** |
| **Rövid leírás** | A játékos által irányított szerelő egy ciszterna elemre lépett |
| **Aktorok** | Player |
| **Forgatókönyv** | A szerelő új pumpát vesz magához |
| **Forgatókönyv** | A szerelő egy új cső végét veszi magához |

|  |  |
| --- | --- |
| **Use-case neve** | **Break pipe** |
| **Rövid leírás** | A cső elrontása, kilyukasztása |
| **Aktorok** | Player |
| **Forgatókönyv** | Egy nomád tönkreteszi a csövet, amin éppen áll |

|  |  |
| --- | --- |
| **Use-case neve** | **Set pump** |
| **Rövid leírás** | A karakter állít a pumpán, amin éppen áll |
| **Aktorok** | Player |
| **Forgatókönyv** | Egy szerelő beállítja az adott pumpát |
| **Forgatókönyv** | Egy nomád elállítja az adott pumpát |

|  |  |
| --- | --- |
| **Use-case neve** | **Repair pump** |
| **Rövid leírás** | A tönkrement pumpa megjavítása |
| **Aktorok** | Player |
| **Forgatókönyv** | Egy szerelő megjavítja az elromlott pumpát, amin éppen áll |

|  |  |
| --- | --- |
| **Use-case neve** | **Picks up pump** |
| **Rövid leírás** | Új pumpa felvétele |
| **Aktorok** | Player |
| **Forgatókönyv** | Egy szerelő új pumpát vesz magához, ha egy ciszternán tartózkodik |

|  |  |
| --- | --- |
| **Use-case neve** | **Picks up the end of pipe** |
| **Rövid leírás** | Egy cső egyik végének felvétele, hogy áthelyezhető legyen |
| **Aktorok** | Player |

|  |  |
| --- | --- |
| **Forgatókönyv** | Egy szerelő felveszi egy cső egyik végét |

|  |  |
| --- | --- |
| **Use-case neve** | **Deploys pump** |
| **Rövid leírás** | Az újonnan felvett pumpa telepítése a rendszerbe |
| **Aktorok** | Player |
| **Forgatókönyv** | Egy szerelő egy cső közepére telepíti az új pumpát |

|  |  |
| --- | --- |
| **Use-case neve** | **Repair pipe** |
| **Rövid leírás** | Egy kilyukasztott cső megfoltozása, megjavítása |
| **Aktorok** | Player |
| **Forgatókönyv** | Egy szerelő megjavítja a lyukas csövet, amin éppen áll |

###### A szkeleton kezelői felületének terve, dialógusok

A szkeleton vezérlésére a konzolon keresztül lesz lehetőség. Minden akció előtt vagy után megjelenik majd egy kérdés, hogy mi legyen a következő interakció. Egyelőre menü alapú az interakció ezért lesz egy segítség menü is ami ezt elmagyarázza.

Folyamat:

A szkeleton elindul és kér egy inputot

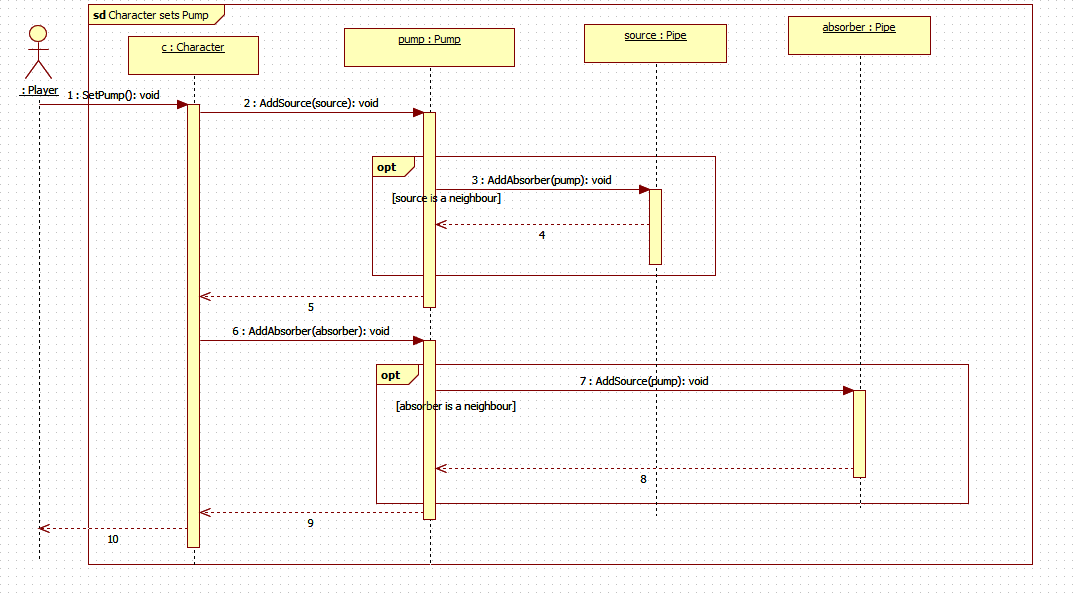
* Pálya módosítása
  + Pálya elem lerakása
    - Cső
    - Pumpa
    - Ciszterna
    - Forrás
  + Elemek kitörlése
  + Elemek áthelyezése
* Karakter lehelyezése
  + Nomád
  + Szerelő
* Karakter interakció végrehajtása
  + Mozgás
  + Javítás
  + Elrontás
  + Állítás
* A pálya tickelése (az egész pálya informálása egy kör elmúlásáról)

A válaszok minden esetben egy logger által írt INFO típusú válaszok lesznek melyek a következő formájúak:

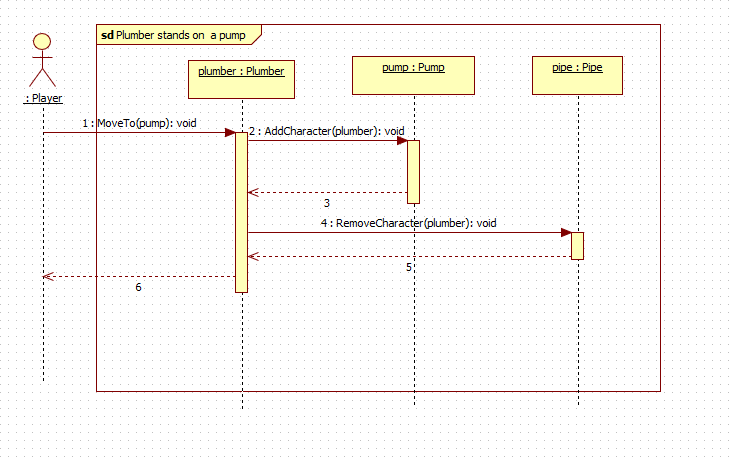
HH-pp-mpmp.ms [INFO]: (interakcióba lépő osztály uuid-ja) az akció részletei. Ezután újra felteszi a program a kérdést, hogy mi legyen a következő akció.

###### Szekvencia diagramok a belső működésre

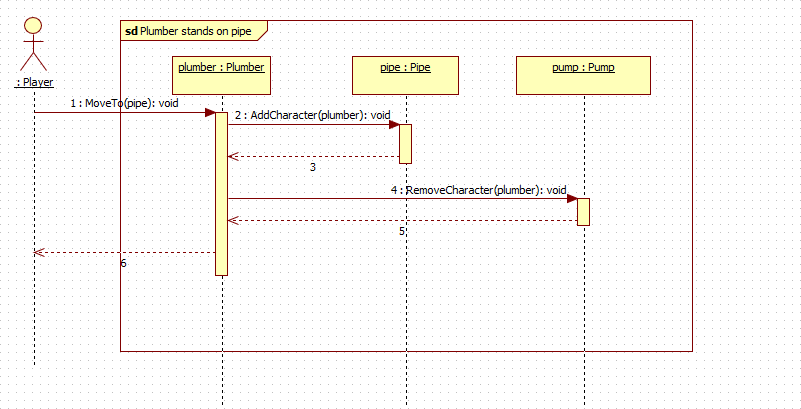
A felhasznált use-case-ek az adott diagram alatt vannak feltüntetve.



