

8. Részletes tervek

66 – *[simon_balazst_szeretnenk_konzulensnek]*

Konzulens:

Simon Balázs

Csapattagok:

Kiss Andor	TXC54G	kissandor4@gmail.com
Konrád Márk	JSPDME	konrad0816@gmail.com
Glávits Balázs Róbert	NMZC9G	glavits.balazs@gmail.com
Máté Botond	ELOYOV	m.botond7@gmail.com
Lant Gábor	P35E36	lant.gabor98@gmail.com

2020. április 13.

Tartalomjegyzék

8	Részletes tervek	5
8.1	Osztályok és metódusok tervei	5
8.2	Osztályok leírása	5
8.2.1	BareHands	5
8.2.2	BareIce	5
8.2.3	BuildStrategy	5
8.2.4	BreakingShovel	5
8.2.5	BreakingShovelDig	6
8.2.6	CantRescue	6
8.2.7	ChillWaterStrategy	6
8.2.8	DigStrategy	6
8.2.9	DryLand	6
8.2.10	Empty	7
8.2.11	Entity	7
8.2.12	Eskimo	7
8.2.13	Food	7
8.2.14	FoodStore	8
8.2.15	Game	8
8.2.16	Igloo	8
8.2.17	Item	9
8.2.18	Naked	9
8.2.19	Part	9
8.2.20	PartStore	9
8.2.21	Player	10
8.2.22	PolarBear	11
8.2.23	PolarExplorer	11
8.2.24	RescueStrategy	11
8.2.25	Rope	11
8.2.26	RopeRescue	11
8.2.27	ScubaGear	12
8.2.28	ScubaWearing	12
8.2.29	Sea	12
8.2.30	Shovel	12
8.2.31	ShovelDig	12
8.2.32	Tent	13
8.2.33	TentKit	13
8.2.34	Tile	13
8.2.35	WaterResistanceStrategy	14
8.2.36	Proto	14
8.2.37	MessagePrinter	15
8.2.38	Command	16
8.2.39	CommandParser	16
8.2.40	TileCommand	16
8.2.41	TileCommandParser	17
8.2.42	BuildingCommand	17
8.2.43	BuildingCommandParser	17

8.2.44	ItemCommand	18
8.2.45	ItemCommandParser	19
8.2.46	EquipCommand	19
8.2.47	EquipCommandParser	20
8.2.48	SelectCommand	20
8.2.49	SelectCommandParser	21
8.2.50	EntityCommand	21
8.2.51	EntityCommandParser	22
8.2.52	ConnectCommand	22
8.2.53	ConnectCommandParser	23
8.2.54	StepCommand	23
8.2.55	StepCommandParser	24
8.2.56	RescueCommand	24
8.2.57	RescueCommandParser	24
8.2.58	ExamineCommand	25
8.2.59	ExamineCommandParser	25
8.2.60	DigCommand	25
8.2.61	DigCommandParser	26
8.2.62	PickUpCommand	26
8.2.63	PickUpCommandParser	26
8.2.64	BuildCommand	26
8.2.65	BuildCommandParser	27
8.2.66	AssembleCommand	27
8.2.67	AssembleCommandParser	27
8.2.68	EatCommand	28
8.2.69	EatCommandParser	28
8.2.70	TurnCommand	28
8.2.71	TurnCommandParser	28
8.2.72	StormCommand	29
8.2.73	StormCommandParser	29
8.2.74	QueryCommand	29
8.2.75	QueryCommandParser	31
8.2.76	Osztály1	31
8.2.77	Osztály2	32
8.3	A tesztek részletes tervei, leírásuk a teszt nyelven	32
8.3.1	Teszteset1	32
8.3.2	Teszteset2	32
8.4	A tesztelést támogató programok tervei	33
8.5	Napló	33

Ábrák jegyzéke

8. Részletes tervek

8.1. Osztályok és metódusok tervei

8.2. Osztályok leírása

8.2.1. BareHands

- A játékos így ás, ha nincs ásója. A kiválasztott cellán csökkennie kell a hó mennyiségnek ásáskor.
- Interfészek:
 - DigStrategy
- Metódusok:
 - bool Dig(Tile t): Csökkenti a tile-on található hó mennyiségét. Minden alkalommal fárasztó az ásás, ezért a visszatérési érték mindig true.

8.2.2. BareIce

- Ilyen a jégtábla, ha nincs rajta iglu. A jégtáblán nincs védelem a vihar elől.
- Ősosztályok:
 - Shelter
- Metódusok:
 - void ChillStorm(Tile t): A paraméterként kapott t Tilen álló játékosok testhője csökken.
 - void BearAttack(Tile t):
 - void Break(Tile t):

8.2.3. BuildStrategy

- A játékos így képes építeni. Iglut vagy sátrat.
- Attribútumok:
 - count: int:
- Metódusok:
 - void Build(): Épít valamit a táblára.
 - void Gain(): Kap egy sátrat.

8.2.4. BreakingShovel

- Törhető ásó osztály.
- Interfészek:
 - Item
- Metódusok:
 - void GiveTo(Player p): A játékos így kap ásót.

8.2.5. BreakingShovelDig

- A játékos így ás, ha törhető ásó van nála.
- Interfészek:
 - DigStrategy
- Attribútumok:
 - lastUsed: bool: Volt-e használva a körben.
 - durability: int: Mennyiszer lehet még ásni vele.
- Metódusok:
 - bool Dig(Tile t): Csökkenti a tile-on található hó mennyiségét.

8.2.6. CantRescue

- A játékos nem tudja kihúzni a csapattársát. A játékos ilyen állapotban van, ha nincs nála kötél.
- Interfészek:
 - RescueStrategy
- Metódusok:
 - void Rescue(Tile water, Tile land): Mivel a játékos ebben az állapotban nem tudja megmenteni a csapattársát, ez a fv nem csinál vele semmit.

