**8. Részletes tervek**

30 – ez\_mode

Konzulens:

Koppa Dániel

Csapattagok

|  |  |  |
| --- | --- | --- |
| Nyilas Péter | Y8QBCS | [nyilaspeter001@gmail.com](mailto:nyilaspeter001@gmail.com) |
| Farkas Bence | D6AE9F | [bencefarkas27@gmail.com](mailto:bencefarkas27@gmail.com) |
| Zsigmond Botond | MV7M6D | [zsigmond.botond.9@gmail.com](mailto:zsigmond.botond.9@gmail.com) |
| Sőregi Larina | M4B9MC | [soregi.larina2001@gmail.com](mailto:soregi.larina2001@gmail.com) |
| Kis Benedek M. | JOYAXJ | [kisbeni2003@gmail.com](mailto:kisbeni2003@gmail.com) |

2023.05.02

# Részletes tervek

## Osztályok és metódusok tervei.

### Character

#### Felelősség

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

#### Ősosztályok

Nincs ősosztálya.

#### Interfészek

Tickable interfészt implementálja.

#### Attribútumok

* **name**: karakter neve
* **uuid**: karakter egyedi azonosítója
* **standingOn**: eltárolja, hogy a karakter a Map mely elemén, melyik Node-on áll.
* **stuckedInPipe**: random generált idő, ameddig a karakter nem léphet el a csőről ha az ragacsos.

#### Metódusok

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

### Cistern

#### Felelősség

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

#### Ősosztályok

Node → Cistern

#### Interfészek

Tickable interfészt implementálja.

#### Attribútumok *A felsoroltokon kívül rendelkezik még a Node absztrakt ősosztály attribútumaival.*

* **prodicedPumps:** A ciszterna maga által készített pumpákat tárolja.

#### Metódusok

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

### Map

#### Felelősség

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

#### Ősosztályok

Nincs ősosztály

#### Interfészek

Tickable interfészt implementálja.

#### Attribútumok

* **waterArrived:** Nyilvántartja, hogy mennyi víz folyt be a ciszternába.
* **waterLost:** Nyilvántartja, hogy mennyi víz folyt el a lyukakon a sivatagi homokra.
* **players:** A pályán levő játékos Character-ek.
* **gameMap:** A pálya elemeit tartalmazó lista.

#### Metódusok

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

### Node

#### Felelősség

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

#### Ősosztályok

Nincs ősosztály.

#### Interfészek

Tickable interfészt implementálja.

#### Attribútumok

* **logger:** egy logger ehhez az osztályhoz.
* **uuid:** a node egyedi azonosítója.
* **characters:** a jelenleg ezen az nodeon álló karakterek listája.
* **neighbours:** a jelenleg ezzel a node-dal szomszédos és a node-ba csatlakoztatott más nodeok listája.
* **sources:** azon nodeok, ahonnan folyik víz ebbe a nodeba.
* **absorbers:** azon nodeok, ahová folyik víz ebből a nodeba.
* **maxCharacters:** az ezen a nodeon a maximálisan elfogadott karakterek száma.
* **maxConnections:** az ehhez a nodehoz maximálisan csatlakoztatható elemek száma.
* **isBroken:** A node állapota, műkódik e vagy sem.
* **flowRate:** hogy éppen mennyi víz folyik keresztül a nodeon.
* **x:** Node x koordináltja a pályán.
* **y:** Node y koordináltja a pályán.