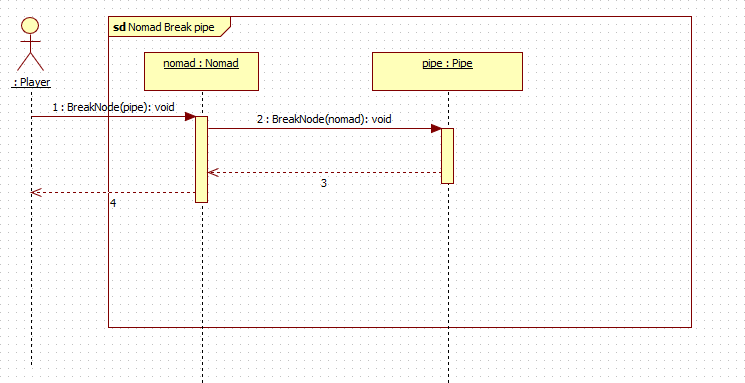
* Move Character
* Character stands on a pump
* Set pump



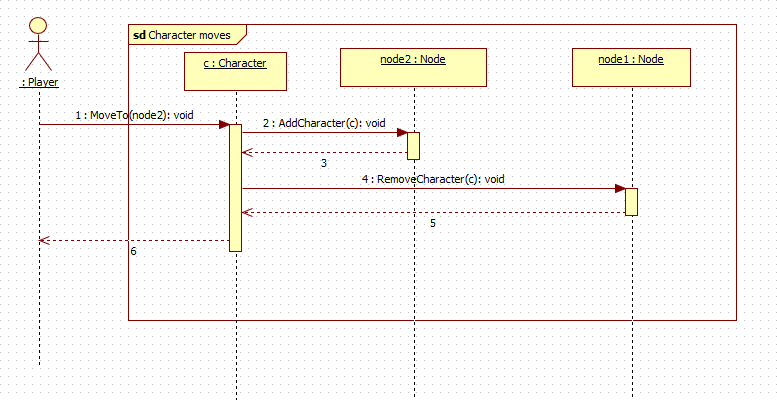
* Move Character
* Plumber stands on a pump

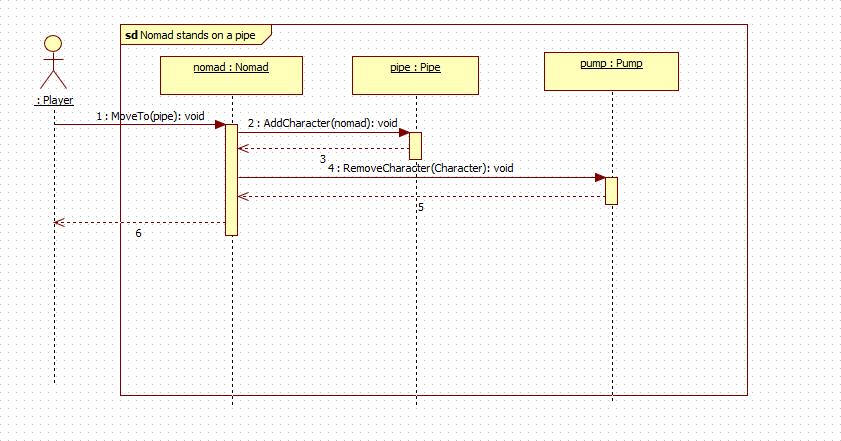


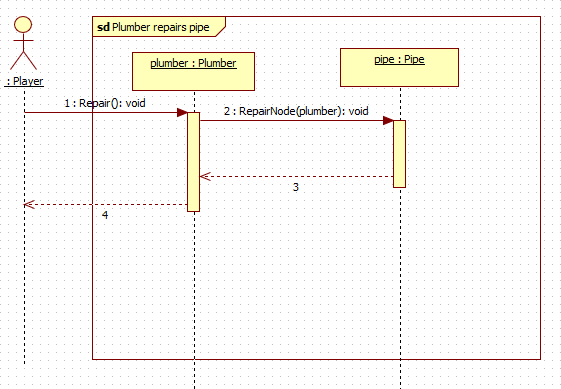
* Move Character
* Plumber stands on a pipe



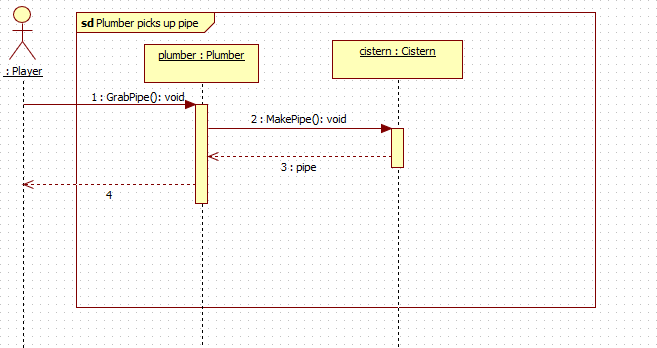
* Move Character
* Nomad stands on a pipe
* Break pipe



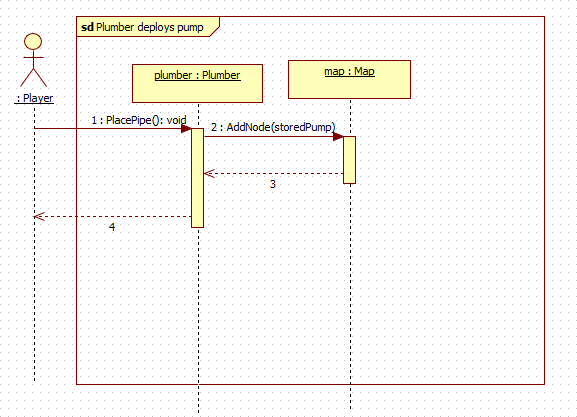
* Move Character
* Move Character
* Nomad stands on a pipe

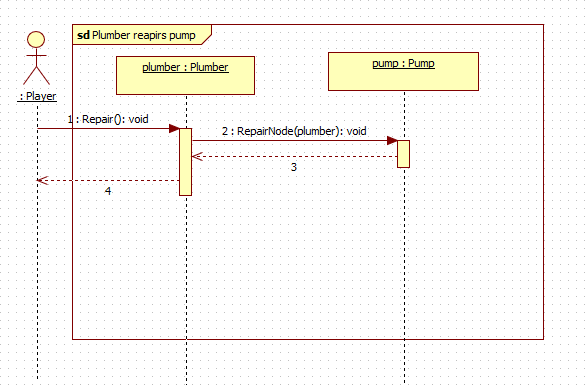
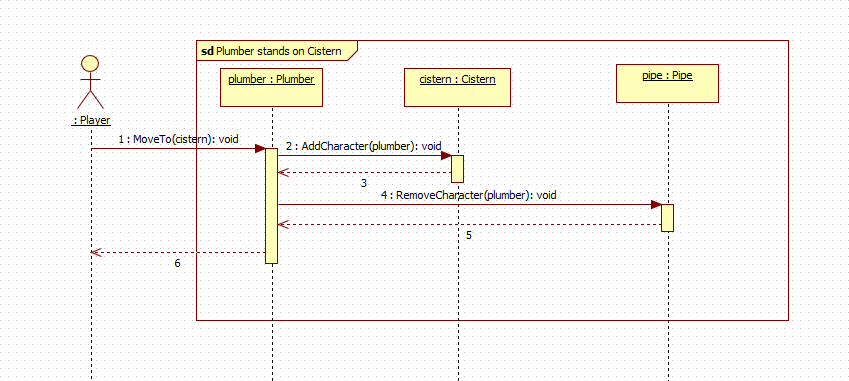


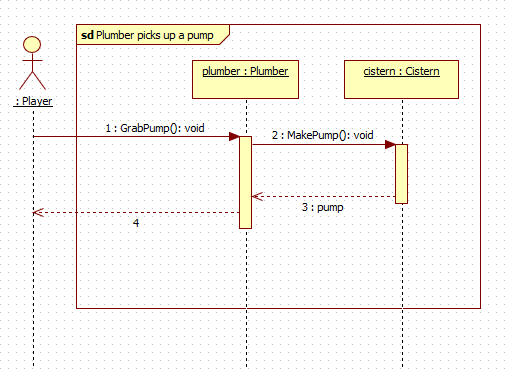
* Move Character
* Plumber stands on a pipe
* Repair Pipe



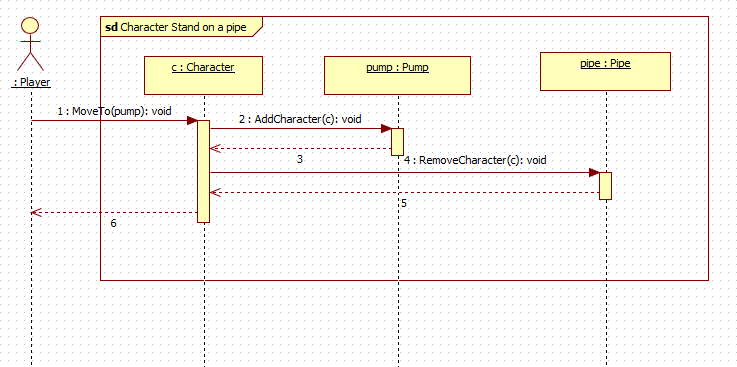
* Move Character
* Plumber stands on a cistern
* Picks up a pipe



* Move Character
* Plumber stands on a pipe
* Deploys Pump
* Move Character
* Plumber stands on a pump
* Repair Pump
* Move Character
* Plumber stands on a cistern