8.2.7. ChillWaterStrategy

- A jégtábla így hűti a vízbe esett játékosokat. Vízben tartózkodás esetén a játékos testhője csökken, a megvalósított stratégia alapján.
- Metódusok:
 - abstract void Chill(Tile t): A stratégiát megvalósító elem dolga implementálni mi történik.

8.2.8. DigStrategy

- A játékos így ás. Ásáskor a cellán a hó mennyiség csökken.
- Metódusok:
 - abstract bool Dig(Tile t): A stratégiát megvalósító elem dolga implementálni mi történik ásáskor. Visszaadja, hogy az ásás fárasztó-e.

8.2.9. DryLand

- A szárazföld nem hűti a játékosokat. A játékos nincsen vízben.
- Interfészek:
 - ChillWaterStrategy
- Metódusok:
 - void Chill(Tile t): A stratégia megvalósítása miatt kér be egy t Tile paramétert, a rajta levő játékosal viszont nem csinál semmit, mert az nincs vízben, nem csökkenti testhőjét.

8.2.10. Empty

- Nincs jégbe fagyott tárgy. Ez az üres eszköz típus, nem képes semmi extra tulajdonságot biztosítani a tulajdonosnak.
- Interfészek:
 - Item
- Metódusok
 - void GiveTo(Player p): A paraméterként kapott játékost nem ruházza fel extra tulajdonsággal, mivel épp nincs itt jégbe fagyott tárgy.

8.2.11. Entity

- Entitás osztály ami a pályát tartózkodhat.
- Metódusok:
 - Step(Direction d): Lép valamilyen irányba.
 - void PlaceOn(Tile t): Ráteszi az entitást egy másik táblára. A kötél használatakor használatos.
 - void Chill(): Hűti az entitést. A testhője csökken.
 - ResistWater(): Így viselkedik vízben.
 - BearAttack(): Így viselkedik, ha megtámadja a medve.

8.2.12. Eskimo

- Játékos fajta. 5 egységnyi testhővel kezd. Képes iglut építeni. A játékos irányítja.
- Ősosztályok:
 - Player
- Metódusok:
 - void BuildIgloo(): Épít egy iglut a mezőre, amin áll. Az iglu megvéd majd a hóvihartól. Beállítja a mező hóvihar stratégiáját Iglusra.

8.2.13. Food

- Élelem, amit a játékos meg tud enni, hogy növelje a testhőjét. Élelem a pályán lesz található.
- Interfészek:
 - Item
- Metódusok:
 - void GiveTo(Player p): A paraméterként kapott játékos kap egy élelmet, az bekerül az élelemtárolójába.

8.2.14. FoodStore

- A játékos ebben a zsebben tárolja az élelmet.
- Attribútumok:
 - count: int: Hány élelem van a játékosnál.
- Metódusok:
 - void feed(Player p): Játékos testhője megnő, az élelem mennyisége csökken, mivel a játékos meg-
eszi azt.
 - void DecrementCount(): Csökkenti a benne található elemek számát.
 - void Gain(): növeli a benne található elemek számát.

8.2.15. Game

- Interface a Model és a Controller között. A játékmesterhez tartozó működést valósítja meg. Felelős a játékban lévő objektumok tárolásáért és létrehozásáért.
- Attribútumok:
 - players: Player[3..*]: Tárolja a játékosokat.
 - icefield: Tile[1..*]: Tárolja a pályát alkotó elemeket.
 - bears: PolarBear[*]: Tárolja a medvé(ke)t, ha több van akkor is.
- Metódusok:
 - void AddTile(t: Tile): Hozzáad egy cellát a játékhoz.
 - void AddPlayer(pl: Player): Hozzáad egy játékost a játékhoz.
 - Tile CreateIce(): Létrehoz egy jégtáblát. Ez a metódus az init szekvencia része.
 - Tile CreateUnstableIce(): Létrehoz egy instabil jégtáblát. Ez a metódus az init szekvencia része.
 - Tile CreateSea(): Létrehoz egy vizet. Ez a metódus az init szekvencia része.
 - Tile CreateHole(): Létrehoz egy lyukat: olyan vizet amit hó fed. Ez a metódus az init szekvencia része.
 - Player CreateEskimo(): Létrehoz egy eszkimó játékost. Ez a metódus az init szekvencia része.
 - PolarBear CreatePolarBear(): Létrehoz egy medvét. Ez a metódus az init szekvencia része.
 - Player CreatePolarExplorer(): Létrehoz egy sarkkutató játékost. Ez a metódus az init szekvencia része.
 - void GameOver(): Ha vége a játéknak, szól a Controllernek, hogy veszítettünk. Külső metódus.
 - void Turn(): Ezt a metódust a Controller hívja körönként, a körök vezénylésére szolgál.
 - void Victory(): Ha vége a játéknak, szól a Controllernek, hogy nyertünk. Külső metódus.

8.2.16. Igloo

- Ezen a jégtáblán iglu áll, a játékosok védve vannak a vihartól. Az ilyen táblán nem csökken a viharban a rajta állók testhője.
- Ősosztályok:
 - Shelter

- Metódusok:
 - void ChillStorm(Tile t): A paraméterként kapott cellán álló játékosok testhője nem csökken, mivel igluban vannak.
 - void BearAttack(Tile t): Így viselkedik a mező ha valaki igluban van és megtámadja a medve.

8.2.17. Item

- Tárgy, a játékos képes ilyeneket felvenni a cellákról. A tárgyak képesek a játékosak képességeket adni. A tárgyak alapvetően jégbe fagyva vannak a pályán.
- Metódusok:
 - void GiveTo(p: Player): A játékos kap valamilyen tárgyat, az Item interfészt megvalósító tárgyak felüldefiniálják ezt.

8.2.18. Naked

- A játékos védtelen a hideg vízzel szemben. A játékos ha így esik vízbe és nem menekítik ki megfullad.
- Interfészek:
 - WaterResistanceStrategy
- Metódusok:
 - void Chill(Player p): Játékosnak nincsen ereje a vízben úszni bűvárruha nélkül.

