3. Analízis modell (I. változat)

5 – runtime_error

Konzulens:
Dobos-Kovács Mihály

Csapattagok

Mizser Ádám Zoltán	SHKGZW	mizser.adam@gmail.com
Tepliczky Olivér	WB6LC5	tepliczkyo@edu.bme.hu
Fekete Álmos Valér	KR5WPC	feketealmos0@gmail.com
Váradi Kristóf	BP17IB	kristofvaradi@edu.bme.hu
Sasvári Szabolcs Attila	**TWOZG6**	**szabolcs.attila.sasvari@edu.bme.hu**

2023.03.13.

3. Analízis modell kidolgozása

3.1 Objektum katalógus

fontos megjegyzés: A Game (játék) objektumunk a Controller-ünk. A nem játékosokhoz köthető Use-Case-eket ő valósítja meg, illetve ő a híd a View és Model között.
Bár az analízis modellnek nem feltétlenül kellene a része lennie, mi mégis megjelenítjük itt, mert számos szekvenciadiagramhoz elengedhetetlen, és ez a tervezést jelentősen megkönnyíti.

3.1.1 Játék

A játékmenetet kezeli (vizsgálja, hogy véget ért-e a játék, valamint a köröket vezérli). Számontartja és módosítja a csőrendszert (turn-önként véletlenszerűen pumpákat ront el, round-onként pedig ciszternák szomszédos üres helyeire új csöveket helyez le). Turn-ök elején folyatja a vizet a forrásokból a szabad végű / lyukas csövekig vagy ciszternáig, illetve pontokat tud osztani a csapatoknak, amit szintén számontart.

Irányokat / célpontokat tud bekérni a felhasználótól. Játékosmozgatást, cső- és pumpalehelyezést, illetve csőfelvevést kezdeményezhet vele. Pumpát is tud adni a jelenlegi játékosnak, amennyiben egy szerelő, ciszernán áll és üres a tárolója.

3.1.2 Csőrendszer

Számontartja a játék elemeit (csöveket, pumpákat, forrásokat és ciszternákat). Ezek közül eltávolíthat csöveket úgy, hogy ne essen komponensekre a csőrendszer, mint gráf. Az eltávolított csövet oda tudja adni annak a szerelőnek, aki az eltávolítást kezdeményezte. A csövek és pumpák elhelyezését is kezeli. Vizsgálja, hogy ezek a műveletek lehetségesek-e egyáltalán.

3.1.3 Forrás

Olyan elem, amely vizet juttat a vele szomszédos csövekbe, azaz a számontartott kimeneteibe, illetve a sivatagba, ha nincsen szomszédja az adott irányba. Ahányszor előfordul az utóbbi, annyi pontot oszt a szabotőrök csapatának.

Számontartja a játékosokat, akik rajta állnak. Játékosokat tud feltenni, illetve eltávolítani magáról.

3.1.4 Ciszterna

Olyan elem, amely számontartja a bemeneteit. Amennyi csőből folyik víz az adott ciszternába, annyi pontot oszt a szerelők csapatának.

Számontartja a játékosokat, akik rajta állnak. Játékosokat tud feltenni, illetve eltávolítani magáról.

3.1.5 Pumpa

Olyan elem, amely számontartja a bemenetét, a kimenetét és a szomszédait. A szomszédai közül be tudja állítani, hogy melyik a bemenete és melyik a kimenete. Vizet képes juttatni a bemeneti csövéből, önmagán keresztül (ő maga is tárol vizet) a kimeneti csövébe. El tud romlani, és ekkor nem képes víz átengedésére. Amikor víz megy át rajta, pontot oszthat a szabotőrők csapatának, ha a kimenete egy szabad végű / lyukas cső vagy sivatag (nincs kimenete).

Számontartja a játékosokat, akik rajta állnak ezzel együtt játékosokat tud feltenni, illetve eltávolítani magáról.

3.1.6 Cső

Olyan elem, amely számontartja a bemenetét és kimenetét, és az előbbiből az utóbbiba vizet képes juttatni. Tárolja a benne lévő vizet. Ki lehet lyukasztani, és meg lehet javítani. Lyukas állapotában nem képes vizet átengedni magán. Amikor vizet enged át, pontot oszthat a szabotőrök csapatának, ha a kimenete egy lyukas cső vagy sivatag (nincs kimenete).

Számontartja a játékost, aki rajta áll, valamint egy játékost tud feltenni, illetve eltávolítani magáról (egyszerre egy játékos lehet egy csövön).

3.1.7 Szerelő

Számon tartja azt az elemet, amin éppen áll, valamint szomszédos elemekre tud lépni, hogyha ez lehetséges (azaz akkor, ha a szomszédos csövön esetleg nem áll már másik játékos).

Manipulálni tudja az elemet, amin éppen áll (csövet és pumpát tud javítani, illetve működő pumpát tud átállítani). Át képes lépni szomszédos elemekre. Ha éppen ciszternán áll, lehetősége van átlépni róla egy másikra.

A tárolója tárolhat vagy egy csövet, vagy egy pumpát. El tud távolítani egy szomszédos, üres csövet, hogy ha a tárolója üres, valamint ezt az csövet később elhelyezni egy szomszédos, üres helyre. Pumpát olyan cső helyére tud letenni, aminek a bemenete és a kimenete is egy cső.

3.1.8 Szabotőr

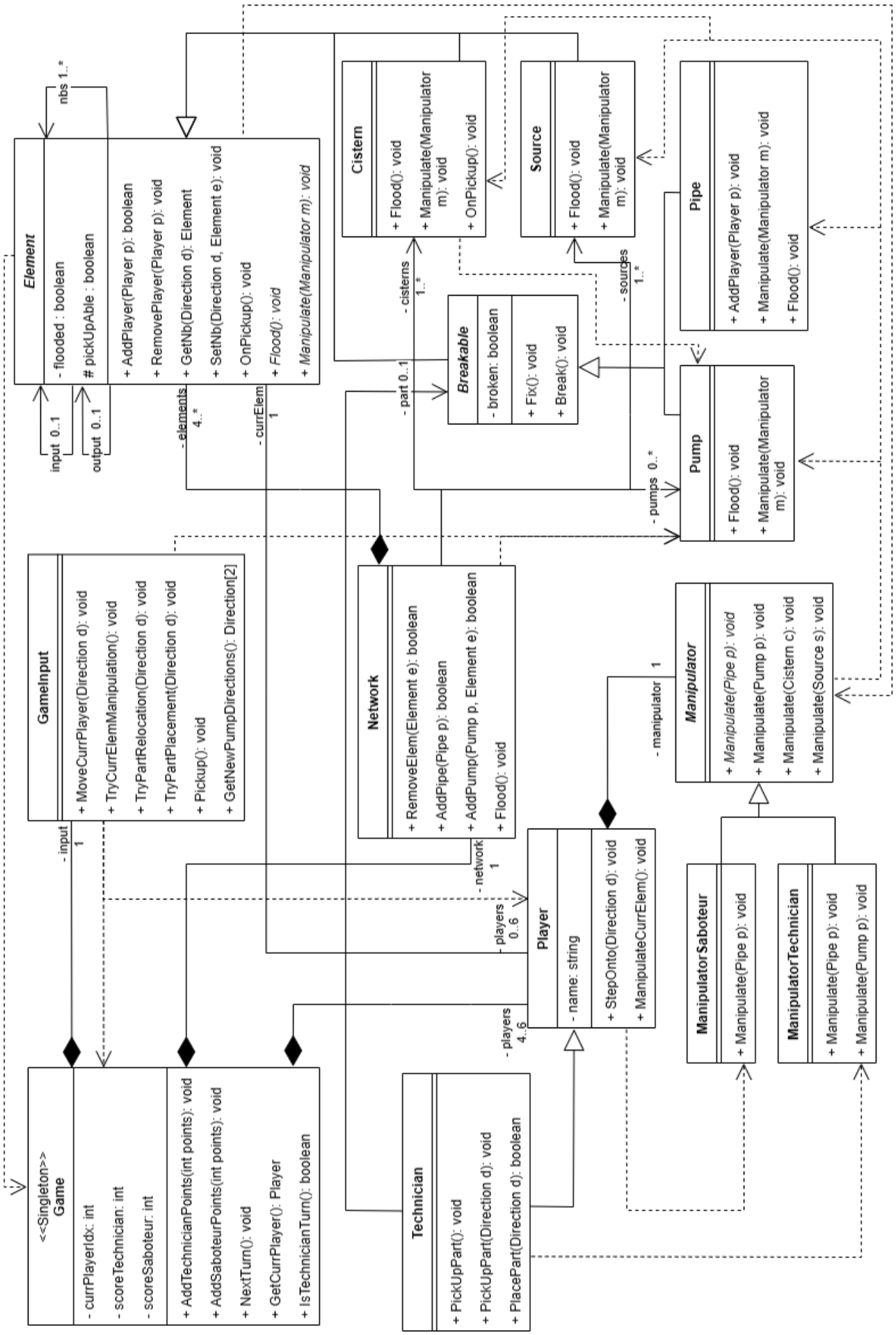
Számon tartja azt az elemet, amin éppen áll, valamint szomszédos elemekre tud lépni, hogyha ez lehetséges (azaz akkor, ha a szomszédos csövön esetleg nem áll már másik játékos).

Manipulálni tudja az elemet, amin éppen áll (csövet tud lyukasztani, illetve pumpát tud átállítani). Át képes lépni szomszédos elemekre. Ha éppen ciszternán áll, lehetősége van átlépni róla egy másikra.

