8. Részletes tervek

44 – the_grinders

Konzulens:
Ludmány Balázs

Csapattagok

Andrási Gellért Péter	FOSASY	andrasi.gellert@edu.bme.hu	(kapcsolattartó)
Nás Levente Gellért	GWPESM	nas.levente@edu.bme.hu	
Mayer Ádám	XYJP9S	mayeradam@edu.bme.hu	
Farkas Fanni	EU7XYP	farkasfanni@edu.bme.hu	
Bertók Dániel	H01HRM	bertokd@edu.bme.hu	

2023.05.03.

8. Részletes tervek

8.1 Osztályok és metódusok tervei.

8.1.1 Desert

- **Felelősség**

Ez az osztály reprezentálja a parancsokat. A Proto osztály belső osztálya.

- **Ősosztályok**

-

- **Interfészek**

-

- **Attribútumok**
  - **+String name:** a parancs nevét reprezentálja
  - +**ArrayList<Object> options:** a parancs lehetséges paramétereit tárolja
  - **+FunctionReference function:** a parancsot demonstráló függvény

- **Metódusok**

-

8.1.2 Desert

- **Felelősség**

A sivatagot reprezentálja, ahova elfolyik a víz az üres végű csövekből.

- **Ősosztályok**

PipelineElement

- **Interfészek**

-

- **Attribútumok**

-

- **Metódusok**
  - **+int seepWater(PipelineElement from, int qty):** Visszatér azzal az értékkel, amennyit elfogadott a szivárgásból, és csökkenti a Game objektum finishedWater tagváltozójának az értékét, vagyis a szerelők pontszámát. Mivel nincs kapacitása, ezért nincs megkötése annak, hogy mennyi vizet tud elfogadni, tehát mindig a kapott qty paraméterrel tér vissza.
  - **+bool accept(Person p):** A PipelineElement-től örökölt metódus, mindig false-al tér vissza, mivel nem lehet a sivatagra lépni.

8.1.3 FileHandler

- **Felelősség**

A fájlkezelés a feladata, a file attribútumában tároltfáljba menti/beolvassa az adatokat.

- **Ősosztályok**

  -

- **Interfészek**

  -

- **Attribútumok**
  - **-File file:** azt a file-t reprezentálja, amiből szeretnénk beolvasni/kiírni az adatokat
  - **-String[] data;**

- **Metódusok**
  - **+bool readFile():** beolvassa a file attribútum tartalmát soronként, eltárolja egy Stringben, majd minden Stringet hozzáad a data tömbhöz. Visszatér, hogy sikeres volt-e a művelet.
  - **+bool writeFile():** kiirja a file attribútumba soronként a data tömb Stringjeit. Visszatér, hogy sikeres volt-e a művelet.

8.1.4 FunctionReference

- **Felelősség**

Interfész, a command függvények hivatkozásához szükséges.

- **Ősosztályok**

-

- **Interfészek**

-

- **Attribútumok**

-

- **Metódusok**
  - **+void runFunction(ArrayList<Object> options):** futtatja az adott példányhoz tartozó függvényt

8.1.5 Game

- **Felelősség**

A játék irányításáért felelős osztály. Ebbe beleértjük a játék indítását és az inicializálását, illetve a pontszámok nyilvántartását. Csak egyszer példányosítható, hiszen egyszerre csak egy játékot lehet játszani. Az összes osztály ismeri.

- **Ősosztályok**

-

- **Interfészek**

-

- **Attribútumok**
  - **-Desert desert:** Az adott játékhoz tartozó, hálózat alatt elhelyezkedő sivatag.
  - **-Timer timer**: Az időzítő, amely ennek a játénak az ütemezését végzi el.
  - **-int finishedWater**: A ciszternákba eljutott víz mennyisége.
  - **-int lostWater**: A lyukas csöveken kifolyó víz mennyisége.
  - **-Person[0..*] players:** A játékot játszó játékosok halmaza.

- **Metódusok**
  - **+void startGame():** Ez a metódus indítja el a játékot. Létrehoz egy új példányt a Timer osztályból. Ezt átadja a saját timer attribútumához tartozó setter függvénynek. Az újonnan létrehozott csőhálózatot inicializálja, vagyis meghívja saját magán az initNetwork() metódust.
  - **+void addLostWater(int qty):** A paraméterként kapott mennyiséget hozzáadja a lostWater attribútumhoz.
  - **+void addFinishedWater(int qty):** A paraméterként kapott értéket hozzáadja a finishedWater tagváltozóhoz.
  - **+void removeFinishedWater(int qty):** A paraméterként kapott értéket levonja a finishedWater tagváltozóból. 0 alá nem csökkentheti.
  - **+void initNetwork():** A vízvezetékhálózat létrehozásáért és inicializálásáért felelős. Minden játékhoz szükséges objektumból annyit hoz létre, amennyi meg van határozva. Ehhez ciklusokat használ. Először létrehoz egy Pump objektumot, majd hozzáadja az eltárolt Timer tickables tömbjéhez az addTickables metódussal. A forrásokat már ciklusban hozza létre, itt is megtörténik az addTickables hívása. Minden létrehozott WaterSource objektumhoz létrehoz egy új csövet is, amelynek a két szomszédja a metódus elején létrehozott pumpa, illetve a ciklus ennek az iterációjában létrehozott forrás. Ennek a beállításához mind a három objektumon meg kell hívni az előbb leírtaknak megfelelő paraméterezéssel az addNeighbor metódust, így hozzáadva a szomszédnak szánt elemet a neihbors tömbhöz. WaterTank, vagyis ciszterna esetében teljesen megegyező módon jár el. A játék kezdetekor minden játékos az egyetlen létrehozott pumpán kezd. Egy ciklusban létrehozza a szerelőket és hozzáadja a players tömbhöz az adott iterációban létrejött szerelőt a Game osztály addPlayer függvényével, illetve a timer tickables tömbjéhez az addTickables segítségével. Mivel a pumpán kezdenek, ezért meghívja a Plumber objektumon a setElement(pu) metódust, ahol a pu a függvény elején létrejött pumpa. Ezt követően az accept(Person p) metódusát is meg kell hívni a pu objektumnak, amelynek a visszatérési értéke nem érdekes, hiszen pumpára mindig léphet játékos. Erre azért van szükség, hogy a pumpa is tudja, hogy rajta van a játékos. Szabotőr esetén teljesen megegyező módon járunk el, csupán azért van szükség külön ciklusra, mivel nincs kikötve, hogy ugyanannyi szerelő és szabotőr játssza a játékot. Legvégül pedig létrehoz egyetlen Desert-et, és átadja ezt paraméterként a setDesert(Desert d) függvénynek.
  - **+rotatePlayers():** Eltávolítja a players tömb első elemét, majd hozzáadja a végére.

8.1.6 Person

- **Felelősség**

A játékos szabotőr vagy vízvezetékszerelő mivoltától független funkcióit foglalja magába, ennek a két osztálynak a közös őse, absztrakt osztály. Felelős a játékos mozgatásáért és a pumpa elállításáért. A Tickable interfészt implementálja, ezért a tick metódust is megvalósítja.

- **Ősosztályok**

-

- **Interfészek**

Tickable

- **Attribútumok**
  - **#PipelineElement element:** A csőhálózatnak azon eleme, amelyiken a játékos éppen áll. Ennek a megfelelő metódusa kerül meghívásra, ha a játékos valamilyen műveletet hajt végre az elemen.

- **Metódusok**
  - **+void move(int index)**: Paraméterként megkapja az aktuális elem azon szomszédjának az indexét, ahányadik eleme a neighbors tömbnek a szomszéd amelyre lépni akar. Először lekérdezi az element tagváltozó index paraméterrel megegyező indexű szomszédját a getNeighborElement(index) függvényhívással. Erre az elemre szeretne lépni a játékos. A függvény törzs maradék része egy if függvénybe van zárva, amelynek a logikai feltétele az előbb kapott szomszéd elem accept(Person p) metódusából visszatérő bool értéke. Ha a következő mező befogadja a játékost, vagyis az accept true-val tér vissza, akkor a remove(Person p) függvénnyel az előző elem eltávolítja a játékost, a Person objektum element attribútumának pedig beállításra kerül a következő mező. Ha az újonnan beállított element tagváltozó getIsSlippery tagfüggvénye true-val tér vissza, akkor game random attribútumának meghívja a decideSlipping függvényét, amely az elemnek annak a szomszédjának az indexével tér vissza, amelyikre átcsúszik a játékos, vagyis amelyik PipelineElement beállításra kerül elementnek. Ehhez az eredeti lépéshez analóg módon a getNeighborElement, accept, és remove hívások követik egymást. Továbbá az előző elementnek, vagyis amelyikről lecsúszott, meg kell hívni a setIsSlippery metódusát paraméterként false-t átadva, mivel a csúszósság hatása csak egy csúszás erejéig tart. Ha az új element getIsSticky metódusának a visszatérési értéke true, vagyis ragadós az elem akkor meghívja a randomnak a decideStuckTime() metódusát, amely eldönti, hogy a Person mennyi ideig ragadjon az elemen.
  - **+void makeSticky():** Meghívja az element makeSticky nevű függvényét. Ha egy csövön van játékos, ragadóssá teszi a csövet a következő játékosnak, egyébként nincs hatással az objektum állapotára a függvény.
  - **+void tick():** A Tickable interface által kapott metódus megvalósítása. Amennyiben a stuckCounter attribútum értéke meghaladja a nullát, akkor annak az értékét csökkenti eggyel. Ha stuckCounter nulla, akkor pedig az element tagváltozónak meghívja a setIsSticky függvényét false paraméterrel, ezzel beállítva, hogy már nem ragadós a cső.

