8. Részletes tervek

22 – beta

Konzulens:
Hartung István

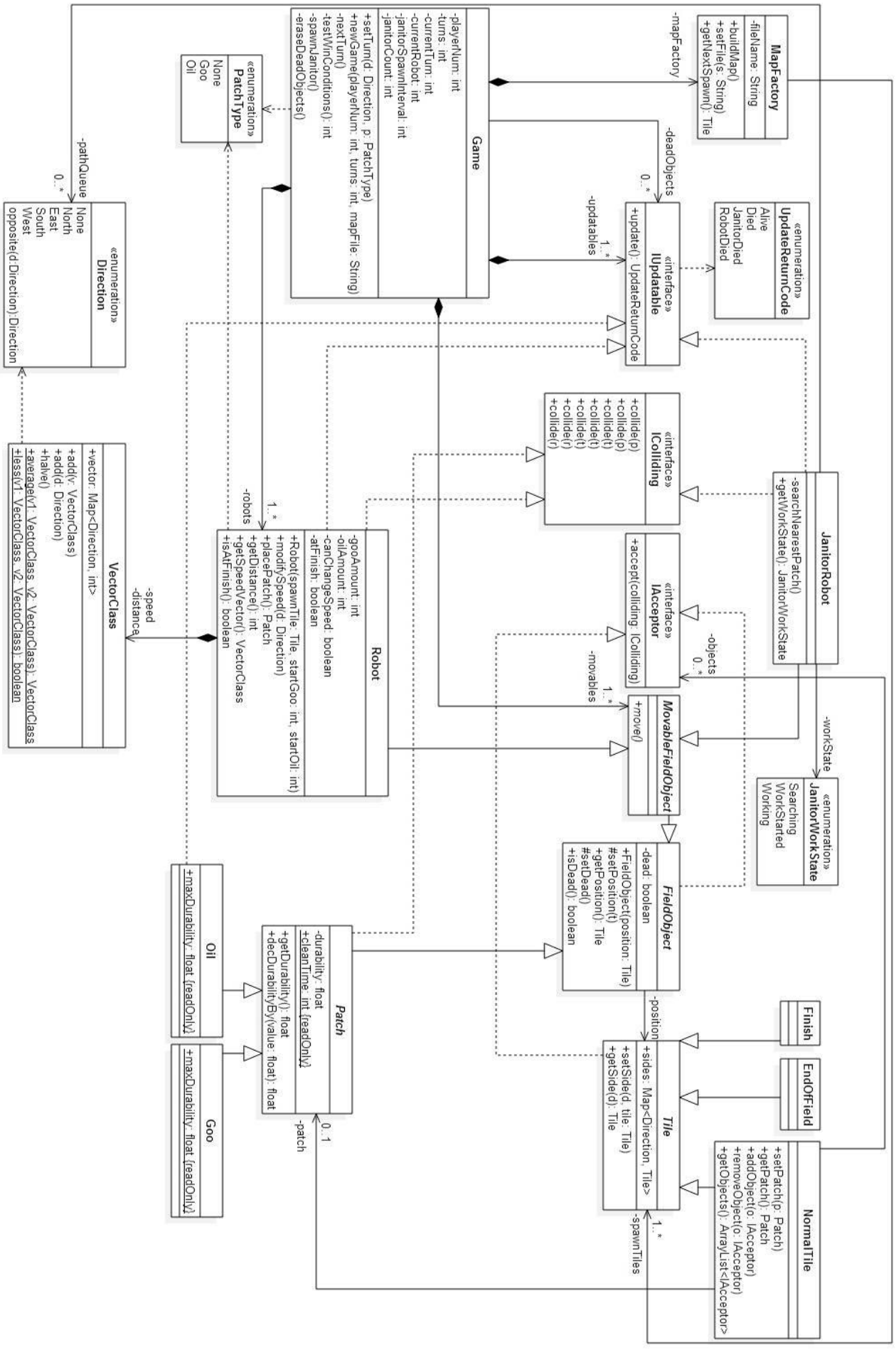
Csapattagok

Dobosy Kristóf	PREJC4	kikko634@gmail.com
Király Ákos	W2BFQI	laroi.akos@gmail.com
Dankó Barnabás	LDK1SP	danko.barna@gmail.com
Kovács Zsolt	H39JKL	marament@gmail.com
Szabó Zsolt	HJ4DMQ	zsolth5@gmail.com

2015. március 02.

8. Részletes tervek

8.1 Osztályok és metódusok tervei.


MapFactory
-fileName: String
+buildMap()
+setFile(s: String)
+getNextSpawn(): Tile
-mapFactory
Game
-playerNum: int
-turns: int
-currentTurn: int
-currentRobot: int
-janitorSpawnInterval: int
-janitorCount: int
+setTurn(d: Direction, p: PatchType)
+newGame(playerNum: int, turns: int, mapFile: String)
-nextTurn()
-testWinConditions(): int
-spawnJanitor()
-eraseDeadObjects()
-deadObjects
-updatables
«enumeration»
UpdateReturnCode
Alive
Died
JanitorDied
RobotDied
«interface»
IUpdatable
+update(): UpdateReturnCode
«enumeration»
PatchType
None
Goo
Oil
-pathQueue
«enumeration»
Direction
None
North
East
South
West
opposite(d:Direction):Direction
«interface»
IColliding
+collide(p)
+collide(p)
+collide(t)
+collide(t)
+collide(t)
+collide(r)
+collide(r)
JanitorRobot
-searchNearestPatch()
+getWorkState(): JanitorWorkState
-workState
«enumeration»
JanitorWorkState
Searching
WorkStarted
Working
«interface»
IAcceptor
+accept(colliding: IColliding)
-objects
-movables
MovableFieldObject
+move()
Robot
-gooAmount: int
-oilAmount: int
-canChangeSpeed: boolean
-atFinish: boolean
+Robot(spawnTile: Tile, startGoo: int, startOil: int)
+modifySpeed(d: Direction)
+placePatch(): Patch
+getDistance(): int
+getSpeedVector(): VectorClass
+isAtFinish(): boolean
-robots
FieldObject
-dead: boolean
+FieldObject(position: Tile)
#setPosition(t)
+getPosition(): Tile
#setDead()
+isDead(): boolean
-position
VectorClass
+vector: Map<Direction, int>
+add(v: VectorClass)
+add(d: Direction)
+halve()
+average(v1: VectorClass, v2: VectorClass): VectorClass
+less(v1: VectorClass, v2: VectorClass): boolean
-speed
-distance
Patch
-durability: float
+cleanTime: int {readOnly}
+getDurability(): float
+decDurabilityBy(value: float): float
-patch
Oil
+maxDurability: float {readOnly}
Goo
+maxDurability: float {readOnly}
Tile
+sides: Map<Direction, Tile>
+setSide(d, tile: Tile)
+getSide(d): Tile
-spawnTiles
Finish
EndOfField
NormalTile
+setPatch(p: Patch)
+getPatch(): Patch
+addObject(o: IAcceptor)
+removeObject(o: IAcceptor)
+getObjects(): ArrayList<IAcceptor>

8.1.1 EndOfField

- **Felelősség**
  Ezen a mezőn nincsen talaj, amire landolni tudna a jármű. Így ha ilyenre lép a körében egy játékos, akkor az kiesik a játékból.

- **Ősosztályok**

Tile

- **Interfészek**

- **Attribútumok**

- **Metódusok**

8.1.2 FieldObject (absztrakt)

- **Felelősség**
  Ez az osztály az őse minden olyan objektumnak, ami mezőn helyezkedhet el. (Robotok, foltok)

- **Ősosztályok**

- **Interfészek**
  IAcceptor

- **Attribútumok**
  Tile position: Azt a mezőt tárolja, amelyen az objektum elhelyezkedik.
  boolean dead: Megadja, hogy az objektum a kör végén kitörlésre kerülhet-e.

- **Metódusok**
  FieldObject(Tile position): Konstruktor, amely az objektumot egyből a helyére hozza létre.
  Tile getPosition(): Visszaadja azt a Tile-t amin elhelyezkedik az Objektum.
  void setPosition(Tile t): Az objektumot elhelyezi a paraméterként megadott mezőre.
  setDead(): A dead attribútum értékét true-ba állítja.
  boolean isDead(): Visszaadja, hogy az objektum a kör végén kitörölhető-e.