8.2.19. Part

- Jégbefagyott alkatrész. Csak akkor ásható ki, ha nincs rajta hó.
- Interfészek:
 - Item
- Metódusok:
 - void GiveTo(Player p): A játékos tárolójába kerül egy darab a rakétapisztolyból.

8.2.20. PartStore

- A játékos ebben a zsebben tárolja az alkatrészeket.
- Attribútumok:
 - count: int: Tárolja hány darab alkatrész van belőle a játékosnál.
- Metódusok:
 - void Gain(PartStore ps): Átveszi az alkatrészeket a paraméterként kapott alkatrésztárolóból.
 - void Gain(int n): Megnö az alkatrészek száma, ami a játékosnál van.

8.2.21. Player

- Játékos osztály, amit a felhasználó irányít a grafikus felületen keresztül. Ilyen típussal nem lehet játszani, csak a leszármazottakkal. Felelőssége a játékos által a controlleren keresztül kiadott műveletek elvégzése. Tárolja a játékos jelenlegi állapotát.
- Ősosztályok:
 - Entity
- Attribútumok:
 - bodyTemp: int: Jelzi a játékos jelenlegi hőmérsékletét, ha 0 akkor megfagy → játék vége.
 - currentTile: Tile: A játékos ismeri a mezőt amin éppen áll.
 - inventory: Item[*]: Tárolja a játékos tárgyait, amik képességekkel tudnak felruházni őt.
 - digStrategy: DigStrategy: Eldönti hogyan képes ásni a játékos.
 - energy: int: Számlálja mennyit mozogott az adott körben a játékos.
 - foodStore: FoodStore: Tárolja a játékos ételeit.
 - game: Game: A játékos ismeri a játékot.
 - partStore: PartStore: Tárolja a játékos rakéta alkatrészeit.
 - rescueStrategy: RescueStrategy: Eldönti, hogy megmenthet egy játékos egy másikat a vízbeesés után.
 - waterResistanceStrategy: WaterResistanceStrategy: Eldönti, hogy a játékos hogyan viselkedik vízbeesés esetén.
- Metódusok:
 - void AssembleFlare(): Összerakja a játék végéhez szükséges rakéta pisztolyt. 1 munkaegység
 - void Chill(): A testhő 1-el csökken, ha 0 alá megy → GameOver.
 - void DecrementEnergy(): Az energiát csökkentő helper metódus.
 - void Dig(): Ezt a metódust a Controller hívja. A játékos havat ás. 1 munkaegység
 - void EatFood(): Ezt a metódust a Controller hívja. A játékos eszik. A testhője megnő 1-el.
 - void Pickup(): Ezt a metódust a Controller hívja. A játékos felvesz egy tárgyat. 1 munkaegység
 - void Equip(inventorySlot: int): Ezt a metódust a Controller hívja. A játékos kiválaszt egy tárgyat használatra.
 - void PlaceOn(Tile t): Init szekvencia része. RopeRescue szekvencia része. Rárak egy játékost egy másik Tile-ra.
 - void RescueTeammate(direction: int): Ezt a metódust a Controller hívja. A játékos kiment egy másikat a vízből. 1 munkaegység
 - void ResistWater(): A játékos testhője a WaterResistance szerint változik.
 - void Step(direction: int): Ezt a metódust a Controller hívja. A játékos lép, ha van még hozzá elég energiája. 1 munkaegység
 - void ToFoodStore(): Élelem megtalálásához helper metódus.

8.2.22. PolarBear

- Jegesmedve osztály. Random lépeget a táblán és ha playert talál megtámadja azt.
- Ősosztályok:
 - Entity
- Metódusok:
 - Step(Directon d): Lép az adott irányba.

8.2.23. PolarExplorer

- Játékos fajta. 4 egységnyi testhővel kezd. Képes megnézni egy cella teherbíró képességét. A játékos irányítja.
- Ősosztályok:
 - Player
- Metódusok:
 - int Examine(direction: int): A játékos megnézheti, hogy egy adott irányban lévő Tile-nak mennyi a teherbírása.

8.2.24. RescueStrategy

- A játékos így húzza ki csapattársát a vízből. A játékos így képes megmenteni a vízbe esett csapattársát a szomszédos celláról, a megvalósított stratégia alapján. Kötél szükséges a másik játékos megmentéséhez.
- Metódusok:
 - abstract void Rescue(Tile water, Tile land): A stratégiát megvalósító elem dolga implementálni mi történik.

8.2.25. Rope

- Jégbe fagyott kötél. Ezzel lehet megmenteni a vízbe esett csapattársat a szomszédos celláról.
- Interfészek:
 - Item
- Metódusok
 - void GiveTo(Player p): A játékos kap egy kötelet. Az bekerül az inventoryjába és a megfelelő stratégiájához is a kötél által adott képesség.

8.2.26. RopeRescue

- A játékos kihúzza csapattársát a vízből. A játékos így menti meg a szomszédos cellán vízbe esett csapattársát.
- Interfészek:
 - RescueStrategy
- Metódusok:
 - void Rescue(Tile water, Tile land): A vízben lévők közül egyvalaki rákerül a kihúzó játékos cellájára.

8.2.27. ScubaGear

- Jégbe fagyott búvárruha. Ezzel lehet életben maradni a vízben.
- Interfészek:
 - Item
- Metódusok:
 - void GiveTo(): A játékos búvárruhát kap. Az bekerül az inventoryjába és a megfelelő stratégiája helyére is a búvárruha által adott képesség.

8.2.28. ScubaWearing

- A játékos testhője nem csökken a vízben. A játékos nem hal bele, ha a vízben marad.
- Interfészek:
 - WaterResistanceStrategy
- Metódusok:
 - void Chill(p: Player): A játékost nem hűti a víz, mivel búvárruhát visel.