- **+void redirectPump(int firstIdx, int secondIdx):** Meghívja az element redirect nevű metódusát. Csak akkor van hatással a játékra, hogyha az element Pump, mivel csak a többi esetén üres a függvény törzse.
- **+void damagePipe():** Meghívja az element damageElement nevű függvényét. Ha egy csövön van a játékos, akkor lyukas állapotba teszi. Egyébként semmi nem történik.

8.1.7 Pipe

- **Felelősség**

A csövek megvalósításáért felelős osztály. Eltárolásra kerül a csövek kapacitása, vízszintje, hogy mennyi ideig nem lehet ismételten kilyukasztani, hogy lyukasak-e, ragadósak-e, csúszósak-e, illetve, hogy foglalt-e, vagyis, hogy áll-e rajta már játékos. Amennyiben egy szerelő szeretné az egyik végét egy másik pumpához csatlakoztatni, annak a végrehajtása ezen osztályon belül valósul meg. Ugyanez vonatkozik arra is, ha a szabotőr szeretné kilyukasztani. Az egyik szomszédja ismeretében le lehet kérdezni a másikat. Az ősosztályán keresztül örökölt accept metódust felüldefiniálja, hogy eldöntse, hogy léphet-e rá játékos. Ugyanígy a víz továbbítása és elszivárgása is örökölt felelősség.

- **Ősosztályok**

PipelineElement -> Pipe

- **Interfészek**

Tickable

- **Attribútumok**
  - **-int capacity**: A cső kapacitása
  - **-int waterLevel**: Az aktuális vízszint. . Befele pumpálás esetén növekszik az értéke, ha belőle pumpálják ki, akkor csökken. Ha eléri a maximumot, akkor nem lehet több vizet pumpálni bele.
  - **-bool isDamaged**: Ennek az értéke mondja meg, hogy lyukas-e a cső. Ha true akkor lyukas, false esetében nem.
  - **-bool isSlippery**: Azt határozza meg, hogy csúszós-e a cső. Ha true az értéke akkor az, false esetén pedig ellenkezőleg.
  - **-bool isSticky**: Azt határozza meg, hogy ragadós-e a cső. Ha true az értéke akkor az, false esetén pedig ellenkezőleg.
  - **-int fixedTime**: Az az idő, amennyi ideig nem lehet újra kilyukasztani.

- **Metódusok**
  - **+void damageElement():** Ha a csövön áll egy játékos, aki szeretné kilyukasztani a csövet, akkor hívja meg az ő damagePipe() függvénye ezt a metódust. Ha a fixedTiem attribútum értéke 0, akkor saját magán meghívja a setIsDamaged(bool d) metódust true paraméterrel, ezzel átállítva az isDamaged logikai változó értékét true-ra. Különben nem történik változás az objektum állapotában.
  - **+bool accept(Person p):** Az ősosztály azonos nevű függvényének a felüldefiniálása. Ha a players tömb nem üres, akkor nem fogadja be, false-al tér vissza, mivel a csövön egy időben csak egy ember állhat. Ellenkező esetben nem áll senki a csövön, szóval befogadja a játékost. Amennyiben befogadja a játékost, akkor hozzáadja a Person objektumot a players tömbhöz, és visszatér true-val, hogy a Person is tudja, hogy beadhatja-e a csövet a setElement(PipelineElement element) függvénynek.

- **+int propagateWater(int from, int qty):** A víz folyásának a megvalósítása. Egy If-Else szerkezetbe van beágyazva a függvény törzse. Az If ág akkor fut le, hogyha lyukas a cső, vagyis, ha az isDamaged értéke true. Ekkor nem befolyik a csőbe a víz, hanem a lyukon keresztül kifolyik belőle. Ilyenkor a Game objektum addLostWater(int qty) függvénye kerül ezesetben meghívásra, paraméterként pedig a qty paraméter és a capacity tagváltozó minimumát kapja meg. Erre azért van szükség, mert nem folyhat ki belőle több víz, mint amennyi van benne. A függvény visszatérési értéke ugyanez a min(qty, capacity) lesz. Az Else ág pedig az az eset, amikor a pumpa sikeresen tud a csőbe vizet pumpálni, mivel az nem lyukas. Elsőként meghívásra kerül a csövön a getOtherNeighbor(PipelineElement element) metódus, az eredeti függvényhívásban kapott from int-tel paraméterként. Ez visszaadja azt a PipelineElement objektumot, amelyik szomszédja a csőnek, de nem egyezik meg a from-mal. Vagyis ez az az elem, amelyikbe kerül a csőből a víz. Ezt követően a setWaterLevel(int qty) függvény segítségével új értéket ad a vízszintnek. A qty az a befolyt víz, a waterlevel meg az eddigi vízszint, így ezeknek az összege kellene, hogy legyen az új vízszint, ez viszont meghaladhatja a kapacitás mértékét. Ezért a setWaterLevel paramétere a kapacitás és az előbb említett összeg minimuma lesz, vagyis min(qty+waterlevel, capacity). Miután a csőbe eljutott a víz, az egyből továbbításra is kerül a túlsó végén lévő PipelineElement objektumba. Ehhez a korábban kapott otherNeighbor-on hívja meg a propagateWater metódust. Az első paramétere a Pipe objektum referenciája, a második pedig a waterLevel. A waterLevelből annyi vizet fogad el az elem, hogy az ő új vízszintje ne haladja meg a kapacitását. Visszatérési értékként megadja, hogy mennyi vizet fogadott el. Erre azért van szükség, mert ennyi víz fogyott el a csőből, tehát ismételten állítani kell a vízszinten, ezúttal a setWaterLevel(waterLevel-waterAccepted) függvényhívással. A legkülső függvény is visszatér, a waterAccepted+waterLevel összeggel.
- **+int seepWater(PipelineElement rom, int qty):** Ez a függvény azt mondja meg, hogy megadott vízmennyiségből a hívott félen mennyi szivárogna (seep) el a talajba, amiatt, hogy annak szabad a vége**.** Meghívja a getOtherNeighbor(PipelineElement element) függvényt a from paraméterrel, ezzel megkapja azt a szomszédot, amelyik a másik végén van. Ennek az objektumnak meghívja a seepWater(PipelineElement from, int qty) függvényét, azonban az eredetileg kapott qty paraméter helyett a qty és a capacity tagváltozó minimumát adja át. A metódus visszatér azzal a mennyiséggel amennyi az adott objektumon keresztül el tudott szivárogni a külső függvényhívás szintén ugyanezzel az értékkel tér vissza.
- **+PipelineElement getOtherNeighbor(PipelineElement element):** Azt az információt használjuk ki, hogy a csöveknek pontosan két szomszédja van. Megvizsgálja mindkét szomszédot, és azzal tér vissza, amelyik nem egyezik meg a függvényparaméterrel.
- **+void fix():** A cső befoltozásának a megvalósítása. Ha az isDamaged változó értéke true, vagyis a cső valóban lyukas, csak akkor hajtódnak végre az objektum állapotát állító események. Ekkor a setIsDamaged által false-ra állítja az isDamaged tagváltozót, a Random osztály decideFixedTime() függvényével pedig eldönti, hogy mennyi ideig nem lehet kilyukasztani a csövet újra. A decideFixedTime egy int-tel tért vissza, ezt átadjuk a setFixedTime függvénynek, ezzel frissítve a fixedTime változó értékét.
- **+bool split(Pump p):** A Plumber osztály addPump() metódusa hívja meg, az ősosztálytól örökölt függvény implementálása. A szomszéd lekérdezése által kapott pump_2 Pump objektumnak meghívja a disconnect(Pipe p) függvényét, az elementtel, mint paraméter. Ez egy logikai változóval tér vissza, hogy sikeres volt-e a művelet

vagy sem. Csak akkor fut le a függvény maradék része, ha ez true-val tért vissza. A cső kettévágásához létre kell hozni egy új csövet, amelynek a bemenete a most lerakott pumpa lesz, a kimenete pedig az a pumpa, ami az eddigi cső végére volt csatlakoztatva. Az eredeti vízszint fele kerül mindkét csőbe így. Tehét a pipelineElement létrehoz egy új Pipe-ot, majd ennek, és az elementnek (Vagyis az eddig is meglévő csőnek) meghívja a setWaterLevel(int qty) függvényét paraméterként a már lekérdezett waterLevel (wl a változó neve) felét megadva. Az element neighbor tömbjéből eltávolítja az 1-es indexű elemet, majd a helyére hozzáadja a pickedUpPump változót. Az új csőnek még nincs egy szomszédja se, ezért először hozzáadja a neighbors-hoz a pickedUpPump-ot, majd az eddig is meglévő pumpát. A pickedUpPump-ot pedig null-al kell egyenlővé tenni, hogy ne lehessen ugyanazt a pumpát végtelenszer lerakni.