8.1.3 Finish

- **Felelősség**
  Ezen a mező annyit tud, hogy van rajta talaj, rá lehet ugrani, és ha a robot ezen e mezőn fejezi be a körét, akkor megnyerte a játékot.

- **Ősosztályok**

Tile

- **Interfészek**
- **Attribútumok**
- **Metódusok**

8.1.4 Game

- **Felelősség**

A program inicializálása, új játék indítása, valamint annak levezénylése. A játékosok számának, robotjainak nyilvántartása, valamint egyéb a játékmenethez köthető információk tárolása, kezelése.

- **Ősosztályok**
- **Interfészek**
- **Attribútumok**
  - **int playerNum**: A játékosok száma a legutóbb indított játékban. A pályára ennyi robot fog kerülni a játék elején.
  - **int turns**: A maximálisan megengedett körök száma. Amennyiben ezt elérjük, a játéknak vége, és a legnagyobb távolságot megtett robot tulajdonosa nyer.
  - **int currentTurn**: A jelenlegi kör száma. (Vagyis az eddig eltelt körök száma.)
  - **int currentRobot**: Annak a robotnak a száma, amelynek éppen az irányítását végzik.
  - **Vector<Robot> robots**: A játék alatt a játékosok által irányítható robotok listája.
  - **MapFactory mapFactory**: A pálya betöltését végző objektum.
  - **Vector<IUpdateable> updateables**: Azon objektumok listája, amelyeken a kör végén valamilyen esemény zajlik le.
  - **Vector<MovableFieldObject> moveables**: A léptetésre képes objektumok tárolására szolgál.
  - **Vector<IUpdateable> deadObjects:** Ha egy objektumot törlésre kerülne, akkor bekerül ebbe a listába.
  - **int janitorSpawnInterval**: Ez mondja meg, hogy hány körönként legyenek új takarítók lerakva.
  - **int janitorCount**: Ez azt tárolja, jelenleg hány takarító robot van a pályán.
- **Metódusok**
  - **void setTurn(Direction d, PatchType type)**: A jelenleg soronlévő robot sebességének módosítása és a lerakandó folt típusának beállítása annak *modifySpeed* és *placePatch* nevű függvényeinek meghívásával.
  - **void newGame(int playerNum, int turns, String mapFile)**: Új játék indítása. Beállítja a maximálisan megengedett körök számát (*turns*), játékosok számát (*playerNum*), létrehozza az azokhoz tartozó robotokat, valamint betölteti és inicializáltatja a pályát a *mapFactory* objektummal.

- **void nextTurn()**: A kör végén hívódik meg, elvégzi a kör végi módosításokat, lépteti a robotokat, olajfoltokat szárítja.
- **int testWinConditions()**: Megvizsgálja, hogy valamelyik robot teljesítette-e a győzelem valamely feltételét é visszatér a győztes robot indexével. Emellett a halott robotokat eltávolítja a játékból.
- **void spawnJanitor()**: Elhelyez egy takarító robotot a pályán.
- **void eraseDeadObjects():** A deadObjects lista elemeit törli ki.

8.1.5 Goo

- **Felelősség**
  Ez a ragacs osztálya, ha ilyen mezőre érkezik a robot, akkor elveszti sebességének a felét.

- **Ősosztályok**

Patch

- **Interfészek**

- **Attribútumok**
  float maxDurability {readOnly}: Csak olvasható érték, ami azt tárolja a ragacs hány ugrás elteltével kopik le a pályáról.

- **Metódusok**

8.1.6 IAcceptor (interfész)

- **Felelősség**
  Visitor mintán alapuló interfész biztosítása azon osztályoknak, melyek interakcióba kerülhetnek a mezőkkel.
- **Ősosztályok**

- **Metódusok**

Az alábbi metódust a leszármazott osztályoknak kell megvalósítaniuk, és az itt megadott helyzet kezelését hivatottak megoldani.

- **void accept(IColliding colliding):** A megadott collide-nak megfelelő objektumra hívódik meg és végzi el a kívánt funkciót.

8.1.7 IColliding (interfész)

- **Felelősség**

Visitor mintán alapuló interfész biztosítása azon osztályoknak, melyek interakcióba kerülhetnek a mezőkkel.

- **Ősosztályok**

- **Metódusok**

Az alábbi metódusokat a leszármazott osztályoknak kell megvalósítaniuk, és az itt megadott helyzetek kezelését hivatottak megoldani.

- **void collide(Oil t)**: Viselkedés leírása, amennyiben a robot olajfoltra lépett.
- **void collide(EndOfField t)**: Viselkedés leírása, amennyiben a robot olyan mezőre lépett, amely már nem tartozik azon részhez, amin érvényesen haladhat.
- **void collide(NormalTile t)**: Viselkedés leírása, amennyiben a robot szabad mezőre lépett.
- **void collide(Goo t)**: Viselkedés leírása, amennyiben a robot ragacsra lépett.
- **void collide(Finish t)**: Viselkedés leírása, amennyiben a robot a pálya végét (célt) jelző mezőre lépett.
- **void collide(Robot r):** Viselkedés leírása, amennyiben a robot egy másik robotra lépett.
- **void collide(JanitorRobot r):** Viselkedés leírása, amennyiben a robot egy takarító robotra lépett.

8.1.8 IUpdateable (interfész)

- **Felelősség**

Ez az interfész biztosítja, hogy a Game osztály figyelni tudja, hogy mely objektumok változtatják állapotukat a kör végén.

- **Ősosztályok**

- **Metódusok**

Az alábbi metódust a leszármazott osztályoknak kell megvalósítaniuk, és az itt megadott helyzet kezelését hivatottak megoldani.

- **UpdateReturnCode update():** Frissíti az adott objektum belső állapotát (amennyiben szükséges, a többi objektummal való interakció alapján), és visszaadja az objektum frissítés utáni állapotát.

8.1.9 JanitorRobot

- **Felelősség**
  Gép által irányított kisebb robot. Feladata a pályán mindig a legközelebbi ragacs, vagy olajfolt megtalálása, elérése és feltakarítása.
- **Ősosztályok**
  - *MoveableFieldObject*
- **Interfészek**
  - IColliding
  - IUpdateable
  - IAcceptor
- **Attribútumok**
  - **JanitorWorkState workState:** A robot állapotát tárolja, hogy éppen mit csinál.
  - **List<Direction> pathQueue:** A robotot a céljával összekötő lépéssorozatot tárolja, minden lépés után a Queue első elemét kitöröljük.
- **Metódusok**
  - **searchNearestPatch():** Visszaadja a legközelebbi folt helyzetét.

8.1.10 MapFactory

- **Felelősség**

Az eltárolt fájlnév alapján a pálya betöltése és inicializálása, az abban megadott kezdőpozíciók eltárolása.

- **Ősosztályok**

- **Interfészek**

- **Attribútumok**
  - **Vector<Tile> spawnTiles**: Azon mezők listája, amelyekről a robotok a játék elején elindulnak.
  - **String fileName**: A felépítendő pálya neve, ami alapján a MapFactory képes megtalálni a fájlt.

- **Metódusok**
  - **void buildMap()**: A megnyitott fájlban található adatok alapján felépíti a pályát.
  - **void setFile(String s)**: Megnyitja a megadott nevű pályafájlt, és bezárja az esetleg korábban megnyitottat.
  - **Tile getNextSpawn()**: Visszaadja a pályához tartozó soronkövetkező kezdőpozíció mezejét.

8.1.11 MovableFieldObject (absztrakt)

- **Felelősség**

Feladata a mozgatható objektumok egységbe zárása. (Robotok, takarítók)

- **Ősosztályok**
  FieldObject

- **Interfészek**

- **Attribútumok**

- **Metódusok**
  - ***void move ():*** Absztrakt függvény, melyet a leszármazott osztályoknak kell megvalósítaniuk az alapján, hogy az objektum milyen módon végzi a pozíciójának megváltoztatását.

