

## 2. Követelmény, projekt, funkcionálitás

### 2.1 Bevezetés

#### 2.1.1 Cél

A dokumentum célja a projekt egyes követelményeinek rögzítése. A játéknak az itt definiált elvárások szerint kell működnie, és a fejlesztési folyamat során ezek szerint kell eljárni.

#### 2.1.2 Szakterület

A kialakítandó szoftver egy játék, tehát elsősorban szabadidős tevékenységeknek használatos. Emellett, mivel multiplayer, ezért csapatépítésre is alkalmas. Továbbá stratégia alkalmazása szükséges, így a játék logikájának megértése feltétlenül szükséges, és idővel a képességeink fejlődhetnek.

#### 2.1.3 Definíciók, rövidítések

*DC - Discord*

*IntelliJ - IntelliJ IDEA - Java fejlesztőkörnyezet (IDE)*

*Drive - Google Drive*

*repo - Git repository*

*Word - Microsoft Office Word*

*JRE - Java Runtime Environment*

#### 2.1.4 Hivatkozások

A feladat a Szoftver projekt laboratórium honlapjáról :

<https://www.iit.bme.hu/file/11582/feladat>

#### 2.1.5 Összefoglalás

A dokumentum további részeiben:

- A feladat szövegéből kiindulva leírjuk, pontosítjuk a funkciókat
- Áttekintjük a potenciális felhasználókat és a korlátozásokat
- Meghatározzuk a funkcionális követelményeket
- Felsoroljuk az egyéb követelményeket
- Definiáljuk a use-caseket, és ábrázoljuk a hozzá tartozó diagramot
- Megírjuk a szótárat, amelyben pontosan meghatározzuk a projekt során használt alapvető fogalmakat
- Elkészítjük a projekt tervet és a naplót

## 2.2 Áttekintés

### 2.2.1 Általános áttekintés

A játékban a grafikus felület biztosítja a pálya megtekintését a játékosok részére, az irányítás egér és billentyűzet segítségével történik. A fontosabb alrendszerök az alábbiak:

- a játék belső logikája
- a játékvezérlő
- a grafikus felület

A játékvezérlő a felhasználótól kapott bemenetet továbbítja a játék belső logikájának. A grafikus felület a játék belső logikájától megkapja a játék állapotát, és azt kirajzolja a grafikus felületre.

## 2.2.2 Funkciók

A szoftver feladata egy több játékost igénylő játék megvalósítása. A játékban egy sivatagon át bonyolult csőrendszer szállítja a vizet a hegyi forrásokból a sivatagon túl elterülő városok ciszternáiba. A sivatagi vízhálózat a következő komponensekből áll: források, ciszternák, pumpák és csövek. A forrásokat, ciszternákat és pumpákat csövekkel lehet összekötni. Fontos tudnivaló, hogy a forrásokat nem lehet közvetlenül a ciszternákkal összekötni, közöttük legalább egy pumpának kell elhelyezkednie. A forrás, ciszterna és pumpa komponenseket aktív elemeknek tekintjük, hiszen ők biztosítják a víz továbbítását/mozgatását a hálózaton belül. Ilyen értelemben a csövekre mondhatjuk, hogy passzív alkotóelemei a rendszernek, mert ők csak tárolják a hozzájuk érkező vizet.

Minden forrásból az összes hozzáköött csőbe folyik víz, ehhez hasonlóan minden ciszternába az összes belekötött csőből érkezhet víz (az üres csőből nyilván nem érkezik víz). A forrásból nem tud kifogni a víz, valamint a ciszterna sem tud teljesen megtelni vízzel a játék során. Pumpa A pumpákra jellemző egy véges egész szám, amely meghatározza az egy pumpába köthető csövek maximális számát. minden pumpán külön-külön állítható, hogy éppen melyik belekötött csőből melyik másik csőbe folyjon a víz, azonban egyszerre csak egy bemeneti és egy kimeneti cső lehet. A többi rákötött cső eközben el van zárva. A pumpa csak akkor tud vizet pumpálni egy csőbe, ha a cső szabad kapacitása ezt lehetővé teszi. Pontosabban csak üres csőbe folyhat tovább víz, egy vízzel teli csőbe nem tud tovább folyni a víz. A pumpák mindegyike rendelkezik egy víztartállyal, amit a víz átemelése közben használ átmeneti tárolóként. A pumpák véletlen időközönként el tudnak romlani. Ilyenkor a pumpa nem tudja beszívni a bemeneti/aktív csőből a vizet, viszont az átmeneti tárolójából még át tud folyni a víz a kimeneti csőbe. Elromlott pumpát továbbra is lehet állítani. Ha egy nem működő pumpánál megindul a vízáramlás, akkor az először az átmeneti tárolójába szívja be a vizet, és csak ebből az átmeneti tárolóból tud tovább folyni a víz a kimeneti csőbe.

Csövek mozgatása A csőhálózat bővíthető, változtatható. A csövek kellően rugalmasak ahhoz, hogy az egyik végüket lecsatlakoztatva egy másik aktív elemhez elvihetők és ott felcsatlakoztathatók legyenek. A ciszternáknál folyamatosan készülnek az új csövek, amelyek egyik vége a ciszternához kapcsolódik, a másik azonban szabad. Azaz a ciszternák felől tetszőlegesen sok új cső bevitelű a vízhálózatba. A szabad végű csövekből a csőbe betáplált víz a homokba folyik.

Karakterek tevékenysége A játékban kétféle karakterrel lehet játszani: a szerelő és szabotőr karakterrel. A csőhálózatot a szerelők tartják karban. Ők javítják meg az elromlott pumpákat, ők állítják át a pumpákat, hogy minden lehetséges módon áthidaljanak a hálózaton, és ha egy cső kilyukad, az ő dolguk a cső megfeszítése is. A kilyukadt csövekből a víz kifolyik a homokba, a csövek végén lévő pumpához már nem jut belőle. A szerelők dolga a ciszternáknál lévő szabad csövekkel a hálózat kapacitásának növelése. A szerelők a ciszternáknál magukhoz tudnak venni új pumpát is, amit egy cső közepén tudnak elhelyezni. A csövet ehhez két részre kell vágni, és a két részt a pumpához kell csatlakoztatni. Akár több pumpát is magukhoz tudnak venni.

A hálózaton élnek a szabotőrök is, akik a pumpákat tudják átállítani és a csöveket szokták kilyukasztani. Fontos kiemelni, hogy a szabotőrök nem tudnak új pumpákat és új csöveket a ciszternáknál magukhoz venni, azonban a meglévő csöveket elmozgathatják.

Karakterek mozgása Mivel a sivatag veszélyes hely, a szerelők és a szabotőrök csak a csőhálózaton haladhatnak, azaz egy adott időpillanatban tartózkodhatnak csövön, pumpán, ciszternán vagy forráson. A pumpáknál kikerülhetik egymást, de a csöveken már nem tudnak

elmenni egymás mellett, egy csövön egyszerre csak egy ember állhat. A karaktereket csak szomszédos mezőkre lehet mozgatni. Tevékenységüket is alapvetően befolyásolja, hogy épp milyen mezőn állnak. Egy szerelő csak azt a csövet tudja megfoltogni, és csak azt a pumpát tudja átállítani/megjavítani, amin éppen áll. Ehhez hasonlóan a szabotőr is csak azt a csövet tudja kilyukasztani, és csak azt a pumpát tudja átállítani, amin tartózkodik. Ha egy szerelő pumpát helyezett el egy csövön, akkor a pumpa elhelyezése után a szerelő az újonnan beszerelt pumpára kerül. Egy adott cső mozgatásához is az szükséges, hogy a karakter azon a csövön tartózkodjon.

A karakterek tevékenységét az aktuális mező/elfoglalt komponens állapota is befolyásolja. Például a szerelők kilyukadt csövön nem tudnak pumpát elhelyezni.

A játékot a két csapat legalább 2-2 játékossal játssza. A szabotőrök dolga, hogy minél több víz folyjon el a lyukakon a homokba, a szerelők pedig azon dolgoznak, hogy minél több víz jusson a ciszternákba. Az a csapat nyer, amelyik a játék végére több vizet szerez.

### 2.2.3 Felhasználók

*A játékkal különböző képzettséggel rendelkező felhasználók lépnek kapcsolatba.*

*Képzetlen felhasználó: A felhasználók nagy része, akik csak számítógép-kezelési ismeretekkel rendelkeznek.*

*Képzett felhasználó: Azok a felhasználók, akik rendszeresen, napi szinten órákat töltenek a játékkal.*

*Alkalmazásprogramozó: A játékot fejleszti, karbantartja.*

### 2.2.4 Korlátozások

*A szoftvernek a biztosított virtuális környezetben futnia kell.*

### 2.2.5 Feltételezések, kapcsolatok

*Hivatkozások című bekezdésben beszűrt hivatkozások a feladat leírását tartalmazzák.*

## 2.3 Követelmények

### 2.3.1 Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Ko mm ent
CSŐ001	Egy cső 1 vagy 2 aktív elemhez kapcsolódhat.	Egy csőhöz megpróbálunk 0, valamint 2-nél több aktív elemet kapcsolni.	Alapvető	funkciók	View track	-
CSŐ002	Egy cső lekapcsolható az	Kiválasztunk egy két aktív	Alapvető	funkciók	View track	-

	<i>egyik aktív eleméről.</i>	<i>elemmel rendelkező csövet és lekapcsoljuk az egyik végét.</i>				
CSŐ003	<i>Egy cső hozzákapcsolható egy olyan aktív elemhez, amihez még nincs hozzákapcsolva.</i>	<i>Kiválasztunk egy szabad végű csövet és hozzákapcsoljuk egy aktív elemhez.</i>	Alapvető	funkciók	<i>View track</i>	-
CSŐ004	<i>Ha egy csőnek van szabad vége, akkor mozgatható.</i>	<i>Kiválasztunk egy csövet, lekapcsoljuk egyik szabad végéről és megpróbáljuk mozgatni.</i>	Alapvető	funkciók	<i>View track</i>	-
CSŐ005	<i>Két aktív elemet csak egy cső köthet össze.</i>	<i>Megpróbáljuk két aktív elem összekötését két csővel.</i>	Alapvető	funkciók	<i>View track</i>	-
CSŐ101	<i>Egy cső vagy teljesen tele van vízzel vagy üres.</i>	<i>Egy üres, nem lyukas csőbe vizet folyatunk.</i>	Alapvető	csapat	<i>View track</i>	-
CSŐ102	<i>Ha egy cső lyukas, akkor kifolyik belőle a víz.</i>	<i>Szabotőr karakterrel kilyukasztunk egy csövet, amiben éppen víz folyik.</i>	Alapvető	funkciók	<i>View track</i>	-
CSŐ103	<i>Ha egy csőnek van szabad vége, akkor kifolyik belőle a víz.</i>	<i>Keresünk/csinálunk egy szabad végű csövet, és vizet folyatunk bele.</i>	Alapvető	funkciók	<i>View track</i>	-
CSŐ201	<i>Ha egy cső nem lyukas, akkor egy szabotőr kilyukaszthatja.</i>	<i>Szabotőr karakterrel keresünk egy nem lyukas csövet, és kilyukasztjuk.</i>	Alapvető	funkciók	<i>Pipe puncture</i>	-
CSŐ202	<i>Ha egy cső lyukas, akkor egy szerelő megfoltozhatja.</i>	<i>Szabotőr karakterrel kilyukasztunk egy csövet, majd szerelő karakterrel megfoltozzuk.</i>	Alapvető	funkciók	<i>Pipe repair</i>	-

<i>FORRÁS0 01</i>	<i>Egy forráshoz több cső is hozzáköthető.</i>	<i>Egy forráshoz ebynél több csövet kötünk.</i>	<i>Fontos</i>	<i>csapat</i>	<i>View track</i>	-
<i>FORRÁS0 02</i>	<i>A forrás az összes hozzá kapcsolódó csőbe pumpálja a vizet.</i>	<i>Egy forráshoz ebynél több csövet kötünk.</i>	<i>Alapvető</i>	<i>csapat</i>	<i>View track</i>	-
<i>FORRÁS0 03</i>	<i>A forrás kifogyhatatlan, azaz végtelen mennyiséggű vizet tartalmaz.</i>	<i>Lejátszunk egy játéket, figyeljük a forrásokból kivezető csöveket.</i>	<i>Alapvető</i>	<i>csapat</i>	-	-
<i>FORRÁS0 04</i>	<i>A forrás nem köthető össze közvetlenül egy ciszternával se.</i>	<i>Megpróbálunk összekötni egy forrást egy ciszternával.</i>	<i>Fontos</i>	<i>csapat</i>	<i>View track</i>	-
<i>CISZTERN A001</i>	<i>Egy ciszternához több cső is hozzáköthető.</i>	<i>Egy ciszternához ebynél több csövet kötünk.</i>	<i>Fontos</i>	<i>csapat</i>	<i>View track</i>	-
<i>CISZTERN A002</i>	<i>A ciszterna a hozzá kötött csövekből szívja magába a vizet.</i>	<i>Egy ciszternához ebynél több csövet kötünk és vizet vezetünk ezekbe.</i>	<i>Alapvető</i>	<i>csapat</i>	<i>View track</i>	-
<i>CISZTERN A003</i>	<i>A ciszterna végtelen mennyiséggű vizet tud tárolni.</i>	<i>Lejátszunk egy játéket, figyeljük a ciszternában bemenő csöveket.</i>	<i>Alapvető</i>	<i>csapat</i>	-	-
<i>CISZTERN A004</i>	<i>A ciszternán állva tetszőlegesen sok új, a ciszternához kapcsolódó cső behető a hálózatba a szerelők által.</i>	<i>Szerelő karakterrel a ciszternához megyünk és új csöveket kérünk.</i>	<i>Alapvető</i>	<i>funkciók</i>	<i>Place pipe</i>	-
<i>CISZTERN A005</i>	<i>A szerelők a ciszternáknál magukhoz tudnak venni új pumpákat korlátlan mennyiségen.</i>	<i>Szerelő karakterrel a ciszternához megyünk és új pumpákat kérünk.</i>	<i>Alapvető</i>	<i>funkciók</i>	<i>Pick up element</i>	-
<i>PUMPA00 1</i>	<i>Egy pumpa több csövet is összeköthet.</i>	<i>Egy pumpához ebynél több csövet kötünk</i>	<i>Alapvető</i>	<i>funkciók</i>	<i>View track</i>	-

		<i>szerelő karakterrel.</i>				
PUMPA00 2	<i>A pumpák maximálisan csak a rájuk jellemző mennyiségek csövet tudnak összekötni.</i>	<i>Az adott mennyiségnél több csövet próbálunk egy pumpához kötni.</i>	Alapvető	funkciók	<i>View track</i>	-
PUMPA00 3	<i>Egy pumpán állítható, hogy melyik csőből melyikbe pumpáljon.</i>	<i>Átállítunk egy pumpát úgy, hogy a bemenő csőben legyen víz.</i>	Alapvető	funkciók	<i>Pump switch</i>	-
PUMPA00 4	<i>Egy pumpának egyszerre egy bemenete és egy kimenete lehet.</i>	<i>Átállítunk egy pumpát.</i>	Alapvető	funkciók	-	-
PUMPA00 5	<i>A pumpák véletlen időközönként elromlanak.</i>	<i>Várunk, amíg egy pumpa elromlik.</i>	Alapvető	funkciók	<i>Pump destroy</i>	-
PUMPA00 6	<i>A pumpa csak üres csőbe tud vizet pumpálni.</i>	<i>Átállítunk egy pumpát úgy, hogy vízzel teli csőbe próbáljon pumpálni.</i>	Fontos	funkciók	<i>View track</i>	-
PUMPA00 7	<i>Mindegyik pumpa rendelkezik víztartállyal.</i>	<i>Egy pumpa bemeneti csövébe vizet folyatunk.</i>	Alapvető	funkciók	<i>View track</i>	-
PUMPA00 8	<i>Ha egy pumpánál megáll a vízáramlás, akkor a víztartály még kiürül, azaz még folyik víz.</i>	<i>Egy pumpánál egy másik pumpa átállításával leállítjuk a vízfolyást.</i>	Fontos	csapat	<i>View track</i>	-
PUMPA00 9	<i>Ha egy pumpánál megindul a vízáramlás, akkor először a víztartály telik meg, utána kezd folyni a víz.</i>	<i>Egy pumpánál egy másik pumpa átállításával megindítjuk a vízfolyást.</i>	Fontos	csapat	<i>View track</i>	-
PUMPA01 0	<i>Ha egy cső nem lyukas, akkor egy szerelő kettévághatja azt és egy pumpával kötheti össze a keletkező részeket.</i>	<i>Szerelő karakterrel elhelyezünk egy nem lyukas cső közepére egy pumpát.</i>	Alapvető	funkciók	<i>Place pump</i>	-

PUMPA01 1	<i>A játékosok tudják állítani a pumpákat.</i>	<i>Átállítunk egy pumpát.</i>	Alapvető	funkciók	Pump switch	-
PUMPA01 2	<i>Ha egy pumpa elromlik, akkor a szerelő meg tudja javítani azt.</i>	<i>Szerelő karakterrel keresünk egy rossz pumpát és megjavítuk azt.</i>	Alapvető	funkciók	Pump repair	
MOZGÁS0 01	<i>A játékosok csak a csőhálózaton haladhatnak.</i>	<i>Egy karakterrel lépünk párat.</i>	Alapvető	funkciók	Move Player	-
MOZGÁS0 02	<i>Egy pumpán egyszerre több játékos is tartózkodhat.</i>	<i>Több karakterrel is ugyanarra a pumpára megyünk.</i>	Alapvető	funkciók	Move Player	-
MOZGÁS0 03	<i>Egy csövön egyszerre csak egy játékos tartózkodhat.</i>	<i>Egy foglalt csőre megpróbálunk rálépni egy másik karakterrel.</i>	Alapvető	funkciók	Move Player	-
MOZGÁS0 04	<i>A játékos csak azt a elemet tudja kezelní, amin éppen tartózkodik.</i>	<i>Egy karakterrel elmozdulás nélkül megpróbálunk több elemet is kezelní, közöttük azt, amin éppen állunk.</i>	Fontos	csapat	Move Player	-
MOZGÁS0 05	<i>Ha egy szerelő pumpát helyezett el, akkor arra a pumpára kerül.</i>	<i>Egy szerelővel pumpát helyezünk el.</i>	Opcionális	csapat	Move Player	-
MOZGÁS0 06	<i>A játékosok csak szomszédos elemekre léphetnek.</i>	<i>Egy karakterrel lépünk egyet.</i>	Fontos	csapat	Move Player	
JÁTÉK001	<i>A játékot két csapat játssza, a szerelők és a szabotörök.</i>	<i>Elindítunk egy játékot, megnézzük a csapatokat.</i>	Alapvető	funkciók	View track	-
JÁTÉK002	<i>Egy csapatnak legalább két tagja van.</i>	<i>Elindítunk egy játékot, megnézzük a csapatok tagjainak számát.</i>	Alapvető	funkciók	View track	-

JÁTÉK003	Egy csapatnak pontosan két tagja van.	Elindítunk egy játékot, megnézzük a csapatok tagjainak számát.	Opcionális	csapat	View track	-
----------	---------------------------------------	--	------------	--------	------------	---

### 2.3.2 Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
1.	IntelliJ IDEA	nincs	Opcionális	csapat	-
2.	Git	nincs	Fontos	csapat	-
3.	Java 18	bemutatás	Alapvető	megrendelő	-

### 2.3.3 Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
1.	Skeleton átadása	bemutatás	Alapvető	megrendelő	2023.04.17.
2.	Prototípus átadása	bemutatás	Alapvető	megrendelő	2023.04.24.
3.	Teljes program átadása	bemutatás	Alapvető	megrendelő	2023.05.31.

### 2.3.4 Egyéb nem funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
-	-	-	-	-	-

## 2.4 Lényeges use-case-ek

### 2.4.1 Use-case leírások

Use-case neve	Move player
Rövid leírás	A játékos a csőhálózaton keresztül mozog.
Aktorok	Player
Forgatókönyv	A játékos a csöveken előre vagy hátra mozog.
Alternatív forgatókönyv	A játékos a pumpákról egyik hozzá kapcsolódó csőre lép.
Alternatív forgatókönyv	A játékos egy csőről egy hozzá kapcsolódó pumpára lép
Alternatív forgatókönyv	A játékos egy ciszternára lép.

<b>Alternatív forgatókönyv</b>	A pumpákon a játékosok kikerülik egymást.
<b>Alternatív forgatókönyv</b>	A játékos egy forrásra lép.

<b>Use-case neve</b>	<b>Pipe puncture</b>
<b>Rövid leírás</b>	A szabotőr kilyukaszt egy csövet.
<b>Aktorok</b>	Saboteur
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>Egy szabotőr kilyukaszt egy csövet.</li> <li>A csőből az összes víz kifolyik.</li> <li>A csövön keresztül nem folyik víz tovább.</li> </ol>

<b>Use-case neve</b>	<b>Pump switch</b>
<b>Rövid leírás</b>	A játékos átállít egy pumpát.
<b>Aktorok</b>	Player
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>A pumpa a játékos által kiválasztott csőből fog szívni.</li> <li>A pumpa a játékos által kiválasztott csőbe fogja pumpálni a vizet.</li> </ol>

<b>Use-case neve</b>	<b>Pipe repair</b>
<b>Rövid leírás</b>	A szerelő megjavít egy csövet.
<b>Aktorok</b>	Mechanic
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>Nem folyik ki több víz a csőből.</li> <li>A pumpa ezután tud szívni a csőből.</li> </ol>

<b>Use-case neve</b>	<b>Pump repair</b>
<b>Rövid leírás</b>	A szerelő megjavít egy pumpát.
<b>Aktorok</b>	Mechanic
<b>Forgatókönyv</b>	A pumpa elkezd vizet pumpálni a két beállított csövön keresztül.

<b>Use-case neve</b>	<b>Pump destroy</b>
<b>Rövid leírás</b>	Egy pumpa elromlik.
<b>Aktorok</b>	Controller
<b>Forgatókönyv</b>	A pumpa nem pumpál több vizet egy csőbe se.

<b>Use-case neve</b>	<b>Pipe connect</b>
<b>Rövid leírás</b>	Egy játékos egy csövet egy aktív elemhez köt.
<b>Aktorok</b>	Player
<b>Forgatókönyv</b>	A játékos hozzáköti egy cső szabad végét egy aktív elemhez.

<b>Use-case neve</b>	<b>Pick up element</b>
<b>Rövid leírás</b>	Egy szerelő felvesz egy csövet vagy egy pumpát egy ciszternánál.
<b>Aktorok</b>	Mechanic
<b>Forgatókönyv</b>	Egy szerelő felvesz egy csövet.
<b>Alternatív forgatókönyv</b>	Egy szerelő felvesz egy pumpát.

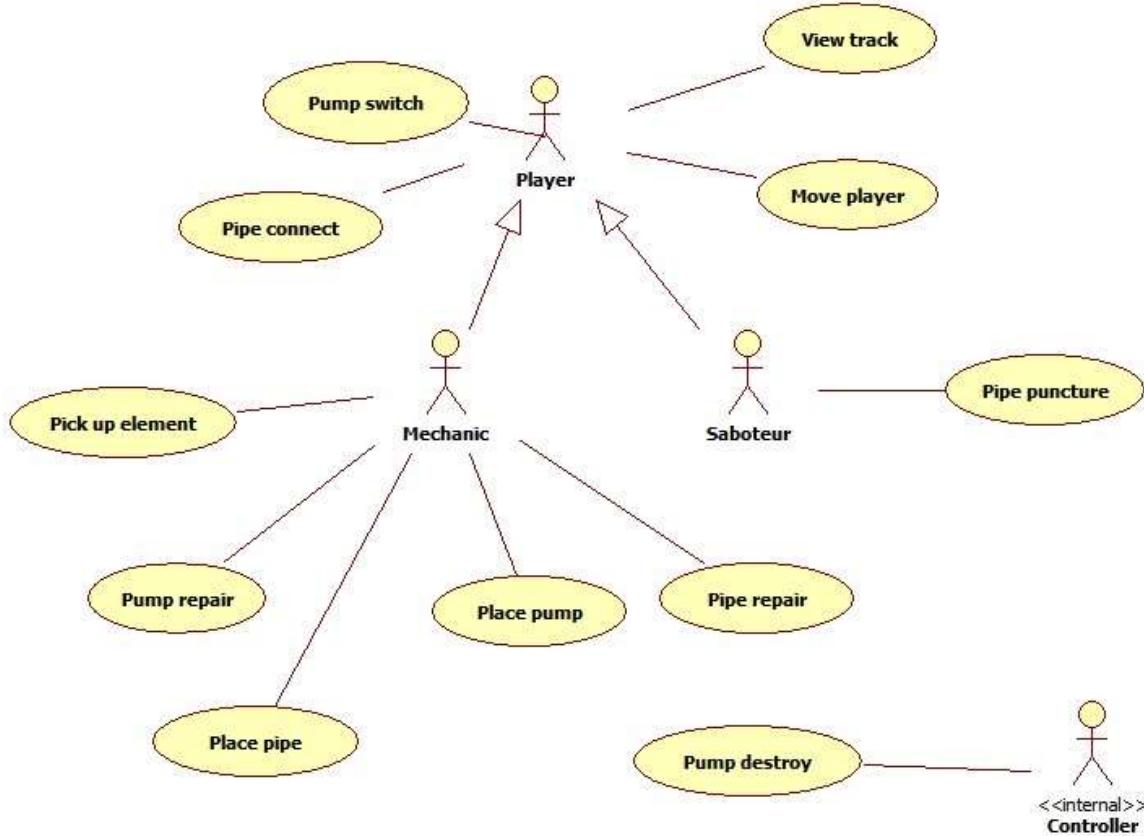
<b>Use-case neve</b>	<b>View track</b>
----------------------	-------------------

<b>Rövid leírás</b>	A játék pályájának a megjelenítése.
<b>Aktorok</b>	Player
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>Kirajzolja a játékos számára a csőhálózat aktuális állapotát.</li> <li>A játékos megtekinti a kirajzolt pályát.</li> </ol>

<b>Use-case neve</b>	<b>Place pump</b>
<b>Rövid leírás</b>	Egy szerelő lerak egy pumpát.
<b>Aktorok</b>	Mechanic
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>A szerelő kettévág egy csövet.</li> <li>A szerelő leteszi a ciszternánál felvett pumpát a két cső közé.</li> </ol>

<b>Use-case neve</b>	<b>Place pipe</b>
<b>Rövid leírás</b>	Egy szerelő lerak egy csövet.
<b>Aktorok</b>	Mechanic
<b>Forgatókönyv</b>	A szerelő leteszi a ciszternánál felvett csövet.

## 2.4.2 Use-case diagram



## 2.5 Szótár

**Aktív elem** a játék olyan komponense, amely a víz haladásáért felelős

**Bemeneti cső** az a cső, amelyből a pumpa aktuálisan szívja a vizet

**Ciszterna** olyan aktív elem, amely vizet szív az összes belekötött csőből, végtelen mennyiséggű vizet tud szívni, az ide befolyt víz a szerelők csapatának ér pontot

**Cső** olyan passzív elem, amely aktív elemeket köt össze, folyik rajta víz

**Csövet kilyukaszt / Lyukas cső** a cső olyan állapotba kerül, amely során a víz kifolyik rajta

**Csövet megfoltoz** a cső lyukas állapotából visszakerül az alap állapotába, így nem folyik ki rajta a víz, hanem halad tovább

**Csövet mozgat** a cső egyik végét le lehet csatlakoztatni egy pumpáról, és azt egy másikra átkötni, illetve a ciszternáknál levő szabad végű csöveget is hozzá lehet kötni egy pumpához

**Forrás** olyan aktív elem, amely vizet pumpál az összes belekötött csőbe, végtelen mennyiséggű vizet tud pumpálni

**Hálózat** a játékban található, összefüggő aktív és passzív elemek rendszere

**Karakter** az egyes játékosokat reprezentálja, lehet őket mozgatni a mezőkön

**Kimeneti cső** az a cső, amelybe a pumpa aktuálisan pumpálja a vizet

**Mező** a játék felületének alapegysége, amely tartalmazhat aktív és passzív elemeket valamint karaktereket, és ezeken keresztül mozoghatnak a játékosok

**Passzív elem** a játék olyan komponense, amely nem felelős a víz haladásáért, csak tárolja azt

**Pumpa** olyan aktív elem, amelybe több cső is bele lehet kötni, azok közül mindig egy bemeneti csőből mozgatja a vizet a kimeneti csőbe, melynek két fázisa a víz szívása és a víz pumpálása, továbbá rendelkezik egy víztartályal

**Pumpa elromlik / Elromlott pumpa** a pumpa olyan állapotba kerül, amely során nem képes vizet szívni és pumpálni, víztartályából a víz a kimeneti csőbe folyik, majd a víztartály kiürül

**Pumpát átállít** megváltozik a pumpa bemeneti és / vagy kimeneti csöve

**Pumpát lerak** amennyiben rendelkezik egy előzetesen felvett pumpával, úgy egy cső kettévágásával csatlakoztatni tudja a pumpát a hálózatba, a pumpa bemeneti és kimeneti csöve a kettévágott cső egy-egy vége lesz

**Pumpát magához vesz** a ciszternáknál készült pumpákból a szerelő elvesz egyet, így később csatlakoztatni tudja a hálózathoz

**Pumpát megjavít** a pumpa elromlott állapotából visszakerül az alap állapotába, így újra képes vizet szívni és pumpálni

**Szabotőr** olyan karakter, akinek az a célja, hogy minél több víz folyjon ki a hálózatból, képes pumpát átállítani és csövet lyukasztani és mozgatni azon a mezőn, amelyiken éppen áll

**Szerelő** olyan karakter, akinek az a célja, hogy minél több víz folyjon a ciszternákba, képes pumpát átállítani, pumpát javítani, csövet megfoltozni, mozgatni, csatlakoztatni, pumpát magához venni és lerakni

**Víz** a játékban az alapvető pontszerző egység, amelynek mozgása a hálózaton keresztüli történik, mértékegység nélküli

**Víz folyik / halad / mozog** a víz a forrásokból a ciszternák felé való mozgatása, amely a csöveken, pumpákon és azok tartályain keresztüli valósul meg

**Víz kifolyik** a víz olyan folyása, mely során nem halad tovább más aktív vagy passzív elembe, hanem kikerül a hálózatból, ez pontot ér a szabotőrök csapatának

**Vizet szív** a víz folyása a pumpa bemeneti csövéből a pumpa víztartályába

**Vizet pumpál** a víz folyása a pumpa víztartályából a pumpa kimeneti csövébe

**Víztartály** olyan passzív elem, amely a pumpához tartozik, és a víz szívása és pumpálása között a vizet tárolja

## 2.6 Projekt terv

hét	feladat	laboralkalom	határidő
2	Követelmény, projekt, funkcionalitás	konzultáció	március. 13. 14:15
3	Analízis modell (I. változat)	konzultáció	márc. 20. 14:15
4	Analízis modell (II. változat)	konzultáció	márc. 27. 14:15
5	Szkeleton tervezése	konzultáció	ápr. 3. 14:15
6	Szkeleton elkészítése	konzultáció	ápr. 17. 14:15
7		- tavaszi szünet -	
8	Prototípus koncepciója	Szkeleton bemutatása	ápr. 24. 14:15
9	Részletes tervezés	konzultáció	máj. 3. labor
10	Prototípus elkészítése	konzultáció	máj. 8. 14:15
11	Grafikus változat tervezés	Prototípus bemutatása	máj. 15. 14:15
12	Grafikus változat elkészítése	konzultáció	máj 31. labor
13		konzultáció	
14	Egyesített dokumentáció	Grafikus verzió bemutatása	jún. 2. 14.00

		egyesített doksi csak elektronikusan	
--	--	--------------------------------------	--

Csoportmunkát támogató eszközök, technikák:

- személyes/online találkozó heti több alkalommal (Messenger, DC)
- dokumentáció Word és Drive segítségével
- a kód írásához IntelliJ-t és Eclipse-t használunk
- a kódot Git segítségével osztjuk meg egymással

Felelősök

Andó Viola	dokumentáció, kód, diagramok
Deé-Lukács András Gergely	dokumentáció, kód egyesítés, beadás helyettes
Kiss Blanka Zselyke (csap.vez.)	dokumentáció, dokumentáció egyesítés, beadás, kód
Skáre Erik	dokumentáció, kód, diagramok
Vörös Vilmos	dokumentáció, kód, diagramok

## 2.7 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2023.03.08. 14:00	3 óra	Andó Deé-Lukács Kiss Skáre Vörös	Értekezlet. A feladat közös értelmezése, funkciók megvitatása, feladatok felosztása Döntés: Andó megcsinálja a szótár felét, Deé-Lukács elkészíti a Use case-eket és a diagramot, Kiss elkészíti a bevezetést és a szótár másik felét, Skáre megírja a követelményeket és funkcionális követelményeket, Vörös megírja a funkciókat
2023.03.08 17:00	1,5 óra	Vörös	Funkciók szöveges megfogalmazása
2023.03.08. 21:00	2 óra	Skáre	Követelmények táblázatos leírása
2023.03.10. 21:00	1,5 óra	Deé-Lukács	Use case-k elkészítése, use case diagram rajzolása
2023.03.10. 15.00	1,5 óra	Andó	Szótár írásának elkezdése
2023.03.11. 11.00	2 óra	Kiss	Szótár befejezésének megírása, Bevezetés megírása

2023.03.11. 17:00	2 óra	Andó Deé-Lukács Kiss Skáre Vörös	Értekezlet, elvégzett feladatok közös átnézése.
2023.03.12. 18:00	1 óra	Skáre	Követelmények ellenőrzésének kidolgozása

## 3. Analízis modell kidolgozása

### 3.1 Objektum katalógus

#### 3.1.1 Pumpa

A pumpa egy olyan objektum a játékban, amire lépve a karakterek tudnak közlekedni a sivatagban. Egyszerre több karakter is tartózkodhat egy pumpán. A pumpába csövek vannak kötve, amiket a pumpa ismer/nyilvántart. minden pumpán ki van választva egy bemeneti és egy kimeneti cső. A pumpán a kimeneti és bemeneti cső változtatható. A pumpának ezen felül két állapota van: hibás és jól működő. Meghibásodott állapotban nem pumpálja át a vizet a bemeneti csőből a kimeneti csőbe. Rendelkezik egy átmeneti tárolóval, amibe a pumpa beszívja a vizet, és onnan engedi tovább.

#### 3.1.2 Cső

A cső egy olyan objektum a játékban, amire lépve a karakterek tudnak közlekedni a sivatagban. Egyszerre csak egy karakter tartózkodhat egy csövön.

Egy cső ismeri a két végén elhelyezkedő objektumokat. Ezek az alábbiak lehetnek: pumpa, ciszterna, forrás. A cső vizet képes magában elraktározni. Ez az objektum egy passzív elem, vagyis a víz mozgatására képtelen. Két állapota van: törött és ép. Törött állapotban képtelen vizet raktározni, a belepumpált víz kifolyik belőle (a homokba).

#### 3.1.3 Ciszterna

A ciszterna egy olyan objektum, amire a játékosok ráléphetnek. Egy ciszternán több karakter is tartózkodhat egyszerre.

A ciszterna vizet szív be a hozzá csatlakoztatott csövekből. A ciszterna ismeri/nyilvántartja a belé kötött csöveget. Ez az objektum felelős az új csövek, pumpák átadásáért.

#### 3.1.4 Forrás

A forrás egy olyan objektum, amire lépve a karakterek közlekedhetnek. A forrás a hozzá bekapcsolt csövekbe pumpálja a vizet. A hozzá bekapcsolt csöveget ismeri.

#### 3.1.5 Szerelő

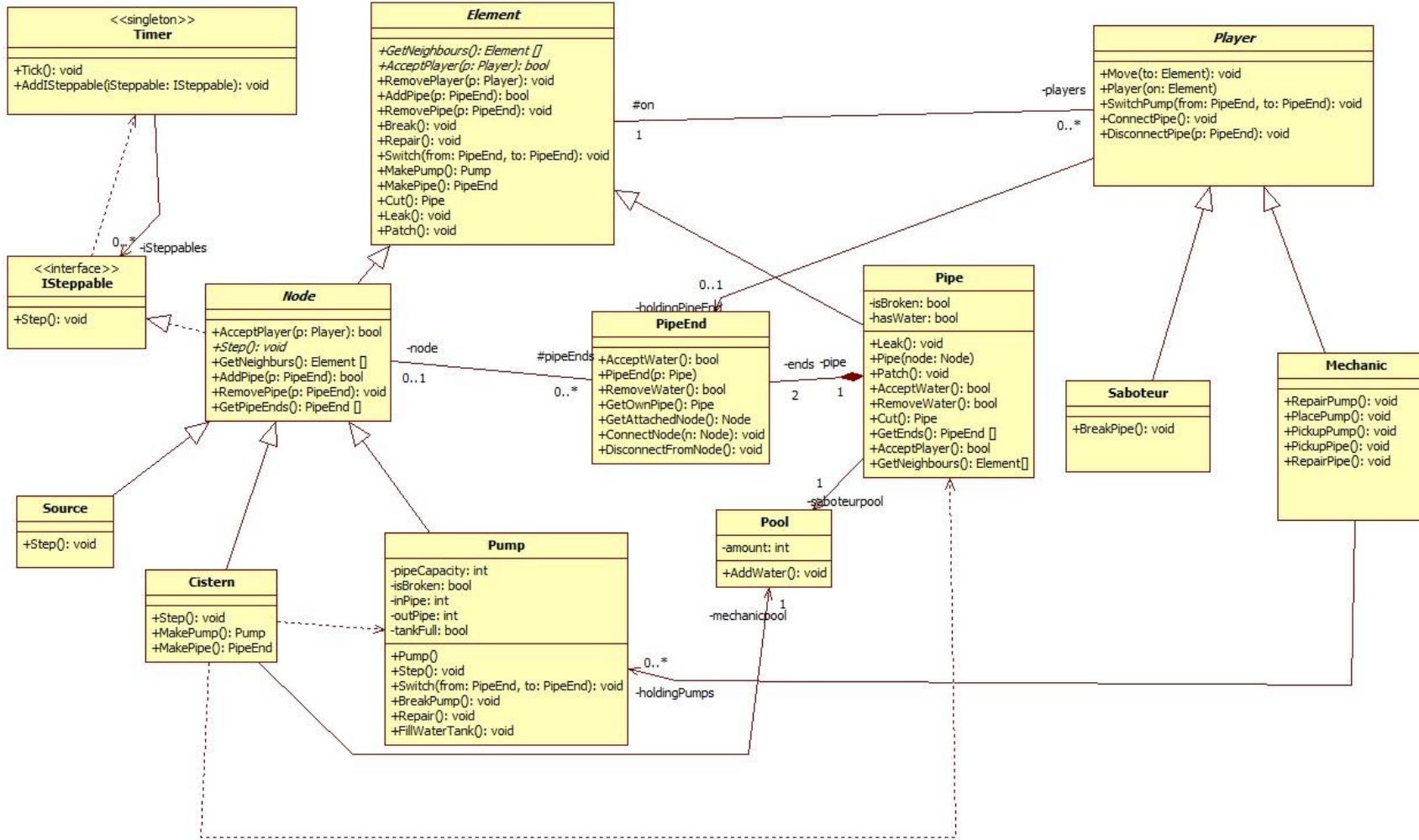
A szerelő karaktere egy olyan objektum, amit egy játékos irányít. Mozgása a ciszterna, forrás, cső, pumpa objektumokra korlátozott. Meg tudja foltozni a kilyukadt csöveget, át tudja állítani a pumpákon a vízáramlás irányát, fel tud

venni új csövet vagy pumpát a ciszteráknál, és be tudja azokat építeni a vízhálózatba.

### **3.1.6 Szabotőr**

A szabotőr karaktere egy olyan objektum, amit egy játékos irányít. Mozgása a ciszterna, forrás, cső, pumpa objektumokra korlátozott. Ki tudja lyukasztani a csöveket, át tudja állítani a pumpákon a vízáramlás irányát, és a meglévő csöveket tudja mozgatni.

### 3.2 Statikus struktúra diagramok



### 3.3 Osztályok leírása

#### 3.3.1 Cistern

- **Felelősség**

A játékban levő ciszternát reprezentálja. Folyamatosan elnyeli a vizet a vele összeköttetésben álló csövekből. Tárolja a befolyt víz mennyiségét. A ciszterna felelőssége a pumpák és a szabadvégű csövek biztosítása a játékosok számára.

- **Ősosztályok**

Element → Node

- **Attribútumok**

- **Pool mechanicPool:** eltárolja a ciszternába befolyt víz mennyiségét

- **Metódusok**

- **void Step():** lépteti a ciszternát
- **Pump MakePump():** létrehoz egy új pumpát és visszaadja
- **PipeEnd MakePipe():** létrehoz egy új csövet és visszaadja a szabad végét

#### 3.3.2 Element

- **Felelősség**

A játékban előforduló passzív és aktív elemeket reprezentálja, ezeken keresztül lehet mozogni. Ismeri a szomszédait, valamint tárolja a rajta levő játékosokat. Absztrakt osztály.

- **Attribútumok**

- **Player[] players:** az elemen álló játékosokat tárolja

- **Metódusok**

- **Element[] GetNeighbours():** visszaadja az adott Element szomszédait, absztrakt
- **bool AcceptPlayer(Player p):** igazzal tér vissza, ha a paraméterként megadott játékos ráléphet az adott Elementre, majd hozzáadja a rajta álló játékosokhoz, absztrakt
- **void RemovePlayer(Player p):** a paraméterként megadott játékos törli a rajta levő játékosok közül
- **bool AddPipe(PipeEnd p):** csatlakoztatja a megadott csővéget egy Node-hoz, üres
- **void RemovePipe(PipeEnd p):** lecsatlakoztatja a megadott csővéget egy Node-ról, üres
- **void Break():** elrontja az Elementet, üres
- **void Repair():** megjavítja az Elementet, üres
- **void Switch(PipeEnd from, PipeEnd to):** átállítja a be- és kimeneti csöveket, üres
- **Pump MakePump():** létrehoz egy új pumpát és visszaadja, üres
- **PipeEnd MakePipe():** létrehoz egy új csövet és visszaadja, üres
- **Pipe Cut():** elvágja a csövet, üres
- **void Leak():** kilyukasztja a csövet, üres
- **void Patch():** megfoltozza a csövet, üres

### 3.3.3 ISteppable

- **Felelősség**

Az időben léptethető dolgokat reprezentáló interfész.

- **Metódusok**

- **void Step():** lépteti az elemet

### 3.3.4 Node

- **Felelősség**

A játékban előforduló ‘csomópontokat’, aktív elemeket reprezentálja. Tárolja a vele összeköttetésben levő csővégeket (PipeEnd). Felelős azért, hogy számon tartsa az aktuális állapotát. Absztrakt osztály.

- **Ősosztályok**

Element

- **Interfészek**

ISteppable

- **Attribútumok**

- **PipeEnd[] pipeEnds:** az adott Node-dal összekötött csővégek

- **Metódusok**

- **Element[] GetNeighbours():** visszaadja az adott Node szomszédait
- **bool AcceptPlayer(Player p):** igazzal tér vissza, ha a paraméterként megadott játékos ráléphet az adott Elementre, majd hozzáadja a rajta álló játékosokhoz
- **bool AddPipe(PipeEnd p):** igazzal tér vissza, ha a paraméterként megadott csővéget lehet csatlakoztatni az adott Node-hoz, és eltárolja a vele összekötött csővégek közé
- **void RemovePipe(PipeEnd p):** a paraméterként megadott csővéget törli a hozzá kötött csővégek közül
- **void Step():** lépteti az adott Nodeot, absztrakt

### 3.3.5 Mechanic

- **Felelősség**

A játékban a szerelő karaktereket reprezentálja. Képes megjavítani az elromlott pumpákat és a lyukas csöveket, új csövet / pumpákat magához venni és letenni.

- **Ősosztályok**

Player

- **Attribútumok**

- **Pump[] holdingPumps:** eltárolja a szerelőnél levő pumpákat

- **Metódusok**

- **void RepairPump():** megjavítja azt a pumpát, amin a szerelő áll

- **void PlacePump()**: elhelyez egy új pumpát úgy, hogy lecsatlakoztatja azt a csövet, amin áll, létrehoz egy újat (mintha kettévágna), és a pumpához csatlakoztatja őket, és az új csövet csatlakoztatja az eredeti cső régi kimenetére
- **void PickUpPump()**: felvesz egy új pumpát
- **void PickUpPipe()**: felvesz egy új csövet, és a végét eltárolja a holdingPipeEnd-be
- **void RepairPipe()**: megfoltozza azt a csövet, amin a játékos áll

### 3.3.6 Pipe

- **Felelősség**

A játékban lévő csöveket reprezentálja, vizet tárol (1 egységnyit). Tárolja a végeit, számon tartja, hogy mennyi víz folyt ki belőle, illetve az aktuális állapotát, tehát hogy lyukas-e, valamint hogy van-e benne víz.

- **Ósosztályok**

Element

- **Attribútumok**

- **bool hasWater**: igaz, ha a tárolóban van víz
- **bool isBroken**: igaz, ha a cső lyukas
- **Pool saboteurPool**: eltárolja a lyukas csőből kifolyt víz mennyiségét
- **PipeEnd ends[]**: eltárolja a csőhöz tartozó végeket

- **Metódusok**

- **Pipe(Node node)**: paraméteres konstruktur
- **bool AcceptPlayer(Player p)**: a saját működésének megfelelően felüldefiniálja az Element AcceptPlayer függvényét, vagyis ellenőrzi, hogy áll-e rajta valaki, és ha nem, akkor igazzal tér vissza, és a paraméterként megadott játékos eltárolja a rajta levő játékosok közé
- **void AcceptWater()**: igazra állítja a hasWater attribútumot
- **bool RemoveWater()**: hamisra állítja a hasWater attribútumot, igazzal tér vissza, ha volt mit eltávolítani
- **void Leak()**: igazra állítja az isBroken attribútumot
- **void Patch()**: hamisra állítja az isBroken attribútumot
- **Pipe Cut()**: elvágja a csövet, és visszaadja az új csövet (a cső másik felét)
- **Element[] GetNeighbours()**: visszaadja az adott Pipe szomszédait
- **PipeEnd[] GetEnds()**: visszaadja a cső végeit

### 3.3.7 PipeEnd

- **Felelősség**

A játékban levő csővégeket reprezentálja. Lehet hozzá Node-ot kötni illetve lecsatlakoztatni róla. Tárolja, hogy milyen Node-dal áll összeköttetésben, és hogy melyik csőhöz tartozik.

- **Attribútumok**

- **Node node**: eltárolja a csővéggel összeköttetésben levő Node-ot

- **Metódusok**

- **void AcceptWater()**: továbbítja a vizet a csőnek

- **bool RemoveWater()**: eltávolítja a vizet a csőből, igazzal tér vissza, ha volt mit eltávolítani
- **Pipe GetOwnPipe()**: visszaadja a csövet, amihez tartozik a csővég
- **Node GetAttachedNode()**: visszaadja, melyik Node-hoz van hozzákapcsolva
- **Pipe GetOwnPipe()**: visszaadja, melyik csőnek a vége az adott példány
- **void ConnectNode(Node n)**: a paraméterként megadott Node-hoz csatlakoztatja az adott csővéget

### 3.3.8 Player

- **Felelősség**

A játékosok által irányítható karaktereket reprezentálja. Eltárolja, hogy melyik Element-en áll, emellett képes átmenni másik Element-re. Eltárolja, hogy van-e nála cső, és tudja a csöveket csatlakoztatni más elemekhez. Absztrakt osztály.

- **Attribútumok**

- **PipeEnd holdingPipeEnd**: eltárolja a játékosnál levő csővéget
- **Element on**: eltárolja, hogy melyik Element-en tartózkodik a játékos

- **Metódusok**

- **void Move(Element to)**: átmozgatja az adott Playert a paraméterben megadott Elementre, amennyiben sikerül, eltárolja az új helyzetének
- **void SwitchPump(PipeEnd from, PipeEnd to)**: átállítja annak a pumpának a bemeneti és kimeneti csövét, amin a karakter áll, a megadott paraméterek alapján
- **void ConnectPipe()**: csatlakoztatja a holding attribútumban levő csövet ahhoz az Elementhez, amin áll
- **void DisconnectPipe(PipeEnd pend)**: a megadott paraméterű csövet lecsatlakoztatja, és hozzáadja a holding-hoz

### 3.3.9 Pool

- **Felelősség**

Számon tartja a belefolyt víz mennyiségét.

- **Attribútumok**

- **int amount**: az adott Pool-ba folyt víz mennyisége

- **Metódusok**

- **void AddWater()**: növeli az amount értékét 1-gyel

### 3.3.10 Pump

- **Felelősség**

A játékban előforduló pumpákat reprezentálja. Eltárolja a kimeneti és bemeneti csövét. Számon tartja a maximálisan beleköthető csövek számát és az aktuális állapotát, tehát hogy el van-e romolva. Az aktuálisan beállított bemeneti csőből pumpálja a vizet a kimeneti csövébe. Emellett át lehet állítani a ki- és bemeneti csöveit, valamint lehet hozzá csövet csatlakoztatni.

- **Ősosztályok**

Element → Node

- **Attribútumok**

- **bool isBroken:** igaz, amennyiben a pumpa el van romolva
- **int pipeCapacity:** statikus, a beleköthető csövek maximális száma
- **Pipe inPipe:** az aktuális bemeneti cső
- **Pipe outPipe:** az aktuális kimeneti cső

- **Metódusok**

- **void Break():** a pumpa állapotát elromlottra állítja
- **void Repair():** a pumpa állapotát visszaállítja működőre
- **void Switch(PipeEnd from, PipeEnd to):** megváltoztatja a bementi és kimeneti csövet a paraméterként megadott azonosítók alapján
- **void Step():** lépteti a pumpát

### 3.3.11 Saboteur

- **Felelősség**

A játékban a szabotör karaktert reprezentálja. Képes kilyukasztani a csöveget.

- **Ősosztályok**

Player

- **Metódusok**

- **void BreakPipe():** kilyukasztja a csövet, amin a játékos áll

### 3.3.12 Source

- **Felelősség**

A játékban levő forrást reprezentálja. Folyamatosan vizet szolgáltat a vele összeköttetésben álló csövek számára.

- **Ősosztályok**

Element → Node

- **Metódusok**

- **void Step():** lépteti a Source-ot

### 3.3.13 Timer

- **Felelősség**

Periodikus időt reprezentál, az időben léptethető dolgokat lépteti, singleton.

- **Attribútumok**

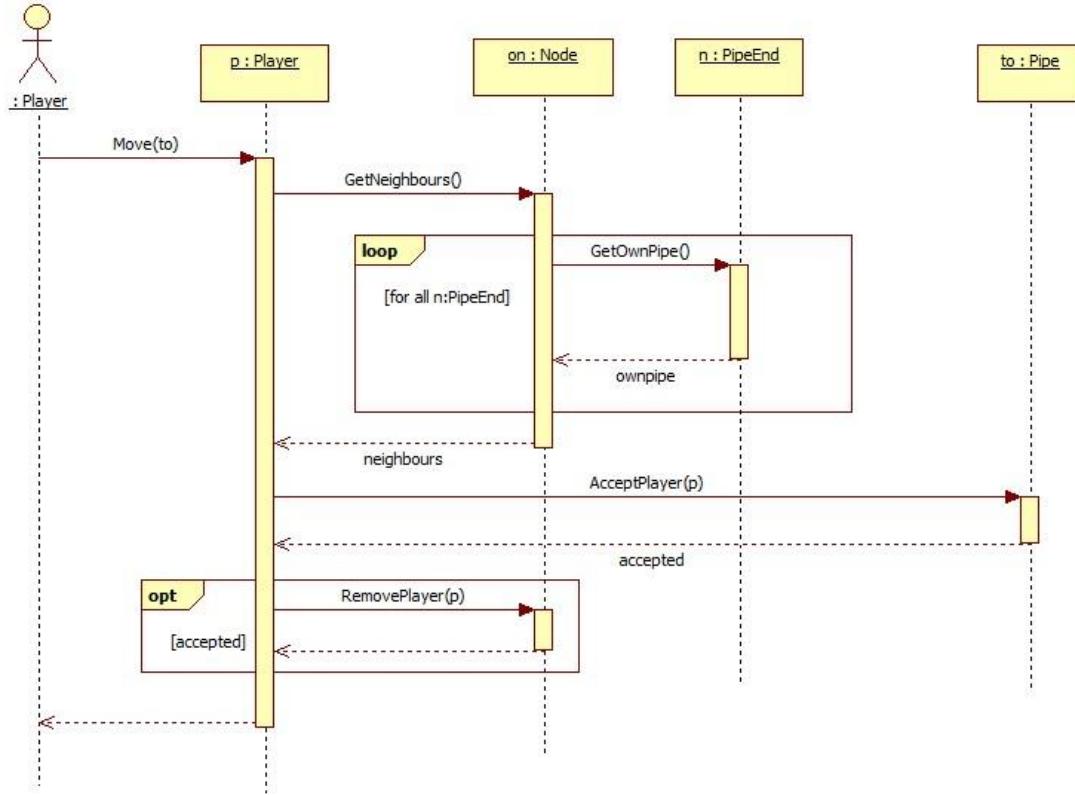
- **ISteppable[] iSteppables:** Az időben léptethető elemeket tárolja.