3.2 Statikus struktúra diagramok

fontos megjegyzések:

- *A geometriát igyekeztünk eltávolítani. A Direction típus a modell szempontjából tetszőleges absztrakció lehet.*
- *A triviális (paraméter nélküli/egyparaméteres) setter/getter függvényeket, viszont az "Osztályok leírása" szekcióban fel vannak ezek is sorolva.*
- *Ha egy ősnek van egy dependeciája, akkor az érvényes minden leszármazottra. A dependenciák csak akkor vannak behúzva, ha nincsen erősebb kapcsolat az osztályok között (leszármazás, kompozíció, aggregáció, asszociáció).*


Element
- flooded : boolean
pickUpAble : boolean
+ AddPlayer(Player p): boolean
+ RemovePlayer(Player p): void
+ GetNb(Direction d): Element
+ SetNb(Direction d, Element e): void
+ OnPickup(): void
+ Flood(): void
+ Manipulate(Manipulator m): void
nbs 1..*
input 0..1
output 0..1
- elements 4..*
- currElem 1
Cistern
+ Flood(): void
+ Manipulate(Manipulator m): void
+ OnPickup(): void
- cisterns 1..*
Source
+ Flood(): void
+ Manipulate(Manipulator m): void
- sources 1..*
Pipe
+ AddPlayer(Player p): void
+ Manipulate(Manipulator m): void
+ Flood(): void
Breakable
- broken: boolean
+ Fix(): void
+ Break(): void
- part 0..1
Pump
+ Flood(): void
+ Manipulate(Manipulator m): void
- pumps 0..*
GameInput
+ MoveCurrPlayer(Direction d): void
+ TryCurrElemManipulation(): void
+ TryPartRelocation(Direction d): void
+ TryPartPlacement(Direction d): void
+ Pickup(): void
+ GetNewPumpDirections(): Direction[2]
- input 1
Network
+ RemoveElem(Element e): boolean
+ AddPipe(Pipe p): boolean
+ AddPump(Pump p, Element e): boolean
+ Flood(): void
- network 1
Manipulator
+ Manipulate(Pipe p): void
+ Manipulate(Pump p): void
+ Manipulate(Cistern c): void
+ Manipulate(Source s): void
- manipulator 1
Player
- name: string
+ StepOnto(Direction d): void
+ ManipulateCurrElem(): void
- players 0..6
- players 4..6
ManipulatorSaboteur
+ Manipulate(Pipe p): void
ManipulatorTechnician
+ Manipulate(Pipe p): void
+ Manipulate(Pump p): void
<<Singleton>>
Game
- currPlayerIdx: int
- scoreTechnician: int
- scoreSaboteur: int
+ AddTechnicianPoints(int points): void
+ AddSaboteurPoints(int points): void
+ NextTurn(): void
+ GetCurrPlayer(): Player
+ IsTechnicianTurn(): boolean
Technician
+ PickUpPart(): void
+ PickUpPart(Direction d): void
+ PlacePart(Direction d): boolean

3.3 Osztályok leírása

3.3.1. Breakable

- **Felelősség**

Ez az elem tönkre tud menni/kilyukadni, valamint ez által meg is lehet javítani.
Ilyen elemet tudnak tárolni a szerelők (más néven part).
(Ez egy absztrakt osztály.)

- **Ősosztályok**

  Element

- **Attribútumok**
  - **- broken: boolean** az adott elem hibás-e vagy sem, azaz át tud ereszteni-e magán vizet.

- **Metódusok**
  - **+ void Fix()** : Hibás elem megjavítására szolgáló függvény.
  - **+ void Break()** : Épp elem tönkretevésére szolgáló függvény.
  - **+ void GetBroken()** : Visszaadja, hogy az adott elem törött-e.
  - **+ boolean GetPickUpAble()** : Visszaadja, hogy az elem felvehető-e. Az Element ősbéli megvalósításhoz hozzá vesz egy új feltételt: ne legyen törött sem a part, hogy felvehető legyen.

3.3.2. Cistern

- **Felelősség**

Amennyi csőből folyik víz ebbe az elembe, annyi pontot oszt a szerelők csapatának.
Erről az elemről a játékosok közvetlen más ciszternákra tudnak lépni.

- **Ősosztályok**

  Element

- **Metódusok**
  - + **void OnPickUp()** : megpróbál pumpát adni a soron lévő játékos tárolójába.
  - **+ void Flood()** : Vízzel tölti fel magát, és pontot ad a szerelőknek. Úgy van felüldefiniálva, hogy ő már nem folyat más elemekbe vizet (végpont).
  - **+ void Manipulate(Manipulator m)** : az átvett manipulátorral manipulálja ezt a konkrét elemet.

3.3.3. Element

- **Felelősség**

Player-eket tud felrakni, illetve eltávolítani magáról, és számon is tartja a rajta tartózkodókat. Tárolja a bemenet/kimenetét, valamint a szomszédjait is, ezeket tudja változtatni, illetve lekérdezni. Továbbá tárolja, hogy van-e víz benne.

- **Attribútumok**
  - **- flooded: boolean** : Értéke igaz, ha víz van benne (jelenleg vizet tárol).
  - **# pickUpAble: boolean** : Értéke igaz, ha az adott elem felvehető, különben hamis.
  - **- players: Player[0..6]** : Azon játékosok halmaza, amelyek az adott elemen tartózkodnak
  - **- nbs: Element[1..*]** : A vele szomszédos elemekre hivatkozik.
  - **- input: Element[0..1]**: A bemenő szomszédra hivatkozik, ha van ilyen.
  - **- output: Element[0..1]** : A kimenő szomszédra hivatkozik, ha van ilyen.

- **Metódusok**
  - **+ void AddPlayer(Player p)** : Felhelyez magára egy játékost, ha ez lehetséges a rajta állók száma szerint.
  - **+ void RemovePlayer(Player p)** : Eltávolítja a számontartott játékosai közül az átadottat.
  - **+ Element GetNb(Direction d)** : Megadott irányba lévő szomszédos elemét adja vissza, ha van ilyen.
  - **+ void SetNb(Direction d, Element e)** : Megadott irányú szomszédjának állítja be az átadott elemet.
  - + **void OnPickup()** : Ha egy elem támogatni akarja azt a funkcionalitást, hogy egy szerelő rajta állva egy part-ot kaphasson a tárolójába, akkor ezt a függvényt kell felülírnia. Alapértelmezett megvalósítása üres törzsű függvény (alapjáraton nem támogatják az elemek ezt a funkcionalitást).
  - ***+ void Flood()*** *: Absztrakt metódus, amely a származtatott osztályokban felül lesz írva a szerint, hogy hogyan eresztik át magukon a vizet.*
  - ***+ void Manipulate(Manipulator m)*** *: Absztrakt metódus, mivel az elem konkrét típusától függ a lefolyása. Az átadott manipulátorral manipulálja az elemet.*
  - **+ boolean GetPickUpAble()** : Visszaadja, hogy az elem felvehető-e („pickUpAble”, nincs benne víz és nem áll rajta egyetlen játékos sem).
  - **+ int GetNbCnt()** : A szomszédos elemek számát adja vissza.
  - **+ int GetPlayerCnt()** : A rajta tartózkodó játékosok számát adja vissza.
  - **+ boolean GetFlooded()** : Igazat ad vissza, ha az víz van benne (vagy víz folyik ki belőle).
  - **+ Element GetInput()** : Visszaadja a bemeneti elemét.
  - **+ Element GetOutput()** : Visszaadja a kimeneti elemét.
  - **+ void SetFlooded(boolean f)** : Az átadott érték szerint állítja be, hogy van-e víz jelenleg benne.
  - **+ void SetInput(Element e)** : A bemenetét az átadott elemre állítja be.
  - **+ void SetOutput(Element e)** : A kimenetét az átadott elemre állítja be.

3.3.4. Game

- **Felelősség**

A játékmenetet kezeli (vizsgálja, hogy véget ért-e a játék, valamint a köröket vezérli). Számontartja és módosítja a csőrendszert (turn-önként véletlenszerűen pumpákat ront el, round-onként pedig ciszternák szomszédos üres helyeire új csöveket helyez le). Turn-ök elején folyatja a vizet a forrásokból a szabad végű / lyukas csövekig vagy ciszternáig, illetve pontokat tud osztani a csapatoknak, amit szintén számontart.

- **Attribútumok**
  - **- currPlayerIdx: int** : A soron lévő játékos indexe a „players” gyűjteményben.
  - **- scoreTechnician: int** : A szerelők pontszámát tárolja.
  - **- scoreSaboteur: int** : A szabotőrök pontszámát tárolja.
  - **- input: Gameinput** : A felhasználóval kommunikáló objektum.(Híd a View és a Model között.)
  - **- players: Player[4..6]** : A játékban résztvevő játékosokat tárolja.
  - **- network: Network** : A pálya elemeit tárolja, szortírozza.

- **Metódusok**
  - **+ void AddTechnicianPoints(int points)** : Ez a metódus ad a szerelőknek pontot.
  - **+ void AddSaboteurPoints(int points)** : Ez a metódus ad a szabotőröknek pontot.
  - **+ void NextTurn()** : A következő turn indítása (frissíti/folyatja a vizet, pontokat oszt, véletlenszerűen pompákat ront el, és ha lement egy teljes kör (round), akkor új csöveket teremt a ciszternák üres szomszédjai helyén). Ha az egyik csapat elérte a győzelemhez szükséges pontok számát, véget vet a játéknak.
  - **+ Player GetCurrPlayer()** : Visszaadja a játékost, aki éppen soron van (akinek turn-je van jelenleg).
  - **+ boolean IsTechnicianTurn()** : Igazat ad vissza, ha éppen szerelő jön az adott körben.
  - **+ GameInput GetInput()** : Visszaadja a felhasználóval kommunikáló objektumot.
  - **+ Network GetNetwork** : Visszaadja a pálya elemeit számontartó objektumot.

3.3.5. GameInput

- **Felelősség**

A felhasználót és a játékot köti össze. Irányokat / célpontokat tud bekérni a felhasználótól. Játékosmozgatást, cső- és pumpalehelyezést, illetve csőfelvevést kezdeményezhet vele. Pumpát is tud adni a jelenlegi játékosnak, amennyiben egy szerelő, ciszernán áll és üres a tárolója.

- **Metódusok**
  - **+ void MoveCurrPlayer(Direction d)** : d irányba próbálja mozgatni az épp soron lévő játékost, arról az elemről, amelyiken éppen tartózkodik.
  - **+ void TryCurrElemManipulation()** : akkor hívandó függvény, amikor éppen azt az elemet szeretné manipulálni / interakcióba lépni vele (csövet lyukasztani / foltozni, pumpát javítani / átállítani vagy a következő ciszternára ugrani) a soron lévő játékos, amelyiken éppen áll.
  - **+ void TryPartRelocation(Direction d)** : akkor hívandú függvény, amikor a soron lévő játékos megpróbál egy tőle d irányba lévő szomszédos csövet felvenni, és a tárolójában eltárolni.
  - **+ void TryPartPlacement(Direction d)** : akkor hívandú függvény, amikor a soron lévő játékos megpróbálja a tárolt part-ját tőle d irányba lehelyezni.
  - **+ void Pickup()** : akkor hívandó függvény, amikor egy játékos egy part-ot próbálna a tárolójába tenni (nem a játéktérről, hanem közvetlenül odaadva). Ez csak akkor lesz sikeres, ha a játékos szerelő, üres a tárolója, és az elem amin áll, támogat ilyen funkcionalitást.
  - **+ Direction[2] GetNewPumpDirections()** : pumpa átállításához visszaadja a soron lévő játékos által bevitt irányokat.