8.1.12 NormalTile

- **Felelősség**

  Ezen az általános mező, amire ugorhat a robot, és gond nélkül tovább is haladhat. Később erre kerülhet rá a ragacs, vagy az olaj.

- **Ősosztályok**

Tile

- **Interfészek**

- **Attribútumok**
  - **Patch p:** A mezőn található folt.
  - **Vector<IAcceptor> objects:** Azon ütközésre képes objektumok listája amik ezen a mezőn vannak.

- **Metódusok**
  - **void setPatch(Patch p):** Beállítja a mezőn lévő foltot a megadott foltra.
  - **Patch getPatch():** Visszaadja a mezőn lévő foltot, vagy null-t.
  - **void addObject(IAcceptor o):** A mezőhöz hozzáadja a paraméterül kapott objektumot. (Ez lehet robot, vagy takarító)
  - **void removeObject(IAcceptor o):** A mezőről eltávolítja a paraméterül kapott objektumot. (Ez lehet robot, vagy takarító)
  - **ArrayList<IAcceptor> getObjects():** Visszaadja a mezőn lév objektumokat (robotok, takarítók).

8.1.13 Oil

- **Felelősség**
  Ezen a mezőre rakható anyag, a robot sebességmódosító képességét veszi el, ha rálép a robot.

- **Ősosztályok**

Patch

- **Interfészek**

IUpdateable

- **Attribútumok**
  float maxDurability {readOnly}: Csak olvasható érték, ami azt tárolja az olaj hány kör elteltével szárad le a pályáról.

- **Metódusok**

8.1.14 Patch (absztrakt)

- **Felelősség**
  Feladata a foltok egységbe zárása.

- **Ősosztályok**
  FieldObject

- **Interfészek**
  IColliding

- **Attribútumok**
  - **float durability:** Az egyes foltnak az aktuális tartósságát tárolja.
  - **int cleanTime {readOnly}:** Az a maximális idő, ami a folt feltakarításához szükséges a takarítónak.

- **Metódusok**
  - **float getDurability():** Visszaadja az aktuális folt tartósságát.
  - **float decDurabilityBy(float value):** Csökkenti az adott folt tartósságát a megadott mennyiséggel.

8.1.15 Robot

- **Felelősség**

A kezdőpozíciótól megtett távolságának, a sebességének, valamint ragacs- és olajkészletének nyilvántartása és módosítása szükség esetén. Az ugrás elvégzése és adott esetben a ragacs/olajfolt letevése eközben a jelenlegi pozícióra.

- **Ősosztályok**
  MoveableFieldObject

- **Interfészek**
  - IColliding
  - IUpdateable

- **Attribútumok**
  - **int gooAmount**: A robotnál lévő ragacsok száma, melyket még a pályára letehet.
  - **int oilAmount**: A robotnál lévő olajadagok száma, melyket még a pályára letehet.
  - **VectorClass speed**: A robot jelenlegi sebessége.
  - **VectorClass distance**: A robot által megtett össztávolság.
  - **boolean canChangeSpeed**: Megmutatja, hogy lehetőség van-e megváltoztatni a robot sebességét a jelenlegi körben. (Ha nem, az értéke *false*.)
  - **boolean atFinish**: Ez jelzi, ha a robot egy olyan mezőn van, amely *Finish* típusú. Ha az értéke *true*, az adott robot beért a célba, tehát nyert.

- **Metódusok**
  - **Robot(Tile spawnTile, int startGoo, int startOil) (konstruktor)**: Inicializálja a robotot: Beállítja a pozícióját, valamint a ragacs- és olajkészletét a kapott értékek alapján.
  - **void collide(Oil t)**: A robot *canChangeSpeed* változóját *false*-ra állítja, ezzel megakadályozva hogy ebben a körben sebességváltoztatás lépjen érvénybe.
  - **void collide(EndOfField t)**: Ennek a függvénynek a meghívása azt jelenti, hogy a robot leesett. Beállítja a *dead* változó értékét *true*-ra.
  - **void collide(NormalTile t)**: A robot üres mezőn van, nem történik semmi különös. (Nincs hatása)

- **void collide(Goo t)**: Lecsökkenti a robot sebességét a felére: Meghívja a robot *speed* változójának *halve* függvényét.
- **void collide(Finish t)**: Amennyiben a robot *Finish* típusú mezőre lépett, nyert. Beállítja az *atFinish* nevű változó értékét *true*-ra.
- **void jump()**: Ugrik, azaz elvégzi a pozíciójának és a megtett távolságának módosítását, ezután meghívja a saját *collide* függvényét azzal a mezővel paraméterként, amire érkezett.
- **void modifySpeed(Direction d)**: Módosítja a sebességét a paraméterében kapott irány szerinti egységvektorral, ha a *canChangeSpeed* változó értéke *false*.
- **void placePatch(PatchType patch)**: Lehelyez maga alá egy a paraméterében meghatározott típusú foltot, úgy hogy kicseréli a maga alatt lévő mezőt a megfelelő típusúra a mező *swapTile* nevű függvényének meghívásával. Ezután csökkenti a megfelelő típusú foltból hátralévő készletének számát.
- **int getDistance()**: Visszaadja a kezdőpozíciótól megtett távolságot. Visszatérési értéke az eltárolt *VectorClass distance* változó alapján kerül kiszámolásra.
- **boolean isDead()**: Visszaadja az *isDead* nevű változó értékét.
- **boolean isAtFinish()**: Visszaadja az *atFinish* nevű változó értékét.

8.1.16 Tile (absztrakt)

- **Felelősség**

A körülötte lévő szomszédos mezők nyilvántartása, saját cseréje esetén a pálya konzisztenciájának megtartása.

- **Ősosztályok**

- **Interfészek**

IAcceptor

- **Attribútumok**
  - **Map<Direction, Tile> sides**: A szomszédos mezők, ahol minden lehetséges irányhoz egy darab szomszéd tartozik.

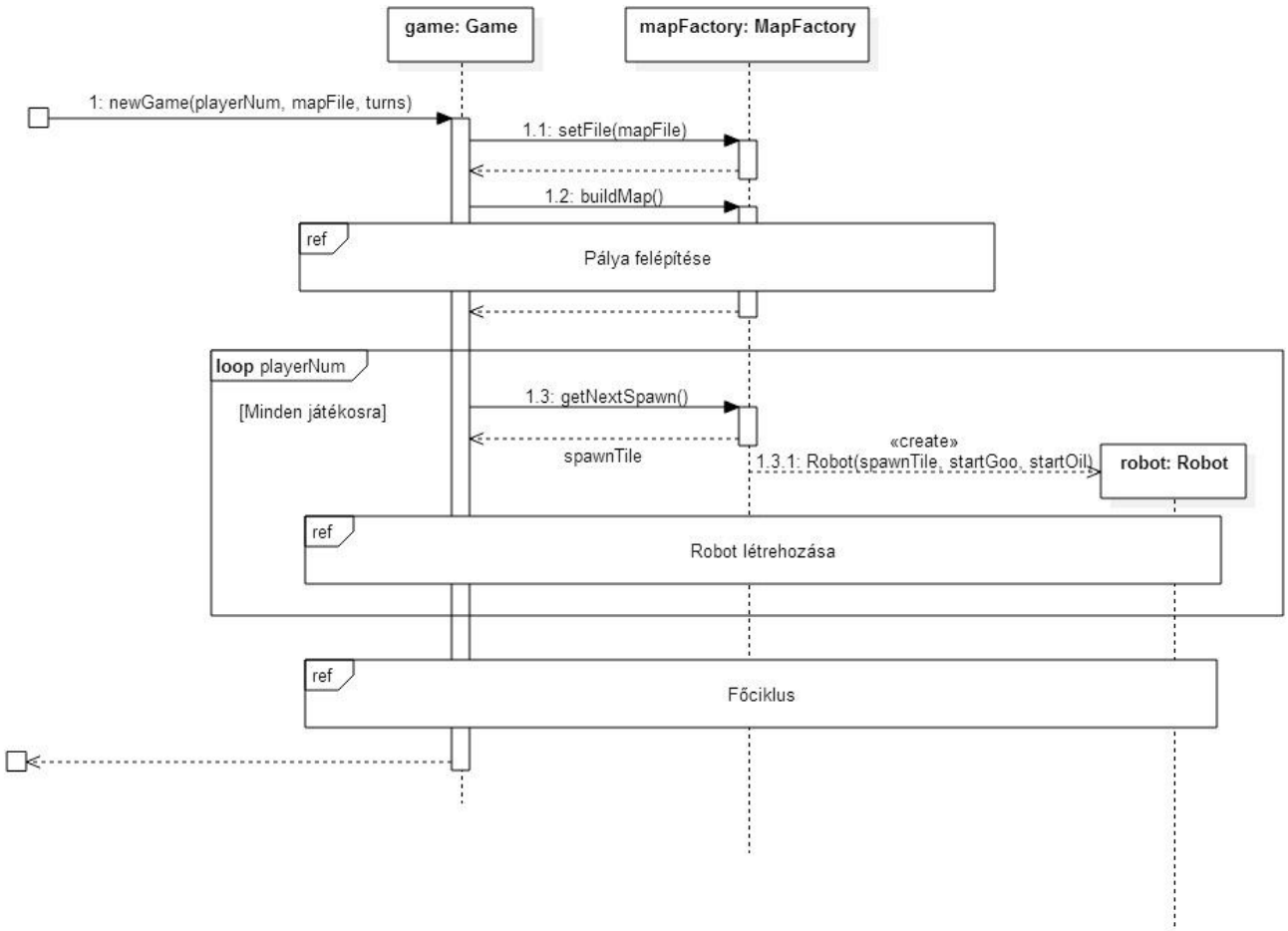
- **Metódusok**
  - **void setSide(Direction d, Tile t)**: Beállítja a megadott *d* irányban lévő szomszédját a *t* paraméterként kapott mezőre.
  - **Tile getSide(Direction d)**: Visszaadja a mező paraméterként kapott irányban lévő szomszédját.