- **+void makeSlippery():** Felüldefiniálja a PipelineElement makeSlippery metódusát. Egy játékos makeSlippery függvénye végrehajtja ezt a metódust, ha épp egy csövön áll. A Pipe objektumnak hívja meg a setIsSlippery függvényét, true paraméterrel.
- **+void makeSticky():** Felüldefiniálja a PipelineElement makeSticky metódusát. Egy játékos makeStickyfüggvénye végrehajtja ezt a metódust, ha épp egy csövön áll. A Pipe objektumnak hívja meg a setIsSticky függvényét, true paraméterrel.
- **+bool getIsSticky():** Csak ez az osztály implementálja nem üres függvénytörzzsel. A isSticky változó értékével tér vissza.
- **+bool getIsSlippery():** Csak ez az osztály implementálja nem üres függvénytörzzsel. A isSlippery változó értékével tér vissza.
- **+void setIsSticky():** Csak ez az osztály implementálja nem üres függvénytörzzsel. A isSticky változó értékékét modósítja.
- **+void tick():** Minden Timer tick eseményre, a cső fixedTime értéke csökken egyel, ha nem nulla.

8.1.8 PipelineElement

- **Felelősség**

Közös őse a hálózatot alkotó elemeknek, vagyis a ciszternáknak, forrásoknak, a sivatagnak, a csöveknek és a pumpáknak. Absztrakt osztály. Minden közös funkció ezen az osztályon belül van megvalósítva. Ez az osztály tud elfogadni és elutasítani mezőre lépni próbálkozó játékost. Felelőssége még eltávolítani játékost a mezőről, ha az ellép onnan. Ismeri, hogy milyen elemekkel áll szomszédságban, le is lehet kérdezni adott elemről, hogy a szomszédja-e, hozzá lehet adni új szomszédot, és azonosító alapján megkapni már meglévőt. Amikor a játékos átállítja a pumpát, megrongálni a csövet, vagy a cső végét akarja másik csőre átkötni, akkor is ennek az osztálynak hívja meg a megfelelő metódusait, amelyeket a Pipe és a Pump osztály felüldefiniál. Eltárolja azt is, hogy cső esetében fel van-e véve az elem.

- **Ősosztályok**

-

- **Interfészek**

-

- **Attribútumok**
  - **#Random random:** Ennek a segítségével valósítja meg a pumpák egymástól független véletlen időközönkénti elromlását, illetve az új csövek létrehozását.

- **#PipelineElement[0..*] neighbors:** Az elem szomszédjainak a halmaza.
- **#Person[0..*] players:** A játékosok akik rajta állnak az elemen. Az aktív elemek miatt tömbnek kell lennie, hiszen azon több játékos is tartózkodhat egyszerre.
- **#bool isPickedUp:** Ennek a segítségével azonosítjuk, ha az elem fel lett-e véve. Alapértelmezett esetben false az értéke, ami azt jelenti, hogy nincs felvéve. Azért kell ezt az alapértelmezett értéket beállítani, mivel nem az összes elemet lehet felvenni.

- **Metódusok**
  - **+void remove(Person p)**: Eltávolítja a paraméterként kapott Person objektumot a players tömb elemei közül.
  - **+void fix():** Csak a cső és a pumpa valósítja meg, mivel a többi leszármazott nem tud elromolni/kilyukadni, ezért megjavítani se lehet őket.
  - **+bool isNeighbor(PipelineElement element):** Végigiterál a neighbors tömbön, és megkeresi, hogy a paraméterként kapott PipelineElement-et tartalmazza-e a tömb. Ha igen, akkor true-val tér vissza, különben a visszatérési érték false.
  - **+PipelineElement getNeighborElement(int idx):** Megkeresi a neighbors tömbnek a paraméterrel megegyező indexű elemét, és visszatér vele.
  - **+void addNeighbor(PipelineElement newElement):** A szomszédokból álló tömbbe(neighbors) felveszi a paraméterként kapott PipelineElement-et.
  - **+void redirect(int firstIdx, int secondIdx):** A Person osztály redirectPump metódusa hívja meg, továbbadva azokat a paramétereket amiket kapott.
  - **+void removeNeighbor(PipelineElement element):** A paraméterként kapott indexű elemet eltávolítja a szomszédokat tároló tömbből.
  - **+int propagateWater(PipelineElement from, int qty):** A víz mozgatását megvalósító függvény. Nem tartozik hozzá megvalósítás, a leszármazottak implementálják.
  - **+void damageElement():** A Person osztály damagePipe() metódusa hívja meg azon a PipelineElement objektumon, amelyiken éppen áll. Ebben az osztályban nem tartozik hozzá implementáció. A játékosok csak csöveket tudnak kilyukasztani, így csak a Pipe osztály valósítja meg nem üres függvényként.
  - **+int seepWater(PipelineElement from, int qty):** A szivárgást valósítja meg. A leszármazottak implementálják.
  - **+bool accept(Person p):** A leszármazottak valósítják meg. Azt dönti el, hogy befogadja-e az elemre lépni szándékozó játékost.
  - **+void tick():** A timer azonos nevű metódusa hívja meg adott időközönként. A leszármazottak felüldefiniálják.
  - **+Pump pickUpPump(Person p):** A ciszternánál új cső felvételét valósítja meg. A leszármazottak közül csak a WaterTank implementálja, hiszen a többi elemnél nem lehetséges ilyen művelet végrehajtása. Visszatér a felvett elemmel.
  - **+Pipe pickUpPipe(int i):** A cső lecsatlakoztatásának és elvitelének a megvalósítása. A leszármazottak közül csakis kizárólag a Pump osztály implementálja nem üres függvénytörzzsel. A Plumber osztály azonos nevű metódusa hívja meg. Visszatér a felvett elemmel.
  - **+bool split(p: Pump):** Új pumpa lerakásához szükséges. A Pipe az egyetlen osztály, amelyik megvalósítja, a többi leszármazott esetében üres a függvény törzse.
  - **+bool disconnect(Pipe p):** Pumpa esetén lecsatlakoztatja az adott elemről a paraméterként átadott csövet. Ha es sikeres volt akkor true-val tér vissza, egyébként false-al.

- **+bool connectPipeEnd(Pipe pickedUpPipe):** A cső egyik felvett végének a hálózatba csatlakoztatásáért felelős. Csak a Pump valósítja meg ténylegesen, mivel csak arra lehet csatlakoztatni csövet. A többi leszármazott osztály esetén üres a törzse.
- **+void makeSlippery():** Csak a cső osztály implementálja nem üres függvénytörzzsel. A cső állapotát átírja csúszósra.
- **+void makeSticky():** Csak a cső osztály implementálja nem üres függvénytörzzsel. A cső állapotát átírja ragadósra.
- **+bool getIsSticky():** Csak a cső osztály implementálja nem üres függvénytörzzsel. A cső isSticky változó értékével tér vissza.
- **+bool getIsSlippery():** Csak a cső osztály implementálja nem üres függvénytörzzsel. A cső isSlippery változó értékével tér vissza.
- **+void setIsSticky():** Csak a cső osztály implementálja nem üres függvénytörzzsel. A cső isSticky változó értékékét modósítja.

8.1.9 Plumber

- **Felelősség**

A vízvezetékszerelőket megvalósító osztály, a Person osztály leszármazottja. Tud felvenni új pumpát, lerakni azt, illetve a már pályán elhelyezkedő pumpákat átállítani. Képes megjavítani mindkét fajta pályaelemet. Csövek végét is le tudja csatlakoztatni és átkötni másik pumpára.

- **Ősosztályok**

Person -> Plumber

- **Interfészek**

-

- **Attribútumok**
  - **-Pipe pickedUpPipe:** A cső, aminek a végét lecsatlakoztatta egy pumpáról, és viszi magával, hogy egy másikra felcsatlakoztassa. Egyszerre csak egy lehet a játékosnál.
  - **-Pump pickedUpPump:** A ciszternáknál felvett pumpa. A csőhöz hasonlóan egy időben csak egyet birtokolhat.

- **Metódusok**
  - **+void fix():** Megjavítja az elemet amin épp áll. Ehhez meghívja a Person osztályból örökölt element adattag fix() függvényét, ami abban az esetben, ha az element Pump vagy Pipe, akkor az isDamaged bool típusú tagváltozóját false-ra állítja.
  - **+Pump pickUpPump():** Ezzel a függvénnyel vesz fel egy lerakható pumpát a ciszternától a játékos. Csak akkor kerül végrehajtásra a függvénytörzs, hogyha a pickedUpPump attribútum null, vagyis nincs nála pumpa. Ekkor meghívja az element tagváltozó azonos nevű függvényét. Csak akkor fog történni bármi, hogyha WaterTank elemen áll, mivel egyedül az valósítja meg ezt a függvényt. Ez a metódus létrehoz egy új pumpát, majd ezt az újonnan létrejött Pump-ot átadja a Plumber setPickedUpPump(Pump p) metódusának, így most a szerelő birtokában van már a pumpa. Visszatér a felvett elemmel.
  - **+void addPump():** A metódus törzsének teljes egésze egy If függvényben helyezkedik el. Ez az If azt ellenőrzi, hogy a pickedUpPump attribútum referenciája nem null-e. Ha null, akkor a játékosnál nincs pumpa, szóval nincs mit leraknia és nem történik semmi. Ellenkező esetben meghívja az element attribútum Split(Pump p)

metódusát. Ezt csak a Pipe implementálja nem üres függvénytörzzsel, ezért, ha más fajta elemen áll a játékos, akkor nem tud lerakni pumpát.