3.3.6. Manipulator

- **Felelősség**

Absztrakt osztály, amely a Visitor tervezési mintát valósítja meg a leszármazottaival együtt.
A leszármazottai egy játékos típusnak készítik el a „gazda” elem manipulálását megvalósító viselkedést minden elem esetére (pumpák, csövek, források, ciszternák).
A szerelők konkrét manipulátora például definiálja, hogy mit tud tenni egy szerelő, ha az előbb említett elemeken állva próbál interaktálni. A függvényei gyakran a kört is léptetik.

- **Metódusok**
  - ***+ void Manipulate(Pipe p)*** *: Absztrakt metódus a játékosok cső manipulálásának leírására.*
  - **+ void Manipulate(Pump p)** : Átállítja az átadott pumpát (GameInput-ot használja a bemenetért). Ezután véget ér a játékos köre (turn).
  - **+ void Manipulate(Cistern c)** : Átlépteti a jelenlegi játékost a következő ciszternára. Ezzel nem ér véget a játékos köre (turn).
  - **+ void Manipulate(Source s)** : Üres törzsű függvény, jelenleg a játékosok nem tesznek semmit a forráson állva. (A jövőbeli bővíthetőségre fenntartva, illetve a Dynamic Dispatch hibátlan működése miatt kell, hogy ne kelljen Type-checkingelni a hívóoldalon.)

3.3.7. ManipulatorSaboteur

- **Felelősség**

A szabotőrök konkrét manipulátora, ami definiálja, hogy hogyan manipulálhatják az összes lehetséges „gazda” elemüket.
A csövek kilyukasztásának viselkedését valósítja meg. A többi viselkedés megfelel az ősbélivel.

- **Ősosztályok**

  Manipulator

- **Metódusok**
  - **+ void Manipulate(Pipe p)** : Kilyukasztja az átadott p csövet. Utána pedig véget ér a jelenlegi játékos köre.

3.3.8. ManipulatorTechnician

- **Felelősség**

A szerelők konkrét manipulátora, ami definiálja, hogy hogyan manipulálhatják az összes lehetséges „gazda” elemüket.
A csövek javításának viselkedését valósítja meg, illetve a pumpák megjavításának viselkedésével helyettesíti a pumpaátállítást, ha az adott pumpa rossz.
A többi viselkedés megfelel az ősbélivel.

- **Ősosztályok**

  Manipulator

- **Metódusok**
  - **+ void Manipulate(Pipe p)** : Megjavítja az átadott p csövet. Utána pedig véget ér a jelenlegi játékos köre.
  - **+ void Manipulate(Pump p)** : Megjavítja az átadott p pumpát, ha az elromlott, különben pedig átállítja (ez esetben meghívja az ősbéli megvalósítását a függvénynek).

3.3.9. Network

- **Felelősség**

Tárolja a pálya elemeit. Csövek és pumpák adhatók hozzá, a csöveket pedig el is lehet távolítani róla. Vizsgálja, hogy ezek a műveletek lehetségesek-e egyáltalán.
A benne tárolt elemek azok, amik ténylegesen meg is lesznek jelenítve a pályán.
Továbbá ez az osztály indítja el a vízfolyást forrásokból.

- **Attribútumok**
  - **- elements: Element[4..*]** : A pálya összes elemét tárolja.
  - **- cisterns: Cistern[1..*]** : Külön csoportosítja minden elem közül a ciszternákat ez a gyűjtemény.
  - **- sources: Source[1..*]** : Külön csoportosítja minden elem közül a forrásokat ez a gyűjtemény.
  - **- pumps: Pump[0..*]** : Külön csoportosítja minden elem közül a pumpákat ez a gyűjtemény.

- **Metódusok**
  - **+ boolean RemoveElem(Element e)** : Megkísérli eltávolítani az átadott elemet a pályáról. Ha ennek semmi akadálya nem volt (pl. nem esne komponensekre a pálya, mint gráf), akkor igazzal tér vissza. Ellenkező esetben nem módosít semmi a pályán.
  - **+ boolean AddPipe(Pipe p)** : Az átadott csövet megkísérli hozzáadni a pályához. Ennek sikerességét adja vissza.
  - **+ boolean AddPump(Pump p, Element e)** : Az átadott pumpával kísérli meg felülírni a másik átadott elemet a pályán. Ennek sikerességét adja vissza.
  - **+ void Flood()** : Felapasztja az összes vizet a pályáról, majd minden forrásból elindítja a vizet, aminek következtében el lesz árasztva vízzel a pálya, és pontokat fognak kapni a csapatok.
  - **+ Element[4..*] GetElements()** : Visszaadja a pálya minden elemét.
  - **+ Pump[0..*] GetPumps()** : Visszaadja a pályán lévő pumpákat.
  - **+ Cistern[1..*] GetCisterns()** : Visszaadja a pályán lévő ciszternákat.

3.3.10. Pipe

- **Felelősség**

Olyan elromolható, felvehető elem, amelyen legfeljebb egy játékos tartózkodhat.
Ki lehet lyukasztani, a lyukas csövet pedig meg is lehet javítani. Ha a cső nem lyukas akkor a kimenetéhez vizet tud juttatni.
Ha lyukas, vagy a kimenete üres, akkor a szabotőrök pontot kapnak, amikor víz érkezik belé.

- **Ősosztályok**

  Element → Breakable

- **Metódusok**
  - **+ void AddPlayer(Player p)** : Felhelyezi az átadott játékost magára, ha nem áll rajta már más valaki.
  - **+ void Manipulate(Manipulator m)** : az átvett manipulátorral manipulálja ezt a konkrét elemet.
  - **+ void Flood()** : Vízzel tölti meg magát. Ha lyukas a cső vagy nincs output-ja akkor pontot kapnak a szabotőrök, különben pedig hívja tovább az output-ján a Flood() függvényt (ereszti tovább a vizet).

3.3.11. Player (/ Saboteur)

- **Felelősség**

A Player osztály a szabotőr osztállyal ekvivalens, tehát a szabotőrök csapatának játékosai Player típusúak. Eltárolja a játékos nevét, az elemet, amelyen éppen áll, és a manipulátorát. Tud mozogni irányokba, és manipulálni tudja a „gazda” elemét.

- **Attribútumok**
  - **- currElem: Element** : Az adott elem, amelyen a játékos tartózkodik
  - **- name: string** : A játékos neve, azonosítója
  - **- manipulator: Manipulator** : A játékos által használt manipulator objektum a „gazda” elemmel való interkaciók lekezeléséhez.

- **Metódusok**
  - + **void MoveTo(Element e)** : Az átadott elemre helyezi a játékost, ha ez lehetséges.
  - **+ void StepOnto(Direction d)** : A játékost a megadott irányban lévő elemre lépteti, ha ez lehetséges.
  - **+ void ManipulateCurrElem()** : Manipulálja azt az elemet, amelyen éppen tartózkodik. A manipulátora határozza meg, hogy mely „gazda” elem típus esetén mit tesz.

3.3.12. Pump

- **Felelősség**

Olyan elromolható, felvehető elem, amely vizet képes áteresztení a bemenetként beállított csövéből a kimenetként beállított csövébe. Pontot oszt a szabotőröknek, ha nincsen kimenete, amikor vizet eresztene át.

- **Ősosztályok**

  Element → Breakable

- **Metódusok**
  - **+ void Flood()** : Ha a pumpa el van romolva, nem tesz semmit. Ha nincs elromolva akkor vízzel tölti fel magát. Ha nincsen outputja, akkor a szabotőrök pontot kapnak, de ha van outputja, akkor az outputra hívja tovább a Flood() függvényt.
  - **+ void Manipulate(Manipulate m)** : az átvett manipulátorral manipulálja ezt a konkrét elemet.

3.3.13. Source

- **Felelősség**

Olyan elem, amely vizet juttat a vele szomszédos elemekbe, és soha sincsen bemenete.

- **Ősosztályok**

  Element

- **Metódusok**
  - **+ void Flood()** : Vízzel tölti fel magát, és minden nem lyukas szomszédjára hívja tovább a Flood() függvényt, azaz ereszti tovább a vizet. Ha nem tudja ezt megtenni az adott szomszédja felé, mert lyukas, vagy nincs arra szomszédja, akkor pontot ad a szabotőröknek.
  - **+ void Manipulate(Manipulator m)** : az átvett manipulátorral manipulálja ezt a konkrét elemet.

3.3.14. Technician

- **Felelősség**

Olyan játékos, aki csövek lyukasztása helyett javítani tudja őket, illetve pumpákat is. Emellett képes vagy egy csövet, vagy egy pumpát tárolni magánál, ami el is tud helyezni a megfelelő feltételek fennállása esetén.