8.1.17 VectorClass

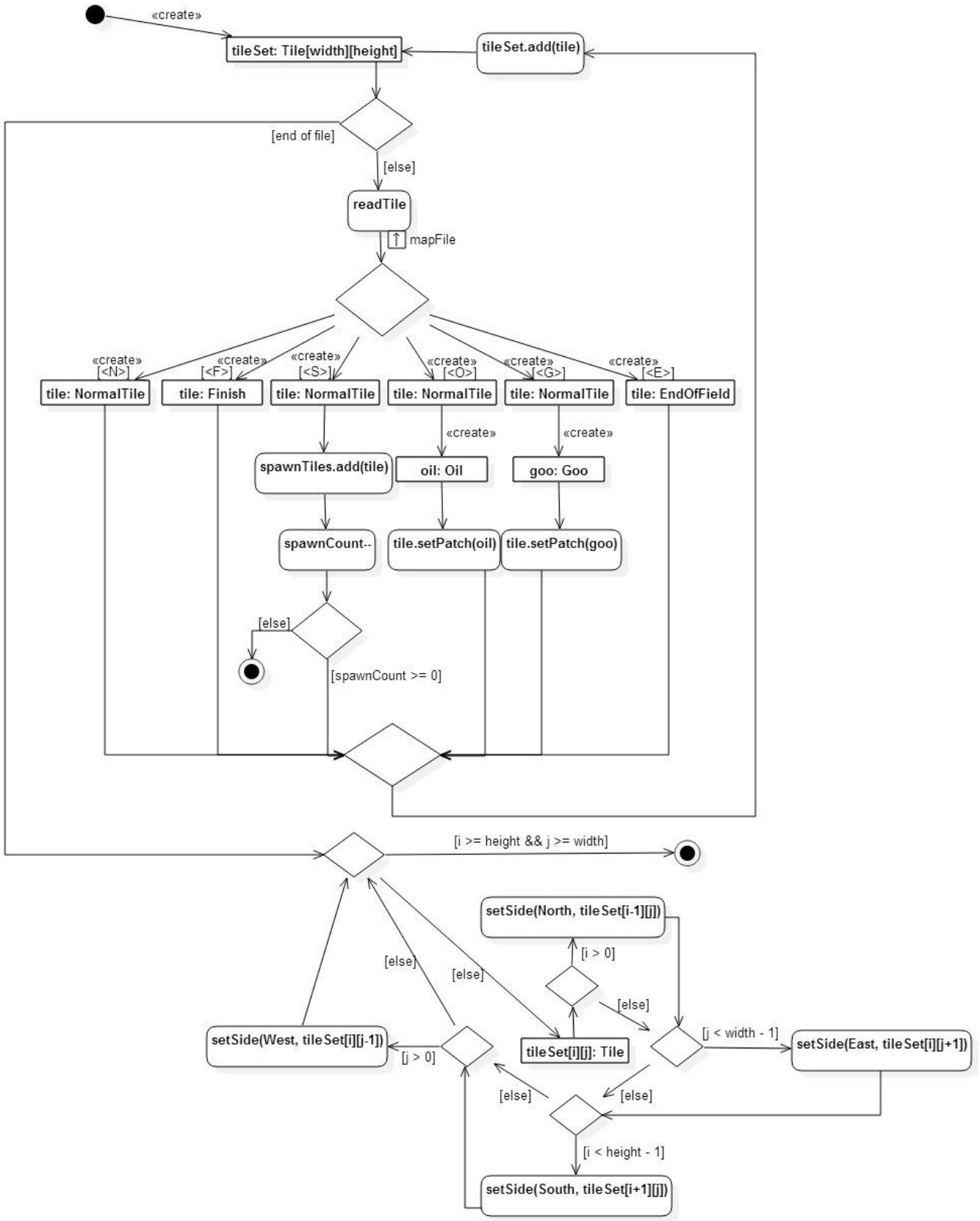
- **Felelősség**

Azon egységvektorok irányainak eltárolása, amelyeket egyenként felhasználva megkapunk egy a pályán navigációra használható vektort.

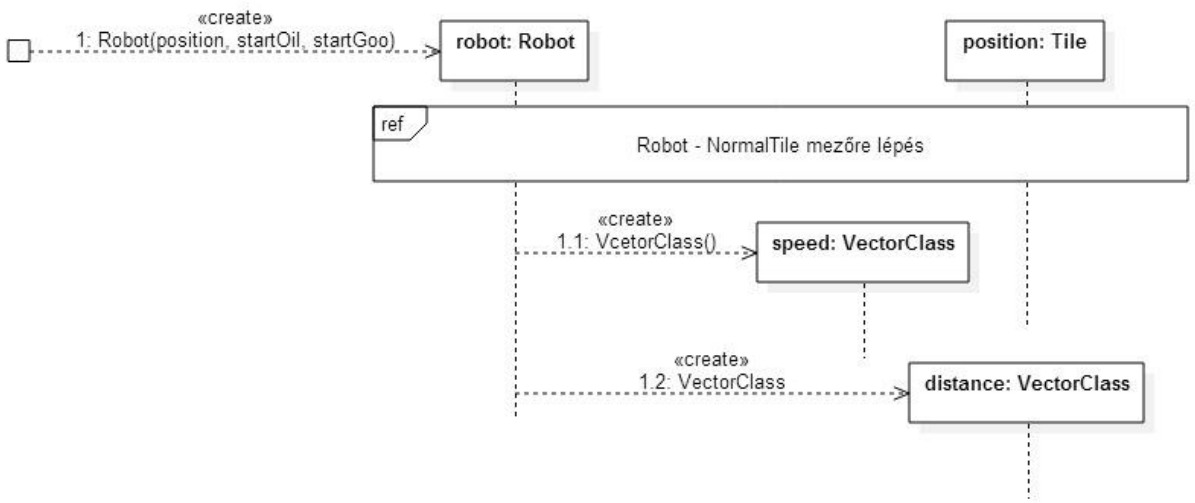
- **Ősosztályok**
- **Interfészek**
- **Attribútumok**
  - **Map<Direction, int> vector**: Azon egységvektorok iránya, melyekből a teljes vektor felépül. Az egyes irányokhoz egy szám van rendelve, ami megmutatja hogy hány egységet kell az adott irányba lépni.
- **Metódusok**
  - **void add(VectorClass v)**: Hozzáadja a paraméterül kapott vektort ehhez, úgy hogy az abban található tároló elemeit hozzáadja a saját tárolójához. (Ez a függvény használandó a távolságvektor megváltoztatására.)
  - **void add(Direction d)**: Hozzáad egy paraméterül kapott irányú egységvektort a belső tárolójához, vagy amennyiben ez ellentétes irányú a tárolóban megtalálható vektorok egyikéhez képest, azt kiveszi a tárolóból. (Ez a függvény használandó a sebességvektor megváltoztatására.)
  - **void halve()**: Megfelezi a vektort: Az eltárolt egységvektorok számát lefelezi minden irányra
  - **VectorClass average(VectorClass v1, VectorClass v2):** Megkap két vektort, és ezek átlagát veszi.
  - **boolean less(VectorClass v1, VectorClass v2):** Visszaadja, hogy v1 < v2.