- **Metódusok**

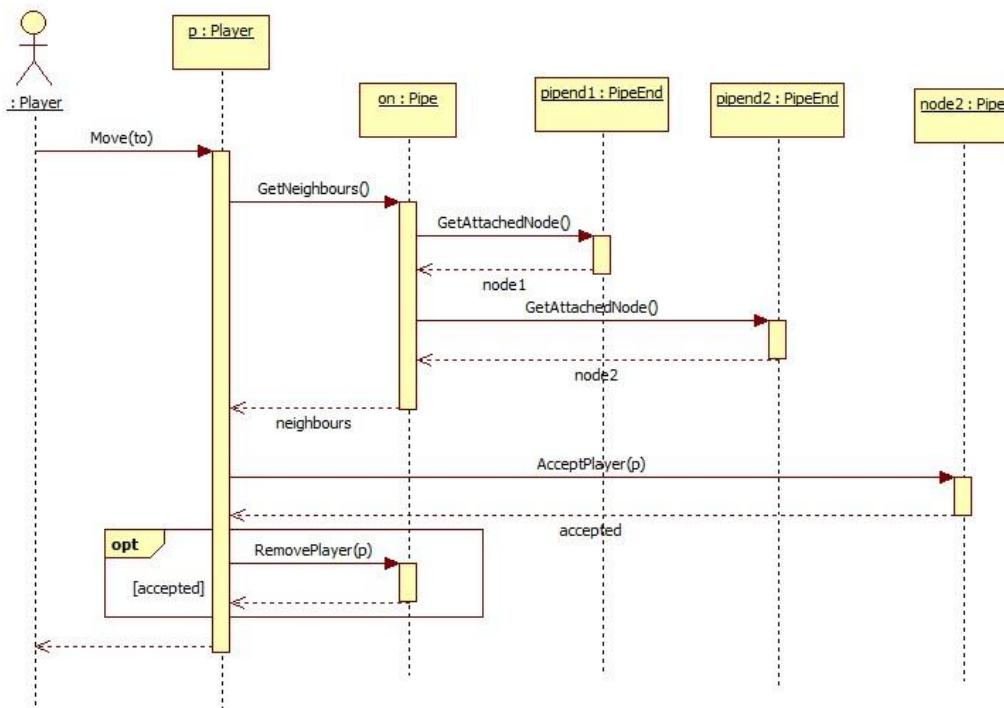
- **void AddISteppable(ISteppable iStappable):** hozzáadja a paraméterben megadott ISteppable-t a steppables-höz

## 3.4 Szekvencia diagramok

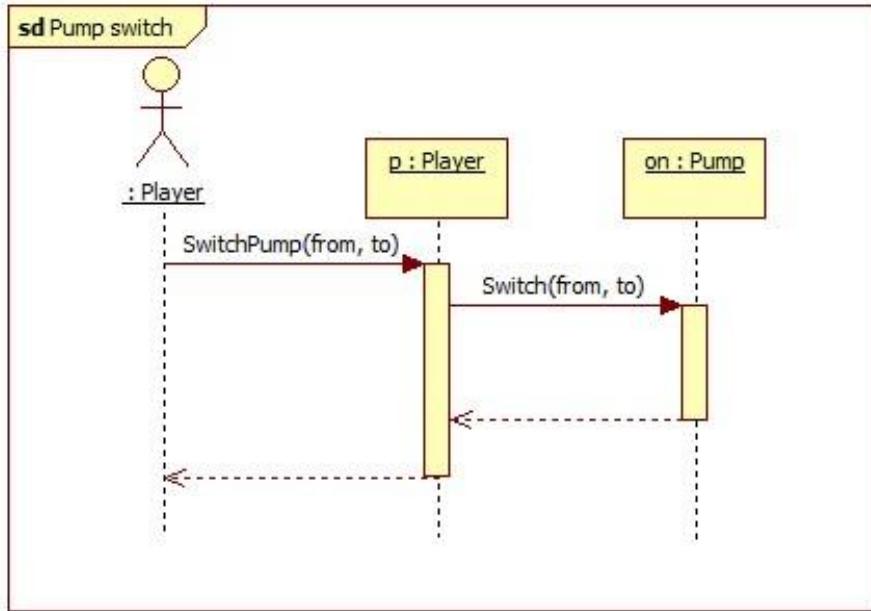
### 3.4.1 Player Moves from Node



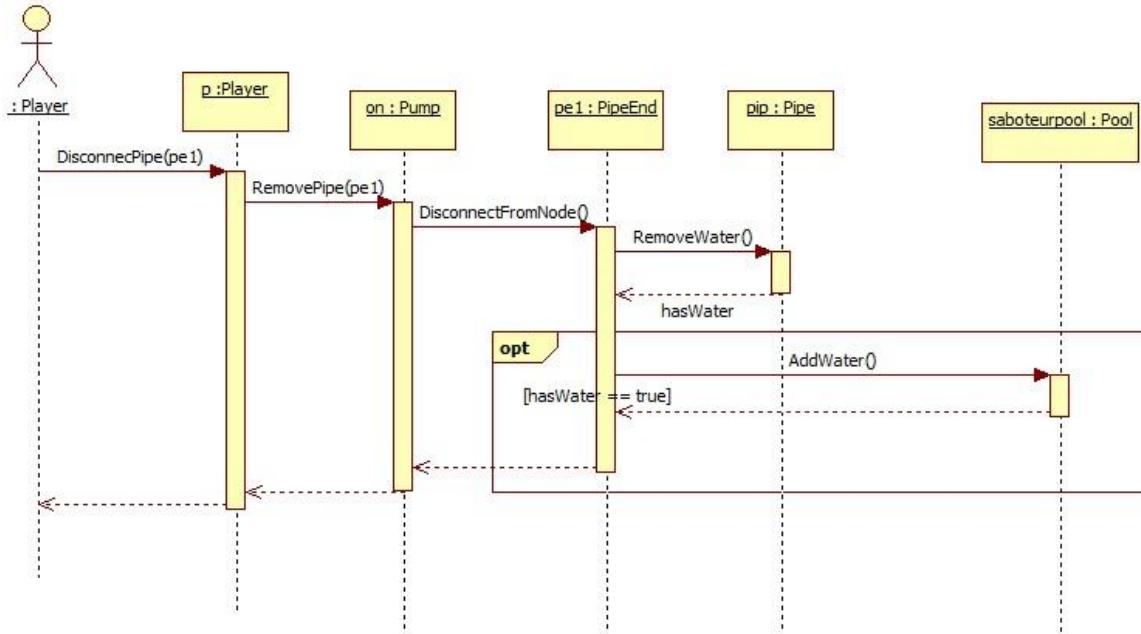
### 3.4.2 Player Moves from Pipe



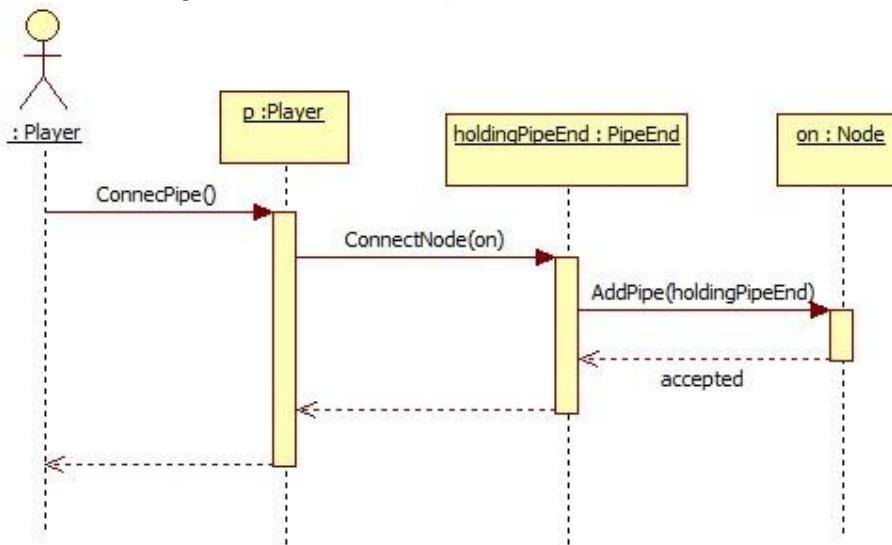
### 3.4.3 Player Switches Pump



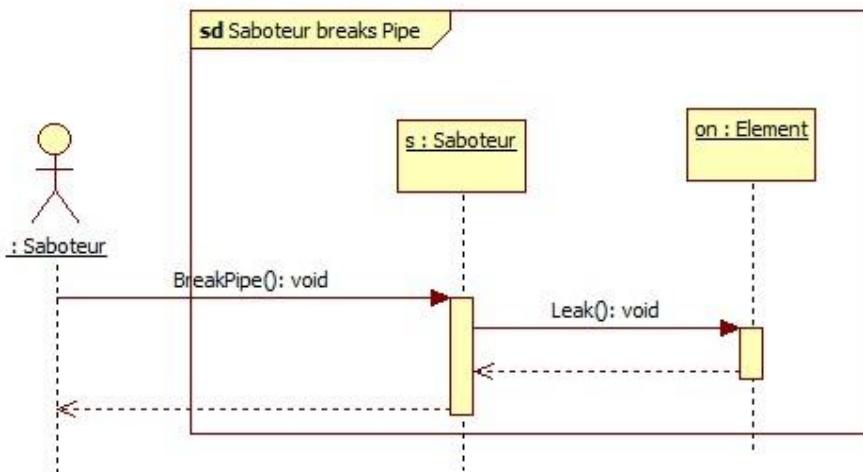
### 3.4.4 Player Disconnects Pipe



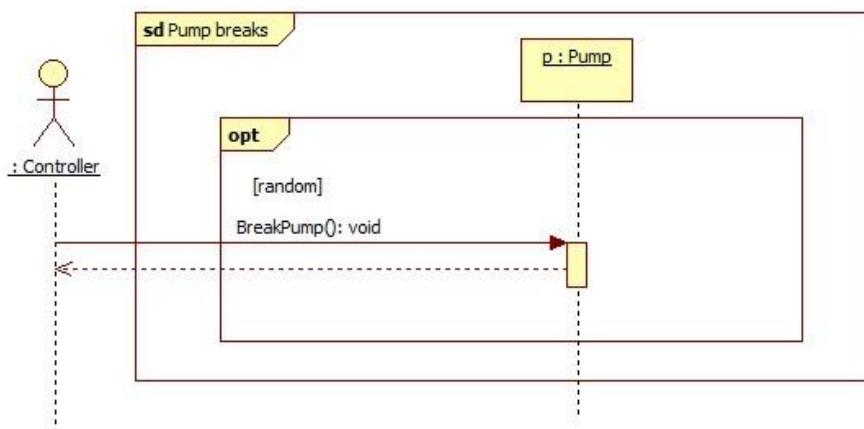
### 3.4.5 Player Connects Pipe



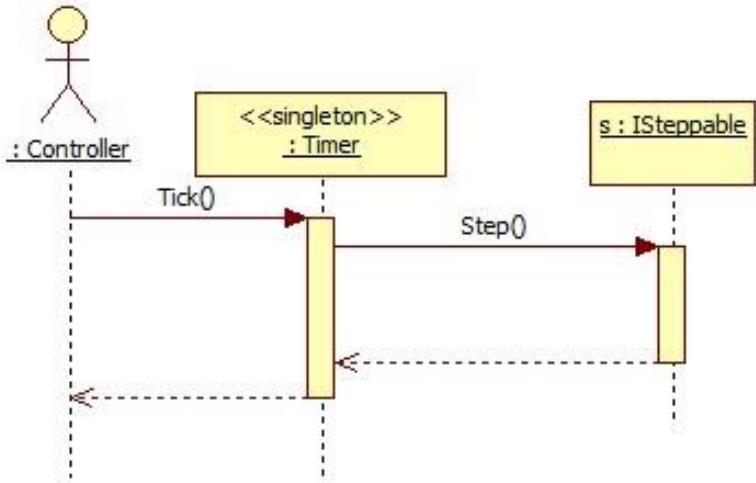
### 3.4.6 Saboteur Breaks Pipe



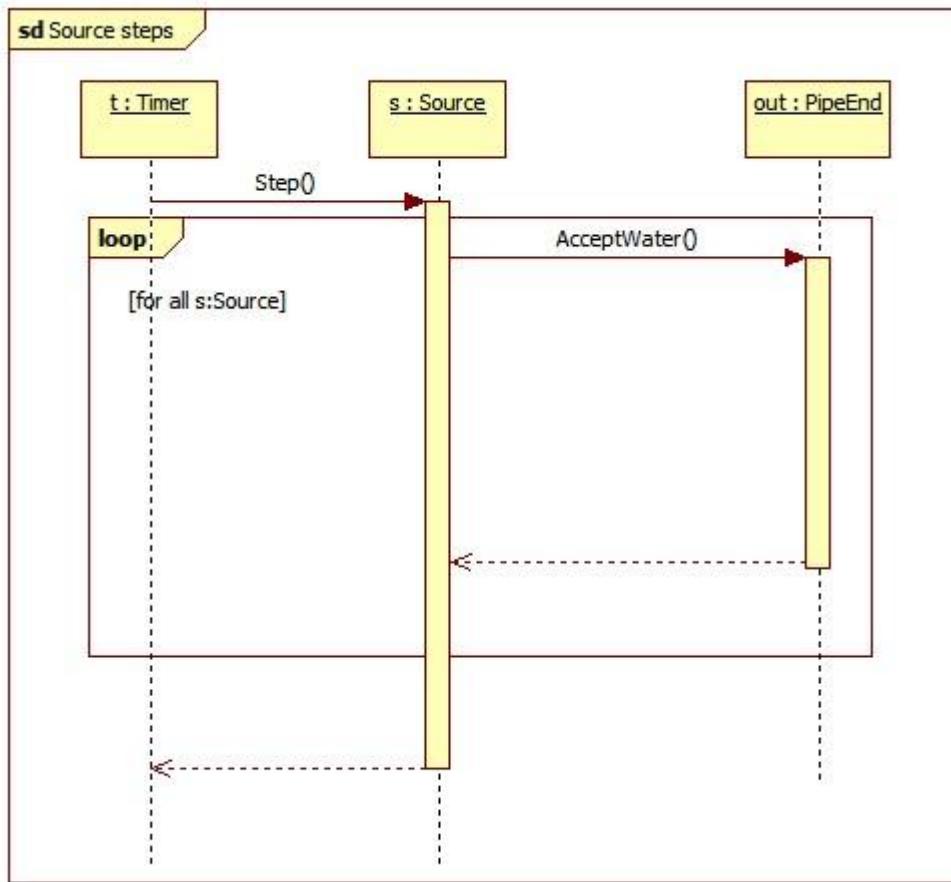
### 3.4.7 Pump Breaks



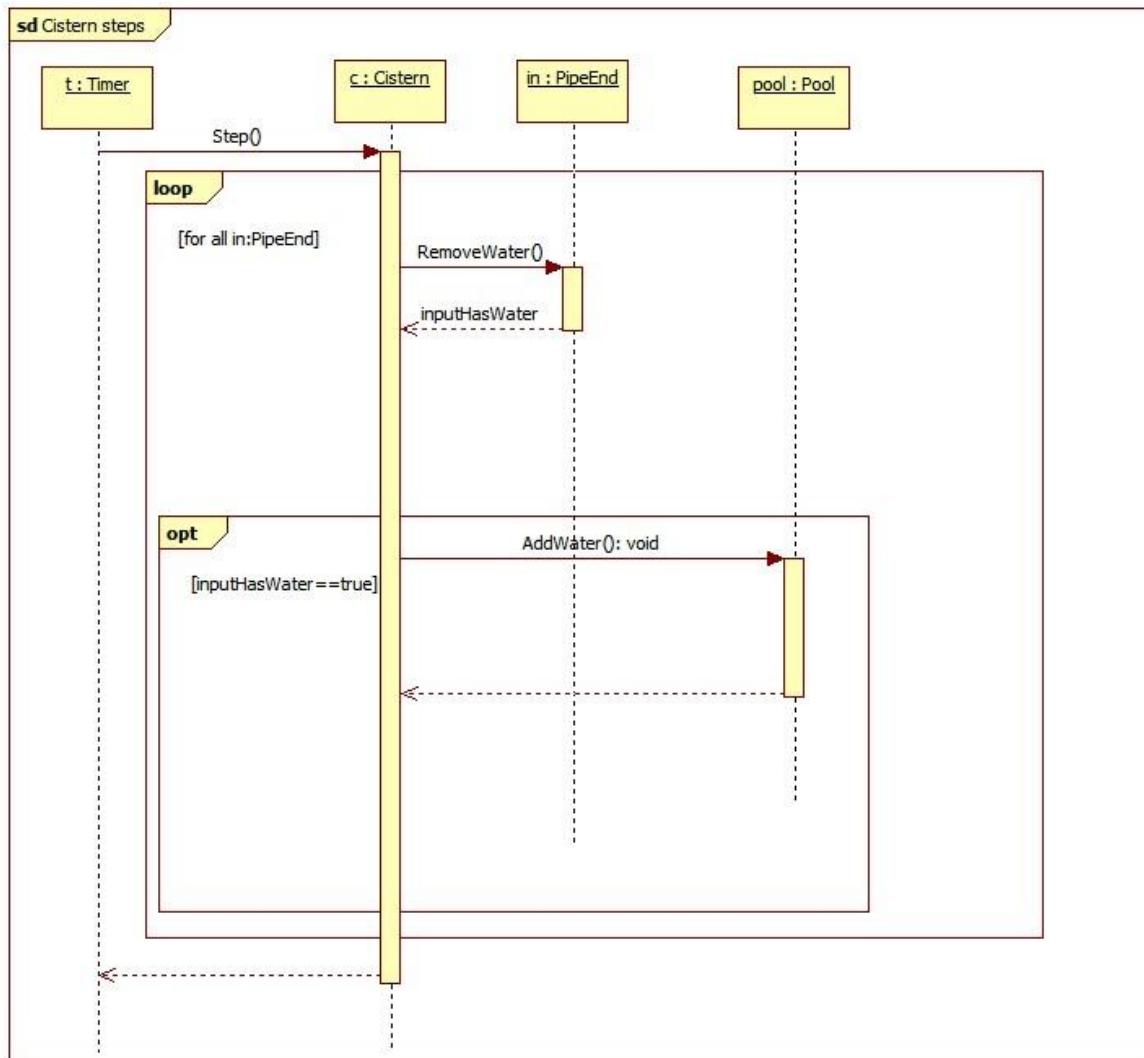
### 3.4.8 ISteppable steps



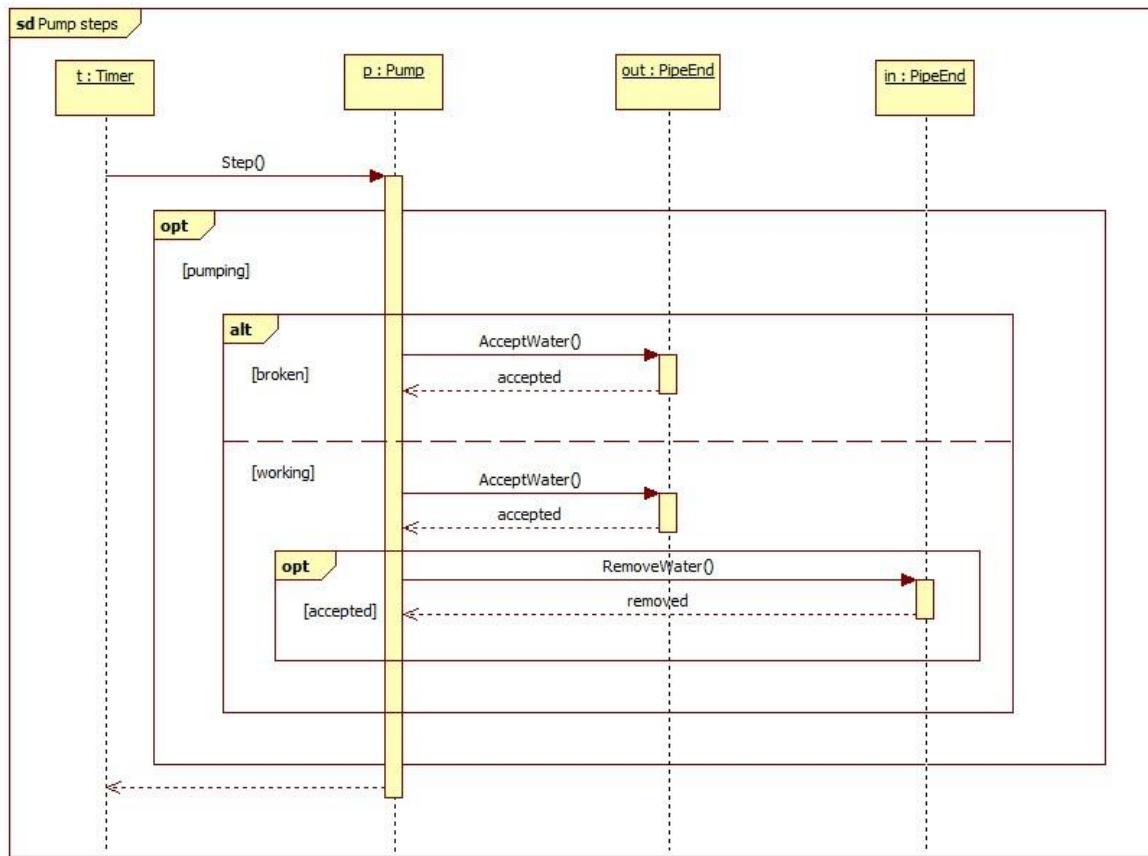
### 3.4.9 Source steps



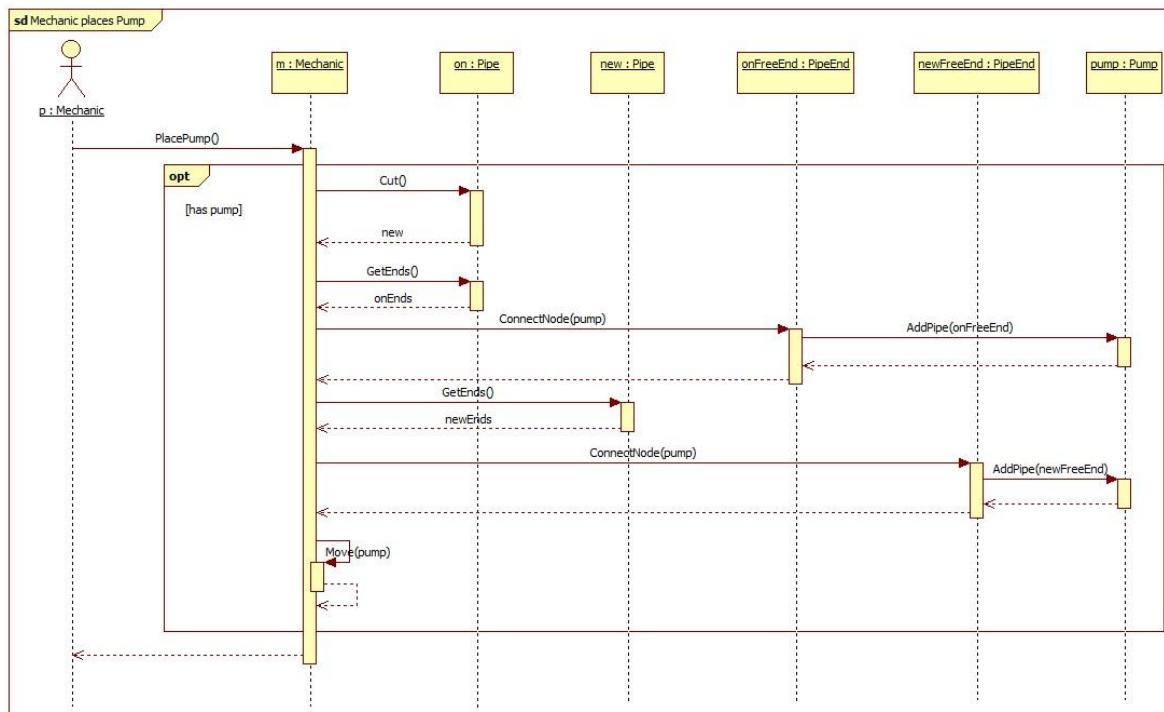
### 3.4.10 Cistern steps



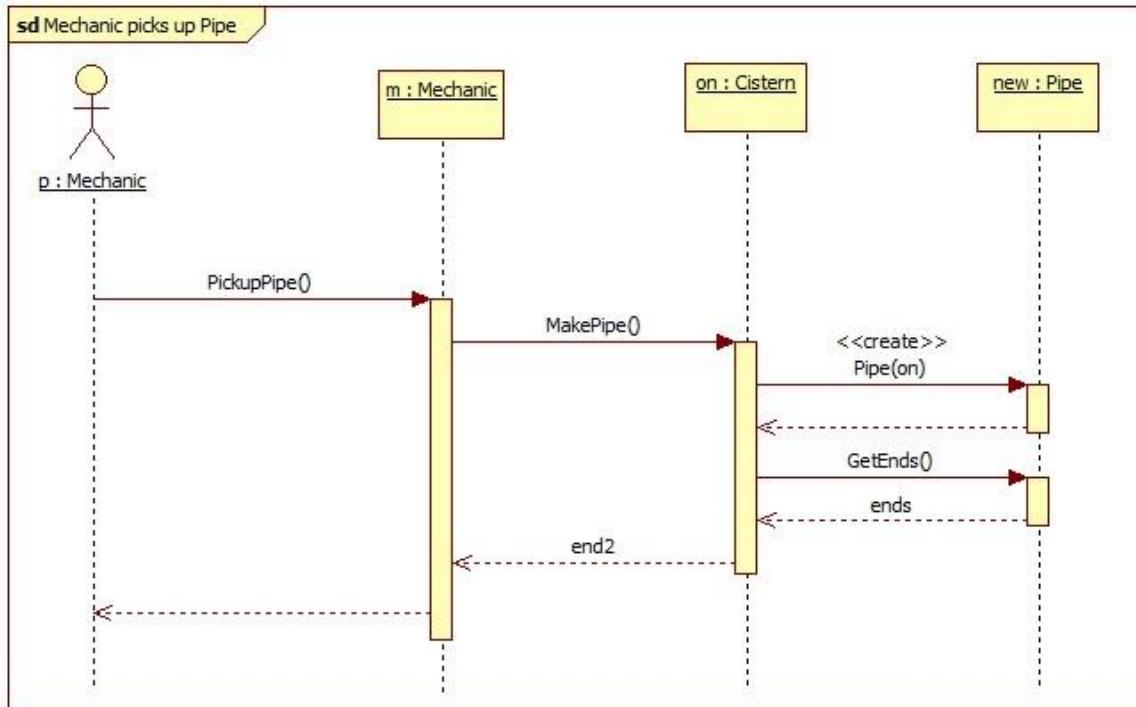
### 3.4.11 Pump steps



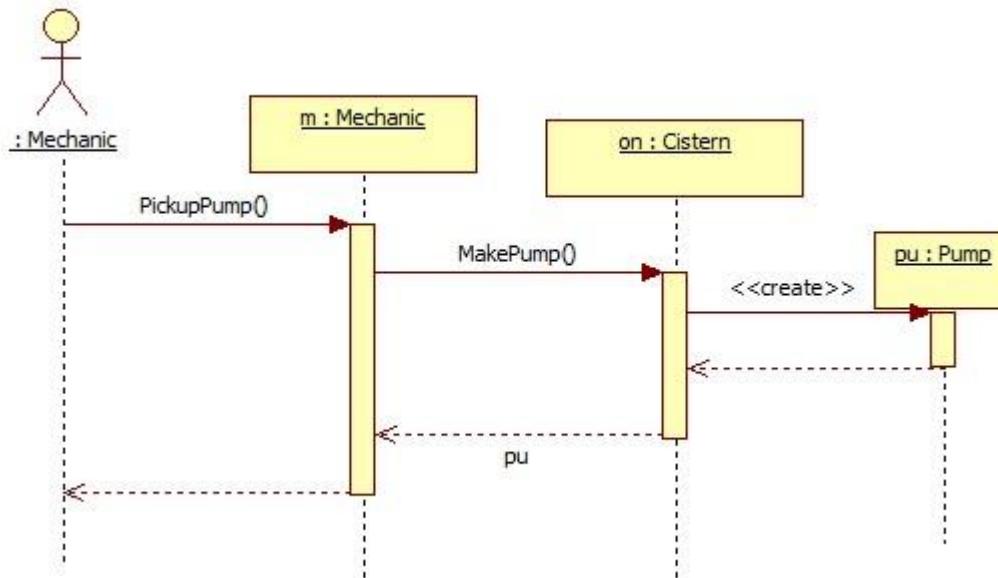
### 3.4.12 Mechanic places Pump



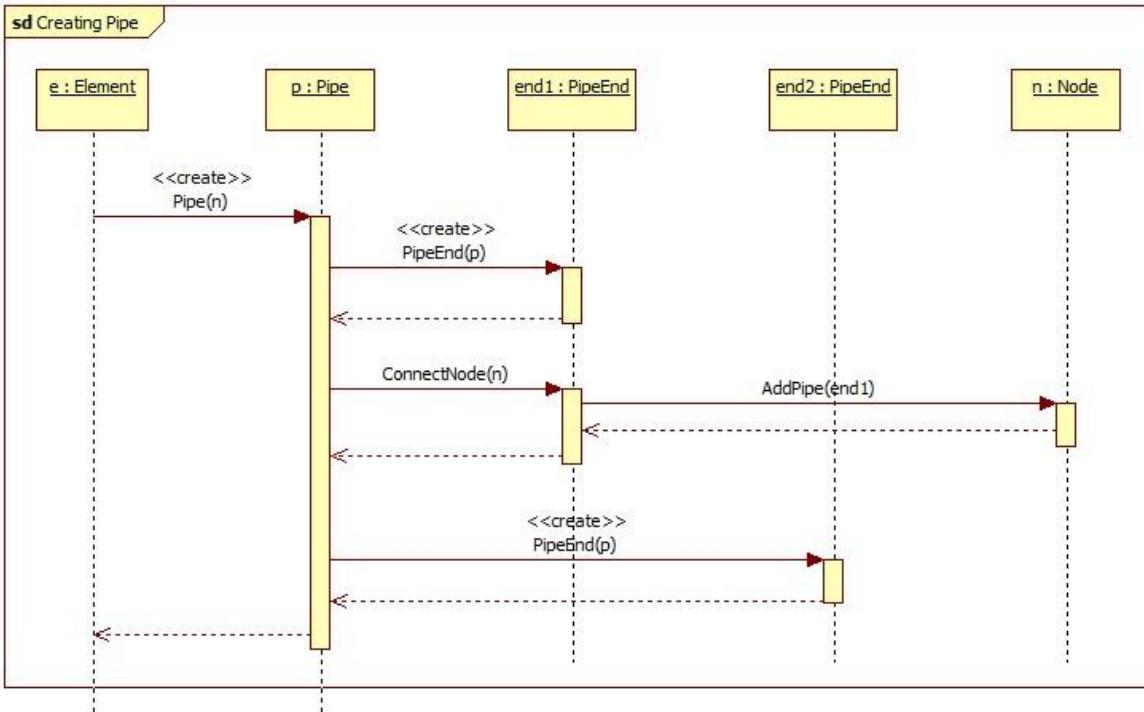
### 3.4.13 Mechanic picks up Pipe



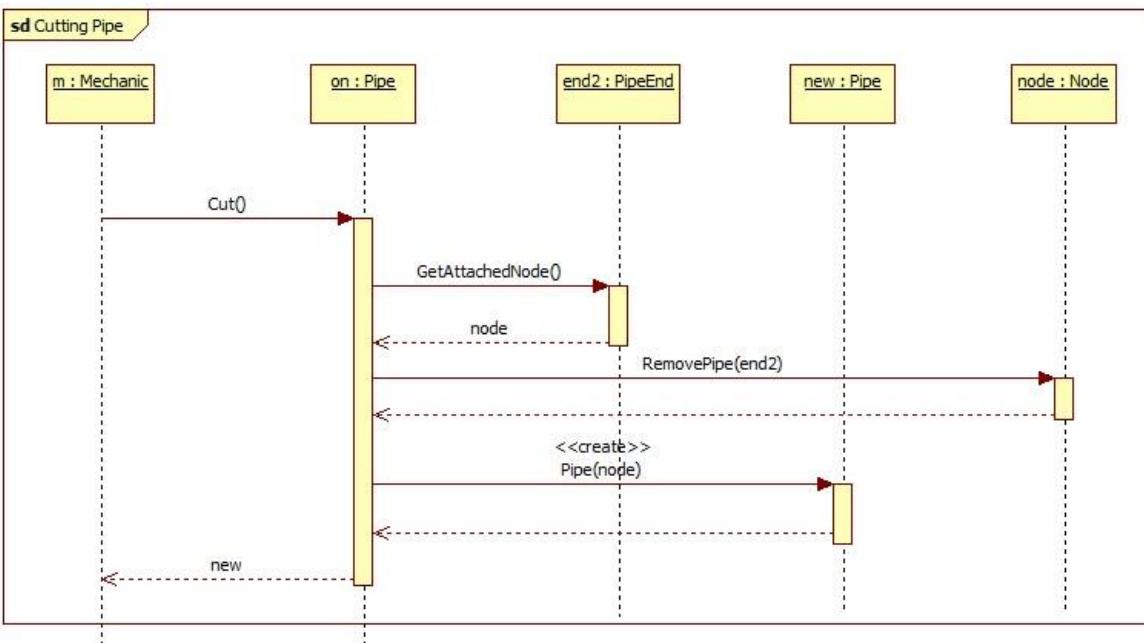
### 3.4.14 Mechanic picks up new Pump



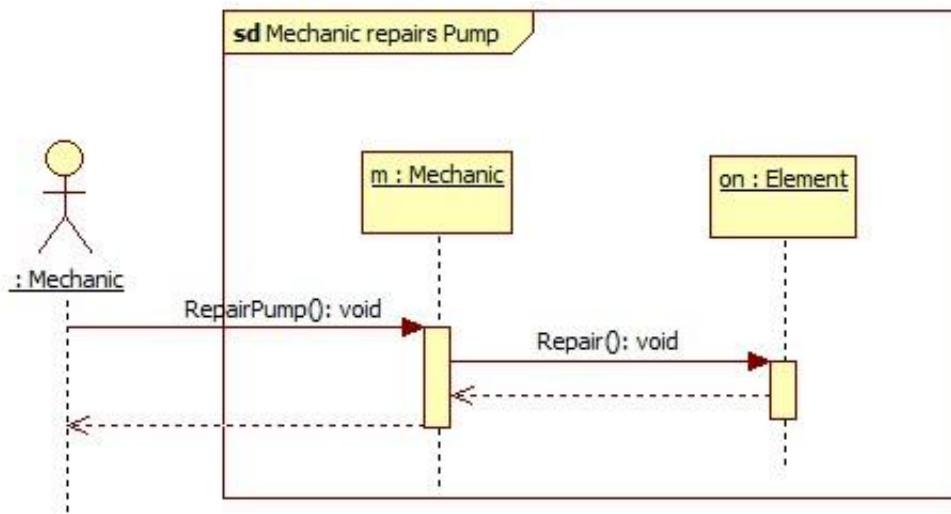
### 3.4.15 Creating Pipe



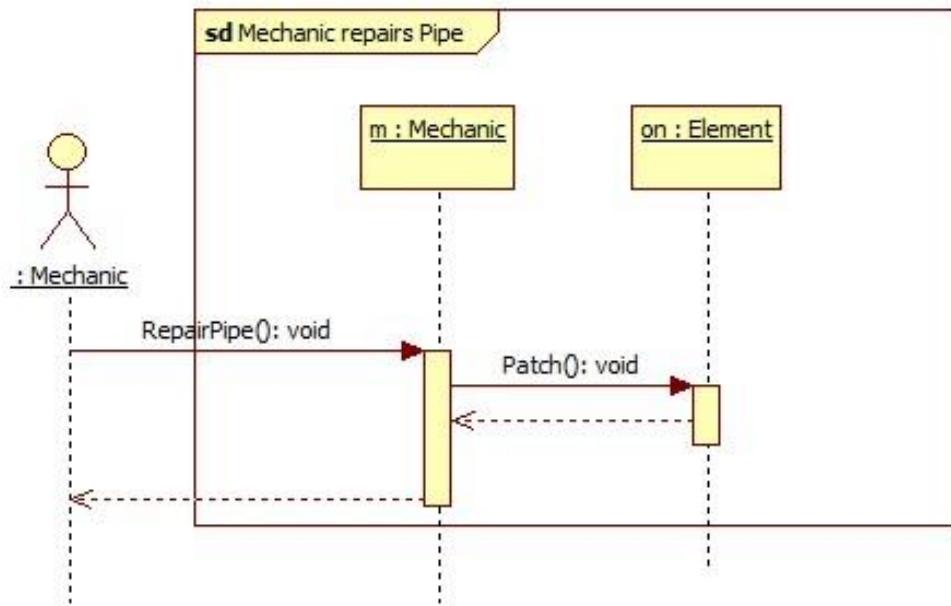
### 3.4.16 Cutting Pipe



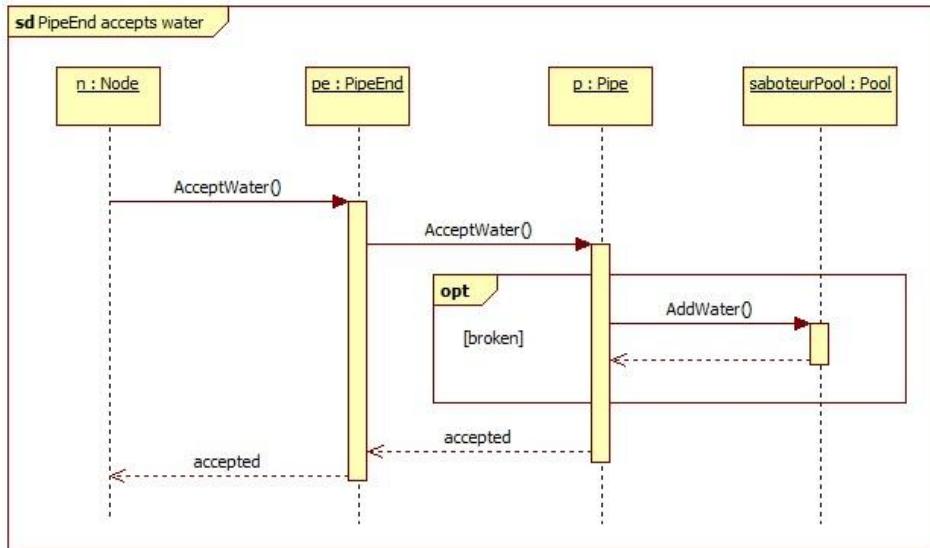
### 3.4.17 Mechanic repairs pump



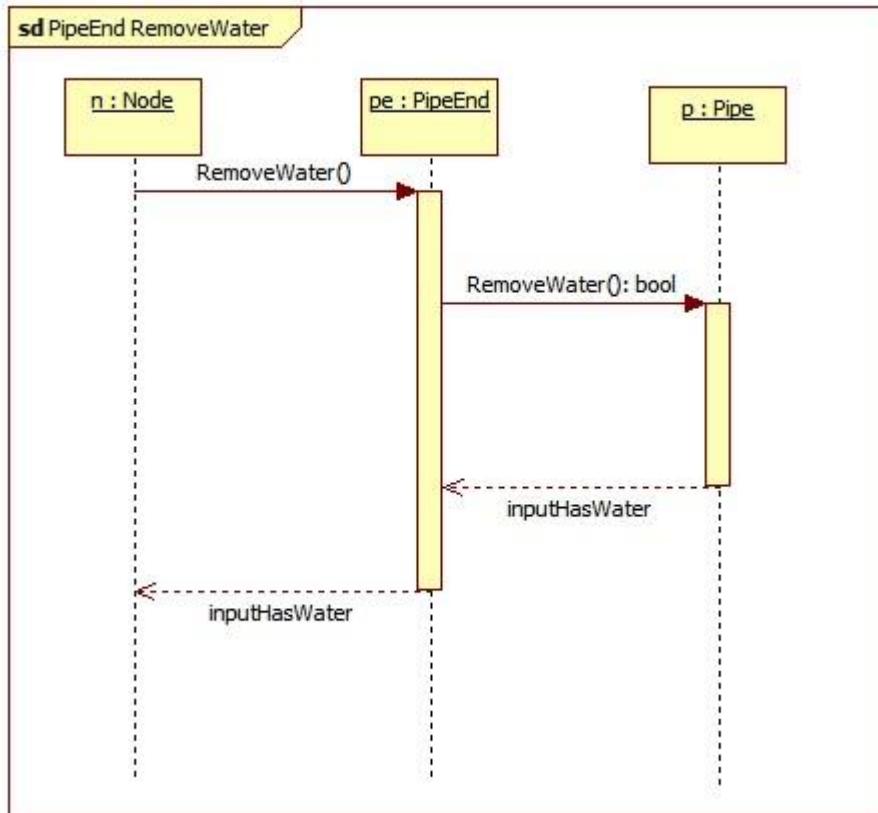
### 3.4.18 Mechanic repairs pipe



### 3.4.19 PipeEnd accepts water

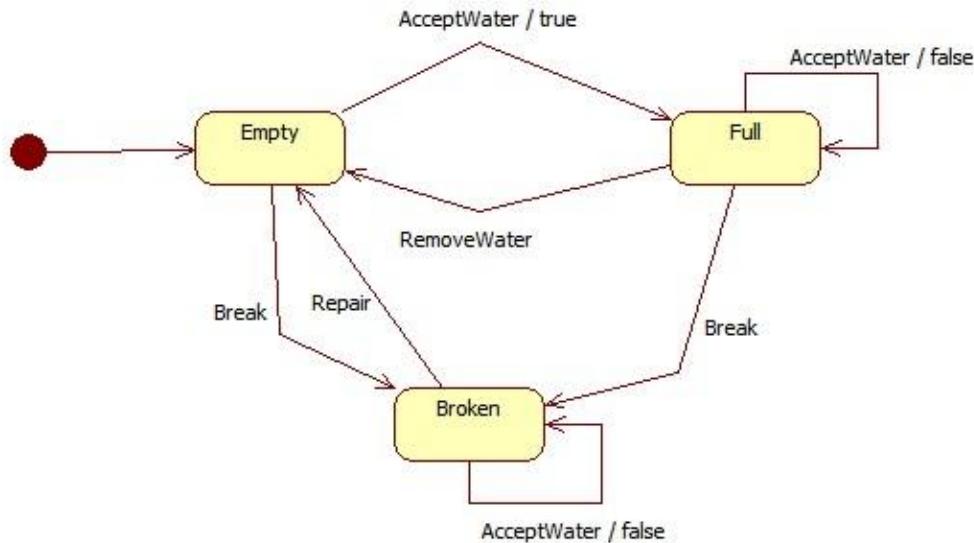


### 3.4.20 Water removal from PipeEnd

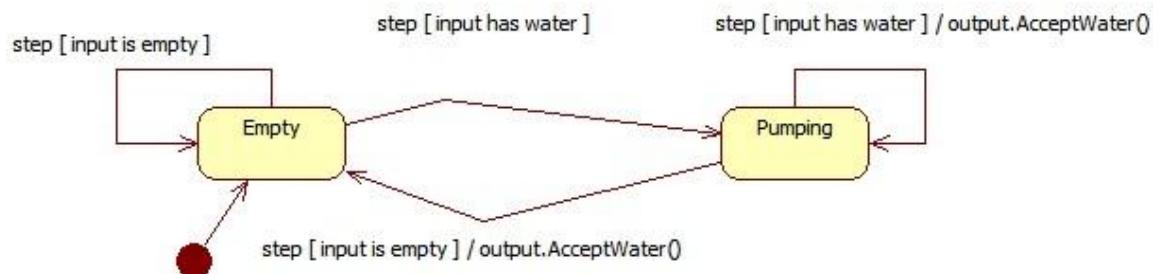


### 3.5 State-chartok

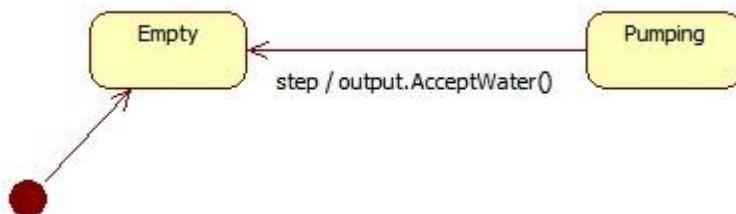
#### 3.5.1. Pipe behaviour



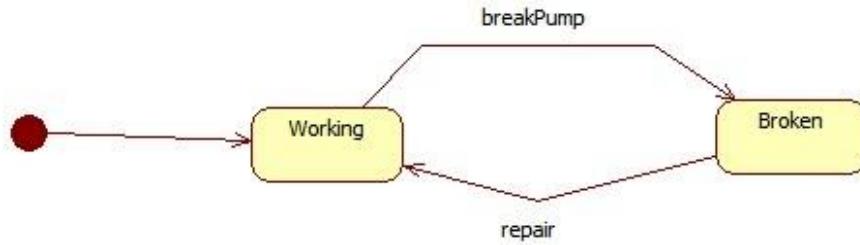
#### 3.5.2. Pump working



#### 3.5.3 Broken pump



### 3.5.4 Break pump



## 3.6 Napló

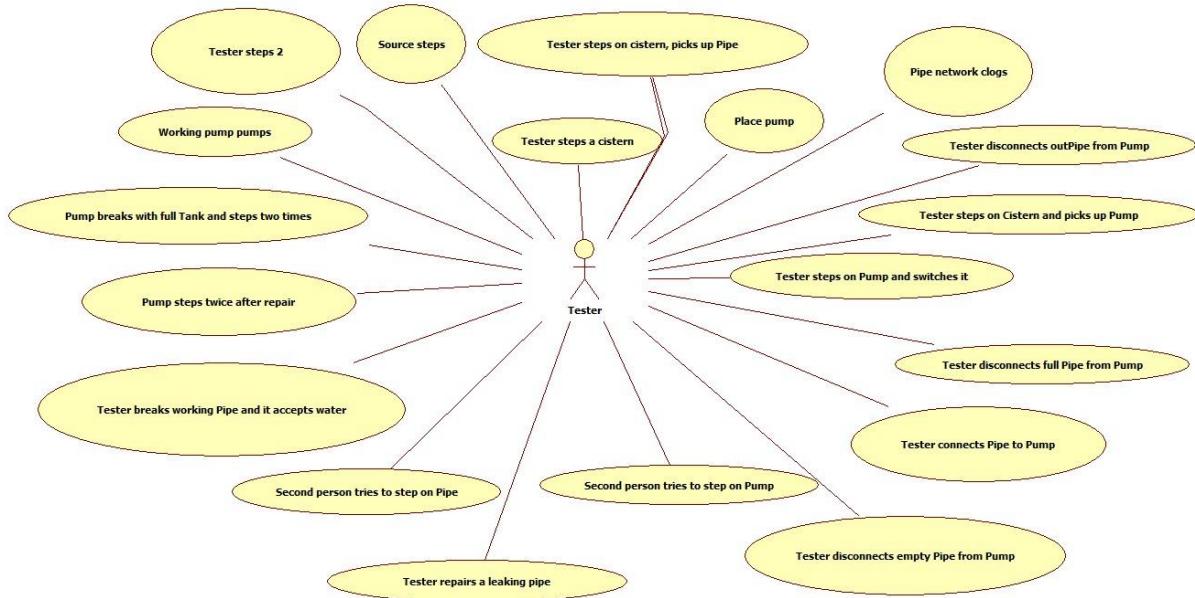
Kezdet	Időtartam	Résznevők	Leírás
2023.03.23. 16.00	2 óra	Andó Deé-Lukács Kiss Skáre Vörös	Értekezlet: Feladatok Osztálydiagram újratervezés Döntések: Andó megcsinálja a 3.4.6, 3.4.7, 3.4.17, 3.4.18 szekvencia diagramokat Deé-Lukács újrarajzolja pumpa állapotdiagramjait, Kiss újraírja az osztályleírásokat, Skáre megcsinálja a 3.4.12-16 szekvencia diagramokat Vörös szekvenciadiagramo kat készít (1-5, 8)
2023.03.24. 12.00	2 óra	Kiss	Osztályleírások módosítása
2023.03.24. 15:00	2 óra	Andó	3.4.6, 3.4.7, 3.4.17, 3.4.18 szekvencia diagramok megrajzolása
2023.03.24. 16:00	1 óra	Vörös	Szekv.diagram: 3.4.1-5, 3.4.8
2023.03.24. 17.00	2 óra	Deé-Lukács	Állapotdiagramok elkészítése
2023.03.25. 11.00	2 óra	Deé-Lukács	Szekvencia diagramok 3.4.9-11,19-20

2023.03.26. 10:00	2 óra	Vörös	Szekvencia folyt. 3.4.1-5, 3.4.8
2023.03.26. 13.00	1 óra	Skáre	Szekvencia diagramok 3.4.12-16
2023.03.26. 20.00	1 óra	Skáre	Szekvencia diagramok 3.4.12-16
2023.03.26. 21.00	2 óra	Andó Deé-Lukács Kiss Skáre Vörös	Beadandó dokumentum áttekintése, ellenőrzése és véglegesítése

# 5. Szkeleton tervezése

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

### 5.1.1 Use-case diagram



### 5.1.2 Use-case leírások

Minden use-case-re / tesztesetre igaz, hogy a tesztelést végző személy (a továbbiakban: tesztelő) a kimeneten megjelenő függvényhívások és visszatérési értékek által ellenőrzi az elvárt működést.

<b>Use-case neve</b>	Tester patches a leaking pipe
<b>Rövid leírás</b>	A tesztelő leteszeli, hogy egy megfoltozott cső tudja -e tárolni a vizet, majd a szomszédos pumpa ki tudja-e szívni belőle a vizet
<b>Aktorok</b>	Tester

<b>Forgatókönyv</b>	1. A tesztelő megjavítja, majd feltölti vízzel a csövet, végül lépteti a csőhöz csatlakoztatott pumpát. A függvényhívások eredményeként a cső elfogadja a vizet, amit ezután a pumpa sikeresen kiszív.
---------------------	--

<b>Use-case neve</b>	Tester steps on pump and switches it
<b>Rövid leírás</b>	A tesztelő leteszeli, hogy tud-e mozogni egy pumpára, majd át tudja-e úgy állítani, hogy a megadott csövek között pumpáljon
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	1. A tesztelő a szomszédos pumpára lép, majd átállítja azt. Végül lépteti a pumpát, majd ellenőrzi, hogy a pumpa valóban a két megfigyelt cső között pumpál-e.

<b>Use-case neve</b>	Tester disconnects a full pipe
<b>Rövid leírás</b>	A tesztelő leteszeli, hogy ha lecsatlakoztat egy vízzel teli csövet, akkor abból kifolyik-e a víz.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	1. A tesztelő lecsatlakoztat egy pumpáról egy olyan csövet, ami tele van vízzel, és ellenőrzi, a víz valóban kifolyik-e a csőből.

<b>Use-case neve</b>	Tester steps a cistern
<b>Rövid leírás</b>	A tesztelő leteszeli, hogy egy két csővel rendelkező ciszterna megfelelően működik-e
<b>Aktorok</b>	Tester

<b>Forgatókönyv</b>	1. A tesztelő lépet egy ciszternát, ami kiszívja a hozzá csatlakoztatott csövekből a vizet, majd a kapott vizet továbbítja a szerelők csapatához tartozó vízhez.
---------------------	--

<b>Use-case neve</b>	Tester steps on Cistern and picks up Pump
<b>Rövid leírás</b>	A tesztelő rálép egy ciszternára, majd felvesz egy pumpát, és ellenőrzi, hogy ez sikerült-e.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	1. A tesztelő rálép egy ciszternára, majd felvesz egy pumpát.

<b>Use-case neve</b>	Second Person tries to step on Pump
<b>Rövid leírás</b>	A tesztelő leteszeli, hogy ha valaki áll már egy pumpán, akkor rá tud-e lépni.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	1. A tesztelő rálép a pumpára, amin már van egy másik játékos, és ez sikerül.

<b>Use-case neve</b>	Pump breaks with full tank and steps two times
<b>Rövid leírás</b>	A pumpa elromlik, majd kétszer lép.
<b>Aktorok</b>	Controller
<b>Forgatókönyv</b>	1. Elromlik a pumpa, a víztartályból kifolyik a víz az üres, nem lyukas kimeneti csőbe, utána nem pumpál többet.

<b>Use-case neve</b>	Tester steps on Cistern and picks up Pipe
----------------------	---

<b>Rövid leírás</b>	A tesztelő rálép egy ciszternára, ahol felvesz egy új csövet.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	A tesztelő rálép egy ciszternára, majd felvesz egy új csövet, amit a ciszterna hoz létre, méghozzá úgy, hogy az egyik végét magához csatlakoztatja, másik végét pedig a testernek adja.

<b>Use-case neve</b>	Tester steps 2
<b>Rövid leírás</b>	A tesztelő kétszer lép sikeresen egymás után.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	A tesztelő először egy csőről egy pumpára, majd erről a pumpáról egy csőre lép.

<b>Use-case neve</b>	Tester connects Pipe to Pump
<b>Rövid leírás</b>	A tesztelő megpróbál egy csövet csatlakoztatni ahoz a pumpához, amin áll.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	A tesztelő az éppen a kezében tartott csővéget próbálja felcsatlakoztatni arra a pumpára, amin áll, a csővég részéről is beállítódik ez a kapcsolatot.

<b>Use-case neve</b>	Tester breaks working Pipe and it accepts water
<b>Rövid leírás</b>	A tesztelő eltör egy teli működő csövet, ami ezután vizet fogad.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	A tesztelő eltör egy csövet, amiben volt víz, így kifolyik, ezzel növelve a szabotőrök pontjait, ezután az eltört, üres

	cső vizet fogad, amelyből a víz kifolyik és a szabotőrök pontjait növeli.
--	---

<b>Use-case neve</b>	Pipe network clogs
<b>Rövid leírás</b>	A csőhálózat feltelik vízzel és eldugul.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	A tesztelő egy két csőből és egy pumpából álló hálózatot feltölt teljesen vízzel, amíg eldugul. Ezután nem tud több vizet bele pumpálni.

<b>Use-case neve</b>	Tester disconnects empty Pipe from Pump
<b>Rövid leírás</b>	A tesztelő lecsatlakoztat egy üres csövet egy pumpáról.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	A tesztelő lecsatlakoztat egy üres csövet egy pumpáról és a csőben nincs víz.

<b>Use-case neve</b>	Pump steps twice after repair
<b>Rövid leírás</b>	A pumpa javítás után kettőt lép.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	Egy elromlott pumpát megjavít a tesztelő, majd lépteti kétszer, mire a pumpa vizet kell pumpáljon.

<b>Use-case neve</b>	Working Pipe Pumps
<b>Rövid leírás</b>	A működő pumpa pumpál.
<b>Aktorok</b>	Tester

<b>Forgatókönyv</b>	Egy működő pumpa először a tartályába mozgatja a vizet, majd elkezdi pumpálni.
---------------------	--

<b>Use-case neve</b>	Second person tries to step on pipe
<b>Rövid leírás</b>	Egy másik játékos egy már foglalt csőre akar rálépní.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	A játékos megpróbál rálépní a foglalt csőre, ez nem sikerül.

<b>Use-case neve</b>	Source steps
<b>Rövid leírás</b>	A tesztelő lépteti a két csővel rendelkező forrást. Az egyik cső üres, a másik tele van.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	Az egyik csőbe sikerül a vízátvezetés, a másikba nem.

<b>Use-case neve</b>	Place pump
<b>Rövid leírás</b>	Egy szerelő felvesz egy pumpát és lerakja az egyik csövön.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	A pumpán álló szerelő felvesz egy pumpát, átmegy egy csőre és lerakja a pumpát.

<b>Use-case neve</b>	Tester disconnects out pipe from pump
<b>Rövid leírás</b>	A tesztelő az egyik pumpáról lekötí a kivezető csövet és kétszer lépteti a pumpát.
<b>Aktorok</b>	Tester

<b>Forgatókönyv</b>	Az első léptetés után a víztartály megtelikvízzel, mivel a bemenő csőben van víz. A második léptetés után nem történik semmi, mert nincsen kivezető cső.
---------------------	--

## 5.2 A szkeleton kezelői felületének terve, dialógusok

[A szkeleton által elfogadott bemenetek , valamint a szöveges konzolon megjelenő kimenetek. A kiemenet formátuma olyan kell legyen, ami alapján a működés összehető a korábbi szekvencia-diagramokkal.]

A szkeleton program indulásakor az üdvözlés után a képernyőn a tesztelhető esetek jelennek meg sorszámozva. A kilistázott menüpontok közül a sorszám begépelésével lehet választani. Ezek a menüpontok lefedik azokat a use-case-ket, amiket ebbe a dokumentumban meghatároztunk.

Minden egyes teszteset elején egy felsorolást láthatunk a tesztesetben szereplő objektumok neéről és típusáról. A felsorolás után az objektumok belső állapotáról is áttekintő információt kapunk. (Nyilván az ilyen szempontból releváns objektumok állapotáról, pl. törött cső, pumpa üres tartállyal).

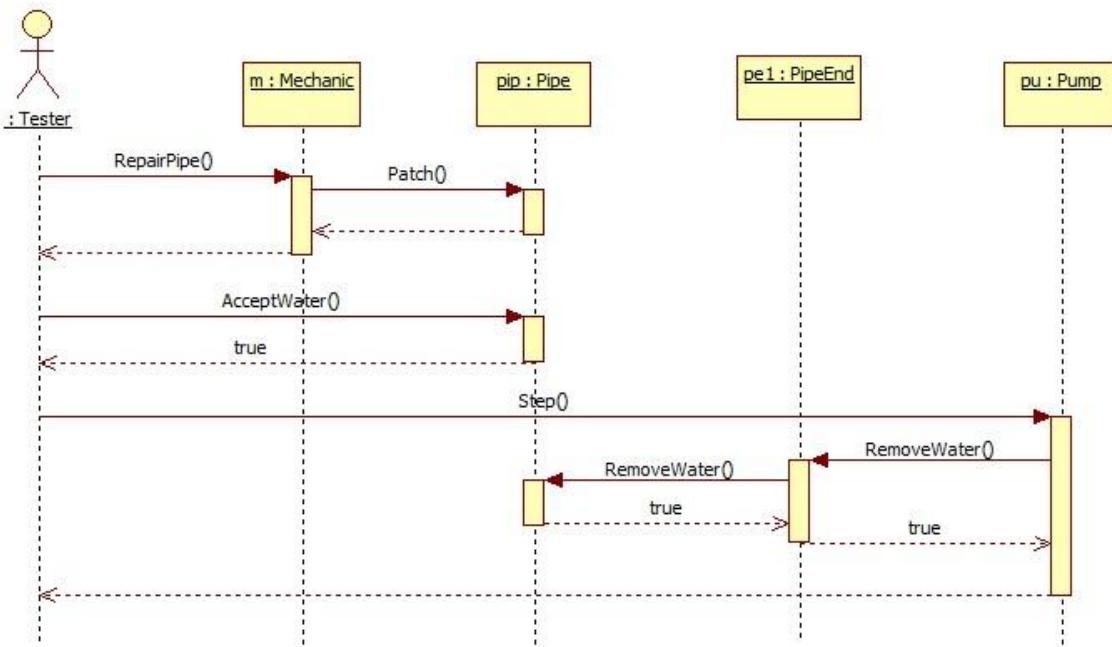
Ezután az összes függvényhívás megjelenik a kimeneten új sorban a következő formába:

(behúzás) (objektum, amin a függvényhívás történt) (meghívott függvény neve) (függvény visszatérési értéke)

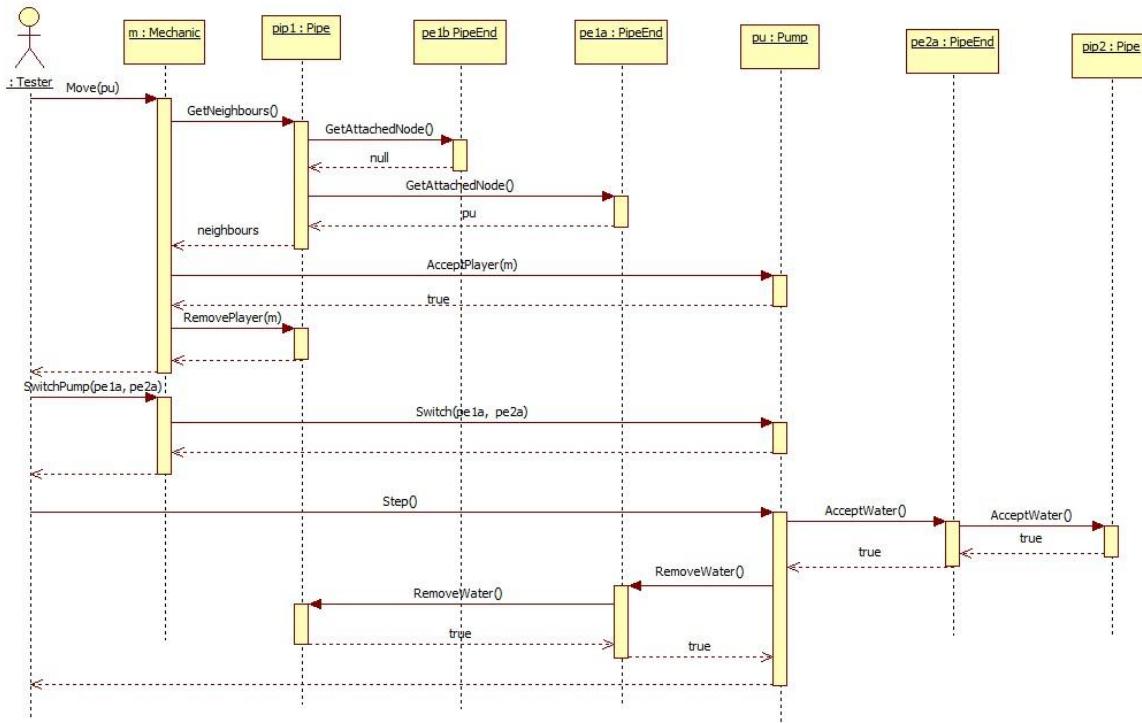
A meghívott függvények mind publikusak, az objektumok belső állapotának változását a kimeneten nem figyelhetjük meg.