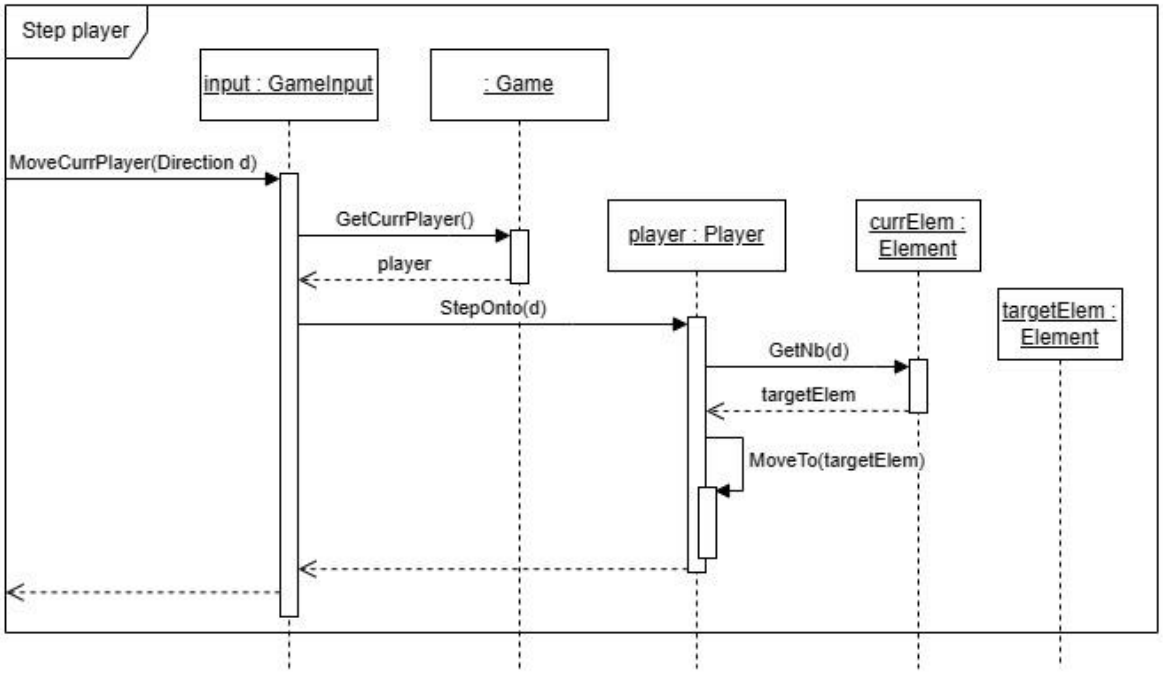
- **Ősosztályok**

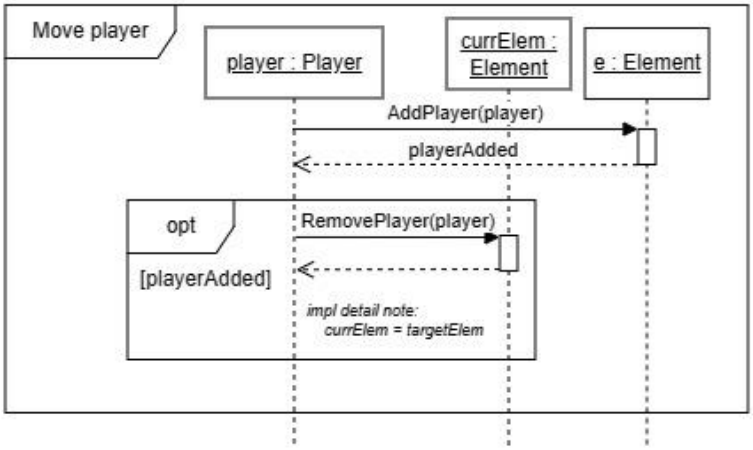
  Player

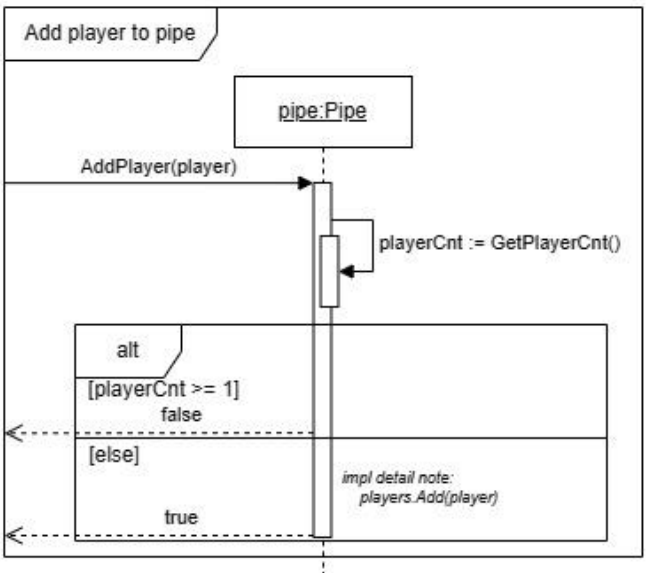
- **Attribútumok**
  - **- part: Breakable[0..1]** : Az az elem, amelyet a szerelő felvett a pályáról.

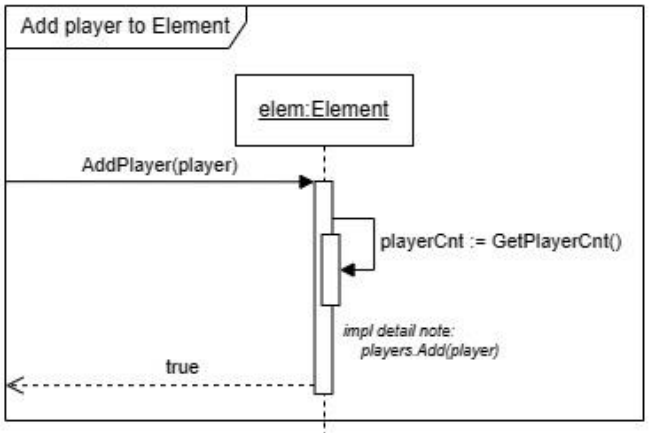
- **Metódusok**
  - **+ void PickUpPart() :** Megkísérel felvenni egy part-ot a jelenlegi elemtől.
  - **+ void PickUpPart(Direction d) :** Megkísérli felvenni a d irányban lévő part-ot.
  - **+ boolean PlacePart(Direction d)** : Megkísérli elhelyezni a tárolt part-ját d irányba. A művelet sikerességével tér vissza.
  - **+ Breakable GetPart()** : Visszaadja azt a Breakable-t („part” attribútumot), ami a szerelőnél van.
  - **+ void SetPart(Breakable b)** : Az átadott Breakable-re állítja a „part” attribútumot.