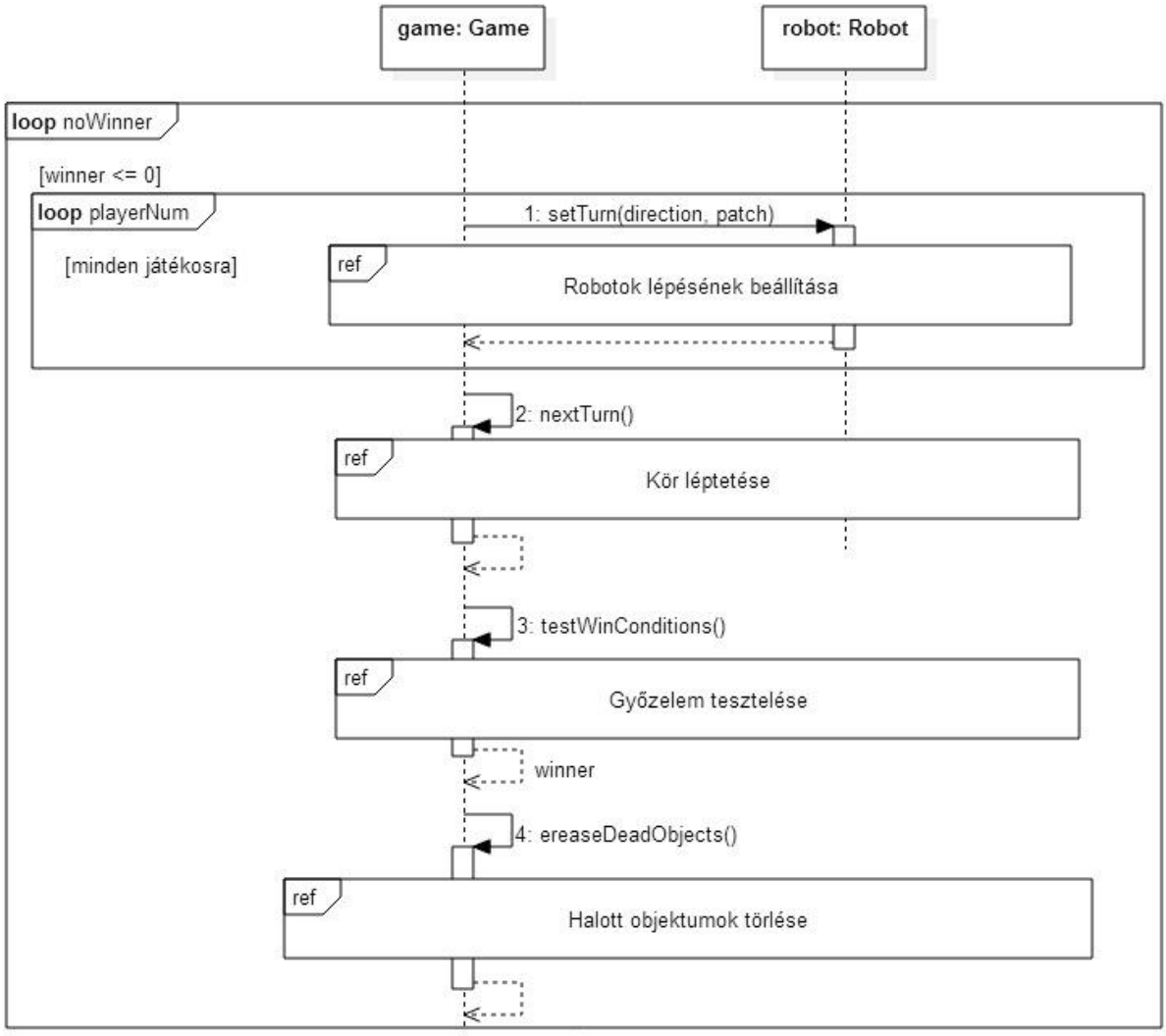
8-1. ábra Új játék



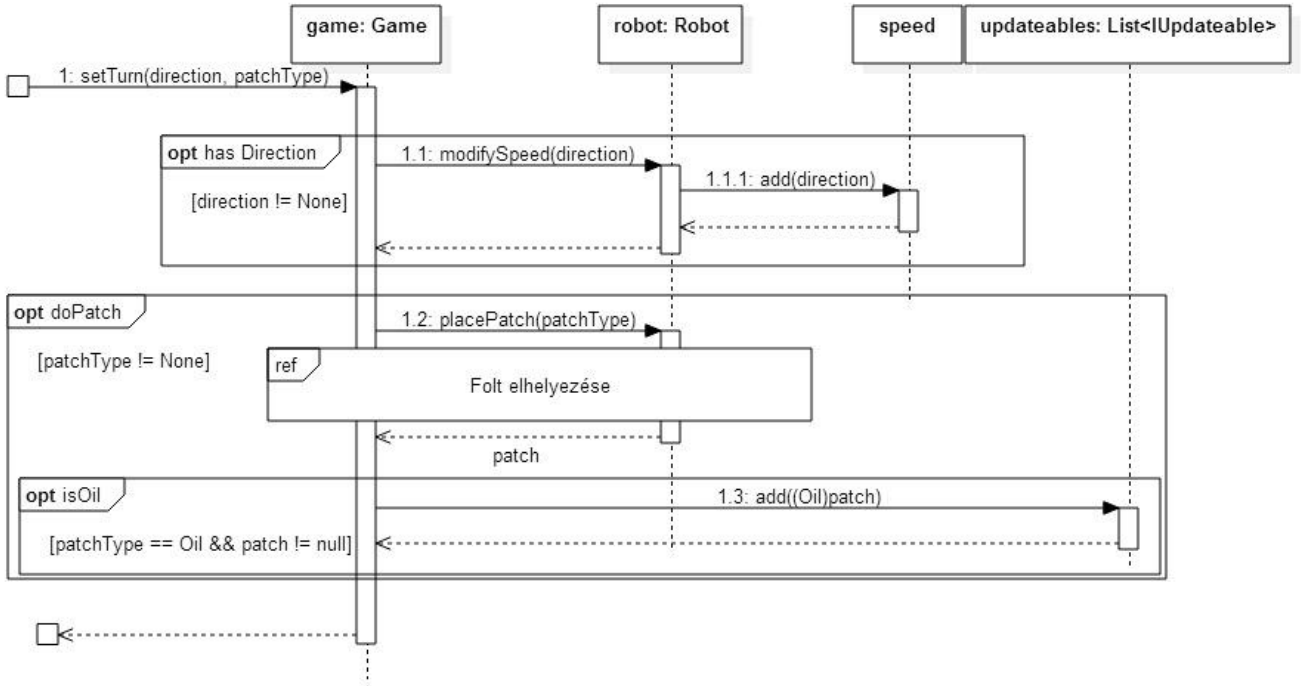
8-2. ábra Pálya felépítése



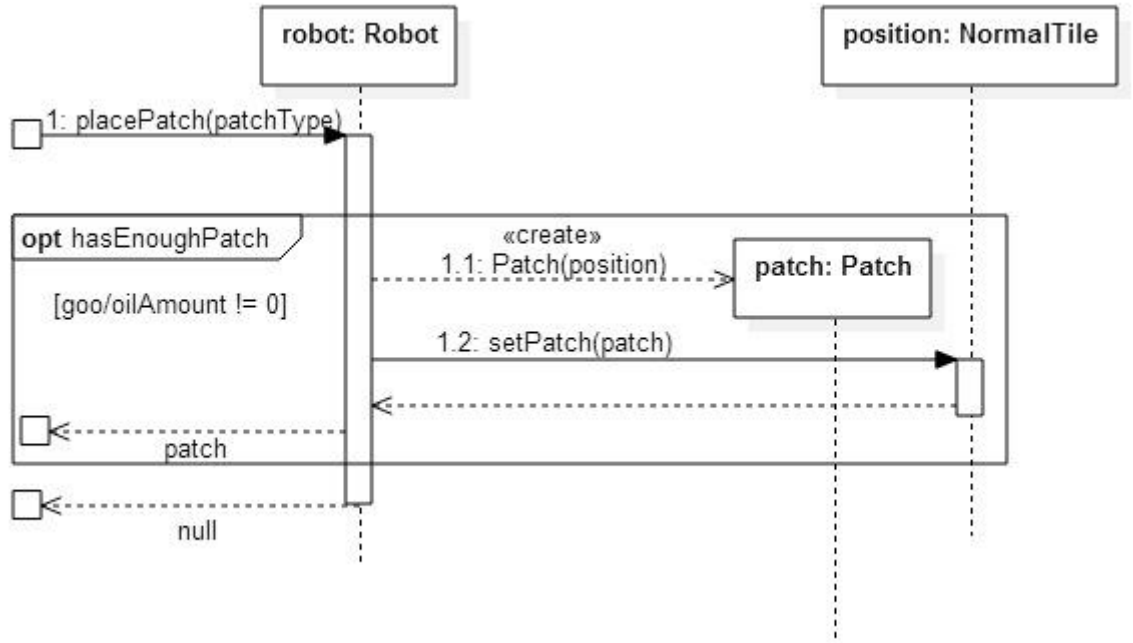