## 5.3 Szekvencia diagramok a belső működésre

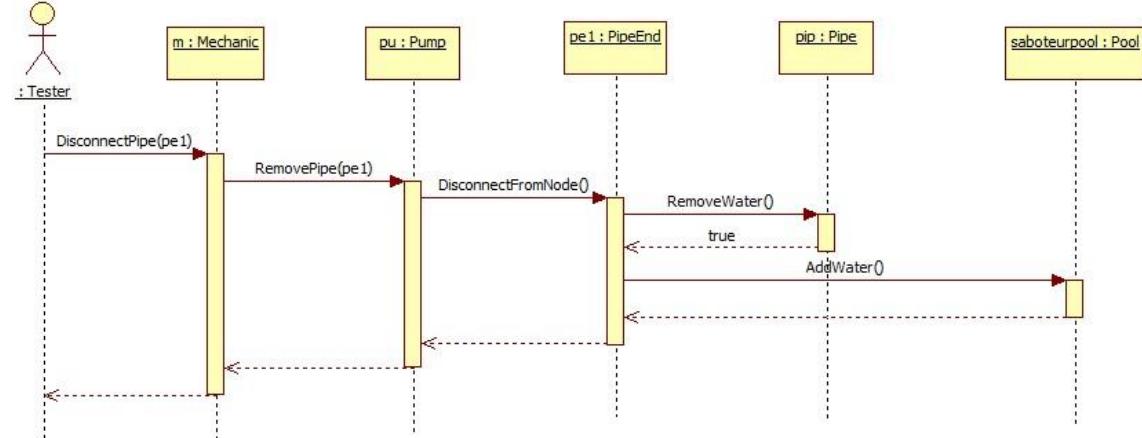
### 5.3.1 Tester repairs a leaking pipe



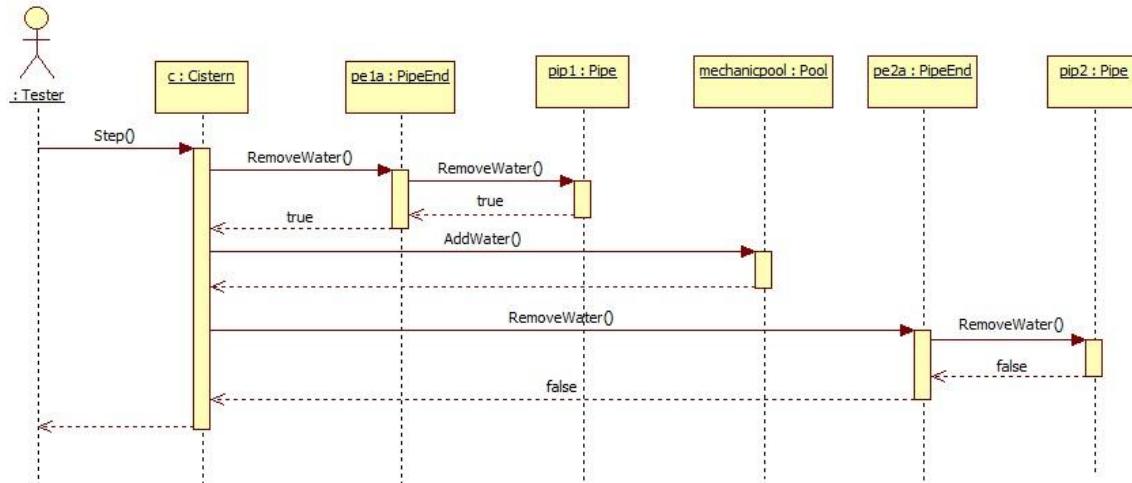
### 5.3.2 Tester steps on Pump and switches it



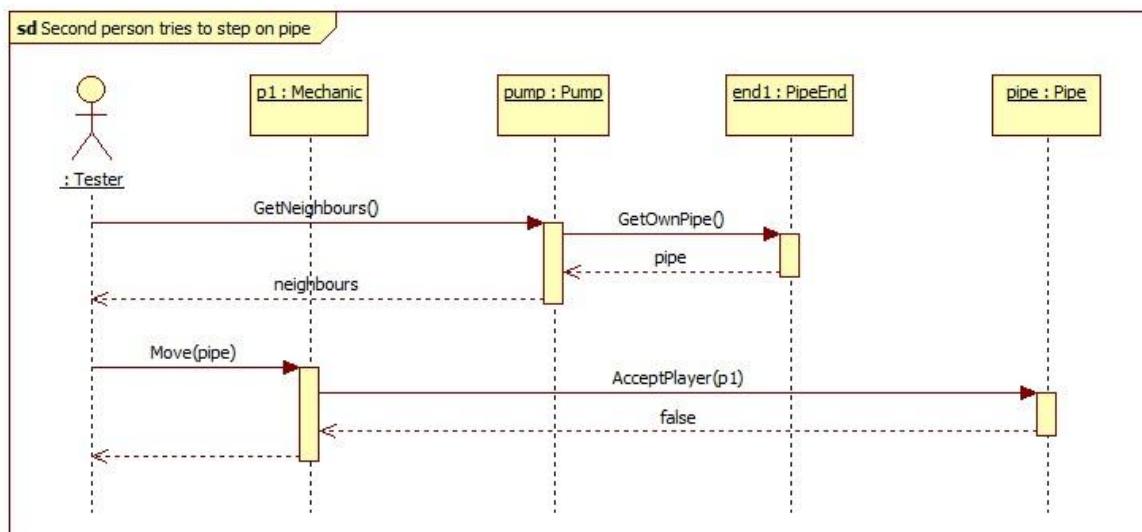
### 5.3.3 Tester disconnects a full Pipe



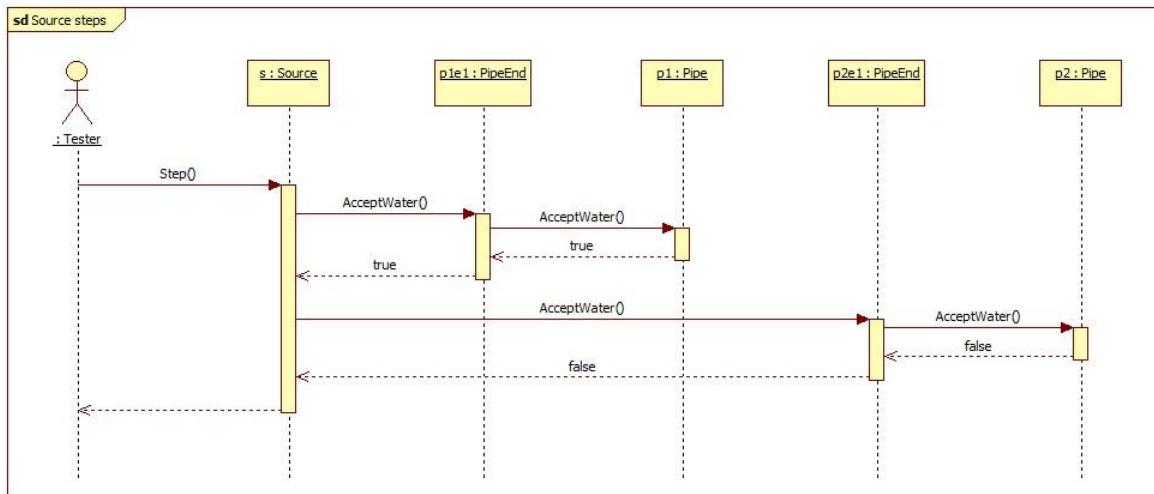
### 5.3.4 Tester steps Cistern



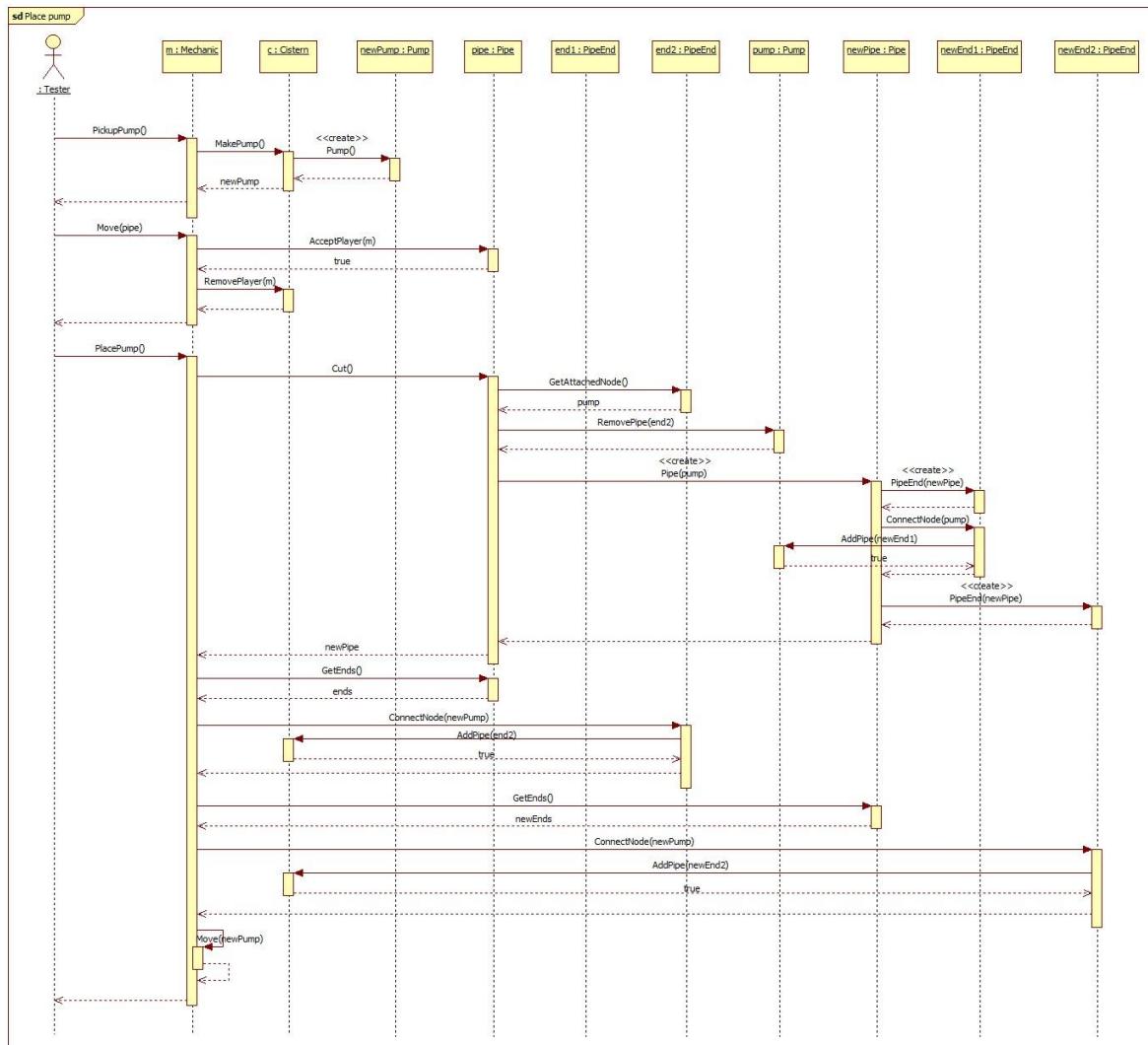
### 5.3.5 Second person tries to step on pipe



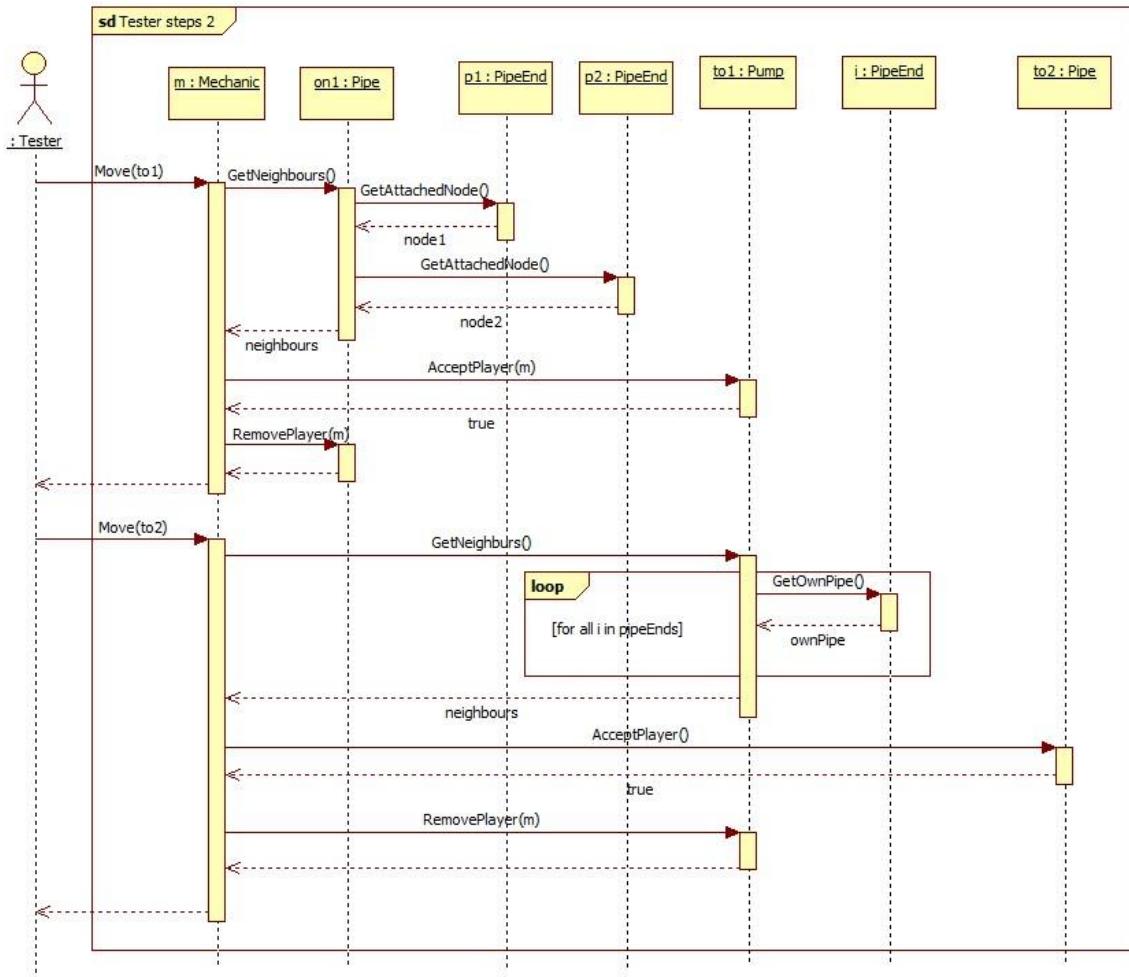
### 5.3.6 Source steps



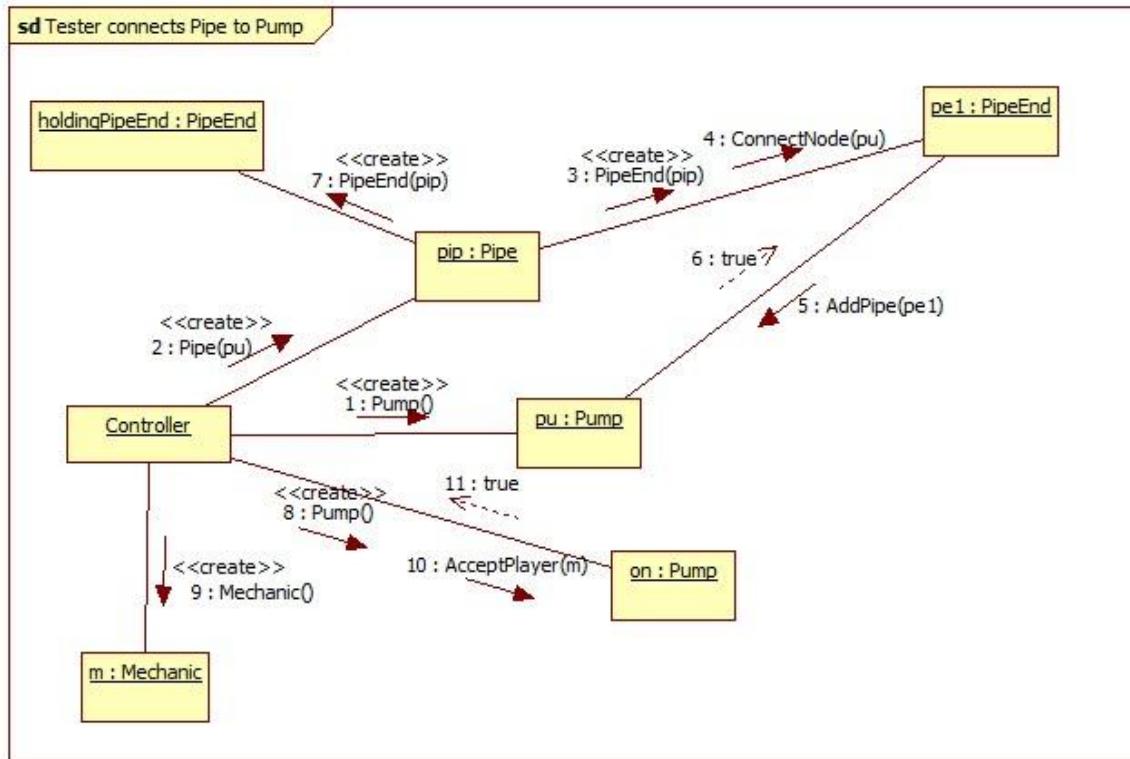
### 5.3.7 Place pump



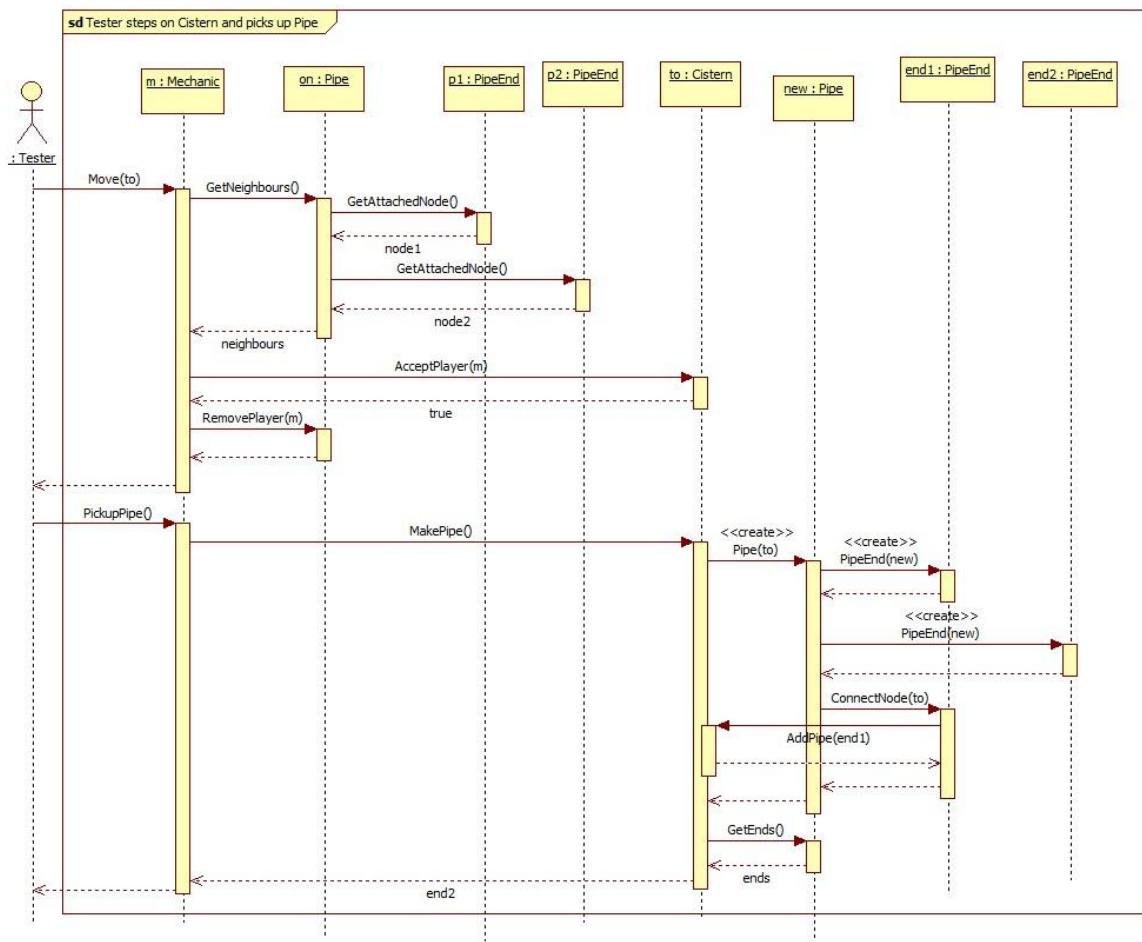
### 5.3.8 Tester steps 2



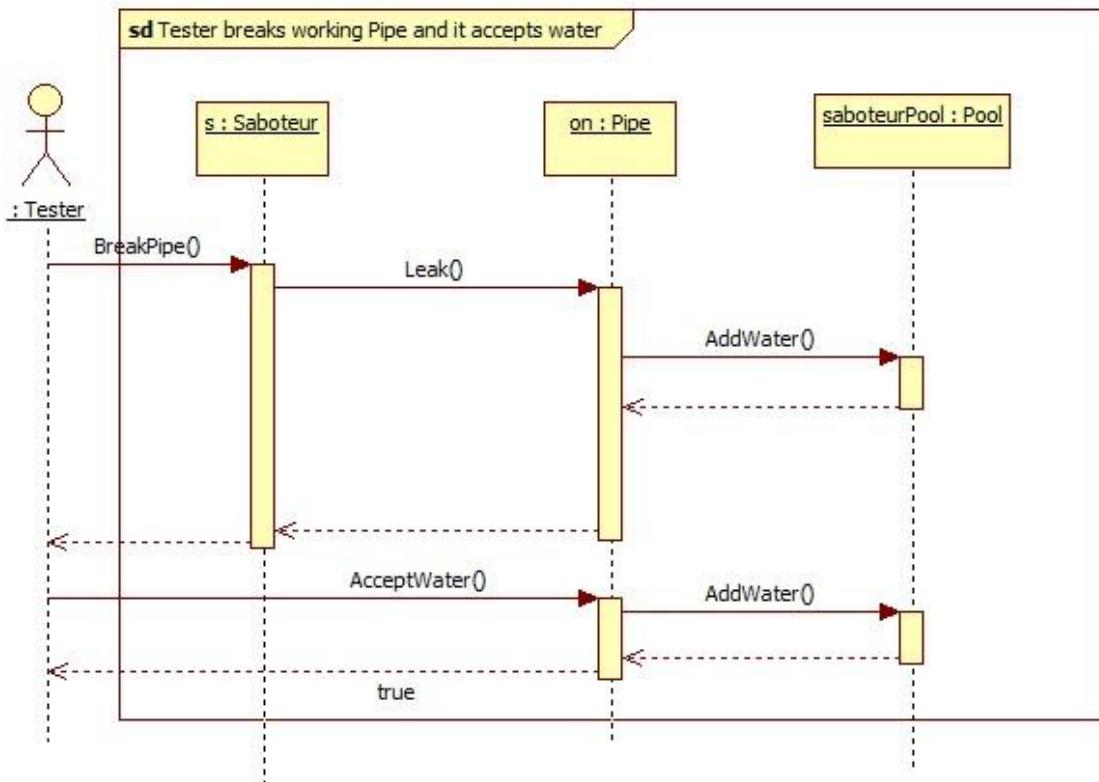
### 5.3.9 Tester connects Pipe to Pump



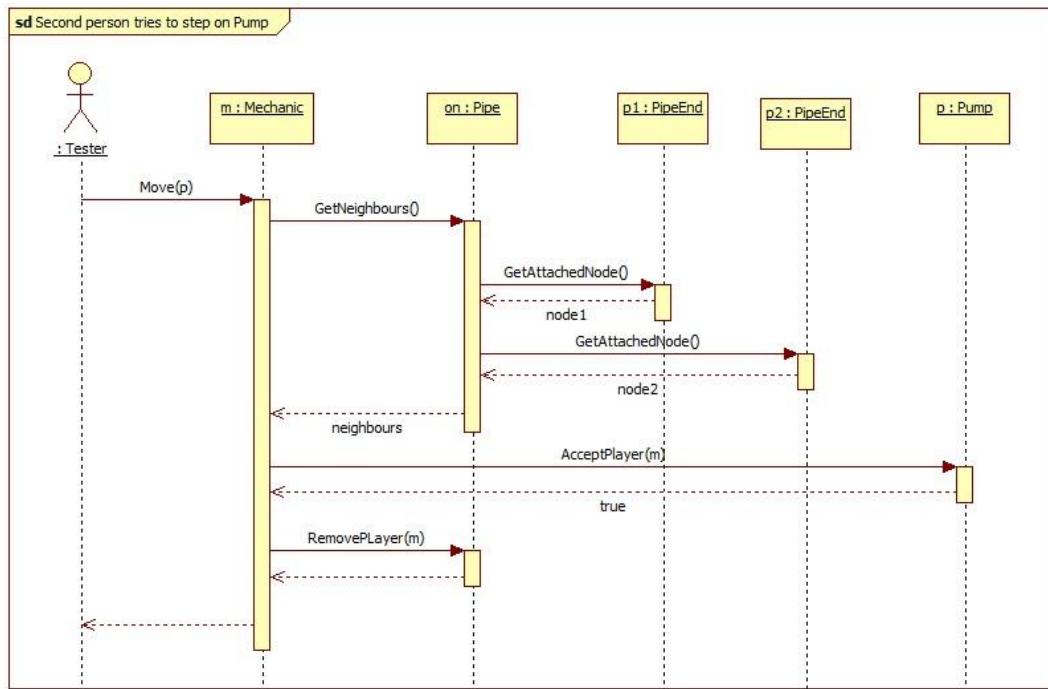
### 5.3.10 Tester steps on Cistern and picks up Pipe



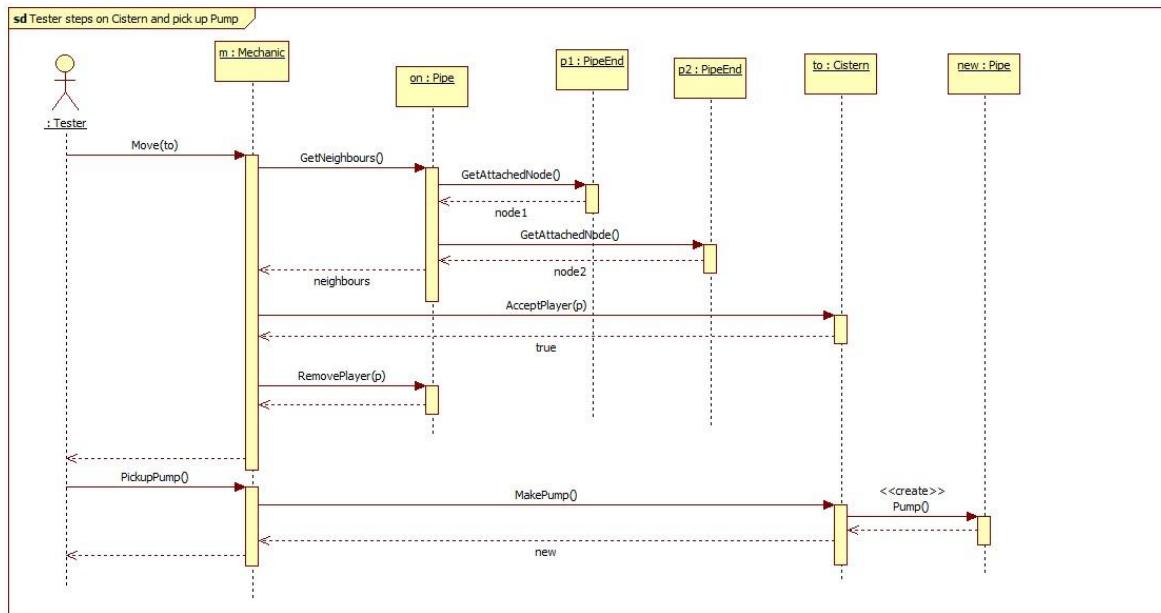
### 5.3.11 Tester breaks working Pipe and it accepts water



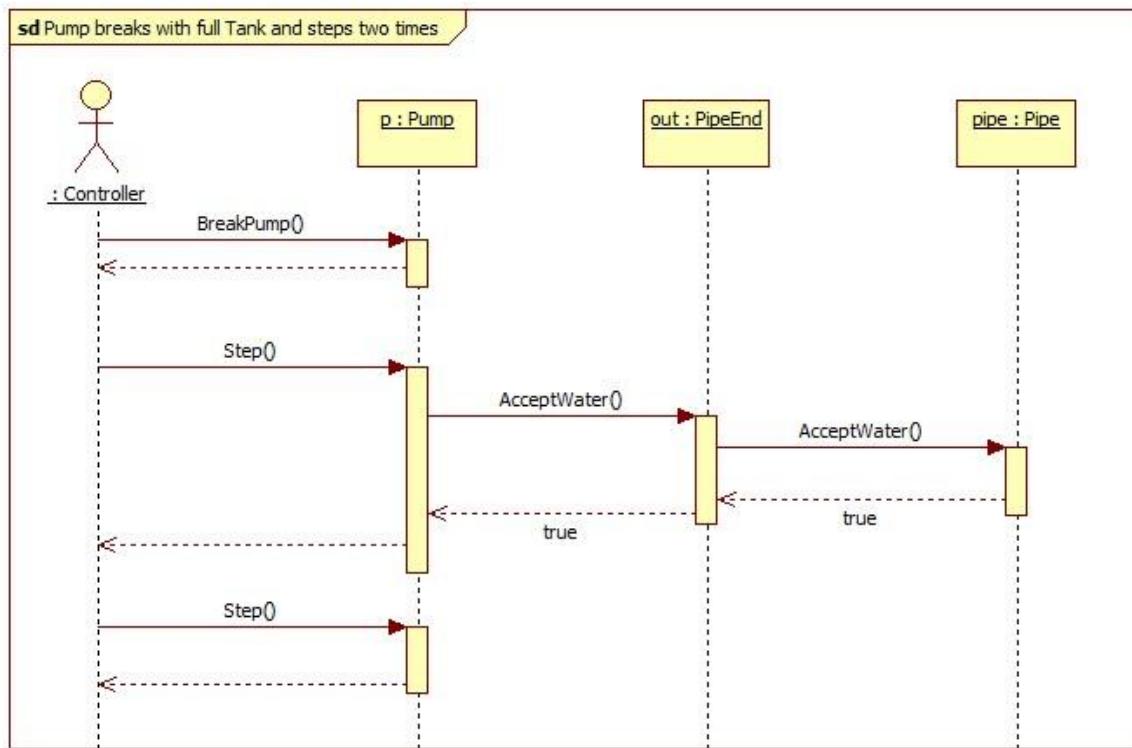
### 5.3.12 Second person tries to step on Pump



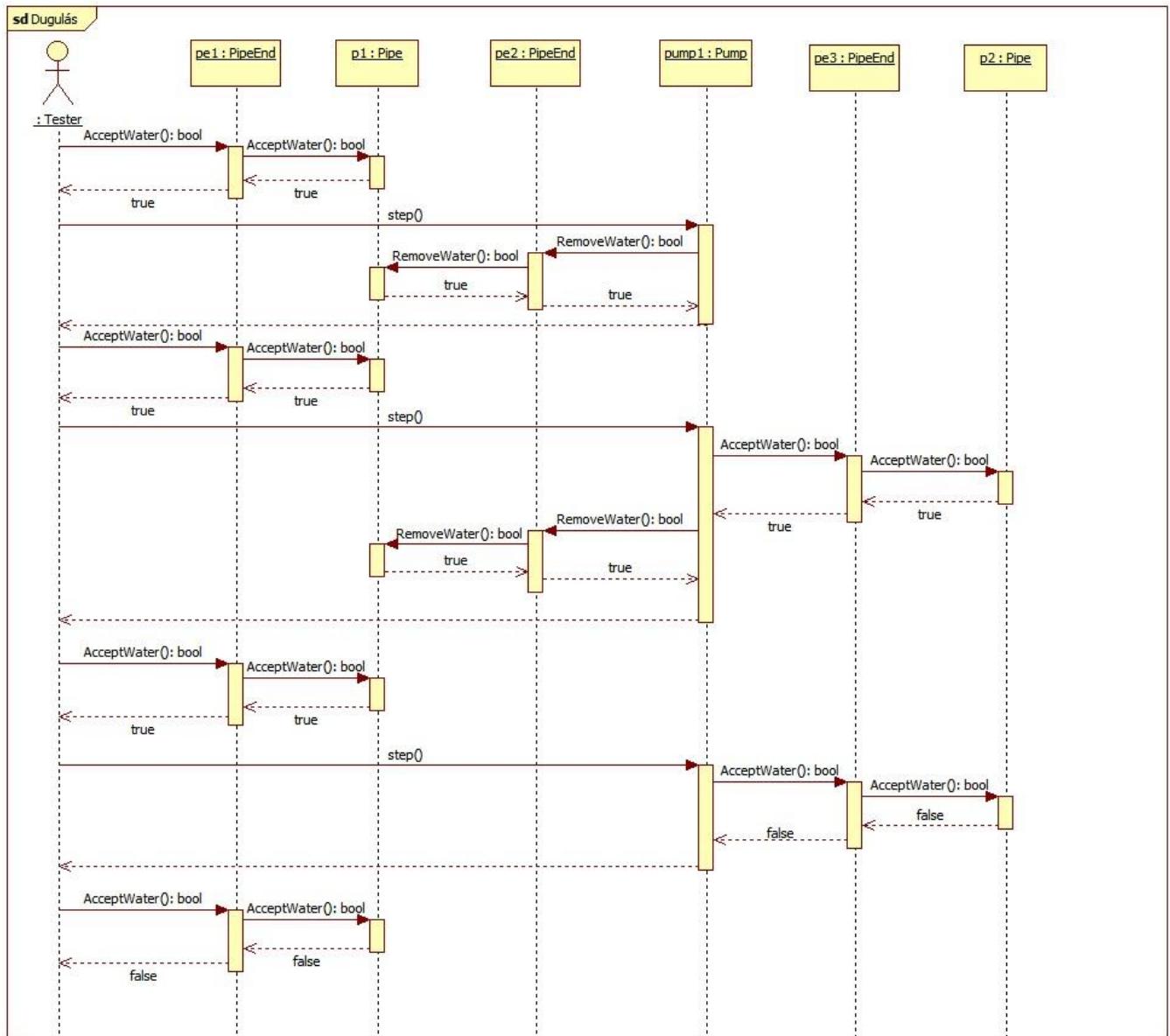
### 5.3.13 Tester steps on Cistern and picks up Pump



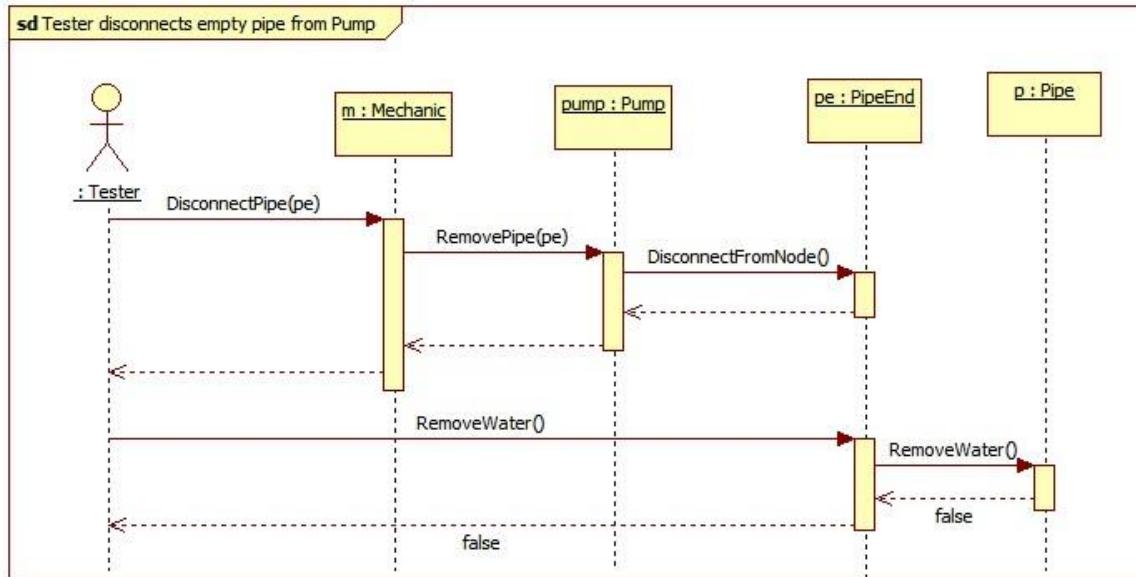
### 5.3.14 Pump breaks with full Tank and steps two times



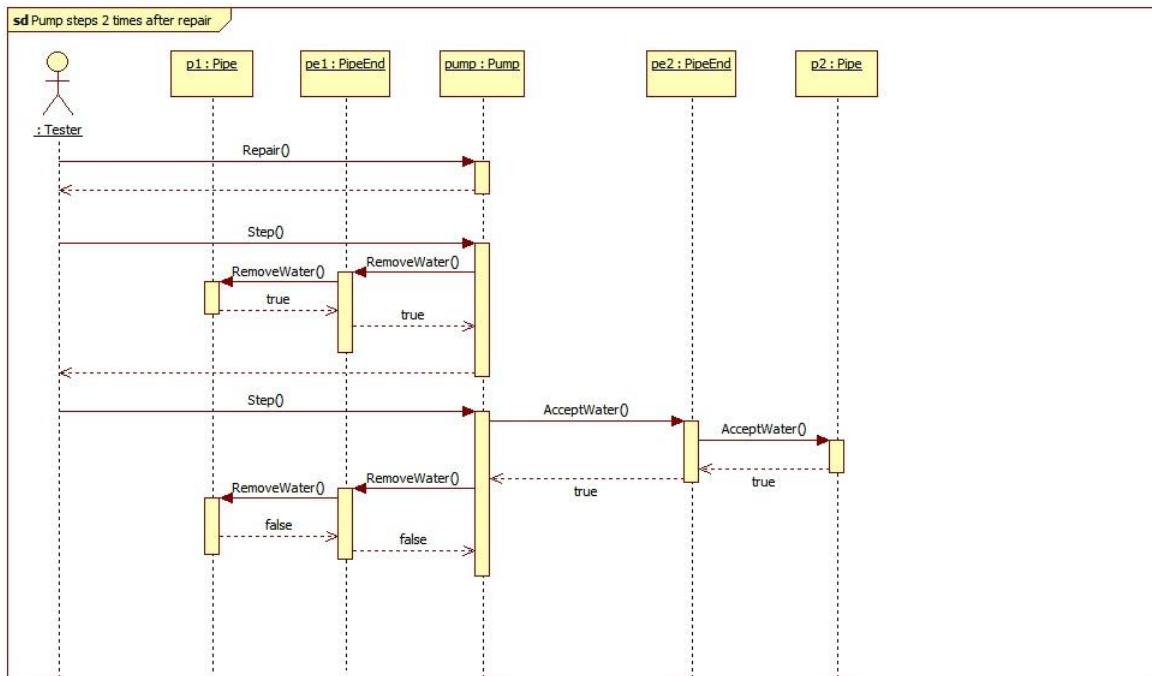
### 5.3.15 Pipe network clogs



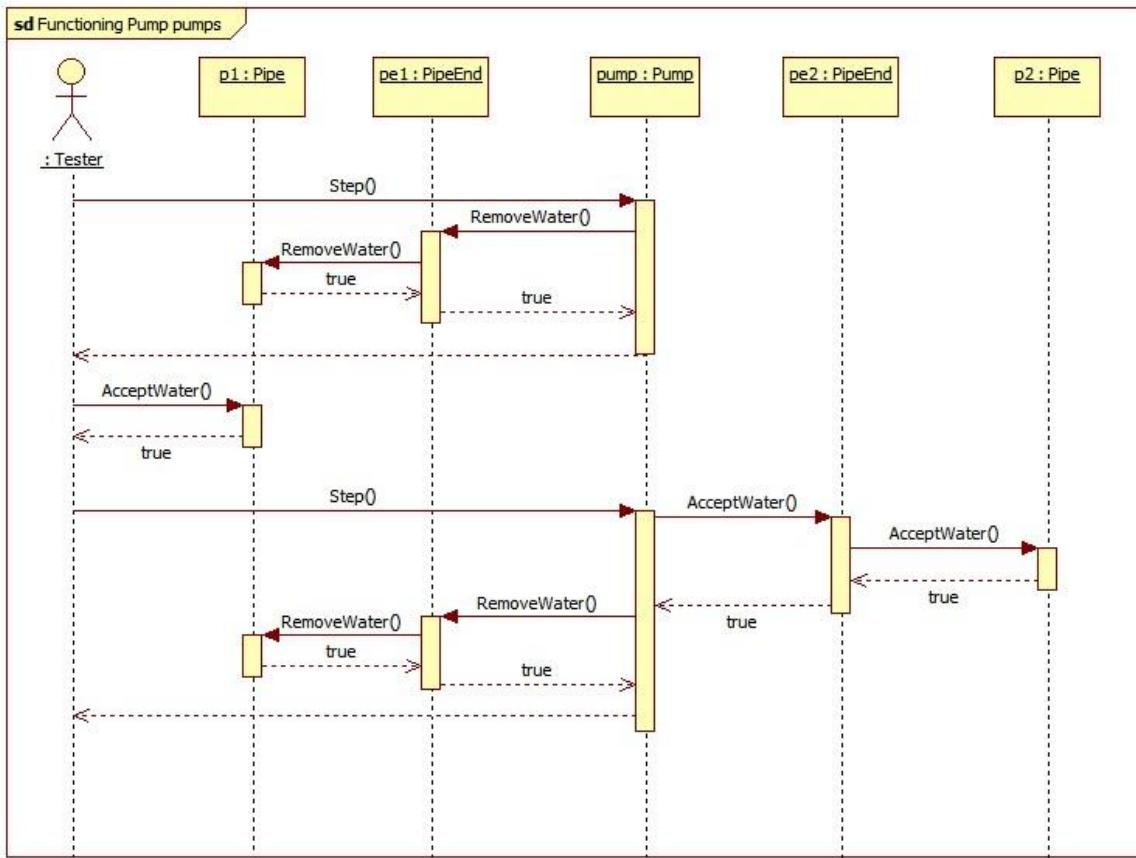
### 5.3.16 Tester disconnects empty Pipe from Pump



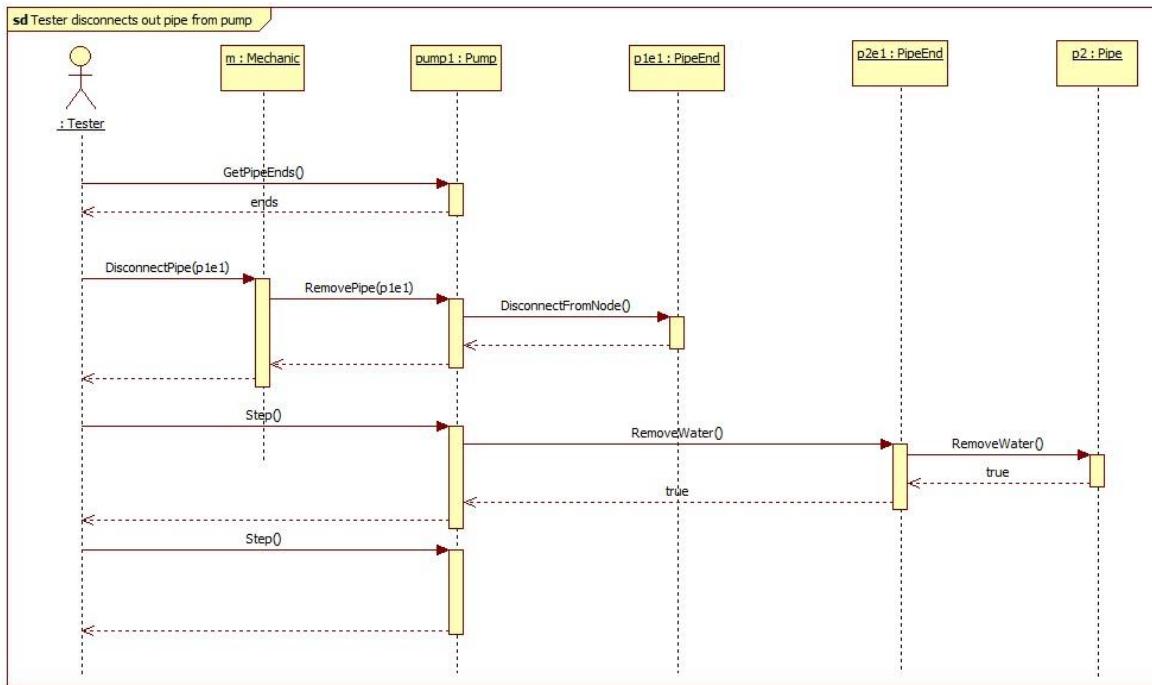
### 5.3.17 Pump steps twice after repair



### 5.3.18 Working Pump pumps



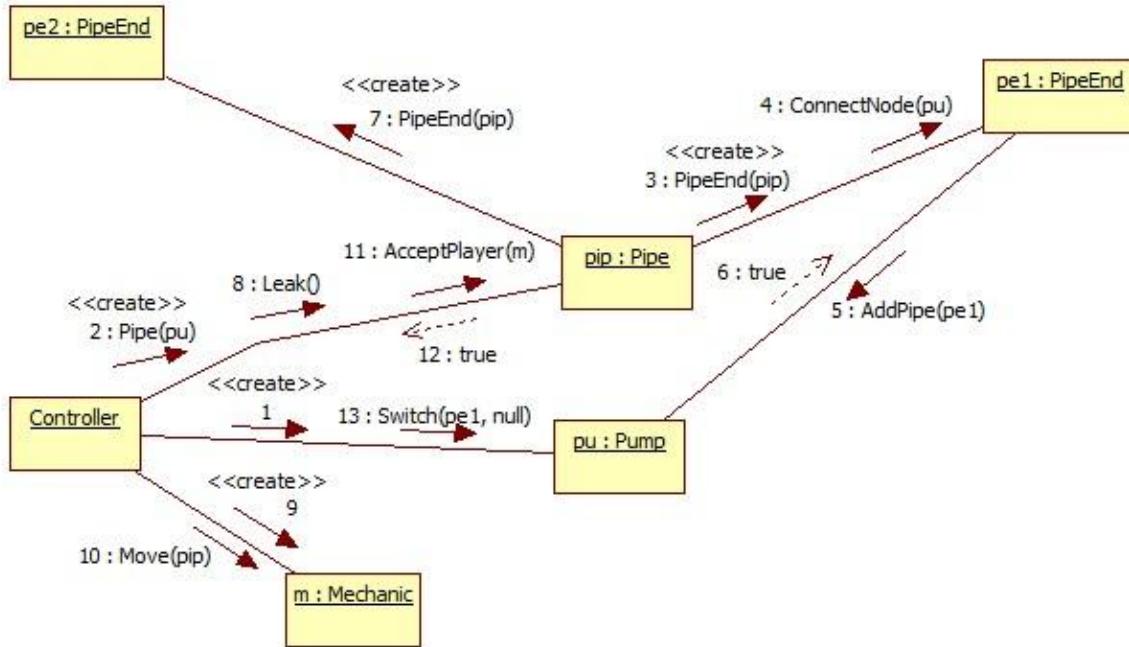
### 5.3.19 Tester disconnects out pipe from pump



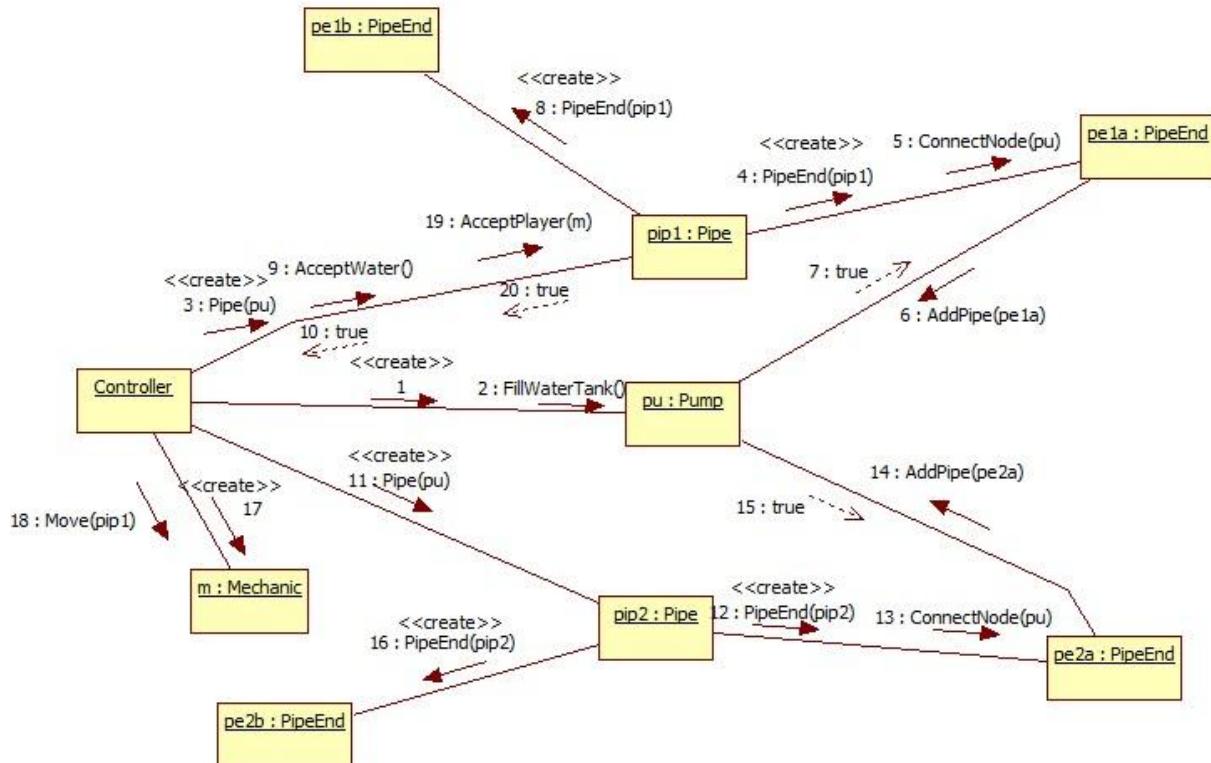
## 5.4 Kommunikációs diagramok

A kommunikációs diagramok az egyes tesztesetek lefutása előtti inicializálást ábrázolják. Az előző fejezetben szereplő szekvenciadiagramokon ábrázolt teszteket a megegyező sorszámú kommunikációs diagram inicializálja.

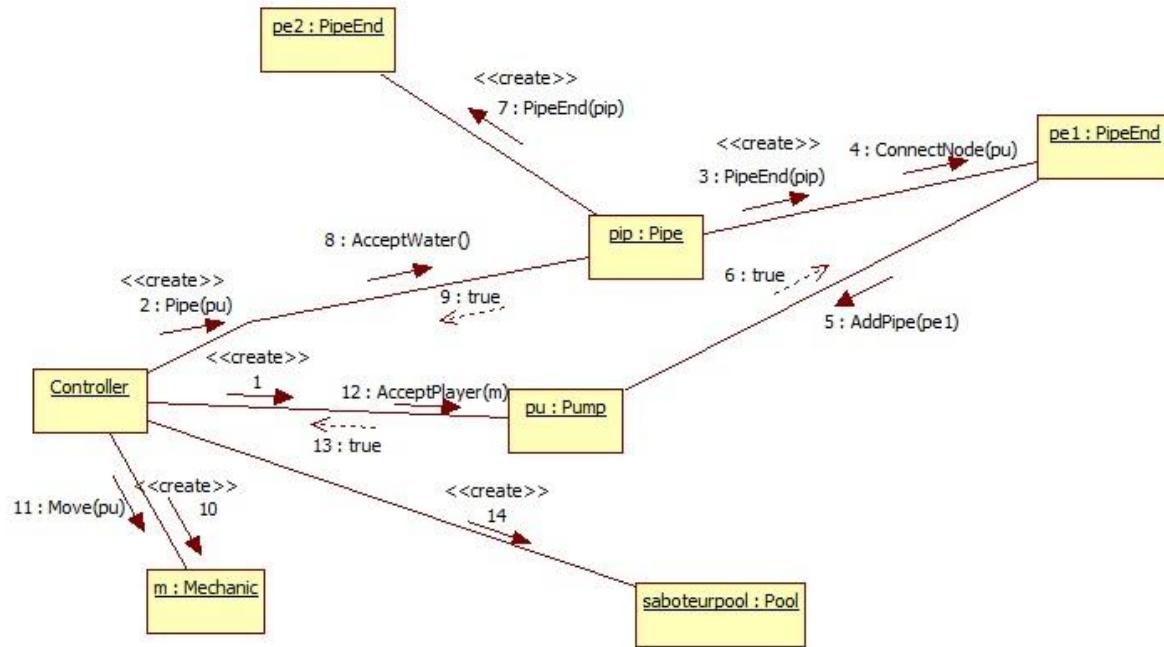
### **5.4.1 Tester repairs a leaking pipe**



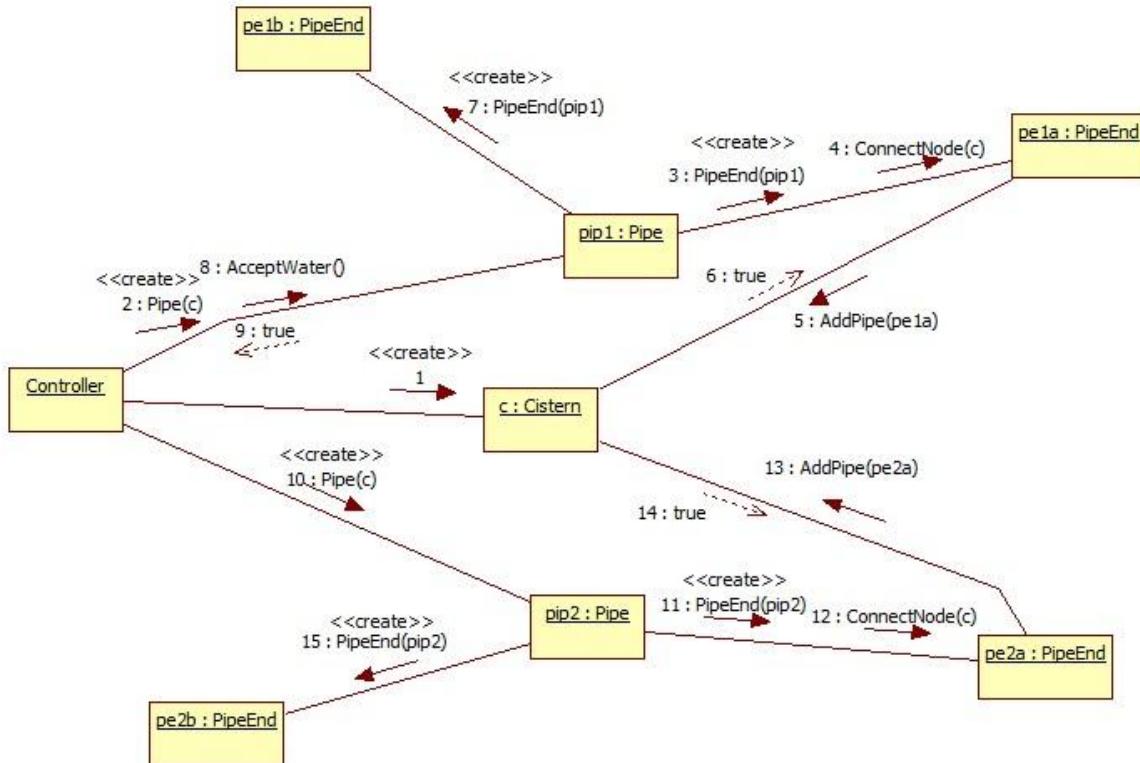
#### **5.4.2 Tester steps on Pump and switches it**



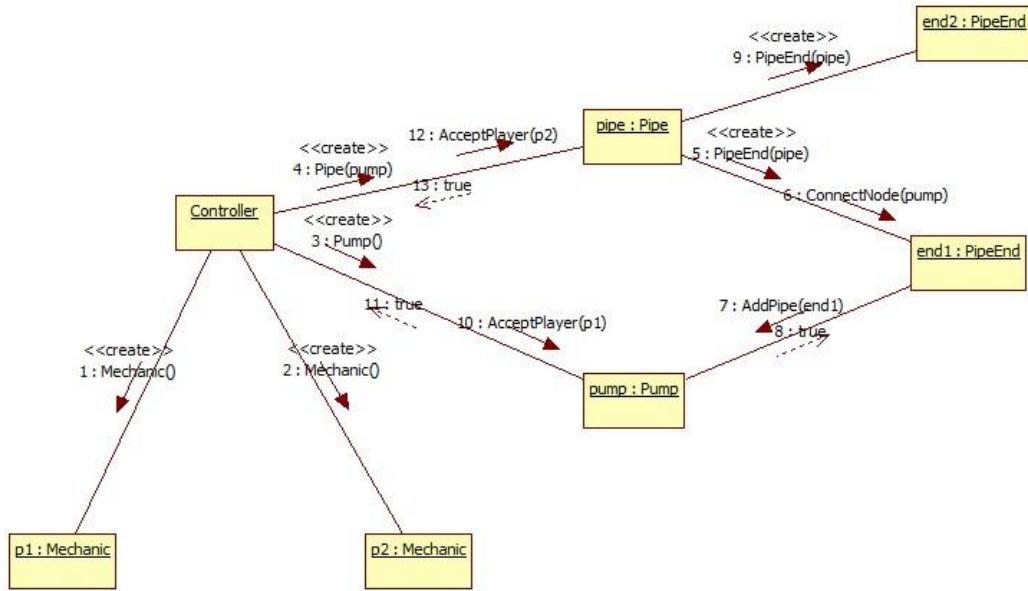
### 5.4.3 Tester disconnects a full Pipe



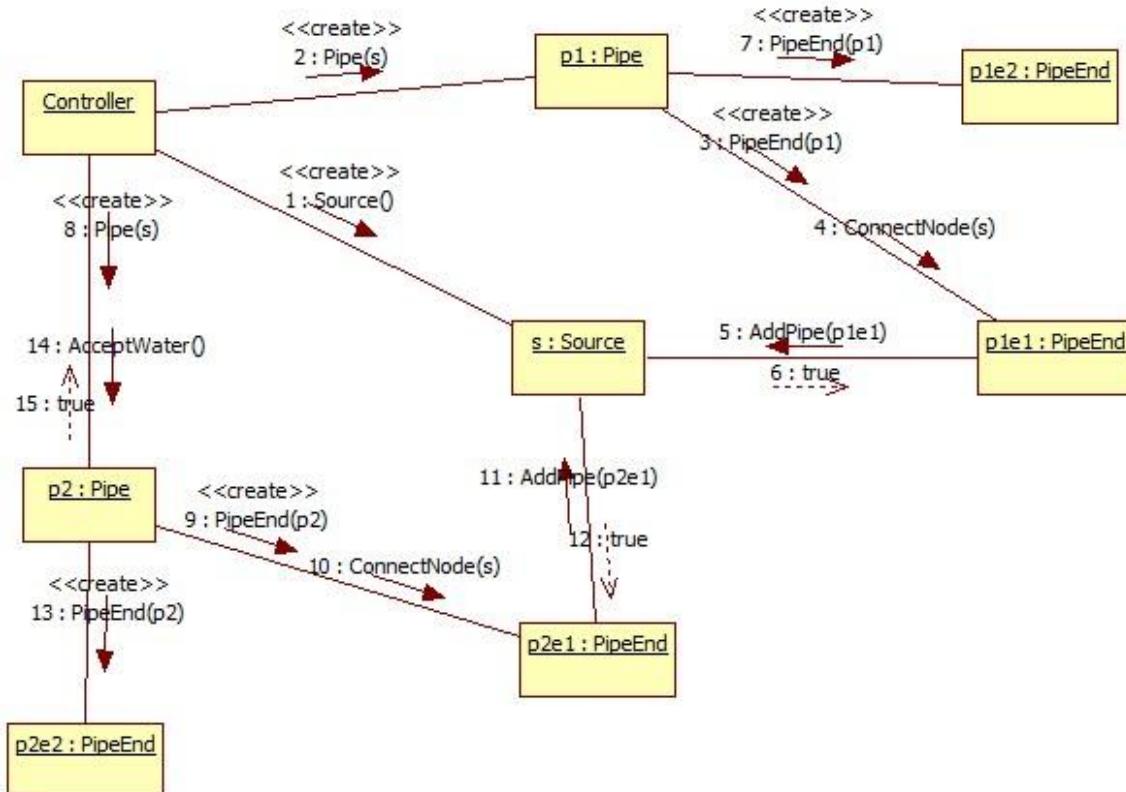
### 5.4.4 Tester steps Cistern



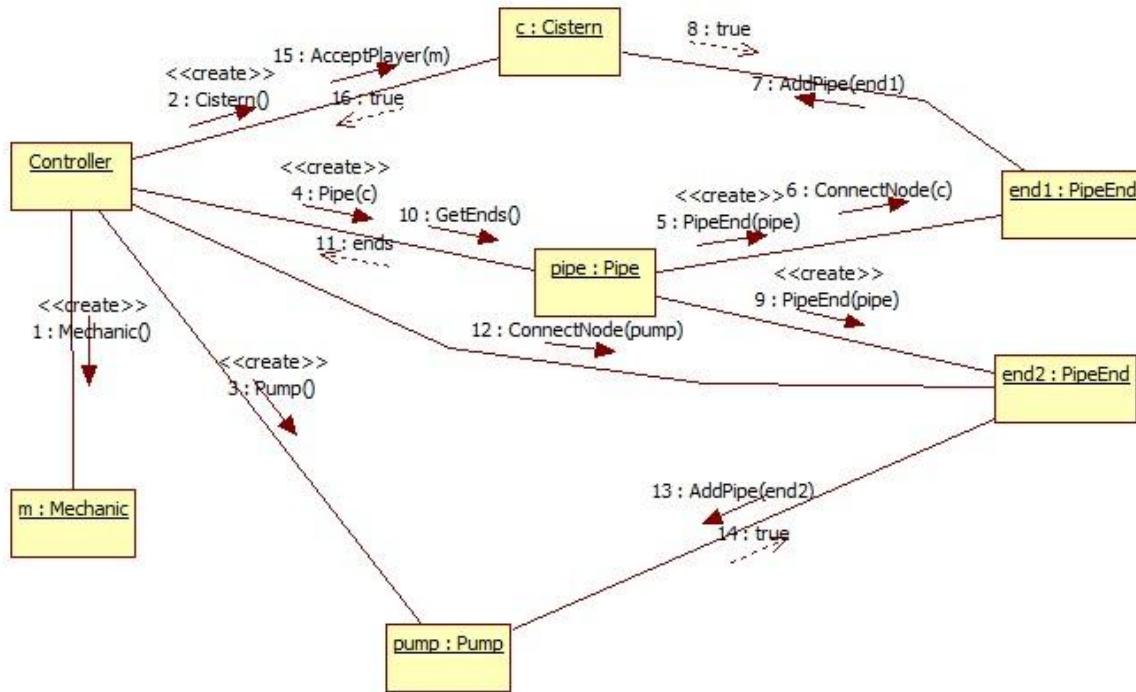
### 5.4.5 Second person tries to step on pipe