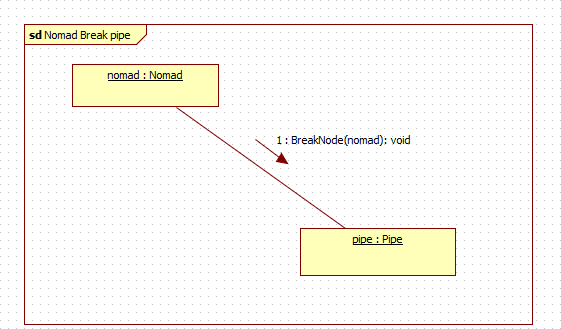
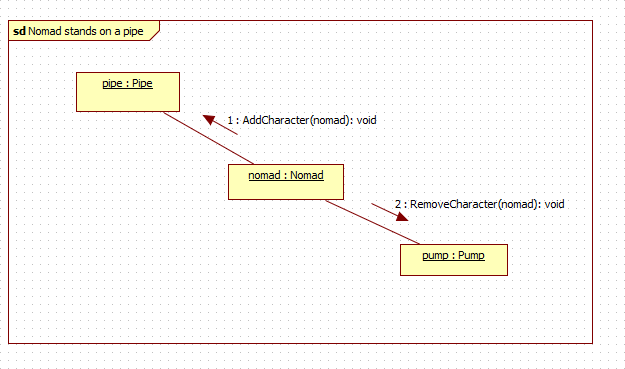


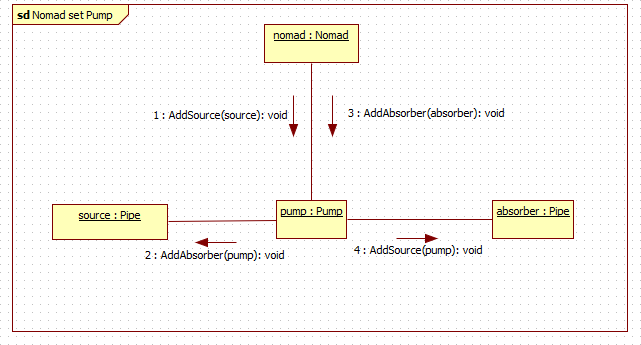
* Move Character
* Plumber stands on a cistern
* Picks up a pump



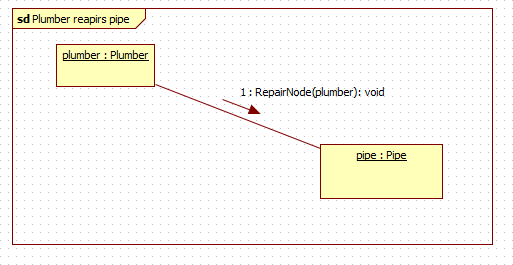
* Move Character
* Nomad stands on a pipe
* Plumber stands on a pipe

###### Kommunikációs diagramok

* + 1. Nomad breaks pipe
    2. **Nomad stands on a pipe**
    3. Nomad stands on a pipe



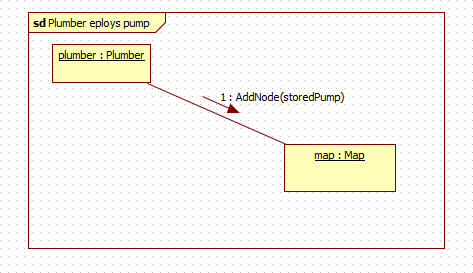
* + 1. **Plumber repairs pipe**



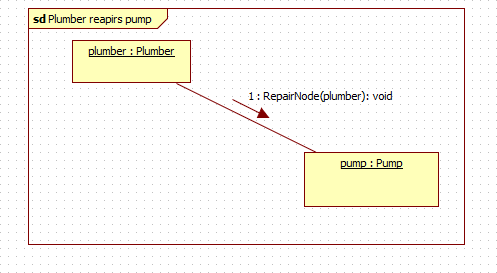
* + 1. Plumber picks up pipe



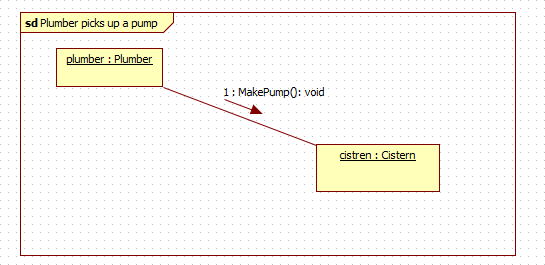
* + 1. **Plumber deploys pump**

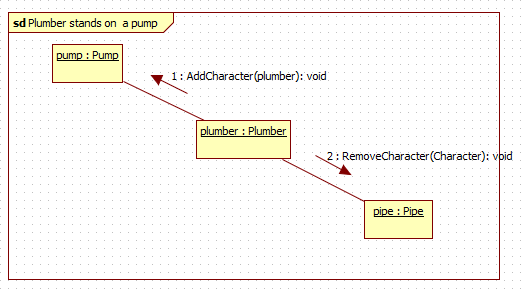
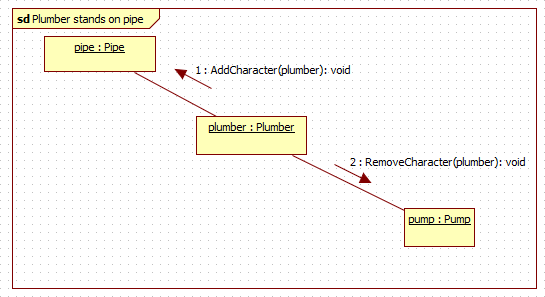
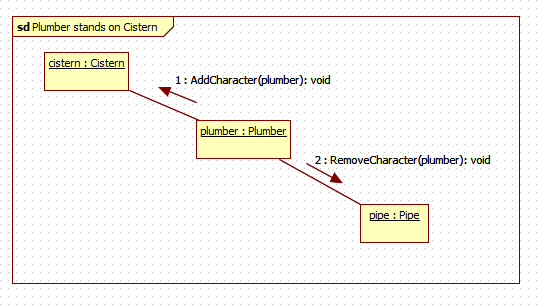
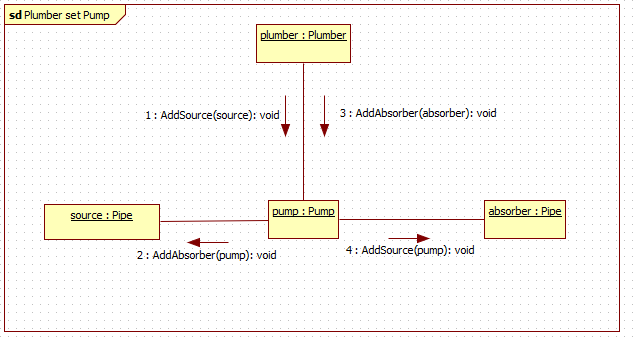


* + 1. Plumber repairs pump



* + 1. **Plumber picks up a pump**



* + 1. Plumber stands on a pump
    2. **Plumber stands on a pipe**
    3. Plumber stands on a cistern
    4. **Plumber sets pump**

1. **Részletes tervek**

###### Osztályok és metódusok tervei.

* + 1. Character
* Felelősség

A játékban szereplő karaktereket megvalósító osztályok ősosztálya. Tárolja, hogy a pálya mely elemén (Node) tartózkodik, és az összes karakter közös viselkedését megvalósítja.

* Ősosztályok

Nincs ősosztálya.

* Interfészek

Tickable interfészt implementálja.

* Attribútumok
  + **name**: karakter neve
  + **uuid**: karakter egyedi azonosítója
  + **standingOn**: eltárolja, hogy a karakter a Map mely elemén, melyik Node-on áll.
  + **stuckedInPipe**: random generált idő, ameddig a karakter nem léphet el a csőről ha az ragacsos.
* Metódusok
  + **void placeTo(Node startPoint)**: A karaktert lehelyezi a pálya egy elemére, a játék kezdetekor.
  + **void moveTo(Node moveTo)**: A karakter mozgását valósítja meg. Áthelyezi a paraméterként megadott Nodera, és leveszi az előző Noderól.
    - if(stuckedInPipe == 0)
      * than moveTo addCharacter(character);
      * standingOn removeCharacter(character);
      * character standingOn = moveTo;
      * endif
  + ***void setPump(Pipe in, Pipe out)****:* A karakter beállítja a Pumpán, amin áll. A Pumpához csatlakoztatott csövek közül melyiket nyitja meg, hogy a pumpába ki és be- folyjon a víz. Absztrakt metódus, leszármazottak implementálják.
  + **string getUuid():** Visszaadja a karakter uuidjét.
  + **string getName():** Visszaadja a karakter nevét.
  + **Node getStaindingOn():** Visszaadja azt a Nodeot, amin a karakter áll.
  + **void breakNode():** Megpróbálja megrongálni azt a Nodeot amin éppen áll. Ha az a Node nem pipe akkor kivételt dob.
    - standingOn breakNode;
  + **void makePipeSticky():** Megpróbálja ragacsossá tenni azt a Nodeot amin áll. Ha az a Node nem pipe akkor kivételt dob.
    - stadningON setSrface(„sticky”, character);
  + **void tick():** Tickable interfészt implementált metódusa. Körönként csökkenti a stuckedInPipe időt, miután ragacsos csőre lépett.
    - if (stuckedInPipe > 0) than stuckedInPipe -= 1;
  + **void stuck():** Ha ragacsos csőre lépünk generálunk egy véletlen számot, és beállítjuk ezt a stuckedInPipe időnek.
    - stuckedInPipe = random int;
    1. Cistern
* Felelősség

A játékban szereplő ciszternákat megvalósító osztály. Ide gyűjtik a vizet a szerelők, és készülnek itt új csövek és pumpák is.