8-3. ábra Robot létrehozása



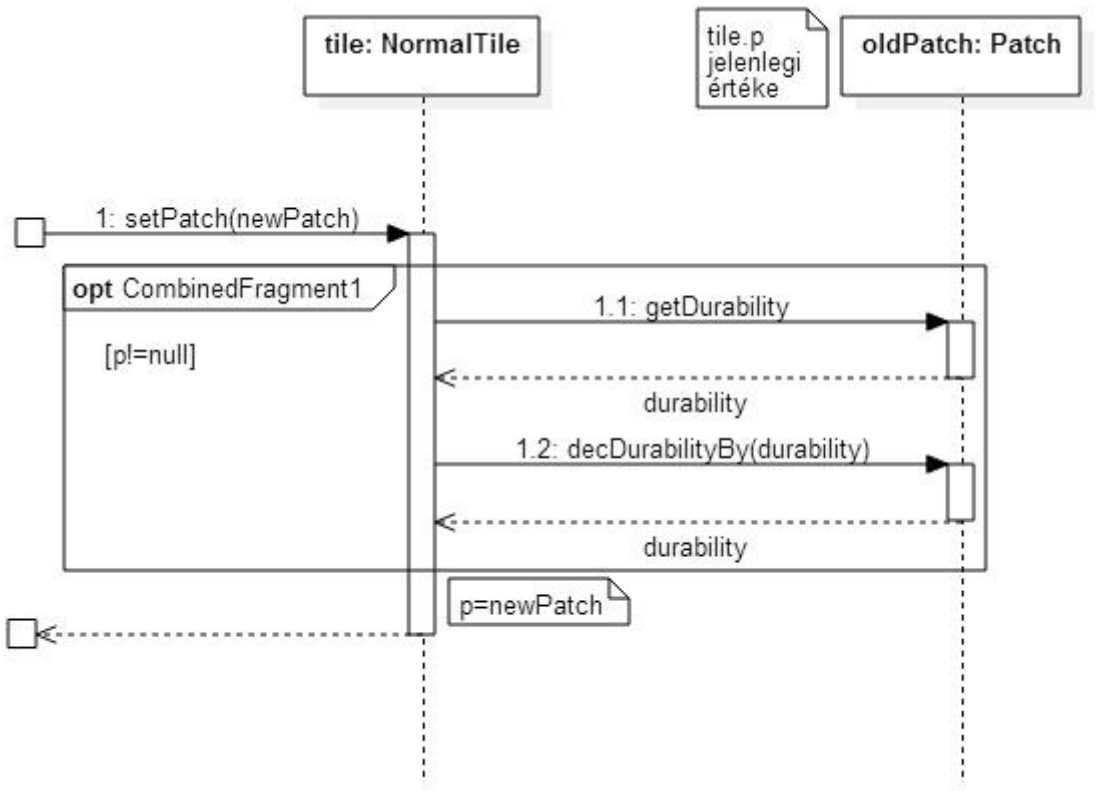
8-4. ábra Főciklus



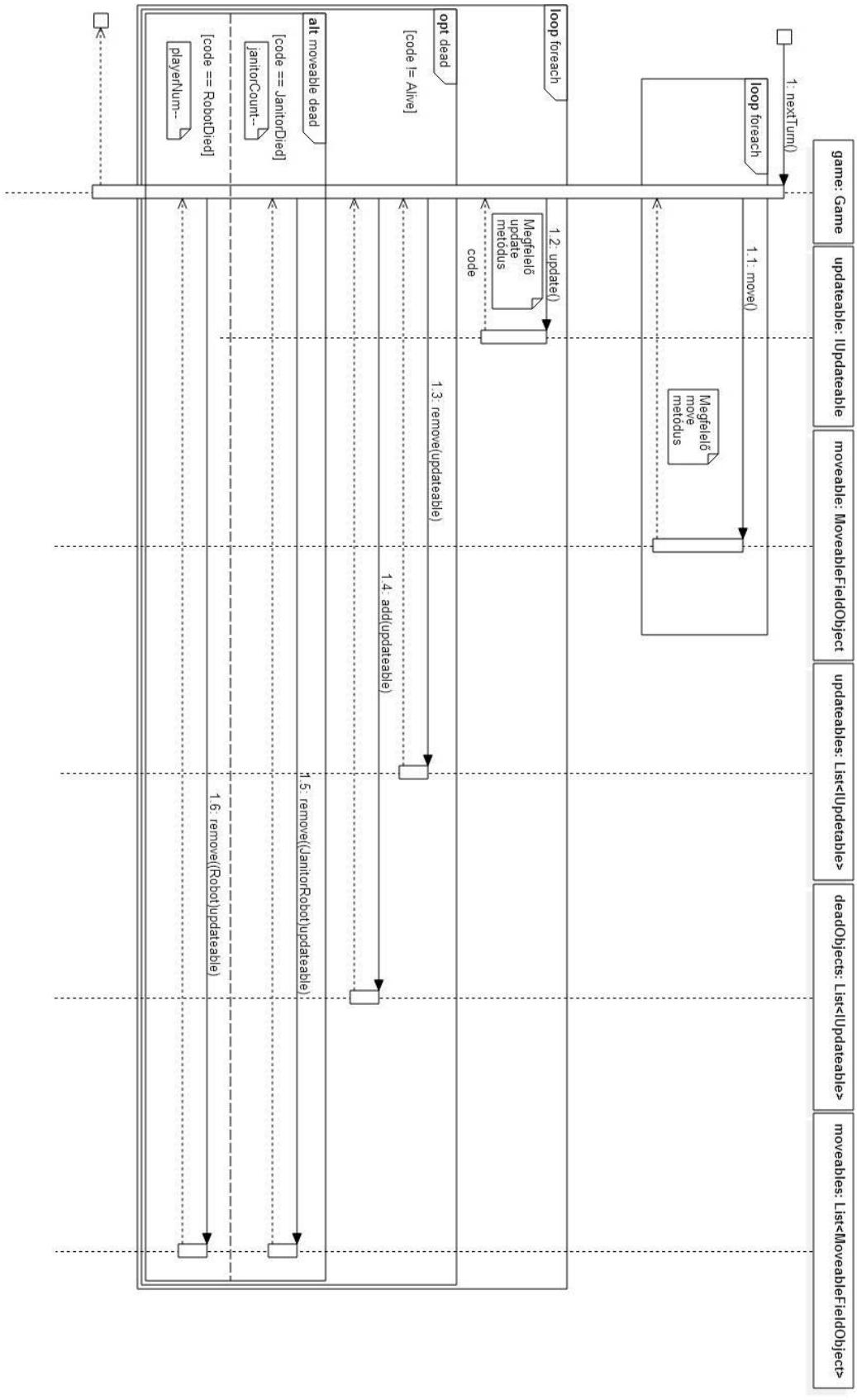
8-5. ábra Robotok lépésének beállítása