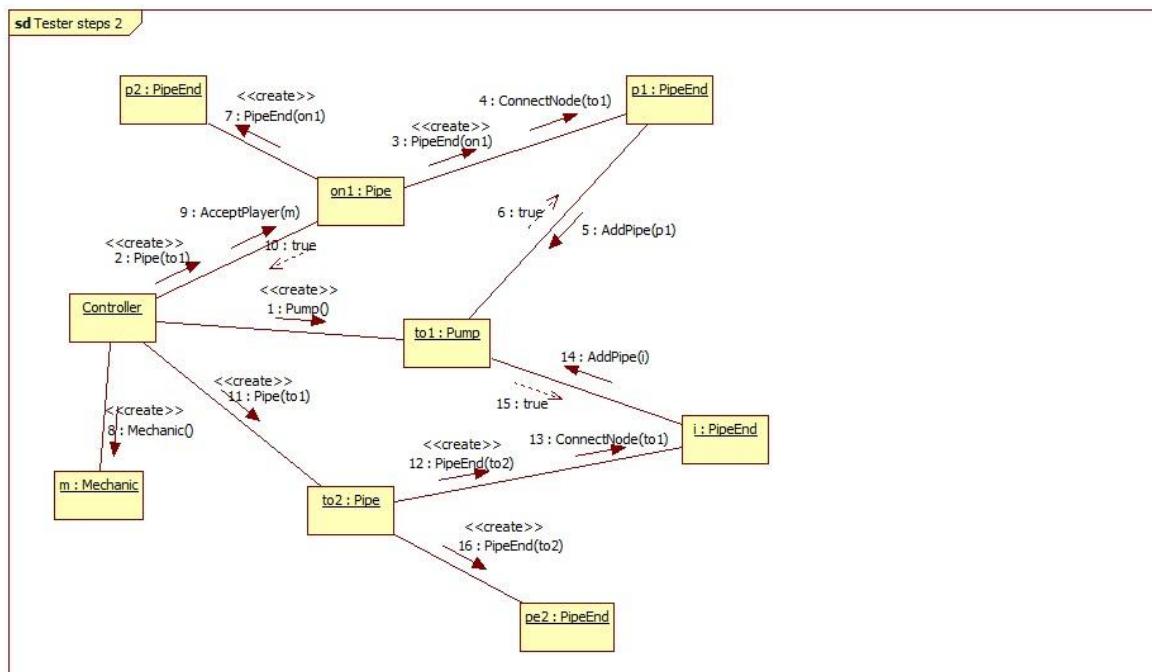
### 5.4.6 Source steps



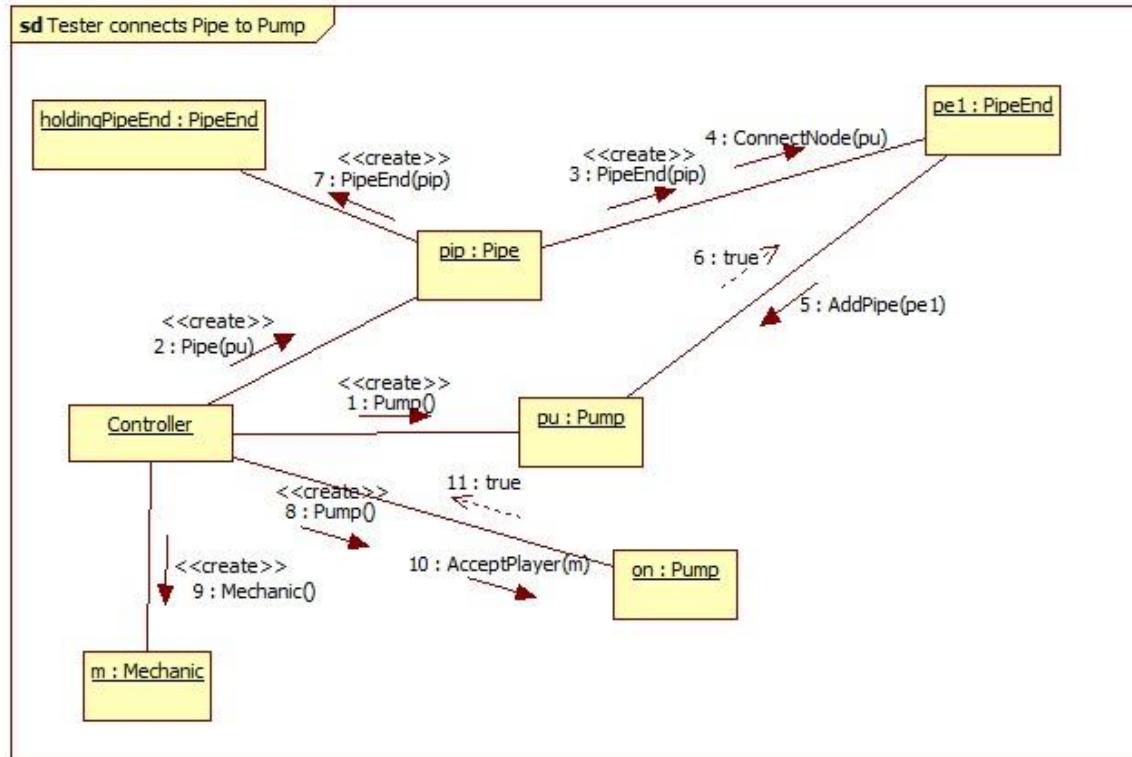
### 5.4.7 Place pump



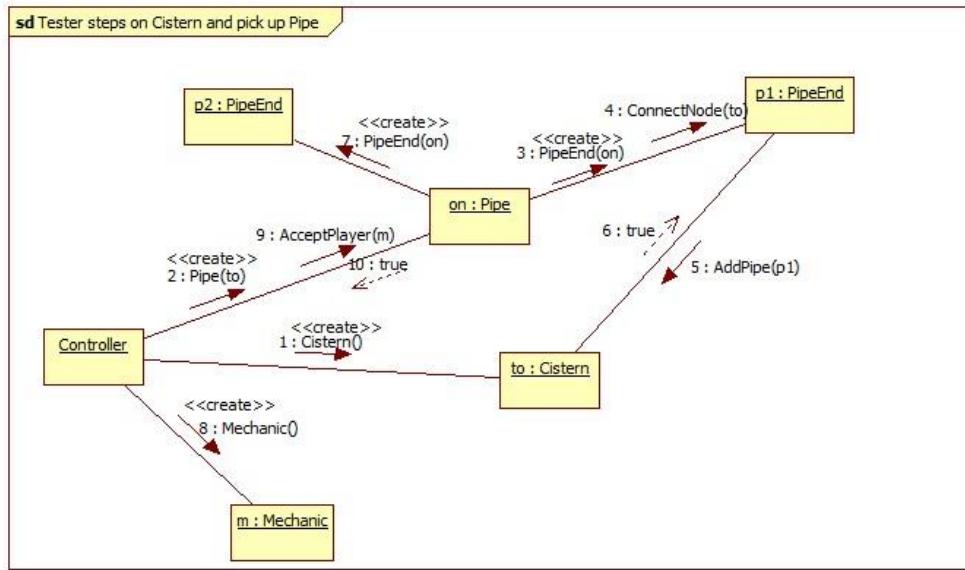
### 5.4.8 Tester steps 2



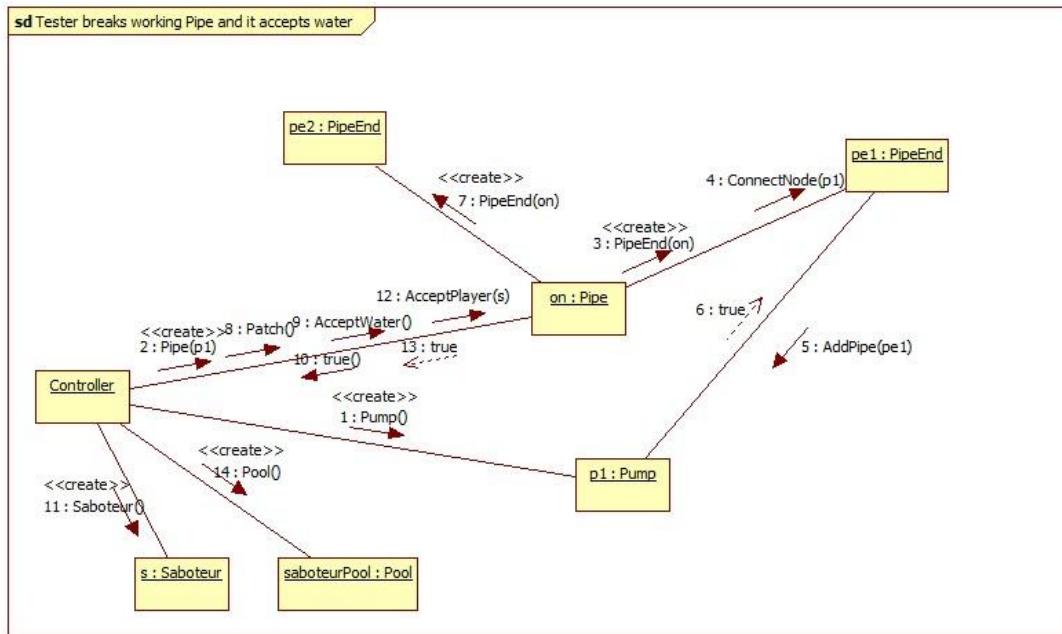
### 5.4.9 Tester connects Pipe to Pump



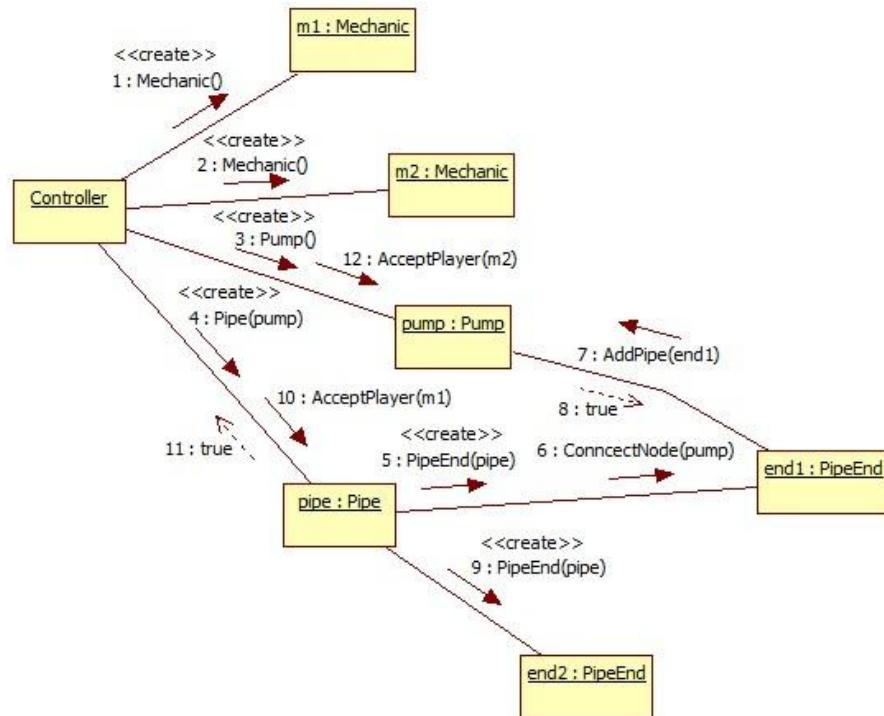
### 5.4.10 Tester steps on Cistern and picks up Pipe



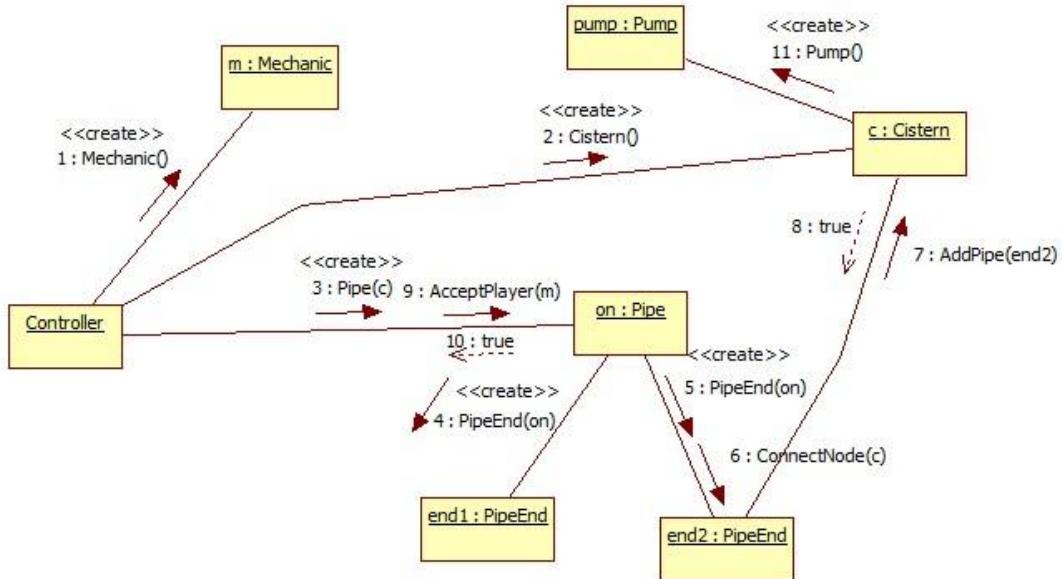
#### **5.4.11 Tester breaks working Pipe and it accepts water**



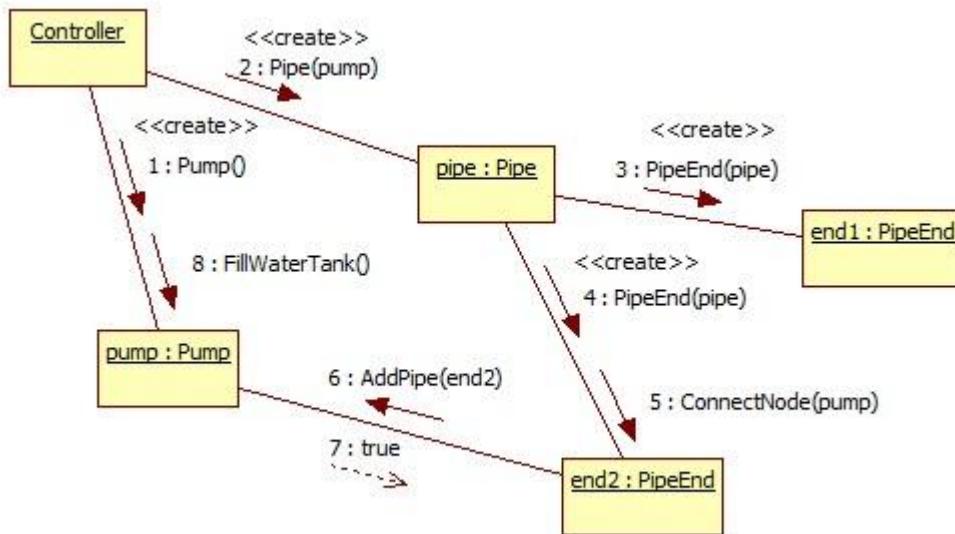
#### **5.4.12 Second person tries to step on Pump**



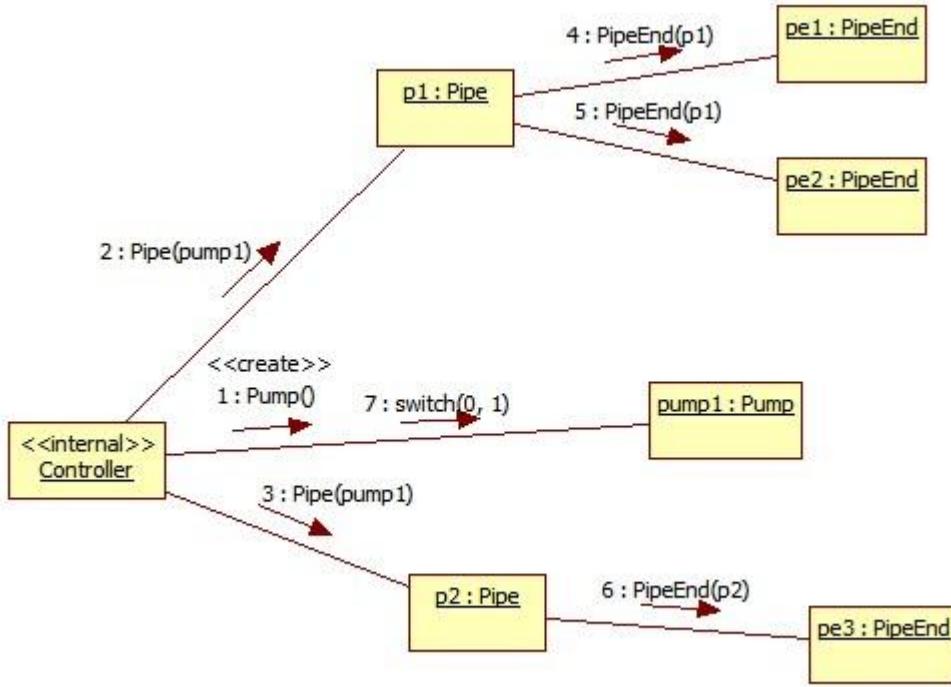
### 5.4.13 Tester steps on Cistern and picks up Pump



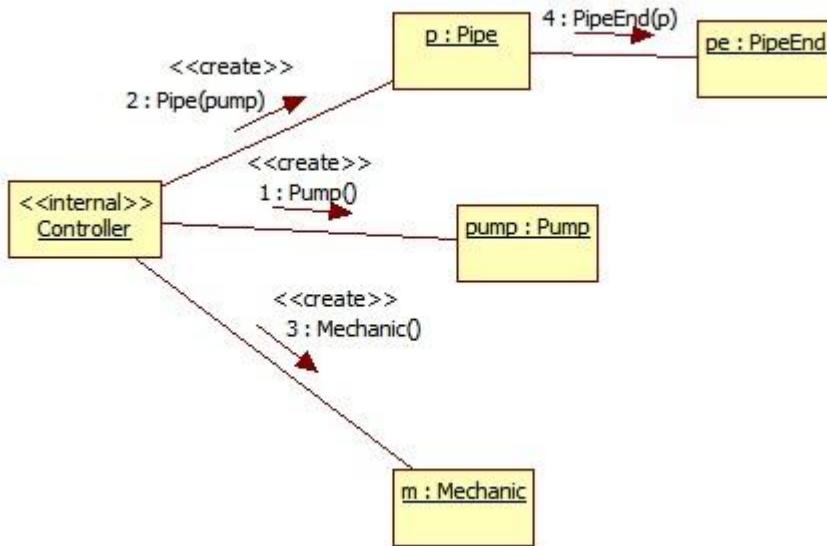
### 5.4.14 Pump breaks with full Tank and steps two times



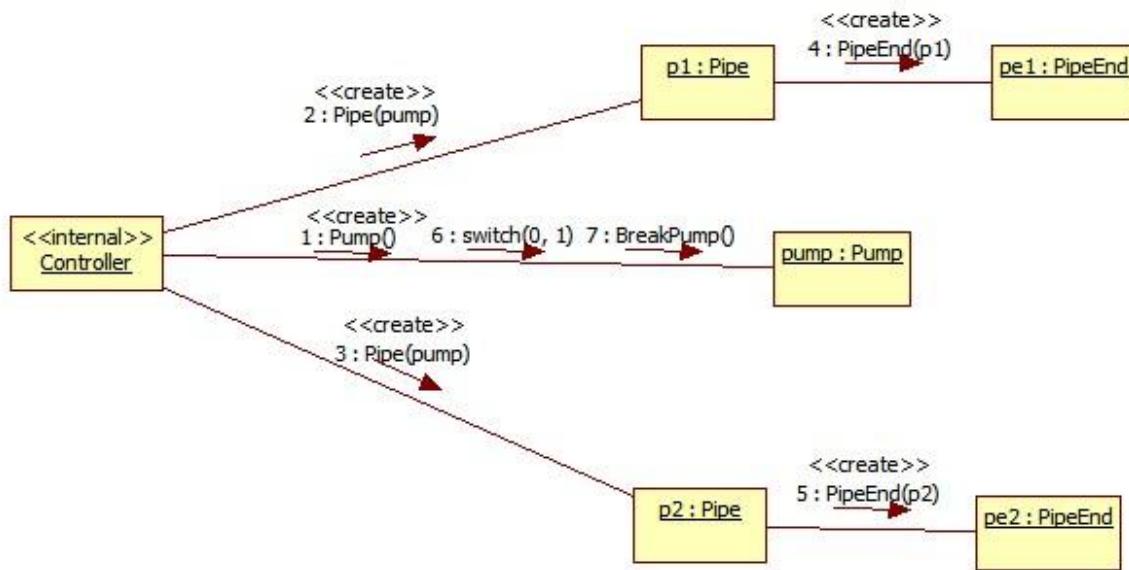
### 5.4.15 Pipe network clogs



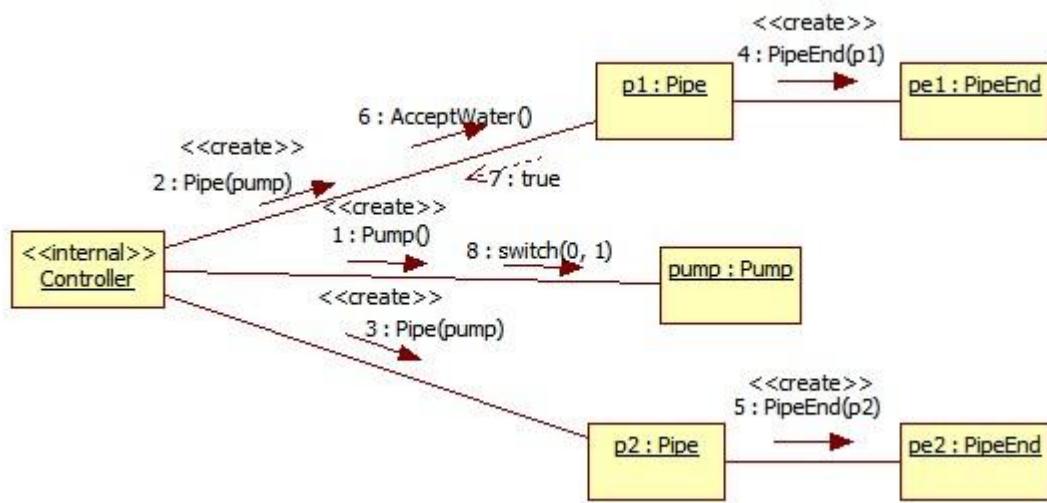
#### 5.4.16 Tester disconnects empty Pipe from Pump



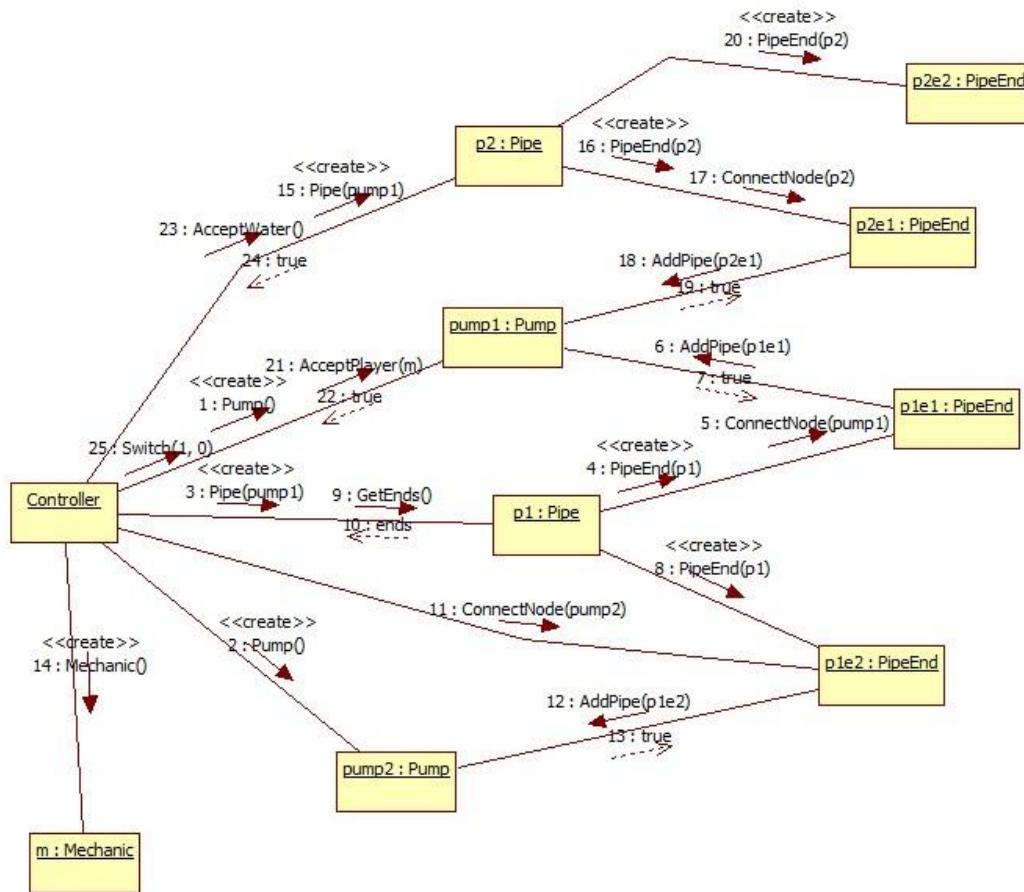
#### 5.4.17 Pump steps twice after repair



#### 5.4.18 Working Pump pumps



#### 5.4.19 Tester disconnects out pipe from pump



## 5.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2023.03.30. 20.30	2 óra	Andó Deé-Lukács Kiss Skáre Vörös	Use casek megírása, feladatok kiosztása
2023.04.01. 14.00	1 óra	Kiss	Use casek leírása: Tester steps on Cistern and picks up Pump, Second person tries to step on Pump, Pump breaks with full Tank and steps two times,

			Szekvenciadiagram: 5.3.12
2023.04.01 16:00	1,5 óra	Andó	Use case leírások, ezekhez tartozó szekvencia és kommunikációs diagramok: Tester steps 2, Tester steps on Cistern and picks up Pipe
2023. 04.01. 11:00	5 óra	Vörös	Use case leírások, szekvencia- és kommunikációs diagramok az 5.3.1-5.3.4-es számúak. Szkeleton dialógusai
2023. 04. 02. 14.00	4 óra	Deé-Lukács	Szekvencia diagramok: 5.3.15-18 Kommunikációs diagramok: 5.4.15-18
2023. 04. 02 15.00	3 óra	Kiss	Szekvenciadiagramok: 5.3.13, 5.3.14 Kommunikációdiagramok: 5.4.12-14
2023.04.02. 14:00	1 óra	Andó	Szekvencia diagramok, kommunikációs diagramok megírása: Tester connects Pipe to Pump, Tester breaks working Pipe and it accepts water

2023.04.02. 22:00	0,5 óra	Andó	Szekvencia és kommunikációs diagramok hibáinak javítása
2023.04.01. 23:15	0,75 óra	Skáre	Az osztálydiagram és szekvenciadiagramok hibáinak kijavítása.
2023.04.02. 00:30	2 óra	Skáre	5.3.5-7 és 5.4.5-7 megrajzolása
2023.04.02. 21:00	2 óra	Andó, Kiss, Skáre, Deé-Lukács, Vörös	Értekezlet: elkészült diagramok ellenőrzése, javítása
2023.04.02. 21:30	1 óra	Skáre	5.3.19 és 5.4.19 megrajzolás

# **6. Szkeleton beadás**

**9 – Capybara**

Konzulens:  
**Goldschmidt Balázs**

## **Csapattagok**

Andó Viola	H51B9J	viola.ando0618@gmail.com
Deé-Lukács András Gergely	JHFLXZ	gergoesjanka@gmail.com
<u>Kiss Blanka Zselyke</u>	C6037J	kissblanka02@gmail.com
Skáre Erik	Z7ZF6D	skareerik55@gmail.com
Vörös Vilmos	HW921K	v.vilmos10@gmail.com

2023. 04. 17.

# 6. Szkeleton beadás

## 6.1 Fordítási és futtatási útmutató

### 6.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Cistern.java	2,06 KB	2023.04.13. 18:30	A ciszterna osztály
Element.java	2.76 KB	2023.04.13. 18:00	Az aktív és passzív elemek űsosztálya
Game.java	0.729 KB	2023.04.15. 14:00	A pontgyűjtő medencék nyilvántartását végző osztály
ISteppable.java	0,151 KB	2023.04.13. 18:03	A léptethető dolgok interfésze
Main.java	3,05 KB	2023.04.13. 20:50	A programot irányító osztály
Mechanic.java	2,51 KB	2023.04.13. 20:10	A szerelők osztály
Node.java	2,92 KB	2023.04.13. 18:45	Az aktív elemek űsosztálya
Pipe.java	4,37 KB	2023.04.13. 19:00	A cső osztály
PipeEnd.java	3 KB	2023.04.13. 19:10	A csővég osztály
Player.java	3,9 KB	2023.04.13. 20:00	A játékosok űsosztálya
Pool.java	0,469 KB	2023.04.13. 21:05	A medence (pontgyűjtő) osztály
Pump.java	2,96 KB	2023.04.13. 19:40	A pumpa osztály
Saboteur.java	0,471 KB	2023.04.13. 20:30	A szabotőr osztály
Skeleton.java	2,83 KB	2023.04.13. 21:30	A naplázást irányító osztály
Source.java	0,530 KB	2023.04.13. 19:20	A forrás osztály
Tester.java	21,8 KB	2023.04.13. 21:45	Tesztfüggvényeket tartalmazó osztály

### 6.1.2 Fordítás

#### 1. Lépés: A környezet előkészítése

Első lépésként fel kell telepíteni a Java 18-at a számítógépre.

A gépen lévő Java JDK a következő parancssal ellenőrizhető a Terminál alkalmazásban

```
java -version
```

Ha a következőt látjuk, akkor további teendőnk a környezettel nincsen.

```
C:\Users\cloud\Downloads\src>java --version
java 18.0.2.1 2022-08-18
Java(TM) SE Runtime Environment (build 18.0.2.1+1-1)
Java HotSpot(TM) 64-Bit Server VM (build 18.0.2.1+1-1, mixed mode, sharing)
```

Ha nem ezt látni, akkor a következő linkről lehet letölteni a megfelelő verziójú Java fejlesztői csomagot. (64-bites Windows platformra):

[https://download.oracle.com/java/18/archive/jdk-18.0.2.1\\_windows-x64\\_bin.exe](https://download.oracle.com/java/18/archive/jdk-18.0.2.1_windows-x64_bin.exe)

Ezután gondoskodni kell arról, hogy a frissen telepített Java hozzá legyen adva a PATH nevű környezeti változóhoz. Ennek az egyik módja a következő parancs lefuttatása parancssorban:

```
set PATH=%PATH%;C:\Program Files\Java\jdk-18.0.2.1\bin
```

Itt a feketével jelölt részt ki kell cserálni a feltelepített Java pontos verziójára.

#### 2. Lépés: Forrásfájlok letöltése



A Herculesre való belépési adatok birtokában letölthető a forrásfájlokat tartalmazó zip file. Ehhez az Eredmények fülre kell navigálni és a Szkeleton beadásánál található file-t kell letölteni.

- **Szkeleton beadása**

[capybara.zip](#):

Ezután a Start menü Terminál alkalmazását kell elindítani, és kiadni a következő parancsot:

```
cd C:\Users\cloud\Downloads
```

Ezután tömörítsük ki a mappát:

```
tar -x -f capybara.zip
```

Ezután navigálunk bele a kitömörített mappába

```
cd capybara\src
```

### 3. Lépés: A program fordítása

Nyissuk meg egy parancssor programot, majd navigálunk el a kitömörített fájlokat tartalmazó mappába. Ezután a következő parancsot futtatva, a program lefordul.

```
javac -d .\output .\*.java .\Model\*.java
```

Linux alatt a '\ helyett a '/' karaktert használjuk.

#### 6.1.3 Futtatás

Windows alatt a következő parancssal futtatható a program:

```
java -cp .\output Main
```

Ha esetleg Linux rendszeren futtatjuk, akkor a '\ helyett a '/' karaktert használjuk.

## 6.2 Értékelés

Tag neve	Tag neptun	Munka százalékban
Andó Viola	H51B9J	20%
Deé-Lukács András Gergely	JHFLXZ	20%
Kiss Blanka Zselyke	C6037J	20%
Skáre Erik	Z7ZF6D	20%
Vörös Vilmos	HW921K	20%

## 6.3 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2023.04.12. 21:00	1,5 óra	Andó Deé-Lukács Kiss Skáre Vörös	<p>Értekezlet.</p> <p>Döntés: Feladatok tagokhoz rendelése:</p> <p>Andó: Element, Source osztályok megírása</p> <p>PlayerSteps2, ConnectPipeToPump, StepOnCisternAndPickUpPipe, BreakPipeAndItAcceptsWater tesztesetek megírása</p> <p>Kiss: Player és Mechanic osztályok megírása, SecondStepOnPump, GetPumpAtCistern, BreakPumpAndTwoStep tesztesetek megírása</p> <p>Deé-Lukács: Pipe network clogs, Working Pump pumps, Pump steps twice after repair, Tester disconnects empty pipe from Pump tesztesetek megírása, Telepítési útmutató megírása</p> <p>PipeEnd, Pool osztályok megírása</p> <p>Skáre: Saboteur és Pipe osztályok megírása, köv. tesztesetek megírása: PlacePump, SourceSteps, SecondPersonTriesToStepOnPipe, DisconnectOutPipeFromPump</p> <p>Vörös: Game, Node, Pump, Cistern, Skeleton osztály megírása, köv. tesztesetek megírása: LeakingPipeRepair, SteppingToAndSwitchPump, Disconnecting Full Pipe, CisternSteps</p>

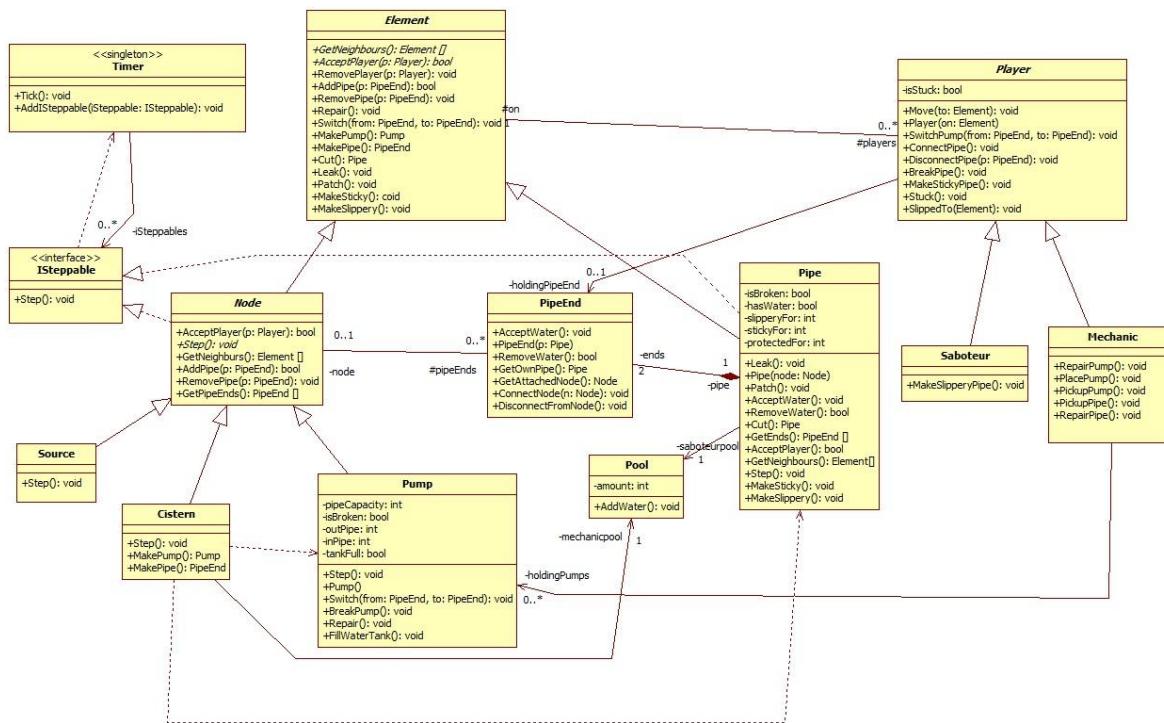
2023.04.13. 16.00	2,5 óra	Vörös	Skeleton osztály megírása, LeakingPipeReapir teszt megírása
2023.04.14. 15.00	3 óra	Kiss	Player és Mechanic osztály megírása
2023.04.14. 17.00	1 óra	Andó	Element és Source osztályok megírása
2023.04.15. 13.00	1,5 óra	Andó	PlayerSteps2, ConnectPipeToPump tesztesetek megírása
2023.04.16. 00.30	2 óra	Skáre	Saboteur és Pipe osztályok megírása, PlacePump elkezdése
2023.04.16 12.00	2 óra	Skáre	PlacePump befejezése és SecondPersonTriesToStepOnPipe
2023.04.16. 14.00	3 óra	Kiss	Tesztesetek megírása: SecondStepOnPump, GetPumpAtCistern, BreakPumpAndTwoStep
2023.04.16. 16.00	5 óra	Deé-Lukács	PipeEnd, Pool osztályok, ClogPipeNetwork, WorkingPumpPumps, StepPumpTwiceAfterRepair, DisconnectEmptyPipeFromPump, kommentezés, hibajavítások, Telepítési útmutató megírása
2023.04.16. 16.00	2 óra	Andó	StepOnCisternAndPickUpPipe, BreakPipeAndItAcceptsWater tesztesetek megírása

2023.04.16. 18.00	4 óra	Vörös	SteppingToAndSwitchPump, DisconnectingFullPipe, CisternSteps tesztek megírása, kommentezés
2023.04.16. 20.00	1 óra	Andó	Kisebb javítások, kommentezés
2023.04.16. 20.00	1 óra	Skáre	SourceSteps és DisconnectOutPipeFromPump
2023.04.16. 21.00	4 óra	Andó Deé-Lukács Kiss Skáre Vörös	Értekezlet Kódok merge-elése, program áttekintése, konfliktusok megoldása

# 7. Protótípus koncepciója

## 7.0 Változás hatása a modellre

### 7.0.1 Módosult osztálydiagram



### 7.0.2 Új vagy megváltozó metódusok

#### Player

- void breakPipe(): kilyukasztja a csövet, amin a játékos áll, ha a cső protectedFor értéke 0, a Saboteur osztályból átkerült ide
- void makeStickyPipe(): ragadóssá teszi a csövet, amin a játékos áll
- void Stuck(): odaragasztja a játékosat ahhoz a csőhöz, amin áll
- void SlippedTo(Element e): átcsúsztatja a játékosat a megadott Elementre

Továbbá az osztályba belekerült egy új attribútum, az isStuck, amely akkor igaz, ha a játékos odaragadt egy csőhöz.

#### Saboteur

- void makeSlipperyPipe(): csúszóssá teszi a csövet, amin a játékos áll

#### Pipe

- void Step(): lépteti a csövet, tehát csökkenti 1-gyel a slipperyFor, a stickyFor és a protectedFor értékét (ha nagyobb 0-nál)
- void MakeSticky(): ragadóssá teszi a csövet
- void MakeSlippery(): csúszóssá teszi a csövet

Emellett az osztályba belekerült 3 új attribútum:

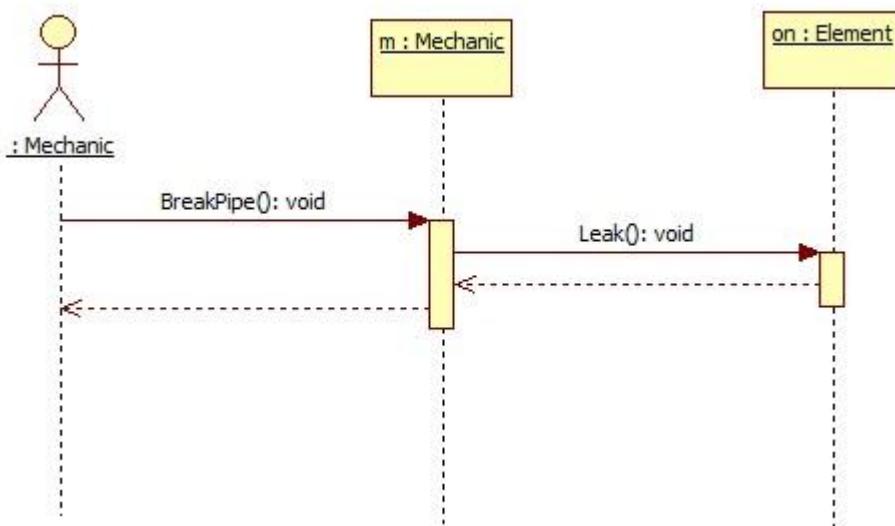
- int slipperyFor: számláló, amely számon tartja az időt, amíg a cső csúszós
- int stickyFor: számláló, amely számon tartja az időt, amíg a cső ragadós
- int protectedFor: számláló, amely számon tartja az időt, amíg a csövet nem lehet kilyukasztani

### Element

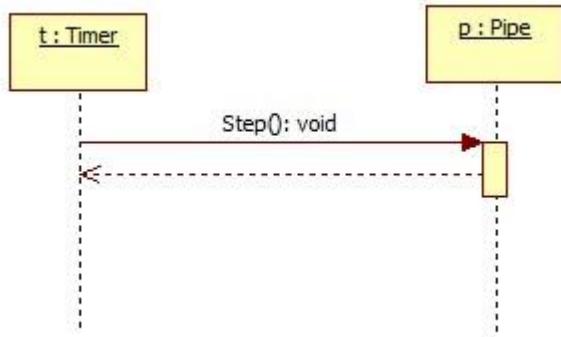
- void MakeSticky(): ragadóssá teszi az elementet, üres
- void MakeSlippery(): csúszóssá teszi az elementet, üres

## 7.0.3 Szekvencia-diagramok

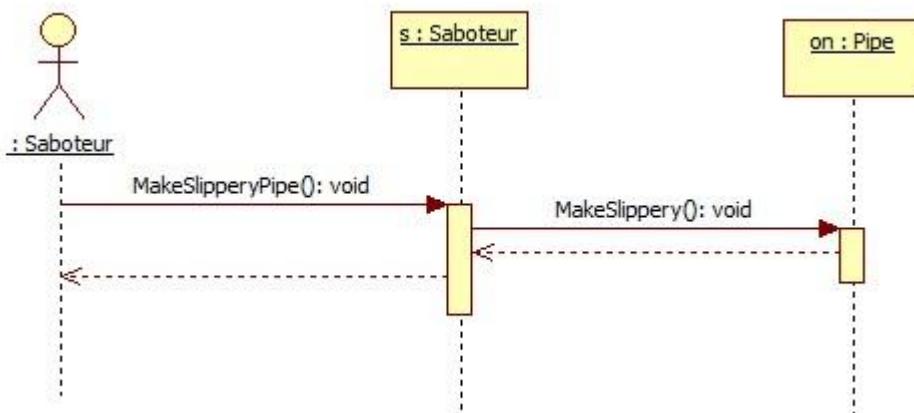
### 7.0.3.1 Mechanic breaks Pipe



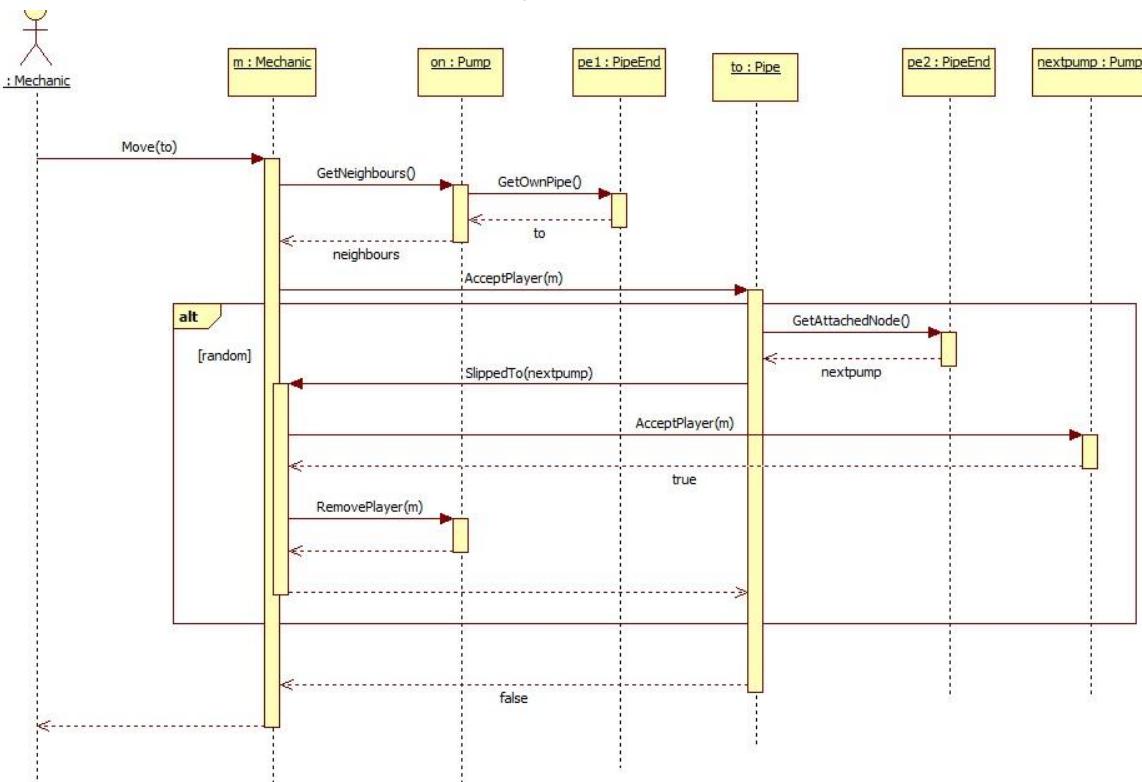
### 7.0.3.2 Pipe steps



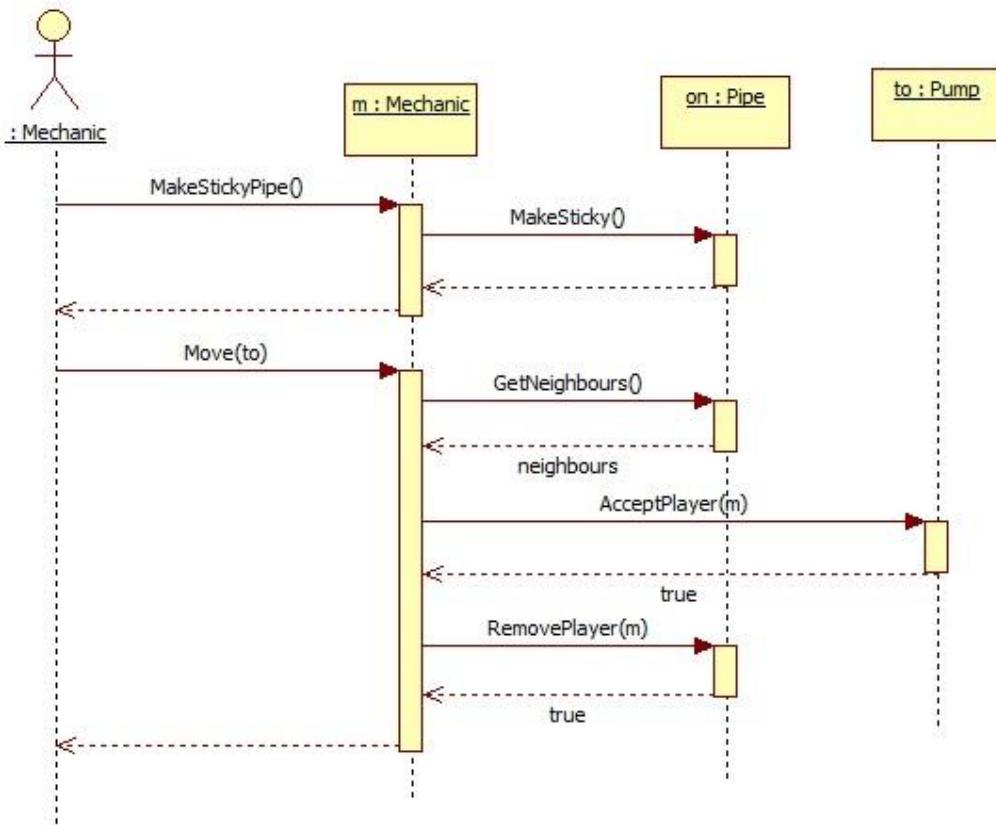
### 7.0.3.3 Saboteur greases Pipe



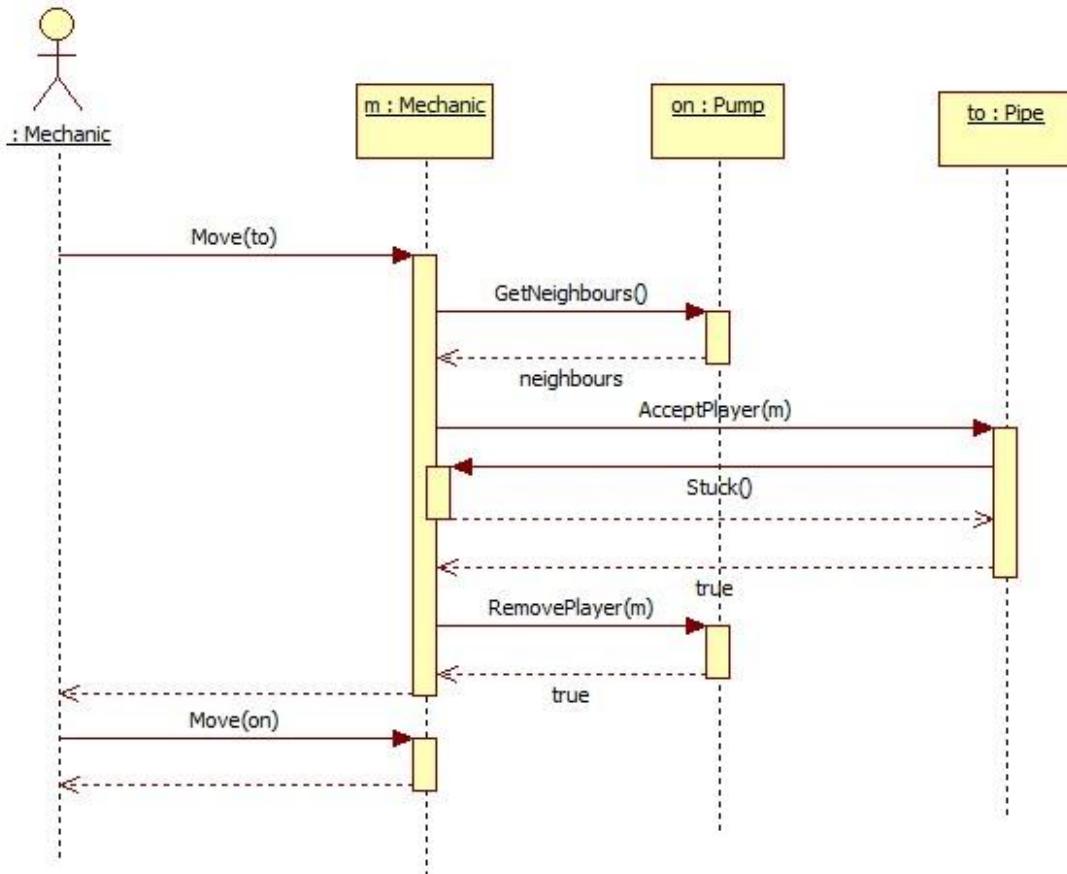
#### 7.0.3.4 Mechanic steps on slippery Pipe



#### 7.0.3.5 Mechanic makes Pipe sticky



### 7.0.3.6 Mechanic steps on sticky Pipe and tries to move again



## 7.1 Prototípus interface-definíciója

### 7.1.1 Az interfész általános leírása

A prototípus program egy parancssori program lesz, vagyis csak a parancssoron keresztül használható/vezérelhető. Grafikus felhasználói felület nem tartozik hozzá.

Fő célja, hogy a felhasználó egysoros parancsokon keresztül létre tudjon hozni egy, a játékban szereplő objektumokból álló modellt, a létrehozott objektumokat össze tudja kapcsolni, valamint az objektumok elvárt működését tudja tesztelni. Ennek megfelelően a program 3 fajta parancsot tartalmaz:

- Objektum alkotó és inicializáló parancsokat (pl. *load*, *add*, *leak*, *fill*)
- Működést/játékmenetet szimuláló parancsokat (pl. *move*, *step*, *break*, *connectpipe*)
- Állapotkérdező parancsot (pl. *state*)

A programnak nem csak egyesével lehet parancsokat megadni. A *load* parancsot használva egy előre létrehozott, a program bemeneti nyelvének megfelelő parancsokat tartalmazó szöveges fájl tartalmát is oda tudjuk adni a programnak, meggyorsítva ezzel az összetett, sok objektumot igénylő működés tesztelését.

## 7.1.2 Bemeneti nyelv

### *add*

**Leírás:** A parancs hozzáad egy új objektumot a hálózathoz

**Opciók:**

- **add <Type> <name>**: A paraméterben megadott típusú elem az adott néven hozzá lesz adva a hálózathoz.

### *break*

**Leírás:** A parancs a már hozzáadott pumpák eltörését teszi lehetővé.

**Opciók:**

- **break <pumpname>**: A paraméterként megadott pumpa eltöríti.

### *connect*

**Leírás:** A parancs a már hozzáadott nodeok összekötését teszi lehetővé már hozzáadott csövek segítségével. A nodeakra és a csőre a hozzáadásukkor meghatározott nevükkel lehet hivatkozni.

**Opciók:**

- **connect <nodename> <pipename>**: Az első paraméterben hivatkozott nodehoz hozzáköti a második paraméterben hivatkozott csövet, ha annak van szabad vége.
- **connect <nodename> <pipename> <nodename>**: Az első paraméterben hivatkozott nodeot és a harmadik paraméterben hivatkozott nodeot összeköti a második paraméterben hivatkozott csővel, ha annak az összes vége szabad.

### *connectpipe*

**Leírás:** A parancs a hivatkozott játékos által tartott csövet hozzáköti ahoz a nodehoz, amin a játékos éppen áll. Ha a játékos kezében nincsen cső, akkor nem történik semmi.

**Opciók:**

- **connectpipe <playernumber>**: A második paraméterben hivatkozott játékosra végrehajta a műveletet.

## ***disconnectpipe***

**Leírás:** A parancs használatával a modellben szereplő, az *add* parancccsal létrehozott játékos tudjuk irányítani, pontosabban egy cső leválasztására késztetni.

**Opciók:**

- **disconnectpipe <playername> <pipename>**: Az első paraméterként megadott játékost irányítjuk, aki az aktuális mezőjén/pozícióján a második paraméterként kapott csövet próbálja leválasztani.

## ***drain***

**Leírás:** A parancs használatával a modellben szereplő, az *add* parancccsal létrehozott csöveknek és pumpáknak a víztartályát tudjuk kiüríteni, amennyiben azok tartalmaznak vizet.

**Opciók:**

- **drain <pipename>**: A paraméterként megadott csövet kiüríti
- **drain <pumpname>**: A paraméterként megadott pumpa víztartályát kiüríti

## ***endround***

**Leírás:** A parancs jelzi a kör végét, és meghívja a Game timer-ének *Tick()* függvényét, tehát léptet minden léptethető objektumot a játékban.

## ***fill***

**Leírás:** A parancs használatával a modellben szereplő, az *add* parancccsal létrehozott csöveknek és pumpáknak a víztartályát tudjuk vízzel feltölteni, amennyiben azok üresek.

**Opciók:**

- **fill <pipename>**: A paraméterként megadott csövet feltölti vízzel
- **fill <pumpname>**: A paraméterként megadott pumpa víztartályát feltölti vízzel.

## ***grease***

**Leírás:** Ez a parancs csúszóssá teszi a megadott csövet.

**Opciók:**

- **grease <pipename>**: A paraméterként megadott nevű csövet csúszóssá teszi.

## ***holdpipe***

**Leírás:** A parancs a megadott nevű szerelő kezébe adja annak a már létrehozott csőnek a végét, aminek a nevét megadjuk paraméterként.

**Opciók:**

- **holdpipe <pipename> <mechanicname>**: A paraméterként megadott szerelő kezébe kerül a megadott nevű cső.

## ***holdpump***

**Leírás:** A parancs a megadott nevű szerelő kezébe adja a megadott nevű pumpát.

**Opciók:**

- **holdpump <pumpame> <mechanicname>**: A paraméterként megadott szerelő kezébe kerül a megadott nevű pumpa.

## ***leak***

**Leírás:** A parancs használatával a modellben szereplő, az *add* parancccsal létrehozott csöveket tudjuk kilyukasztani.

**Opciók:**

- **leak <pipename>**: A paraméterként megadott cső kilyukad.

## ***leakpipe***

**Leírás:** A parancs a csövek kilyukasztására használatos. Amennyiben a hivatkozott játékos éppen egy csövön áll, akkor ez a cső kilyukad. (ha ez a cső nem lyukas)

**Opciók:**

- **leakpipe <playername>**: A második paraméterben hivatkozott játékosra végrehajta a műveletet.

## ***load***

**Leírás:** A parancs egy pálya betöltésére használatos egy fájlból. A fájl a fenti parancsokat tartalmazhatja, a loadnak az eredménye ugyanaz, mintha ezeket a parancsokat sorban futtatnánk le.

**Opciók:**

- **load <file>**: Lefuttatja az első paraméterben hivatkozott fájlból (lehet abszolút és relatív elérési útvonal) megadott parancsokat.

## *move*

**Leírás:** A parancs a játékosok mozgatására alkalmas. Átmozgatja a megadott játékost egy adott elemre.

**Opciók:**

- **move <playername> <elementname>**: Az első paraméterben megadott nevű játékost átmozgatja a második paraméterben megadott nevű elemre.

## *pickup*

**Leírás:** A parancs használatával egy szerelővel kezdeményezünk egy pumpa vagy egy cső felvételét. A parancs hatására a ciszterna a megadott néven létrehoz egy új csövet/pumpát, és a szerelő a kezébe adja, amennyiben egy ciszternán áll éppen. A felvenni kívánt objektum típusát a "pipe", mint cső és "pump", mint pumpa opcionálisan lehet megadni.

**Opciók:**

- **pickup --pipe <pipename> <mechanicname>**: A paraméterként megadott szerelő annál a ciszternánál, amin éppen áll, felvesz egy csövet
- **pickup --pump <pumpname> <mechanicname>**: A paraméterként megadott szerelő annál a ciszternánál, amin éppen áll, felvesz egy pumpát.

## *placepump*

**Leírás:** A parancs egy pumpa letevésére alkalmas. Használatával egy szerelő leteszt egy pumpát arra a csőre, amin áll.

**Opciók:**

- **placepump <mechanicname>**: A paraméterként megadott nevű szerelő leteszi a nála levő pumpát a csőre, amin áll.

## *random*

**Leírás:** Ez a parancs egy csőnek vagy pumpának a megjavítására alkalmas.

**Opciók:**

- **random on**: Véletlen elemek bekapsolása.
- **random off**: Véletlen elemek kikapsolása

## *repair*

**Leírás:** Ez a parancs egy csőnek vagy pumpának a megjavítására alkalmas.

**Opciók:**

- **repair <pipename> <mechanicname>**: Az első paraméterként megadott nevű csövet a második paraméterként megadott nevű szerelő megfoltozza.
- **repair <pumpname> <mechanicname>**: Az első paraméterként megadott nevű pumpát a második paraméterként megadott nevű szerelő megjavítja.

### ***slipperypipe***

**Leírás:** A parancs a már létrehozott csövek csúszóssá tételere használatos. Amennyiben a hivatkozott szabotőr egy csövön áll, úgy ezt a csövet csúszóssá teszi (ha ez a cső nem csúszós éppen).

**Opciók:**

- **slipperypipe <saboturname>**: A paraméterként megadott szabotőrre végrehajtja a műveletet.

### ***state***

**Leírás:** A parancs használatával a modellben szereplő, az *add* parancccsal létrehozott objektumok állapotát tudjuk lekérdezni, ellenőrizni.

**Opciók:**

- **state <mechanicname>**: A paraméterként megadott játékos helyzetét és a nála lévő csőnek és pumpáknak nevét kapjuk meg.
- **state <saboturname>**: A paraméterként megadott játékos helyzetét és a nála lévő csőnek a nevét kapjuk meg.
- **state [-options] <pumpname>**: A paraméterként kapott pumpa állapotát kapjuk meg a kiírt opcióktól függően. Az opciókat egy “-” vagy dash jel után kell felsorolni közvetlenül a state szó után. Ha nem írunk a parancsba egyet sem, akkor a pumpa összes jellemzőjét visszakapjuk.

Az opciók jelentése:

- o: pumpán álló játékosok
- p: pumpához csatlakoztatott csövek
- f: a pumpán beállított bemeneti cső
- t: a pumpán beállított kimeneti cső
- b: a pumpa töröttsége: eltört-e vagy sem
- a: a pumpa átmeneti tárolójának állapota

Például ha csak a pumpára csatlakoztatott csöveket és a ki- és bemeneti csövekre vagyunk kíváncsiak, a következő parancsot adjuk ki: *state -pft pumpname*

- **state [-options] <pipename>**: A paraméterként kapott cső állapotai jelennek meg a megadott opcióktól függően. Az opciókat egy “-” vagy dash jel után kell felsorolni közvetlenül a state szó után. Ha nem írunk a parancsba egyet sem, akkor a cső összes jellemzőjét visszakapjuk.

Az opciók jelentése:

- o: csövön álló játékos
- e: cső két végén lévő csomópontok
- b: a cső állapota: lyukas-e vagy nem
- w: a cső állapota: üres-e vagy teli van vízzel
- p: a cső védett állapotba van-e vagy nem
- s: a cső ragadóssága
- l : a cső csúszóssága

Például ha csak a cső két végén lévő csomópontokra és a vízzel telítettségre vagyunk kíváncsiak, a következő parancsot adjuk ki: *state -ew pipename*

- **state <cisternname>**: A paraméterként kapott ciszterna objektumon álló játékest, valamint a ciszternához csatlakoztatott csöveket kapjuk meg.
- **state <sourcename>**: A paraméterként kapott forrás objektumon álló játékest, valamint a forráshoz csatlakoztatott csöveket kapjuk meg.
- **state pools**: A szabotőrök és szerelők csapata által gyűjtött vízmennyiséget kapjuk meg.

### **step**

**Leírás:** A parancs használatával a modellben szereplő, az *add* parancccsal létrehozott léptethető objektumokat tudjuk léptetni.

**Opciók:**

- **step <pumpname>**: Lépteti a paraméterben megadott pumpát.
- **step <pipename>**: Lépteti a paraméterben megadott csövet.
- **step <sourcename>**: Lépteti a paraméterben megadott forrást.
- **step <cisternname>**: Lépteti a paraméterben megadott ciszternát.

### **sticky**

**Leírás:** A parancs a már hozzáadott cső ragadóssá tételeit valósítja meg. Amennyiben egy ragadós csőre hivatkozva hívjuk meg a parancsot, annak ugyanolyan a hatása, mintha a csőről levensen a ragadást, majd újra ragadóssá tennénk.

**Opciók:**

- **sticky <pipename>**: Ragadóssá teszi az első paraméterben hivatkozott csövet a leírásnak megfelelően.

### **stickypipe**

**Leírás:** A parancs a már létrehozott csövek ragadóssá tételere használatos. Amennyiben a hivatkozott játékos egy csövön áll, úgy ezt a csövet ragadóssá teszi.

**Opciók:**

- **stickypipe <playername>**: A paraméterként megadott játékosra végrehajtja a műveletet.

### ***switch***

**Leírás:** A parancs a pumpák átállítására használatos. Ezzel a parancssal nem a játékos állítja át a pumpát, amin áll, hanem a megadott nevű pumpának lehet a bemenetét és a kimenetét beállítani.

#### **Opciók:**

- **switch <pumpname> <pipename\_from> <pipename\_to>**: Az első paraméterben megadott nevű pumpának fogja beállítani a bemenetét a második paraméterként megadott nevű csőre és a kimenetét a harmadik paraméterként megadott csőre.

### ***switchpump***

**Leírás:** A parancs a pumpák átállítására használatos. Amennyiben a hivatkozott játékos éppen egy pumpán áll, úgy ennek a pumpának beállítja a bemeneti és kimeneti csövét a hivatkozottakra.