8.2.29. Sea

- Ez a cella tenger, hűti a játékosokat.
- Interfészek:
 - ChillWaterStrategy
- Metódusok:
 - void Chill(Tile t): Minden rajta álló testhője csökken a WaterResistanceStrategy szerint.

8.2.30. Shovel

- Jégbe fagyott ásó. Ezzel lehet több havat eltakarítani a celláról.
- Interfészek:
 - Item
- Metódusok:
 - void GiveTo(): A játékos ásót kap, ami bekerül az inventoryjába és a megfelelő stratégiájához is bekerül az ásó által adott képesség.

8.2.31. ShovelDig

- Egyszer lehet ásni vele fáradtság nélkül is.
- Interfészek:
 - DigStrategy
- Attribútumok:
 - lastUsed: bool: Volt-e már használva a körben.
- Metódusok:
 - bool Dig(Tile t): Csökkenti a tile-on található hó mennyiségét. Minden második alkalommal fárasztó.

8.2.32. Tent

- Sátor osztály. Le lehet rakni táblára.
- Ősosztályok:
 - Shelter
- Metódusok:
 - void ChillStorm(Tile t): Így viselkedik a tábla, ha sátor van rajta hóviharban.
 - void Break(Tile t): Így viselkedik a tábla, ha eltörik miközben valaki a sátorban van.

8.2.33. TentKit

- Sátor építését lehetővé teszi.
- Interfészek:
 - Item
- Metódusok:
 - void GiveTo(Player p): A játékos így kap sátor alapanyagot.

8.2.34. Tile

- Cella, ilyenekből áll a jégmező ahol a játékosok játszanak.
- Attribútumok:
 - - chillStormStrategy: ChillStormStrategy: Eldönti, kinek változik a testhője vihar esetén.
 - - chillWaterStrategy: ChillWaterStrategy: Eldönti, kinek változik a testhője víz esetén.
 - - item: Item: Ezt a tárgyat lehet kiásni belőle.
 - - neighborTiles: Tile[*]: Szomszédos cellákat ismer.
 - - occupants: Entity[*]: Rajta lévő entitások.
 - - snow: int: Rajta lévő hómennyiség.
 - - weightLimit: int: Rajta lévő játékosok számának maximuma.
- Metódusok:
 - - void Add(Entity): Hozzáad egy entitást a táblához.
 - + void BreakShelter(): Ez a metódus eltávolítja a sátrat a tábláról.
 - + void BearAttack(): Ezt a metódust a Controller hívja
 - + void ChillStorm(): Ezt a metódust a Controller hívja viharban. Hűti a játékosokat, ha nincsenek igluban vagy sátorban.
 - + void ChillWater(): Ezt a metódust a Controller hívja körönként. Hűti a játékosokat, ha ez a cella víz.
 - + void DecrementSnow(): A hómennyiséget csökkentő helper függvény.
 - - void Remove(Entity): Eltávolítja a rajta álló entitást.
 - + Item TakeItem(): A játékos megkapja a tartalmazott tárgyat.

```

1 ||      if( Tile has item on it)
2 ||          Remove item from the tile
3 ||          return the item

```

- + Tile NeighborAt(direction): Visszaadja az adott irányban szomszédos cellát.
- + StepOn(Entity): Játékos rálép a cellára, ha többen vannak mint a korlát, a jégtábla átfordul. A függvény futása során beállítja a megfelelő adattagokat az új értékekre.

```

1 |         Add palyer to the tile
2 |         if ( weight limit is exceeded ) {
3 |             Set tile to sea
4 |             Chill the occupying players on the tile
5 |         }

```

- + StepOff(Entity): Játékos lelép a celláról. A függvény futása során beállítja a megfelelő adattagokat az új értékekre.

8.2.35. WaterResistanceStrategy

- Így reagál a játékos a hideg vízre. A vízben búvárruh nélkül nem lehet mozogni. A vízből ha búvárruha nélkül nem húznak ki, nem lehet életben maradni.
- Metódusok:
 - abstract void Chill(Player p): A stratégiát megvalósító elem dolga implementálni mi történik.

8.2.36. Proto

- Felelősség
Beolvas parancsokat, értelmezi és futtatja őket.
- Attribútumok
 - +game: Game;
A teljes játékot tartalmazza.
 - -running: boolean;
A parancsok feldolgozása megállítható vele.
 - -parsers: CommandParser[*];
Ilyen parancsokat tud feldolgozni.
 - -selectedTile: Tile[0..1];
 - -selectedPlayer: Player[0..1];
 - -selectedBear: PolarBear[0..1];
 - +selectTile(Tile t);
Beállítja a selectedTile-t és lenullozza a selectedPlayert és a selectedBeart.
 - +selectPlayer(Player t);
Beállítja a selectedPlayer-t és lenullozza a selectedTile-t és a selectedBeart.
 - +selectBear(PolarBear t);
Beállítja a selectedBeart és lenullozza a selectedTile-t és a selectedPlayert.
 - +hasSelectedTile(): boolean;
 - +hasSelectedPlayer(): boolean;
 - +hasSelectedBear(): boolean;
 - +getSelectedTile(): Tile; Kivételt dob ha nincs kiválasztva dolog.
 - +getSelectedPlayer(): Player;
Kivételt dob ha nincs kiválasztva dolog.

- +getSelectedBear(): PolarBear;
Kivételt dob ha nincs kiválasztva dolog.

- Metódusok

- +Proto();

```
1 | create game;
2 | create MessagePrinter(this);
3 | game.subscribe(the message printer);
4 | createParsers();
```

- -createParsers();

Készít egy-egy példányt a beépített CommandParseerekből és feltölti velük a parsers kollekciót.

- +run();

Fut a parancsértelmezés.

```
1 | running = true;
2 | while (running) {
3 |     getCommand();
4 |     try {
5 |         command.execute(this);
6 |     } catch (an exception that we threw) {
7 |         print a meaningful error message;
8 |     }
9 | }
```

- +stop();

Megáll a parancsértelmezés. A running változó false lesz.