- **+Pipe pickUpPipe(int i):** Az új pumpa felvételéhez hasonlóan itt is megvizsgálja, hogy nincs-e lecsatlakoztatott végű cső már a játékosnál, vagyis hogy a pickedUpPipe változó null-e. Ha nincs nála, vagyis null, akkor meghívja az element tagváltozó azonos nevű függvényét. Ezt csak a pumpa implementálja, hiszen csak arról lehet csövet lecsatlakoztatni. Ez a függvény lekérdezi, hogy a szomszédokat tartalmazó i indexű eleme melyik cső. Ezután az element disconnect függvényének ezt átadja paraméterként, majd egy logikai változó formájában visszakapva azt, hogy sikeresen le lett-e csatlakozva. A függvénytörzs maradék része ennek a bool-nak csak true értéke esetén kerül végrehajtásra. Itt már csak annyi történik, hogy a Plumber objektum pickedUpPipe attribútumát egyenlővé teszi azzal a csővel, amit lecsatolt. Visszatér a csővel.
- **+void connectPipeEnd():** A cső egyik felvett végének a hálózatba csatlakoztatásáért felelős. Azonos logikával működik, mint az addPump függvény. Elöször megnézi, hogy van-e cső, amit le lehet rakni (pickedUpPipe nem null-e), és ha van, csak akkor csinál bármit is. Az element attribútumon meghívja a vele analóg nevű metódust.

8.1.10 Proto

- **Felelősség**

Ez az osztály felelős a program megfelelő működtetéséért. Kezeli a parancsokat, amelyeket bekér a felhasználóktól, és a képernyőre kiíratásokat is ez az osztály végzi el. Eltárolja az összes objektumot, hozzájuk rendelve azt a nevet, amellyel a játékosok hivatkozhatnak rá. A játékmenet köreit is kezeli.

- **Ősosztályok**

-

- **Interfészek**

-

- **Attribútumok**
  - **-Map<String,Object> nameObjectMap**: Név - objektum párokban tárolja a játék során használt objektumokat. A felhasználók név szerint tudják megadni a parancsoknak az objektumokat, ezért szükséges ilyen módon eltárolni.
  - **-Map<Object,String> objectNameMap:** Objektum-név párokban tárolja a játék során használt objektumokat.
  - **-bool isDebug:** Azt mondja meg, hogy a játék debug módban-e lett indítva. Bizonyos parancsok csak akkor érhetőek el, ha ennek az értéke true.
  - **-ArrayList<Command> commands:** A parancsokat tartalmazó lista. Megfelelő bemenet esetén innen kerül kiválasztása, hogy melyik fusson le.
  - **-FileHandler fileHandler:** A pályák fájlból való beolvasását és kiírását kezeli.

- **Metódusok**
  - **+Object getByName(String name):** Visszatér a nameObjectMap tagváltozó get függvényének a name paraméterre adott visszatérési értékével.

- **+String getByObject(Object object):** Visszetér az objectNameMap tagváltozó get függvényének az object paraméterre adott visszatérési értékével.
- **+void addObject(String name, Object object):** Hozzáadja a paraméterül átadott név-objektum párost mindkét Map típusú attribútumhoz.
- **-String readCommand():** A szabványos bemenetről beolvas egy parancsot. Egyelőre még nem dolgozza fel, hanem a teljes paranccsal visszatér.
- **-ArrayList<String> parseCommand(String cmd):** Létrehoz egy String típusú elemekből álló ArrayList típusú listát. A paraméterként megadott stringet (amely a felhasználó által megadott parancs) szóközök mentén feldarabolja, minden darabot ebbe az ArrayListbe rakja bele, majd visszatér vele.
- **+void PlayGame():** A játékmenet körönkénti megvalósításáért felelős metódus. A readCommand függvénnyel beolvasunk egy parancsot, majd a parseCommandnak beadjuk a visszatérési értékét. Ha az így kapott lista első eleme nem load, akkor ezt addig ismételjük, ameddig nem lesz az. Ekkor megtörténik a file-ból beolvasás, a parancs második paramétere alapján pedig beállítjuk az isDebug változó értékét. Ezt követően egy while ciklusba van foglalva a függvénytörzs hátralevő része, amelynek minden iterációja egy játékos körének felel meg. A ciklus feltétele az, hogy egy minden iterációban inkrementált integer értéke kisebb vagy egyenlő-e, mint a körök megadott száma. Ezt követően egy belső ciklusban ellenőrizzük, hogy milyen parancsot ad meg az aktuális játékos. Addig tart a játékos köre, ameddig nem ad meg olyan parancsot, amelynek egy játékos a paramétere, de nem move. Minden iterációban meghívja a commands lista megfelelő parancsának a runFunction metódusát. Ennek paraméterként átadjuk parancs parseCommand stringjei alapján az objektumlistából lekérdezett objektumokból álló listát. A belső ciklust követően FIFO szerűen a következő játékosra kerül sor hasonlóképpen, ehhez a game rotatePlayers metódusát kell meghívni.
- **+void load (ArrayList<Object> options):** Betölti a megadott pályát.
- **+void save (ArrayList<Object> options):** A bemeneti formátummal megegyező formában kiírja az objektumok aktuális állapotát a kimenetre.
- **+void state (ArrayList<Object> options):** Kiírja a megadott elemről rendelkezésre álló adatokat.
- **+void exit (ArrayList<Object> options):** Megszakítja az adott játékmenetet, a load paranccsal lehet új játékot indítani.
- **+void move (ArrayList<Object> options):** Megadott játékossal megpróbálunk lépni a megadott irányba.
- **+void damage (ArrayList<Object> options):** A megadott játékos megpróbálja kilyukasztani az elemet, amin áll. Ha nem csövön áll, nem történik semmi.
- **+void repair (ArrayList<Object> options):** A megadott játékos megpróbálja megjavítani az elemet, amin áll.
- **+void pickUp (ArrayList<Object> options):** Az adott játékos megpróbál felvenni megadott típusú elemet.
- **+void connectPipeEnd (ArrayList<Object> options):** A megadott játékos csatlakoztatja (valamelyik) nála lévő csővéget az elemhez, amin áll, ha az engedi.
- **+void addPump (ArrayList<Object> options):** A megadott játékos megpróbálja lerakni a nála lévő pumpát (már ha van nála) az elemre, amin áll.
- **+void redirect (ArrayList<Object> options):** A megadott játékos megpróbálja átirányítani a pumpát, amin áll, ha valóban pumpán áll.
- **+void makeSlippery(ArrayList<Object> options):** Az adott játékos megpróbál felvenni megadott típusú elemet.

- **+void makeSticky(ArrayList<Object> options):** A megadott játékos megpróbálja ragadóssá tenni az elemet, amin áll.
- **+void plumberPoints(ArrayList<Object> options):** Kiírja a szerelők aktuális pontszámát.
- **+void saboteurPoints(ArrayList<Object> options):** Kiírja a szabotőrök aktuális pontszámát.
- **+void dbgTick(ArrayList<Object> options):** A megadott objektumon meghívja a tick() függvényt.
- **+void dbgTickAll(ArrayList<Object> options):** Minden Tickable interfészt megvalósító objektumon meghívja a tick() függvényt.
- **+void dbgCreatePipe(ArrayList<Object> options):** Új csövet hoz létre a megadott ciszternán.
- **+void dbgCreatePump(ArrayList<Object> options):** Új pumpát hoz létre a megadott ciszternán.
- **+void dbgCreatePump(ArrayList<Object> options):** Elrontja a megadott elemet. Abban különbözik a damage utasítástól, hogy itt akkor is elronthatunk egy elemet, ha nem áll rajta senki, illetve, ha az adott elem Pumpa.
- **+void dbgSetLeakable (ArrayList<Object> options):** Beállítja, hogy az adott cső meddig ne lehessen kilyukasztható. 0 esetén azonnal kilyukaszthatóvá válik a cső.

8.1.11 Pump

- **Felelősség**

Ez az osztály a pumpákat implementálja. A pumpák a csomópontjai a vízvezeték-rendszernek, ezért akárhányan állhatnak rajta. Amikor a játékos átállítja a pumpát, akkor valójában ennek az osztálynak a felelőssége beállítani, hogy a rá csatlakoztatott csövek közül melyikből, melyikbe pumpáljon.

- **Ősosztályok**

PipelineElement->Pump

- **Interfészek**

Tickable

- **Attribútumok**
  - **-int fromPipe**: Az a cső, amelyikből pumpálja a vizet.
  - **-int toPipe**: A cső, amelyikbe pumpálja a vizet.
  - **-PumpTank tank**: A pumpának a tartálya.