#### **Opciók:**

- **switchpump <playername> <pip1> <pip2>**: A pumpának - amin az első paraméter által hivatkozott játékos áll - átállítja a bemeneti csövét a második paraméterben hivatkozott csőre, és a kimeneti csövét a harmadik paraméterben hivatkozott csőre.

### **Komment**

**Leírás:** A “//” - azaz két forward slash - jellel kezdődő sorok kommentnek számítanak, a programot nem befolyásolják, a modellen nincs semmilyen hatásuk. Egyetlen céljuk a prototípus parancsokat tartalmazó szöveges fájlok jobb olvashatósága, értelmezhetősége.

#### **Opciók:**

- **//Ez egy comment: A két kezdőkaraktert bármi követheti (whitespace karakterek is), a sor végén sortörésnek kell szerepelnie.**

### **7.1.3 Kimeneti nyelv**

A prototípus program a megadott parancsokat beolvassa, feldolgozza, és végrehajtja. Visszajelzést a legtöbb eseten csak akkor ad, ha hibás, az elvárt típustól eltérő típusú paramétereket adtunk meg a parancsnak. Ha a paraméterek típusa megegyezett az elvárással, de a modell szabályai, a játék logikája miatt az nem idézett elő változást a modellben, akkor sem kapunk visszajelzést a programtól.

Hibás típusú paraméter megadásánál az alábbi üzenetet kapjuk:

Invalid argument! Please check the correct syntax of the command in the documentation.

Ha olyan objektumok adunk egy parancsnak paraméterként, amit előtte nem hoztunk létre/nem adtunk hozzá az aktuális modellhez, a következő üzenetet kapjuk:

Unknown object! Note that all referred objects are to be added to the running model.

A parancsok közül egyedül a *state* állapotlekérdező parancs ad visszajelzést a lefutását követően. A parancs általános célja az objektumok privát, belső állapotának megismerése. A kimenetek első szavai megfeleltethetőek a lekérdezett objektum egyes mezőinek, a második rész pedig a mezőnek megfelelő értéket mutatja (null érték is elköpzelehető). Paramétertől függően az alábbi kimeneteket adhatja:

- *state <sourcename>*

Output: Players: <playername1> .. <playernameN>  
ConnectedPipes: <pipename1> .. <pipenameN>

- *state <cisternname>*

Output: Players: <playername1> .. <playernameN>  
ConnectedPipes: <pipename1> .. <pipenameN>

- *state -opftba <pumpname>*

Output: Players: <playername1> .. <playernameN>  
ConnectedPipes: <pipename1> .. <pipenameN>  
InPipe: <pipename> [ha nincs kiválasztva, null]  
OutPipe: <pipename> [ha nincs kiválasztva, null]  
isBroken: true/false  
TankFull: true/false

- *state -oebwslp <pipename>*

Output: Player: <playername>  
End1: <nodenname> [szabad vég esetén null]  
End2: <nodenname> [szabad vég esetén null]  
isBroken: true/false  
hasWater: true/false  
isSticky: true/false  
isSlippery: true/false  
isProtected: true/false

- *state pools*

Output: Mechanics: <long>  
Saboteurs: <long>

- *state <saboturname>*  
 Output: on: <elementname>  
     holdingPipe <pipename> [ha nincs a kezében cső, null]
- *state <mechanicname>*  
 Output: on: <elementname>  
     holdingPipe <pipename> [ha nincs a kezében cső, null]  
     holdingPumps: <pumpname1> .. <pumpnameN>

## 7.2 Összes részletes use-case

<b>Use-case neve</b>	Add Object
<b>Rövid leírás</b>	A tesztelő hozzáad egy új objektumot a hálózathoz a megadott típussal és névvel.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja az add &lt;Type&gt; &lt;name&gt; parancsot.</li> <li>2. Létrejön a megadott típusú és nevű objektum, és eltároljuk az objektumok közé.</li> </ol>

<b>Use-case neve</b>	Break pump
<b>Rövid leírás</b>	A tesztelő eltör egy pumpát.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a break &lt;pumpname&gt; parancsot.</li> <li>2. A paraméterként megadott pumpa eltörök.</li> </ol> <p>1a. Ha a pumpa már törött volt, akkor nem történik változás.</p>

<b>Use-case neve</b>	Connect node(s) by pipe
<b>Rövid leírás</b>	A tesztelő összeköthet nodeokat csövekkel.

<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő egy csövet, aminek van legalább egy szabad vége hozzájárható egy nodehoz.</li> <li>2. A tesztelő két nodeot összeköt egy csővel, aminek minden vége szabad.</li> <li>3. A tesztelő olyan csővel akar összekötést végezni, aminek nincs szabad vége, ilyenkor nem történik semmi.</li> </ol>

<b>Use-case neve</b>	Connect player's holding pipe
<b>Rövid leírás</b>	A tesztelő lerakatthatja egy játékossal az általa tartott csövet.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő lerakatja egy olyan játékos csövét, aki ténylegesen fog egy csövet.</li> <li>2. A tesztelő egy olyan játékos csövét próbálja lerakatni, aki nem fog csövet, ilyenkor nem történik változás.</li> </ol>

<b>Use-case neve</b>	Disconnect pipe by player
<b>Rövid leírás</b>	A tesztelő lecsatlakoztat egy játékossal egy csövet.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a <i>disconnectpipe</i> parancsot</li> <li>2. A parancs paraméterében megadott játékos leválasztja a szintén paraméterben megadott csövet a játékos pozíóján, ami lehet: pumpa, ciszterna, forrás            1a: Ha a játékos csövön áll, nem történik semmi            1b: Ha a kijelölt cső nincs bekötve a játékos pozíciójához, nem történik semmi.         </li> </ol>

<b>Use-case neve</b>	Drain Pipe
<b>Rövid leírás</b>	A tesztelő kiüríti a csövet.
<b>Aktorok</b>	Tester

<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a drain &lt;pipename&gt; parancsot.</li> <li>2. A csóból kiürül a víz.</li> </ol> <p>1a: Ha a cső üres volt, nem történik semmi.</p>
---------------------	--

<b>Use-case neve</b>	Drain Pump
<b>Rövid leírás</b>	A tesztelő kiüríti a pumpa víztartályát.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a drain &lt;pumpname&gt; parancsot.</li> <li>2. A pumpa víztartálya kiürül.</li> </ol> <p>1a: Ha a tartály üres volt, nem történik semmi.</p>

<b>Use-case neve</b>	Fill pipe
<b>Rövid leírás</b>	A tesztelő megtölt vízzel egy csövet.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a fill &lt;pipename&gt; parancsot.</li> <li>2. A paraméterként megadott cső megtelik vízzel.</li> </ol> <p>1a. Ha a cső már teli volt vízzel, nem történik változás.</p>

<b>Use-case neve</b>	Fill pump
<b>Rövid leírás</b>	A tesztelő megtölti vízzel egy pumpa víztartályát.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a fill &lt;pumpname&gt; parancsot.</li> <li>2. A paraméterként megadott pumpa víztartálya megtelik vízzel.</li> </ol> <p>1a. Ha a tartály már teli volt vízzel, nem történik változás.</p>

<b>Use-case neve</b>	Grease pipe
<b>Rövid leírás</b>	A tesztelő csúszóssá tehet egy csövet.
<b>Aktorok</b>	Tester

<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő egy nem csúszós csövet csúszóssá tesz.</li> <li>2. A tesztelő egy csúszós csövet újra ragadóssá tesz, ilyenkor olyan, mintha csak a második csúszóssá tétel történt volna meg.</li> </ol>
---------------------	--

<b>Use-case neve</b>	Leak
<b>Rövid leírás</b>	A tesztelő kilyukaszt egy csövet
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a <i>leak</i> parancsot</li> <li>2. A paraméterként megadott cső kilyukad.</li> </ol> <p>1a: Ha a cső már lyukas volt, nem történik változás.</p>

<b>Use-case neve</b>	Leak pipe by player
<b>Rövid leírás</b>	A tesztelő kilyukasztatja egy játékossal azt a csövet, amin a játékos éppen áll.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kilyukasztatja egy játékossal azt a csövet, amin a játékos éppen áll.</li> <li>2. A tesztelő megpróbálja kilyukasztatni a játékos csövét, de ha a játékos nem is csövön áll, ilyenkor nem történik semmi.</li> </ol>

<b>Use-case neve</b>	Load
<b>Rövid leírás</b>	A tesztelő betölthet egy pályát egy fájlból. (azaz futtatónak a fájlban megadott parancsok)
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő betölti a megadott fájlból a pályát, azaz futtatónak a parancsok.</li> <li>2. A tesztelő helytelen elérési útvonallal próbálkozik, ekkor hiba keletkezik.</li> </ol>

<b>Use-case neve</b>	Make pipe sticky
<b>Rövid leírás</b>	A tesztelő ragadóssá tehet egy csövet.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő egy nem ragadós csövet ragadóssá tesz.</li> <li>2. A tesztelő egy ragadós csövet újra ragadóssá tesz, ilyenkor olyan, mintha csak a második ragadóssá tétel történt volna meg.</li> </ol>

<b>Use-case neve</b>	Move player to Element
<b>Rövid leírás</b>	A tesztelő átmozgat egy játékost egy elemre
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a <i>move</i> parancsot.</li> <li>2. A játékos áthelyeződik a megadott elemre arról, amin volt.</li> </ol>

<b>Use-case neve</b>	Pick up pipe by mechanic
<b>Rövid leírás</b>	A tesztelő felvetet egy szerelővel egy csövet annál a ciszternánál, ahol éppen áll.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a <i>pickup &lt;pipename&gt; &lt;mechanicname&gt;</i> parancsot.</li> <li>2. A paraméterként megadott szerelővel felveteti a paraméterként megadott csövet, majd ennek a szabad vége lesz szerelő holdingPipeEnd-je.             <ol style="list-style-type: none"> <li>1a. Ha a szerelő nem ciszternán áll, akkor nem történik semmi.</li> <li>1b. Ha már van a szerelőnek holdingPipeEnd-je, akkor nem történik semmi.</li> </ol> </li> </ol>

<b>Use-case neve</b>	Pick up pump by mechanic
----------------------	--------------------------

<b>Rövid leírás</b>	A tesztelő felvetet egy szerelővel egy pumpát annál a ciszternánál, ahol éppen áll.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a pickup &lt;pumpname&gt; &lt;mechanic&gt; parancsot.</li> <li>2. A paraméterként megadott szerelővel felveteti a paraméterként megadott pumpát, majd hozzáadja a szerelő eddig felvett pumpáihoz.</li> </ol> <p>1a. Ha a szerelő nem ciszternán áll, akkor nem történik semmi.</p>

<b>Use-case neve</b>	Place pump
<b>Rövid leírás</b>	A tesztelő egy szerelővel letetet egy pumpát arra a csőre, amin a szerelő áll.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő letetet egy pumpát egy szerelővel a csőre, amin a szerelő áll.</li> <li>2. A szerelő megpróbál letenni egy pumpát, de nem csövön áll, így nem történik semmi.</li> </ol>

<b>Use-case neve</b>	Random
<b>Rövid leírás</b>	A tesztelő ki- és bekapcsolja a véletlenszerűséget
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a <i>random on/off</i> parancsot</li> <li>2. A modell működésében megváltozik a véletlen szerepe</li> </ol>

<b>Use-case neve</b>	Repair
<b>Rövid leírás</b>	A tesztelő megjavítat egy csövet vagy pumpát egy szerelővel
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő megjavítat egy csövet egy szerelővel</li> </ol>

	2. A tesztelő megjavítat egy pumpát egy szerelővel
--	--

<b>Use-case neve</b>	Slippery pipe by saboteur
<b>Rövid leírás</b>	A tesztelő csúszóssá teteti egy szabotőrrel azt a csövet, amin a szabotőr éppen áll.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<p>1. A tesztelő kiadja a slipperypipe &lt;saboturname&gt; parancsot.</p> <p>2. A tesztelő a paraméterként megadott szabotőrrel csúszóssá teteti azt csövet, ahol az éppen tartózkodik, ha az éppen nem csúszós.</p> <p>1a. Ha a szabotőr nem csöön áll, akkor nem történik semmi.</p>

<b>Use-case neve</b>	State query
<b>Rövid leírás</b>	A tesztelő lekérdezi egy adott objektum állapotát.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<p>1. A tesztelő kiadja a state parancsot.</p> <p>2. A parancs paraméterében kijelölt objektum állapota megjelenik a konzolon (Lásd: Kimeneti nyelv)</p>

<b>Use-case neve</b>	Step Cistern
<b>Rövid leírás</b>	A tesztelő lépteti a ciszternát
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<p>1. A tesztelő kiadja a step &lt;cisternname&gt; parancsot.</p> <p>2. A Ciszterna lép, vagyis elnyeli a beleérkező vizet.</p>

<b>Use-case neve</b>	Step Pipe
<b>Rövid leírás</b>	A tesztelő lépteti a csövet.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<p>1. A tesztelő kiadja a step &lt;pipename&gt; parancsot.</p>

	2. A Cső lép.
--	---------------

<b>Use-case neve</b>	Step Pump
<b>Rövid leírás</b>	A tesztelő lépteti a pumpát.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a step &lt;pumpname&gt; parancsot.</li> <li>2. A Pumpa lép, vagyis vizet pumpál, ha nem üres a víztartálya.</li> </ol>

<b>Use-case neve</b>	Step Source
<b>Rövid leírás</b>	A tesztelő lépteti a forrást.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a step &lt;sourcename&gt; parancsot.</li> <li>2. A Forrás lép, vagyis vízzel tölti meg a belekötött csöveket.</li> </ol>

<b>Use-case neve</b>	Sticky pipe by player
<b>Rövid leírás</b>	A tesztelő ragadóssá teteti egy játékossal azt a csövet, amin az éppen áll.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a stickypipe &lt;playername&gt; parancsot.</li> <li>2. A tesztelő a paraméterként megadott játékossal ragadóssá teteti azt csövet, ahol az éppen tartózkodik, ha az éppen nem ragadós.</li> </ol> <p>1a. Ha a játékos nem csövön áll, akkor nem történik semmi.</p> <p>1b. Ha a cső már ragadós volt, akkor a parancs kiadása után csak a második ragadássá tétele számít.</p>

<b>Use-case neve</b>	Switch pump
----------------------	-------------

<b>Rövid leírás</b>	A tesztelő átállít egy pumpát.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő kiadja a <i>switch</i> parancsot.</li> <li>2. A tesztelő átállítja a parancsnak megadott nevű pumpát.</li> </ol>

<b>Use-case neve</b>	Switch pump by player
<b>Rövid leírás</b>	A tesztelő átállítatja egy játékossal azt a pumpát, amin a játékos éppen áll.
<b>Aktorok</b>	Tester
<b>Forgatókönyv</b>	<ol style="list-style-type: none"> <li>1. A tesztelő átállítatja egy játékossal azt a pumpát, amin a játékos éppen áll.</li> <li>2. A tesztelő megpróbálja átállítatni egy játékossal a pumpáját, de a játékos nem is pumpán áll, ilyenkor nem történik semmi.</li> </ol>

### 7.3 Tesztelési terv

[A tesztelési tervben definiálni kell, hogy a be- és kimeneti fájlok egybevetésével miként végezhető el a program tesztelése. Meg kell adni magas szintű teszt forgatókönyveket. Az egyes teszteket elég informálisan, szabad szövegként leírni, tesztesetenként egy-öt mondatban. minden teszthez meg kell adni, hogy mi a célja, a proto mely funkcionálitását, osztályait stb. teszteli. Az alábbi táblázat minden teszt-esethez külön-külön elkészítendő.]

<b>Teszteset neve</b>	Broken, empty pump steps
<b>Rövid leírás</b>	A tesztelő létrehoz egy pumpát, eltöri, majd hozzácsatlakozat két csövet csövet, melyeket beállít a pumpán ki- és bemeneti csőként. Ezután a bemeneti csövet feltölti vízzel. Legvégül lépteti a pumpát.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodni arról, hogy a pumpa törött állapotban nem pumpál</li> <li>- Megbizonyosodni arról, hogy a kimeneti csőbe nem jutott víz, a bemeneti csőben megmaradt a víz.</li> </ul>

<b>Teszteset neve</b>	Cistern steps
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amely egy ciszternát és 2 hozzákapcsolt csövet tartalmaz, ezután az első csőbe vizet rak. Ekkor lépteti a ciszternát.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy az első csőből a ciszterna eltávolítja a vizet, a második csőből megpróbál, de nem tud vizet eltávolítani.</li> <li>- Megbizonyosodás arról, hogy a szerelők pool-jához adódik víz és kapnak pontot.</li> </ul>

<b>Teszteset neve</b>	Leaking pipe gets water
<b>Rövid leírás</b>	A tesztelő létrehoz egy csövet, amelyet rögtön ki is lyukaszt. Ezután vizet vezet a csőbe. Ennek ki kell folynia a szabotőrök gyűjtőhelyére.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy a kilyukasztott csőbe folyatott víz a megfelelő helyre kerül.</li> <li>- Megbizonyosodás arról, hogy a szabotőrök pontszáma növekedik-e, ha egy csövön kifolyik a víz.</li> </ul>

<b>Teszteset neve</b>	Mechanic repairs broken Pump
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amely egy pumpát és 2 hozzákapcsolt csövet tartalmaz, ezután beállítja a pumpa bemeneti- és kimeneti csövét az 1. és 2. csőre, az 1. csövet feltölti vízzel, valamint elrontja a pumpát. Ezután megjavítja a pumpát és lépteti kétszer.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy a pumpa javítás után helyesen működik, azaz első léptetésnél kiszívja az 1. csőből a vizet, a második léptetésnél pedig a 2. csövet megtölți.</li> </ul>

<b>Teszteset neve</b>	Mechanic picks up Pump at Source
-----------------------	----------------------------------

<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amely egy forrást tartalmaz. Ezután létrehoz egy szerelőt és rálépteti a forrásra. Itt a játékos megpróbál felvenni egy pumpát.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy a játékos nem tud felvenni pumpát a forrásnál.</li> </ul>

<b>Teszteset neve</b>	Mechanic places Pump on Cistern
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amely egy ciszternát tartalmaz. Ezután létrehoz egy szerelőt és rálépteti a ciszternára. Ekkor a szerelő megpróbál letenni egy pumpát a ciszternára.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy a játékos nem tud letenni pumpát a ciszternára.</li> </ul>

<b>Teszteset neve</b>	Mechanic places Pump on Source
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amely egy forrást tartalmaz. Ezután létrehoz egy szerelőt és rálépteti a forrásra. Ekkor a szerelő megpróbál letenni egy pumpát a forrásra.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy a játékos nem tud letenni pumpát a forrásra.</li> </ul>

<b>Teszteset neve</b>	Mechanic repairs leaking pipe
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amin csak egy cső van és egy játékos, amelyet rálépett a csőre. Kilyukasztja a csövet, majd a játékossal megjavítja a csövet. Ezután leteszeli a cső állapotát.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás, hogy a cső megjavítása helyes-e, azaz, hogy a megjavítás után a cső az elvárt állapotba kerül-e.</li> </ul>

<b>Teszteset neve</b>	Mechanic repairs undamaged pipe
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amin csak egy cső van és egy játékos, amelyet ráléptet a csőre, (amely ép állapotban van). Ezután megjavítatja a játékossal a csövet, majd leteszeli a cső állapotát.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás, arról, hogy a cső állapota nem változott, azaz az ép csőre a javítás nincs hatással.</li> </ul>

<b>Teszteset neve</b>	Mechanic repairs working Pump
<b>Rövid leírás</b>	A tesztelő létrehoz egy pumpát, ami működőképes állapotban van, majd megpróbálja megszerelni.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodni arról, hogy a pumpa állapota nem változott.</li> </ul>

<b>Teszteset neve</b>	Mechanic picks up Pipe at Cistern
<b>Rövid leírás</b>	A tesztelő létrehoz egy ciszternát és egy játékest, amelyet ráléptet a ciszternára. Ezután játékossal felvetet egy csövet a ciszternánál. Ezután leteszeli, hogy létrejött-e az új cső, és ez a játékos kezében van-e.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás, hogy a ciszterna képes-e a csőgyártásra.</li> <li>- Megbizonyosodás, hogy az újonnan gyártott cső az ezt kérő játékos kezébe kerül-e.</li> </ul>

<b>Teszteset neve</b>	Mechanic picks up Pipe at Source
<b>Rövid leírás</b>	A tesztelő létrehoz egy forrást és egy játékest, amelyet ráléptet a forrásra. Ezután játékossal felvetet egy csövet a forrásnál. Ezután leteszeli, hogy létrejött-e az új cső, és ez a játékos kezében van-e.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodni arról, hogy a forrás nem képes a csőgyártásra.</li> </ul>

	<ul style="list-style-type: none"> <li>- Megbizonyosodás, hogy a játékos kezébe nem került cső.</li> </ul>
--	--

<b>Teszteset neve</b>	Mechanic picks up Pump at Cistern
<b>Rövid leírás</b>	A tesztelő létrehoz egy ciszternát és egy játékost, amelyet ráléptet a ciszternára. Ezután játékossal felvetet egy pumpát a ciszternánál. Ezután leteszteli, hogy létrejött-e az új pumpa, és ez a játékosnál van-e.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás, hogy a ciszterna képes-e pumpát gyártani.</li> <li>- Megbizonyosodás, hogy az újonnan gyártott pumpa a játékoshoz kerül-e.</li> </ul>

<b>Teszteset neve</b>	Mechanic places pump on pipe
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, egy ciszternával, egy pumpával és az ezeket összekötő csővel. Létrehoz egy játékost, akit ráléptet a ciszternára, ahol felvetet vele egy új pumpát. Ezután rálépteti a csőre, ahol letetei vele az előbb felvett pumpát, ami sikerülni fog.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy a ciszterna képes-e pumpa gyártására.</li> <li>- Megbizonyosodás arról, hogy az újonnan legyártott pumpa a játékoshoz kerül-e.</li> <li>- Megbizonyosodás arról, hogy az új pumpát le lehet rakni csövön.</li> </ul>

<b>Teszteset neve</b>	Mechanic places Pump on Pump
<b>Rövid leírás</b>	A tesztelő létrehoz egy ciszternát és egy pumpát, amelyeket összeköt egy csővel. A játékost rálépteti a ciszternára és egy csövet vetet fel vele. Ezután átlépteti a játékost a csövön keresztül a pumpára és megpróbálja lerakni a felvett pumpát. Ennek hatására semmi nem történik.

<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy a ciszterna képes-e pumpa gyártására.</li> <li>- Megbizonyosodás arról, hogy az újonnan legyártott pumpa a játékoshoz kerül-e.</li> <li>- Megbizonyosodás arról, hogy az újonnan létrehozott pumpát nem lehet lerakni egy nodeon.</li> </ul>
--------------------	---

<b>Teszteset neve</b>	Mechanic repairs broken pump and it steps 2
<b>Rövid leírás</b>	A tesztelő létrehoz egy pumpát, amelyet eltör. Ezenkívül létrehoz egy szerelőt, akit rálépett a pumpára, majd megjavítja vele azt. Továbbá a pumpához hozzácsatlakoztat két csövet, beállítja a vízáramlás irányát, és feltölti vízzel a bemeneti csövet. Ezután lépteti kétszer a pumpát.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy a pumpa állapota megváltozott nem töröttrre.</li> <li>- Megbizonyosodás a pumpa léptetésének helyességéről, azaz először a bemeneti csőből a tartályba, majd onnan a kimeneti csőbe kerül a víz.</li> </ul>

<b>Teszteset neve</b>	Pipe network clogs
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amely egy pumpát és 2 hozzákapcsolt csövet tartalmaz, ezután beállítja a pumpa bemeneti- és kimeneti csövét az 1. és 2. csőre, valamint az első csőbe vizet rak. Ezután folyamatosan tölti vízzel az első csövet és lépteti a pumpát.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy mivel a víz a második csőből nincs hova tovább folyjon, egy idő után nem lehet vizet tölteni az első csőbe, mivel a hálózat tele van.</li> </ul>

<b>Teszteset neve</b>	Player switches pump and steps it
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amely egy pumpát és 4 hozzákapcsolt csövet tartalmaz, ezután beállítja a pumpa bemeneti- és kimeneti csövét az 1. és 2. csőre, valamint a

	3. csőbe vizet rak. Ezután átállítja a pumpa bemeneti- és kimeneti csövét a 3. és a 4. csőre és lépteti a pumpát.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás a pumpa átállításának helyességéről.</li> <li>- Megbizonyosodás a pumpa léptetésének helyességéről (léptetésnél a bemeneti csövén van víz)</li> <li>- Megbizonyosodás, hogy a csőből a víz eltávolítás helyes-e.</li> </ul>

<b>Teszteset neve</b>	Player breaks Pipe
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, ami egy játékost, egy pumpát és egy hozzákötött csövet tartalmaz. Beállítja a pumpa kimeneti csövének a csövet és megtölti a pumpa víztartályát. Rálépteti a csőre a játékost, majd megpróbálja kilyukasztani a csövet, ami sikerülni fog. Ezután lépteti egyet a pumpát, így bele fogja pumpálni a vizet a csőbe, de a csőből ki fog folyni a víz.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy csövet ki lehet kilyukasztani.</li> <li>- Megbizonyosodás arról, hogy lyukas csőből kifolyik a víz, tehát a léptetés után a cső üres.</li> <li>- Megbizonyosodás arról, hogy a kifolyt víz növeli a saboteurPool-ban levő mennyiséget</li> </ul>

<b>Teszteset neve</b>	Player connects Pipe to Source
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amely egy csövet és egy forrást tartalmaz. Ezután megpróbálja a csövet összekapcsolni a forrással. Ekkor lépteti a forrást.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy a cső és a forrás sikeresen összekapcsolódott.</li> <li>- Megbizonyosodás arról, hogy a léptetés után a csőbe került víz.</li> </ul>

<b>Teszteset neve</b>	Player connects pipe to Cistern
-----------------------	---------------------------------

<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, ami egy játékost, egy ciszternát és egy csövet tartalmaz. A játékost rálépteti a ciszternára és a cső egyik végét odaadja a játékosnak. Ezután megpróbálja a tartott csővéget felcsatlakoztatni a ciszternára, ami sikerülni fog.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy csövet lehet ciszternához csatlakoztatni.</li> </ul>

<b>Teszteset neve</b>	Player connects pipe to Pipe
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amely két csövet tartalmaz és egy játékost. A játékos fogja az egyik csövet, a másikon pedig áll. A tesztelő megpróbálja lerakatni a csövet a játékossal, amely nem fog sikerülni, mert nem nodeon áll.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás a cső megfogásának helyességéről.</li> <li>- Megbizonyosodás, hogy a csövet nem lehet Pipe-on lerakni.</li> </ul>

<b>Teszteset neve</b>	Player connects pipe to pump
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, ami egy játékost, egy pumpát és egy csövet tartalmaz. A játékost rálépteti a pumpára és a cső egyik végét odaadja a játékosnak. Ezután megpróbálja a tartott csővéget felcsatlakoztatni a pumpára, ami sikerülni fog, ha a pumpa csőkapacitása ezt megengedi.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás a cső megfogásának helyességéről.</li> <li>- Megbizonyosodás, arról, hogy csövet lehet pumpához csatlakoztatni, ha azon még van szabad hely.</li> </ul>

<b>Teszteset neve</b>	Player disconnects empty Pipe
-----------------------	-------------------------------

<b>Rövid leírás</b>	A tesztelő létrehoz egy pumpát, hozzácsatlakoztat üres egy csövet. Végül leválasztja a csövet a pumpáról
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodni arról, hogy a cső leválasztásánál nem folyt el víz</li> </ul>

<b>Teszteset neve</b>	Player disconnects full pipe
<b>Rövid leírás</b>	A tesztelő létrehoz egy pumpát, hozzácsatlakoztat egy csövet, amelyet megtölt vízzel, majd kiválasztja azt kimeneti csőként. Ezenkívül létrehoz egy játékost, akit a pumpára léptet. Végül leválasztja a csövet a pumpáról és a kifolyt víz mennyiséget hozzáadja a szabotőrök medencéjéhez.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodni arról, hogy a cső leválasztásánál kifolyik a víz</li> <li>- Megbizonyosodni arról, hogy a szabotőrök pontot kapnak</li> <li>- Megbizonyosodni arról, hogy a lekötött csővég a játékos kezében marad</li> </ul>

<b>Teszteset neve</b>	Player disconnects pump's output
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amely egy pumpát és két hozzákapcsolódó csövet tartalmaz. Ezután az első csőbe vizet tölt, majd lépteti a pumpát. Ekkor lecsatlakoztatja a pumpa kimeneti csövét, vizet tölt a bemeneti csőbe és megint lépteti a pumpát.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy a pumpa ilyenkor nem tud pumpálni vizet és nem szív a bemenetről sem.</li> </ul>

<b>Teszteset neve</b>	Player leaks out pipe and pump steps one
<b>Rövid leírás</b>	A tesztelő létrehoz egy pumpa és cső sorozatot, amely így néz ki: 1.CS - 1.P - 2.CS - 2.P - 3.CS,

	<p>ahol a "CS" a csövet, a "P" a pumpát, a "-" a köztük lévő összeköttetést jelenti.</p> <p>Ezután az 1. csőbe vizet rak és létrehoz egy játékoszt, amely a 2. csövön áll és kilyukasztja azt, majd feltölti az 1. pumpa víztartályát.</p> <p>Ezután lépteti az 1. és 2. pumpát ilyen sorrendben.</p> <p>Az elvárt eredmény az, hogy a 2. pumpa víztartálya nem töltődik fel és a 3. csőben se lesz víz, mivel a 2. csőből kifolyt a víz.</p>
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás arról, hogy egy pumpa bemeneti csövének kilyukasztása megfelelő hatással van a pumpa viselkedésére.</li> <li>- Megbizonyosodás arról, hogy egy olyan pumpa, amelynek tele van a víztartálya és van a bemeneti csövén víz pumpál-e a kimeneti csövébe.</li> </ul>

<b>Teszteset neve</b>	Player moves to slippery pipe from pump
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, ami egy játékoszt, egy pumpát és egy csövet tartalmaz. A játékoszt a pumpára lépteti, a csövet, amit a pumpához kötött, pedig csúszóssá teszi. Ezután a játékoszt megpróbálja ráléptetni a csúszós csőre, azonban ez nem fog sikerülni, mivel annak egy szomszédjára kerül véletlenszerűen.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodni arról, hogy a játékos valóban a csúszós cső valamelyik szomszédjára került a lépése után.</li> </ul>

<b>Teszteset neve</b>	Player moves to sticky and tries to move again
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, ami egy játékoszt, két pumpát és egy közéjük bekötött csövet tartalmaz. A játékoszt az első pumpára lépteti, a csövet pedig csúszóssá teszi. Ezután a játékoszt megpróbálja ráléptetni a ragadós csőre, ami sikerülni fog. Utána megpróbálja a csőről a másik pumpára mozgatni a játékoszt, de ez nem fog sikerülni, mivel a játékos odaragadt a csőhöz.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodni arról, hogy a játékos nem tud azonnal továbbhaladni a ragadós csőről.</li> </ul>

<b>Teszteset neve</b>	Player switches pump and steps it
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amely egy pumpát és 4 hozzákapcsolt csövet tartalmaz, ezután beállítja a pumpa bemeneti- és kimeneti csövét az 1. és 2. csőre, valamint a 3. csőbe vizet rak. Ezután átállítja a pumpa bemeneti- és kimeneti csövét a 3. és a 4. csőre és lépteti a pumpát.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás a pumpa átállításának helyességéről.</li> <li>- Megbizonyosodás a pumpa léptetésének helyességéről (léptetésnél a bemeneti csövén van víz)</li> <li>- Megbizonyosodás, hogy a csőből a víz eltávolítás helyes-e.</li> </ul>

<b>Teszteset neve</b>	Pump breaks with full tank and it steps 2
<b>Rövid leírás</b>	A tesztelő létrehoz egy pumpát, amelynek a víztartályát megtöltivízzel és hozzácsatlakoztat 2 csövet. Ezután beállítja a vízáramlás irányát, a bemeneti csövet megtöltivízzel, eltöríti a pumpát, majd lépteti kétszer.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodni arról, hogy kezdetben a víztartályban valóban van víz.</li> <li>- Megbizonyosodni arról, hogy a pumpa valóban törött állapotba kerül.</li> <li>- Megbizonyosodni arról, hogy ha eltöríti a pumpa, akkor annak teli víztartályából még kifolyik a víz a kimeneti csőbe.</li> <li>- Megbizonyosodni arról, hogy törött pumpa nem tud vizet szívni a bemeneti csövéből.</li> </ul>

<b>Teszteset neve</b>	Pump steps
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amely egy pumpát és 2 hozzákapcsolt csövet tartalmaz, ezután beállítja a pumpa bemeneti- és kimeneti csövét az 1. és a 2. csőre, valamint az 1. csőbe vizet rak. Ezután lépteti a pumpát.

<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás a pumpa léptetésének helyességéről (léptetésnél a bemeneti csövén van víz)</li> <li>- Megbizonyosodás, hogy a csőből a víz eltávolítás helyes-e.</li> </ul>
--------------------	--

<b>Teszteset neve</b>	Pump with empty tank steps 2
<b>Rövid leírás</b>	A tesztelő létrehoz egy pumpát, hozzácsatlakoztat két csövet, beállítja a vízáramalás irányát, majd a bemeneti csövet megtöltivízzel. Ezután lépteti kettőt a pumpát.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodni arról, hogy a pumpa először az átmeneti tárolóját tölti tele</li> <li>- Megbizonyosodni arról, hogy teli tárolóval a pumpa eljuttatja a vizet a kimeneti csőbe.</li> </ul>

<b>Teszteset neve</b>	Source steps
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, amely egy forrást és két hozzákötött üres, nem lyukas csövet tartalmaz. Ezután lépteti a forrást.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodás a forrás léptetésének helyességéről, tehát hogy a léptetés után minden két hozzákötött cső tartalmaz vizet.</li> </ul>

<b>Teszteset neve</b>	Undamaged pipe gets water
<b>Rövid leírás</b>	A tesztelő létrehoz egy pályát, egy pumpával és egy (ép, üres) csövel, amit hozzáköt a pumpához. Ezután lépteti a pumpát (a kimeneti cső az előbb létrehozott cső lesz), amely a kimeneti csövébe pumpálja a vizet, amely pedig tudja fogadni azt.
<b>Teszt célja</b>	<ul style="list-style-type: none"> <li>- Megbizonyosodni arról, hogy a cső valóban tudja-e fogadni a vizet, azaz üres állapotból teli állapotba kerül.</li> </ul>

## 7.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása

A tesztelést támogató segédprogram egy PowerShell szkript lesz. Feladata az lesz, hogy az előre megírt teszteseteket végrehajtsa a prototípus programon és összehasonlítsa az eredményt a teszteset előre definiált, elvárt eredményével.

A segédprogram a Rунтест nevű szkript lesz. minden tesztesetnek létrehozunk egy bemeneti fájl, ami a tesztesethez szükséges prototípus program parancsait tartalmazza. Ugyanígy minden tesztesethez létrehozunk egy kimeneti fájlt, ami a prototípus parancsok kiadása után elvárt kimenetet tartalmazza. Elvárás, hogy ezek fájlok a szkriptet tartalmazó mappában legyenek elhelyezve. A szkript ezeket olvassa be futása során és használja a prototípus program bemenetén, illetve a prototípus program kimenetének ellenőrzésekor.

A tesztelőnek két opciója van: vagy az összes, előre megírt tesztesetet végrehajtja, vagy 1 db tesztesetet hajt csak végre. Mivel a tesztesetek számozva vannak, a szkript futtatásakor argumentumként csak a teszteset számát kell megadni. Például ha a kettes számú tesztet akarjuk végrehajtani:

```
Rунтест.ps1 2
```

Ha az összes tesztesetet akarjuk lefuttatni, nem kell plusz argumentum:

```
Rунтест.ps1
```

A szkript futtatásához ideiglenesen meg kell változtatni a végrehajtási házirendet géünkön, hogy digitális aláírás nélkül is futtathassuk a szkriptet.

A segédprogram konzolos üzenetben értesíti a tesztelőt, hogy a lefuttatott tesztek közül melyik volt sikeres, melyik nem. A sikertelenül lefutott teszteseteknél jelzi, az aktuális kimenet mely soraiban van eltérés az elvárt kimenethez képest.

## 7.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
--------	-----------	------------	--------

2023. 04. 20. 20:00	2,5 óra	Andó Deé-Lukács Kiss Skáre Vörös	<p>Értekezlet.</p> <p>Döntés: Részfeladatok felosztása, változatások, koncepció megbeszélése</p> <p>Andó:</p> <p>Parancsok és az ezekhez tartozó use-casek: fill, break, pickup, slipperypipe, stickypipe</p> <p>Tesztetek:</p> <ul style="list-style-type: none"> <li>-Player moves to slippery pipe from pump</li> <li>-Player connects pipe to pump</li> <li>-Player disconnects full pipe</li> <li>-Mechanic repairs undamaged pipe</li> <li>-Mechanic places pump on pipe</li> <li>-Undamaged pipe gets water</li> <li>-Mechanic repairs broken pump and it steps 2</li> <li>-Pump breaks with full tank and it steps 2</li> </ul> <p>Kiss:</p> <p>osztálydiagram, megváltozott metódusok szekvenciadiagramok: 7.0.3.1-3</p> <p>Prototípus parancsainak leírása:</p> <ul style="list-style-type: none"> <li>-add &lt;Type&gt; &lt;name&gt;</li> <li>-drain &lt;pipename&gt;   &lt;pumpname&gt;</li> <li>-step&lt;pumpname&gt;   &lt;pipename&gt;   &lt;cisternname&gt;   &lt;sourcename&gt;</li> </ul> <p>Parancsokhoz tartozó use-case leírások</p> <p>Prototípus tesztetek leírása:</p> <ul style="list-style-type: none"> <li>-Player moves to sticky Pipe and tries to move again</li> <li>-Source steps</li> <li>-Player connects Pipe to Cistern</li> <li>-Player breaks Pipe</li> <li>-Mechanic picks up Pump at Cistern</li> </ul> <p>Skáre:</p>
------------------------	---------	--	---

- connect nodename pipename {nodename}
- sticky pipename
- connectpipe playernname
- leakpipe playernname
- switchpump playernname pip1  
pip2
- load

parancsok leírása és hozzá tartozó use-casek.

- Player switches pump and steps it
- Pump steps
- Player connects pipe to Pipe
- Mechanic repairs leaking pipe
- Mechanic picks up Pipe a Cistern
- Mechanic places Pump on Pump
- Leaking pipe gets water
- Player leaks out pipe and pump steps one

tesztesetek leírásának megadása.

Deé-Lukács:

Prototípus parancsainak leírása:  
move, grease, switch, placepump,  
repair

Prototípus teszteseteinek leírása:

- Pipe network clogs
- Cistern steps
- Player connects pipe to Source
- Mechanic repairs broken pump
- Mechanic picks up Pump at Source
- Mechanic places Pump on Cistern
- Mechanic places Pump on Source
- Player disconnects outpipe from pump and pump steps

Vörös: Szekvenciadiagram: 7.0.3.4-6

Tesztelést segítő program, kimeneti nyelv, parancsok: leak, state,  
disconnectpipe

tesztesetek: Pump with empty tank  
steps 2, broken pump steps, player  
disconnects empty pipe, Mechanic  
repairs working pump, Mechanic  
picks up pipe at source

2023.04.21. 21:00	2 óra	Kiss	módosított osztálydiagram elkészítése, megváltozott metódusok leírása, szekvenciadiagramok: 7.0.3.1-3
2023.04.22. 19:30	2 óra	Skáre	<ul style="list-style-type: none"> <li>1. connect nodename pipename {nodename}</li> <li>2. sticky pipename</li> <li>3. connectpipe playername</li> <li>4. leakpipe playername</li> <li>5. switchpump playername pip1 pip2</li> <li>6. load</li> </ul> Parancsok leírása (7.1) és hozzá kapcsolódó use-casek leírása (7.2).
2023.04.22 21:00	1 óra	Skáre	<ul style="list-style-type: none"> <li>1. Player switches pump and steps it</li> <li>2. Pump steps</li> <li>3. Player connects pipe to Pipe</li> <li>4. Mechanic repairs leaking pipe</li> <li>5. Mechanic picks up Pipe a Cistern</li> <li>6. Mechanic places Pump on Pump</li> <li>7. Leaking pipe gets water</li> <li>8. Player leaks out pipe and pump steps one</li> </ul> tesztesetek leírásának megadása.
2023.04.22. 21:00	3, 5 óra	Vörös	Tesztesetek leírása: Pump with empty tank steps 2, broken pump steps, player disconnects empty pipe, Mechanic repairs working pump, Mechanic picks up pipe at source Parancsok leírása: state, leak, disconnectpipe Prototípus interfész definíciója Kimeneti nyelv def. Szekvenciák: 7.0.3.4, 7.0.3.5, 7.0.3.6
2023.04.22. 22:00	2 óra	Andó	Prototípus parancsainak leírása: -fill <pipename/pumpname> -break <pumpname>

			<ul style="list-style-type: none"> <li>-pickup &lt;pipename/pumpname&gt;</li> <li>&lt;mechanicname&gt;</li> <li>-slipperypipe &lt;sabournname&gt;</li> <li>-stickypipe &lt;playername&gt;</li> </ul> <p>Az ezekhez kapcsolódó use-case leírások:</p> <ul style="list-style-type: none"> <li>-Fill pipe</li> <li>-Fill pump</li> <li>-Break pump</li> <li>-Pick up pipe by mechanic</li> <li>-Pick up pump by mechanic</li> <li>-Slippery pipe by saboteur</li> <li>-Sticky pipe by player</li> </ul>
2023.04.23. 10:00	2 óra	Kiss	<p>Prototípus parancsainak leírása:</p> <ul style="list-style-type: none"> <li>-add &lt;Type&gt; &lt;name&gt;</li> <li>-drain &lt;pipename&gt;   &lt;pumpname&gt;</li> <li>-step&lt;pumpname&gt;   &lt;pipename&gt;   &lt;cisternname&gt;   &lt;sourcename&gt;</li> </ul> <p>Parancsokhoz tartozó use-case leírások</p> <p>Prototípus teszteseteinek leírása:</p> <ul style="list-style-type: none"> <li>-Player moves to sticky Pipe and tries to move again</li> <li>-Source steps</li> <li>-Player connects Pipe to Cistern</li> <li>-Player breaks Pipe</li> <li>-Mechanic picks up Pump at Cistern</li> </ul>
2023.04.23. 11:00	1 óra	Vörös	Tesztelést segítő program funkcionalitása, terve
2023.04.23. 11:00	2 óra	Andó	<p>Prototípus teszteseteinek leírása:</p> <ul style="list-style-type: none"> <li>-Player moves to slippery pipe from pump</li> <li>-Player connects pipe to pump</li> <li>-Player disconnects full pipe</li> <li>-Mechanic repairs undamaged pipe</li> <li>-Mechanic places pump on pipe</li> <li>-Undamaged pipe gets water</li> <li>-Mechanic repairs broken pump and it steps 2</li> <li>-Pump breaks with full tank and it steps 2</li> </ul>

2023.04.23. 12.00	3 óra	Deé-Lukács	<p>Prototípus parancsainak leírása:</p> <ul style="list-style-type: none"> <li>-move &lt;playername&gt;</li> <li>&lt;elementname&gt;</li> <li>-grease &lt;pipename&gt;</li> <li>-switch &lt;pumpname&gt;</li> <li>&lt;pipename_from&gt; &lt;pipename_to&gt;</li> <li>-placepump &lt;mechanicname&gt;</li> <li>-repair &lt;pipename&gt; &lt;pumpname&gt;</li> <li>&lt;mechanicname&gt;</li> </ul> <p>Prototípus teszteseteinek leírása:</p> <ul style="list-style-type: none"> <li>-Pipe network clogs</li> <li>-Cistern steps</li> <li>-Player connects pipe to Source</li> <li>-Mechanic repairs broken pump</li> <li>-Mechanic picks up Pump at Source</li> <li>-Mechanic places Pump on Cistern</li> <li>-Mechanic places Pump on Source</li> <li>-Player disconnects outpipe from pump and pump steps</li> </ul>
----------------------	-------	------------	--

# 8. Részletes tervez

## 8.1 Osztályok és metódusok tervez.

### 8.1.1 Cistern

- **Felelősség**

A játékban szereplő ciszternákat megvalósító osztály. Fő felelőssége, hogy a hozzá csatlakoztatott csövekből kiszívja a vizet, majd a beérkezett víz mennyiségétől függően -pontosabban, hogy mennyi csőből érkezett víz- növeli a szerelők csapata által gyűjtött vízmennyiséget. Emellett tartózkodhat rajta tetszőleges számú karakter, akik szabadon választhatnak le, illetve csatlakoztathatnak fel csöveget a ciszternára.

- **Ősosztály**

Element → Node

- **Interfész**

Mivel a Node osztály leszármazottja, megvalósítja az ISteppable interfészt.

- **Metódusok**

- **+Cistern()**: Az osztály konstruktora. Attribútumai nem térnek el az ősosztály (Node) attribútumaitól.
- **void +Step()**: Ebben a függvényben a ciszterna végrehajtja a következőt: az összes hozzá csatlakoztatott csővégre meghívja a RemoveWater() függvényt, és ha az igaz értékkel tér vissza - azaz érkezett víz a csőből, akkor a ciszterna megnöveli a szabotőrök vízgyűjtőjének méretét, pontosabban meghívja a saboteurPool objektumra az AddWater() függvényt. A saboteurPool referenciáját a Game osztály megfelelő getter függvényének meghívásával éri el.

### 8.1.2 Element

- **Felelősség**

A játékban előforduló passzív és aktív elemeket reprezentálja, ezeken keresztül lehet mozogni, tehát képes játékoszt eltávolítani és elfogadni. Felelős a rajta lévő játékos/játékosok nyilvántartásáért. Absztrakt ősosztály.

- **Attribútumok**

- **List<Player> #players**: az adott elemen álló játékosokat tároló lista

- **Metódusok**

- **+Element()**: Az osztály konstruktora, belsőjében létrehozza, a rajta lévő játékosokat tároló listát, (kezdetben nulla elemmel).

- **+List<Element> GetNeighbours():** Visszaadja az adott Element szomszédait. Absztrakt metódus, amely a Pipe és Node osztályokban van megvalósítva.
- **+bool AcceptPlayer(Player p):** A paraméterként megadott játékost rálépteti az adott Element-re. Igazzal tér vissza, ha sikeres a ráléptetés. Absztrakt metódus, amely a Pipe és Node osztályokban van megvalósítva.
- **+void RemovePlayer(Player p):** A paraméterként megadott játékost törli az adott Element-en lévő játékosok közül.
- **+bool AddPipe(PipeEnd p):** Csatlakoztatja a megadott csővéget az adott elemhez. Igazzal tér vissza, ha sikeres a csatlakoztatás. Itt alapesetben hamissal tér vissza, hogy ha csőre próbálnánk meghívni, akkor ne történjen semmi. A Node osztályban van felüldefiniálva.
- **+void RemovePipe(PipeEnd p):** lecsatlakoztatja a megadott csővéget az adott elemről. Itt alapesetben üres a függvénytörzse, a Node osztályban van felüldefiniálva.
- **+void Repair():** megjavítja az adott Elementet. Itt üres a függvénytörzse, a Pump definiálja felül.
- **+void Switch(PipeEnd from, PipeEnd to):** Az adott elemen a be- és kimeneti csövek átállítására alkalmas függvény. Itt üres a függvénytörzse, a Pump definiálja felül.
- **+Pump MakePump():** Az adott elemnél új pumpa létrehozására alkalmas függvény, amely normális esetben egy új pumpát ad vissza. Itt a függvénytörzse null-t ad vissza, hogyha nem cisternán próbáljuk meghívni, akkor ne történjen semmi. A Cistern definiálja felül.
- **+PipeEnd MakePipe():** Az adott elemnél új cső létrehozására alkalmas függvény, amely normális esetben egy új csővét ad vissza. Itt a függvénytörzse null-t ad vissza, hogyha nem cisternán próbáljuk meghívni, akkor ne történjen semmi. A Pipe osztály definiálja felül.
- **+Pipe Cut():** Az adott elem elvágására alkalmas függvény, amely egy új csövet ad vissza normális esetben. Itt a függvénytörzse null-t ad vissza, hogyha nem csövön próbáljuk meghívni, akkor ne történjen semmi. A Pipe osztály definiálja felül.
- **+void Leak():** Kilyukasztja az adott elemet. Itt üres a függvénytörzse, a Pipe definiálja felül.
- **+void Patch():** Megfoltozza az adott elemet. Itt üres a függvénytörzse, a Pipe definiálja felül.
- **+void MakeSticky():** Ragadóssá teszi az adott elemet. Itt üres a függvénytörzse, a Pipe definiálja felül.
- **+void MakeSlippery():** Csúszóssá teszi az adott elemet. Itt üres a függvénytörzse, a Pipe definiálja felül.
- **+List<PipeEnd> GetEnds():** Visszaadja az adott elem “végeit”. Itt a függvénytörzsében csak null-t ad vissza, a Pipe osztályban van felüldefiniálva.
- **+void GetState():** Kiírja az adott Element-en lévő játékosokat, alábbi formában:

Player(s): <playername1> .. <playernameN>

### 8.1.3 Game

- **Felelősség**

A játék vezérléséért felelős osztály. Fő felelőssége a bemeneten megadott parancsok végrehajtása, és a felhasználó által létrehozott objektumok nyilvántartása és kezelése, működtetése.

- **Attribútumok**

- **HashMap<String, Pipe> -pipes:** a prototípus működése/használata során létrehozott cső objektumokat tárolja, pontosabban String-Pipe kulcs-érték párokat, ahol a String az adott objektum neve, amit létrehozáskor adtunk neki.
- **HashMap <String, Pump> -pumps:** a prototípus működése/használata során létrehozott pumpa objektumokat tárolja, pontosabban String-Pump kulcs-érték párokat, ahol a String az adott objektum neve, amit létrehozáskor adtunk neki.
- **HashMap <String, Cistern> -cisterns:** a prototípus működése/használata során létrehozott ciszterna objektumokat tárolja, pontosabban String-Cistern kulcs-érték párokat, ahol a String az adott objektum neve, amit létrehozáskor adtunk neki.
- **HashMap <String, Source> -sources:** a prototípus működése/használata során létrehozott forrás objektumokat tárolja, pontosabban String-Source kulcs-érték párokat, ahol a String az adott objektum neve, amit létrehozáskor adtunk neki.
- **HashMap <String, Mechanic> -mechanics:** a prototípus működése/használata során létrehozott szerelő objektumokat tárolja, pontosabban String-Mechanic kulcs-érték párokat, ahol a String az adott objektum neve, amit létrehozáskor adtunk neki.
- **HashMap <String, Saboteur> -saboteurs:** a prototípus működése/használata során létrehozott szabotör objektumokat tárolja, pontosabban String-Saboteur kulcs-érték párokat, ahol a String az adott objektum neve, amit létrehozáskor adtunk neki.
- **bool -determinism:** logikai érték, mely meghatározza, hogy a prototípus programban létrehozott modell működése véletlenszerűen vagy determinisztikusan működjön.
- **Pool #saboteurPool:** a szerelők által gyűjtött pontokat tárolja
- **Pool #mechanicPool:** a szabotörök által gyűjtött pontokat tárolja
- **Timer -timer:** a játék időzítője, amelynek minden Tick()-je lépteti a léptethető objektumokat
- **int rounds:** azt tárolja, hogy hány kör van hátra a játékból, egy kör egy step-nek felel meg

- **Metódusok**

- **Pool +GetSaboteurPool():** visszaadja a saboteurPool-t
- **Pool +GetMechanicPool():** visszaadja a mechanicPool-t
- **void +main(String[] args):** egy ciklusban olvassa be a játékosok parancsait, addig, amíg a rounds nagyobb 0-nál, és lefuttatja azokat, amikor eléri a 0-t a rounds, akkor meghívja az EndGame() függvényt
- **void +StartGame(int rounds):** elindítja a játékot, beállítja a rounds értékét a paraméterben megadott számra
- **void +EndGame():** annak a csapatnak a nevét írja ki, akinek több víz gyűlt össze a Pool-jában
- **void +Add(String type, String name):**  
ha type Pipe

**akkor** létrehozunk egy új csövet, amit a megadott néven hozzáadunk a pipes HashMap-hez

**ha type Pump**

**akkor** létrehozunk egy új pumpát, amit a megadott néven hozzáadunk a pumps HashMap-hez

**ha type Cistern**

**akkor** létrehozunk egy új ciszternát, amit a megadott néven hozzáadunk a cisterns HashMap-hez

**ha type Source**

**akkor** létrehozunk egy új ciszternát, amit a megadott néven hozzáadunk a sources HashMap-hez

**ha type Saboteur**

**akkor** létrehozunk egy új szabotört, amit a megadott néven hozzáadunk a saboteurs HashMap-hez

**ha type Mechanic**

**akkor** létrehozunk egy új szerelőt, amit a megadott néven hozzáadunk a mechanics HashMap-hez

- **void +Break(String pumpname):** A paraméterként megadott nevű pumpát eltöri, azaz megkeresi a pumps HashMap-ben, majd meghívja a Pump osztály BreakPump() metódusát.
- **void +Connect(String node1, String pipe, String node2)**  
HA node2 null:  
    node1 és pipe összekötése  
KÜLÖNBEN  
    node1 és pipe összekötése  
    node2 és pipe összekötése
- **void +ConnectPipe(String playername):**  
A paraméterként megadott nevű játékos felcsatlakoztatja a csövet amit éppen fog arra a Nodera, amin éppen áll. Ha nem fog csövet vagy nem Nodeon áll, akkor nem történik semmi.
- **void +DisconnectPipe(String playername, String pipename):** a paraméterként megadott nevű szerelő/szabotőr lecsatlakoztatja a szintén paraméterként megadott nevű csövet arról a mezőről (lehet ez pumpa, ciszterna, forrás). A függvény megkeresi az adott nevű Mechanic/Saboteur objektumot, majd ha megtalálta azokat, a Mechanic/Saboteur DisconnectPipe függvényét meghívja az adott cső megfelelő végével. Erre a Pipe GetEnds függvényét használja.
- **void +Drain(String name):**  
minden Pipe-ra a pipes HashMapben  
    ha Pipe neve egyezik name paraméterrel  
        akkor meghívja a Pipe-on a RemoveWater() függvényt és visszatér  
    minden Pump-ra a pumps HashMapben  
        ha Pump neve egyezik name paraméterrel  
            akkor meghívja a Pump-on az EmptyWaterTank() függvényt és visszatér
- **void +EndRound():** meghívja a timer Tick() függvényét és csökkenti a rounds értékét 1-gyel
- **void +Fill(String name):**

minden Pipe-ra a pipes HashMapben  
 ha Pipe neve egyezik name paraméterrel  
 akkor meghívja a Pipe-on az AcceptWater() függvényt és visszatér

minden Pump-ra a pumps HashMapben  
 ha Pump neve egyezik name paraméterrel  
 akkor meghívja a Pump-on az FillWaterTank() függvényt és  
 visszatér

- **void +HoldPipe(String pipename, String mechanicname):** megkeresi a megadott nevű szerelőt a mechanics HashMapben és a megadott nevű csövet a pipes HashMap-ben, és meghívja a szerelőn a HoldPipe függvényt, paraméterként megadva a csövet
  - **void +HoldPump(String pumpname, String mechanicname):** megkeresi a megadott nevű szerelőt a mechanics HashMapben és a megadott nevű pumpát a pumps HashMap-ben, és meghívja a szerelőn a HoldPump függvényt, paraméterként megadva a pumpát
  - **void +Move(String playerName, String elementName):** Megkeresi a -pipes, -sources, -cisterns, -pumps hashmapekbén az elementName kulccsal tárolt objektumot, ahova a játékos fog lépni. Ezután a -mechanics vagy -saboteurs hashmapekből megkeresi a playerName kulcsnak tartozó játékost. Végül úgy lépteti rá a játékost az elemre, hogy meghívja a játékoson a Move(Element to) metódust, ahol a to paraméter lesz az elem, ahova lépni fog a játékos.
  - **void +State(String objectname, string args):** függvény, melynek célja a prototípusban létrehozott objektumok állapotának lekérdezése. Paraméterként átveszi a lekérdezendő objektum nevét, és megkeresi a privát kollekcióban a névhez tartozó objektumot, amin meghívja a GetState függvényt. Ha a második paraméterként kapott args string nem üres, akkor ezeket átadja a Pump és Pipe paraméteres GetState függvényének. Más típusú objektumok lekérdezésénél ezt az args paramétert nem vesszük figyelembe.
  - **void +Step(String name):**
- minden Pipe-ra a pipes HashMapben  
 ha Pipe neve egyezik name paraméterrel  
 akkor meghívja a Pipe-on a Step() függvényt és visszatér
- minden Pump-ra a pumps HashMapben  
 ha Pump neve egyezik name paraméterrel  
 akkor meghívja a Pump-on a Step() függvényt és visszatér
- minden Source-ra a sources HashMapben  
 ha Source neve egyezik name paraméterrel  
 akkor meghívja a Source-on a Step() függvényt és visszatér
- minden Cistern-re a cisterns HashMapben  
 ha Cistern neve egyezik name paraméterrel  
 akkor meghívja a Cistern-en a Step() függvényt és visszatér
- **void +Leak(String pipename):** A paraméterként megadott nevű csövön meghívja a Leak függvényt, ami a cső eltörését végzi. A megfelelő nevű csövet a pipes nevű HashMap-ból keresi ki. Amennyiben nem talált megfelelő nevű Pipe objektumot, nem hajt végre semmit.
  - **+Random(bool on):** setter függvény, mely beállítja a *determinism* mező értékét, ami meghatározza a modell működésének véletlenszerűségét.

- **void +Sticky(String pipe):**  
A paraméterként megadott nevű csövön meghívja a MakeSticky függvényt, azaz ragadóssá teszi a csövet. Ha nincs ilyen cső, akkor nem történik semmi.
- **void +LeakPipe(String playername):**  
A megadott nevű játékos kilyukasztja a csövet, amin éppen áll. Ha nem csövön áll, akkor nem történik semmi.
- **void SwitchPump(String playername, String pip1, String pipe2):**  
A megadott nevű játékos átállítja azt a pumpát, amin éppen áll: a bemeneti cső a pip1 nevű, a kimeneti cső a pip2 nevű.
- **void +Grease(String pipeName):** A paraméterként megadott csövet csúszóssé teszi. Először kikeresi a -pipes hashmapból a **pipeName** kulcsnak tartozó csövet, majd meghívja rajta a **MakeSlippery()** metódust.
- **void +SlipperyPipe(String saboteurName):** A paraméterként megadott szabotőr csúszóssé teszi azt a csövet, amin éppen áll. Megkeresi a saboteurs HashMap-ben a paraméterként megadott szabotőrt, majd meghívja rajta a **MakeSlipperyPipe()** metódust.
- **void +StickyPipe(String playerName):** A paraméterként megadott játékos ragadóssá teszi azt a csövet, amin éppen áll, azaz a a saboteurs vagy mechanics HashMap-ekben megkeresi a paraméterként megadott nevű játékost, majd meghívja rajta a **MakeStickyPipe()** metódust.
- **void Switch(String pumpName, String input, String output):** A paraméterként megadott pumpát átállítja. Kikeresi a -pumps hashmapból a **pumpName** kulcsnak tartozó pumpát, és az **input**, **output** paramétereiben megadott csöveket a -pipes hashmapból. Ezután a csövektől bekéri a végeiket a **GetEnds()** metódus segítségével és kiválasztja azokat a végeket, amik a megkeresett pumpához csatlakoznak, a **GetAttachedNode()** segítségével. Végül a pumpán meghívja a **Switch()** metódust, ami paraméterül kapja a megfelelő csővégeket.
- **void +PickUp(String type, String objectName, String mechanicName):** A paraméterként megadott szerelő felvesz egy, a paraméterként megadott type-nak megfelelő (Pipe vagy Pump) objektumot annál a ciszternánál, amin éppen áll. Ha a type cső, akkor meghívja a paraméterként megadott, és a mechanics HashMap-ból kikeresett szerelő PickUpPipe() metódusát, majd beleteszi az újonnan létrehozott csövet a pipes HashMap-be a paraméterként megadott néven. Ha a type pumpa volt, akkor meghívja a paraméterként megadott, és a mechanics HashMap-ból kikeresett szerelő PickUpPump() metódusát, majd beleteszi az újonnan létrehozott pumpát a pumps HashMap-be a paraméterként megadott néven.
- **void +PlacePump(String mechanicName):** A paraméterként megadott nevű játékos letesz egy pumpát. Kikeresi a -mechanics hashmapból a **mechanicName** kulcsnak tartozó játékost, és meghívja rajta a **PlacePump()** metódust.
- **void +Repair(String elementName):** Megjavítja a paraméterként megadott nevű csövet vagy pumpát. Kikeresi a -pipes, -pumps hashmapekből az **elementName** kulcsnak tartozó elemet. Ezután, ha ez cső volt, akkor meghívja rajta a **Patch()** metódust, ha pumpa, akkor a **Repair()** metódust.