- -getCommand(): Command;

Beolvas egy parancsot a standard bemenetről.

```
1 | while (true) {
2 |     read line;
3 |     strip comments and trailing whitespace;
4 |     tokenize by spaces;
5 |     if (there are tokens) {
6 |         the first token is the keyword;
7 |         find CommandParser by keyword;
8 |         if (not found) print a meaningful error message;
9 |         else return CommandParser.parse(tokens);
10 |    }
11 | }
```

8.2.37. MessagePrinter

- Felelősség

Kiírja a konzolra a játék eseményeket.

- Interfészek

GameObserver

- Attribútumok

- -proto: Proto;

- Metódusok

- +MessagePrinter(proto: Proto);
- +victory();
Győzelem üzenet kiírása, aztán proto.stop().
- +gameOver();
Vereség üzenet kiírása, aztán proto.stop().
- +explore(Tile);
Tile.weightLimit kiírása.

8.2.38. Command

- Felelősség
Parancs, végrehajtható formában.
- Metódusok
 - +execute(state: Proto): abstract void;
Végrehajtás az adott állapoton.
 - +toString(): abstract String;
Így jelenik meg a konzolon.

8.2.39. CommandParser

- Felelősség
Elkészít egy fajta parancsot.
- Attribútumok
 - +/keyword: abstract String {readOnly};
A parancs kulcszava.
- Metódusok
 - +parse(tokens: String[1..*] {seq}): abstract Command;
Parancs elkészítése tokenekből.

8.2.40. TileCommand

- Felelősség
- Interfészek
Command
- Metódusok
 - +toString(): String;
l || return "tile " + snow + " " + weightLimit;
 - +execute(state: Proto);
Készít egy Tile-t Game.createTile használatával, majd kiválasztja proto.selectTile-el.

8.2.41. TileCommandParser

- Felelősség
- Interfészek
CommandParser
- Attribútumok
 - +/keyword: String = "tile";
- Metódusok
 - +parse(tokens: String[1..*] {seq}): Command;
 - 1 || snow is the second token as a decimal integer;
 - 2 || if (the thid token equals "*") weightLimit is 999;
 - 3 || else weightLimit is the third token as a decimal integer;
 - 4 || create TileCommand;

8.2.42. BuildingCommand

- Felelősség
- Interfészek
Command
- Metódusok
- Attribútumok
 - -type: String;
 - +BuildingCommand(type: String);
 - +toString(): String;
 - 1 || return "building " + type;
 - +execute(state: Proto);
 - 1 || if (type equals "igloo") create Igloo;
 - 2 || if (type equals "tent") create Tent;
 - 3 || set state.selectedTile.shelter;

8.2.43. BuildingCommandParser

- Felelősség
- Interfészek
CommandParser
- Attribútumok
 - +/keyword: String = "building";
- Metódusok

```

- +parse(tokens: String[1..*] {seq}): Command;
1 | the second token is the type;
2 | accept only "igloo" or "tent";
3 | create BuildingCommand;

```

8.2.44. ItemCommand

- Felelősség

- Interfészek
Command

- Metódusok

- Attribútumok

```

- -type: String;
- +count: int = 1;
- +durability: int = -1;
- +ItemCommand(type: String);
- +toString(): String;
1 | if (count > 1) {
2 |   if (type equals "shovel" and durability > -1)
3 |     return "item shovel " + count + " durability " + durability;
4 |   else
5 |     return "item " + type + " " + count;
6 | }
7 | else {
8 |   if (type equals "shovel" and durability > -1)
9 |     return "item shovel durability " + durability;
10 |  else
11 |    return "item " + type;
12 | }

- +execute(state: Proto);
1 | if (state has tile selected and count > 1)
2 |   throw an exception;
3 | if (state has no tile selected and state has no player selected)
4 |   throw an exception;
5 | for (count times) {
6 |   if (type equal "empty") create Emty;
7 |   if (type equal "food") create Food;
8 |   if (type equal "part") create Part;
9 |   if (type equal "scubagear") create ScubaGear;
10 |  if (type equal "rope") create Rope;
11 |  if (type equal "tentkit") create TentKit;
12 |  if (type equal "shovel") {
13 |    if (durability > -1) create BreakingShovel with durability;
14 |    else create Shovel;
15 |  }
16 |  if (state has tile selected)
17 |    set state.selectedTile.item;

```

```

18 || if (state has player selected)
19 ||     add item to player inventory;
20 || }

```

8.2.45. ItemCommandParser

- Felelősség

- Interfészek
CommandParser

- Attribútumok

- +/keyword: String = "item";

- Metódusok

```

- +parse(tokens: String[1..*] {seq}): Command;
1 || the second token is the type;
2 || accept only "empty", "food", "part", "scubagear", "rope", "tentkit", "
  || shovel"
3 || create ItemCommand with type;
4 || if (type equals "shovel") {
5 ||     if (the third token equals "durability") {
6 ||         the fourth token is the durability as a decimal integer;
7 ||         set the ItemCommand.durability;
8 ||     }
9 ||     else {
10 ||         the third token is the count as a decimal integer;
11 ||         set the ItemCommand.count;
12 ||         if (the fourth token equals "durability") {
13 ||             the fifth token is the durability as a decimal integer;
14 ||             set the ItemCommand.durability;
15 ||         }
16 ||     }
17 || }
18 || else {
19 ||     the third token is the count as a decimal integer;
20 ||     set the ItemCommand.count;
21 || }
22 || return the ItemCommand;

```

8.2.46. EquipCommand

- Felelősség

- Interfészek
Command

- Metódusok

- Attribútumok

- -index: int;

```

- +EquipCommand(index:  int);
- +EquipCommand();
  "equip all" parancs. Az index -1;
- +toString():  String;
1 || if (index > -1) return "equip " + index;
2 || else return "equip all";

- +execute(state:  Proto);
1 || if (index > -1)
2 ||     state.selectedPlayer.equip(index);
3 || else {
4 ||     for (all inventory indices)
5 ||         state.selectedPlayer.equip(index);
6 || }