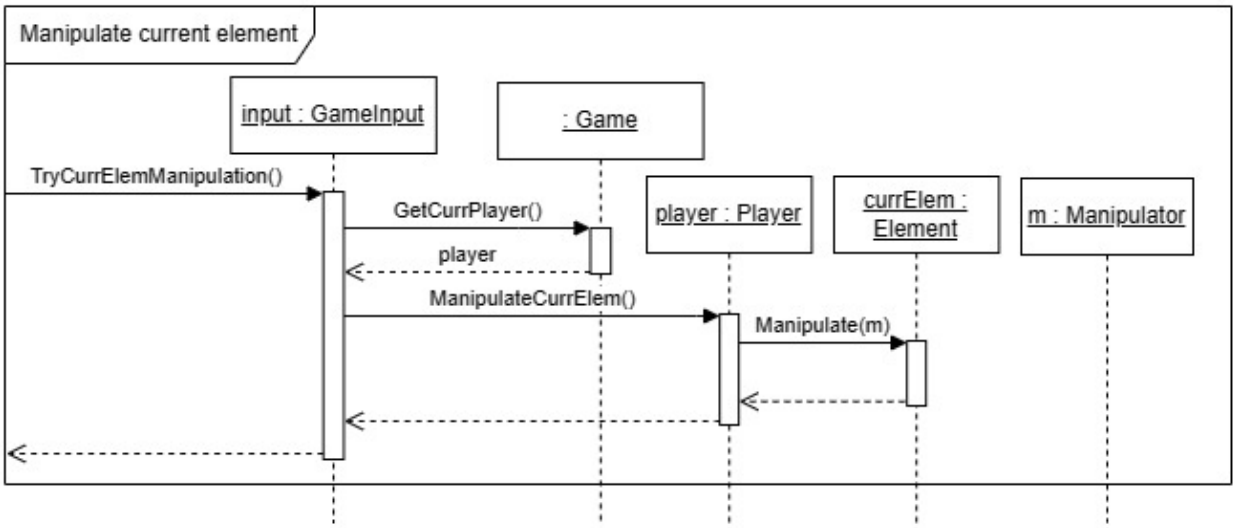
3.4. Szekvencia diagramok

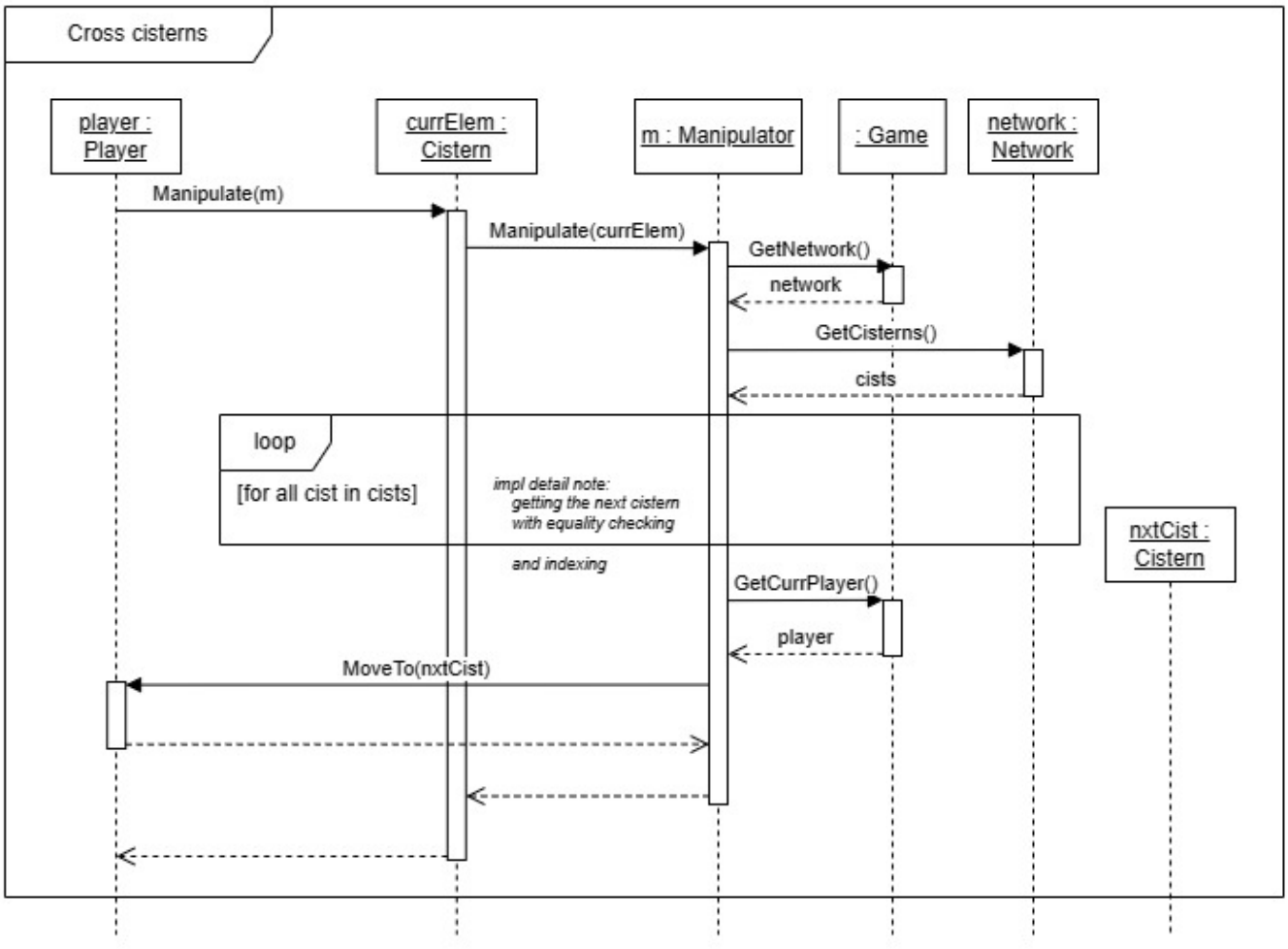


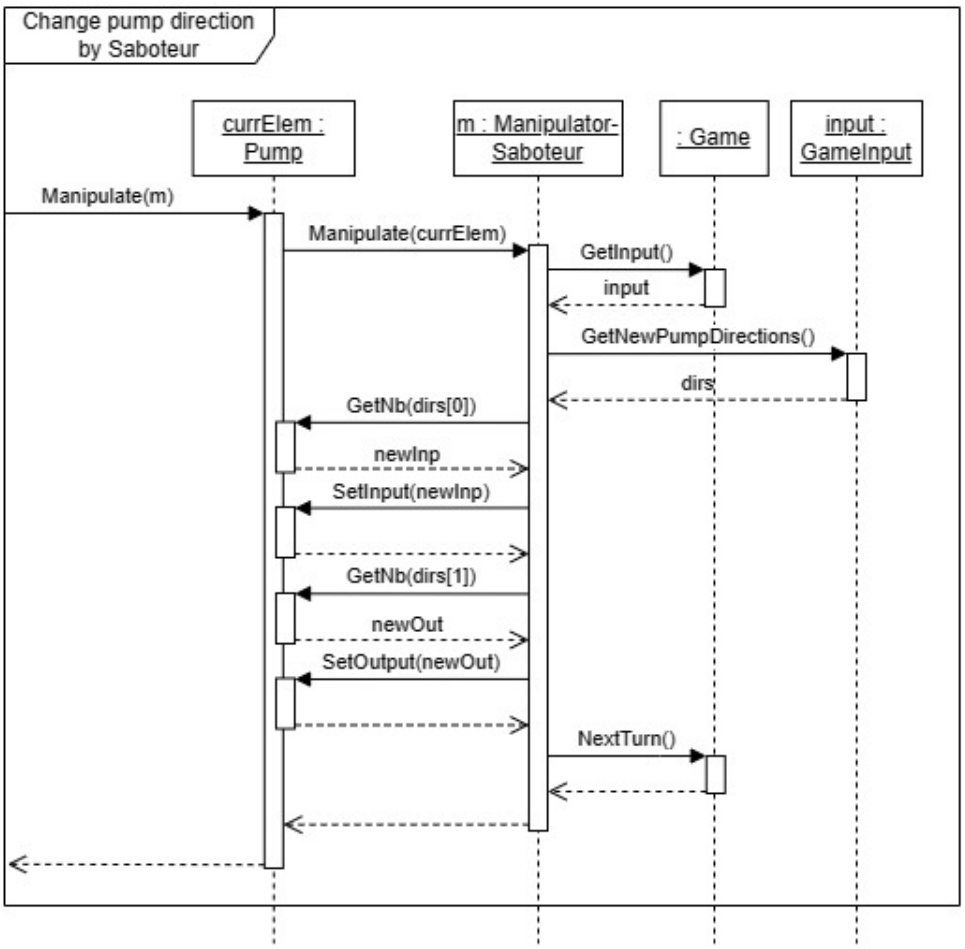


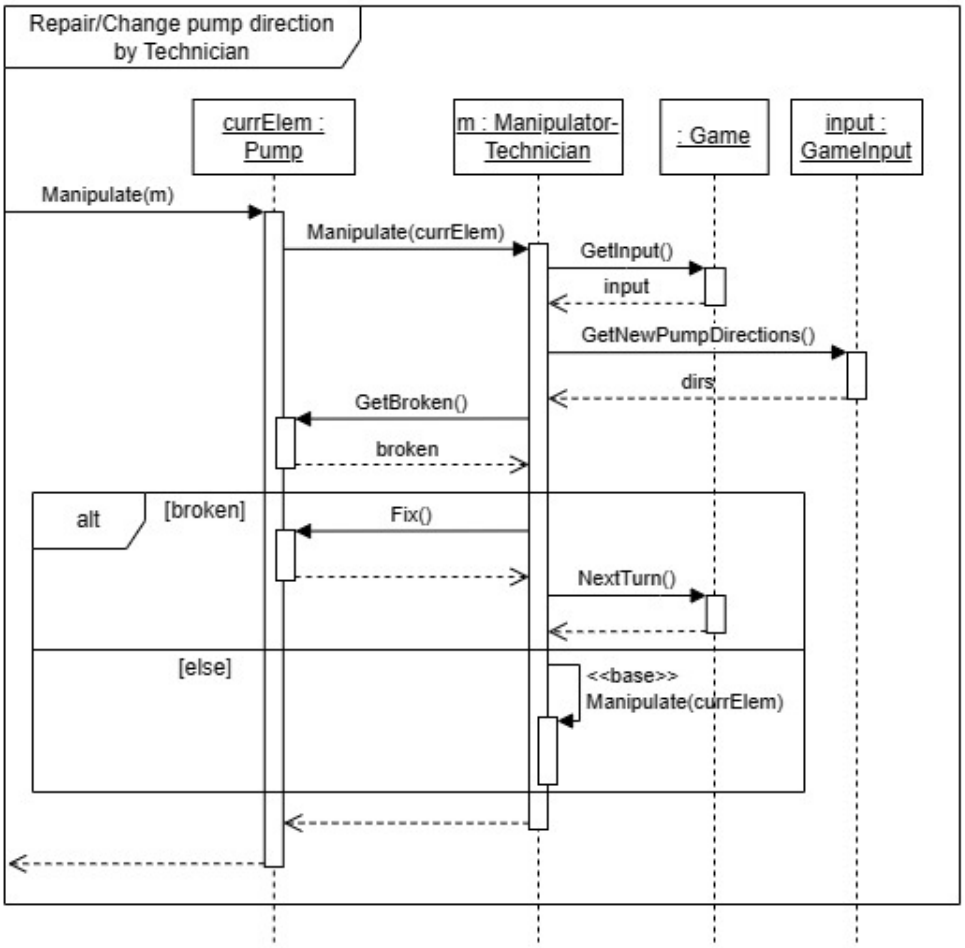


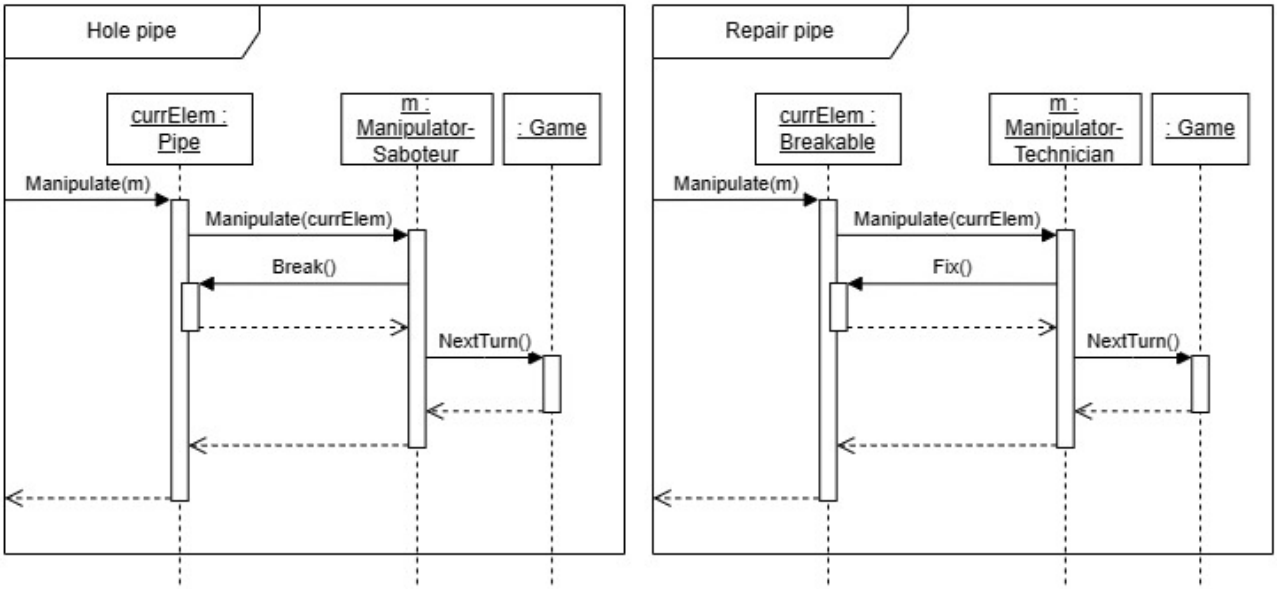


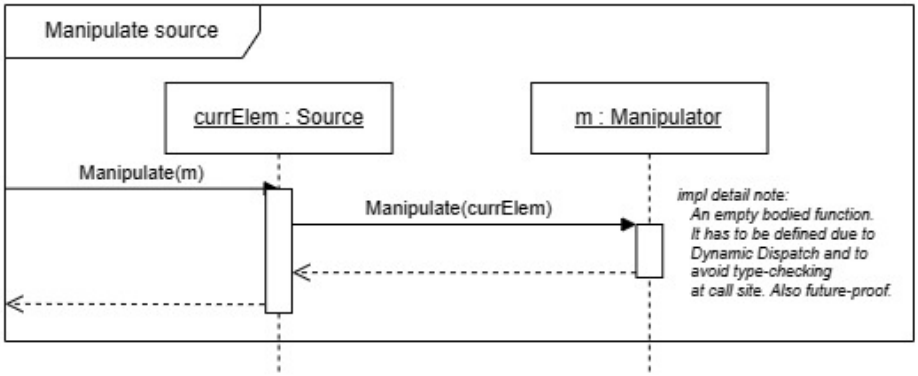

Manipulate current element
input : GameInput
: Game
player : Player
currElem : Element
m : Manipulator
TryCurrElemManipulation()
GetCurrPlayer()
player
ManipulateCurrElem()
Manipulate(m)