- **Metódusok**
  - **+void redirect(int firstIdx, int secondIdx)**: Az ősosztálytól örökölt metódus implementációja. A Person osztály azonos paraméterezésű redirectPump függvénye hívja meg. Mindkét paraméter esetében megvizsgálja, hogy az ahhoz az indexű elem nem null-e, és csak akkor fut tovább, ha egyik se az, hiszen csak akkor lehet kimenetnek és bemenetnek beállítani. Emellett egy harmadik logikai feltétel, hogy a két cső ne egyezzen meg egymással. Ezután már csak két függvényhívás következik, amelyekkel a ki- és bemenet kerül beállításra: setFromPipe(first), setToPipe(second).
  - **+int propagateWater(PipelineElement from, int qty):** A bemeneti csőből a kimenetre való pumpálást valósítja meg. Az egész függvénytörzs egy If-Else

szerkezetben helyezkedik el. Az If ágban két dolgot vizsgálunk, hogy a from paraméterű cső van-e a bemenetre csatlakoztatva, hiszen csak akkor tud belőle pumpálni vizet. A másik az, hogy a pumpa el van-e romolva. A két logikai feltétel között vagy kapcsolat van, mivel bármelyik beteljesül már meghiúsul a művelet. Ezek alapján a feltétel a következő: (from!=fromPipe)||(isDamaged == true). Ha az If ág fut le, akkor a függvény annyit csinál, hogy visszatér 0-val, hiszen annyi vizet sikerült továbbítani. A lényegi történések az else ágban történnek. Ahogy bejut a bemeneti csőből a víz a pumpába, az annak a tartályába kerül tárolásra. Ez azt jelenti, hogy a Pump a tank paraméterén meghívja az addWater(int qty) metódust, a legkülső függvény qty paraméterét továbbadva. Ez qty-vel növeli a tank storedWater attribútumát, maximum a tartály kapacitásáig. Most, hogy a pumpában van már a víz, azt át kell pumpálni a következő csőre. Ennek a csőnek meg kell hívni a propagateWater(PipelineElement from, int qty) metódusát, this és storedWater paraméterezéssel, hiszen a forrása a vízfolyamnak a pumpa amelyik az egész hívásláncot elindította, és szeretné a tartályában lévő összes vizet továbbadni. A PropagateWater visszatér azzal az értékkel, amennyi vizet képes volt elfogadni, anélkül, hogy túllépte volna a kapacitását. Ennyi vizet kell eltávolítani a pumpa tartályából a tank.removeWater(acceptedWater) függvényhívással. A metódus legvégül visszatér a storedWaterrel.

- **+bool disconnect(Pipe p):** Akkor tér vissza true-val, ha sikeresen eltávolította a paraméterben megadott elemet a neighbor tömbből. Ez akkor lehet sikeres, hogyha nem olyan csövet akar lecsatlakoztatni, amelyikből, vagy amelyikbe pumpál. Vagyis egy if függvénybe kell zárni a műveletet, amelynek a feltétele: [p!=fromPipe&&p!=toPipe]. Ha ez nem teljesül, akkor nem történik semmi, és false-al tér vissza a metódus. Egyébként meghívja magán a removeNeighbor metódust, paraméterként átadva azt a p objektumot, amit a disconnect kapott paraméterként. Ezt követően tér vissza, értelemszerűen true-val.
- **+Pipe pickUpPipeEnd(int i):** Az ősosztálytól örökölt metódus, egyedül ez az osztály implementálja. A Plumber osztály ezzel megegyező nevű metódusa használja. Ez a függvény lekérdezi, hogy a szomszédokat tartalmazó neighbors tömb i indexű eleme melyik cső. Ezután az element disconnect függvényének ezt átadja paraméterként, majd egy logikai változó formájában visszakapja azt, hogy sikeresen le lett-e csatlakozva. A függvénytörzs maradék része ennek a bool-nak csak true értéke esetén kerül végrehajtásra. Itt már csak annyi történik, hogy a Plumber objektum pickedUpPipe attribútumát egyenlővé teszi azzal a csővel, amit lecsatolt. Visszatér a csővel.
- **+void connectPipeEnd():** A cső egyik felvett végének a hálózatba csatlakoztatásáért felelős. Az előző függvényhez hasonlóan csak ez az egy osztály valósítja meg. Megpróbálja hozzáadni a neigbors tömbhöz az addNeighbor függvény segítségével a pickedUpPipe-ot. Ha ez sikerült akkor az addNeighbor visszatér true-val, ellenkező esetben false-al. Ha true-val tért vissza, akkor a pickedUpPipe tagváltozó értékét null-ra állítja.
- **+bool accept(Person p):** Mindig true-val tér vissza, mert akárhányan ráléphetnek egyszerre a pumpára.
- **+void fix():** Az isDamaged értékét true-ra állítja.
- **+Pipe removePipe(int idx):** A paraméterrel megegyező indexű elemet eltávolítja a neighbors tömbből, és visszatér az eltávolított csővel.
- **+void tick():** A Tickable interface által kapoĺaystt metódus megvalósítása. Minden tick-re meghívja a random nevű attribútum decidePump() metódusát, amely eldönti, hogy elromoljon-e a pumpa. Ez alapján állítja be az isDamaged értékét.

8.1.12 PumpTank

- **Felelősség**

A pumpák víztartályának a reprezentációja. Ebben kerül eltárolásra a pumpában lévő víz mennyisége, és ez az osztály felelős ennek a mennyiségnek a megváltoztatásáért is.

- **Ősosztályok**
  -
- **Interfészek**
  -
- **Attribútumok**
  - **-int storedWater**: A pumpa tankjában lévő vízmennyiség.
  - **-int capacity**: A tartály kapacitása.
- **Metódusok**
  - **+int addWater(int qty):** A storedWater-t növeli a paraméterként kapott értékkel a kapacitásáig. Visszatér azzal, hogy mennyi vizet tudott sikeresen hozzáadni.
  - **+void removeWater(int qty):** A storedWater értékét csökkenti a paraméterként kapott qty értékkel. 0 alá nem mehet.

8.1.13 Random

- **Felelősség**

Ez az osztály minden olyan eseményért felelős, ami véletlen következik be. Ilyen például a pumpa elromlása és az új csövek generálása.

- **Ősosztályok**
  -
- **Interfészek**
  -
- **Attribútumok**
  -
- **Metódusok**
  - **+bool decidePump():** Egy random generált érték alapján eldönti, hogy elromoljon-e a pumpa, vagy sem, majd visszatér a döntéssel.
  - **+bool decideNewPipe():** Egy random generált érték alapján eldönti, hogy létrejöjjön-e új pumpa a ciszternánál, majd visszatér a döntéssel.
  - **+int decideFixedTime():** Egy random javítási időt generál egy csőnek és visszatér vele. Ha egy csőnek a fixedTime értéke nem nulla, nem lehet kilyukasztani.
  - **+int decideStuckTime():** Egy random időt generál egy játékosnak, ha egy ragadós csőre lép, majd visszatér vele. Ha egy játékosnak a stuckTime értéke nem nulla, nem léphet.

- **+int decideSlipping():** Egy random 0 vagy 1 értéket generál, majd visszatér vele. Ha 0, akkor a cső 0. indexre csúszik a játékos, egyébként az 1. indexre.

8.1.14 Saboteur

- **Felelősség**

A Plumber-hez hasonlóan a Person osztály leszármazottja. Az örökölt felelősségein kívül implementálja a csövek csúszóssá tevését.

- **Ősosztályok**

  Person -> Saboteur

- **Interfészek**

  -

- **Attribútumok**

  -

- **Metódusok**
  - **+void makeSlippery():** Meghívja az element attribútum makeSlippery nevű függvényét. Ha a játékos pont egy csövön van, akkor csúszósra állítja a csövet, amíg a következő játékos rá nem lép.

8.1.15 Tickable

- **Felelősség**

Interfész, az összes aktív elem (forrás, ciszterna, pumpa), illetve a cső és a Person osztály is megvalósítja, ez felelős a periodikusan történő eseményeikért. (Pl.: víz továbbítása, új cső keletkezése, stuckCounter csökkentése, FixedTime csökkentése, stb.)

- **Ősosztályok**

  -

- **Metódusok**
  - **+void tick()**: az interfészt megvalósító osztályok ebben a függvényben valósítják meg a periodikus működésüket.

8.1.16 Timer

- **Felelősség**

Periodikus időzítő. Ez időzíti a Tickable interfészt megvalósító osztályok (Pump, Pipe, WaterTank, WaterSource, Person) periodikus működését.

- **Ősosztályok**

  -

- **Interfészek**

  -

- **Attribútumok**
  - **-Tickable[0..*] tickables**: heterogén kollekció, azokat a pályaelemeket tárolja, amelyek megvalósítják a Tickable interfészt.

- **Metódusok**
  - **+void tick():** a tickables kollekció minden elemének meghívja a tick() függvényét.
  - **+void addTickable(Tickable obj):** hozzáadja a paraméterként kapott újabb elemet a tickables kollekcióhoz.

8.1.17 WaterSource

- **Felelősség**

A vízforrást reprezentálja. A rájuk kötött csövekbe a víz eljuttatása a feladata.

- **Ősosztályok**

  PipelineElement -> WaterSource

- **Interfészek**

  Tickable

- **Attribútumok**

  -

- **Metódusok**
  - **+int propagateWater(PipelineElement from, int qty):** mindig 0-val tér vissza, a forrásnak nem lehet vizet továbbítani.
  - **+bool accept(Person p):** Mindig false-al tér vissza, hiszen a forrásra nem léphet játékos.
  - **+void tick():** Végigiterál a neighbors tömbön. Minden tömbelemnek lekérdezi a kapacitását, meghívja a propagateWater(PipelineElement from, int qty) metódusát, önmagát és a lekérdezett kapacitást megadva paraméternek, pontosan annyi vizet juttat a forrás a csőbe, amennyi elfér benne.