* Ősosztályok

Node  Cistern

* Interfészek

Tickable interfészt implementálja.

* **Attribútumok** *A felsoroltokon kívül rendelkezik még a Node absztrakt ősosztály attribútumaival.*
  + **prodicedPumps:** A ciszterna maga által készített pumpákat tárolja.
* Metódusok
  + **void tick()**: időközönként számolja a ciszternába beérkezett vizet, a hozzá kapcsolt csövekből.
    - if (size of sources < maxConnections)
      * than newPipe = Cistern makePipe();
      * add newPipe to sources;
    - endif
    - while(iterate throught sources)
      * Map waterArrives += Node flowRate;
    - end
  + **Pipe makePipe()**: Új csöveket készít, amit lehelyez a Map-re és magához csatolja.
    - create Pipe temp;
    - temp connect(Cistern);
    - Map addNode(temp, x, y);
    - return temp;
  + **Pump makePump()**: Új pumpákat készít, amit elraktároz a producedPump-ba.
    - create Pump temp;
    - add temp to producedPumps;
  + **Pump givePump():** A rajta álló szerelőnek ad egy pumpát ha az akar felvenni.
    - if (producedPump is not empty)
      * than remove and return Pump from producedPump;
    - else return null;
  + **void repairNode(Character character):** Próbálja megjavítja az objektumot, de csak kivételt dob.
  + **void breakNode(Character character):** Próbálja elrontja az objektumot, de csak kivételt dob.
  + **void setSurface(string surface, Character c):** Próbálja csúsuóssá/ragacsossá az objektumot, de csak kivételt dob.
    1. Map
* Felelősség

A játékteret megvalósító osztály, pálya. Játéktért elemeit (Node-okat), és a játékos karaktereket csoportosítja.

* Ősosztályok

Nincs ősosztály

* Interfészek

Tickable interfészt implementálja.

* Attribútumok
  + **waterArrived:** Nyilvántartja, hogy mennyi víz folyt be a ciszternába.
  + **waterLost:** Nyilvántartja, hogy mennyi víz folyt el a lyukakon a sivatagi homokra.
  + **players:** A pályán levő játékos Character-ek.
  + **gameMap:** A pálya elemeit tartalmazó lista.
* Metódusok
  + **void FillMap ()**:Map feltöltése elemekkel és játékososkkal.
  + **Node getNode(int x, int y):** Visszaadja a megadott koordinátán lévő Nodeot, ha van ilyen.
  + **int getNodeCount():** Megszámolja és visszaadja a pályán lévő Nodeok számát.
    - count = 0;
    - for each column in gameMap
      * for each node in column
        + if node is not null then

count = count+1;

* + - * + end if
      * end for
    - end for
    - return count;
  + **void addNode (Node node, int x, int y)**: Új pályaelem (Node) hozzáadása a pályához a megadott koordinátákra
  + **void removeNode(Node n)**: Kiveszi a megadott Nodeot a pályából.
  + **void addPlayer(Character c, Node n):** A paraméterként megadott játékos hozzáadása a játékhoz, és lehelyezése a megadott Nodera.
  + **void Tick():** Időközönként (tick) számolja a ciszternába folyt, és a homokba elfolyt víz mennyiséségt.
    - for each nodes array in gameMap
      * for each node in nodes
        + node.tick();
      * end for
    - end for
  + **void PlayerLostHandler(Character c):** Ha egy player alatt eltűnik a node akkor ez a metódus fogja visszarakni a megfelelő Nodera.
    - if player position is null
      * place player to start position
    - end if
  + **int playerCount():** Visszaadja a játékosok számát.
  + **void printPlayers():** Kiírja a pályán lévő játékos nevét és hogy melyik Nodeon áll.
    - for i from 0 to players size()-1
      * player = players.[i];
      * print player’s name and the node’s uuid;
    - end for
  + **void clearMap():** Kiüríti a pályát.
  + **Character getPlayer(int idx):** Visszaadja az adott indexű játékost.
  + **void loadMap(string fileName):** Betölt a megadott nevű fájlból egy pályát.
    - if file does not end with ”.json” then
      * print ”the file must be a .json configuration file!”
    - else
      * load the map
    - end
  + **void saveMap(string fileName):** Kimenti a megadott nevű fájlba a pályát.
    - if file does not end with ”.json” then
      * print ”the file must be a .json configuration file!”
    - else
      * print the map
    - end
  + **void printNodes():** Kiírja a pálya elemeinek, azaz a Nodeokank a koordinátáit és a uuidket.
    - for each nodes array in gameMap
      * for each node in nodes
        + if node is not null then

print x and y coordinates and uuid of the node;

* + - * + end if
      * end for
    - end for
    1. Node
* Felelősség

A Node osztály egy absztrakt osztály, ezt implementálja minden osztály, amik a elemei lesznek, és játéktérre tehetjük.

* Ősosztályok

Nincs ősosztály.

* Interfészek

Tickable interfészt implementálja.

* Attribútumok
  + **logger:** egy logger ehhez az osztályhoz.
  + **uuid:** a node egyedi azonosítója.
  + **characters:** a jelenleg ezen az nodeon álló karakterek listája.
  + **neighbours:** a jelenleg ezzel a node-dal szomszédos és a node-ba csatlakoztatott más nodeok listája.
  + **sources:** azon nodeok, ahonnan folyik víz ebbe a nodeba.
  + **absorbers:** azon nodeok, ahová folyik víz ebből a nodeba.
  + **maxCharacters:** az ezen a nodeon a maximálisan elfogadott karakterek száma.
  + **maxConnections:** az ehhez a nodehoz maximálisan csatlakoztatható elemek száma.
  + **isBroken:** A node állapota, műkódik e vagy sem.
  + **flowRate:** hogy éppen mennyi víz folyik keresztül a nodeon.
  + **x:** Node x koordináltja a pályán.
  + **y:** Node y koordináltja a pályán.
* Metódusok
  + **String getUuid()**: visszaadja az uuid attribútum értékét.
  + **ArrayList<Character> getCharactersOn():** Visszaadja a nodeon álló karakterke listáját.
  + **ArrayList<Node> getNeighbours():** Visszaadja a node-hoz kötött nodeok listáját.
  + **void placeCharacter(Character character):** elhelyez egy karaktert a nodeon. Játék inicializáláakor használatos, vagy ha a játékos alól eltűnt elem miatt vissza kell helyezni a kezdő nodera.
  + **void addCharacter(Character character):** Hozzáad egy karaktert a characters listához, ha még fér karakter a nodera.
    - if (size of characters >= macCharacters)
      * than throw exception;
    - else while(iterate throught neighbours)
      * if (a neighbour contains character) than add character to characters
    - end
  + **void removeCharacter(Character character):** kivesz egy karaktert a characters listából.
    - if(characters contains character)
      * than remove character from characters
  + ***void repairNode(Character character):*** Megjavítja az objektumot. Absztrakt, leszármazottak implementálják.
  + ***void breakNode(Character character):*** Elrontja az objektumot. Absztrakt, leszármazottak implementálják.
  + **void addFlowRate(Node source, double excededFlow):** Növeli a flowRate-et a megadott értékkel.
    - if(sources contains source)
      * than flowRate += excededFlow;
      * while(iterate throught absorbers)
        + addFlowRate();
        + end
    - endif
  + **void removeFlowRate(Node source, double flowRate):** Csökkenti a flowRate-et a megadott étékkel.
    - if(sources contains source)
      * than flowRate -= flowRate;
      * while(iterate throught absorbers)
        + removeFlowRate();
        + end
    - endif
  + **void setFlowRate(double flowRate):** Beállítja a flowRate-et a megadott értékre.
  + **void tick():** Időközönként, azaz körönként számolja a flowRate-et.
    - calculateFlowRate();
  + **void calculateFlowRate():** Számolja a Nodeon átfolyó vít mennyiségét.
    - if(Node is broken)
      * than Map waterLost += Node flowRate;
      * while(iterate througth absorber nodes)
        + absotber node removeFlowRate(Node, Node flowRate);
        + end
    - else
      * while(iterate througth absorber nodes)
        + absotber node addFlowRate(Node, Node flowRate);
        + end
  + **void connect(Node node2):** Hozzácsatlakoztatja a megadott Nodeot.
    - node1 neighbours add node2;
    - if(node2 neigbours not contains node1)
      * than node2 connect(node1);
  + **void disconnect(Node node):** Lecsatlakoztatja a megadott Nodeot.
    - node1 neighbours remove node2;
    - if(node2 neigbours contains node1)
      * than node2 disconnect(node1);
  + ***void setSurface(string surface, Character c):*** Absztrakt metódus amit a leszármazottak implementálnak. Célja, hogy a karakterek tudják ragacsossá/csúszóssá teni a pályaelemek egy részét (pipe).
    1. Nomad
* Felelősség

A szabotőr nomád-karaktereket megvalósító osztály. A nomád ki tudja lyukasztani a csöveket (Pipe), el tudja állítani a pumpákat és ragacsossá illetve csúszóssá tudja tenni a csövetket.