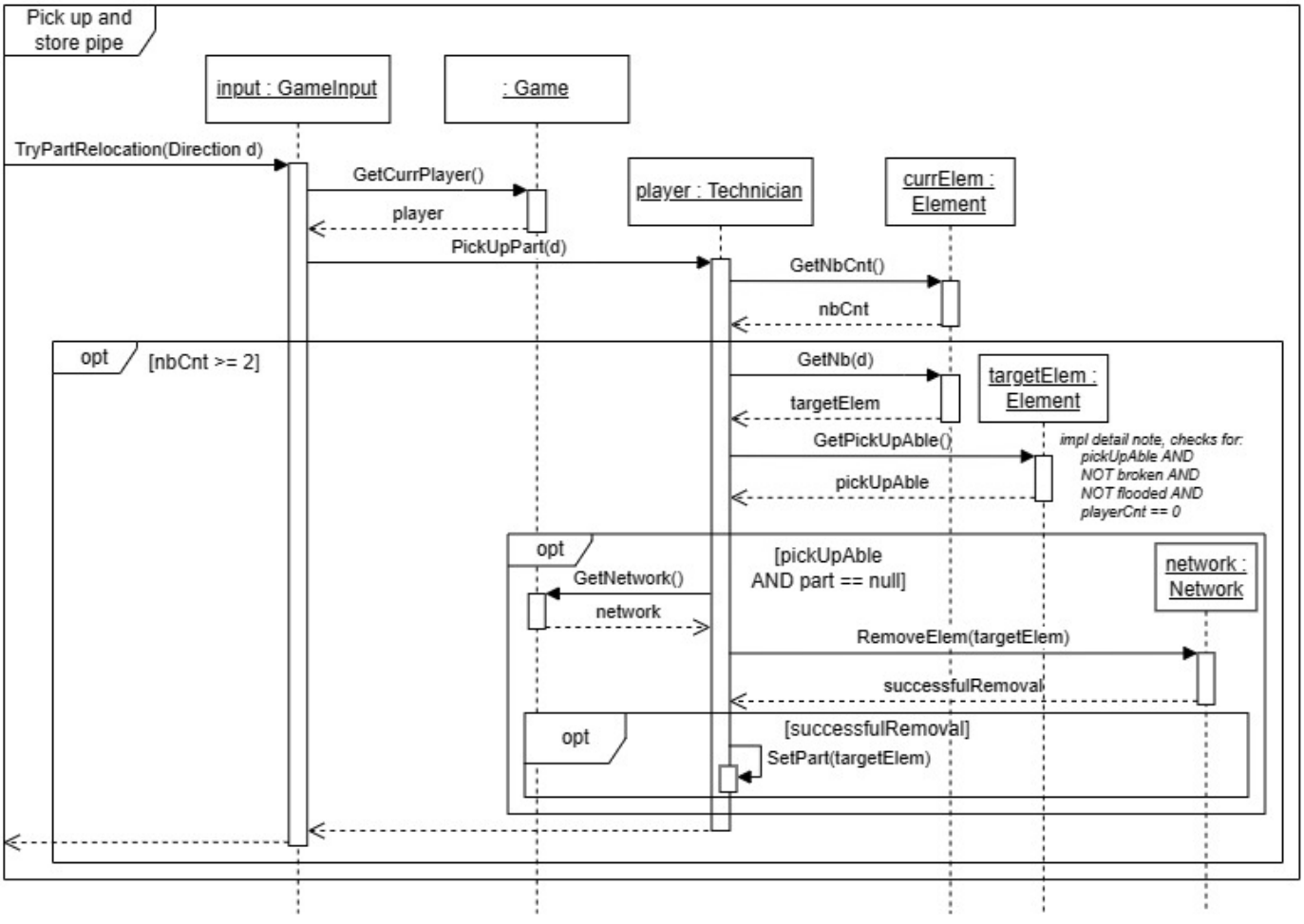

Cross cisterns
player : Player
currElem : Cistern
m : Manipulator
: Game
network : Network
Manipulate(m)
Manipulate(currElem)
GetNetwork()
network
GetCisterns()
cists
loop
[for all cist in cists]
impl detail note:
getting the next cistern
with equality checking
and indexing
nxtCist : Cistern
GetCurrPlayer()
player
MoveTo(nxtCist)

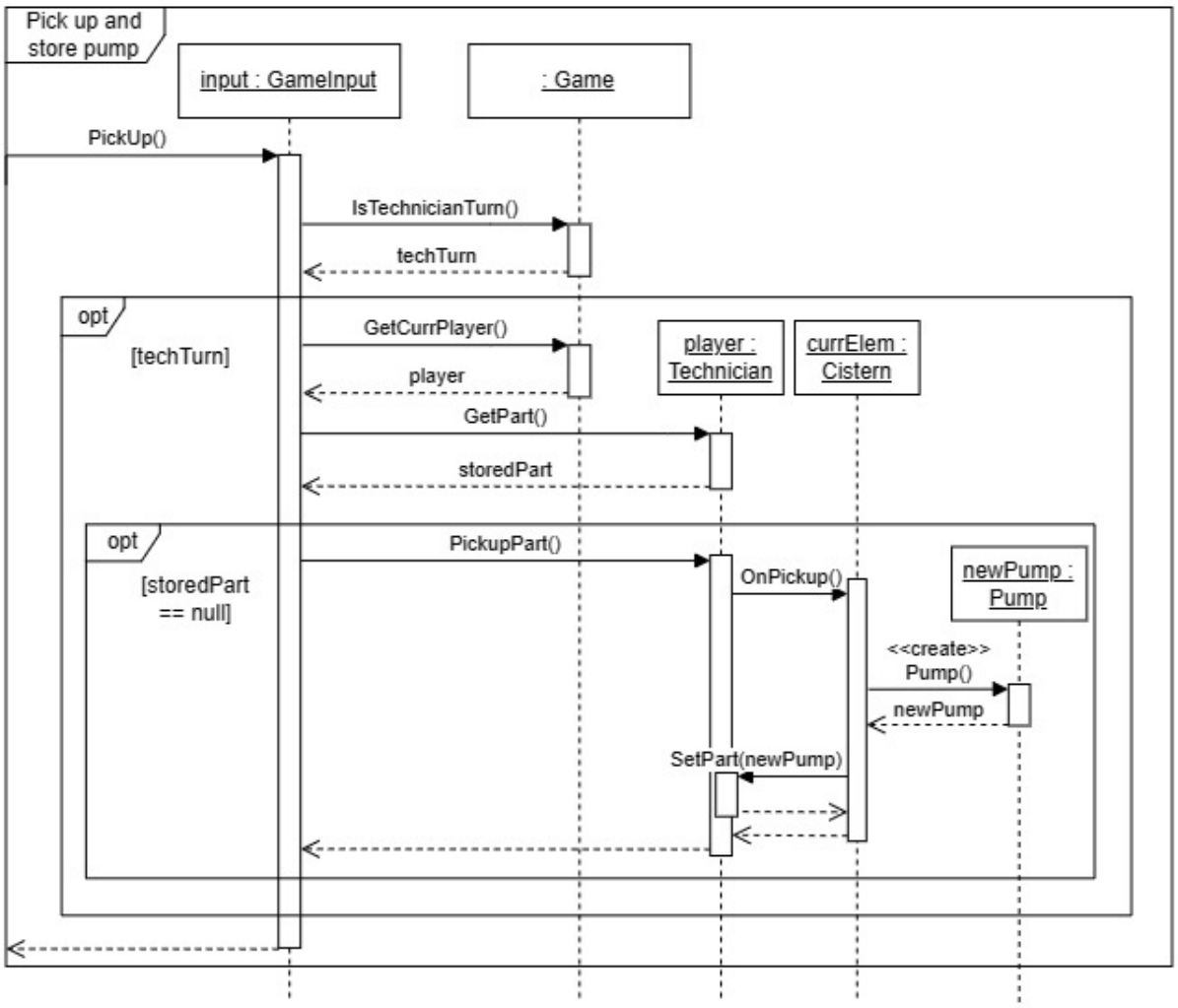

Change pump direction by Saboteur
currElem : Pump
m : Manipulator-Saboteur
: Game
input : GameInput
Manipulate(m)
Manipulate(currElem)
GetInput()
input
GetNewPumpDirections()
dirs
GetNb(dirs[0])
newInp
SetInput(newInp)
GetNb(dirs[1])
newOut
SetOutput(newOut)
NextTurn()

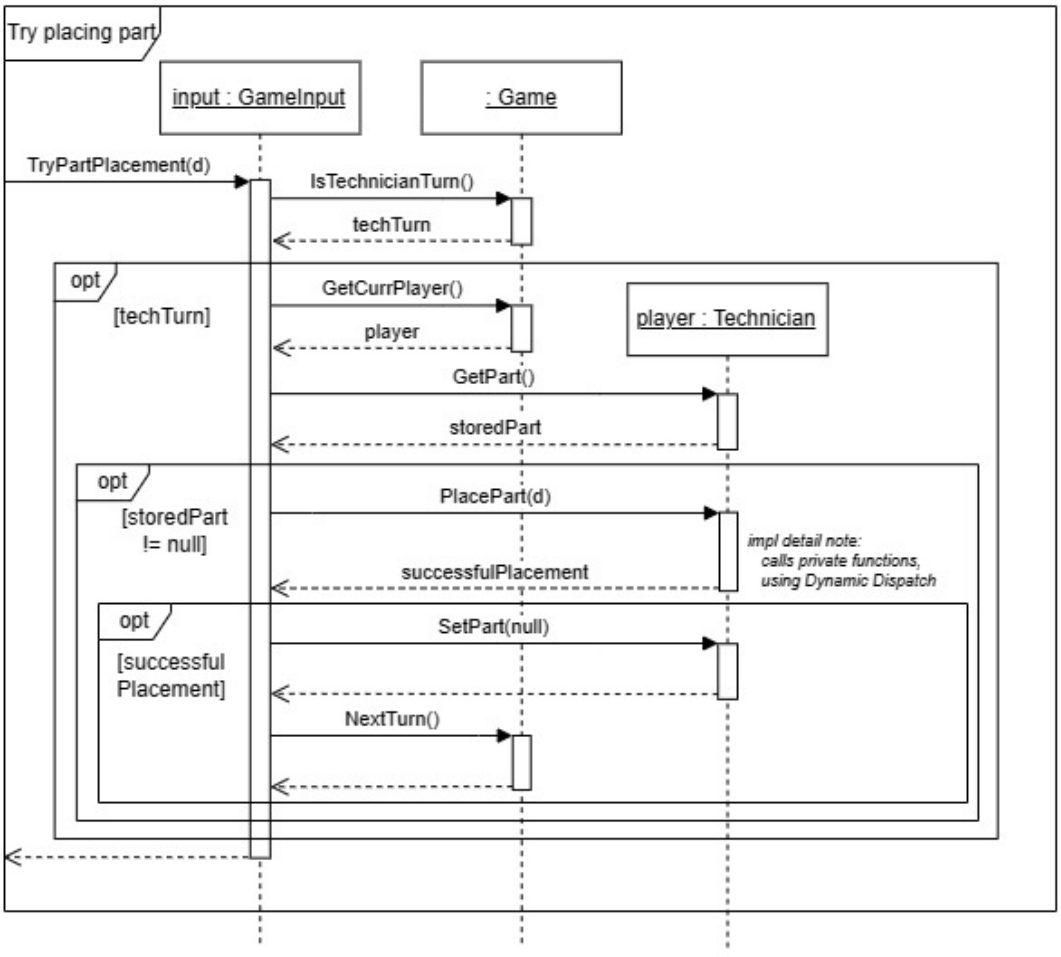

Repair/Change pump direction by Technician
currElem : Pump
m : Manipulator-Technician
: Game
input : GameInput
Manipulate(m)
Manipulate(currElem)
GetInput()
input
GetNewPumpDirections()
dirs
GetBroken()
broken
alt
[broken]
Fix()
NextTurn()
[else]
<<base>>
Manipulate(currElem)