#### Metódusok

* **String getUuid()**: visszaadja az uuid attribútum értékét.
* **ArrayList<Character> getCharactersOn():** Visszaadja a nodeon álló karakterke listáját.
* **ArrayList<Node> getNeighbours():** Visszaadja a node-hoz kötött nodeok listáját.
* **void placeCharacter(Character character):** elhelyez egy karaktert a nodeon. Játék inicializáláakor használatos, vagy ha a játékos alól eltűnt elem miatt vissza kell helyezni a kezdő nodera.
* **void addCharacter(Character character):** Hozzáad egy karaktert a characters listához, ha még fér karakter a nodera.
  + if (size of characters >= macCharacters)
    - than throw exception;
  + else while(iterate throught neighbours)
    - if (a neighbour contains character) than add character to characters
  + end
* **void removeCharacter(Character character):** kivesz egy karaktert a characters listából.
  + if(characters contains character)
    - than remove character from characters
* ***void repairNode(Character character):*** Megjavítja az objektumot. Absztrakt, leszármazottak implementálják.
* ***void breakNode(Character character):*** Elrontja az objektumot. Absztrakt, leszármazottak implementálják.
* **void addFlowRate(Node source, double excededFlow):** Növeli a flowRate-et a megadott értékkel.
  + if(sources contains source)
    - than flowRate += excededFlow;
    - while(iterate throught absorbers)
      * addFlowRate();
      * end
  + endif
* **void removeFlowRate(Node source, double flowRate):** Csökkenti a flowRate-et a megadott étékkel.
  + if(sources contains source)
    - than flowRate -= flowRate;
    - while(iterate throught absorbers)
      * removeFlowRate();
      * end
  + endif
* **void setFlowRate(double flowRate):** Beállítja a flowRate-et a megadott értékre.
* **void tick():** Időközönként, azaz körönként számolja a flowRate-et.
  + calculateFlowRate();
* **void calculateFlowRate():** Számolja a Nodeon átfolyó vít mennyiségét.
  + if(Node is broken)
    - than Map waterLost += Node flowRate;
    - while(iterate througth absorber nodes)
      * absotber node removeFlowRate(Node, Node flowRate);
      * end
  + else
    - while(iterate througth absorber nodes)
      * absotber node addFlowRate(Node, Node flowRate);
      * end
* **void connect(Node node2):** Hozzácsatlakoztatja a megadott Nodeot.
  + node1 neighbours add node2;
  + if(node2 neigbours not contains node1)
    - than node2 connect(node1);
* **void disconnect(Node node):** Lecsatlakoztatja a megadott Nodeot.
  + node1 neighbours remove node2;
  + if(node2 neigbours contains node1)
    - than node2 disconnect(node1);
* ***void setSurface(string surface, Character c):*** Absztrakt metódus amit a leszármazottak implementálnak. Célja, hogy a karakterek tudják ragacsossá/csúszóssá teni a pályaelemek egy részét (pipe).

### Nomad

#### Felelősség

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

#### Ősosztályok

Character → Nomad .

#### Interfészek

Tickable interfészt implementálja.

#### Attribútumok

* **name**: karakter neve
* **uuid**: karakter egyedi azonosítója
* **standingOn**: eltárolja, hogy a karakter a Map mely elemén, melyik Node-on áll.
* **stuckedInPipe**: random generált idő, ameddig a karakter nem léphet el a csőről ha az ragacsos.

#### Metódusok

* **void setPump(Pipe in, Pipe out)**: A nomád beállítja a Pumpán, amin áll. A Pumpához csatlakoztatott csövek közül melyiket nyitja meg, hogy a pumpába ki és be- folyjon a víz.
  + if(in is neighbour of standingOn)
    - than setActiveInput(in);
  + if(out is neighbour of standingOn)
    - setActiveOutput(out);
* **void makePipeSlippery()**:A nomád csúszóssá teszi a cső felszínét.
  + standingOn setSurface(„slippery”, Nomad);

### Plumber

#### Felelősség

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

#### Ősosztályok

Character → Plumber

#### Interfészek

Tickable interfészt implementálja.

#### Attribútumok

* **name**: karakter neve
* **uuid**: karakter egyedi azonosítója
* **standingOn**: eltárolja, hogy a karakter a Map mely elemén, melyik Node-on áll.
* **stuckedInPipe**: random generált idő, ameddig a karakter nem léphet el a csőről ha az ragacsos.
* **pickedUpPump:** A plumber inventoryában a ciszternán felvett pumpát tárolja.
* **pickedUpPipe:** A plumber inventoryában a mindkét végén lecsatlakoztatott majd felvett csöveket tárolja.
* **draggedPipe:** Az egyik végén lecsatlakoztatott, és átmozgatott cső.