* Ősosztályok

Character  Nomad .

* Interfészek

Tickable interfészt implementálja.

* Attribútumok
  + **name**: karakter neve
  + **uuid**: karakter egyedi azonosítója
  + **standingOn**: eltárolja, hogy a karakter a Map mely elemén, melyik Node-on áll.
  + **stuckedInPipe**: random generált idő, ameddig a karakter nem léphet el a csőről ha az ragacsos.
* Metódusok
  + **void setPump(Pipe in, Pipe out)**: A nomád beállítja a Pumpán, amin áll. A Pumpához csatlakoztatott csövek közül melyiket nyitja meg, hogy a pumpába ki és be- folyjon a víz.
    - if(in is neighbour of standingOn)
      * than setActiveInput(in);
    - if(out is neighbour of standingOn)
      * setActiveOutput(out);
  + **void makePipeSlippery()**:A nomád csúszóssá teszi a cső felszínét.
    - standingOn setSurface(„slippery”, Nomad);
    1. Plumber
* Felelősség

Ez az osztály a játékban lévő vízszerelők osztálya. A vízszerelők meg tudnak javítani csöveket és pumpákat, le tudnak rakni új csöveket és pumpákat, illetve beállíthatják a pumpákat, hogy a ciszternákba tereljék a vizet. A már lerakott csövek átmozgatásával is tudják a vizet a ciszternába terelni, továbbá ők is tudnak csövet lyukasztani

* Ősosztályok

Character  Plumber

* Interfészek

Tickable interfészt implementálja.

* Attribútumok
  + **name**: karakter neve
  + **uuid**: karakter egyedi azonosítója
  + **standingOn**: eltárolja, hogy a karakter a Map mely elemén, melyik Node-on áll.
  + **stuckedInPipe**: random generált idő, ameddig a karakter nem léphet el a csőről ha az ragacsos.
  + **pickedUpPump:** A plumber inventoryában a ciszternán felvett pumpát tárolja.
  + **pickedUpPipe:** A plumber inventoryában a mindkét végén lecsatlakoztatott majd felvett csöveket tárolja.
  + **draggedPipe:** Az egyik végén lecsatlakoztatott, és átmozgatott cső.
* Metódusok
  + **void repair()**: megjavítja a Node-ot, amin áll.
    - standingOn repairNode(Plumber);
  + **void placePump()**: elhelyez egy pumpát oda, ahol áll. Ha csövön áll sikeresen lehelyezi, ha més nodeon akkor kivételt dob.
    - if(has a pickedPump)
      * than Map addNode(pickedPump);
      * create Pipe newPipe;
      * Map addNode(newPipe);
      * newPipe connect(pickedUp);
      * standingOn connect(pickedUp); endif
  + **void placePipe()**: Lehelyezi egy csövet a pályára, és hozzáköti ahhoz a nodehoz amin áll.
    - if(has draggedPipe)
      * than standingOn connect(draggedPipe);
      * draggedPipe = null;
      * return;
    - else if(has pickedUpPipe)
      * than standingOn connect(pickedUpPipe);
      * Map addNode(pickedUpPipe)
      * pickedUpPipe = null;
      * return;
  + **void pickupPump():** Megpróbál felvenni egy pumpát. Ha nem ciszternán áll akkor kivételt dob.
    - Pump P = standingOn givePump();
    - if(P is not null)
      * than pickedUpPump = P;
    - endif
  + **void pickupPipe():** Lecsatlakoztatja a cső egyik végét és elhúzza egy másik elemhez. Ha a cső mindkét vége disconnected akkor felveszi az egészet az inventoryába.
    - while(iterate throught neighbours of standingOn)
      * if(standingOn has at least2 neighbours)
        + than draggedPipe = iterated Node;
        + break;
        + endif
    - end
    - if(draggedPipe neighbours is empty)
      * than pickedUpPipe = draggedPipe;
      * draggedPipe = null;
    - endif
  + **void setPump(Pipe in, Pipe out)**: A plumber beállítja a Pumpán, amin áll. A Pumpához csatlakoztatott csövek közül melyiket nyitja meg, hogy a pumpába ki és be- folyjon a víz.
    - if(in is neighbour of standingOn)
      * than setActiveInput(in);
    - if(out is neighbour of standingOn)
      * setActiveOutput(out);
    1. Pipe
* Felelősség

Ez az osztály implementálja a játékban a csöveket. A csöveket ki lehet lyukasztani és meg lehet javítani, továbbá csatlakoztatni lehet őket más objektumokhoz (összesen maximum kettőhöz), és összesen 1 játékos állhat rajtuk. A csöveket a játékosok ragacsossá tudják tenni, továbbá a nomad típusú játékosok csúszóssá is.

* Ősosztályok

Node  Pipe

* Interfészek

Tickable interfészt implementálja.

* **Attribútumok:** *A felsoroltokon kívül rendelkezik még a Node absztrakt ősosztály attribútumaival.*
  + **capacity**: Azt mutatja mennyi a kapacitása ennek a csőnek (vagyis azt, mennyi víz fér bele).
  + **isSlippery:** Csúszós-e a cső felszíne.
  + **isStickey:** Ragacsos-e a cső felszíne.
  + **unbreakableTill:** Cső megfoltozásakor generált véletlen szám, ameddig nem lehet újra kilyukasztani.
* Metódusok
  + **double getCapacity()**: Visszaadja a capacity értékét, vagyis azt, mennyi víz fér még bele.
  + **void repairNode(Character character)**: Megjavítja/megfoltozza a csövet és generál egy véletlen számot, ameddig nem lehet újra kilyukasztani, ezt a számot a unbreakableTill-be menti.
    - if(Pipe is broken)
      * than isBroken = false;
      * unbreakableTill = random int;
  + **void breakNode(Character character)**: Kilyukasztja, elrontja a csövet akkor, ha az unbreakableTill 0, egyéb különben nem lehet lyukasztani
    - if(Pipe is not Broken and unbreakableTill == 0)
      * than isBroken = true;
  + **void tick()**: Körönként számolja a flowRatet, és csökkenti a unbreakableTill-t ha az nem érte el a 0-t.
    - calculateFlowRate();
    - if(unbreakableTill > 0) than unbreakableTill -= 1;
  + **void setSurface(string type, Character c):** A cső felszínét állítja be a paraméterként beadott típusúra. Ha a type paraméter sticky akkor az isStickey lesz igaz, ha slippery akkor az isSlippery, ha pedig none akkor mind a kettő false lesz.
    - if(type == sticky) than isSticky=true;
    - else if(type == slippery) than isSlippery = true;
    1. Pump
* Felelősség

Ez a játékban szereplő pumpák osztálya. Egy pumpa állítható, el tud romlani, és meg lehet javítani. Egy pumpán több játékos is állhat egyszerre.

* Ősosztályok

Node  Pump

* Interfészek

Tickable interfészt implementálja.