```

8.2.47. EquipCommandParser

- Felelősség

- Interfészek
CommandParser

- Attribútumok

- +/keyword: String = "equip";

- Metódusok

- +parse(tokens: String[1..*] {seq}): Command;

```

1 || if (the second token equals "all") create EquipCommand;
2 || else {
3 ||     the second token is the index as a decimal integer;
4 ||     create EquipCommand with index;
5 || }

```

8.2.48. SelectCommand

- Felelősség

- Interfészek
Command

- Metódusok

- Attribútumok

- -type: String;
 - -index: int;
 - +SelectCommand(type: String, index: int);
 - +toString(): String;

```

1 || if (index > -1) return "equip " + index;
2 || else return "equip all";

- +execute(state: Proto);

1 || if (type equals "tile") state.selectTile(game.tiles[index]);
2 || if (type equals "polarbear") state.selectBear(game.bears[index]);
3 || if (type equals "player") state.selectPlayer(game.player[index]);

```

8.2.49. SelectCommandParser

- Felelősség

- Interfészek
CommandParser

- Attribútumok

- +/keyword: String = "select";

- Metódusok

- +parse(tokens: String[1..*] {seq}): Command;

```

1 || the second token is the type;
2 || accept only "tile", "polarbear", "player";
3 || if (the type equals "polarbear" and there is no third token)
4 ||   the index is 0;
5 || else the index is the third token as a decimal integer;
6 || create SelectCommand with type and index;

```

8.2.50. EntityCommand

- Felelősség

- Interfészek
Command

- Attribútumok

- -type: String;
 - -playerBodyHeat: int;
 - -playerEnergy: int;

- Metódusok

- +EntityCommand(-type: String);
 - +EntityCommand(-type: String, -int: playerBodyHeat);
 - +EntityCommand(-type: String, -int: playerBodyHeat, -int: playerEnergy);
 - +toString(): String;

```

1 | if (type equals "eskimo" or "polarexplorer") {
2 |   if (playerBodyHeat > -1){
3 |     if (playerEnergy > -1)
4 |       return "entity " + type + " " + playerBodyHeat + " " +
        playerEnergy;
5 |     else
6 |       return "entity " + type + " " + playerBodyHeat;
7 |   }
8 |   else return "entity " + type;
9 | }
10| else return "entity polarbear";

- +execute(state: Proto);

1 | if (type equals "eskimo" or "polarexplorer") {
2 |   if (type equals "eskimo")
3 |     state.game.createEskimo();
4 |   if (type equals "polarexplorer")
5 |     state.game.createPolarExplorer();
6 |   if (playerBodyHeat > -1)
7 |     set player bodyHeat;
8 |   if (playerEnergy > -1)
9 |     set player energy;
10|   state.selectPlayer();
11| }
12| if (type equals "polarbear") {
13|   state.game.createBear();
14|   state.selectBear();
15| }

```

8.2.51. EntityCommandParser

- Felelősség

- Interfészek
CommandParser

- Attribútumok

- +/keyword: String = "entity";

- Metódusok

- +parse(tokens: String[1..*] {seq}): Command;

```

1 | the second token is the type;
2 | accept only "eskimo", "polarexplorer", "polarbear";
3 | if (there is a third token)
4 |   it is the playerBodyHeat as a decimal integer;
5 | if (there is a fourth token)
6 |   it is the playerEnergy as a decimal integer;
7 | create EntityCommand;

```

8.2.52. ConnectCommand

- Felelősség

- Interfészek
Command
- Attribútumok
 - -indices: int[*];
- Metódusok
 - +toString(): String;
 - 1 || "connect " + the indices joined by spaces;
 - +execute(state: Proto);
 - 1 || for (each index in indices) {
 - 2 || add state.game.tiles[index] to the state.currentTile.neighbors
 - collection;
 - 3 || }

8.2.53. ConnectCommandParser

- Felelősség
- Interfészek
CommandParser
- Attribútumok
 - +/keyword: String = "connect";
- Metódusok
 - +parse(tokens: String[1..*] {seq}): Command;
 - 1 || all tokens except the first one are indices as decimal integers;
 - 2 || create ConnectCommand;

8.2.54. StepCommand

- Felelősség
- Interfészek
Command
- Metódusok
- Attribútumok
 - -direction: int;
 - +StepCommand(direction: int);
 - +toString(): String;
 - 1 || return "step " + direction;
 - +execute(state: Proto);
 - A kiválasztott játékos lép;

8.2.55. StepCommandParser

- Felelősség
- Interfészek
CommandParser
- Attribútumok
 - +/keyword: String = "step";
- Metódusok
 - +parse(tokens: String[1..*] {seq}): Command;
 - 1 || the second token is the direction as a decimal integer;
 - 2 || create StepCommand with direction;

8.2.56. RescueCommand

- Felelősség
- Interfészek
Command
- Metódusok
- Attribútumok
 - -direction: int;
 - +RescueCommand(direction: int);
 - +toString(): String;
 - 1 || return "rescue " + direction;
 - +execute(state: Proto);
 - A kiválasztott játékos kihúzza csapattársát;

8.2.57. RescueCommandParser

- Felelősség
- Interfészek
CommandParser
- Attribútumok
 - +/keyword: String = "rescue";
- Metódusok
 - +parse(tokens: String[1..*] {seq}): Command;
 - 1 || the second token is the direction as a decimal integer;
 - 2 || create RescueCommand with direction;

8.2.58. ExamineCommand

- Felelősség

- Interfészek
Command

- Metódusok

- Attribútumok

```
- -direction:  int;
- +ExamineCommand(direction:  int);
- +toString():  String;
1 || return "examine " + direction;
- +execute(state:  Proto);
```

A kiválasztott sarkkutató felderít. Ha nem sarkkutató van kiválasztva, akkor kivételt dob.