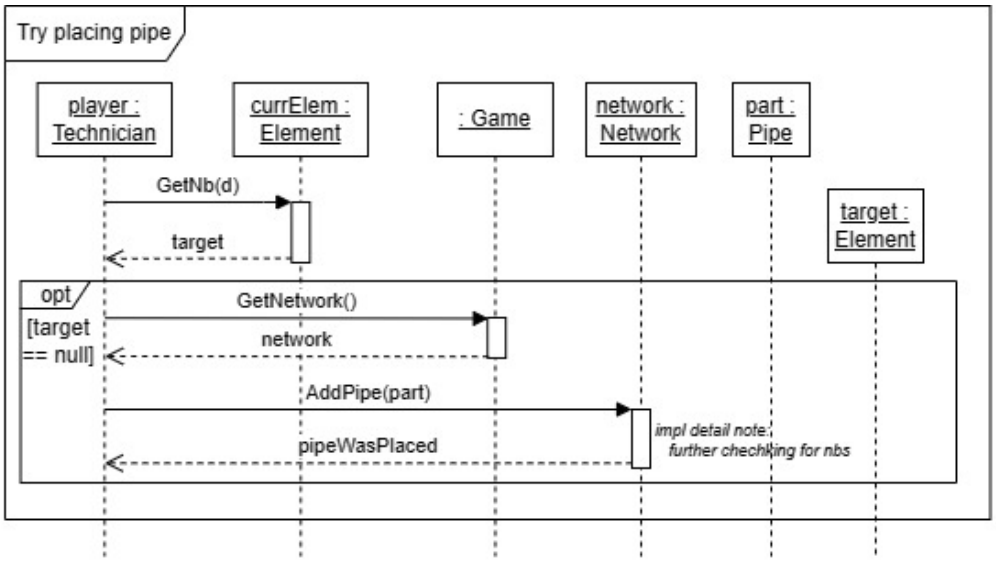

Hole pipe
currElem : Pipe
m : Manipulator-Saboteur
: Game
Manipulate(m)
Manipulate(currElem)
Break()
NextTurn()
Repair pipe
currElem : Breakable
m : Manipulator-Technician
: Game
Manipulate(m)
Manipulate(currElem)
Fix()
NextTurn()

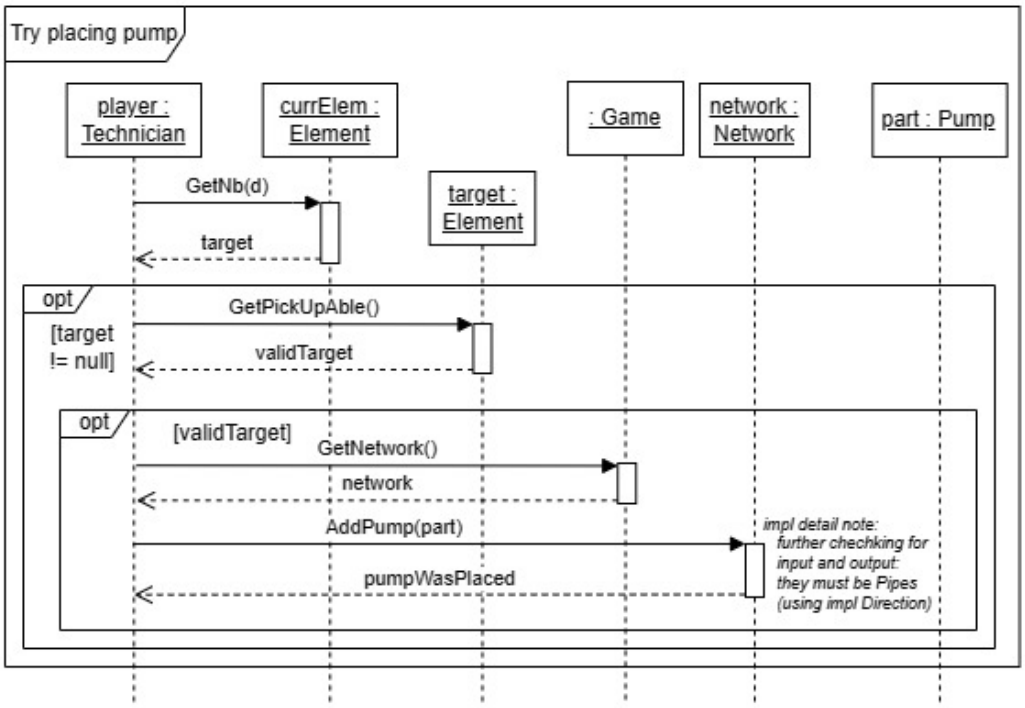

Manipulate source
currElem : Source
m : Manipulator
Manipulate(m)
Manipulate(currElem)
impl detail note:
An empty bodied function.
It has to be defined due to
Dynamic Dispatch and to
avoid type-checking
at call site. Also future-proof.

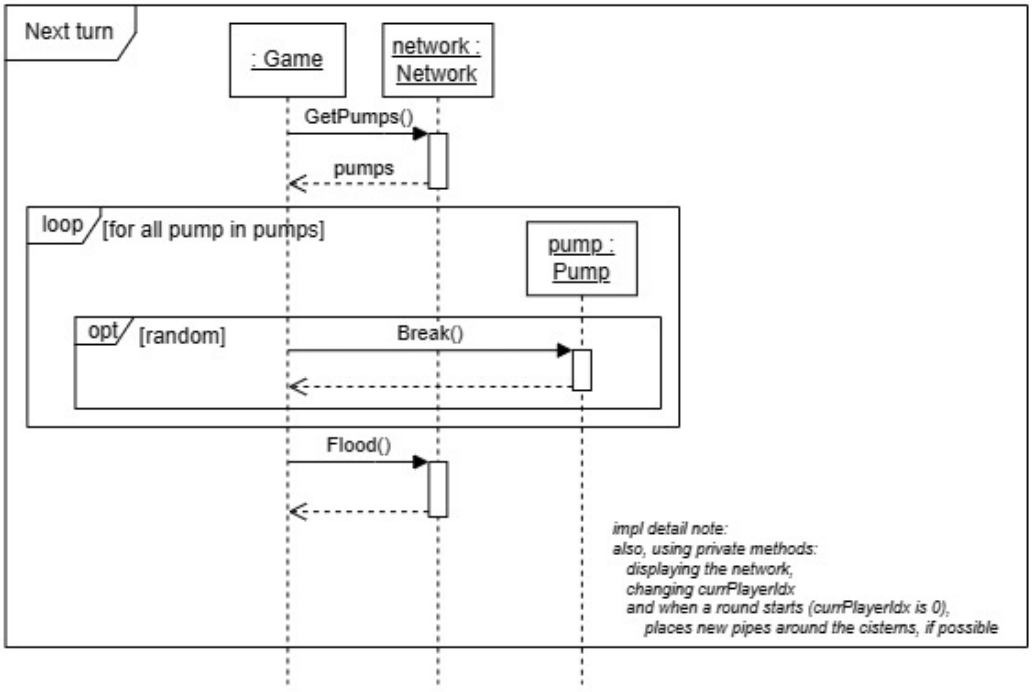

Pick up and store pipe
input : GameInput
: Game
player : Technician
currElem : Element
TryPartRelocation(Direction d)
GetCurrPlayer()
player
PickUpPart(d)
GetNbCnt()
nbCnt
opt
[nbCnt >= 2]
GetNb(d)
targetElem
targetElem : Element
GetPickUpAble()
pickUpAble
impl detail note, checks for:
pickUpAble AND
NOT broken AND
NOT flooded AND
playerCnt == 0
opt
[pickUpAble
AND part == null]
GetNetwork()
network
network : Network
RemoveElem(targetElem)
successfulRemoval
opt
[successfulRemoval]
SetPart(targetElem)

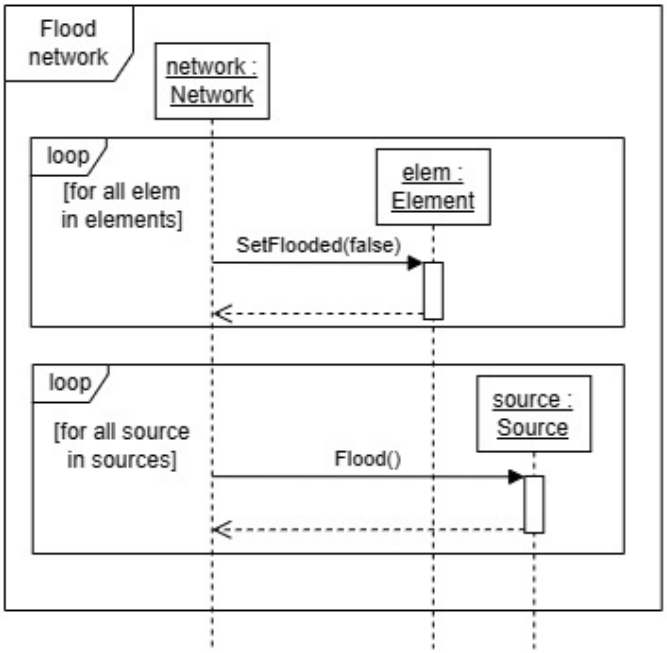

Pick up and store pump
input : GameInput
: Game
PickUp()
IsTechnicianTurn()
techTurn
opt
[techTurn]
GetCurrPlayer()
player
player : Technician
currElem : Cistern
GetPart()
storedPart
opt
[storedPart == null]
PickupPart()
OnPickup()
newPump : Pump
<<create>>
Pump()
newPump
SetPart(newPump)