* **Attribútumok:** *A felsoroltokon kívül rendelkezik még a Node absztrakt ősosztály attribútumaival.*
  + **internalBufferLevel**: ezt arra használjuk, hogy a flowRate-et számoljuk.
  + **activeInput**: ez a pumpa bemenetei közül tárolja azt a csövet, amelyik aktív.
  + **activeOutput**: a pumpa kimenetei közül tárolja azt a csövet, amelyik aktív.
* Metódusok
  + **void repairNode(Character character)**: Megjavítja a pumpát.
    - if(Pump is broken)
      * than isBroken = false;
  + **void breakNode(Character character)**: Elrontja a pumpát.
    - if(Pump is not Broken)
      * than isBroken = true;
  + **void setActiveInput(Pipe p)**: Beállítja az activeInput-ot, azaz hogy, a szomszédjai közül a megadott csőből folyjon be víz a pumpába.
    - activeInput = p;
  + **void setActiveOutput(Pipe p)**: Beállítja az activeOutput-ot azaz hogy, a szomszédjai közül a megadott csőbe folyjon ki víz a pumpából.
    - activeOutput = p;
  + **void calculateFlowRate()**: Számolja a flowRatet
    - if(Pump is not broken)
      * than if(absorbers conatins activeInput)
        + than if(absorbers conatins activeOutput)

setFlowRate(activeInput flowRate);

activeOutput += flowRate;

* + - * + else internalBufferLevel += activeInput flowRate;
      * else if(internalBufferLevel > 0)
        + than setFlowRate(min{internalBufferLevel, activeInput capacity});
        + activeOutput flowRate += flowRate;
    - else
      * setFlowRate(0);
      * internalBufferLevel += activeInput flowRate;
  + **void tick()**: Időközönként (random) elrontja a pupmpát, és számolja a flowRate-et.
    - calculateFlowRate();
    - if(random int 1) breakNode();
    1. Tickable
* Felelősség

Ez az interfész minden objektumnak, aminek időközönként kell csinálnia valamit.

* Ősosztályok

Nincsenek ősosztályok.

* Metódusok
  + ***void tick()***: Az adott időközönként végzett tevékenységre való megvalósítandó metódus.
    1. WaterSpring
* Felelősség

A WaterSprring egy olyan speciális elem, amivel a ciszternát tudjuk feltölteni. Nem lehet sem elrontani, sem megjavítani, illetve nincs maximum kapacitása.

* Ősosztályok

Node  WaterSpring

* Interfészek

Tickable interfészt implementálja.

* **Attribútumok:** *A Node absztrakt ősosztály attribútumaival rendelkezik.*
* Metódusok
  + **void addCharacter(Character character)**: Hozzáad egy karaktert a charactersOn listához.
  + **void repairNode(Character character)**: Beállítjuk, hogy ez nem lehetséges, kivételt fog dobni.
  + **void breakNode(Character character)**: Beállítjuk, hogy ez nem lehetséges, kivételt fog dobni.
  + **void tick()**: Időközönként számolja a flowRate-et.
    - while(iterate throught neighbours)
      * if(neighbour is not borken)
        + than neighbour addFlowRate(waterSpring, 1);
        + endif

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

* + 1. Pálya kiírása
       - Leírás

A program betölti és kirajzolja a konzolra a játékpályát.

* + - * Ellenőrzött funkcionalitás, várható hibahelyek

Ellenőrizzük, hogy a program helyesen tölti-e be a pályát. Hiba lehet, ha nem úgy néz ki a pálya, mint ahogy elvárnánk; vagy ha nem található a fájl.

* + - * Bemenet

*load*

* + - * Elvárt kimenet

*Successfully loaded map.*

(megjegyzés: ezután minden tesztesetnél feltételezzük, hogy a pálya létezik a teszt lefutásakor)

* + 1. Játékos hozzáadása
       - Leírás

A program fölvesz egy új játékost a pályára.

* + - * Ellenőrzött funkcionalitás, várható hibahelyek

Ellenőrizzük, hogy a játékost jó helyre vettük-e föl, és jó játékost. Hiba lehet, ha nem vesz fel játékost, vagy rossz helyre, rossz fajtát.

* + - * Bemenet

*add*𝖰 *character*𝖰 *place*𝖰 *<karakter neve (tetszőleges)>* 𝖰 *<node helye>*

* + - * Elvárt kimenet

*Successfully added character.*

* + 1. Új pumpaelem felvétele
       - Leírás

A szerelő magához vesz egy új pumpát a ciszternáktól.

* + - * Ellenőrzött funkcionalitás, várható hibahelyek

Ellenőrizzük, hogy a szerelő jó helyen áll-e, és hogy felvett-e egy pumpát (ezt üzeneten jelzi a játékosnak, ha tényleg ciszternán áll). Hiba lehet, ha nem értesíti a játékost hogy nem ciszternán áll, vagy nem vette föl a pumpát.

* + - * Bemenet

*add* 𝖰 *node* 𝖰 *pump* 𝖰 *<x koordináta>* 𝖰 *<y koordináta>*

* + - * Elvárt kimenet

*Successfully added pump.*

* + 1. Új csőelem felvétele
       - Leírás

A szerelő magához vesz egy új csövet a ciszternáktól.

* + - * Ellenőrzött funkcionalitás, várható hibahelyek

Ellenőrizzük, hogy a szerelő jó helyen áll-e, és hogy felvett-e egy csövet (ezt üzeneten jelzi a játékosnak, ha tényleg ciszternán áll). Hiba lehet, ha nem értesíti a játékost hogy nem ciszternán áll, vagy nem vette föl a csövet.

* + - * Bemenet

*add* 𝖰 *node* 𝖰 *pipe* 𝖰 *<x koordináta>* 𝖰 *<y koordináta>*

* + - * Elvárt kimenet

*Successfully added pipe.*

* + 1. Pumpa elhelyezése
       - Leírás

A szerelő új pumpát telepít a hálózatra.

* + - * Ellenőrzött funkcionalitás, várható hibahelyek

Ellenőrizzük, hogy a szerelő jó helyen áll-e, (üzeneten jelzi a játékosnak, ha nem jó helyen áll), és hogy van-e pumpa nála (ha nincs, akkor is szól a játékosnak). Hiba lehet, ha nem telepíti, pedig van nála és jó helyen áll; ha telepít úgy is, ha nincs nála; vagy ha rossz helyre telepíti.

* + - * Bemenet

*character* 𝖰 *place* 𝖰 *pump*

* + - * Elvárt kimenet

*Successfully deployed pump.*

* + 1. Cső áthelyezése
       - Leírás

Adott cső bekötése a hálózatba.

* + - * Ellenőrzött funkcionalitás, várható hibahelyek

Ellenőrizzük, hogy a szerelő jó helyen áll-e, és hogy van-e cső nála (ha bármelyik nem teljesül, azt jelzi a játékosnak). Hiba lehet, ha nem jó helyen teszi le a csövet, ha letesz egy csövet, pedig nincs nála, vagy ha nem tesz le csövet, pedig lehetne.

* + - * Bemenet

*character* 𝖰 *place* 𝖰 *pipe*

* + - * Elvárt kimenet

*Successfully placed pipe.*

* + 1. Karakter mozgatás
       - Leírás

A karakterek mozgatása egyik node-ról a másikra.

* + - * Ellenőrzött funkcionalitás, várható hibahelyek
      * Az adott karakter áthelyezését vizsgálja, ami történhet csőre, pumpára, valamint a szerelők esetében ciszterna elemre. Ehhez lekéri a szomszédos elemeket. Ha csőre szeretne lépni, akkor megvizsgálja azt is, hogy azon tartózkodik-e másik játékos. Ha igen, ezt üzenet jelzi. Egyéb esetben áthelyezi a játékost. Ha a lépés egy cső elemre történik, megnézi annak csúszósságát. Ha a cső csúszik, akkor a játékos annak egyik végéhez kapcsolódó elemre kerül. Ha a cső ragad, akkor a továbblépési idő számlálója elindul. Hiba lehet, ha mozgatja a játékost, pedig nem szabadna, vagy ha nem mozgatja a játékost, pedig az tudna a célra lépni.
      * Bemenet

*character* 𝖰 *move*

* + - * Elvárt kimenet

*Successfully moved character.*

* + 1. Cső lyukasztása
       - Leírás

Adott karakter kilyukasztja a csövet, amin éppen áll.

* + - * Ellenőrzött funkcionalitás, várható hibahelyek

Ellenőrizzük, hogy jó helyen áll-e, nincs-e már eltörve a cső, vagy a legutóbbi megjavítás óta eltelt idő még nem érte el a megadottat. Bármelyik feltétel nem teljesül, azt jelzi a játékosnak, egyéb esetben kilyukasztja a csövet. Hiba lehet, ha kilyukasztja a csövet, amit nem szabadna, vagy ami már lyukas, illetve ha nem lyukasztja ki, pedig ki szabadna.

* + - * Bemenet

*character* 𝖰 *break*

* + - * Elvárt kimenet

*Successfully broken pipe.*

* + 1. Pumpa elállítása
       - Leírás

Adott karakter elállítja a pumpát, amin éppen áll.

* + - * Ellenőrzött funkcionalitás, várható hibahelyek

Ellenőrizzük, hogy jó helyen áll-e, ha nem, azt jelezzük a játékosnak. Hiba lehet, ha nem értesíti a játékost hogy rossz helyen áll, vagy értesíti, pedig jó helyen állt, illetve ha nem állítja át a pumpát, pedig lehetne.

* + - * Bemenet

*character* 𝖰 *set*

* + - * Elvárt kimenet

*Successfully set pump.*

* + 1. Pumpa javítása
       - Leírás

A szerelő megjavítja a pumpát, amin áll.