8.2.59. ExamineCommandParser

- Felelősség

- Interfészek
CommandParser

- Attribútumok

```
- +/keyword:  String = "examine";
```

- Metódusok

```
- +parse(tokens:  String[1..*] {seq}):  Command;
1 || the second token is the direction as a decimal integer;
2 || create ExamineCommand with direction;
```

8.2.60. DigCommand

- Felelősség

- Interfészek
Command

- Metódusok

```
- +toString():  String;
1 || return "dig";
- +execute(state:  Proto);
  A kiválasztott játékos ás;
```

8.2.61. DigCommandParser

- Felelősség
- Interfészek
CommandParser
- Attribútumok
 - `+/keyword: String = "dig";`
- Metódusok
 - `+parse(tokens: String[1..*] {seq}): Command;`
Visszaad egy DigCommandot.

8.2.62. PickupCommand

- Felelősség
- Interfészek
Command
- Metódusok
 - `+toString(): String;`
`l || return "pickup";`
 - `+execute(state: Proto);`
A kiválasztott játékos felvesz egy tárgyat.

8.2.63. PickupCommandParser

- Felelősség
- Interfészek
CommandParser
- Attribútumok
 - `+/keyword: String = "pickup";`
- Metódusok
 - `+parse(tokens: String[1..*] {seq}): Command;`
Visszaad egy PickupCommandot.

8.2.64. BuildCommand

- Felelősség
- Interfészek
Command
- Metódusok

```
- toString(): String;  
1 || return "build";  
  
- execute(state: Proto);  
  A kiválasztott játékos épít.
```

8.2.65. BuildCommandParser

- Felelősség
- Interfészek
 CommandParser
- Attribútumok
 - +/keyword: String = "build";
- Metódusok
 - +parse(tokens: String[1..*] {seq}): Command;
 Visszaad egy BuildCommandot.

8.2.66. AssembleCommand

- Felelősség
- Interfészek
 Command
- Metódusok
 - toString(): String;
 1 || return "assemble";

 - execute(state: Proto);
 A kiválasztott játékos összerakja a rakétát.

8.2.67. AssembleCommandParser

- Felelősség
- Interfészek
 CommandParser
- Attribútumok
 - +/keyword: String = "assemble";
- Metódusok
 - +parse(tokens: String[1..*] {seq}): Command;
 Visszaad egy AssembleCommandot;

8.2.68. EatCommand

- Felelősség
- Interfészek
Command
- Metódusok
 - +toString(): String;
 - 1 || return "eat";
 - +execute(state: Proto);
A kiválasztott játékos eszik.

8.2.69. EatCommandParser

- Felelősség
- Interfészek
CommandParser
- Attribútumok
 - +/keyword: String = "eat";
- Metódusok
 - +parse(tokens: String[1..*] {seq}): Command;
Visszaad egy EatCommandot;

8.2.70. TurnCommand

- Felelősség
- Interfészek
Command
- Metódusok
 - +toString(): String;
 - 1 || return "turn";
 - +execute(state: Proto);
Új kör kezdődik a játékban.

8.2.71. TurnCommandParser

- Felelősség
- Interfészek
CommandParser
- Attribútumok

```
- +/keyword: String = "turn";
```

- Metódusok

```
- +parse(tokens: String[1..*] {seq}): Command;
  Visszaad egy TurnCommandot;
```

8.2.72. StormCommand

- Felelősség

- Interfészek
Command

- Metódusok

```
- +toString(): String;
1 || return "storm";

- +execute(state: Proto);

1 || for (each tile in state.game.tiles)
2 ||   tile.chillStorm();
```

8.2.73. StormCommandParser

- Felelősség

- Interfészek
CommandParser

- Attribútumok

```
- +/keyword: String = "storm";
```

- Metódusok

```
- +parse(tokens: String[1..*] {seq}): Command;
  Visszaad egy StormCommandot.
```

8.2.74. QueryCommand

- Felelősség

- Interfészek
Command

- Metódusok

```
- +toString();
  { return "query"; }

- +execute(state: Proto);
  Parancsok formájában írja ki a játék állapotát.
```

```

1 | for (command: makeCommands(state.game))
2 |   print line command.toString();

- -makeCommands(Game game): Command[*] {seq};
  A parancsok listázása.

1 | result is a writable collection;
2 | for (each tile in game.tiles) {
3 |   add makeTileCommand(tile) to result;
4 |   if (tile is not instance of BareIce)
5 |     add makeBuildingCommand(tile) to result;
6 |   if (item is not instance of Empty)
7 |     add makeItemCommand(item) to result;
8 |   for (each entity in tile.occupants) {
9 |     add makeEntityCommand(entity) to result;
10 |    if (entity is instance of Player) {
11 |      add makePlayerCommand(player) to result;
12 |      add listPlayerEquippedItems(player) to result;
13 |      add "equip all" command to result;
14 |      for (item: player.inventory)
15 |        add makeItemCommand(item) to result;
16 |    }
17 |  }
18 | }
19 | for (each tile in game.tiles) {
20 |   add makeSelectTileCommand(tile, game) to result;
21 |   add makeConnectCommand(tile, game) to result;
22 | }
23 | return result

- -listPlayerEquippedItems(player: Player): ItemCommand[*] {seq};
  Megvizsgálja, hogy milyen tárgyak vannak a játékos használatában, és listázza azokat.

1 | result is a writable collection;
2 | if (player.buildStrategy.count > 0)
3 |   add makeItemCommand(TentKit, player.buildStrategy.count) to result;
4 | if (player.foodStore.count > 0)
5 |   add makeItemCommand(Food, player.foodStore.count) to result;
6 | if (player.partStore.count > 0)
7 |   add makeItemCommand(Part, player.partStore.count) to result;
8 | if (player.rescueStrategy is instance of RopeRescue)
9 |   add makeItemCommand(Rope) to result;
10 | if (player.waterResistanceStrategy is instance of ScubaWearing)
11 |   add makeItemCommand(ScubaGear) to result;
12 | if (player.digStrategy is instance of ShovelDig)
13 |   add makeItemCommand(Shovel) to result;
14 | if (player.digStrategy is instance of BreakingShovelDig) {
15 |   make BreakingShovel with durability player.digStrategy.durability;
16 |   add makeItemCommand(the BreakingShovel) to result;
17 | }
18 | return result;

- -makeTileCommand(tile: Tile): TileCommand;
  Készít egy TileCommandot tile.snow és tile.weightLimit tulajdonságokal.

- -makeBuildingCommand(tile: Tile): BuildingCommand;
  Készít egy BuildingCommandot a tile.shelter alapján.