8.1.18 WaterTank

A ciszterna megvalósítása. Eltárolja és jelenti a Gamenek, hogy mennyi víz van benne, ez a vízmennyiség csökkenhet is, ugyanis el tud szivárogni a víz, ha van olyan Pipe rákötve, amelynek szabad a vége.

- **Ősosztályok**

  PipelineElement -> WaterTank

- **Interfészek**

  Tickable

- **Attribútumok**
  - **-int waterLevel:** A ciszternában lévő vízszint.

- **Metódusok**
  - **+int propagateWater(PipelineElement from, int qty):** egy hozzá kapcsolódó Pipe hívja meg rajta. Híváskor a saját vízszintjéhez hozzáádja a beérkező (qty) vízmennyiséget, majd a Game osztály addFinishedWater(int qty) függvényét meghívva frissíti a Gameben feljegyzett vízmennyiséget.
  - **+bool accept(Person p):** mindig true-val tér vissza, mert akárhányan állhatnak a ciszternán.
  - **+void tick():** A WaterTank tick() implementációja. Meghívása után először végigiterál a neighbors tömbjén, az összes elemen meghívja a seepWater(PipelineElement from, int qty), függvényét, önmagát és a tárolt vízmennyiséget megadva paraméternek, majd ennek a visszatérési értékét kivonja a tárolt vízmennyiségből és meghívja a Game removeFinishedWater(int qty) függvényét, szintén az előző függvény visszatérési értékét megadva paraméternek. Ezután a Random decideNewPump() függvényét hívja, ha az hamis értékkel tér vissza, nem történik semmi, ha igazzal, akkor a Gametől lekéri a Desert referenciáját, létrehoz egy új Pipe objektumot és beállítja a szomszédainak önmagát és a kapott Derest referenciát.
  - **+void addPipe(Pipe p):** hozzáadja a paraméterben megadott csövet a neighbors tömbhöz.
  - **+void removePipe(Pipe p):** kitörli a paraméterben megadott csövet a neighbors tömbből.
  - **+Pump pickUpPump(Person p):** a Plumber pickUpPump() függvénye hívja meg, amikor új pumpát szeretne felvenni. Ilyenkor a függvény létrehoz egy új Pump objektumot, majd meghívja a Plumber setPickedUpPump(Pump p) függvényét, paraméterként az új Pumpot megadva.

8.2 A tesztek részletes tervei, leírásuk a teszt nyelvén

A be-/kimenetek leírásánál előfordulnak olyan sorok (jellemzően a state parancs kimenete), amelyek a dokumentumban nem férnek ki egy sorba, azonban a program kimenetén egy sorban szerepelnek.

*A Pick Up Pipe End teszteseteknél egy új parancsot kellett létrehozni. Ennek leírása:

Bemenet:
`pickUpPipeEnd <player> <index>`
Leírás: Az adott játékos megpróbál felvenni egy csővéget.
Opciók:
- **`<player>`** - játékos, aki megpróbálja felvenni az elemet
- **`<index>`** - a pumpa melyik szomszéd cső vége

Kimenet:
`pickUpPipeEnd`
Csővég felvétele megtörtént.
Csővég felvétele sikertelen.
A játékos körének vége.

8.2.1 Damage Pipe (Damageable Pipe)

- **Leírás**
  Ez a teszteset a csövek kilyukasztását teszteli. Egy normális működőképes csövet próbál kilyukasztani.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - a cső lyukas állapotát jelző tagváltozó át kell legyen állítva (state pipe -> isDamaged = true)
- **Bemenet**
```
load DamageDamageablePipe.dat debug
damage playerOnPipe
state pipe
```
- **Elvárt kimenet**
```
Pálya betöltése sikeres.
Rongálás sikeres.
Pipe pipe
 neighbors:waterSource,desert;isPickedUp:false;capacity:0;waterLevel
 :0;isDamaged:true;isSlippery:false;fixedTime:0;isSticky:false
```

8.2.2 Damage Pipe (Not Damageable Pipe)

- **Leírás**
  Ez a teszteset a csövek kilyukasztását teszteli. Egy már nemrég megjavított csövet próbál kilyukasztani a játékos.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - a cső lyukas állapotát leíró tagváltozó false kell maradjon (state pipe -> isDamaged = false)
  - a cső fixedTime tagváltozója nem nulla (state pipe -> fixedTime > 0)

- **Bemenet**
```
load DamageDamageablePipe.dat debug
damage playerOnPipe
repair playerOnPipe
dbgSetLeakable pipe 5
damage PlayerOnPipe
state pipe
```
- **Kimenet**
```
Pálya betöltése sikeres.
Rongálás sikeres.
Javítás sikeres.
Rongálás sikertelen.
Pipe pipe
 neighbors:waterSource,desert;isPickedUp:false;capacity:0;waterLevel
 :0;isDamaged:false;isSlippery:false;fixedTime:5;isSticky:false
```

8.2.3 Fix Pump

- **Leírás**
  Ez a teszteset a pumpák javítását teszteli. Egy elromlott pumpát próbál javítani a játékos.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - a pumpa romlott állapotát leíró tagváltozó értéke false a javítást követően (state pump -> isDamaged = false)
- **Bemenet**
```
load FixPump.dat debug
dbgDamage Pump pump
state pump
repair playerOnPump
state pump
```
- **Kimenet**
```
Pálya betöltése sikeres.
Elem elromolva.
Pump pump
 neighbors:endPipe1,endPipe2;isPickedUp:false;isDamaged:true;toPipeI
 dx:1;fromPipeIdx:0
Javítás sikeres.
Pump pump
 neighbors:endPipe1,endPipe2;isPickedUp:false;isDamaged:false;toPipe
 Idx:1;fromPipeIdx:0
```

8.2.4 Fix Pipe

- **Leírás**
  Ez a teszteset a csövek befoltozását teszteli. Egy elromlott csövet próbál befoltozni egy játékos.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - a cső lyukas állapotát leíró tagváltozó értéke false a befoltozást követően (state pipe -> isDamaged = false)
  - a cső fixedTime tagváltozó értéke nem nulla (state pipe -> fixedTime > 0)
- **Bemenet**

```
load FixPipe.dat debug
damage playerOnPipe
state pipe
repair playerOnPipe
dbgSetLeakable pipe 5
state pipe
```

- **Kimenet**

```
Pálya betöltése sikeres.
Rongálás sikeres.
Pipe pipe
 neighbors:waterSource,desert;isPickedUp:false;capacity:0;waterLevel
 :0;isDamaged:true;isSlippery:false;fixedTime:0;isSticky:false
Javítás sikeres.
A cső nem lyukasztható ki ennyi ideig: 5.
Pipe pipe
 neighbors:waterSource,desert;isPickedUp:false;capacity:0;waterLevel
 :0;isDamaged:false;isSlippery:false;fixedTime:5;isSticky:false
```

8.2.5 Pick Up Pipe (from Pump)

- **Leírás**
  Ez a teszteset a csővégek lecsatlakoztatását és felcsatlakoztatását teszteli. Egy pumpán lévő játékos megpróbál lecsatlakoztatni egy csővéget, aztán újra felcsatlakoztatja. A felcsatlakoztatást követően, a játékos újból felveszi a csővéget, átlép egy másik pumpára, és ott is megpróbálja lecsatlakoztatni az egyik csővéget.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - a cső felvett állapotát leíró tagváltozó értéke true a lecsatlakoztatást követően (state pipe -> isPickedUp = true)
  - a játékos pickedUpPipe referenciája nem null (state player -> pickedUpPipe != null)
  - a felcsatlakoztatást követően, a játékos pickedUpPipe referenciája null és a felcsatlakoztatott cső isPickedUp tagváltozója false
  - Ha már van egy felvett csővég a játékosnál, nem tud felvenni
- **Bemenet**

```
load PickUpPipeFromPump.dat debug
pickUpPipeEnd plumber 0
state pipe
state plumber
connectPipeEnd plumber
state pipe
state plumber
pickUpPipeEnd plumber 0
move plumber 0
move plumber 0
pickUpPipeEnd plumber 0
```

- **Kimenet**

```
Pálya betöltése sikeres.
Csővég felvétele megtörtént.
Pipe pipe
 neighbors:pump2,pump;isPickedUp:true;capacity:0;waterLevel:0;isDama
 ged:false;isSlippery:false;fixedTime:0;isSticky:false
Plumber plumber
```

```
 stuckCounter:0;pickedUpPipe:pipe;pickedUpPump:null;element:pump
Csatlakoztatás sikeres.
Pipe pipe
 neighbors:pump2,pump;isPickedUp:false;capacity:0;waterLevel:0;isDam
 aged:false;isSlippery:false;fixedTime:0;isSticky:false
Plumber plumber
 stuckCounter:0;pickedUpPipe:null;pickedUpPump:null;element:pump
Csővég felvétele megtörtént.
Új pozíció: pipe2
Új pozíció: pump2
Csővég felvétele sikertelen.
```