* + - * Ellenőrzött funkcionalitás, várható hibahelyek

A szerelőt egy Pumpára helyezi és ellenőrzi, hogy a pumpa törött-e. Ha igen, végrehajtja a műveletet. Egyéb esetben jelzi a problémát. Hiba lehet, ha nem értesíti a játékost hogy nem pumpán áll, vagy ha nem javítja meg, pedig meg kellene.

* + - * Bemenet

*character* 𝖰 *repair*

* + - * Elvárt kimenet

*Successfully repaired pump.*

* + 1. Cső javítása
       - Leírás

A szerelő megjavítja a csövet, amin áll.

* + - * Ellenőrzött funkcionalitás, várható hibahelyek

A szerelőt egy csőre helyezi és ellenőrzi, hogy a cső lyukas-e. Ha igen, végrehajtja a műveletet. Egyéb esetben jelzi a problémát. Hiba lehet, ha nem értesíti a játékost hogy nem pumpán áll, vagy ha nem javítja meg, pedig meg kellene.

* + - * Bemenet

*character* 𝖰 *repair*

* + - * Elvárt kimenet

*Successfully repaired pipe.*

* + 1. Cső csúszóssá tétele
       - Leírás

A nomád szabotőr csúszóssá teszi a csövet, amin éppen áll.

* + - * Ellenőrzött funkcionalitás, várható hibahelyek

A nomádot egy csőre helyezi és ellenőrzi, hogy az a cső csúszós-e. Ha nem csúszós csövön áll, végrehajtja a műveletet. Egyéb esetben jelzi a problémát. Hiba lehet, ha nem értesíti a játékost hogy nem csövön áll, vagy ha nem teszi csúszóssá, pedig kellene.

* + - * Bemenet

*character* 𝖰 *slippery*

* + - * Elvárt kimenet

*Successfully made pipe slippery. Slippery when wet.*

* + 1. Cső ragadóssá tétele
       - Leírás

A nomád szabotőr ragadóssá teszi a csövet, amin éppen áll.

* + - * Ellenőrzött funkcionalitás, várható hibahelyek

A nomádot egy csőre helyezi és ellenőrzi, hogy az a cső ragadós-e. Ha nem ragadós csövön áll, végrehajtja a műveletet. Egyéb esetben jelzi a problémát. Hiba lehet, ha nem értesíti a játékost hogy nem csövön áll, vagy ha nem teszi ragadóssá, pedig kellene.

* + - * Bemenet

*character* 𝖰 *sticky*

* + - * Elvárt kimenet

*Successfully made pipe sticky.*

###### A tesztelést támogató programok tervei

A tesztelést végző segéd program teszt eseteket olvas be majd futtatja az alkalmazást és végrehajtja a megadott teszt esetet. A teszt esetek az előbbiekben specifikált bemeneti nyelven vannak írva, illetve minden teszt esethez tartozik egy elvárt kimenet. A futtatást követően a tesztelő program összehasonlítja az elvárt kimenetet a tényleges kimenettel, ha ez nem egyezik, akkor a teszteset során valamilyen hibába botlottunk, ekkor a tesztelő jelzi hogy a teszt futása hibás volt. Abban az esetben ha meg egyezik az elvárt kimenet az aktuális kimenettel a tesztet sikeresnek titulálja a tesztelő program.

#### Prototípus beadása

###### Fordítási és futtatási útmutató

Megjegyzés: A Tervekhez képest a character setPump metódusa nem absztrakt, hanem implementál, mivel nincs szükség a felüldefiniálásra a leszármazottakban.

* + 1. Fájllista

|  |  |  |  |
| --- | --- | --- | --- |
| **Fájl neve** | **Méret** | **Keletkezés ideje** | **Tartalom** |
| Main.java | 17 334 | 2023. február 23. | Main fájl |
| Map.java | 12 617 | 2023. március 1. | Ez az osztály felelős a játék pályájáért. |
| ProtoTest.java | 10 551 | 2023. május 12. | A prototípus tesztelését valósítja meg. |
| Tickable.java | 122 | 2023. március 1. | Interfész a tick-elhető osztályoknak. |
| Character.java | 4 496 | 2023. március 1. | A játékban szereplő karakterekért felelős osztály. Absztrakt osztály az összes karakternek. |
| Nomad.java | 552 | 2023. március 1. | A nomád karaktert reprezentáló osztály. |
| Plumber.java | 4 073 | 2023. március 1. | A szerelő karaktert reprezentáló osztály. |
| InvalidPlayerActionException.java | 262 | 2023. március 1. | Kivételt dob, ha egy játékos érvénytelen műveletet próbál végrehajtani. |
| InvalidPlayerMovementException.java | 268 | 2023. március 1. | Kivételt dob, ha egy játékos érvénytelen pozícióba próbál mozogni. |
| NotFoundExeption.java | 255 | 2023. március 1. | Kivételt dob, ha egy játékos nem található a szomszédos elemen, ahova mozogni akar. |
| ObjectFullException.java | 227 | 2023. március 1. | Kivételt dob, ha egy objektum elérte maximum kapacitását. |
| NotJSONArray.java | 49 387 | 2023. május 2. | A map kiirasahoz tartozo file beolvasasahoz sugseges osztaly |
| NotJSONException.java | 1 067 | 2023. május 2. | A map kiirasahoz tartozo file beolvasasahoz sugseges osztaly |
| NotJSONObject.java | 86 517 | 2023. május 2. | A map kiirasahoz tartozo file beolvasasahoz sugseges osztaly |
| NotJSONPointer.java | 9 070 | 2023. május 2. | A map kiirasahoz tartozo |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | file beolvasasahoz sugseges osztaly |
| NotJSONPointerException.java | 556 | 2023. május 2. | A map kiirasahoz tartozo file beolvasasahoz sugseges osztaly |
| NotJSONPropertyIgnore.java | 664 | 2023. május 2. | A map kiirasahoz tartozo file beolvasasahoz sugseges osztaly |
| NotJSONPropertyName.java | 747 | 2023. május 2. | A map kiirasahoz tartozo file beolvasasahoz sugseges osztaly |
| NotJSONString.java | 745 | 2023. május 2. | A map kiirasahoz tartozo file beolvasasahoz sugseges osztaly |
| NotJSONTokener.java | 14 980 | 2023. május 2. | A map kiirasahoz tartozo file beolvasasahoz sugseges osztaly |
| NotJSONWriter.java | 12 869 | 2023. május 2. | A map kiirasahoz tartozo file beolvasasahoz sugseges osztaly |
| Cistern.java | 1 503 | 2023. március  17. | Ciszterna node-ot reprezentáló osztály. |
| Node.java | 5 960 | 2023. március  17. | A Node osztály a pályára helyezhető összes objektum alaposztálya. |
| Pipe.java | 1 207 | 2023. március  17. | Cső node-ot reprezentáló osztály. |
| Pump.java | 2 227 | 2023. március  17. | Pumpa node-ot reprezentáló osztály. |
| WaterSpring.java | 1 323 | 2023. március  17. | Egy speciális node, amit a ciszterna feltöltéséhez használható. |
| tesztek.txt | 435 | 2023. május 14 | Teszt inputok |
| forditas.txt | 567 | 2023. 05. 14. | A fordításhoz a segítség |

* + 1. **Fordítás**

forditas terminalban!!!!!!!:

mkdir .\out

cp .\lib\\*.jar .\out

javac -d .\out\ .\src\main\java\com\ez\_mode\Main.java .\src\main\java\com\ez\_mode\\*.java

.\src\main\java\com\ez\_mode\characters\\* .\src\main\java\com\ez\_mode\exceptions\\*

.\src\main\java\com\ez\_mode\objects\\* .\src\main\java\com\ez\_mode\notJson\\* -cp ".\lib\log4j-api-2.20.0.jar;.\lib\log4j-core-2.20.0.jar"

* + 1. Futtatás

futtatas terminalbol!!!!!!!:

cd ./out

java -cp ";%CLASSPATH%;.\log4j-api-2.20.0.jar;.\log4j-core-2.20.0.jar" com.ez\_mode.Main

cmdből a cp-t át kell írni copyra, és a ;-t át kell írni :ra

###### Tesztek jegyzőkönyvei

* + 1. MapPrintTest

|  |  |
| --- | --- |
| **Tesztelő neve** | Sőregi Larina |
| **Teszt időpontja** | 2023.05.15.9:30 |

* + 1. **CharacterAddTest**

|  |  |
| --- | --- |
| **Tesztelő neve** | Farkas Bence |
| **Teszt időpontja** | 2023.05.15.10:01 |

* + 1. AddNewPumpTest

|  |  |
| --- | --- |
| **Tesztelő neve** | Sőregi Larina |
| **Teszt időpontja** | 2023.05.15.10:02 |