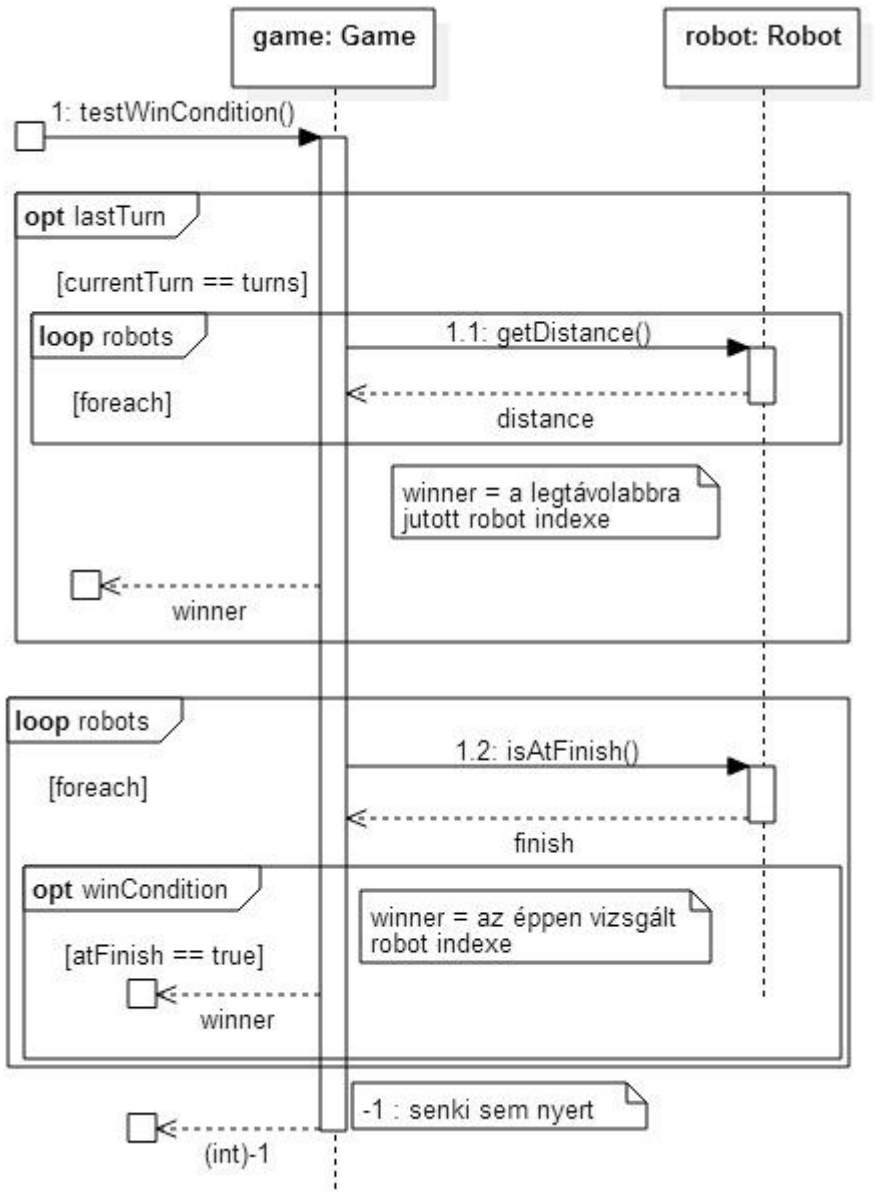
8-6. ábra Folt elhelyezése



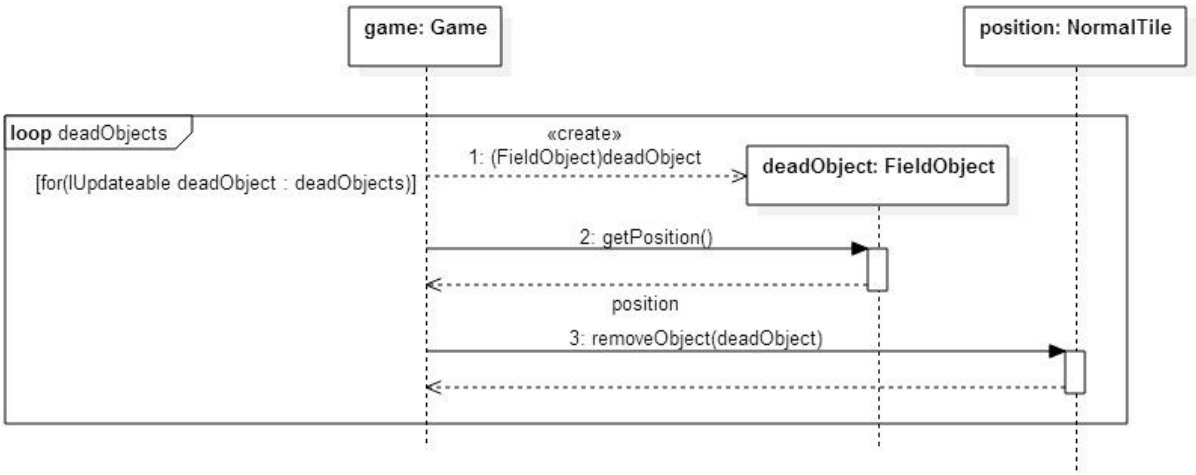
8-7. ábra Folt elhelyezése 2



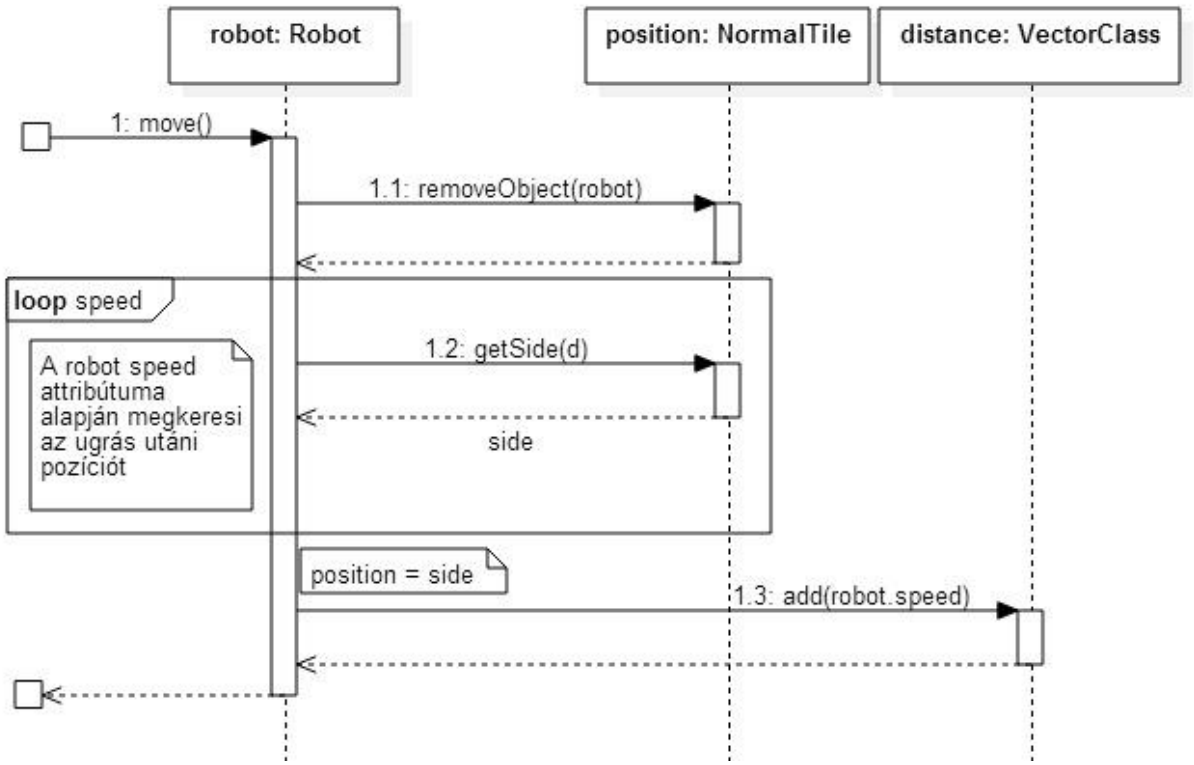