8.2.6 Pick Up Pipe End (from WaterTank)

- **Leírás**
  Ez a teszteset a csővégek lecsatlakoztatását teszteli. Egy ciszternán lévő játékos megpróbál felvenni egy csővéget.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - a cső felvett állapotát leíró tagváltozó értéke true a lecsatlakoztatást követően (state pipe -> isPickedUp = false)
  - a játékos pickedUpPipe referenciája nem null (state player -> pickedUpPipe != null)
- **Bemenet**
  ```
  load PickUpPipeFromOtherElement.dat debug
  pickUp Pipe plumber
  state pipe
  state plumber
  ```
- **Kimenet**
  ```
  Pálya betöltése sikeres.
  Elem felvétele sikertelen.
  Pipe pipe
   neighbors:waterTank;isPickedUp:true;capacity:0;waterLevel:0;isDamag
   ed:false;isSlippery:false;fixedTime:0;isSticky:false
  Plumber plumber
   stuckCounter:0;pickedUpPipe:pipe;pickedUpPump:null;element:waterTan
   k
  ```

8.2.7 Pick Up Both Pipe Ends

- **Leírás**
  Ez a teszteset mindkét csővég lecsatlakoztatását teszteli egyidőben. Egy szerelő az egyik cső végén lévő pumpáról felveszi a csővéget amíg egy másik szerelő túloldalt szintén felveszi a másik csővéget.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - a cső felvett állapotát leíró tagváltozó értéke true a felcsatlakoztatást követően (state pipe -> isPickedUp = true)
  - mindkét játékos pickedUpPipe referenciája nem null (state player -> pickedUpPipe != null)
  - sikeresen fel lett véve mindkét csővég

- **Bemenet**
```
load PickUpBothPipeEnds.dat debug
pickUpPipeEnd plumber 0
pickUpPipeEnd plumber2 0
state pipe
state plumber
state plumber2
```
- **Kimenet**
```
Pálya betöltése sikeres.
Csővég felvétele megtörtént.
Csővég felvétele megtörtént.
Pipe pipe
 neighbors:pump,pump2;isPickedUp:true;capacity:0;waterLevel:0;isDama
 ged:false;isSlippery:false;fixedTime:0;isSticky:false
Plumber plumber
 stuckCounter:0;pickedUpPipe:pipe;pickedUpPump:null;element:pump
Plumber plumber2
 stuckCounter:0;pickedUpPipe:pipe;pickedUpPump:null;element:pump2
```

8.2.8 Pick Up Pipe (from Other Element)

- **Leírás**
  Ez a teszteset a csővégek lecsatlakoztatását teszteli. Egy nem pumpán lévő játékos megpróbál felvenni egy csövet.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - a cső felvett állapotát leíró tagváltozó értéke false a lecsatlakoztatást követően (state pipe -> isPickedUp = false)
  - a játékos pickedUpPipe referenciája null (state player -> pickedUpPipe = null)
- **Bemenet**
```
load PickUpPipeFromOtherElement.dat debug
pickUp Pipe plumber
pickUpPipeEnd plumber 0
state pipe
state plumber
```
- **Kimenet**
```
Pálya betöltése sikeres.
Elem felvétele sikertelen.
Csővég felvétele sikertelen.
Pipe pipe
 neighbors:waterSource,waterTank;isPickedUp:false;capacity:0;waterLe
 vel:0;isDamaged:false;isSlippery:false;fixedTime:0;isSticky:false
Plumber plumber
 stuckCounter:0;pickedUpPipe:null;pickedUpPump:null;element:waterTan
 k
```

8.2.9 Place New Pump

- **Leírás**
  Ez a teszteset a ciszternák által generált pumpák felvevését, illetve lerakását teszteli. Egy ciszternán lévő játékos megpróbál egy új pumpát felvenni és lerakni egy csőre. Ez a teszteset azt is teszteli, hogy ha nincs a játékosnál egy felvett pumpa, akkor

természetesen nem sikerül a letevés, illetve, ha már felvett a játékos egy pumpát, nem tud felvenni még egy pumpát.

- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - sikeresen generált egy pumpát a ciszterna
  - a generált pumpa isPickedUp tagváltozója megfelelően változik a felvevést követően (state pump -> isPickedUp = true)
  - a játékos pickedUpPump referenciája nem null (state player -> pickedUpPump != null)
  - a lerakott pumpa szomszédjai az eredti cső kettévágva
  - ha a játékosnak nincs pumpája (a letevést követően), akkor nem sikerül letenni egy pumpát
- **Bemenet**

```
load PlaceNewPump.dat debug
pickUp Pump plumber
pickUp Pump plumber
state plumber
state newPump
move plumber 0
state plumber
addPump plumber
state plumber
state newPump
addPump plumber
```

- **Kimenet**

```
Pálya betöltése sikeres.
Elem felvétele megtörtént.
Elem felvétele sikertelen.
Plumber plumber
 stuckCounter:0;pickedUpPipe:null;pickedUpPump:newPump;element:water
 Tank
Pump pump
 neighbors:;isPickedUp:true;isDamaged:false;toPipeIdx:-
 1;fromPipeIdx:-1
Új pozíció: pipe
Plumber plumber
 stuckCounter:0;pickedUpPipe:null;pickedUpPump:newPump;element:pipe
Pumpa lerakása sikeres.
Plumber plumber
 stuckCounter:0;pickedUpPipe:null;pickedUpPump:null;element:pipe
Pump pump
 neighbors:pipe,newPipe;isPickedUp:false;isDamaged:false;toPipeIdx:1
 ;fromPipeIdx:0
Pumpa lerakása sikertelen.
```

8.2.10 Person Move To Free

- **Leírás**
  Ez a teszteset a játékosok mozgását teszteli. A teszteset minden elemre való lépést teszteli, de egyelőre csak szabad elemek vannak.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - sivatagra meg forrásra nem tud lépni a játékos

  - minden más elemre sikeres a move függvény hívás
  - minden lépéskor a játékos element referenciája megváltozik (state player -> element)
- **Bemenet**
```
load PersonMoveToFree.dat debug
move player 0
move player 1
move player 1
move player 1
move player 0
move player 2
move player 1
```
- **Kimenet**
```
Pálya betöltése sikeres.
Új pozíció: pipe1
Új pozíció: pump
Új pozíció: pipe2
Új pozíció: pipe2
Új pozíció: pump
Új pozíció: pipe3
Új pozíció: waterTank
```

8.2.11 Person Move To Occupied

- **Leírás**
  Ez a teszteset a játékosok mozgását teszteli, de itt van foglalt elem is.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - foglalt csőre való lépéssel próbálkozás esetén sikertelen a move függvény hívás
  - a játékos element referenciája nem változhat
- **Bemenet**
```
load PersonMoveToOccupied.dat debug
move player 0
```
- **Elvárt kimenet**
```
Pálya betöltés sikeres.
Új pozíció: pump
```

8.2.12 Propagate Water

- **Leírás**
  Ez a teszteset azt teszteli, hogy egy teljesen ép pályán, ahol létezik út a forrás és a ciszterna között, a megfelelő módon áramlik-e a víz, valamint, hogy ennek megfelelően inkrementálódik a szerelők pontszáma.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - tickre meghívódik a propagateWater függvények láncolata
  - waterSource egy cső teljes kapacitásával megegyező mennyiségű vizet ad ki, ez a mennyiség érkezik meg a waterTankbe (state waterTank -> waterLevel=5)
  - a szerelők pontszáma is pontosan ennyivel növekszik (plumberPoints -> 5)
- **Bemenet**
```
load PropagateWater.dat debug
dbgTickAll
```

```
state waterTank
plumberPoints
```

- **Elvárt kimenet**

```
Pálya betöltése sikeres.
tickAll.
WaterTank waterTank neighbors:pipe1;isPickedUp:false;waterLevel:5
A szerelők pontszáma: 5.
```

8.2.13 Seep Water From Pipe End

- **Leírás**
  Ez a teszteset teszteli, hogy képes-e létrehozni új csövet a ciszterna és hogy ezen megfelelően kifolyik e a benne lévő víz, ha szabad a vége. Azt is ellenőrzi, hogy ennek a következményeképp helyesen változik-e a csapatok pontszáma.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - létrejön az új cső és a két szomszédja a waterTank és a desert (state newPipe -> neighbors:waterTank,desert)
  - ciszternában (ebben a tesztesetben) eredetileg 10 egységnyi víz van
  - tickre a ciszternától meghívódik a seepWater függvények láncolata és egy csőkapacitásnyi víz elfolyik
  - ciszternában már csak 5 egységnyi víz van (state waterTank -> waterLevel=5)
  - a szerelők pontszáma 10 helyett 5 lesz (plumberPoints -> 5)
  - a szabotőrök pontszáma 0 helyett 5 lesz (saboteurPoints -> 5)
- **Bemenet**

```
load SeepWaterFromPipeEnd.dat debug
dbgCreatePipe waterTank
state newPipe
state waterTank
plumberPoints
saboteurPoints
dbgTickAll
state waterTank
plumberPoints
saboteurPoints
```