* + 1. **AddNewPipeTest**

|  |  |
| --- | --- |
| **Tesztelő neve** | Farkas Bence |
| **Teszt időpontja** | 2023.05.15.10:00 |

* + 1. PlacePumpTest

|  |  |
| --- | --- |
| **Tesztelő neve** | Sőregi Larina |
| **Teszt időpontja** | 2023.05.15.9:40 |

* + 1. **PlacePipeTest**

|  |  |
| --- | --- |
| **Tesztelő neve** | Farkas Bence |
| **Teszt időpontja** | 2023.05.15.10:06 |

* + 1. MakePipeStickyTest

|  |  |
| --- | --- |
| **Tesztelő neve** | Sőregi Larina |
| **Teszt időpontja** | 2023.05.15.9:45 |

* + 1. **MoveCharacterTest**

|  |  |
| --- | --- |
| **Tesztelő neve** | Farkas Bence |
| **Teszt időpontja** | 2023.05.15.10:08 |

* + 1. BreakPipeTest

|  |  |
| --- | --- |
| **Tesztelő neve** | Sőregi Larina |
| **Teszt időpontja** | 2023.05.15.9:50 |

* + 1. **SetPumpTest**

|  |  |
| --- | --- |
| **Tesztelő neve** | Farkas Bence |
| **Teszt időpontja** | 2023.05.15.10:10 |

* + 1. RepairPumpTest

|  |  |
| --- | --- |
| **Tesztelő neve** | Sőregi Larina |
| **Teszt időpontja** | 2023.05.15.9:55 |

* + 1. **RepairPipeTest**

|  |  |
| --- | --- |
| **Tesztelő neve** | Farkas Bence |
| **Teszt időpontja** | 2023.05.15.10:11 |

* + 1. **MakePipeSlipperyTest**

|  |  |
| --- | --- |
| **Tesztelő neve** | Sőregi Larina |
| **Teszt időpontja** | 2023.05.15.10:00 |

1. **Grafikus felület specifikációja**

###### A grafikus interfész

A szoftver elindításakor megjelenik a főmenü, ahol el tudjuk indítani a játékot:

Itt további

gombok is szerepelni fognak.

Ezen kívül, a különböző elemek a pályán:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| *Nomád szabotőr* | *Szerelő* | *Homok* | *Üres cső* |
|  |  |  |  |
| *Vízzel teli cső* | *Lyukas cső* | *Eltörés* | *Javítás* |
|  |  |  |  |
| *Vízzel teli pumpa* | *Üres pumpa* | *Rossz pumpa* | *Csúszós cső* |
|  |  |  |  |
| *Ragadós cső* | *Ciszterna* | *Vízforrás* | *Pumpa beállítása* |
|  |  |  |  |
| *Fel- és lefelé*  *mozgás* | *Jobbra és balra*  *mozgás* | *Cső fölvétele* | *Pumpa fölvétele* |

Egy minta a pálya kinézetéről:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **GAME End game** | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

###### A grafikus rendszer architektúrája

* + 1. A felület működési elve

A modell *push* alapú: bármilyen változás vagy esemény bekövetkezte (esetünkben egérrel kattintás) értesíti az összes osztályt a változásról, ami az adott osztályt érinti és a megfelelő metódust hívja meg.

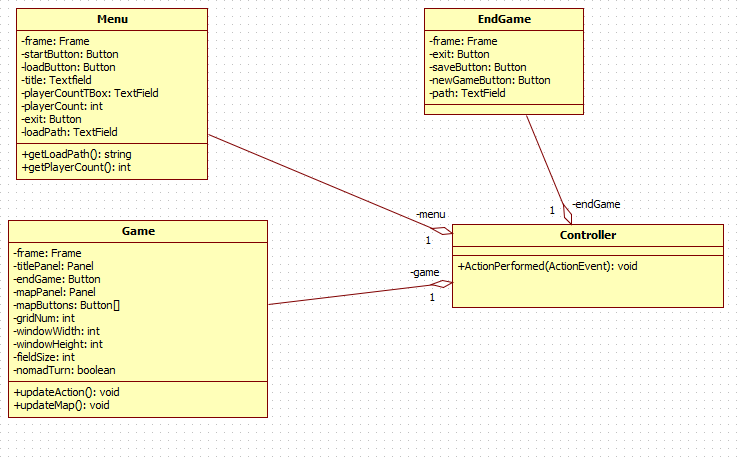
A program elindításakor egy menü indul el, ezért a *Menu* osztály felelős. Ennek a feladata a

játék elindítása, kilépés a játékból, játékosok hozzáadása, valamit egy meglévő játék betöltése. A *Controller* osztály az egyes gombok lenyomását figyelni. A menüt és a játékot is teljes egészében egérrel irányítjuk, ha rákattintunk egy gombra, meghívja az actionPerformed

metódust. A játék futása közben csak a legalsó sorban lévő ,,action bar”-on lévő

tevékenységek végezhetők el az adott karakterrel. Ez a nomád és a szerelő esetében változhat, például a szerelőnél repair ikon, a nomádnál a break ikon jelenik meg. Ez attól függően

változik, hogy melyik csapat következik, melyik csapat köre van éppen.

* + 1. **A felület osztály-struktúrája**

###### A grafikus objektumok felsorolása

* + 1. Controller
* Felelősség

Az osztály, amely figyeli a gombok eseményeit. Több metódus, de működésük elve hasonló.

* + 1. Game
* Felelősség

Magát a pályát kirajzoló és azt, valamint a játékos tevékenységét kezelő osztály.

* Attribútumok
  + **…ImagePath (pl. cisternImagePath)**: a képek helyei, amiket használunk a pálya elemeihez
  + **gridNum**: hogy mekkora a pálya
  + **windowWidth, windowHeight**: az ablak szélessége és magassága
  + **fieldSize**: hogy mekkora egy-egy mező a pályán
  + **frame**: maga az ablak
  + titlePanel
  + **textField**
  + mapPanel
  + **mapButtons**
  + endGame

ezek a különböző megjelenített dolgok, a gombok amikkel kezeljük a játékot, a panelek amiken láthatjuk az információkat, stb.

* + **…Icon (pl. cisternIcon)**: az ikonok, vagyis a pálya elemeinek kinézete
  + **nomadTurn**: egy változó, ami megmondja, melyik csapat köre van éppen
* Metódusok
  + **public void actionMap():** update-eli a pályán látható elemeket
  + **public void updateAction()**: update-eli az action bar elemeit
    1. Menu
* Felelősség

A program elindításakor megjelenő menüért felel.

* Attribútumok
  + **frame**: maga az ablak
  + **startButton**: a játék elindítására szolgáló gomb
  + **title:** a panel címe (vagyis, hogy „Menu”), majd, hogy melyik csapat köre
  + **loadButton**: előzőleg mentett pálya betöltése
  + **title**: kiírja azt, hogy Menu
  + **playerCountTBox**: Mennyi játékos lesz beírható ide
  + **playerCount**: Játékosszám
  + **exit**: kilépés gomb
  + **loadPath**: előzőleg mentett pálya elérési útjának megadása
* Metódusok
  + **public void getLoadPath():** elérési útja a pályának
  + **public void getPlayerCount():** játékosok (karakszerek/csapat) száma
    1. EndGame
* Felelősség

A játék végén megjelenő ablak. Itt lehet kilépni, új játékot indítani, menteni a pályát és ennek elérési útját megadni.

* Attribútumok
  + **frame**: maga az ablak
  + **exit**: gomb a játékból való kilépéshez
  + **saveButton:** pálya kimentése fájlba
  + **newGameButton:** új játékot indít, vagyis visszadob a start menübe
  + **path:** a mentendő fájl elérési útját ide kell megadni

###### Kapcsolat az alkalmazói rendszerrel

**14. Összefoglalás**

***14.1 A projektre fordított összes munkaidő***

|  |  |
| --- | --- |
| **Tag neve** | **Munkaidő (óra)** |
| *Sőregi Larina* | 46 |
| *Nyilas Péter* | 43 |
| *Kis Benedek* | 45 |
| *Farkas Bence* | 42 |
| *Zsigmond Botond* | 30 |
| **Összesen** | **206** |

***A feltöltött programok forrássorainak száma***

|  |  |
| --- | --- |
| **Fázis** | **Kódsorok száma** |
| Szkeleton |  |
| Prototípus |  |
| Grafikus változat |  |
| **Összesen** |  |

***14.2 Projekt összegzés***

**14.2.1 Mit tanultak a projektből konkrétan és általában?**

**14.2.2 Mi volt a legnehezebb és a legkönnyebb?**

**14.2.3 Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?**

**14.2.4 Ha nem, akkor hol okozott ez nehézséget?**

**14.2.5 Milyen változtatási javaslatuk van?**

**14.2.6 Milyen feladatot ajánlanának a projektre?**

**14.2.7 Egyéb kritika és javaslat**