### 8.1.4 Mechanic

- **Felelősség**

A játékban a szerelő karaktereket reprezentálja. Képes megjavítani az elromlott pumpákat és a lyukas csöveket, új csövet / pumpákat magához venni és letenni.

- **Ősosztályok**

Player

- **Attribútumok**

- **Pump[] -holdingPumps:** eltárolja a szerelőnél lévő pumpákat

- **Metódusok**

- **Mechanic(Element on):** konstruktur, meghívja az ős konstrukturát a megadott paraméterrel

- **void +RepairPump():** meghívja a Repair() függvényt a Mechanic on Element-jén

- **void +PlacePump():**

ha holdingPumps nem üres

akkor meghívja Cut() függvényt az on Elementen, ami visszatérít egy új csövet

ha a Cut() visszatérési értéke, vagyis az új cső nem null

akkor

1. a GetEnds() függvény segítségével lekéri a csőnek a végeit, amin áll
2. a második csővéghez csatlakoztatja a holdingPumps első pumpáját, úgy, hogy meghívja rajta a ConnectNode függvényt
3. a GetEnds() függvény segítségével lekéri az új cső végeit
4. a második csővéghez csatlakoztatja a holdingPumps első pumpáját, úgy, hogy meghívja rajta a ConnectNode függvényt
5. rálép a Move függvény segítségével az újonnan lerakott pumpára
6. törli a holdingPumps első pumpáját

- **void +PickUpPump():** Meghívja a MakePump() függvényt az on Element-en, és ha nem null a visszatérési értéke, hozzáadja a holdingPumps-hoz

- **void +PickUpPipe():** Ha null a holdingPipeEnd, akkor meghívja a MakePipe()-ot az on Element-en, és ha a visszatérési értéke nem null, beállítja a holdingPipeEnd-nek

- **void +RepairPipe():** Meghívja a Patch() függvényt az on Elementen

- **void +GetState():** meghívja az őse getState() függvényét, valamint kiírja a kezében tartott pumpákat az alábbi formátumban:

on: <elementname>

holdingPipe <pipename> [ha nincs a kezében cső, null]

holdingPumps: <pumpname1> .. <pumpnameN>

### 8.1.5 Node

- **Felelősség**

A játékban előforduló ‘csomópontokat’, aktív elemeket reprezentálja. Felelős a rácsatlakoztatott csővégek nyilvántartásáért. Akárhány játékos tartózkodhat rajta egyszerre, akik szabadon csatlakoztathatnak fel és le csöveket. Absztrakt ōsosztály.

- **Ósosztályok**

Element

- **Interfészek**

ISteppable

- **Attribútumok**

- **PipeEnd[] #pipeEnds:** az adott Node-dal aktuálisan összekötött csővégek. A csővégek referenciáit egy fix hosszú tömbben tároljuk.

- **Metódusok**

- **+Node():** Az osztály konstruktora, amely belsejében inicializálódik a csővégek tömbje (azaz minden egyik elem null)
- **+bool AcceptPlayer(Player p):** Megvalósítja az Element *+bool AcceptPlayer(Player p)* absztrakt metódusát a saját viselkedésének megfelelően. Hozzáadja az adott Node-on álló játékosokhoz (ezt az attribútumot az Element osztályból örökli) a paraméterként megadott játékos. Igazzal tér vissza, ha a paraméterként megadott játékos ráléphet az adott Node-ra (Node esetén minden igazzal tér vissza).
- **+void Step():** Lépteti az adott Nodeot. Ez az implementációja az ISteppable interfész *+void Step()* metódusának, amely absztrakt, hiszen az egyes leszármazottakban másképpen kell megvalósítani.
- **+List<Element> GetNeighbours():** Megvalósítja az Element *+List<Element> GetNeighbours()* absztrakt metódusát a saját viselkedésének megfelelően. Visszaadja az adott Node szomszédait (amelyek Pipe-ok), méghozzá úgy, hogy végigmegy a hozzá csatlakoztatott csővégek tömbjén, és ha annak aktuális eleme nem null, akkor lekérdezi attól, hogy melyik csőhöz tartozik, amit hozzáad a már korábban lokálisan létrehozott szomszédok listájához. Végül ezt a listát adja vissza.
- **+bool AddPipe(PipeEnd p):** Felüldefiniálja az Element *+bool AddPipe(PipeEnd p)* metódusát. Ha a paraméterként megadott csővéget lehet csatlakoztatni az adott Node-hoz (van még szabad hely), akkor eltárolja a vele összeköttetésben lévő csővégek tömbjének végére, és a csővég “oldaláról” is elvégzi a beállításokat, azaz hozzáköti magát az adott Node-hoz, majd igazzal tér vissza. Ha eleve nem volt szabad hely, akkor hamissal tér vissza.
- **+void RemovePipe(PipeEnd p):** Felüldefiniálja az Element *+void RemovePipe(PipeEnd p)* metódusát. A paraméterként megadott csővéget lecsatlakoztatja az adott Node-ról, majd törli a Node hozzáköött csővégei közül (null-ra állítja).
- **+PipeEnd[] GetPipeEnds():** visszaadja az adott Node-hoz az aktuálisan hozzácsatlakoztatott csővégek tömbjét.

- **+void GetState():** Kiírja, hogy mely játékosok állnak rajta éppen és, hogy mely csővégek vannak hozzákötve az adott Node-hoz, az alábbi formában:

Players: <playername1> .. <playernameN>  
 ConnectedPipes: <pipename1> .. <pipenameN>

### 8.1.6 Pipe

- **Felelősség**

A játékban lévő csöveget reprezentálja, amelyek legfőbb feladata a víz tárolása. A végein keresztül tud vizet fogadni. Lehet lyukas, ilyenkor kifolyatja a belé érkező vizet. Lehet csúszós, ilyenkor a rálépő játékos véletlenszerűen átrakja az egyik végére. Lehet ragadós, ilyenkor a rálépő játékos leragasztja egy ideig (azaz nem tud továbblépni). Amennyiben megfoltozták, valamennyi ideig nem lehet kilyukasztani.

- **Ősosztály**

Element

- **Interfész**

ISteppable

- **Attribútumok**

- **int slipperyFor:** számláló, amely számon tartja az időt, amíg a cső csúszós, alapértelmezetten 0.
- **int stickyFor:** számláló, amely számon tartja az időt, amíg a cső ragadós, alapértelmezetten 0.
- **int protectedFor:** számláló, amely számon tartja az időt, amíg a csövet nem lehet kilyukasztani, alapértelmezetten 0.
- **bool hasWater:** igaz, ha a tárolóban van víz, alapértelmezetten hamis.
- **bool isBroken:** igaz, ha a cső lyukas, alapértelmezetten hamis.
- **Pool saboteurPool:** eltárolja a lyukas csőből kifolyt víz mennyiségét
- **PipeEnd ends[]:** eltárolja a csőhöz tartozó végeket

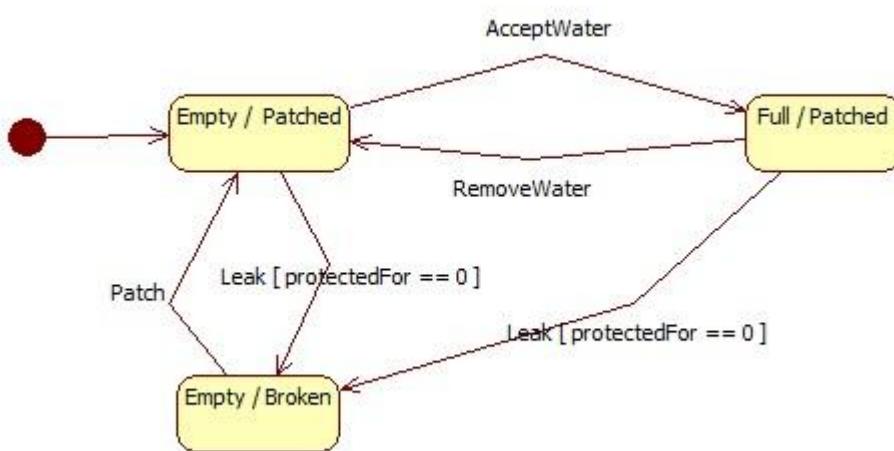
- **Metódusok**

- **void +Step():**  
 HA slipperyFor nem nulla:  
     slipperyFor csökkentése 1-el.  
 HA stickyFor nem nulla:  
     stickyFor csökkentése 1-el.  
 HA protectedFor nem nulla:  
     protectedFor csökkentése 1-el.
- **void +MakeSticky():**  
 HA stickyFor nem nulla:  
     stickyFor beállítása random értékre a játék keretei között.
- **void +MakeSlippery():**

- HA slipperyFor nem nulla:
  - slipperyFor beállítása random értékre a játék keretei között.
- **+Pipe(Node node):**  
node hozzáadása az ends listához.
- **bool +AcceptPlayer(Player p):**  
HA nincs játékos a csövön:
  - HA slipperyFor nem nulla:
    - p hozzáadása a players listához
    - p átléptetése a cső valamelyik végére véletlenszerűen
    - visszatérés hamissal
  - KÜLÖNBEN HA stickyFor nem nulla:
    - p leragasztása és p hozzáadása a players listához
    - visszatérés igazzal
  - KÜLÖNBEN
    - p hozzáadása a players listához
    - visszatérés igazzal
  - KÜLÖNBEN
    - visszatérés hamissal
- **void +AcceptWater():**  
HA isBroken igaz:
  - saboteurPool-ba víz vezetése
- **bool +RemoveWater():**  
hasWater hamisra állítása  
visszatérés hasWater módosítás előtti értékével.
- **void +Leak():**  
HA protectedFor nulla:
  - isBroken igazra állítása
- **void +Patch():**  
HA isBroken igaz:
  - isBroken hamisra állítása
  - protectedFor beállítása random értékre a játék keretei között
- **Element[] +GetNeighbours():**  
Lista létrehozása neighbours néven, amely Elementeket tartalmaz  
CIKLUS amely végigmegy ends-en:
  - az aktuális end-hez tartozó Node hozzáadása a neighbours listához
  - visszatérés a neighbours listával
- **PipeEnd[] +GetEnds():**  
visszatérés az ends listával
- **Pipe +Cut():**  
HA az aktuális csőnek nincs szabad vége:
  - a második véghez tartozó Noderól az aktuális cső eltávolítása
  - új cső létrehozása, amely az előzőleg eltávolított véghez kapcsolódik

visszatérés az új csővel  
**KÜLÖNBEN**  
visszatérés nullal

- **Állapotdiagram**



### 8.1.7 PipeEnd

- **Felelősség**

A játékban levő csővégeket reprezentálja. Lehet hozzá Node-ot kötni illetve lecsatlakoztatni róla. Tárolja, hogy milyen Node-dal áll összeköttetésben, és hogy melyik csőhöz tartozik.

- **Attribútumok**

- **Pipe -pipe:** annak a csőnek a referenciaja, amihez tartozik a csővég.
- **Node -node:** eltárolja a csővéggel összeköttetésben levő aktív elemet. Ha a csőnek ez a vége szabad, akkor az értéke **null**.

- **Metódusok**

- **+PipeEnd(Pipe p):** az osztály konstruktora, inicializálja a **-pipe** attribútumot a **p** paraméterként kapott cső referenciaival. A **-node** attribútum kezdőértéke **null**.
- **bool +AcceptWater():** Továbbítja a vizet a csőnek, aminek ez a csővég a vége. Ha a cső képes vizet befogatni **igazzal** tér vissza, különben **hamissal**.
- **bool +RemoveWater():** Kisírja, eltávolítja a vizet a csőből, aminek ez a csővég a vége. Ha a csőből lehetett vizet kiszívní, **igazzal** tér vissza, különben **hamissal**.
- **Pipe +GetOwnPipe():** Getter a csővéghoz tartozó csőre. Visszatéríti a **-pipe** attribútum értékét.
- **Node +GetAttachedNode():** Getter arra az aktív elemre, amire rá van a csővég kapcsolva. Visszatéríti a **-node** attribútum értékét.
- **void +ConnectNode(Node n):** A paraméterként megadott Node-hoz csatlakoztatja a csővéget, azaz a **-node** attribútum megkapja az **n** nevű paraméterben tárolt Node referenciaját.

- **void +DisconnectFromNode()**: Lecsatlakoztatja a csővéget arról az aktív elemről, amire rá volt kapcsolva, ha víz volt a csőben, az kifolyik. Először megpróbál vizet eltávolítani a csőből. Ha ez sikeres, a szabótörök kapnak pontot. Ezután a **-node** attribútum értéke **null** lesz.
- **void +GetState()**: Kiírja a csővég belső állapotát a következő formában:  
pipe: <pipename>  
node: <nodename>

## 8.1.8 Player

- **Felelősség**

A játékosok által irányítható karaktereket reprezentálja. Eltárolja, hogy melyik Element-en áll, felelős azért, hogy csak akkor képes átmenni másik Element-re, ha nincs leragasztva. Eltárolja a nála levő csővéget, és tudja a csöveket csatlakoztatni más elemekhez. Absztrakt osztály.

- **Attribútumok**

- **PipeEnd #holdingPipeEnd**: eltárolja a játékosnál levő csővéget
- **Element #on**: eltárolja, hogy melyik Element-en tartózkodik a játékos
- **bool -isStuck**: igaz, amennyiben a játékos hozzáragadt a csőhöz

- **Metódusok**

- **void +Move(Element to)**:

**ha** isStuck hamis

akkor lekéri az on Element szomszédait, vagyis meghívja rajta a GetNeighbours() függvényt

minden szomszédra megnézi, hogy egyezik-e a to Elementtel

**ha** to egyezik valamelyik szomszéddal

akkor meghívja a to Elementen az AcceptPlayer függvényt,  
paraméterként saját magát adja meg

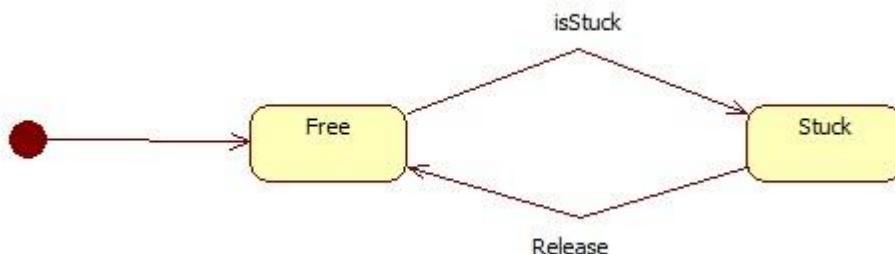
**ha** az AcceptPlayer visszatérési értéke igaz

akkor eltávolítja saját magát az on Elementről, vagyis meghívja a RemovePlayer(saját maga) függvényt az on Elementen és beállítja on Elementnek a to Elementet

- **+Player(Element on)**: Az osztály konstruktora, a paraméterben megadott Element-et eltárolja a Player on attribútumában
- **void +SwitchPump(PipeEnd from, PipeEnd to)**: átállítja annak a pumpának a bemeneti és kimeneti csövét, amin a karakter áll, a megadott paraméterek alapján, vagyis meghívja az on Elementen a Switch(from, to)-t
- **void +ConnectPipe()**: ha a holdingPipeEnd nem null, csatlakoztatja azt ahoz az Elementhez, amin áll, vagyis meghívja a holdingPipeEnd-end a ConnectNode(on) függvényt

- **void +DisconnectPipe(PipeEnd pipeend):** ha a holdinPipeEnd null, a megadott paraméterű csővéget lecsatlakoztatja, vagyis meghívja az on Elementen a RemovePipe(pipeend) függvényt, és eltárolja a pipeend-et a holdinPipeEnd-ben
- **void +BreakPipe():** kilyukaszta a csövet, amin a játékos áll, vagyis meghívja a Leak függvényt az on Elementen
- **void +MakeStickyPipe():** ragadóssá teszi a csövet, amin a játékos áll, vagyis meghívja a MakeSticky-t az on Elementen
- **void +Stuck():** odarasztja a játékost ahhoz a csőhöz, amin áll, vagyis igazra állítja az isStuck attribútumot
- **void +Release():** hamisra állítja az isStuck attribútumot
- **void +HoldPipe(Pipe p):** p cső második végét (amit a GetEnds() függvénykel kap meg) beállítja a holdingPipeEnd-nek
- **void +HoldPump(Pump p):** hozzáadja p-t a holdingPumps-hoz
- **void +SlippedTo(Element e):** átcúsztatja a játékost a megadott Elementre, vagyis eltárolja azt az on attribútumában
- **void +GetState():** kiírja a játékos helyzetét és a nála levő csövet az alábbi formátumban:  
on: <elementname>  
holdingPipe <pipename> [ha nincs a kezében cső, null]

- **Állapotdiagram:**



### 8.1.9 Pool

- **Felelősség**

Számon tartja a belefolyt víz mennyiségét.

- **Attribútumok**

- **int -amount:** az adott Pool-ba eddig befolyt víz mennyisége

- **Metódusok**

- **+Pool():** Az osztály konstruktora. Az osztály úgy jön létre, hogy az **-amount** attribútum kezdőértéke 0.
- **void +AddWater():** Növeli az amount értékét 1-gyel

- **void +GetWater():** Getter a befolyt víz mennyiségére. Visszatéríti az **-amount** attribútum értékét.

### 8.1.10 Pump

- **Felelősség**

A játékban előforduló pumpákat, azok működését megvalósító osztály. Legfőbb felelőssége, hogy a hozzá csatlakoztatott csövek közül a két kiválasztott között vizet mozgasson minden ütemben. Nyilvántartja a hozzá csatlakoztatott csöveget, valamint a pumpálás szempontjából ki- és bemeneti csöveget. A pumpálás irányát természetesen meg lehet változtatni rajta. Rendelkezik még egy átmeneti tárolóval, ami arra szolgál, hogy ha nem érkezik víz a bementén, akkor is még egy ütemig tudja biztosítani a vizet a kimeneti csőnek. Létezik nem működőképes állapota is, ilyenkor nem pumpál egyáltalán, legfeljebb az átmeneti tárolója leereszt, azaz kiengedi a vizet a kimeneti csőbe.

- **Ősztály**

Element → Node

- **Interfész**

Mivel a Node osztály leszármazottja, megvalósítja az ISteppable interfészt

- **Attribútumok**

- **int -pipeCapacity:** a pumpa azon jellemzője, ami meghatározza a pumpához csatlakoztatható csövek maximális számát.
- **bool -isBroken:** a pumpa működési állapota. A pumpa meghibásodott állapotban nem tud további vizet szívni, csak az átmeneti tárolójából a vizet kiengedni.
- **int -outPipe:** A pumpán aktuálisan kiválasztott bemeneti cső sorszáma. Értéke 0 és a csatlakoztatható pumpák maximális száma(*pipeCapacity*) közötti egész értéket vehet fel. Ha értéke zérus, akkor a pumpa még inicializálatlan állapotban van, vagy a legutóbbi bemeneti csövét csatlakoztatták anélkül, hogy új bemeneti csövet állítottak be. A sorszám az osztály *pipeEnds* kollekciójában/tömbjében értelmezett, azaz a tömb megfelelő sorszámú eleme lesz a bemeneti cső(vég).
- **int -inPipe:** A pumpán aktuálisan kiválasztott kimeneti cső sorszáma. Értéke 0 és a csatlakoztatható pumpák maximális száma(*pipeCapacity*) közötti egész értéket vehet fel. Ha értéke zérus, akkor a pumpa még inicializálatlan állapotban van, vagy a legutóbbi kimeneti csövét csatlakoztatták anélkül, hogy új kimeneti csövet állítottak volna be. A sorszám az osztály *pipeEnds* kollekciójában/tömbjében értelmezett, azaz a tömb megfelelő sorszámú eleme lesz a kimeneti cső(vég).
- **bool -tankFull:** a pumpa átmeneti tárolójának állapota. Igaz érték esetén a tárolót vízzel telinek, hamis érték esetén üresnek tekintjük.

- **Metódusok**

- **+Pump():** az osztály konstruktora. Példányosításkor egy pumpa alapértelmezetten működőképes állapotban van, egyetlen csővég sincs

hozzákapcsolva, a ki- és bemeneti csövek nullás sorszámmal jelöltek, és az átmeneti tároló üres.

- **void +Step():** lásd: activity diagram a következő alcímnél
- **void +Switch(from: PipeEnd, to:PipeEnd):** a pumpán a vízáramlás iránya megváltozik, a paraméterként kapott csővégek lesznek a ki- és bemeneti csövek. Ha mind a két csővég benne van a *pipeEnds* tömbben, akkor a paraméterek tömbbeli sorszámát veszik fel az *inPipe* és *outPipe* attribútumok. Amennyiben a egyik paraméterként kapott csővég nincs a pumpához csatlakoztatva, azaz nem szerepel a *pipeEnds* tömbben, akkor a pumpa megfelelő állapota(*inPipe* vagy *outPipe*) változatlan marad a függvényhívás után.
- **void +BreakPump():** Setter függvény, amely a pumpa állapotát törött/nem működőképes állapotba állítja, vagyis az *isBroken* mező értékét igazba állítja.
- **void Repair():** Setter függvény, amely a pumpa állapotát nem törött/működőképes állapotba állítja, vagyis az *isBroken* mező értékét hamisra állítja.
- **void +FillWaterTank():** Setter függvény, amely megtölти a pumpa átmeneti tárolójátvízzel, vagyis a *tankFull*mező értékét igazra állítja.
- **void +EmptyWaterTank():** Setter függvény, amely kiüríti a pumpa átmeneti tárolóját, vagyis a *tankFull*mező értékét hamisra állítja
- **void GetState():** állapotlekérdező függvény, mely kiírja a pumpa aktuális állapotát a következő alakban:

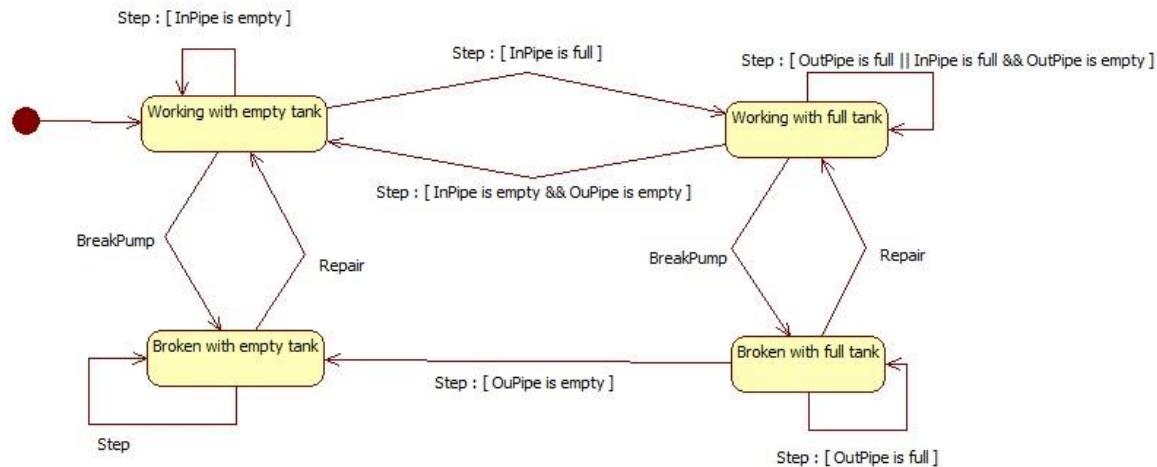
```

Players: <playername1> .. <playernameN>
ConnectedPipes: <pipename1> .. <pipenameN>
InPipe: <pipename>           [ha nincs semmi kiválasztva, null]
OutPipe: <pipename>          [ha nincs semmi kiválasztva, null]
isBroken: true/false
TankFull: true/false

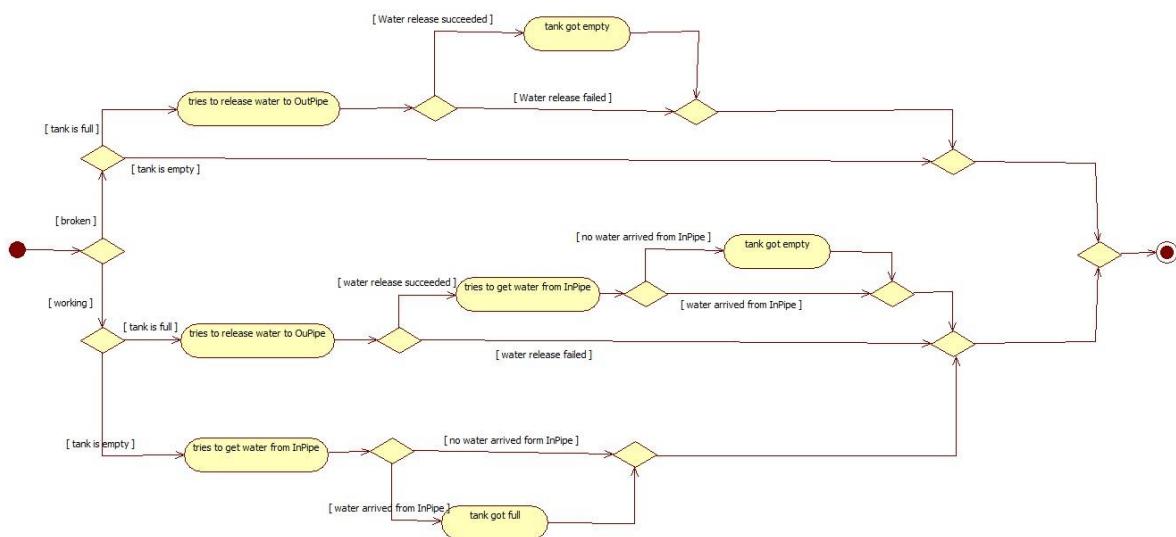
```

- **void GetState(string args) :** állapotlekérdező függvény, mely csak bizonyos belső állapotait írja ki az adott pumpának. A kiírt jellemzőket a paraméter határozza meg, melynek karakterei a pumpa egyes tulajdonságait kódolják. (Lásd: 0.1 Bemeneti nyelv javítása, bővítése alcím alatt a state parancs leírását) A tulajdonságok kiírásának formátuma megegyezik a (paraméter nélküli) *GetState()* függvényben definiált formátummal.

### 8.1.10.1 Pump state chart



### 8.1.10.2 Pump.Step()



### 8.1.11 Saboteur

- **Felelősség**

A játékban a szabotőr karaktert reprezentálja. Képes kilyukasztani a csöveket.

- **Ősosztály**

Player

- **Metódusok**

- **void +MakeSlipperyPipe():**  
csúszóssá teszi azt az Elementet, amin a játékos éppen áll
- **void +BreakPipe():**  
kilyukasztja azt az Elementet, amin a játékos éppen áll

### 8.1.12 Source

- **Felelősség**

A játékban lévő forrást reprezentálja. Folyamatosan vizet pumpál a vele összeköttetésben álló csövek számára.

- **Ősosztályok**

Element → Node

- **Metódusok**

- **+public Source():** Az osztály konstruktora, attribútumai nem térnek el az ősosztály attribútumaitól.
- **+void Step():** Lépteti a Source-ot, azaz megvalósítja a Node **+void Step()** absztrakt metódusát a saját viselkedésének megfelelően. Végigmegy a vele összeköttetésben lévő csővégek tömbjén (ezt az attribútumot a Node-ból örökli), és ha az aktuális csővég nem null, akkor a csővég elfogadja a továbbítandó vizet.

### 8.1.13 Timer

- **Felelősség**

Periodikus időt reprezentál, az időben léptethető dolgokat lépteti, singleton.

- **Attribútumok**

- **ArrayList<ISteppable> -iSteppables:** Az időben léptethető elemeket tárolja.
- **Timer -instance:** A singleton osztály példánya.

- **Metódusok**

- **+Timer():** A timer osztály konstruktora. Inicializálja az **-iSteppables** attribútumot egy üres ArrayList<ISteppable>-re.
- **void +AddISteppable(ISteppable iStoppable):** hozzáadja a paraméterben megadott ISteppable objektumot az **-iSteppable** paraméterben tárolt listához.
- **void +Tick():** Meghívja az **-iSteppables** attribútumban tárolt összes elemnek a **Step()** metódusát
- **Timer +GetInstance():** Hogyha az osztály még nem volt példányosítva, azaz az **-instance** attribútum **null**, létrehoz egy új példányt, amit megkap az **-instance** attribútum és visszatéríti. Különben visszatéríti az **-instance** attribútum értékét.

- **void +RemoveISteppable(ISteppable iSteppable):** eltávolítja a paraméterként megadott iSteppable objektumot
- **void +Clear():** Kiüríti az -iSteppables paraméterben tárolt listát.

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

[A tesztek részletes tervei alatt meg kell adni azokat a bemeneti adatsorozatokat, amelyekkel a program működése ellenőrizhető. minden bemenő adatsorozathoz definiálni kell, hogy az adatsorozat végrehajtásától a program mely részeinek, funkcióinak ellenőrzését várjuk és konkrétan milyen eredményekre számítunk, ezek az eredmények hogyan vethetők össze a bemenetekkel. A tesztek leírásakor az előző dokumentumban (proto koncepciója) megadott szintakszist kell használni.]

### 8.2.1 Player moves to sticky Pipe and tries to move again

- **Leírás:** A teszt egy játékos ragadós csőre való lépését, és utána a továbblépésre való próbálkozását ellenőrzi. Ennek során létrehozunk egy játékest, egy csövet és két pumpát, amik közé bekötjük a csövet. A játékest ráléptetjük az egyik pumpára, a csövet pedig csúszóssá tesszük. Ezután a játékos rálép a ragadós csőre, ami sikerülni fog, majd átlépni a másik pumpára, de ez nem fog sikerülni, mivel a játékos odaragadt a csőhöz.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** Ellenőrizzük, hogy a játékos nem tud továbblépni a ragadós csőről, helyzete a cső marad. Hiba, ha a játékos helyzete megváltozik.
- **Bemenet**

```
//init
add Pump on
add Pump to
add Pipe pip
add Mechanic m
connect on pip to
sticky pip
move m on

//action
move m pip
move m to

//state queries
state m
state -os pip
state -o to
```

- **Elvárt kimenet**

```
on: pip
holdingPipe: null
holdingPumps:

Player: m
isSticky: true
```

Players:

### 8.2.2 Player moves to slippery pipe from pump

- **Leírás:** A teszt egy játékos csúszós csőre való lépését teszteli két pumpa és egy ezeket összekötő csúszóssá tett cső segítségével. Kezdetben az egyik pumpán áll a játékos, és a cél a másik pumpára való átkerülés.
- **Ellenőrzött funkcionális, várható hibahelyek:** A játékos valóban a csúszós cső másik szomszédjára kerül a lépése után, és nem pedig a csúszós csőre. Továbbá, hogy a kezdeti pozíciójáról el lett távolítva.
- **Bemenet**

```
//init
add Pump on
add Pipe to
add Pump realTo
connect on to realTo
grease to
add Mechanic m
move m on
random off

//action
move m to

//state queries
state m
state -op on
state -oel to
state -op realTo
```

- **Elvárt kimenet**

```
on: realTo
holdingPipe: null
holdingPumps:

Players:
ConnectedPipes: to

Player:
End1: on
End2: realTo
isSlippery: true

Players: m
ConnectedPipes: to
```

### 8.2.3 Player switches pump and steps it

- **Leírás:**

A teszt egy pumpa átállítását és az ezutáni pumpálását teszteli, abban az esetben, amikor az átállítás után a bemeneti csőben van víz.

- **Ellenőrzött funkcionalitás, várható hibahelyek:**

A pumpa bemeneti és kimeneti csövének átállítása, a bemeneti csőből a víz eltávolítása, a pumpa víztartályának megtelése.  
Lehetséges, hogy a pumpa nem állítja át a belső állapotát, így a bemeneti csövén nem lesz víz, azaz nem történik semmi. Lehetséges, hogy a bemeneti csőben benne marad a víz. Lehetséges, hogy a víztartály nem telik meg és a víz továbbfolyik.

- **Bemenet**

```
//init
add Pump pump
add Pump pump2
add Pipe pip1
add Pipe pip2
add Pipe pip3
add Pipe pip4
connect pump pip1
connect pump pip2
connect pump pip3 pump2
connect pump pip4
switch pump pip1 pip2
fill pip3

//action
switch pump pip3 pip4
step pump

//state queries
state pump
state -ew pip3
state -ew pip4
```

- **Elvárt kimenet**

```
Players:
ConnectedPipes: pip1 pip2 pip3 pip4
inPipe: pip3
outPipe: pip4
isBroken: false
TankFull: true

End1: pump
End2: pump2
hasWater: false

End1: pump
End2: null
hasWater: false
```

### 8.2.4 Pump with empty tank steps 2

- **Leírás:** A teszt egy pumpa működését teszteli abban az esetben, amikor működőképes/nem törött állapotban van, viszont az átmeneti tárolója üres. A két hozzá csatlakoztatott pumpa segítségével tudjuk ezt ellenőrizni úgy, hogy a bemenetként beállított csőbe vizet töltünk, majd kétszer léptetjük a pumpát. A két cső szabad végeire még rá kell kapcsolnunk egy-egy pumpát azért, hogy a működés

során ne folyjon el víz a csövekből. Mivel ezek a pumpák csak segédfunkciót látnak el, az Ő állapotukat figyelmen kívül hagyjuk.

- **Ellenőrzött funkcionalitás, várható hibahelyek:** A pumpa ilyen esetben elvárt funkcionálitása, hogy az első ütemben csak feltölti az átmeneti tárolóját a beszívott vízzel, és csak a második lépéstre juttatja el a vizet a kimeneti csőbe. Hibás működésre utal, ha már az első lépést végére, vagy ha még a második ütemre sem érkezik meg a kimeneti csőbe a víz.
- **Bemenet**

```
//init
add Pump pump
add Pipe pip1
add Pipe pip2
add Pump helppump1
add Pump helppump2
connect pump pip1 helppump1
connect pump pip2 helppump2
switch pump pip1 pip2
fill pip1

//1st step
step pump
state -ew pip1
state -pfta pump
state -ew pip2

//2nd step
step pump
state -ew pip1
state -pfta pump
state -ew pip2
```

- **Elvárt kimenet**

```
End1: pump
End2: helppump1
hasWater: false

ConnectedPipes: pip1 pip2
InPipe: pip1
OutPipe: pip2
TankFull: true

End1: pump
End2: helppump2
hasWater: false

End1: pump
End2: helppump1
hasWater: false

ConnectedPipes: pip1 pip2
InPipe: pip1
OutPipe: pip2
TankFull: false

End1: pump
End2: helppump2
hasWater: true
```

### 8.2.5 Pipe network clogs

- **Leírás:** A teszt célja, hogy ellenőrizze, hogy az jelenség, amikor egy adott csőhálózat feltelik vízzel és eldugul, megfelelően működik. Ezt úgy tesszük meg, hogy létrehozunk egy pályát, ami tartalmaz egy pumpát, két hozzá kapcsolt csövet és egy forrást. A pumpának a bemenetét és kimenetét értelemszerűen ezekre a csövekre állítjuk, majd elkezdünk folyamatosan pumpálni és vizet tölteni a bemeneti csőbe.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** Azt várjuk el a felépített hálózatuktól, hogy 2 lépés után fel legyen telve vízzel, mivel a kimeneti csőből a vizet nem szívja ki semmi. Hibalehetőség lehet, ha a hálózat nem telik fel, nem dugul el és lehet a végtelenségig vizet tölteni a bemeneti csőbe.
- **Bemenet**

```
//inicializálás
add Pipe p1
add Pipe p2
add Pump pump
add Pump pump_plug
add Source s
connect s p1
connect pump_plug p2
connect pump p1
connect pump p2
switch pump p1 p2

//tesztelés
step s
step pump
step s
step pump
step s
step pump
state -w p1
state -a pump
state -w p2
```

- **Elvárt kimenet**

```
hasWater: true
TankFull: true
hasWater: true
```

### 8.2.6 Source steps

- **Leírás:** A teszt célja, hogy ellenőrizze, megfelelően működnek-e a források. Jelen esetben ezt úgy tesszük, hogy létrehozunk két üres, nem lyukas csövet, amit hozzákötünk a forráshoz. Ezután léptetjük a forrást, amely során vizet pumpál a belekötött összes csőbe.

- **Ellenőrzött funkcionalitás, várható hibahelyek:** Azt várjuk el a forrástól, hogy megtöltsé vízzel az összes hozzákötött csövet. Hibalehetőség lehet, hogy a hozzákötött csövekből valamelyik üres marad.
- **Bemenet**

```
//init
add Source s
add Pipe pip1
add Pipe pip2
connect s pip1
connect s pip2

//step
step s
state -w pip1
state -w pip2
```

- **Elvárt kimenet**

```
hasWater:true

hasWater:true
```

### 8.2.7 Cistern steps

- **Leírás:** A teszt célja, hogy ellenőrizze, hogy megfelelően működnek-e a ciszternák. Ezt úgy tesszük meg, hogy létrehozunk két üres, nem lyukas csövet és egy ciszternát. A csövegeteket hozzákötjük a ciszternához. Ezután az első csőbe vizet töltünk, majd léptetjük a ciszternát.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** Azt várjuk el, hogy a szerelők kapjanak egy pontot, se többet, se kevesebbet. Az előbbi kettő hibalehetőség.
- **Bemenet:**

```
//inicializálás
add Cistern c
add Source s
add Pipe p1
add Pipe p2
connect s p1
connect c p1
connect c p2

//tesztelés
step s
step c
state pools
```

- **Elvárt kimenet**

```
Mechanics: 1
Saboteurs: 0
```

### 8.2.8 Pump steps

- **Leírás:**

A teszt egy pumpa működését teszteli abban az esetben, amikor a bemeneti csövén van víz és a víztartály üres.

- **Ellenőrzött funkcionalitás, várható hibahelyek:**

A bemeneti csőből a víz eltávolításának tesztelése, valamint a víztartály viselkedésének tesztelése.

Lehetséges, hogy a bemeneti csőben benne marad a víz. Lehetséges, hogy rögtön a kimeneti csőbe kerül a víz, a víztartály megkerülésével.

- **Bemenet**

```
//init
add Pump pump
add Pump pump2
add Pipe pip1
add Pipe pip2
connect pump pip1 pump2
connect pump pip2
switch pump pip1 pip2
fill pip1
```

```
//action
step pump
```

```
//state queries
state pump
state -ew pip1
state -ew pip2
```

- **Elvárt kimenet**

```
Players:
ConnectedPipes: pip1 pip2
InPipe: pip1
OutPipe: pip2
isBroken: false
TankFull: true
```

```
End1: pump
End2: pump2
hasWater: false
```

```
End1: pump
End2: null
hasWater: false
```

### 8.2.9 Broken, empty Pump steps

- **Leírás:** A teszt egy pumpa működését teszteli abban az esetben, amikor az nem működőképes/törött, és átmeneti tárolója is üres. A két hozzá csatlakoztatott cső segítségével tudjuk ellenőrizni úgy, hogy a bemenetként beállított csőbe vizet töltünk, a pumpát eltörjük/elrontjuk, majd kétszer léptetjük a pumpát. A két cső szabad végeire még rá kell kapcsolnunk egy-egy pumpát azért, hogy a működés során ne folyjon el víz a csövekből. Mivel ezek a pumpák csak segédfunkciót látnak el, az ő állapotukat figyelmen kívül hagyjuk.

- **Ellenőrzött funkcionális, várható hibahelyek:** A pumpa ilyen esetben elvárt funkcionálitása, hogy nem szívja ki a bemeneti csőben lévő vizet, és átmeneti tárolója és a kimeneti cső is üresen marad. Hibás működésre utal, ha a bemeneti csőből eltűnik a víz, az átmeneti tároló megtelik, vagy a kimeneti csőbe jut víz.
- **Bemenet**

```
//init
add Pump pump
add Pipe pip1
add Pipe pip2
add Pump helppump1
add Pump helppump2
connect pump pip1 helppump1
connect pump pip2 helppump2
switch pump pip1 pip2
break pump
fill pip1

//step
step pump
state -ew pip1
state -pftba pump
state -ew pip2
```

- **Elvárt kimenet**

```
End1: pump
End2: helppump1
hasWater: true

ConnectedPipes: pip1 pip2
InPipe: pip1
OutPipe: pip2
isBroken: true
TankFull: false

End1: pump
End2: helppump2
hasWater: false
```

### 8.2.10 Player connects pipe to Pump

- **Leírás:** A teszt célja, hogy megbizonyosodjunk arról, hogy egy játékos képes a kezében tartott csővéget csatlakoztatni egy már létrehozott pumpához, amin a játékos áll és amin van szabad hely.
- **Ellenőrzött funkcionális, várható hibahelyek:** A játékos kezéből valóban eltűnt a korábban tartott csővég. Pumpához valóban lehet csővéget csatlakoztatni, ha azon van még szabad hely.
- **Bemenet**

```
//init
add Pump on
add Pipe holdingPipe
add Mechanic m
move m on
holdpipe holdingPipe m
```

```
//action
connectpipe m

//state queries
state m
state -op on
```

- **Elvárt kimenet**

```
on: on
holdingPipe: null
holdingPumps:

Players: m
ConnectedPipes: holdingPipe
```

### 8.2.11 Player connects Pipe to Cistern

- **Leírás:** A teszt célja, hogy megbizonyosodjunk arról, hogy a ciszternához jól lehet csövet csatlakoztatni. Ezt úgy tesszük meg, hogy létrehozunk egy ciszternát, egy játékos, aki csatlakoztatni fog, és egy csövet. A cső végét odaadjuk a játékos kezébe, és ráléptetjük őt a ciszternára. Ezután megpróbáljuk csatlakoztatni a ciszternához a csövet.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** Azt várjuk el, hogy a ciszternához kötött csövek között megjelenjen a csatlakoztatott cső, valamint a csőhöz csatlakoztatott csomópontok között jelenjen meg a ciszterna, és a játékos kezéből tűnjön el a csővég. Hiba, ha a ciszternához csatlakoztatott csövek között nincs ott a cső, vagy a csőhöz csatlakoztatott csomópontok között nincs ott a ciszterna, vagy a játékos kezében maradt a cső vége.
- **Bemenet**

```
//init
add Cistern c
add Mechanic m
move m c
add Pipe pip
holdpipe pip m
connectpipe m

state c
state -e pip
state m
```

- **Elvárt kimenet**

```
Player: m
ConnectedPipes: pip

End1: c
End2: null

on: c
holdingPipe: null
holdingPumps:
```

### 8.2.12 Player connects Pipe to Source

- **Leírás:** A teszt célja az, hogy leellenőrizze, helyesen működik-e egy cső felcsatlakoztatása egy forrásra a játékos által. Ezt úgy teszi meg, hogy létrehoz egy pályát, ami egy forrást, egy csövet és egy játékost tartalmaz, aki csatlakoztatni fog. A cső végét odaadjuk a játékosnak, ráléptetjük őt a forrásra. Ezután megpróbáljuk csatlakoztatni a csövet. Végül léptetjük a forrást.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** Azt várjuk el, hogy a forráshoz csatlakoztatott csövek között megjelenjen a létrehozott cső, valamint a csőhöz csatlakoztatott aktív elemek között megjelenjen a forrás, ezenkívül eltűnjön a játékos kezéből a csővég. Léptetés után elvárjuk, hogy a csőben legyen víz. Hibalehetőség, ha a forráshoz csatlakoztatott csövek között nincs ott a létrehozott cső, a cső szomszédai között nincs ott a forrás, a csővég nem tűnik el a játékos kezéből vagy a csőbe nem kerül víz.
- **Bemenet:**

```
//inicIALIZÁLÁS
add Source s
add Mechanic m
add Pipe p
move m s
holdpipe p m

//tesztelés
connectpipe m
connect plug p
step s
state s
state -ew p
state m
```

- **Elvárt kimenet:**

```
Player: m
ConnectedPipes: p

End1: s
End2: plug
hasWater: true

on: s
holdingPipe: null
holdingPumps:
```

### 8.2.13 Player connects Pipe to Pipe

- **Leírás:** A teszt célja, hogy megbizonyosodjunk arról, hogy a csöveget nem lehet csövekkel összekötni. Ezt úgy tesszük meg, hogy létrehozunk két pumpát, közéjük bekötünk egy csövet, amire ráléptetünk egy játékost (miután azt is létrehoztuk), és

egy másik cső végét a kezébe adjuk. Ezután megpróbáljuk csatlakoztatni a kezében lévő csövet a csőhöz, amin áll, ami nem fog sikerülni.

- **Ellenőrzött funkcionalitás, várható hibahelyek:** Azt várjuk el, hogy a játékos kezében maradjon a csővég, a helyzete maradjon a cső és a csőhöz csatlakoztatott elemek továbbra is a két pumpa legyen. Hiba, ha a játékos kezéből eltűnik a cső, megláltozik a játékos helyzete vagy ha megjelenik a kézben lévő cső a csőhöz csatlakoztatott Node-ok között.
- **Bemenet**

```
//init
add Pump p1
add Pump p2
add Pipe on
add Pipe hand
add Mechanic m
connect p1 on p2
move m on
holdpipe hand m
connectpipe m

state m
state -oe on
```

- **Elvárt kimenet**

```
on: on
holdingPipe: hand
holdingPumps:

Player: m
End1: p1
End2: p2
```

### 8.2.14 Player disconnects empty Pipe

- **Leírás:** A teszt egy cső leválasztásának hatását teszteli abban az esetben, amikor a leválasztott cső üres volt. Ezt úgy teszteljük, hogy egy pumpához kötünk egy üres, nem lyukas csövet, majd leválasztjuk a pumpára rakott szerelő által. A szerelő karakter helyett tesztelhetnénk a szabotőr karakterrel is a csőleválasztást, de a teszteset szempontjából ez lényegtelen.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** Az elvárt funkcionalitás az, hogy a cső ugyanúgy üres mint leválasztás előtt, valamint nem folyt el belőle víz. Ezt úgy ellenőrizzük, hogy a szabotőrök pontja a kezdő 0 értéken maradt a leválasztás után. A teszt sikertelen, ha elfolyt víz, tehát a szabotőrök szereztek vizet/pontot a leválasztással.
- **Bemenet**

```
//init
add Pump pump
add Pipe pip
add Mechanic m
move m pump
```

```
disconnectpipe m pip
state -ew pip
state -p pump
state pools
```

- **Elvárt kimenet**

```
End1: null
End2: null
hasWater: false

ConnectedPipes:

Mechanics: 0
Saboteurs: 0
```

### 8.2.15. Player disconnects full pipe

- **Leírás:** A teszteset célja, hogy megbizonyosodjunk arról, hogy ha egy játékos egy tele csövet próbál lecsatlakoztatni egy pumpáról, ahol éppen áll, akkor ez sikerül neki, és a szabotőrök pontját növeli.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** A cső leválasztásánál valóban kifolyik a víz, így a szabotőrök pontot kapnak, illetve, hogy a lekötött csővég a játékos kezében marad.
- **Bemenet**

```
//init
add Pump on
add Pump pump
add Pipe disconnected
connect on disconnected pump
fill disconnected
add Mechanic m
move m on

//action
disconnectpipe m disconnected

//state queries
state -op on
state -p pump
state -ew disconnected
state m
state pools
```

- **Elvárt kimenet**

```
Players: m
ConnectedPipes:

ConnectedPipes: disconnected

End1: null
End2: pump
hasWater: false

on: on
```

```
holdingPipe: disconnected
holdingPumps:
```

```
Mechanic: 0
Saboteur: 1
```

### 8.2.16 Player breaks Pipe

- **Leírás:** A teszt célja, hogy ellenőrizze a lyukas csövek megfelelő működését. Ennek során létrehozunk egy játékos, egy pumpát és egy hozzá kötött csövet, amit beállítunk kimeneti csőnek. A játékos ráléptetjük a csőre és megtöljük a pumpa víztartályát vizsel. A játékos megpróbálja kilyukasztani a csövet, ami sikerül neki, majd léptetjük egyszer a pumpát, amiből ki kell folyjon a víz.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** A lyukasztás után a pumpa állapotában meg kell jelenjen, hogy lyukas, és a pumpa léptetése után sem tartalmazhat vizet. Emellett a szabotőrök Pool-jában 1 pont meg kell jelenjen.
- **Bemenet**

```
//inicializálás
add Pump pump
add Pipe p1
connect pump p1
switch pump null p1
fill pump
add Mechanic m
move m p1

//tesztelés
leakpipe m
step pump
state -bw p1
state pools
```

- **Elvárt kimenet**

```
isBroken: true
hasWater: false

Mechanics: 0
Saboteurs: 1
```

### 8.2.17 Mechanic repairs broken Pump

- **Leírás:** A teszt célja, hogy ellenőrizze egy pumpa helyes működését, miután egy játékos megjavította. Ezt úgy tesszük meg, hogy létrehozunk egy pályát, ami egy pumpát, két hozzá kapcsolt csövet és egy forrást tartalmaz. Ezután beállítjuk a pumpát, hogy az első csőből szívjon és a másodikba pumpáljon. Az első csövet megtöljük vizsel, majd elrontjuk a pumpát. Ezután létrehozunk egy szerelőt, akit

ráléptetünk a pumpára. A játékos ekkor megjavítja a pumpát. Végül léptetjük a pumpát kétszer.

- **Ellenőrzött funkcionális, várható hibahelyek:** A pumpa a fentiek elvégzése után helyesen kell pumpáljon, azaz a szívócsőből el kell tűnjen a víz az első lépés után, majd a kimeneti csőben meg kell jelenjen a második lépés után. Hiba, ha ez nem így történik.
- **Bemenet**

```
//inicializálás
add Pump pump
add Pump pump_plug
add Pipe p1
add Pipe p2
add Source s
connect pump p1
connect pump p2
connect pump_plug p2
switch pump p1 p2
connect s p1
step s
add Mechanic m
move m pump

//tesztelés
repair pump m
state -b pump
step pump
state -w p1
state -w p2
step pump
state -w p1
state -w p2
```

- **Elvárt kimenet**

```
isBroken: false
hasWater: false
hasWater: false
hasWater: false
hasWater: true
```

## 8.2.18 Mechanic repairs leaking pipe

- **Leírás:**  
A teszt egy törött cső megjavítását teszteli, hogy a cső a megfelelő állapotba kerül-e.
- **Ellenőrzött funkcionális, várható hibahelyek:**  
Az elvárt funkcionális, hogy a cső törött állapotból a foltozott állapotba kerül. Hiba lehet, hogy a cső nem kerül át a foltozott állapotba.
- **Bemenet**

```
//init
add Pipe pip
add Mechanic m
move m pip
leak pip

repair pip m
```

- state -b pip
- **Elvárt kimenet**

```
isBroken: false
```

### 8.2.19 Mechanic repairs working Pump

- **Leírás:** A teszt egy pumpa megjavításának hatását teszteli abban az esetben, amikor a megjavítandó pumpa működőképes/nemtörött állapotban van. Ehhez a teszthez csak egy nem törött pumpát kell létrehoznunk, majd egy szerelő karaktert elhelyezünk a pumpán, és vele elvégeztetni a javítást.
- **Ellenőrzött funkcionális, várható hibahelyek:** Az elvárt funkcionális az, hogy a pumpa ugyanúgy működőképes/nem törött állapotban lesz a javítás után, tehát állapotváltozás nem történt az ép cső javításánál. Hibára utal, ha a pumpa eltör, vagy ha a szerelő nem áll a pumpán a javítás után.
- **Bemenet**

```
//init
add Pump pump
add Mechanic m
move m pump

repair pump m
state -ob pump
```

- **Elvárt kimenet**

```
Players: m
isBroken: false
```

### 8.2.20. Mechanic repairs undamaged pipe

- **Leírás:** A teszeset lényege, hogy egy szerelő megjavítson egy ép (nem lyukas) csövet, amin áll, és amelynek hatására nem történik változás a cső állapotában.
- **Ellenőrzött funkcionális, várható hibahelyek:** A cső állapota változatlan marad.
- **Bemenet**

```
//init
add Pipe on
add Mechanic m
move m on
//action
repair on m
//state queries
state -ob on
```

- **Elvárt kimenet**

```
Player: m
isBroken: false
```

### 8.2.21 Mechanic picks up Pump at Cistern

- **Leírás:** A teszt során azt ellenőrzük, hogy megfelelően működik a pumpák felvétele a ciszternákon. Ennek során létrehozunk egy ciszternát és egy szerelőt, akit ráléptetünk a ciszternára. Ezután a szerelő megpróbál felvenni egy pumpát, ami sikerül neki.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** Ellenőrzük, hogy a ciszterna legyártotta a pumpát és az a játékos kezébe került. Hiba, ha a játékosnál nincsen pumpa.
- **Bemenet:**

```
//inicializálás
add Cistern c
add Mechanic m
move m c
//tesztelés
pickup --pump p1 m
state m
```

- **Elvárt kimenet:**

```
on: c
holdingPipe: null
holdingPumps: p1
```

### 8.2.22 Mechanic picks up Pump at Source

- **Leírás:** A teszt célja az, hogy megmutassa, hogy a pumpa felvétele megfelelően működik a pályán. Ezt úgy teszi meg, hogy létrehoz egy pályát, ami egy forrást tartalmaz. Ezután létrehoz egy szerelőt és rálépteti a forrásra, ahol ez megpróbál felvenni egy pumpát.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** Megfelelő funkcionalitás esetén a játékos nem tud felvenni pumpát. Hibalehetőség, ha a játékos mégis fel tudott venni pumpát.
- **Bemenet:**

```
//inicializálás
add Source s
add Mechanic m
move m s
//tesztelés
pickup --pump p m
state m
```

- **Elvárt kimenet:**

```
on: s
holdingPipe: null
holdingPumps:
```

### 8.2.23 Mechanic picks up Pipe at Cistern

- **Leírás:** A teszt során azt ellenőrzük, hogy megfelelően működik a csövek felvétele a ciszternákon. Ennek során létrehozunk egy ciszternát és egy szerelőt, akit

ráléptetünk a ciszternára. Ezután a szerelő megpróbál felvenni egy csövet, ami sikerül neki.

- **Ellenőrzött funkcionalitás, várható hibahelyek:** Ellenőrizzük, hogy a ciszterna legyártotta a csövet és az a játékos kezébe került. Hiba, ha a játékosnál nincsen cső.
- **Bemenet:**

```
//inicializálás
add Cistern c
add Mechanic m
move m c

//tesztelés
pickup --pipe pip m
state m
```

- **Elvárt kimenet:**

```
on: c
holdingPipe: pip
holdingPumps:
```

### 8.2.24 Mechanic picks up Pipe at Source

- **Leírás:** A teszt egy cső felvételének hatását teszeli olyan esetben, amikor a szerelő a csőfelvételt egy forráson próbálja végrehajtani. Ezt úgy teszteljük, hogy létrehozunk egy forrást és egy szerelőt, majd rátesszük a szerelő karaktert a forrásra.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** Az elvárt funkcionalitás az, hogy a csőfelvétel kezdeményezésnek ne legyen hatása se a forráshoz csatlakozó csövek, se a szerelő kezében lévő csövek számára. Mivel a forrás nem tud csövet gyártani, a forrásnál nem lesz új bekötött cső, és a szerelő kezébe sem kerül új cső. Hibára utal, ha a létezik a forráshoz csatlakozó cső, vagy ha a szerelő kezébe került cső.
- **Bemenet**

```
//init
add Source source
add Mechanic m
move m source

pickup --pipe new m
state source
state m
```

- **Elvárt kimenet**

```
Players: m
ConnectedPipes:

on: source
holdingPipe: null
holdingPumps:
```

### 8.2.25 Mechanic places Pump on Pipe

- **Leírás:** A teszeset célja, hogy megbizonyosodjunk arról, hogy egy szerelő képes letenni egy nála lévő pumpát, arra a már létrehozott csőre (ez alapesetben üres, ép, nem csúszós, nem ragadós), amin éppen áll.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** A szerelő átkerült a lerakott pumpára, és pumpái közül eltűnik a lerakott pumpa. Arról a csőről, ahol a szerelő eredetileg állt, lekerül a szerelő, és megváltozik az egyik szomszédja a lerakott pumpára.
- **Bemenet**

```
//init
add Pump pump1
add Pump pump2
add Pipe on
connect pump1 on pump2
add Pump pickedUp
add Mechanic m
move m on
holdpump pickedUp m
//action
placepump m
//state queries
state m
state -oe on
state -op pickedUp
```

- **Elvárt kimenet**

```
on: pickedUp
holdingPipe: null
holdingPumps:

Player:
End1: pump1
End2: pickedUp

Players: m
ConnectedPipes: on newPipe
```

### 8.2.26 Mechanic places Pump on Cistern

- **Leírás:** A teszt célja az, hogy ellenőrizze, hogy egy pumpa lerakása a pályán megfelelően működik. Ezt úgy ellenőrzük, hogy létrehozunk egy pályát, ami egy ciszternát tartalmaz és egy szerelőt. A szerelő rááll a ciszternára, felvesz egy pumpát majd megpróbálja lerakni.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** Az elvárt működés az, hogy a szerelő ne tudja letenni a pumpát és a kezében marad. Hiba, ha mégis le tudja és/vagy eltűnik a kezéből.
- **Bemenet:**

```
//inicjalizálás
add Cistern c
add Mechanic m
move m c
pickup --pump pump m

//tesztelés
```

```
placepump m
state m
```

- **Elvárt kimenet:**

```
on: c
holdingPipe: null
holdingPumps: pump
```

### 8.2.27 Mechanic places Pump on Source

- **Leírás:** A teszt célja az, hogy ellenőrizze, hogy egy pumpa lerakása a pályán megfelelően működik. Ezt úgy ellenőrzük, hogy létrehozunk egy pályát, ami egy ciszternát tartalmaz, egy forrást és egy szerelőt. A szerelő rááll a ciszternára, felvesz egy pumpát, átmegy a forrásra majd megpróbálja lerakni.
- **Ellenőrzött funkcionális, várható hibahelyek:** Az elvárt működés az, hogy a szerelő ne tudja letenni a pumpát és a kezében marad. Hiba, ha mégis le tudja és/vagy eltűnik a kezéből.
- **Bemenet:**

```
//inicializálás
add Source s
add Mechanic m
add Pump pump
holdpump pump m
move m s
//tesztelés
placepump m
state m
```

- **Elvárt kimenet:**

```
on: s
holdingPipe: null
holdingPumps: pump
```

### 8.2.28 Mechanic places Pump on Pump

- **Leírás:** A teszt egy ciszternánál felvett pumpa lerakását teszteli egy másik pumpán. A játékos áll egy ciszternán ahol felvesz egy pumpát majd átlép egy csövön keresztül egy pumpára és megpróbálja lerakni a felvett pumpát, ekkor nem történik semmi.
- **Ellenőrzött funkcionális, várható hibahelyek:** Az elvárt működés az, hogy nem történik semmi, azaz a játékosnál még mindig ott van a pumpa.
- **Bemenet:**

```
//inicializálás
add Cistern cistern
add Pump pump
add Pipe pip
connect cistern pip pump
add Mechanic m
move m cistern

//tesztelés
pickup --pump new m
```

```

move m pip
move m pump
placepump m
state m
● Elvárt kimenet:
on: pump
holdingPipe: null
holdingPumps: new

```

### 8.2.29 Leaking pipe gets water

- **Leírás:**

A teszt egy lyukas csőbe vezet vizet.

- **Ellenőrzött funkcionalitás, várható hibahelyek:**

Ellenőrzi, hogy a csőből tényleg kifolyik a víz és hogy ez megfelelő helyre kerül.

Lehetséges, hogy a víz nem folyik ki a csőből vagy a szabotőrök nem kapják meg a pontot.