- **Elvárt kimenet**

```
Pálya betöltése sikeres.
Cső létrehozva.
Pipe newPipe
 neighbors:waterTank,desert;isPickedUp:false;capacity:5;waterLevel:
 0;isDamaged:false;isSlippery:false;fixedTime:0;isSticky:false
WaterTank waterTank
 neighbors:newPipe;isPickedUp:false;waterLevel:10
A szerelők pontszáma: 10.
A szabotőrök pontszáma: 0.
TickAll.
WaterTank waterTank neighbors:newPipe;isPickedUp:false;waterLevel:5
A szerelők pontszáma: 5.
A szabotőrök pontszáma: 5.
```

8.2.14 Seep Water From Damaged Pipe

- **Leírás**
  Ez a teszteset a lyukas csövön történő vízáramoltatáshoz kapcsolódó működéseket ellenőrzi.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - a lyukas csőben nem maradhat víz, mindnek ki kell folynia (state pipe -> waterLevel = 0)
  - a cső másik végén 0 víz mehet ki (state waterTank -> waterLevel = 0)
  - a szerelők pontszáma nem nőhet emiatt a cső miatt (plumberPoints -> 0)
  - a szabotőrök pontszámának nőnie kell (saboteurPoints -> 5)
- **Bemenet**
```
load SeepWaterFromPipe.dat debug
dbgTickAll
state pipe
state waterTank
plumberPoints
sabouteurPoints
```
- **Elvárt kimenet**
```
Pálya betöltése sikeres.
TickAll.
Pipe pipe neighbors:waterSource,waterTank;isPickedUp:false;capacity
 :5;waterLevel:0;isDamaged:true;isSlippery:false;fixedTime:0;isStic
 ky:false
WaterTank waterTank neighbors:pipe;isPickedUp:false;waterLevel:0
A szerelők pontszáma: 0.
A szabotőrök pontszáma 5.
```

8.2.15 Redirect Pump

- **Leírás**
  Ez a teszteset a pumpák átirányítását teszteli. Megpróbál egy megengedett átállítást, illetve egy nem megengedettet végrehajtani.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - megengedett átállítás: az új forrástól fogadjon vizet, a régitől ne; az új kimenetre továbbítsa a vizet, a régire ne
  - nem megengedett átállítás: ne engedje ugyanazt a csövet be és kimenetnek is megadni
- **Bemenet**
```
load RedirectPump.dat debug
redirect playerOnPump damagedPipe endPipe1
dbgTickAll
state waterTank2
redirect playerOnPump goodPipe endPipe2
dbgTickAll
state waterTank1
state waterTank2
redirect playerOnPump goodPipe goodPipe
```
- **Elvárt kimenet**
```
Pálya betöltése sikeres.
Az átirányítás sikeres.
TickAll.
```

[FF1] megjegyzést írt: Ide még be lehetne rakni h minden atallitas utan irja ki a pumpa allapotat is, h lassuk h tenyleg atirta e a fromPipe, toPipeot

[AGP2R1] megjegyzést írt: Arra nem elég az, hogy ugye kiírja, hogy sikeres/sikertelen, aztán a működésből látjuk, hogy tényleg az történik-e, amit akarunk?

[FF3R1] megjegyzést írt: Dee, vegulis

```
WaterTank waterTank1
 neighbors:endPipe1;isPickedUp:false;waterLevel:0
Az átirányítás sikeres.
TickAll.
WaterTank waterTank1
 neighbors:endPipe1;isPickedUp:false;waterLevel:0
WaterTank waterTank2
 neighbors:endPipe2;isPickedUp:false;waterLevel:5
Az átirányítás sikertelen.
```

8.2.16 Make Sticky

- **Leírás**
  Ez a teszteset a csövek ragadóssá tételét teszteli. Egy játékos ragadóssá teszi a csövet, és lelép róla. Ezután a következő játékos rálép a csőre, és megnézzük, hogy rajta ragad-e. Ezen kívül azt is ellenőrizzük, hogy pumpát ragadóssá tud-e tenni a játékos.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - A játékos, aki ragadóssá tette a csövet, el tud-e lépni a csőről (azt várjuk, hogy igen).
  - A következő játékos rajta ragad-e a csövön (azt várjuk, hogy igen).
  - A cső isSticky változója true lesz-e (azt várjuk, hogy igen).
  - A pumpát ragadóssá lehet-e tenni (azt várjuk, hogy nem).
- **Bemenet**

```
load MakeSticky.dat debug
makeSticky saboteur
state pipe
move saboteur 0
move victim 0
move victim 1
makeSticky saboteur
```

- **Elvárt kimenet**

```
Pálya betöltése sikeres.
Ragadóssá tevés sikeres.
Pipe pipe
 neighbors:pump1,pump2;isPickedUp:false;capacity:5;waterLevel:0;isD
 amaged:false;isSlippery:false;fixedTime:0;isSticky:true
Új pozíció: pump1.
Új pozíció: pipe.
Új pozíció: pipe.
Ragadóssá tevés sikertelen.
```

8.2.17 Make Slippery

- **Leírás**
  Ez a teszteset a csövek csúszóssá tételét teszteli. Egy játékos csúszóssá teszi a csövet, és lelép róla. Ezután a következő játékos rálép a csőre, és megnézzük, hogy átcsúszik-e. Ezen kívül azt is ellenőrizzük, hogy pumpát csúszóssá tud-e tenni a játékos.
  A kiinduló állapotban a saboteur a pipe-on áll, majd onnan lép a pump1-re. A victim a pump2-ről indul. Először megpróbál ennek a 0. szomszédjára (pipe) lépni, ekkor

csúszik a pump1-re. Ezután megpróbál a pump1 1. szomszédjára lépni, ami szintén a pipe, ami csúszós, emiatt annak a 0. szomszédjára (mert debug módban vagyunk, nem randomban) csúszik, tehát marad a pump1-en.

- **Ellenőrzött funkcionalitás, várható hibahelyek**
  - A következő játékos átcsúszik-e a csövön (azt várjuk, hogy igen), vagyis mivel debug módban vagyunk, mindig a cső 0. szomszédjára kerül-e a cső helyett.
  - A cső isSlippery változója true lesz-e (azt várjuk, hogy igen).
  - A pumpát csúszóssá lehet-e tenni (azt várjuk, hogy nem).
- **Bemenet**

```
load MakeSlippery.dat debug
makeSlippery saboteur
state pipe
move saboteur 0
move victim 0
move victim 1
makeSticky saboteur
```

- **Elvárt kimenet**

```
Pálya betöltése sikeres.
Csúszóssá tevés sikeres.
Pipe pipe
 neighbors:pump1,pump2;isPickedUp:false;capacity:5;waterLevel:0;isD
 amaged:false;isSlippery:true;fixedTime:0;isSticky:false
Új pozíció: pump1.
Új pozíció: pump1.
Új pozíció: pump1.
Csúszóssá tevés sikertelen.
```

[FF4] megjegyzést írt: Itt lehet h erdemes lenne kiiratni annak a mezonek az adatait, amin eppen a victim all, h erthetobb legyen h kinek ki hanyadik szomszedja, min hivja a moveot es ez alapjan hova kerul

[AGP5R4] megjegyzést írt: Ezt inkább a leírásba írtam le, mert a state kiírásokból szerinten nehéz lenne kiigazodni rajta.

[FF6R4] megjegyzést írt: Oksi

8.3 A tesztelést támogató programok tervei

A 8.2 fejezetben definiált tesztek hatékony futtatása a következő módon végezhető el:

1. Fordítsuk le a programot a 6.1.2 fejezet leírása alapján, az `src` mappából indulva (tehát a parancsokat az `src` mappában adjuk ki).
2. Ezután a parancssorban lépjünk vissza a szülőmappába. Pl.: `cd ..` paranccsal.
3. Adjuk ki a `type input\inputX.txt | java -cp .\src; Main > result\resultX.txt` parancsot, ahol `X` a futtatandó teszteset (8.2.X fejezet) száma. Ekkor elkészül a program kimenetét tartalmazó resultX.txt fájl.
4. A kapott és a várt kimenet összehasonlítására használjuk a `fc result\resultX.txt expected\expectedX.txt` parancsot, ahol `X` az előző (3.) lépésben futtatott teszt száma.

8.4 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2023.04.26. 12:00	0,5 óra	Andrási, Bertók, Farkas, Mayer, Nás	Értekezlet. Döntések: - 8.1: - első fele: Bertók - második fele: Farkas - 8.2: - eleje: Nás - közepe: Mayer - vége: Andrási - 8.3: Andrási Határidő: 2023.05.03.
2023.04.27. 17:45	0,75 óra	Andrási	8.3
2023.04.27. 22:00	3 óra	Andrási	8.2.16, 8.2.17, 8.2.18, 8.2.19
2023.04.28. 11:00	0,5 óra	Nás	8.2.3, 8.2.4
2023.04.28. 15:00	2 óra	Farkas	8.1. második fele
2023.04.30. 11:00	1 óra	Nás	8.2.5, 8.2.6, 8.2.7, 8.2.8
2023.05.02. 8:00	0,75 óra	Nás	8.2.9, 8.2.11
2023.05.02. 13:00	1 óra	Mayer	8.2.12, 8.2.13, 8.2.14
2023.05.02. 16:00	0.5 óra	Nás	8.2.9 javítások + 8.2.10
2023.05.02. 20:00	0.5 óra	Andrási	8.2.17, 8.2.19 javítások
2023.05.02. 18:00	1 óra	Farkas	8.1