#### Metódusok

* **void repair()**: megjavítja a Node-ot, amin áll.
  + standingOn repairNode(Plumber);
* **void placePump()**: elhelyez egy pumpát oda, ahol áll. Ha csövön áll sikeresen lehelyezi, ha més nodeon akkor kivételt dob.
  + if(has a pickedPump)
    - than Map addNode(pickedPump);
    - create Pipe newPipe;
    - Map addNode(newPipe);
    - newPipe connect(pickedUp);
    - standingOn connect(pickedUp); endif
* **void placePipe()**: Lehelyezi egy csövet a pályára, és hozzáköti ahhoz a nodehoz amin áll.
  + if(has draggedPipe)
    - than standingOn connect(draggedPipe);
    - draggedPipe = null;
    - return;
  + else if(has pickedUpPipe)
    - than standingOn connect(pickedUpPipe);
    - Map addNode(pickedUpPipe)
    - pickedUpPipe = null;
    - return;
* **void pickupPump():** Megpróbál felvenni egy pumpát. Ha nem ciszternán áll akkor kivételt dob.
  + Pump P = standingOn givePump();
  + if(P is not null)
    - than pickedUpPump = P;
  + endif
* **void pickupPipe():** Lecsatlakoztatja a cső egyik végét és elhúzza egy másik elemhez. Ha a cső mindkét vége disconnected akkor felveszi az egészet az inventoryába.
  + while(iterate throught neighbours of standingOn)
    - if(standingOn has at least2 neighbours)
      * than draggedPipe = iterated Node;
      * break;
      * endif
  + end
  + if(draggedPipe neighbours is empty)
    - than pickedUpPipe = draggedPipe;
    - draggedPipe = null;
  + endif
* **void setPump(Pipe in, Pipe out)**: A plumber beállítja a Pumpán, amin áll. A Pumpához csatlakoztatott csövek közül melyiket nyitja meg, hogy a pumpába ki és be- folyjon a víz.
  + if(in is neighbour of standingOn)
    - than setActiveInput(in);
  + if(out is neighbour of standingOn)
    - setActiveOutput(out);

### Pipe

#### Felelősség

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

#### Ősosztályok

Node → Pipe

#### Interfészek

Tickable interfészt implementálja.

#### Attribútumok: *A felsoroltokon kívül rendelkezik még a Node absztrakt ősosztály attribútumaival.*

* **capacity**: Azt mutatja mennyi a kapacitása ennek a csőnek (vagyis azt, mennyi víz fér bele).
* **isSlippery:** Csúszós-e a cső felszíne.
* **isStickey:** Ragacsos-e a cső felszíne.
* **unbreakableTill:** Cső megfoltozásakor generált véletlen szám, ameddig nem lehet újra kilyukasztani.

#### Metódusok

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

### Pump

#### Felelősség

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

#### Ősosztályok

Node → Pump

#### Interfészek

Tickable interfészt implementálja.

#### Attribútumok: *A felsoroltokon kívül rendelkezik még a Node absztrakt ősosztály attribútumaival.*

* **internalBufferLevel**: ezt arra használjuk, hogy a flowRate-et számoljuk.
* **activeInput**: ez a pumpa bemenetei közül tárolja azt a csövet, amelyik aktív.
* **activeOutput**: a pumpa kimenetei közül tárolja azt a csövet, amelyik aktív.

#### Metódusok

* **void repairNode(Character character)**: Megjavítja a pumpát.
  + if(Pump is broken)
    - than isBroken = false;
* **void breakNode(Character character)**: Elrontja a pumpát.
  + if(Pump is not Broken)
    - than isBroken = true;
* **void setActiveInput(Pipe p)**: Beállítja az activeInput-ot, azaz hogy, a szomszédjai közül a megadott csőből folyjon be víz a pumpába.
  + activeInput = p;
* **void setActiveOutput(Pipe p)**: Beállítja az activeOutput-ot azaz hogy, a szomszédjai közül a megadott csőbe folyjon ki víz a pumpából.
  + activeOutput = p;
* **void calculateFlowRate()**: Számolja a flowRatet
  + if(Pump is not broken)
    - than if(absorbers conatins activeInput)
      * than if(absorbers conatins activeOutput)
        + setFlowRate(activeInput flowRate);
        + activeOutput += flowRate;
      * else internalBufferLevel += activeInput flowRate;
    - else if(internalBufferLevel > 0)
      * than setFlowRate(min{internalBufferLevel, activeInput capacity});
      * activeOutput flowRate += flowRate;
  + else
    - setFlowRate(0);
    - internalBufferLevel += activeInput flowRate;
* **void tick()**: Időközönként (random) elrontja a pupmpát, és számolja a flowRate-et.
  + calculateFlowRate();
  + if(random int 1) breakNode();

### Tickable

#### Felelősség

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

#### Ősosztályok

Nincsenek ősosztályok.

#### Metódusok

* ***void tick()***: Az adott időközönként végzett tevékenységre való megvalósítandó metódus.