- **Bemenet**

```

//init
add Pipe pip
leak pip

fill pip
state -w pip
state pools

```

- **Elvárt kimenet**

```

hasWater: false

Mechanics: 0
Saboteurs: 1

```

### 8.2.30. Undamaged pipe gets water

- **Leírás:** A teszeset egy ép (nem lyukas) cső vízfogadását teszteli, méghozzá, úgy hogy egy már létrehozott(ez alapesetben nem törött) pumpát összeköt az előbb említett csővel (ami alapesetben üres is), beállítja azt a kimeneti csövének, lépteti a pumpát, amelynek tartálya tele van, végül pedig a cső fogadja a vizet.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** A cső valóban tudja-e fogadni a vizet, azaz üres állapotból teli állapotba kerül. A pumpa tartálya valóban kiürül-e.
- **Bemenet**

```

//init
add Pump pump1
add Pipe pipe
add Pump pump2
connect pump1 pipe pump2
switch pump1 null pipe
fill pump1
//action
step pump1
//state queries

```

```
state -pta pump1
state -w pipe
```

- **Elvárt kimenet**

```
ConnectedPipes: pipe
OutPipe: pipe
TankFull: false

hasWater: true
```

### 8.2.31 Player disconnects Pump's output

- **Leírás:** A teszt célja az, hogy ellenőrizze, hogy mi történik akkor, ha egy pumpa kimeneti csövét lecsatlakoztatják. Ezt úgy vizsgáljuk meg, hogy létrehozunk egy pályát, ami egy pumpát tartalmaz és két hozzáköött csövet. Beállítjuk a pumpa bemenetét az első, kimenetét a második csőre, majd vizet töltünk az első csőbe és léptetjük a pumpát. Létrehozunk egy szerelőt, aki segítségével ezután lecsatlakoztatjuk a kimeneti csövet, vizet töltünk a bemeneti csőbe és megint léptetjük a pumpát.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** Az elvárt működés az, hogy a pumpa ilyenkor nem tud pumpálni és a bemenetről sem szív vizet.
- **Bemenet:**

```
//inicializálás
add Pump pump
add Pipe p1
add Pipe p2
add Mechanic m
add Source s
connect s p1
move m pump
connect pump p1
connect pump p2
switch pump p1 p2
step s

//tesztelés
step pump
disconnectpipe m p2
fill p1
step pump
state pump
state -w p1
```

- **Elvárt kimenet:**

```
Players: m
ConnectedPipes: p1
inPipe: p1
outPipe: null
isBroken: false
TankFull: true

hasWater: true
```

### 8.2.32 Player leaks out pipe and pump steps one

- **Leírás:**

Létrejön egy 3 csőből és 2 pumpából álló sorozat, az első csőbe víz kerül, az első pumpa víztartályába víz kerül és egy játékos kilyukasztja a második csövet. Ezután léptetjük az első majd a második pumpát.

- **Ellenőrzött funkcionális, várható hibahelyek:**

A teszt célja, hogy megbizonyosodjunk arról, hogy egy pumpa bemeneti csövének kilyukasztása megfelelő hatással van a pumpa viselkedésére, valamint egy olyan pumpa, amelynek tele van a víztartálya és van a bemeneti csövén víz pumpál-e a kimeneti csövébe. Lehetséges, hogy továbbfolyik a víz a második pumpába és a harmadik csőbe.

- **Bemenet:**

```
//inicializálás
add Pipe pip1
add Pipe pip2
add Pipe pip3
add Pump pump1
add Pump pump2
connect pump1 pip1
connect pump1 pip2 pump2
connect pump2 pip3
fill pip1
fill pump1
add Mechanic m
move m pip2
leakpipe m

//tesztelés
step pump1
step pump2
state -a pump2
state -w pip3
state pools
```

- **Elvárt kimenet:**

```
TankFull: false
hasWater: false
Mechanics: 0
Saboteurs: 1
```

### 8.2.33. Mechanic repairs broken pump and it steps 2

- **Leírás:** A teszt célja, hogy megbizonyosodjunk arról, hogy egy szerelő képes megjavítani egy törött pumpát, amin áll, majd az immáron megjavított pumpa léptetése helyesen működik, azaz a már létrehozott, és a pumpának a bemeneti csöveként beállított megtöltött csőből a pumpa kiszívja a vizet, amely az átmeneti tárolójába (ami alapesetben üres) kerül, majd a második léptetés során, a pumpa kipumpálja a vizet a már létrehozott, a pumpa kimeneti csöveként beállított csőbe, így eltávolítva azt a víztartályából.

- **Ellenőrzött funkcionalitás, várható hibahelyek:** A pumpa állapota megváltozott nem töröttré. A bemeneti csőből és a tartályból kikerült a víz, míg a kimeneti cső megtelt vízzel.
- **Bemenet:**

```
//init
add Pump on
add Pump pump1
add Pump pump2
add Pipe inPipe
add Pipe outPipe
connect pump1 inPipe on
connect on outPipe pump2
switch on inPipe outPipe
fill inPipe
break on
add Mechanic m
move m on
//action
repair on m
step on
step on
//state queries
state -w inPipe
state -opftba on
state -w outPipe
```

- **Elvárt kimenet:**

```
hasWater: false

Players: m
ConnectedPipes: inPipe outPipe
InPipe: inPipe
OutPipe: outPipe
isBroken: false
TankFull: false

hasWater: true
```

### 8.2.34. Pump breaks with full tank and it steps 2

- **Leírás:** A teszeset célja, hogy ellenőrizze egy olyan pumpa működését, aminek átmeneti tárolója tele volt, mikor meghibásodott. A teszesetben létrehozunk egy pumpát, megtöljük az átmeneti tárolóját, „eltörjük”/meghibásodott állapotba hozzuk, majd hozzácsatlakoztatunk két csövet, melyeket aztán be is állítunk rajta be- és kimeneti csőként. Ezután a bemeneti csövét megtöljük vízzel. Végül léptetjük kettőt a pumpákat.
- **Ellenőrzött funkcionalitás, várható hibahelyek:** Ebben a helyzetben a pumpától azt várjuk el, hogy az első léptetésre engedje ki az átmeneti tárolójából a vizet a kimeneti csőbe, de a bemenetről ne szívjon vizet. A második ütemben a kiürült tárolóval, eltörve már ne végezzen semmilyen műveletet. Hibás működésre utal, ha a második léptetés után az átmeneti tárolója nem üres, ha a bemeneti csőből kiürült a víz, esetleg a kimeneti cső üres maradt.
- **Bemenet**

```
//init
add Pump pump
add Pipe pip1
add Pipe pip2
add Pump help1
add Pump help2
connect pump pip1 help1
connect pump pip2 help2
switch pump pip1 pip2
fill pump
fill pip1

//break and step 2
break pump
step pump
step pump
state -w pip1
state -pftba pump
state -w pip2
```

- **Elvárt kimenet**

```
hasWater: true

ConnectedPipes: pip1 pip2
InPipe: pip1
OutPipe: pip2
isBroken: true
TankFull: false

hasWater: true
```

## 8.3 A tesztelést támogató programok tervei

A tesztelő programot egy Runtest nevű Powershell szkript fogja megvalósítani. Kiterjesztése ps1 lesz, tehát a futtatható fájl neve Runtest.ps1. A program parancssorból futtatható lesz, és parancssor argumentumokat is meg lehet neki adni. Mivel a szkriptet mi írtuk, nem rendelkezik digitális aláírással, így operációs rendszerünk nem fogja engedni futtatni. Ezt orvosolandó, át kell állítani a végrehajtási házirendet. Ezt a következő parancssal tehetjük meg adminisztrátori jogosultsággal.

```
Set-ExecutionPolicy Unrestricted
```

A szkript futtatása után célszerű visszaállítani az eredeti, biztonságosabb beállítást.

```
Set-ExecutionPolicy RemoteSigned
```

Program pontosan egy argumentumot vár, aminek egy 1-től 34-ig terjedő egész számnak kell lennie. Ezek a számok megfelelnek az előre definiált teszteseteknek, amelyek a játék prototípusának elvárt viselkedését, funkcionalitásait hivatottak ellenőrizni. A tesztesetek sorszámozása megegyezik jelen dokumentumban lévő számozással. Ha például a 10-es számot adjuk meg futtatáskor, akkor a program a 10-es sorszámú tesztesetet végzi el, és a

kimenetet értesít minket a teszt sikereségéről és az esetleges eltérésekről az elvárt kimenethez képest. Ezután a program abbahagyja futását. Azt is megtehetjük, hogy nem adunk meg egyetlen számot se futtatáskor; ekkor a tesztelő program minden a 34 tesztesetet elvégzi, és visszajelzést ad a tesztesetek sikereségéről. Ebben az esetben a konkrét eltérésekről nem tájékoztat a program, mert az esetlegesen nagyszámú hibától a kimenet átláthatatlan lenne.

Tehát az összes teszteset végrehajtásához a program futtatása parancssorból:

```
>> Rунтест.ps1
```

Ha csak egy tesztesetet szeretnénk végrehajtani, például a 10-est:

```
>> Rунтест.ps1 10
```

Kimenet sikeres teszt esetén:

```
>> A 10-es sorszámu teszteset hiba nélkül lefutott!
```

Kimenet sikertelen teszt esetén:

```
>> A 10-es sorszámu teszteset sikertelen volt!
```

```
>> Found in Line 2: <text>
```

```
>> Expected in Line 2: <text>
```

A tesztelő program működése három alapvető fontosságú műveletből áll. Először beolvassa az argumentumként megadott tesztesethez tartozó bemenetet, amit egy inputXX.txt nevű szöveges fájlból olvas be. A fájl nevének vége megegyezik azon teszteset sorszámaival, aminek a bemenetét tartalmazza. A 10-es teszteset bemenetét például az input10.txt fájl, a 3-as teszteset bemenetét az input03.txt nevű fájl tartalmazza. Ezek a bemeneti fájlok a prototípus program általunk definiált nyelvben lévő parancsokat tartalmaz, valamint kommenteket. A parancsok közötti kommenteket a prototípus program nem veszi figyelembe. A bemeneti fájlokat a tesztelő szkript abban a mappában keresi, ahol ő maga is el van helyezve. A parancsokat tartalmazó fájlok megfelelő elhelyezéséről a tesztelőnek kell gondoskodni a program futtatása/indítása előtt. Ha az összes tesztesetet hajtjuk végre, akkor a segédprogram 34 különböző fájl tartalmát olvassa be.

A második alapvető művelet a prototípus program futtatása a beolvasott fájlban lévő parancsokkal. Ha csak egy tesztesetet hajtunk végre, akkor egyszer futtatja a segédprogram

a prototípust. Ha az összes tesztesetet végre akarjuk hajtani, akkor összesen 34-szer kell futtatni a prototípust 34 különböző bemenettel.

Végül a prototípus futtatása után a prototípus program kimenetét összehasonlítja az adott teszteset elvárt kimenetével. Az elvárt kimenetet szintén szöveges fájlokból olvassa be, melyek a következő elnevezési konvenciót kell követniük: az elvárt kimenetet tartalmazó fájl neve outputXX elnevezésűnek kell lennie, ahol az XX a teszteset sorszáma. Például a 4-es teszteset elvárt kimenetét az output04.txt fájlban fogja keresni a tesztelő program. Ezen szöveges fájloknak is azonos mappába kell lenniük a szkripttel.

Az elvárt kimenet beolvasása után végül összehasonlítja egymással soronként az elvárt és tényleges kimenetet a segédprogram. Karakterre pontosan ellenőriz, tehát egyetlen egy karakter-eltérés esetén is hibát jelez.

## 8.4 Napló

Kezdet	Időtartam	Résznevők	Leírás
2023. 04.26. 12:00	1 óra	Kiss Andó Skáre Deé-Lukács Vörös	Értekezlet: feladatok felosztása, bementi és kimeneti nyelv javítása: comment, state javítása, holdpipe, holdpump parancsok felvétele  Andó: Osztályok leírása: Node, Element, Source Tesztesetek: 2., 10., 15., 20., 25., 30., 33.  Deé-Lukács: Osztályleírás: PipeEnd, Pool, Timer Tesztesetek: 5, 7, 12, 17, 22, 26, 27, 31  Kiss: Osztályleírás: Player, Mechanic, Game

			<p>Tesztesetek: 1, 6, 11, 13, 16, 21, 23</p> <p>Skáre: Osztályleírás: Pipe, Saboteur</p> <p>Tesztesetek: 3, 8, 18, 28, 29, 32</p> <p>Vörös: bemeneti és kimeneti nyelv bővítése, javítása, tesztelő program tervezése, Pump, Cistern osztály leírása, tesztesetek leírása: 4, 9, 14, 19, 24, 34</p>
2023.04.27. 15:00	2, 5 óra	Vörös	bemeneti és kimeneti nyelv bővítése, javítása, Pump, Cistern osztály leírása, tesztesetek leírása: 4, 9, 14, 19, 24, 34
2023.04.29. 10.00	3 óra	Kiss	Osztályleírás: Player, Mechanic, Game-ben: nem parancshoz kapcsolódó metódusok és Add, Drain, HoldPipe, HoldPump, Step
2023. 04. 29. 14:30	2,5 óra	Deé-Lukács	Osztályleírás: PipeEnd, Pool, Timer
2023.04.29. 22:00	2 óra	Andó	Element, Source, Node osztályok leírása
2023.04.30. 11.00	2,5 óra	Kiss	Tesztesetek: 1. Player moves to sticky pipe and tries to move again 6. Source steps 11. Player connects pipe to Cistern 13. Player connects pipe to Pipe 16. Player breaks pipe

			21. Mechanic picks up Pump at Cistern 23. Mechanic picks up Pipe a Cistern
2023. 04. 30. 13.30	2,5 óra	Deé-Lukács	Tesztek megírása: Pipe network clogs Cistern steps Player connects pipe to Source Mechanic repairs broken pump Mechanic picks up Pump at Source Mechanic places Pump on Cistern Mechanic places Pump on Source Player disconnects outpipe from pump and pump steps
2023.04.30. 16:00	2,5 óra	Andó	A 2., 10. 15., 20., 25., 30., 33. tesztesetek megírása
2023.04.30. 19:00	1 óra	Vörös	Tesztelő program terve
2023.05.01. 23:00	2 óra	Skáre	Osztályleírás: Pipe, Saboteur
2023.05.02. 01:00	1,5 óra	Skáre	Tesztesetek megírása: <ul style="list-style-type: none"> <li>• Player switches pump and steps it</li> <li>• Pump steps</li> <li>• Mechanic repairs leaking pipe</li> <li>• Mechanic places Pump on Pump</li> <li>• Leaking pipe gets water</li> <li>• Player leaks out pipe and pump steps one</li> </ul>
2023.05.02. 17:00	0,5 óra	Skáre	Parancsokhoz tartozó metódusok leírása: <ul style="list-style-type: none"> <li>• connect</li> </ul>

			<ul style="list-style-type: none"> <li>• sticky</li> <li>• connectpipe</li> <li>• leakpipe</li> <li>• switchpump</li> </ul>
2023.05.02. 20.00	1 óra	Deé-Lukács	Parancsokhoz tartozó metódusok leírása: <ul style="list-style-type: none"> <li>• move</li> <li>• grease</li> <li>• switch</li> <li>• placepump</li> <li>• repair</li> </ul>
2023.05.02.21:00	1 óra	Kiss, Skáre, Andó, Deé-Lukács, Vörös	Értekezlet: osztályok, tesztek átbeszélése, játszás megvalósítása
2023.05.02. 22:00	1 óra	Andó	Parancsokhoz tartozó metódusok leírása: <ul style="list-style-type: none"> <li>• fill</li> <li>• break</li> <li>• slipperypipe</li> <li>• stickypipe</li> <li>• pickup</li> </ul>

# 10. Prototípus beadása

9 – Capybara

Konzulens:  
Goldschmidt Balázs

## Csapattagok

Andó Viola	H51B9J	viola.ando0618@gmail.com
Deé-Lukács András Gergely	JHFLXZ	gergoesjanka@gmail.com
<u>Kiss Blanka Zselyke</u>	C6037J	kissblanka02@gmail.com
Skáre Erik	Z7ZF6D	skareerik55@gmail.com
Vörös Vilmos	HW921K	v.vilmos10@gmail.com

2023. 05. 14.

## 10. Prototípus beadása

### 10.1 Fordítási és futtatási útmutató

#### 10.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Cistern.java	2 KB	2023.05.10. 10:48	A ciszterna osztály
Element.java	4 KB	2023.05.10. 10:48	Az aktív és passzív elemek űsosztálya
Game.java	38 KB	2023.05.10. 10:48	A játékért felelős osztály
ISteppable.java	1 KB	2023.05.10. 10:48	A léptethető dolgok interfésze
Main.java	18 KB	2023.05.10. 10:48	A programot irányító osztály
Mechanic.java	3 KB	2023.05.10. 10:48	A szerelők osztály
Node.java	3 KB	2023.05.10. 10:48	Az aktív elemek űsosztálya
Pipe.java	7 KB	2023.05.10. 10:48	A cső osztály
PipeEnd.java	3 KB	2023.05.10. 10:49	A csővég osztály
Player.java	6 KB	2023.05.10. 10:49	A játékosok űsosztálya
Pool.java	1 KB	2023.05.10. 10:49	A medence (pontgyűjtő) osztály
Pump.java	5 KB	2023.05.10. 10:49	A pumpa osztály
Saboteur.java	1 KB	2023.05.10. 10:49	A szabotőr osztály
Source.java	1 KB	2023.05.10. 10:49	A forrás osztály
Timer.java	4 KB	2023.05.10. 10:49	Az időzítő osztály

#### 10.1.2 Fordítás

##### 1. Lépés: A környezet előkészítése

Első lépésként fel kell telepíteni a Java 18-at a számítógépre.

A gépen lévő Java JDK a következő parancssal ellenőrizhető a Terminál alkalmazásban

```
java -version
```

Ha a következőt látjuk, akkor további teendőnk a környezettel nincsen.

```
C:\Users\cloud\Downloads\src>java --version
java 18.0.2.1 2022-08-18
Java(TM) SE Runtime Environment (build 18.0.2.1+1-1)
Java HotSpot(TM) 64-Bit Server VM (build 18.0.2.1+1-1, mixed mode, sharing)
```

Ha nem ezt látni, akkor a következő linkről lehet letölteni a megfelelő verziójú Java fejlesztői csomagot. (64-bites Windows platformra):

[https://download.oracle.com/java/18/archive/jdk-18.0.2.1\\_windows-x64\\_bin.exe](https://download.oracle.com/java/18/archive/jdk-18.0.2.1_windows-x64_bin.exe)

Ezután gondoskodni kell arról, hogy a frissen telepített Java hozzá legyen adva a PATH nevű környezeti változóhoz. Ennek az egyik módja a következő parancs lefuttatása parancssorban:

```
set PATH=%PATH%;C:\Program Files\Java\jdk-18.0.2.1\bin
```

Itt a feketével jelölt részt ki kell cserálni a feltelepített Java pontos verziójára.

## 2. Lépés: Forrásfájlok letöltése

A konzolens által megadott helyről le kell tölteni a Capybara.zip fájlt, ami tartalmazza a prototípus program forráskódját, a tesztelő szkriptet és a teszteléshez használt bemeneti fájlokat.

Ezután a Start menü Terminál/PowerShell alkalmazását kell elindítani, és kiadni a következő parancsot:

```
cd C:\Users\cloud\Downloads
```

Ezután tömörítsük ki a mappát:

```
tar -x -f capybara.zip
```

Ezután navigálunk bele a kitömörített mappába

```
cd capybara\src
```

## 3. Lépés: A program fordítása

Nyissuk meg egy PowerShell ablakot adminisztrátorként, majd navigálunk el a kitömörített fájlokat tartalmazó mappába. Ezután a következő parancsot futtatva, a program lefordul.

```
javac Main.java
```

### 10.1.3 Futtatás

Windows alatt a következő parancssal futtatható a program:

```
java Main
```

Amennyiben a tesztelő szkript segítségével akarjuk tesztelni a prototípust, a következő parancsot adjuk ki, hogy futtatni tudjuk a szkriptet.

```
Set-ExecutionPolicy Unrestricted
```

A parancs kiadása után vállaljuk a kockázatot:

```
Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y
```

Ezután futtathatjuk a szkriptet a tesztelni kívánt teszeset sorszámaival. Például a 10-es sorszámú teszesetet akarjuk végrehajtani.

```
.\Runttest.ps1 10
```

Amennyiben az összes teszesetet akarjuk egyszerre végrehajtani:

```
.\Runttest.ps1
```

Elképzelhető, hogy a szkript futtatása előtt is megerősítést kér a PowerShell, adjuk meg neki.

```
Security warning
Run only scripts that you trust. While scripts from the internet can be useful, this script can potentially harm your
computer. If you trust this script, use the Unblock-File cmdlet to allow the script to run without this warning
message. Do you want to run C:\Users\cloud\Documents\proto-master2\proto-master2\src\Runttest.ps1?
[D] Do not run [R] Run once [S] Suspend [?] Help (default is "D"): R
```

A tesztelés végeztével állítsuk vissza a végrehajtási tervet az alapértelmezett beállításra:

```
Set-ExecutionPolicy RemoteSigned
```

## 10.2 Tesztek jegyzőkönyvei

### 10.2.1 Player moves to sticky Pipe and tries to move again

Tesztelő neve	Kiss Blanka Zselyke
Teszt időpontja	2023.05.14. 18.36

Tesztelő neve	Kiss Blanka Zselyke
Teszt időpontja	2023.05.14. 18.30
Teszt eredménye	A kimenet a 6. sorban eltért

<b>Lehetséges hibaok</b>	A megadott kimenet hibás volt, isSlippery helyett isSticky-t tesztelünk, így valóban jó kimenetet adott a proto.
<b>Változtatások</b>	Átírtam a hibás kimenetet isSlippery-ról isSticky-re.

### 10.2.2 Player moves to slippery pipe from pump

<b>Tesztelő neve</b>	Andó Viola
<b>Teszt időpontja</b>	2023.05.14. 18:00

### 10.2.3 Player switches pump and steps it

<b>Tesztelő neve</b>	Skáre Erik
<b>Teszt időpontja</b>	2023.05.14. 20:15

<b>Tesztelő neve</b>	Skáre Erik
<b>Teszt időpontja</b>	2023.05.14. 20:10
<b>Teszt eredménye</b>	<pre>Found in Line 1:Players: Expected in line 1:ConnectedPipes: pip1 pip2 pip3 pip4 Found in Line 2:ConnectedPipes: pip1 pip2 pip3 pip4 Expected in line 2:InPipe: pip3 Found in Line 3:inPipe: pip3 Expected in line 3:OutPipe: pip4 Found in Line 4:outPipe: pip4 Expected in line 4:TankFull: true Found in Line 5:isBroken: false Expected in line 5: Found in Line 6:TankFull: false Expected in line 6:End1: pump Found in Line 7: Expected in line 7:End2: null Found in Line 8:End1: pump Expected in line 8:hasWater: false Found in Line 9:End2: null Expected in line 9: Found in Line 10:hasWater: false Expected in line 10:End1: pump Found in Line 11: Expected in line 11:End2: null Found in Line 12:End1: pump Expected in Line 12:hasWater: false Found in Line 13:End2: null Expected in line 13: Found in Line 14:hasWater: false Expected in line 14:</pre>
<b>Lehetséges hibaok</b>	A pumpa bemeneti csövébe került víz, azonban mivel a csőnek az egyik vége szabad volt, így rögtön ki is folyt belőle a víz. Valamint a state lekérdezésnél kimaradtak állapotSOROK.
<b>Változtatások</b>	Felvettetem egy új pumpát, amit hozzákötöttem a bemeneti csőhöz, hogy ne folyjon ki rögtön a víz. Valamint a kimeneti fájlba hozzáadtam a kimaradt állapotSOROKAT.

### 10.2.4 Pump with empty tank steps 2

<b>Tesztelő neve</b>	Vörös Vilmos
<b>Teszt időpontja</b>	2023.05.14 10:30

### 10.2.5 Pipe network clogs

<b>Tesztelő neve</b>	Deé-Lukács András Gergely
----------------------	---------------------------

<b>Teszt időpontja</b>	2023.05.14 21:00
------------------------	------------------

### 10.2.6      Source steps

<b>Tesztelő neve</b>	Kiss Blanka Zselyke
<b>Teszt időpontja</b>	2023.05.14. 18.45

<b>Tesztelő neve</b>	Kiss Blanka Zselyke
<b>Teszt időpontja</b>	2023.05.14. 18.38
<b>Teszt eredménye</b>	Kifolyik a víz a csőből.
<b>Lehetséges hibaok</b>	A csövek egyik vége szabad volt, így azokból kifolyt a víz.
<b>Változtatások</b>	Hozzáadtam a hálózathoz egy-egy pumpát, amiket összekötöttem a csövekkel, hogy ne legyen szabad végük.

### 10.2.7      Cistern steps

<b>Tesztelő neve</b>	Deé-Lukács András Gergely
<b>Teszt időpontja</b>	2023.05.14 21:10

### 10.2.8      Pump steps

<b>Tesztelő neve</b>	Skáre Erik
<b>Teszt időpontja</b>	2023.05.14. 20:25

<b>Tesztelő neve</b>	Skáre Erik
<b>Teszt időpontja</b>	2023.05.14. 20:20

Teszt eredménye	<pre>Found in line 1:Players:  Expected in line 1:ConnectedPipes: pip1 pip2  Found in line 2:ConnectedPipes: pip1 pip2  Expected in line 2:InPipe: pip1  Found in line 3:inPipe: pip1  Expected in line 3:OutPipe: pip2  Found in line 4:outPipe: pip2  Expected in line 4:TankFull: true  Found in line 5:isBroken: false  Expected in line 5:  Found in line 6:TankFull: false  Expected in line 6:End1: pump  Found in line 7:  Expected in line 7:End2: null  Found in line 8:End1: pump  Expected in line 8:hasWater: false  Found in line 9:End2: null  Expected in line 9:  Found in line 10:hasWater: false  Expected in line 10:End1: pump  Found in line 11:  Expected in line 11:End2: null  Found in line 12:End1: pump  Expected in line 12:hasWater: false  Found in line 13:End2: null  Expected in line 13:  Found in line 14:hasWater: false  Expected in line 14:</pre>
Lehetséges hibaok	A teszt logikája jó, azonban a kimeneti fájlban a state lekérdezésnél kimaradt 2 állapotot.
Változtatások	Hozzáadtam a kimeneti fájhoz a megfelelő állapotsorokat.

### 10.2.9 Broken, empty Pump steps

Tesztelő neve	Vörös Vilmos
Teszt időpontja	2023.05.14 10:45

Tesztelő neve	Vörös Vilmos
Teszt időpontja	2023.04.15. 10:43
Teszt eredménye	<pre>Test number 9 failed!  Found in line 2:End2: helppump1  Expected in line 2:End2: null  Found in line 12:End2: helppump2  Expected in line 12:End2: null</pre>
Lehetséges hibaok	Hibás elvárt kimeneti fájl. A frissített bemenettel együtt az elvárt kimenetet nem lett frissítve a segédobjektumok felvételekor.
Változtatások	Kimeneti fájl javítása.

### 10.2.10 Player connects pipe to Pump

Tesztelő neve	Andó Viola
---------------	------------

Teszt időpontja	2023.05.14. 18:05
-----------------	-------------------

### 10.2.11 Player connects Pipe to Cistern

Teszteleő neve	Kiss Blanka Zselyke
Teszt időpontja	2023.05.14. 18.55

Teszteleő neve	Kiss Blanka Zselyke
Teszt időpontja	2023.05.14. 18.50
Teszt eredménye	Nem jól kötötte a ciszternához a csövet.
Lehetséges hibaok	Rossz parancsot használtam, a pikcup –pipe helyett holdpipe kell, mivel az előbbi a ciszternával gyártatja le a csövet, de most nem azt teszteljük, hanem amikor már egy kezében levő csövet csatlakoztat.
Változtatások	Hozzáadtam a hálózathoz egy csövet, amit felvettettem a holdpipe parancsal a pickup –pipe helyett.

### 10.2.12 Player connects Pipe to Source

Teszteleő neve	Deé-Lukács András Gergely
Teszt időpontja	2023.05.14 21:20

Teszteleő neve	Deé-Lukács András Gergely
Teszt időpontja	2023.04.15. 21:15
Teszt eredménye	<pre>Test number 12 failed! Found in line 6:hasWater: false Expected in line 6:hasWater: true</pre>
Lehetséges hibaok	Hibás teszeset, a forráshoz kötött cső vége szabad, a teszt eset elvárja, hogy benne maradjon a víz, ami helyes működéskor kifolyik.
Változtatások	A teszesethez hozzá lett adva egy pumpa a cső végére, ami eldugta, hogy ne tudjon kifolyni belőle a víz.

### 10.2.13 Player connects Pipe to Pipe

Teszteleő neve	Kiss Blanka Zselyke
Teszt időpontja	2023.05.14. 18.59

Teszteleő neve	Kiss Blanka Zselyke
Teszt időpontja	2023.05.14. 18.57
Teszt eredménye	Nem létező objektum state-jét nem lehet kiírni, hiba.
Lehetséges hibaok	A pip nem létezik az adott állapotban, az on állapotát kell kiírni.
Változtatások	Átírtam a pip-et on-ra.

### 10.2.14 Player disconnects empty Pipe

Teszteleő neve	Vörös Vilmos
Teszt időpontja	2023.05.14 10:35

### 10.2.15 Player disconnects full pipe

Teszteleő neve	Andó Viola
Teszt időpontja	2023.05.14. 18:10

**10.2.16 Player breaks Pipe**

Tesztelő neve	Kiss Blanka Zselyke
Teszt időpontja	2023.05.14. 19.21

Tesztelő neve	Kiss Blanka Zselyke
Teszt időpontja	2023.05.14. 19.15
Teszt eredménye	Nem lyukas a cső és nincs pontja a szabotöröknek.
Lehetséges hibaok	Nem elég switch-eltüntetni, connectálni is kell egymáshoz a pumpot és a p1-et, valamint a Mechanic a p1-re kell mozogjon, nem a pumpra.
Változtatások	Beírtam egy connectet a switch előtt és átírtam move m p1-re a pump-ot.

**10.2.17 Mechanic repairs broken pump**

Tesztelő neve	Deé-Lukács András Gergely
Teszt időpontja	2023.05.14 21:25

**10.2.18 Mechanic repairs leaking pipe**

Tesztelő neve	Skáre Erik
Teszt időpontja	2023.05.14. 20:30

**10.2.19 Mechanic repairs working Pump**

Tesztelő neve	Vörös Vilmos
Teszt időpontja	2023.05.14 10:38

Tesztelő neve	Vörös Vilmos
Teszt időpontja	2023.05.14. 10:37
Teszt eredménye	Test number 19 failed! Found in line 1:Players: m Expected in line 1:Player: m
Lehetséges hibaok	Hibás az elvárt kimenetet tartalmazó fájl.
Változtatások	Elvárt kimenet pontosítása.

**10.2.20 Mechanic repairs undamaged pipe**

Tesztelő neve	Andó Viola
Teszt időpontja	2023.05.14. 18:15

**10.2.21 Mechanic picks up Pump at Cistern**

Tesztelő neve	Kiss Blanka Zselyke
Teszt időpontja	2023.05.14. 19.25

**10.2.22 Mechanic picks up pump at Source**

Tesztelő neve	Deé-Lukács András Gergely
Teszt időpontja	2023.05.14 21:30

**10.2.23 Mechanic picks up Pipe at Cistern**

Tesztelő neve	Kiss Blanka Zselyke
Teszt időpontja	2023.05.14. 19.27

**10.2.24 Mechanic picks up Pipe at Source**

Tesztelő neve	Vörös Vilmos
Teszt időpontja	2023.05.14 10:35

**10.2.25 Mechanic places Pump on Pipe**

Tesztelő neve	Andó Viola
Teszt időpontja	2023.05.14. 18:25

Tesztelő neve	Andó Viola
Teszt időpontja	2023.05.14. 18:20
Teszt eredménye	<pre>Test number 25 failed! Found in line 6:End1: pump1 Expected in line 6:End1: pump Found in line 9:Players: m Expected in line 9:Player: m</pre>
Lehetséges hibaok	Hibás az elvárt kimenetet tartalmazó fájl.
Változtatások	Elvárt kimenet pontosítása.

**10.2.26 Mechanic places Pump on Cistern**

Tesztelő neve	Deé-Lukács András Gergely
Teszt időpontja	2023.05.14 21:35

**10.2.27 Mechanic places Pump on Source**

Tesztelő neve	Deé-Lukács András Gergely
Teszt időpontja	2023.05.14 21:40

**10.2.28 Mechanic places Pump on Pump**

Tesztelő neve	Skáre Erik
Teszt időpontja	2023.05.14. 20:40

Tesztelő neve	Skáre Erik
Teszt időpontja	2023.05.14. 20:35
Teszt eredménye	<pre>Found in line 3:holdingPumps: Expected in line 3:holdingPumps: new</pre>
Lehetséges hibaok	A teszt logikája jó, azonban a Google Docs nem megfelelően kezelte a dupla kötőjelet, ezért innen kimásolva más karakter jelent meg, így az egyik parancs nem hajtódott végre.

**10.2.29 Leaking pipe gets water**

Tesztelő neve	Skáre Erik
Teszt időpontja	2023.05.14. 20:45

**10.2.30 Undamaged pipe gets water**

Tesztelő neve	Andó Viola
Teszt időpontja	2023.05.14. 18:35

**10.2.31 Player disconnects Pump's output**

Tesztelő neve	Deé-Lukács András Gergely
Teszt időpontja	2023.05.14 21:45

**10.2.32 Player leaks out pipe and pump steps one**

Tesztelő neve	Skáre Erik
Teszt időpontja	2023.05.14. 20:55

Tesztelő neve	Skáre Erik
Teszt időpontja	2023.05.04. 20:50
Teszt eredménye	Found in line 1:outPipe: null Expected in line 1:TankFull: false
Lehetséges hibaok	A pumpa tankjának állapotát “-a”-val és nem “-t”-vel kell lekérdezni. Ez volt a hiba a bemeneti fájlban.
Változtatások	Bemeneti fájl javítása

**10.2.33 Mechanic repairs broken pump and it steps 2**

Tesztelő neve	Andó Viola
Teszt időpontja	2023.05.14. 18:40

**10.2.34 Pump breaks with full tank and it steps 2**

Tesztelő neve	Vörös Vilmos
Teszt időpontja	2023.05.14 10:40

Tesztelő neve	Vörös Vilmos
Teszt időpontja	2023.05.04 10:38
Teszt eredménye	Test number 34 failed! Found in line 7:TankFull: false Expected in line 7:TankFull: true
Lehetséges hibaok	Rossz, elírt elvárt kimeneti fájl
Változtatások	Kimeneti fájl javítása

### 10.3 Értékelés

Tag neve	Tag neptun	Munka százalékban
Andó Viola	H51B9J	20%
Kiss Blanka Zselyke	C6037J	20%
Deé-Lukács András Gergely	JHFLXZ	20%
Skáre Erik	Z7ZF6D	20%
Vörös Vilmos	HW921K	20%

## 10.4 Napló

Kezdet	Időtartam	Résznevők	Leírás
2023.05.09. 19:00	1 óra	Andó Deé-Lukács Kiss Skáre Vörös	<p>Értekezlet.</p> <p>Döntések: Feladatok felosztása</p> <p>Andó: Element, Source, Node osztályok implementálása A break, fill, pickup, slipperypipe, stickypipe parancsoknak megfelelő függvények implementálása a Game-be</p> <p>Deé-Lukács: PipeEnd, Pool, Timer osztályok implementálása, move, grease, switch, placepump, repair parancsok implementálása</p> <p>Kiss: Game osztály alap, Player, Mechanic osztályok megírása Prototípus parancsaiból: add, drain, holdpipe, holdpump, step parancsok implementálása</p> <p>Skáre: Pipe és Saboteur osztályok megírása Connect, Sticky, ConnectPipe, LeakPipe, Switch, Load implementálása a Game osztályba</p> <p>Vörös: Cistern, Pump osztályok bővítése, proto parancsok: leak, state, disconnetpipe, tesztelő szkript megírása</p>
2023.05.10. 10:50	2,5 óra	Kiss	Game osztály alap, Player, Mechanic osztályok megírása Prototípus parancsaiból: add, drain, holdpipe, holdpump, step implementálása Game-ben
2023.05.10. 17:00	2 óra	Vörös	Cistern, Pump osztályok bővítése. Proto parancsok implementálása: state, leak, disconnectpipe, komment

2023.05.11. 13:00	1 óra	Andó	Tevékenység: Element, Node, Source osztályok elkészítése
2023.05.11. 18:00	2 óra	Vörös	State parancs javítása, Game osztály parancsértelmezésének előkészítésé
2023.05.12. 14:00	2,5 óra	Deé-Lukács	Tevékenység: PipeEnd, Pump, Timer osztályok megírása
2023.05.13. 10:00	2,5 óra	Deé-Lukács	Tevékenység: Main osztályban a következő parancsok implementálása: move, grease, switch, placepump, repair A Game osztály következő metódusai: Move(), Grease(), SwitchPump(), PlacePump(), Repair() Timer osztály használata a Main és Game osztályokban
2023.05.13. 11:00	1,5 óra	Andó	Tevékenység: A Game osztály Break(), Fill(), PickUp(), StickPipe(), SlipperyPipe() függvényeinek, és az ezekhez tartozó kód részletek megírása a Main-be
2023.05.13. 15:00	3,5 óra	Skáre	Tevékenység: A Pipe és Saboteur osztályok megírása/kiegészítése a változtatott követelményeknek megfelelően. Connect, Sticky, ConnectPipe, LeakPipe, Switch, Load parancsoknak megfelelő metódusok implementálása a Game osztályba. A fenti metódusok hívásának felvezetése a Mainbe.
2023.05.13. 20:00	4 óra	Andó Deé-Lukács Kiss Skáre Vörös	Értekezlet: kód részletek egyeztetése, debuggolása  Döntés: tesztesetek beosztása  Andó: 2,10,15,20,25,30, 33  Deé-Lukács: 5, 7, 12, 17, 22, 26, 27, 31  Kiss: 1, 6, 11, 13, 16, 21, 23  Skáre: 3, 8, 18, 28, 29, 32

			Vörös: 4, 9, 14, 19, 24, 34
2023.04.14. 10:00	2 óra	Vörös	Tesztelő szkript javítása, tesztesetek futtatása, jegyzőkönyvezése: 4, 9, 14, 19, 24, 34
2023.05.14. 15:00	1 óra	Andó	Tevékenység: A tesztesetek futtatása és az ezekhez tartozó jegyzőkönyvek megírása (2,10,15,20,25,30, 33 tesztesetek)
2023.05.14 20:00	1 óra	Skáre	Tevékenység: A tesztesetek futtatása és az ezekhez tartozó jegyzőkönyvek megírása. (3, 8, 18, 28, 29, 32)
2023.05.14. 21:00	1 óra	Deé-Lukács	Tevékenység: Tesztesetek futtatása és az ezekhez tartozó jegyzőkönyvek megírása ( 5, 7, 12, 17, 22, 26, 27, 31)
2023.05.14. 21:00	2 óra	Kiss	Tesztesetek futtatása és az ezekhez tartozó jegyzőkönyvek megírása (1, 6, 11, 13, 16, 21, 23) Fájllista dokumentáció, dokumentum ellenőrzése

# 11. Grafikus felület specifikációja

## 11.1 A grafikus interfész

A pálya kirajzolása során az alábbi ábrákat fogjuk kirajzolni és használni az egyes objektumok megjelenítésére:

Forrás:



Ciszterna:



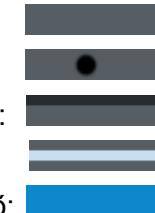
Pumpa:

- Sima pumpa:
- Törött pumpa:



Cső:

- Sima cső:
- Lyukas cső:
- Ragadós cső:
- Csúszós cső:
- Vízzel teli cső:



A lyukas, ragadós, csúszós, teli csövek bármely kombinációja előfordulhat, tehát például lehet olyan cső, amely csúszós és lyukas, vagy csúszós és vízzel teli is.

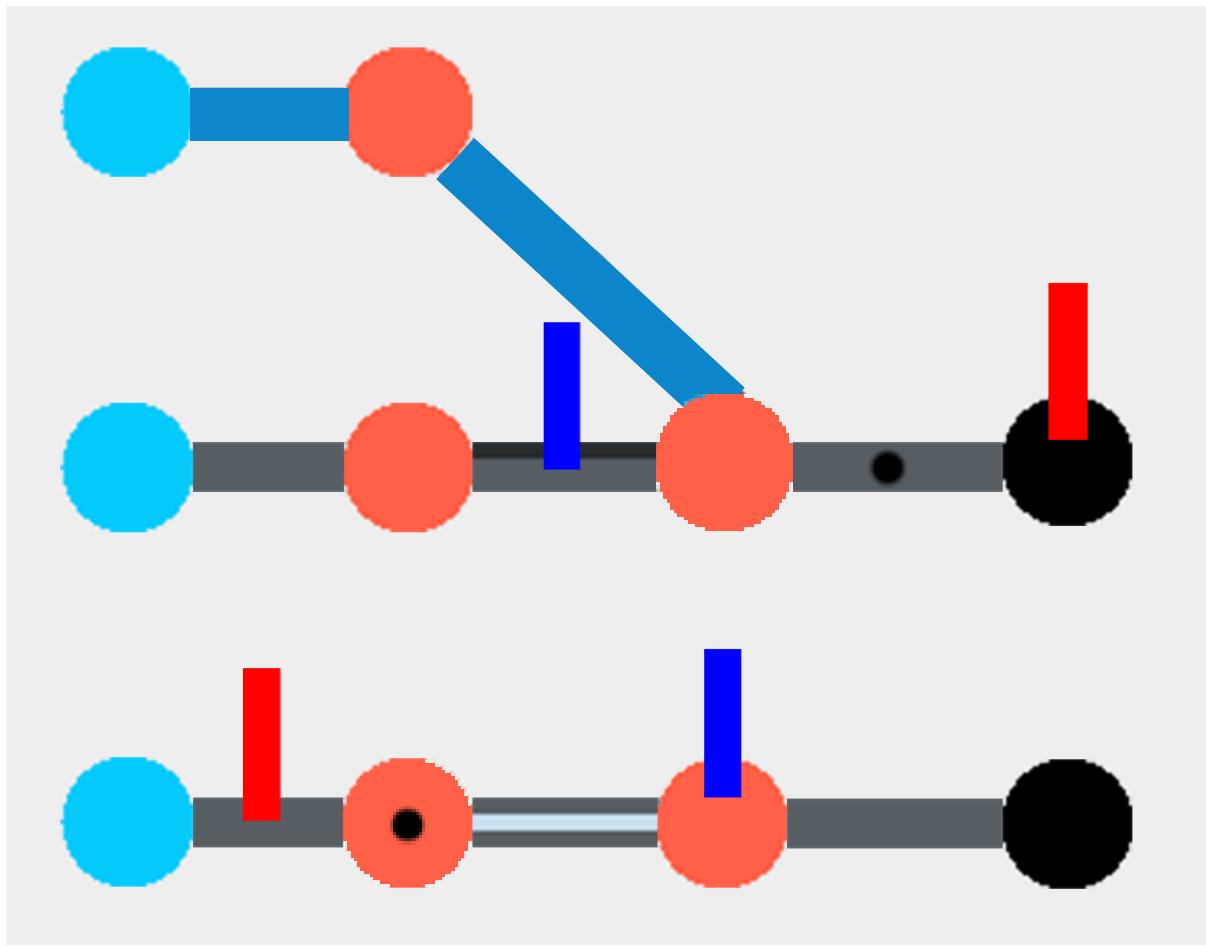
Szerelő:



Szabotőr:



Ezek alapján egy pálya például a következőképpen nézhet ki:



A játék alapvetően egy ablakból fog állni, a program indításakor megjelenik egy alappálya (például a fent látható). A pálya megjelenítése mellett az ablak tartalmazni fog egy JMenu-t az alábbi felépítés szerint:

- File
  - Save: elmenti az aktuális játékállást
  - Load: betöltye elmentett játékállást
- Edit
  - Add Mechanic: hozzáad a játékhoz egy új szerelőt
  - Add Saboteur: hozzáad a játékhoz egy új szabotőrt
- Game
  - Start: elindítja a játékot
  - Pause: megállítja a játékot

Ezek alapján a menü az alábbi módon fog kinézni:



Továbbá a pálya mellett látható lesz egy panel, amelyen egy JList-ből lehet kiválasztani az összes jelen lévő játékos közül az aktuálisan lépni kívánó játékost. A játékosok neve a szerepüköből (Mechanic / Saboteur) és az azonosítójukból (növekvő pozitív egész számok) tevődik össze, ez jelenik meg a listában.

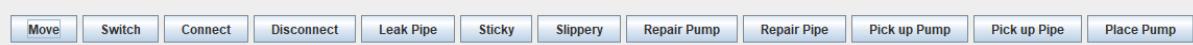
Mechanic1
Mechanic2
<b>Mechanic3</b>
Saboteur1
Saboteur2
Saboteur3

Ezenkívül egy másik panelen, a pálya mellett még megjelennek az összes lehetséges, a játékos által végrehajtható műveletek gombai (JButton), amelyek az alábbiak lehetnek:

- *Move*: A gomb megnyomásának hatására, majd a pályán, az aktív játékost tartalmazó elem egyik szomszédjának kattintással történő kiválasztása után, végrehajtódik az akció
- *Switch*: A gomb megnyomása után, a pályán, ahol az aktív játékos éppen tartózkodik (csak akkor fog történni bármi is, ha pumpán áll), annak a bemeneti és kimeneti csövének kattintással történő kiválasztása után megtörténik az akció.
- *Connect*: A gomb megnyomása után az aktív játékos kezében lévő csővéget, felcsatlakoztatja arra az aktív elemre ahol éppen áll. (Csak akkor fog történni bármi is, ha pumpán, forráson vagy ciszternán áll.)
- *Disconnect*: A gomb megnyomása után az aktív játékos, kiválasztja az egyik szomszédját kattintással, annak az aktív elemnek, ahol éppen áll, ezzel lecsatlakoztatva egy csövet. (Csak akkor fog történni bármi is, ha pumpán, forráson vagy ciszternán áll.)
- *Leak Pipe*: A gomb megnyomásának hatására kilyukad az az elem, amin az aktív játékos éppen áll, ezzel megtörténik az akció. (Csak akkor fog bármi is történni ha a játékos csövön áll).
- *Sticky*: A gomb megnyomásának hatására ragadós lesz az az elem, amin az aktív játékos éppen áll. (Csak akkor fog bármi is történni ha a játékos csövön áll).
- *Slippery*: A gomb megnyomásának hatására csúszós lesz az az elem, amin az aktív játékos éppen áll. (Csak akkor fog bármi is történni ha a játékos csövön áll és szabotőr).
- *Repair Pump*: A gomb megnyomásának hatására megjavul, működőképessé válik az az elem, amin az aktív játékos éppen áll. (Csak akkor fog bármi is történni, ha a játékos pumpán áll és szerelő.)
- *Repair Pipe*: A gomb megnyomásának hatására megjavul, működőképessé válik az az elem, amin az aktív játékos éppen áll. (Csak akkor fog bármi is történni, ha a játékos csövön áll és szerelő.)
- *Pick up Pump*: A gomb megnyomásának hatására az aktív játékos felvesz egy pumpát, annál az aktív elemnél, ahol éppen áll, ezzel végrehajtva az akciót. (Csak akkor fog bármi is történni, ha a játékos ciszternán áll és szerelő.)
- *Pick up Pipe*: A gomb megnyomásának hatására az aktív játékos felvesz egy csövet, annál az aktív elemnél, ahol éppen áll, ezzel végrehajtva az akciót. (Csak akkor fog bármi is történni, ha a játékos ciszternán áll és szerelő.)

- *Place Pump:* A gomb megnyomásának hatására az aktív játékos lerak egy pumpát arra az elemre, amin éppen áll, ezzel végrehajtva az akciót. (Csak akkor fog bármi is történni, ha a játékos csövön áll és szerelő.)

Ezek alapján a gombok az alábbi elrendezésben lesznek láthatóak:



## 11.2 A grafikus rendszer architektúrája

### 11.2.1 A felület működési elve

A grafikus megjelenítés megvalósításának koncepciója tervezünk szerint pull alapú; a modell objektumainak megjelenítését végző osztályok nem értesülnek az objektumok állapotának változásáról, hanem a megjelenítést végző osztályok olyan gyakran kérdezik le az általuk megfigyelt objektumok aktuális állapotát (100 ms-onként), hogy felhasználóként nem érzékelünk késleltetést a modell és a nézet változásai között.

A modell és a megjelenítő osztályokat a Main osztályon keresztül kapcsoljuk össze. A Main osztály tölti be részben a Controller szerepét: tárolja a játékban szereplő játékosokat, valamint a többi modell elemet és az adott elemhez tartozó megjelenítő osztályt. A modell elemeinek tárolásához felhasználtuk a prototípus programhoz megírt Game osztályt, ez úgymond egy másodlagos Controller szerepet tölt be. Ezen keresztül tárolja a modell elemeket a Main, viszont a megjelenített játékos-lista miatt a játékosokat ő is nyilvántartja. A modell változtatását, befolyásolását egyes műveletek esetén az aktív játékoson keresztül végzi a Main, más esetekben pedig közvetve, a Game osztály megfelelő metódusával változtat a modellen.

A Main a megjelenített objektumokat is közvetve ismeri csak. Egy saját, megjelenítést segítő "vászon" segédosztályban tároljuk a megjelenítést végző objektumokat. Ezáltal egyszerűbb a kirajzolás és a gyakori frissítés/újrarendezés. Ez az megemlített segédosztály dolgozza fel, melyik megjelenített objektumra kattintottunk, és már az azonosított objektummal együtt értesíti a Main-t/kontrollert a kattintásról.

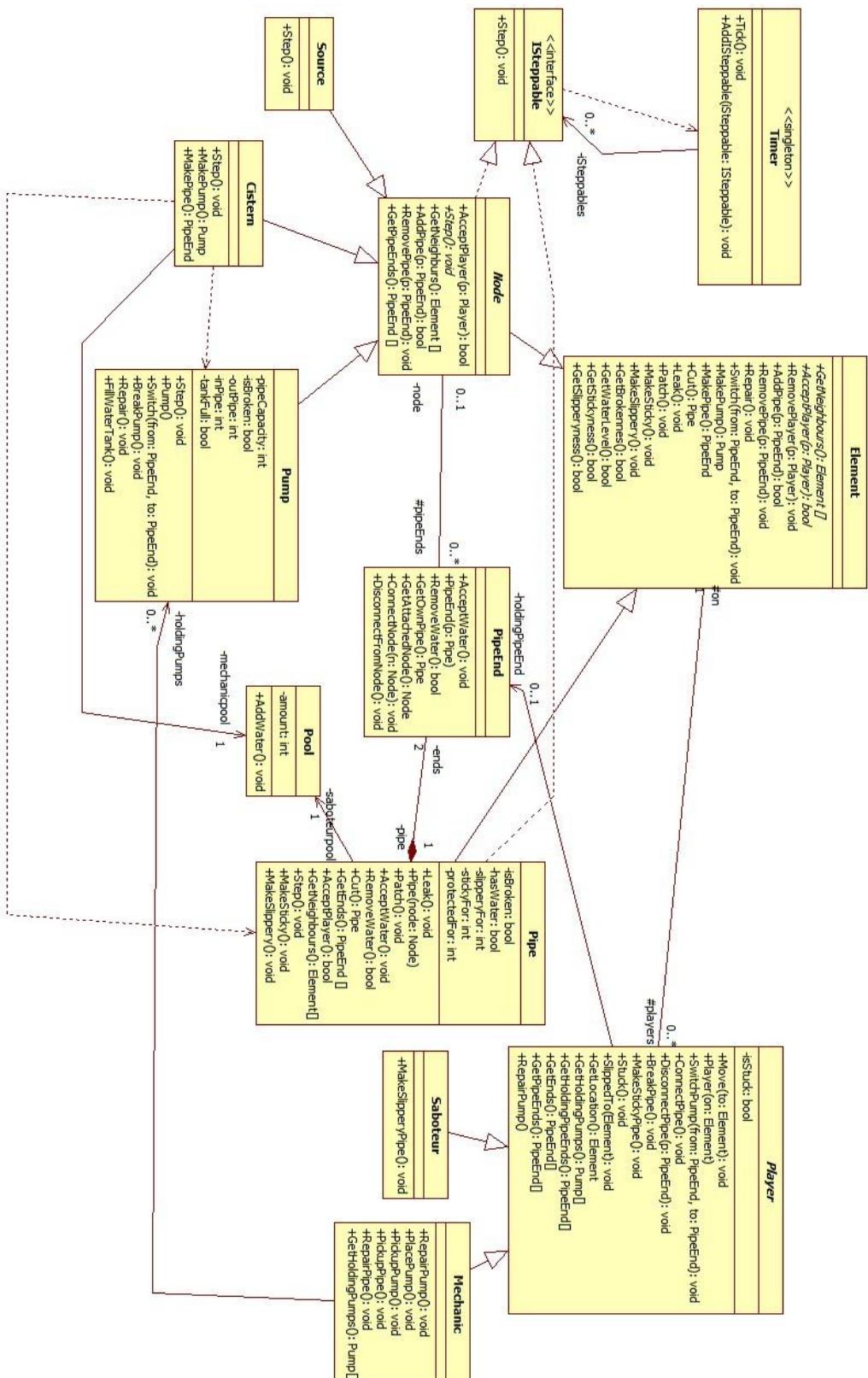
A Main osztály kezeli le a felhasználói interakciókat (képernyőn megjelenő gomb klikselése, pályán látható objektum klikselése). Az interakciók hatására nem minden változtat a modellen, csak akkor, ha elég információ érkezett a felhasználótól (pl. mozgatás gombra klikselés után még nem klikkelt egyetlen mezőre sem). Maga a Main nem jelzi senkinek, ha változás történt, hanem egy RefreshTimer segítségével valósítja meg a megjelenítő osztályok gyakori frissítését.

Ha a Main olyan műveletet hajt végre a modellen, melynek során új, megjelenítendő objektum jön létre (pl. új pumpa elhelyezése), akkor annak a Main példányosít egy megjelenítő objektumot. Ha egy műveletben a modell elem már létezik, de a játék logikája miatt az adott művelet elvégzéséig nem kellett megjeleníteni (pl. cső csatlakoztatása), akkor is új megjelenítő objektum jön létre az adott elemhez. Ezzel összhangban, ha egy modell

elemet magát nem kell kitörölni, de a pályáról el kell tüntetni, akkor a Main megsemmisíti az elemet megjelenítő objektumot.

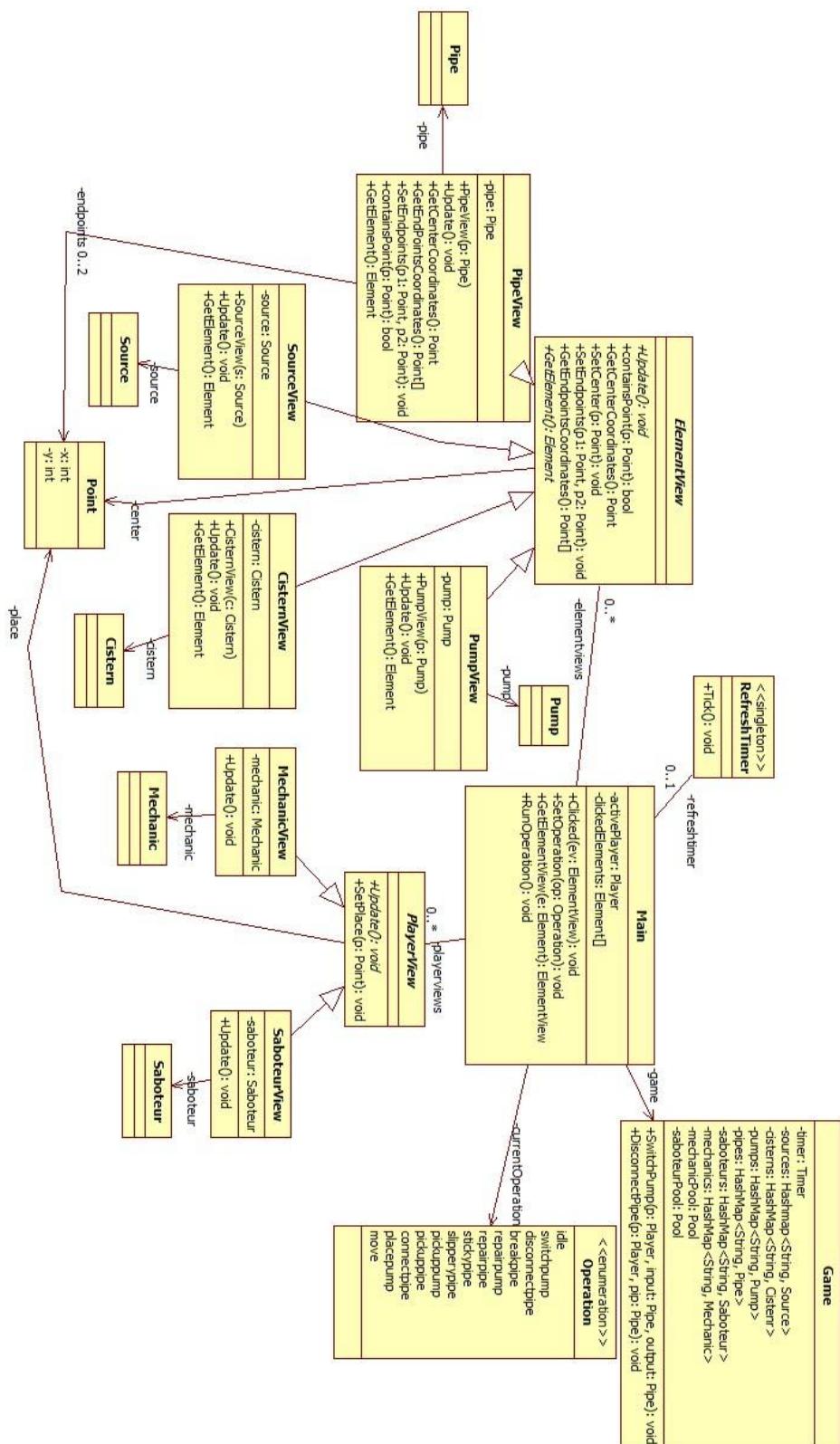
A Controller mellett minden megjelenítésért felelős objektum is ismeri azt a modell elemet, aminek a megjelenítéséért felel. Ha egy ilyen (View) objektumot frissítenek (Update metódus hívása), akkor az lekérdezi a hozzá tartozó modell elem/objektum összes olyan tulajdonságát/állapotát, ami fontos a megjelenítés szempontjából.

## **11.2.2 A felület osztály-struktúrája**



A modell működése nem változott. A modell interfésze annyiban változott, hogy egy pá� metódus felkerült a **Player** és **Element** őssosztályokba. Ezek a következők:

**GetBrokenness()**, **GetWaterLevel()**, **GetStickyness()**, **GetSlipperyness()**, **GetLocation()**,  
**GetHoldingPumps()**, **GetHoldingPipeEnds()**, **GetEnds()**, **GetPipeEnds()**, **RepairPump()**



## 11.3 A grafikus objektumok felsorolása

### 11.3.1 CisternView

- **Felelősség**

A modellben szereplő **Cistern** objektumoknak a megjelenítésére szolgáló osztály. Ez tárol minden információt, végez minden műveletet, ami egy ciszterna megjelenítéséhez szükséges a csőhálózatban.

- **Ősosztályok**

ElementView

- **Attribútumok**

- **Cistern -cistern:** A játék modelljében lévő **Cistern** objektum referenciája, amit ez az objektum megjelenít.

- **Metódusok**

- **+CisternView(Cistern c):** Az osztály konstruktora. Beállítja a **-cistern** attribútum értékét egy paraméterként megadott ciszterna referenciájára a modellből.
- **void +Update():** A ciszterna grafikai reprezentációjának a frissítését megvalósító függvény. Újrarázsolja a ciszternát, a modellben lévő legfrissebb állapota szerint.
- **Element +GetElement():** Getter, amely visszatér az adott CisternView objektum megjelenített Cistern objektummal.

### 11.3.2 ElementView

- **Felelősség**

Ez egy absztrakt osztály, ami a modellben szereplő Element objektumoknak a megjelenítésére szolgál. Ez tárol minden információt, végez minden műveletet, ami egy a csőhálózat egy elemének a megjelenítéséhez szükséges.

- **Attribútumok**

- **Point -center:** A grafikai elemnek a középpontját tároló attribútum.

- **Metódusok**

- **void +Update():** A grafikai elem frissítését megvalósító függvény. Meghívásakor az elem újrarázsolódik a legfrissebb állapota szerint.
- **bool +containsPoint(Point p):** Ez a metódus megmondja, ha a paraméterként megadott pont a grafikai elemhez tartozik-e. Igazat térít vissza, ha a pont benne van.
- **Point +GetCenterCoordinates():** Ez a metódus visszatéríti a grafikai elem közepét, azaz a **-center** attribútum értékét.
- **void +SetCenter(Point p):** Ez a metódus beállítja egy elem középpontjának a koordinátáit, azaz a **-center** attribútum értékét.

- **Element +GetElement()**: Absztrakt metódus, ami visszatéríti az ElementView objektum által megjelenített objektumot. Az ősosztály még nem jelenít meg semmit, ezért ezt a gettert csak a leszármazottak implementálják.

### 11.3.3 Game

#### Felelősség

A játék vezérléséért felelős osztály. Fő felelőssége a modellbeli objektumot tárolása és a rajtuk elvégezni kívánt műveletek végrehajtása. A grafikus játék megvalósításában a “controller” szerep egyik megvalósítója (a Main osztály mellett).