```

- `-makeItemCommand(item: Item): ItemCommand;`
Készít egy `ItemCommand`-ot, az item típusa alapján. Ha ez `BreakingShovel`, akkor a `durability`-t is beleteszi.
- `-makeItemCommand(item: Item, int count): ItemCommand;`
Készít egy `ItemCommand`-ot, számosság megadásával.
- `-makeEntityCommand(entity: Entity): EntityCommand;`
Készít egy `EntityCommand`-ot. Ha `Player`, akkor a `player.bodyHeat` és `player.energy` is bele kerül.
- `-makeSelectTileCommand(tile: Tile, game: Game): SelectCommand;`
Készít egy `SelectCommand`-ot, a `tile` `game.tiles`-beli indexével.
- `-makeConnectCommand(tile: Tile, game: Game): ConnectCommand;`
Készít egy `ConnectCommand`-ot. Megkeresi a `tile.neighbors` indexeit a `game.tiles` tömbben és azokat rakja a `ConnectCommand`-ba.

8.2.75. `QueryCommandParser`

- Felelősség
- Interfészek
`CommandParser`
- Attribútumok
 - `+keyword: String = "query";`
- Metódusok
 - `+parse(tokens: String[1..*] {seq}): Command;` Visszaad egy `QueryCommand`-ot.

8.2.76. Osztály1

- Felelősség
[Mi az osztály felelőssége. Kb 1 bekezdés. Ha szükséges, akkor state-chart is.]
- Ősosztályok
*[Mely osztályokból származik (öröklési hierarchia)
Legősebb osztály → Ősosztály2 → Ősosztály3...]*
- Interfészek
[Mely interfészeket valósítja meg.]
- Attribútumok
[Milyen attribútumai vannak]
 - `attributum1`: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa
 - `attributum2`: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa
- Metódusok
[Milyen publikus, protected és privát metódusokkal rendelkezik. Metódusonként precíz leírás, ha szükséges, activity diagram is a metódusban megvalósítandó algoritmusról.]
 - `int foo(Osztály3 o1, Osztály4 o2)`: metódus leírása, láthatósága (UML jelöléssel)
 - `int bar(Osztály5 o1)`: metódus leírása, láthatósága (UML jelöléssel)

8.2.77. Osztály2

- Felelősség
[Mi az osztály felelőssége. Kb 1 bekezdés. Ha szükséges, akkor state-chart is.]
- Ősosztályok
*[Mely osztályokból származik (öröklési hierarchia)
Legősebb osztály → Ősosztály2 → Ősosztály3...]*
- Interfészek
[Mely interfészeket valósítja meg.]
- Attribútumok
[Milyen attribútumai vannak]
 - attribútum1: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa
 - attribútum2: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa
- Metódusok
[Milyen publikus, protected és privát metódusokkal rendelkezik. Metódusonként precíz leírás, ha szükséges, activity diagram is a metódusban megvalósítandó algoritmusról.]
 - int foo(Osztály3 o1, Osztály4 o2): metódus leírása, láthatósága (UML jelöléssel)
 - int bar(Osztály5 o1): metódus leírása, láthatósága (UML jelöléssel)

8.3. A tesztek részletes tervei, leírásuk a teszt nyelvén

[A tesztek részletes tervei alatt meg kell adni azokat a bemeneti adatsorozatokot, amelyekkel a program működése ellenőrizhető. Minden bemenő adatsorozathoz definiálni kell, hogy az adatsorozat végrehajtásától a program mely részeinek, funkcióinak ellenőrzését várjuk és konkrétan milyen eredményekre számítunk, ezek az eredmények hogyan vethetők össze a bemenetekkel.]

8.3.1. Teszteset1

- Leírás
[szöveges leírás, kb. 1-5 mondat.]
- Ellenőrzött funkcionalitás, várható hibahelyek
- Bemenet
[a proto bemeneti nyelvén megadva (lásd előző anyag)]
- Elvárt kimenet
[a proto kimeneti nyelvén megadva (lásd előző anyag)]

8.3.2. Teszteset2

- Leírás
[szöveges leírás, kb. 1-5 mondat.]
- Ellenőrzött funkcionalitás, várható hibahelyek
- Bemenet
[a proto bemeneti nyelvén megadva (lásd előző anyag)]
- Elvárt kimenet
[a proto kimeneti nyelvén megadva (lásd előző anyag)]

8.4. A tesztelést támogató programok tervei

[A tesztadatok előállítására, a tesztek eredményeinek kiértékelésére szolgáló segédprogramok részletes terveit kell elkészíteni.]

8.5. Napló

Kezdet	Időtartam	Résztevők	Leírás
2020.04.09. 16:00	1 óra	Glávits	Tesztnyelv fejlesztése
2020.04.09. 19:00	1 óra	Glávits	Parser tervezése
2020.04.10. 18:00	3 óra	Glávits	Parser tervezése
2020.04.11. 22:00	3 óra	Glávits	Parser tervezése
2020.04.11. 23:00	1 óra	Lant	Osztály leírások
2020.04.11. 16:00	2 óra	Glávits	Parser tervezése
2020.04.11. 17:00	2 óra	Konrád	Tesztek írása
2020.04.11. 17:00	2 óra	Máté	Tesztek írása
2020.04.11. 21:00	1 óra	Glávits	Tesztek írása