### WaterSpring

#### Felelősség

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

#### Ősosztályok

Node → WaterSpring

#### Interfészek

Tickable interfészt implementálja.

#### Attribútumok: *A Node absztrakt ősosztály attribútumaival rendelkezik.*

#### Metódusok

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

## A tesztek részletes tervei, leírásuk a teszt nyelvén

### Pálya kiírása

* **Leírás**

A program betölti és kirajzolja a konzolra a játékpályát.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

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

* **Bemenet**

*load*

* **Elvárt kimenet**

*Successfully loaded map.*

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

### Játékos hozzáadása

* **Leírás**

A program fölvesz egy új játékost a pályára.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

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

* **Bemenet**

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

* **Elvárt kimenet**

*Successfully added character.*

### Új pumpaelem felvétele

* **Leírás**

A szerelő magához vesz egy új pumpát a ciszternáktól.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

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

* **Bemenet**

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

* **Elvárt kimenet**

*Successfully added pump.*

### Új csőelem felvétele

* **Leírás**

A szerelő magához vesz egy új csövet a ciszternáktól.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

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

* **Bemenet**

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

* **Elvárt kimenet**

*Successfully added pipe.*

### Pumpa elhelyezése

* **Leírás**

A szerelő új pumpát telepít a hálózatra.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

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

* **Bemenet**

*character ⏎ place ⏎ pump*

* **Elvárt kimenet**

*Successfully deployed pump.*

### Cső áthelyezése

* **Leírás**

Adott cső bekötése a hálózatba.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

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

* **Bemenet**

*character ⏎ place ⏎ pipe*

* **Elvárt kimenet**

*Successfully placed pipe.*

### Karakter mozgatás

* **Leírás**

A karakterek mozgatása egyik node-ról a másikra.

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

*character ⏎ move*

* **Elvárt kimenet**

*Successfully moved character.*

### Cső lyukasztása

* **Leírás**

Adott karakter kilyukasztja a csövet, amin éppen áll.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

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

* **Bemenet**

*character ⏎ break*

* **Elvárt kimenet**

*Successfully broken pipe.*

### Pumpa elállítása

* **Leírás**

Adott karakter elállítja a pumpát, amin éppen áll.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

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

* **Bemenet**

*character ⏎ set*

* **Elvárt kimenet**

*Successfully set pump.*

### Pumpa javítása

* **Leírás**

A szerelő megjavítja a pumpát, amin áll.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

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

* **Bemenet**

*character ⏎ repair*

* **Elvárt kimenet**

*Successfully repaired pump.*

### Cső javítása

* **Leírás**

A szerelő megjavítja a csövet, amin áll.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

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

* **Bemenet**

*character ⏎ repair*

* **Elvárt kimenet**

*Successfully repaired pipe.*

### Cső csúszóssá tétele

* **Leírás**

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

* **Ellenőrzött funkcionalitás, várható hibahelyek**

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

* **Bemenet**

*character ⏎ slippery*

* **Elvárt kimenet**

*Successfully made pipe slippery.*

*Slippery when wet.*

### Cső ragadóssá tétele

* **Leírás**

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

* **Ellenőrzött funkcionalitás, várható hibahelyek**

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

* **Bemenet**

*character ⏎ sticky*

* **Elvárt kimenet**

*Successfully made pipe sticky.*

## A tesztelést támogató programok tervei

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

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2023.04.26. 12:00 | 10 perc | Farkas  Kis  Sőregi  Nyilas | Értekezlet.  Heti munka felosztása |
| 2023.04.26. 12:15 | 2,5 óra | Farkas | Osztályok, metódusok dokumentálása. |
| 2023. 04. 26. 13:30 | 30 perc | Nyilas, Zsigmond | Tesztesetekről szóló értekezlet. |
| 2023. 04. 26. 14:00 | 3 óra | Nyilas | Tesztesetek kidolgozása. |
| 2023. 04. 26. 14:00 | 3 óra | Zsigmond | Tesztesetek dokumentálása. |
| 2023 04.29 11:00 | 1,5 óra | Farkas | Bonyolultabb, hosszabb metódusok pszeudo-kódos leírása |
| 2023. 04. 29. 15:30 | 1,5 óra | Zsigmond | Pszeudo-kódos leírások befejezése, néhány előzőleg megírt javítása |
| 2023. 05. 02 19:00 | 1 óra | Kis  Sőregi  Nyilas  Farkas | Záró értekezlet |