#### Attribútumok

- **Timer -timer**: a játék időzítője, amelynek minden Tick()-je lépteti a léptethető objektumokat
- **HashMap<String, Pipe> -pipes**: a játék működése/használata során létrehozott cső objektumokat tárolja, pontosabban String-Pipe kulcs-érték párokat, ahol a String az adott objektum neve, amit létrehozáskor adtunk neki.
- **HashMap <String, Pump> -pumps**: a játék működése/használata során létrehozott pumpa objektumokat tárolja, pontosabban String-Pump kulcs-érték párokat, ahol a String az adott objektum neve, amit létrehozáskor adtunk neki.
- **HashMap <String, Cistern> -cisterns**: a játék működése/használata során létrehozott ciszterna objektumokat tárolja, pontosabban String-Cistern kulcs-érték párokat, ahol a String az adott objektum neve, amit létrehozáskor adtunk neki.
- **HashMap <String, Source> -sources**: a játék működése/használata során létrehozott forrás objektumokat tárolja, pontosabban String-Source kulcs-érték párokat, ahol a String az adott objektum neve, amit létrehozáskor adtunk neki.
- **HashMap <String, Mechanic> -mechanics**: a játék működése/használata során létrehozott szerelő objektumokat tárolja, pontosabban String-Mechanic kulcs-érték párokat, ahol a String az adott objektum neve, amit létrehozáskor adtunk neki.
- **HashMap <String, Saboteur> -saboteurs**: a játék működése/használata során létrehozott szabotőr objektumokat tárolja, pontosabban String-Saboteur kulcs-érték párokat, ahol a String az adott objektum neve, amit létrehozáskor adtunk neki.
- **Pool -saboteurPool**: a szerelők által gyűjtött pontokat tárolja
- **Pool -mechanicPool**: a szabotőrök által gyűjtött pontokat tárolja
- **Timer -timer**: a játék időzítője, amelynek minden Tick()-je lépteti a léptethető objektumokat

#### Metódusok

- **void SwitchPump(Player p, Pipe from, Pipe to)**: a paraméterként megadott (aktív) játékossal átállítatjuk azt a pumpát, amin áll. Először meg kell keresni a csövekhez tartozó csővégeket, majd ezek birtokában tudjuk meghívni a játékos megfelelő függvényét.
- **void DisconnectPipe(Player p, Pipe pip)**: a paraméterként megadott (aktív) játékossal lecsatlakoztatjuk a kiválasztott csövet. Először meg kell keresni a paraméterként kapott csőhöz tartozó csővéget (ami a játékos mezőjéhez csatlakozik), majd ennek birtokában tudjuk meghívni a játékos megfelelő függvényét.

### 11.3.4 Main

#### Felelősség

Az alkalmazás fő osztálya. Azonkívül, hogy Main osztály, ez látja el a részben controller szerepét is. Kezeli a kattintásokat, közvetetten tárolja a grafikai elemeket és elvégzi rajtuk a műveleteket.

- **Attribútumok**
  - **Player -activePlayer**: Azt a játékost tárolja, aki éppen aktív és műveletet végez.
  - **Player[] -players**: A játékosokat tárolja, akik a játékban részt vesznek
  - **Element[] -clickedElements**: Azokat az elemeket tárolja, amikre rákattintottunk.
  - **Game -game**: A prototípus programban használt, a játékot vezérlő osztály egy példánya. Ebben az attribútumban tároljuk a modell elemeket, valamint ezen keresztül változtatjuk a modellt, ha a modell felépítése miatt (pl. csővégek) nem tudunk közvetlenül az aktív játékoson keresztül műveletet végezni/a modellen változtatni.
  - **Operation -currentOperation**: A jelenlegi művelethez tartozó **Operation** értékét tartalmazza.
  - **RefreshTimer -refreshTimer**: A megjelenítés frissítését végző időzítő
- **Metódusok**
  - **void +Clicked(ev:ElementView)**: Ezt a metódust hívja meg a vászon, ha azt érzékeli, hogy valamelyik kirajzolt elemre kattintottunk rajta. Az ev paraméter tartalmazza a legfelső olyan objektumot, ami a kattintás pontját lefedi. A modell elemet - miután lekérdezte a megjelenítő objektumtól - beteszi a **-clickedElements** kollekcióba.
  - **void +SetOperation**: Beállítja a **-currentOperation** attribútum értékét.
  - **Element +GetElementView(Element e)**: Visszatéríti a paraméterként megadott modell elemhez tartozó grafikai elemet.
  - **void +RunOperation()**: Gombnyomás vagy kattintás hatására a Main végrehajtja az éppen aktuális beállított műveletet attól függően, elég információ/kattintott elem áll-e rendelkezésre. A következő műveletekhez szükséges plusz információ a felhasználótól: SwitchPump - elvégzéséhez két kiválasztott modell elem szükséges. Move/DisconnectPipe - elvégzéséhez egy kiválasztott modell elem szükséges.

### 11.3.5 MechanicView

- **Felelősség**

A modellben szereplő **Mechanic** (szerelők) objektum grafikus megjelenítését megvalósító osztály. Ez tárol minden információt, végez minden műveletet, ami egy szerelő megjelenítéséhez szükséges a csőhálózatban.

- **Ősosztályok**

PlayerView

- **Attribútumok**
  - **Mechanic -mechanic**: A játék modelljében levő **Mechanic** referencia, amit ez az objektum megjelenít.

- **Metódusok**
- **void +Update()**: A szerelő grafikai reprezentációjának a frissítését megvalósító függvény. Újrarájzolja a szerelőt a pályán a modellben levő legfrissebb állapota szerint.

### 11.3.6 Operation

- **Felelősség**

Ez egy felsorolás. A játékban végezhető műveleteket tartalmazza.

- **Értékek**
- **idle**: Semmilyen műveletet nem végzünk
- **switchpump**: Egy pumpa átállítása
- **disconnectpipe**: Egy cső lecsatlakoztatása
- **breakpipe**: Egy cső kilyukasztása
- **repairpump**: Egy pumpa megjavítása
- **repairpipe**: Egy cső megfeszítése
- **stickypipe**: Egy cső ragadóssá tétele
- **slipperypipe**: Egy cső csúszóssá tétele
- **pickuppump**: Egy pumpa felvétele.
- **pickuppipe**: Egy cső felvétele
- **connectpipe**: Egy cső csatlakoztatása
- **placepump**: Egy pumpa letevése
- **move**: Egy játékos mozgatása

### 11.3.7 PipeView

- **Felelősség**

A modellben szereplő **Pipe** objektumoknak a megjelenítésére szolgáló osztály. Ez tárol minden információt, végez minden műveletet, ami egy cső megjelenítéséhez szükséges a csőhálózatban.

- **Ősosztályok**

ElementView

- **Attribútumok**
- **Pipe -pipe**: A játék modelljében levő **Pipe** objektum referenciája, amit ez az objektum megjelenít.
- **Point[] -endpoints**: A cső két végének a koordinátái a grafikus megjelenítésben.
- **Metódusok**
- **+PipeView(Pipe p)**: Az osztály konstruktora. Beállítja a **-pipe** attribútum értékét egy paraméterként megadott cső referenciájára a modellből.

- **void +Update()**: A cső grafikai reprezentációjának a frissítését megvalósító függvény. Újrarajzolja a csövet, a modellben levő csőnek a legfrissebb állapota szerint.
- **void +setEndpoints(Point p1, Point p2)**: Beállítja a cső végpontjainak a koordinátáit.
- **Point[] +GetEndPointsCoordinates()**: Visszatéríti a cső végpontjainak a koordinátáit.
- **bool +containsPoint()**: Ez a metódus megmondja, ha a paraméterként megadott pont a csőhöz tartozik-e. Igazat térít vissza, ha a pont benne van.
- **Point +GetCenterCoordinates()**: Visszatéríti a cső közepét. Ezt úgy számolja ki, hogy a két végpont komponenseinek az átlagát adja vissza.
- **Element +GetElement()**: Getter, amely visszatér az adott PipeView objektum pipe referenciajával, ami az aktuálisan megjelenített csövet mutatja.

### 11.3.8 PlayerView

- **Felelősség**

Absztrakt osztály, ami a modellben szereplő Playerek (játékosok) grafikus megjelenítésére szolgál. Ez tárol minden információt, végez minden műveletet, ami egy játékos megjelenítéséhez szükséges.

- **Attribútumok**

- **Point -place**: A játékos helyének a koordinátáit tárolja.

- **Metódusok**

- **void +Update()**: Ez a metódus egy játékos megjelenítését frissíti. Újrarajzolja a játékost, amelyiket reprezentálja.
- **void SetPlace(Point p)**: Elhelyezi a játékost a paraméterként megadott koordinátára. Beállítja a **-place** attribútum értékét a paraméterként megadott pontra.

### 11.3.9 Point

- **Felelősség**

Ez az osztály a játékbablakban levő koordinátákat reprezentálja 2 dimenzióban.

- **Attribútumok**

- **int -x**: A koordináta x komponense.
- **int -y**: A koordináta y komponense.

- **Metódusok**

- **+Point(int x, int y)**: Az osztály konstruktora. Beállítja a koordináta x és y komponenseit.
- **int getX()**: Visszatéríti a koordináta **x** komponensét
- **int getY()**: Visszatéríti a koordináta **y** komponensét

### 11.3.10 PumpView

- **Felelősség**

A modellben szereplő **Pump** objektumoknak a megjelenítésére szolgáló osztály. Ez tárol minden információt, végez minden műveletet, ami egy pumpa megjelenítéséhez szükséges a csőhálózatban.

- **Ősosztályok**

ElementView

- **Attribútumok**

- **Pump -pump:** A játék modelljében levő **Pump** objektum referenciája, amit ez az objektum megjelenít.

- **Metódusok**

- **+PumpView(Pump p):** Az osztály konstruktora. Beállítja a **-pump** attribútum értékét egy paraméterként megadott pumpa referenciájára a modellből.
- **void +Update():** A pumpa grafikai reprezentációjának a frissítését megvalósító függvény. Újrarájzolja a pumpát, a modellben lévő legfrissebb állapota szerint.
- **Element +GetElement():** Getter, amely visszatér az adott PumpView objektum által megjelenített Pump példánnyal

### 11.3.11 RefreshTimer

- **Felelősség**

Singleton osztály. A grafikus felület megjelenítésének a periodikus frissítését végző osztály.

- **Ősosztályok**

Thread

- **Attribútumok**

- **Metódusok**

- **void +Tick():** A periódusidő letelésekor meghívódó metódus. Ez a függvény hívja meg az összes grafikus elem **Update()** függvényét, amivel minden újrarájzolódik.
- **void +run():** Az osztály szálfüggvénye. Ebben fut egy végtelen ciklus, ami meghívja a **Tick()** metódust, majd vár egy kicsit, majd ismétli ugyanezt.

### 11.3.12 SaboteurView

- **Felelősség**

A modellben szereplő **Saboteur** (szabotörök) objektum grafikus megjelenítését megvalósító osztály. Ez tárol minden információt, végez minden műveletet, ami egy szabótör megjelenítéséhez szükséges a csőhálózatban.

## · Ősosztályok

PlayerView

### · Attribútumok

- **Saboteur -saboteur:** A játék modelljében levő **Saboteur** referenciája, amit ez az objektum megjelenít.

### · Metódusok

- **void +Update():** A szabotőr grafikai reprezentációjának a frissítését megvalósító függvény. Újrarázsolja a szabotőrt a pályán a modellben levő legfrissebb állapota szerint.

### 11.3.13 SourceView

#### · Felelősség

A modellben szereplő **Source** objektumoknak a megjelenítésére szolgáló osztály. Ez tárol minden információt, végez minden műveletet, ami egy forrás megjelenítéséhez szükséges a csőhálózatban.

## · Ősosztályok

ElementView

### · Attribútumok

- **Source -source:** A játék modelljében levő **Source** objektum referenciája, amit ez az objektum megjelenít.

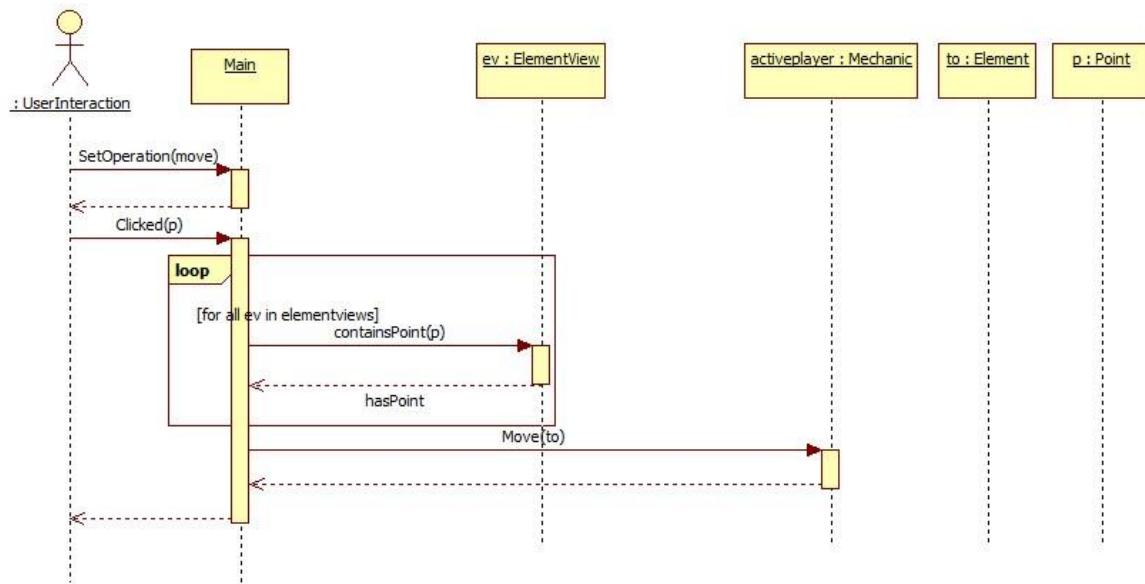
### · Metódusok

- **+SourceView(Source s):** Az osztály konstruktora. Beállítja a **-source** attribútum értékét egy paraméterként megadott forrás referenciájára a modellből.
- **void +Update():** A forrás grafikai reprezentációjának a frissítését megvalósító függvény. Újrarázsolja a forrást, a modellben lévő legfrissebb állapota szerint.
- **Element +GetElement():** Getter, amely visszatér az adott SourceView objektum által megjelenített Source példánnyal.

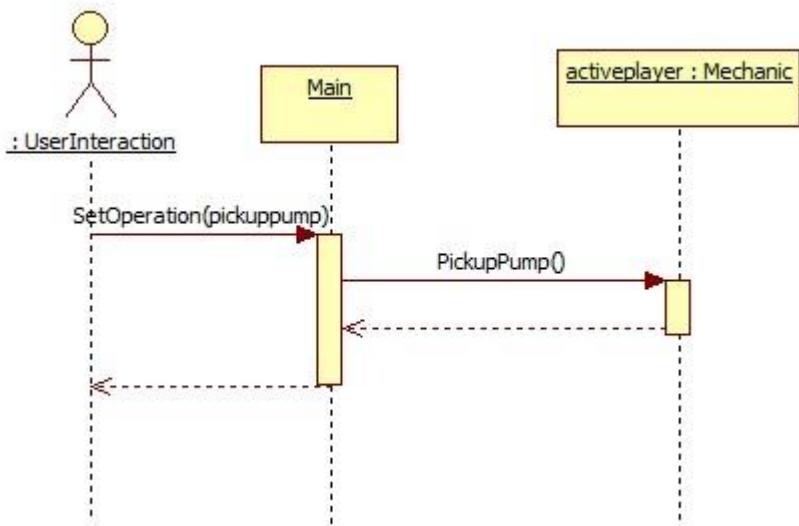
## 11.4 Kapcsolat az alkalmazói rendszerrel

Megjegyzés: A szekvenciákon a Main osztály gyakran számol pontokkal (Point objektum), amiken nem változtat, csak számol velük, így az átláthatóság érdekében nem mindenhol tüntettük fel őket. Ennek ellenére létező objektumok. Emellett a Main osztály megjelenítő objektumokat tartalmazó kollekcióiban történő keresést sem tüntettük fel minden esetben, hiszen az ehhez szükséges loop frame-ek nehezen olvashatóvá, és hosszúvá tettek volna a diagramokat.

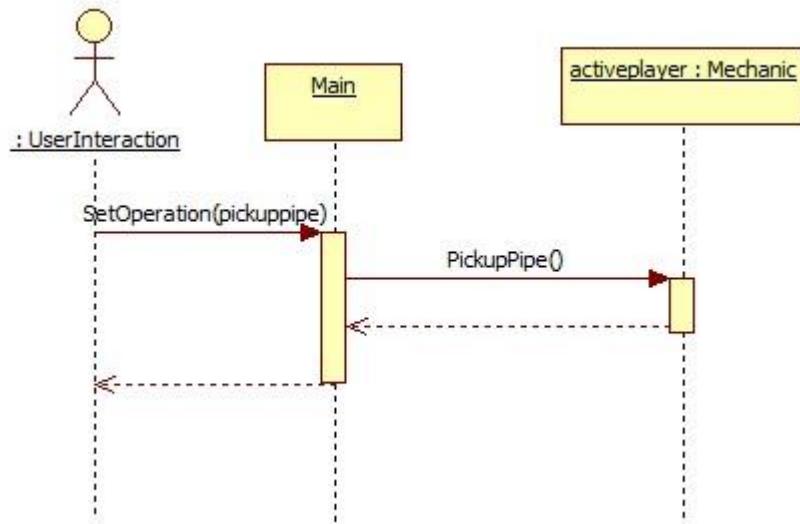
### 11.4.1 Moves



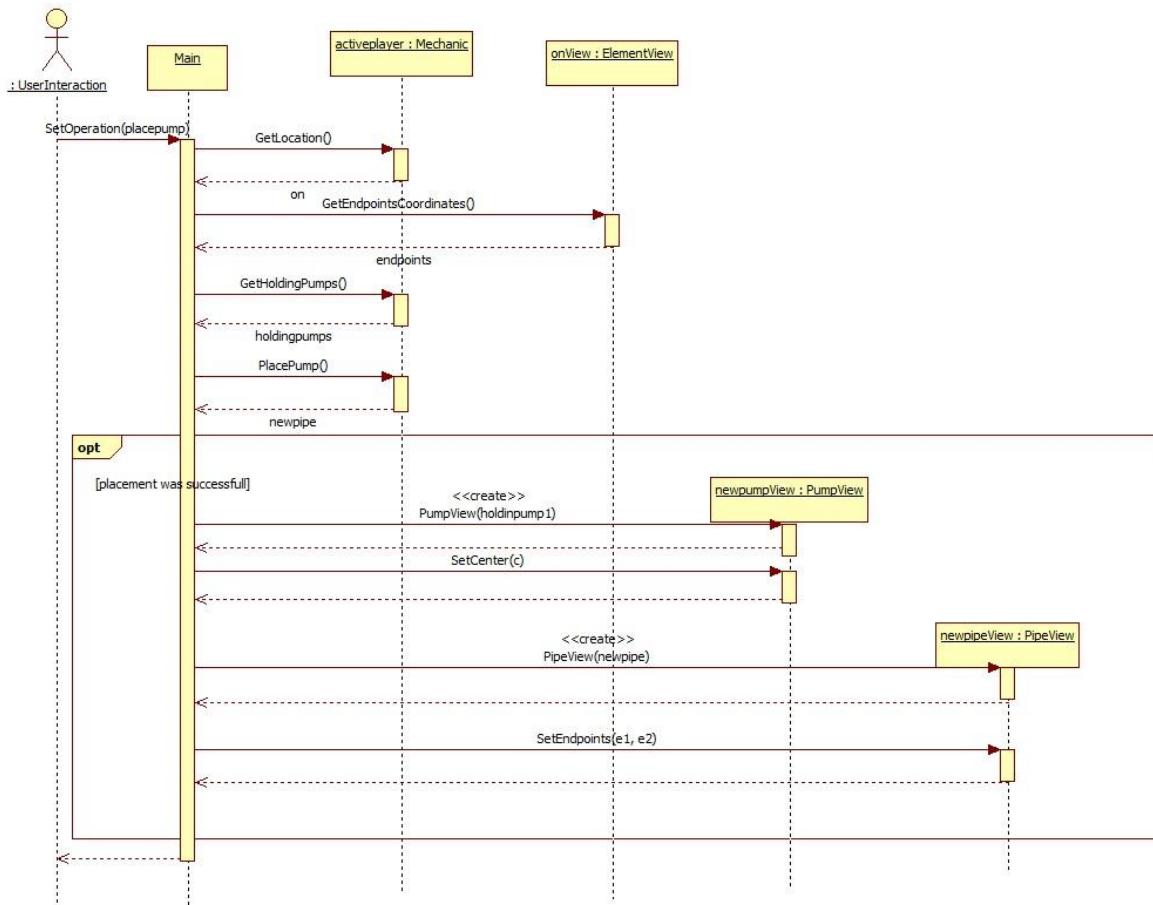
### 11.4.2 Pickup Pump



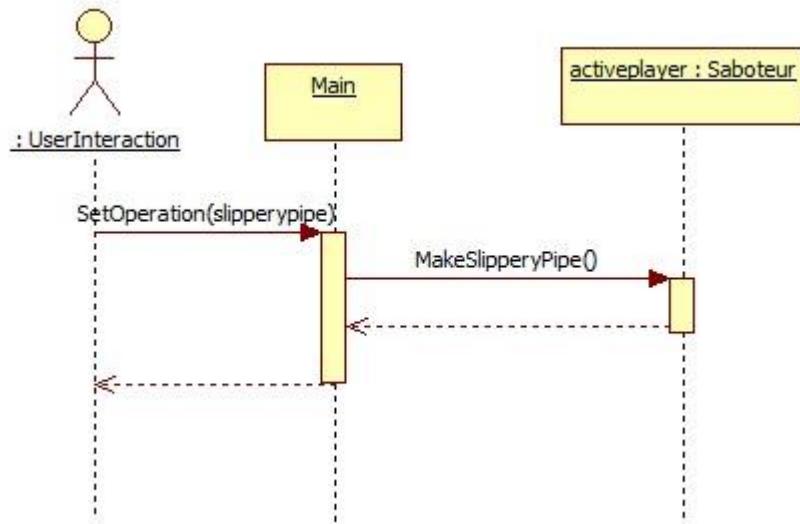
### 11.4.3 Pickup Pipe



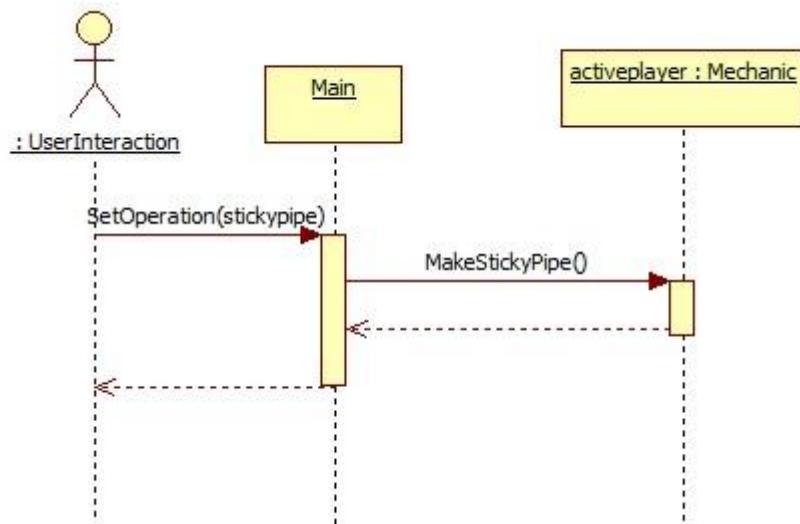
### 11.4.4 Place Pump



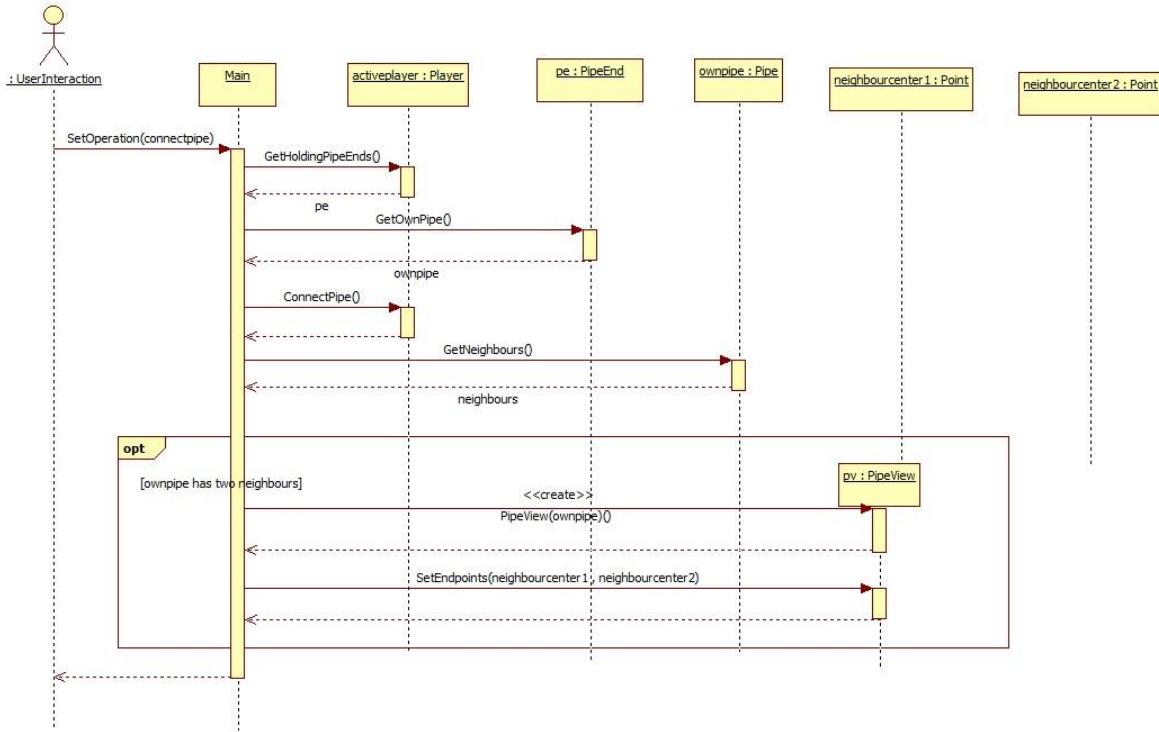
### 11.4.5 Make Pipe Slippery



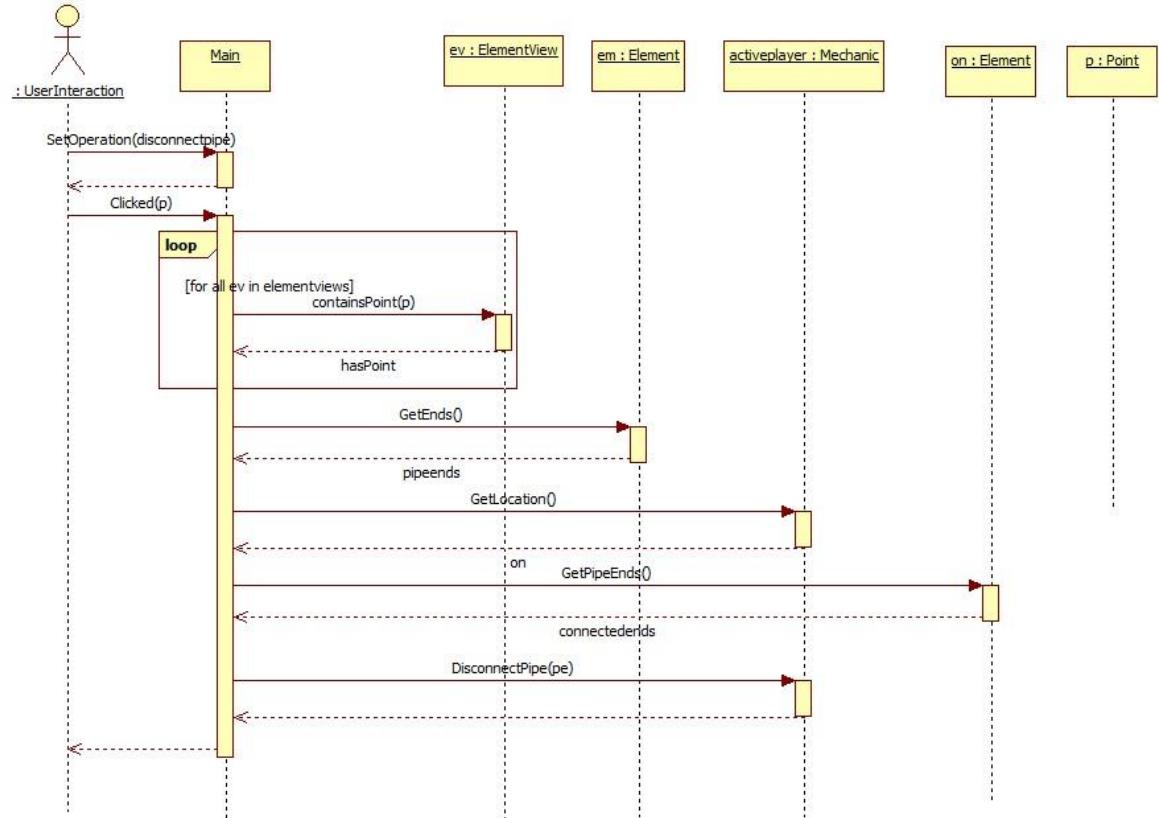
### 11.4.6 Make Pipe Sticky



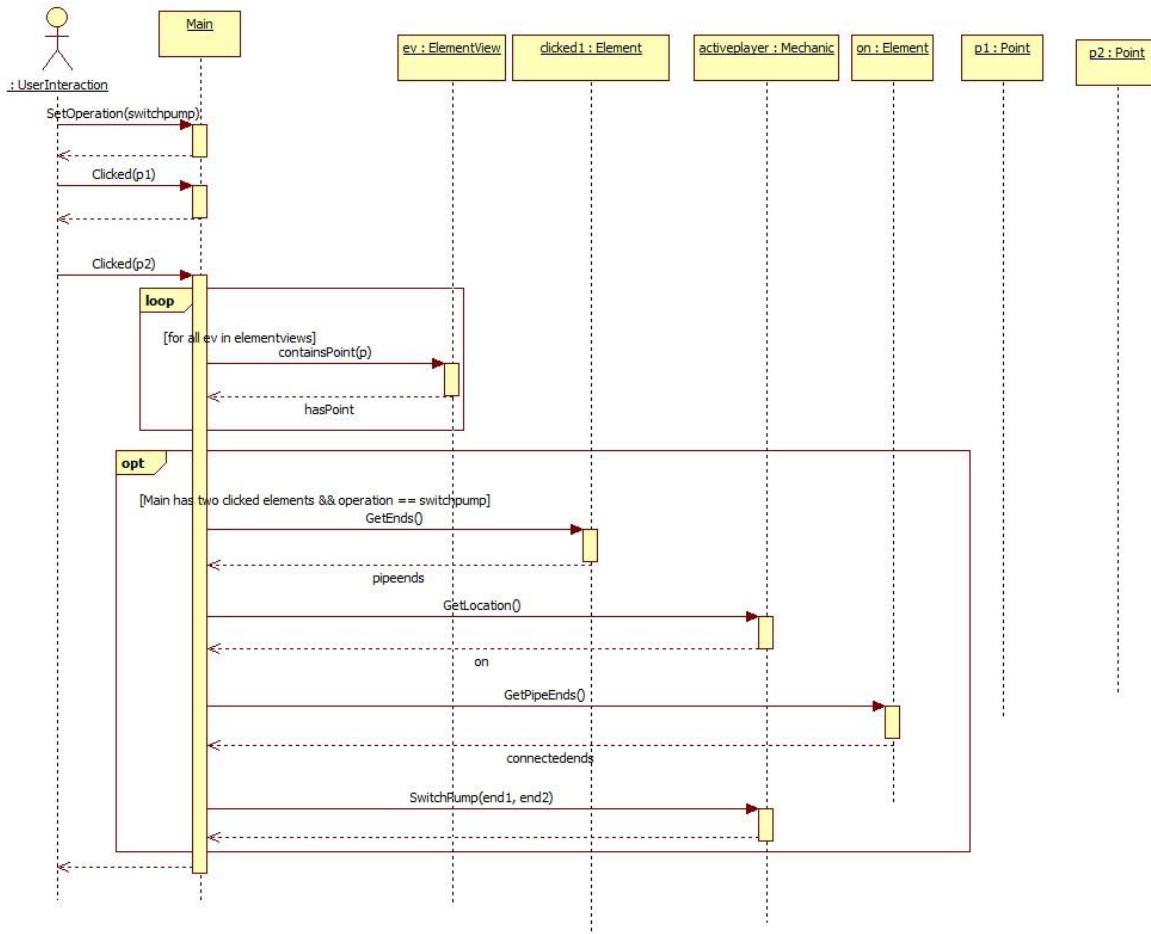
### 11.4.7 Connect Pipe



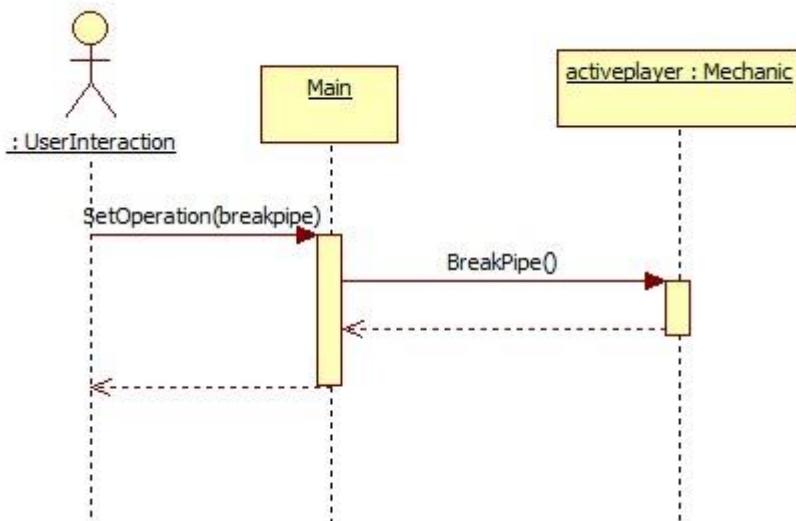
### 11.4.8 Disconnect Pipe



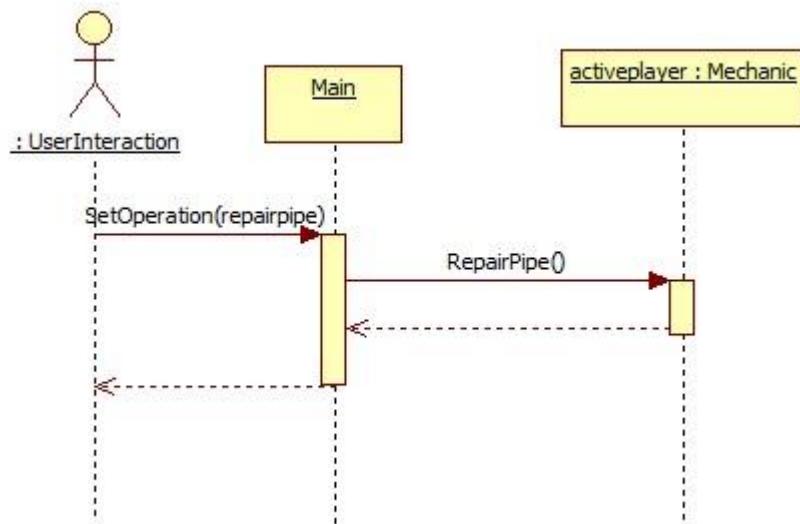
### 11.4.9 Switch Pump



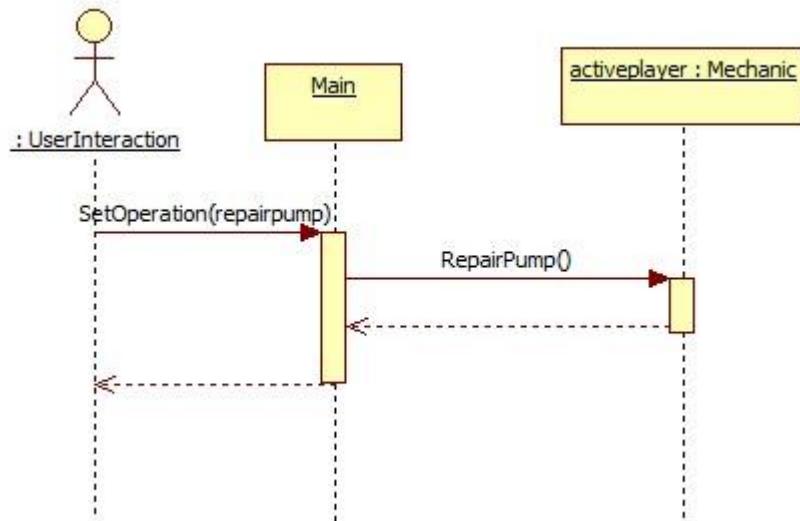
### 11.4.10 Break Pipe



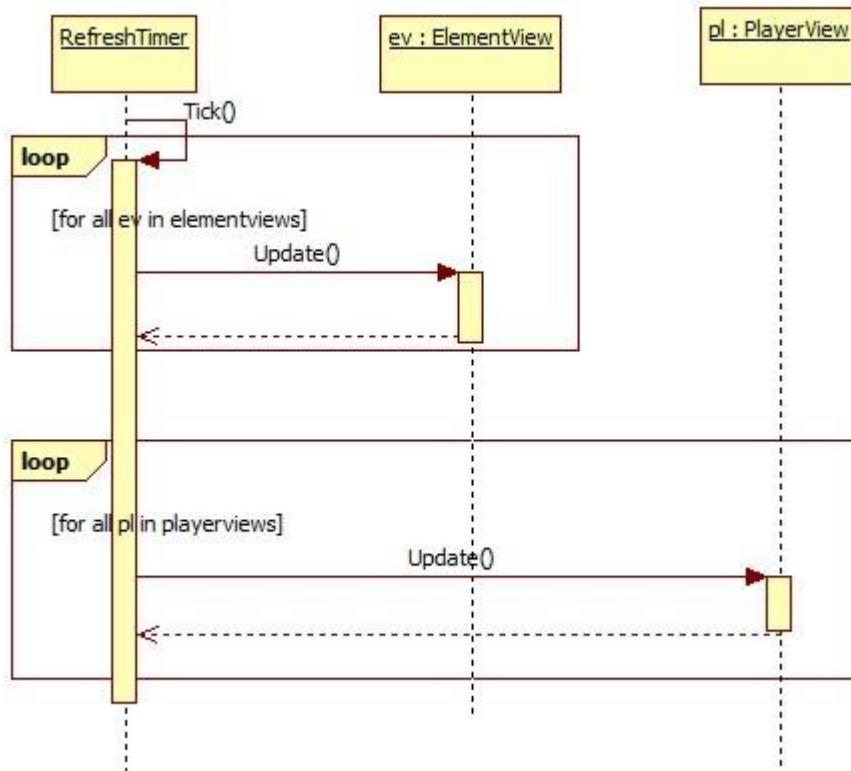
### 11.4.11 Repair Pipe



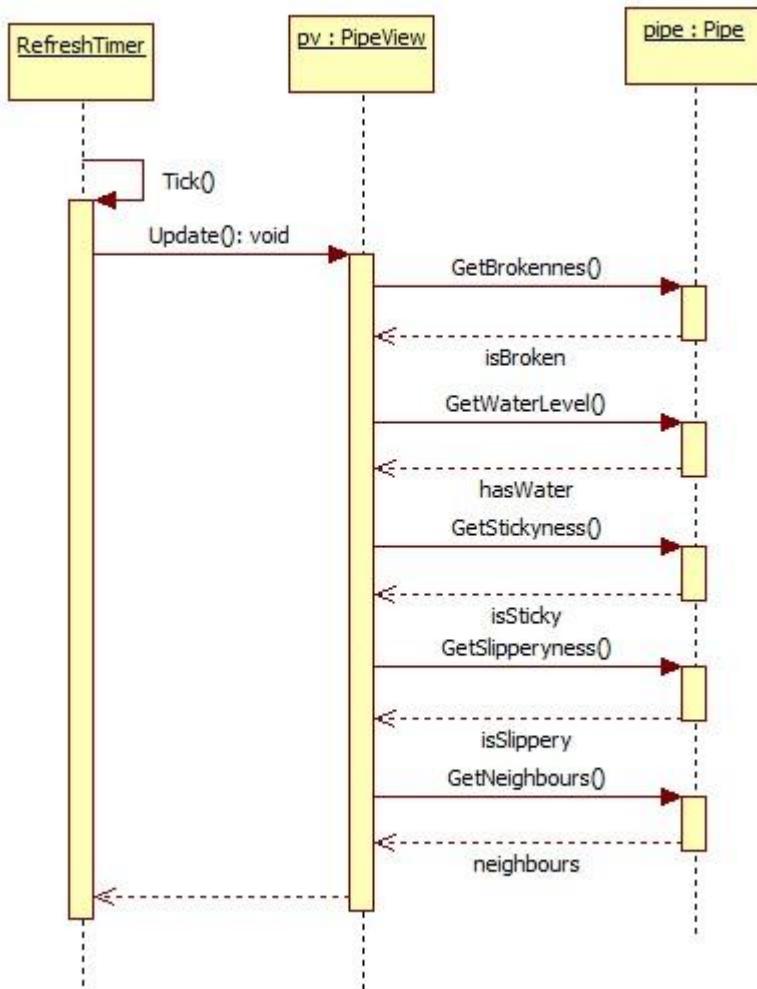
### 11.4.12 Repair Pump



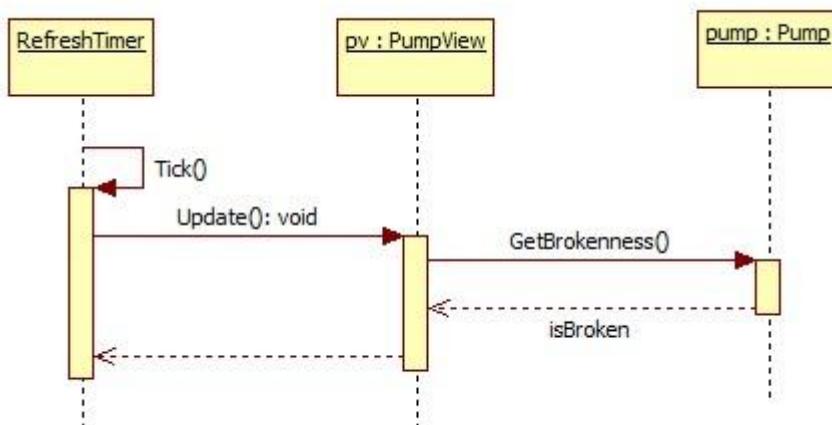
### 11.4.13 Update View objects



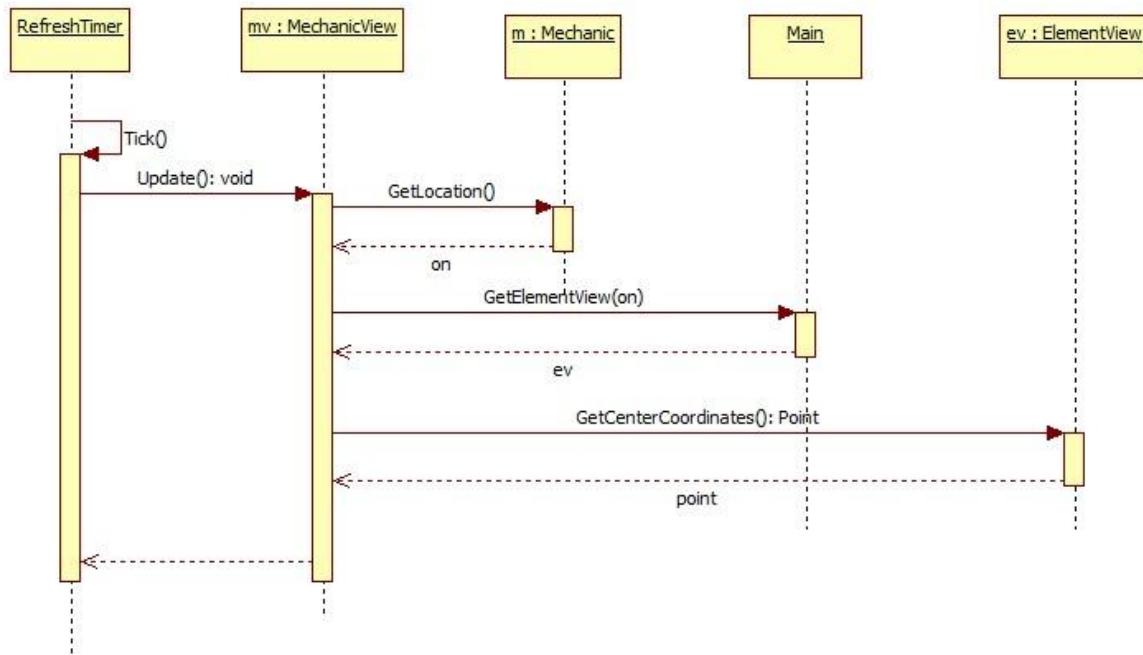
### 11.4.14. Update Pipe



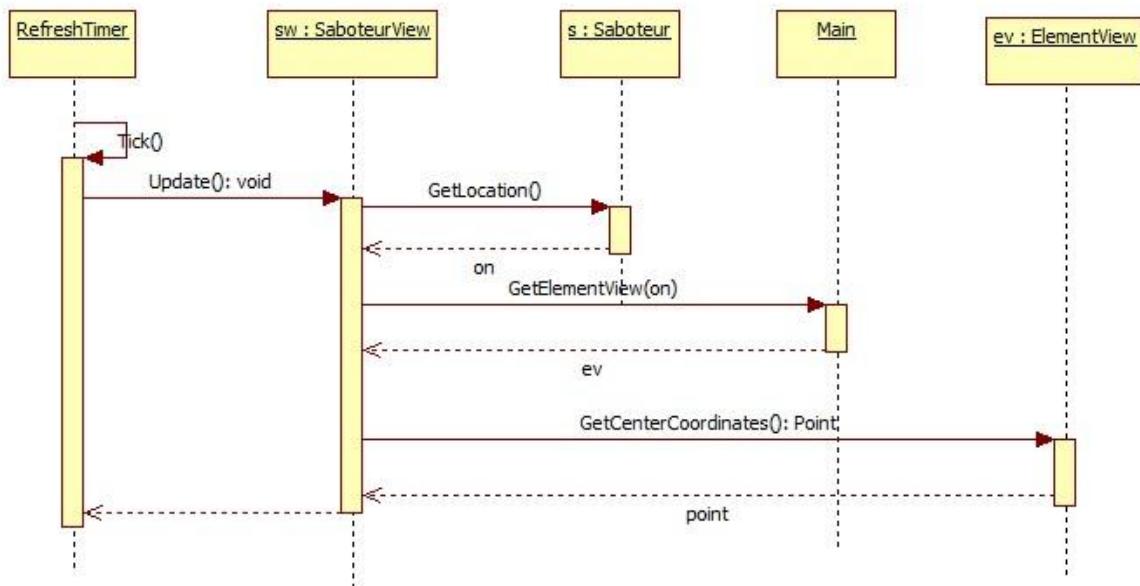
### 11.4.15 Update Pump



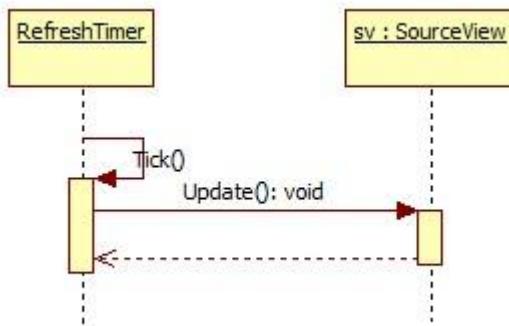
### 11.4.16 Update Mechanic



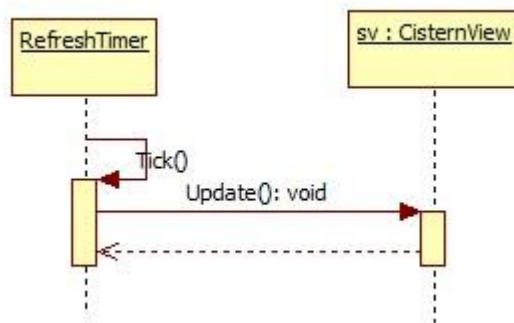
### 11.4.17 Update Saboteur



### 11.4.18 Update Source



### 11.4.19 Update Cistern



## 11.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2023.05.19. 10:00	1 óra	Andó Deé-Lukács Kiss Skáre Vörös	Értekezlet. Döntés: Andó és Kiss megtervezik a grafikus interfész Deé-Lukács megírja az osztályok leírását Vörös: szekvenciadiagramok rajzolása Skáre: Osztálydiagram terve, rajza

2023.05.19. 22:00	4 óra	Skáre	Osztálydiagram tervezése
2023.05.20. 14.20	1 óra	Andó Kiss	Értekezlet. Grafikus felület terveinek egyeztetése Döntés: Andó megcsinálja a grafikus interfész leírását Kiss elkészíti a pálya elemeit, a minta pályát, és a felhasználói felület további elemeit
2023.05.20. 16.00	2 óra	Kiss	Pálya elemeinek elkészítése, minta pálya összerakása
2023.05.21. 12.50	1 óra	Kiss	Felhasználói felület paneleinek elkészítése
2023.05.21: 13:00	5 óra	Vörös	Szekvencia diagramok (1-13 ) rajzolása, osztálydiagram bővítése, architektúra leírások készítése
2023.05.21. 14:00	1 óra	Andó	A grafikus interfész leírása
2023.05.21. 18.00	2,5 óra	Deé-Lukács	Osztályleírások elkészítése
2023.05.21. 18:00	1,5 óra	Andó	A 14, 15, 16,17,18,19 szekvencia diagramok megrajzolása
2023.05.21. 20.30	1,5 óra	Andó  Deé-Lukács  Kiss  Skáre  Vörös	Értekezlet. Elkészült munka ellenőrzése, egyeztetése, utolsó simítások
2023.05.21. 22.00	0.5 óra	Deé-Lukács	A modell osztálydiagramjába a változások beírása

## 13. Grafikus változat beadása

### 13.1 Fordítási és futtatási útmutató

#### 13.1.1 Fájllista

##### Control

Fájl neve	Méret	Keletkezés ideje	Tartalom
GameEnding.java	1 KB	2023.05.29 12:00	A játék lehetséges kimeneteleinek típusait reprezentáló osztály
GameWindow.java	11 KB	2023.05.29. 13.00	A játék ablakát megjelenítő osztály
Main.java	15 KB	2023.05.29. 13.00	A programot irányító osztály
Operation.java	1 KB	2023.05.29 12:00	A lehetséges műveleteket reprezentáló osztály

##### Model

Fájl neve	Méret	Keletkezés ideje	Tartalom
Cistern.java	2 KB	2023.05.10. 10:48	A ciszterna osztály
Element.java	4 KB	2023.05.10. 10:48	Az aktív és passzív elemek űsosztálya
Game.java	47 KB	2023.05.10. 10:48	A játékért felelős osztály
ISteppable.java	1 KB	2023.05.10. 10:48	A léptethető dolgok interfésze
Mechanic.java	3 KB	2023.05.10. 10:48	A szerelők osztálya
Node.java	3 KB	2023.05.10. 10:48	Az aktív elemek űsosztálya
Pipe.java	7 KB	2023.05.10. 10:48	A cső osztálya
PipeEnd.java	3 KB	2023.05.10. 10:49	A csővég osztálya
Player.java	7 KB	2023.05.10. 10:49	A játékosok űsosztálya
Pool.java	1 KB	2023.05.10. 10:49	A medence (pontgyűjtő) osztálya
Pump.java	5 KB	2023.05.10. 10:49	A pumpa osztálya

Saboteur.java	1 KB	2023.05.10. 10:49	A szabotőr osztálya
Source.java	1 KB	2023.05.10. 10:49	A forrás osztálya
Timer.java	5 KB	2023.05.10. 10:49	Az időzítő osztály

**View**

Fájl neve	Méret	Keletkezés ideje	Tartalom
Canvas.java	5 KB	2023.05.29 12:00	A rajzolási területet megvalósító osztály
CisternView.java	2 KB	2023.05. 26. 18.00	A ciszternák megjelenítéséért felelős osztály
DesertMath.java	1 KB	2023.05.29 12:00	Matematikai műveleteket megvalósító segédosztály
ElementView.java	2 KB	2023.05.26 17:00	A játékban szereplő mezők megjelenítéséért felelős (ős)osztály
MechanicView.java	2 KB	2023.05.28. 22:00	A játékban szereplő szerelők megjelenítésérrt felelős osztály
PipeView.java	4 KB	2023.05.29. 00:00	A csövek megjelenítéséért felelős osztály
PlayerView.java	1 KB	2023.05.28. 22:00	A játékosok megjelenítéséért felelős absztrakt osztály
Point.java	1 KB	2023.05.26 17:00	A vászonon kirajzolt pontokat reprezentáló osztály
PumpView.java	2 KB	2023.05. 26. 18.00	A pumpák megjelenítéséért felelős osztály
RefreshTimer.java	3 KB	2023.05.29. 13.00	A periodikus frissítéséért felelős singleton osztály
SaboteurView.java	2 KB	2023.05.28. 22:00	A játékban szereplő szabotőrok megjelenítéséért felelős osztály
SourceView.java	2 KB	2023.05. 26. 18.00	A források megjelenítéséért felelős osztály

### 13.1.2 Fordítás és telepítés

#### 1. Lépés: A környezet előkészítése

Első lépésként fel kell telepíteni a Java 18-at a számítógépre.

A gépen lévő Java JDK a következő parancssal ellenőrizhető a Terminál alkalmazásban

```
java -version
```

Ha a következőt látjuk, akkor további teendőnk a környezettel nincsen.

```
C:\Users\cloud\Downloads\src>java --version
java 18.0.2.1 2022-08-18
Java(TM) SE Runtime Environment (build 18.0.2.1+1-1)
Java HotSpot(TM) 64-Bit Server VM (build 18.0.2.1+1-1, mixed mode, sharing)
```

Ha nem ezt látni, akkor a következő linkről lehet letölteni a megfelelő verziójú Java fejlesztői csomagot. (64-bites Windows platformra):

[https://download.oracle.com/java/18/archive/jdk-18.0.2.1\\_windows-x64\\_bin.exe](https://download.oracle.com/java/18/archive/jdk-18.0.2.1_windows-x64_bin.exe)

Ezután gondoskodni kell arról, hogy a frissen telepített Java hozzá legyen adva a PATH nevű környezeti változóhoz. Ennek az egyik módja a következő parancs lefuttatása parancssorban:

```
set PATH=%PATH%;C:\Program Files\Java\jdk-18.0.2.1\bin
```

Itt a feketével jelölt részt ki kell cserálni a feltelepített Java pontos verziójára.

#### 2. Lépés: Forrásfájlok letöltése

A konzulens által megadott helyről le kell tölteni a Capybara.zip fájlt, ami tartalmazza a prototípus program forráskódját, a tesztelő szkriptet és a teszteléshez használt be- és kimeneti fájlokat.

Ezután a Start menü Terminál/PowerShell alkalmazását kell elindítani, és kiadni a következő parancsot:

```
cd C:\Users\cloud\Downloads
```

Ezután tömörítsük ki a mappát:

```
tar -x -f capybara.zip
```

Ezután navigálunk bele a kitömörített mappába

```
cd capybara\src
```

### 3. Lépés: A program fordítása

Nyissuk meg egy parancssor programot, majd navigáljunk el a kitömörített fájlokat tartalmazó mappába. Ezután a következő parancsot futtatva, a program lefordul.

```
javac -d .\output .\Model\*.java .\View\*.java  
.Control\*.java
```

Linux alatt a '\ helyett a '/' karaktert használjuk.

#### 13.1.3 Futtatás

Windows alatt a következő parancssal futtatható a program:

```
java -cp .\output Control.Main
```

Ha esetleg Linux rendszeren futtatjuk, akkor a '\ helyett a '/' karaktert használjuk.

## 13.2 Értékelés

Tag neve	Tag neptun	Munka százalékban
Andó Viola	H51B9J	20%
Kiss Blanka Zselyke	C6037J	20%
Skáre Erik	Z7ZF6D	20%
Deé-Lukács András Gergely	JHFLXZ	20%
Vörös Vilmos	HW921K	20%

### 13.3 Napló

Kezdet	Időtartam	Résznevők	Leírás
2023.05.24. 14:00	2 óra	Andó Deé-Lukács Kiss Skáre Vörös	Értekezlet. Döntés:  Andó: Megcsinálja a PlayerView, MechanicView, SaboteurView osztályokat  Deé-Lukács: Main, GameWindow implementálása, Controller mechanizmus implementálása, szerializálás  Kiss: Megírja a CisternView, PumpView, SourceView osztályokat és a dokumentációt  Skáre: megírja PipeView-t  Vörös: ElementView ősosztály, megjelenítést segítő saját JPanel segédosztály
2023.05.26 17:00	1 óra	Vörös	ElementView, Point osztály implementálás
2023.05. 26. 18.00	1 óra	Kiss	CisternView, PumpView, SourceView
2023.05.28. 22:00	1.5 óra	Andó	Tevékenység: PlayerView MechanicView, SaboteurView implementálása
2023.05.29. 00:00	2,5 óra	Skáre Erik	PipeView implementálása
2023.05.29. 13.00	5 óra	Deé-Lukács	Main, GameWindow implementálása, Controller mechanizmus implementálása
2023.05.29 12:00	2 óra	Vörös	Canvas implementálása

2023.05.29 21:00	3 óra	Andó, Deé-Lukács, Vörös, Kiss, Skáre	Értekezlet: tesztelés, debuggolás, javítások
2023.05.30. 20.00	3 óra	Deé-Lukács	Szerializálás implementálása, modellhibák javítása
2023.05.30 23.00	2 óra	Kiss	Dokumentáció elkészítése

## 14. Összefoglalás

### 14.1 A projektre fordított összes munkaidő

Tag neve	Munkaidő (óra)
Andó Viola	87,1
Deé-Lukács András Gergely	93,6
Kiss Blanka Zselyke	88,1
Skáre Erik	89,35
Vörös Vilmos	92,6
<b>Összesen</b>	<b>450,75</b>

- A feltöltött programok forrássorainak száma**

Fázis	Kódsorok száma
Szkeleton	1066
Prototípus	1693
Grafikus változat	2409
<b>Összesen</b>	<b>5168</b>

### 14.2 Projekt összegzés

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

A projekt során számos fontos tanulságot vonhattunk le. Konkrétan azt tanultuk, hogy a hatékony kommunikáció és az összehangolt együttműködés kulcsfontosságú a sikeres eredmény eléréséhez. Emellett azt tapasztaltuk, hogy az összes csapattagnak egyértelműen kell értenie a feladatokat és át kell látnia a teljes képet, hogy önállóan is tudjunk dolgozni. Valamint rendszeres egyeztetésekre van szükség az aktuális helyzetről és a felmerülő problémákról, mert legtöbbször voltak olyan tényezők, amelyeket nem vettünk figyelembe a közös tervezésnél, és így a terv megváltoztatására volt szükség. Általánosságban pedig felismerhettük, hogy az átgondolt tervezés, a felelősségek elosztása és a rugalmasság képessége elengedhetetlenek a projekt hatékony végrehajtásában.

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

A projekt során azt tapasztaltuk, hogy nehéz volt a feladatokat egyenletesen elosztani, különösen az elején. Gyakran fordult elő, hogy a részfeladatok egymástól függtek, így szükség volt a többi csapattag eredményeire, mielőtt folytathattuk volna a saját munkánkat. Emiatt gyakran szükség volt a csapattagok közötti szoros együttműködésre és információcserére. Ez kihívást jelentett a hatékony időmenedzsment szempontjából. Emellett a feladatok időtartamának becslése is problémát okozott, mivel sokszor nem tudtuk pontosan megítélni, mennyi időt vesznek majd igénybe. Ez néha azt eredményezte, hogy valaki többet dolgozott, mint a többiek, de igyekeztünk ezt kiegyenlíteni a következő beadandó feladatoknál. Így a projekt legnehezebb része a határidők betartása volt. Emellett a csapat együttműködésének koordinálása és az esetleges nézeteltérések kezelése is komplex feladat volt.

A legkönnyebb rész a projekt során az volt, amikor az előkészítő munkálatokat végeztük, és a tervezek, stratégiák kidolgozása zajlott. Ez az időszak kevésbé volt stresszes, mivel még nem kellett konkrét feladatokat végrehajtanunk, hanem inkább ötleteket és tervezet fogalmaztunk meg. Ez lehetővé tette számunkra, hogy kreatívan gondolkodjunk és a csapat tagjainak különböző nézőpontjait figyelembe véve alakítsuk ki a megvalósítási stratégiát. Ebben a fázisban nagyobb szabadságot és rugalmasságot élvezhettünk, ami pozitív hatással volt a motivációnakra és a csapat dinamikájára.

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

Igen, összhangban állt.

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

-

**14.2.5 Milyen változtatási javaslatuk van?**

A beadások mennyiségét csökkentenénk, ugyanis a félév során nehéz volt összehangolni a többi kötelezettségünkkel a heti 1 beadandót.

**14.2.6 Milyen feladatot ajánlanának a projektre?**

-

**14.2.7 Egyéb kritika és javaslat**

-