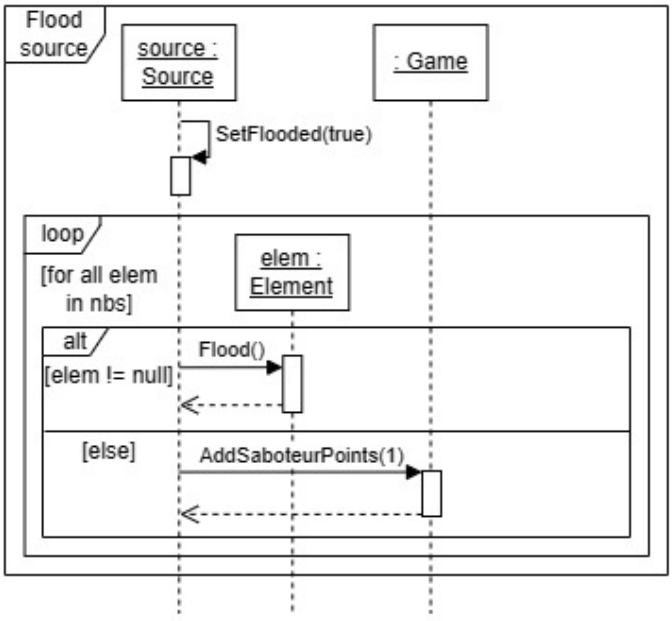

Try placing part
input : GameInput
: Game
TryPartPlacement(d)
IsTechnicianTurn()
techTurn
opt
[techTurn]
GetCurrPlayer()
player
player : Technician
GetPart()
storedPart
opt
[storedPart != null]
PlacePart(d)
impl detail note:
calls private functions,
using Dynamic Dispatch
successfulPlacement
opt
[successful Placement]
SetPart(null)
NextTurn()


Try placing pipe
player : Technician
currElem : Element
: Game
network : Network
part : Pipe
target : Element
GetNb(d)
target
opt
[target == null]
GetNetwork()
network
AddPipe(part)
pipeWasPlaced
impl detail note:
further chechking for nbs

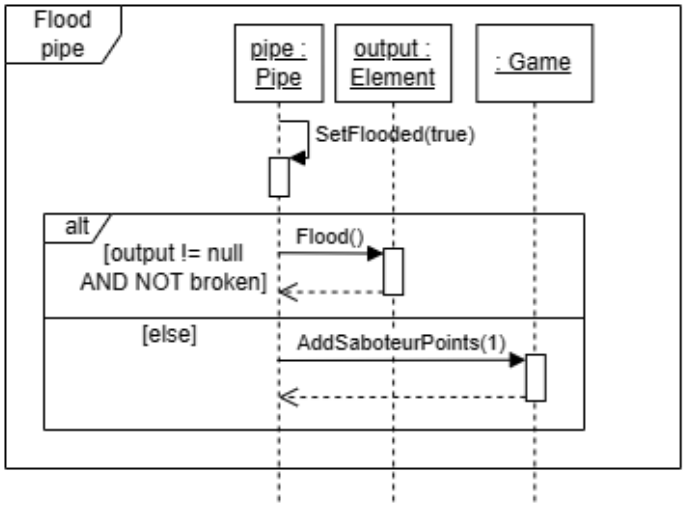

Try placing pump
player : Technician
currElem : Element
: Game
network : Network
part : Pump
target : Element
GetNb(d)
target
opt
[target != null]
GetPickUpAble()
validTarget
opt
[validTarget]
GetNetwork()
network
AddPump(part)
pumpWasPlaced
impl detail note:
further chechking for input and output:
they must be Pipes
(using impl Direction)

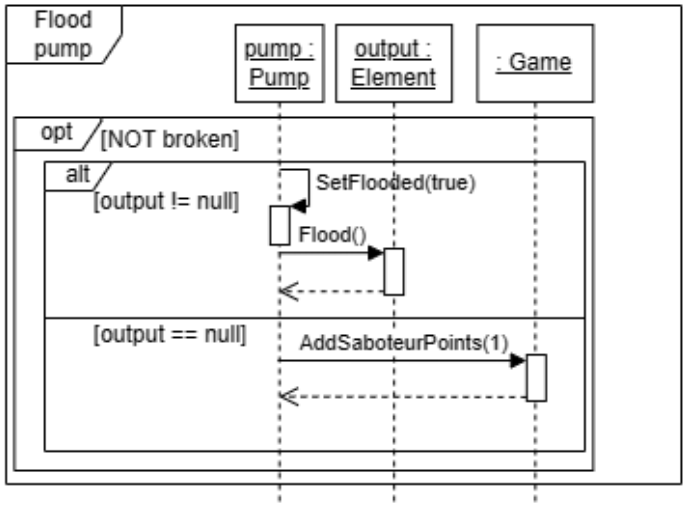

Next turn
: Game
network : Network
GetPumps()
pumps
loop
[for all pump in pumps]
pump : Pump
opt
[random]
Break()
Flood()
impl detail note:
also, using private methods:
displaying the network,
changing currPlayerIdx
and when a round starts (currPlayerIdx is 0),
places new pipes around the cisterns, if possible

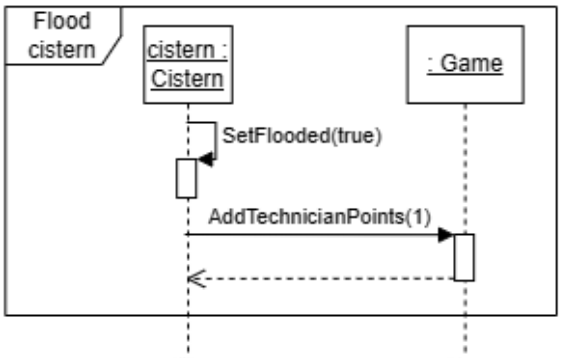




impl detail note:
"nbs" and "output" are available through Getters

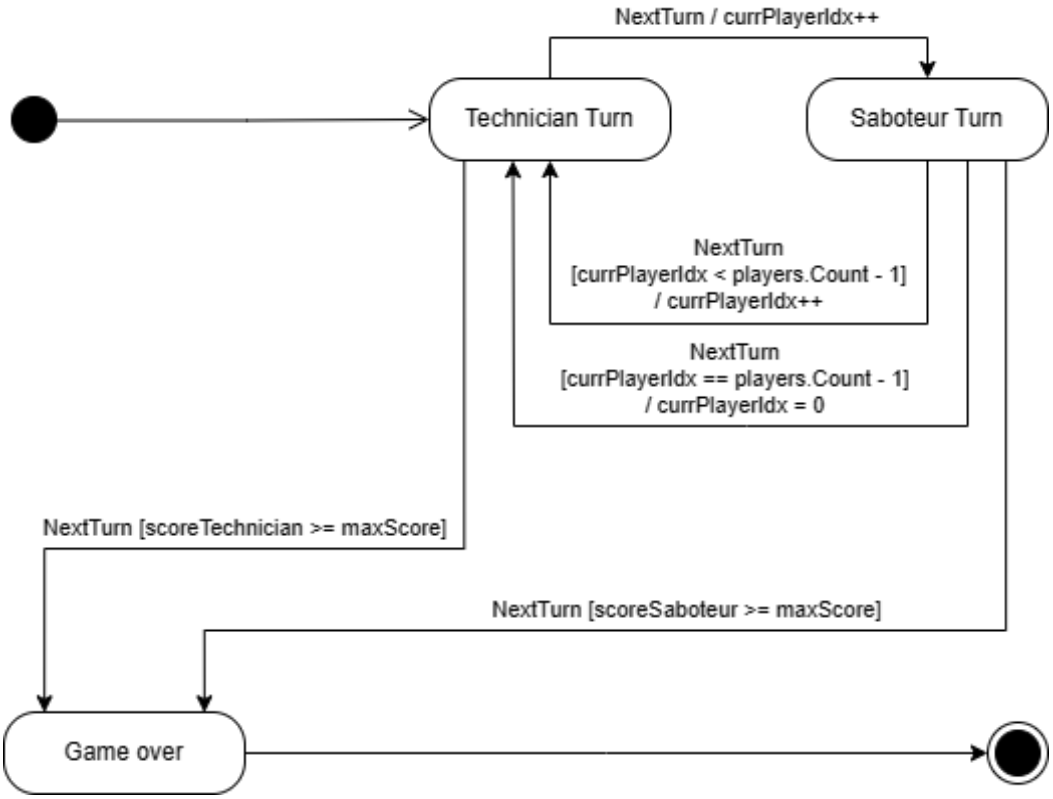






3.5. State-chartok

3.5.1 Játékosok köreinek (turn) váltakozása



3.6. Napló

Kezdet	**Időtartam**	**Résztvevők**	**Leírás**
2023.03.15. 16:30	0,5 óra	Mizser Váradi Tepliczky Fekete Sasvári	Értekezlet. Döntés: Az eheti feladatokat kollaboratív módon, specifikus felosztás nélkül végezzük. ---------- **Objektum katalógus** Határidő: 2023.03.16. 8:00. Elkészült: 2023.03.15. 19:00 **Osztálydiagram, Osztályok leírása** Határidő: 2023.03.20. 6:00 Elkészült: 2023.03.20. 0:00 **Szekvencia diagramok** Határidő: 2023.03.20. 11:00 Elkészült: 2023.03.20. 1:00
2023.03.15. 17:00	2 óra	Mizser Váradi Tepliczky Fekete Sasvári	Objektum katalógus meghatározása
2023.03.16 20:00	2,5 óra	Mizser Tepliczky Fekete	Osztálydiagram vázlata
2023.03.16 20:30	2 óra	Sasvári	Osztálydiagram vázlata
2023.03.17. 6:00	1,5 óra	Váradi	Osztálydiagram szerkesztése
2023.03.17 22:00	1 óra	Fekete	Osztályok leírásának szerkesztése
2023.03.18 14:00	1 óra	Mizser Tepliczky Fekete Váradi	Osztálydiagram szerkesztése
2023.03.18 15:00	2 óra	Tepliczky Fekete	Objektum katalógus szerkesztése
2023.03.18 15:00	2 óra	Mizser	Osztálydiagram további részének megalapozása
2023.03.18 17:00	1 óra	Mizser Tepliczky Fekete	Osztálydiagram bővítése
2020.03.18. 21:00	2 óra	Mizser	Visitor minta megtervezése a Manipulator osztályhoz
2023.03.19. 11:30	1,5 óra	Sasvári	Objektum katalógus, Osztálydiagram Review

2023.03.19. 14:00	1 óra	Sasvári	Osztálydiagram bővítése
2023.03.19. 14:00	2 óra	Mizser	Osztálydiagram bővítése
2023.03.19. 15:00	4 óra	Váradi	Osztálydiagram bővítése
2023.03.19. 15:30	5 óra	Fekete	Osztályok leírásának bővítése
2023.03.19. 15:45	4,75 óra	Tepliczky	Osztályok leírásának bővítése
2023.03.19. 16:00	4,5 óra	Sasvári	State-chartok, Szekvencia diagramok
2023.03.19. 21:00	4 óra	Mizser Fekete	Szekvencia diagramok befejezése, osztálydiagram- és -leíársok módosítása

M: 17
V: 13.5
T: 13.75
F: 19
S: 13