8-8. ábra Kör léptetése



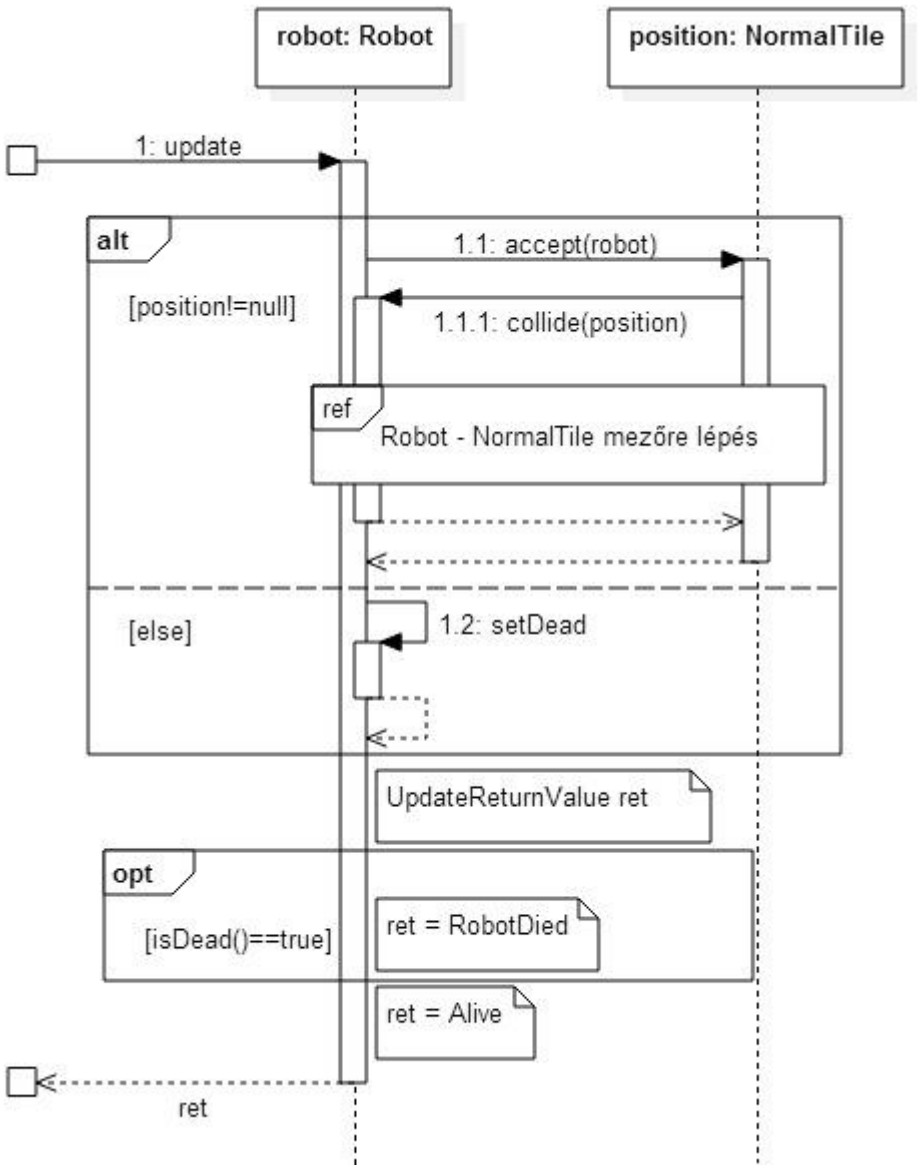
8-9. ábra Győzelem tesztelése



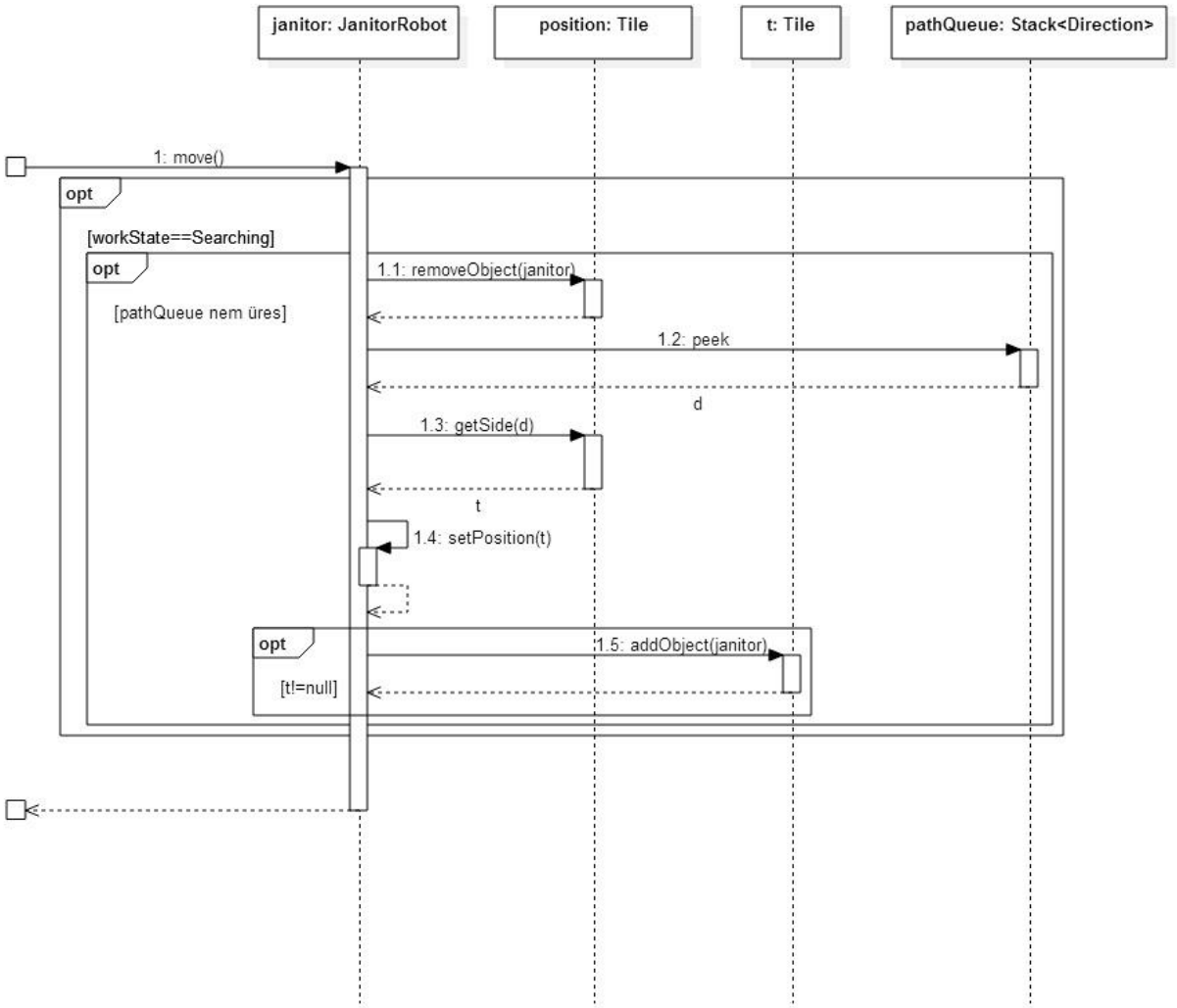
8-10. ábra Halott objektumok törlése



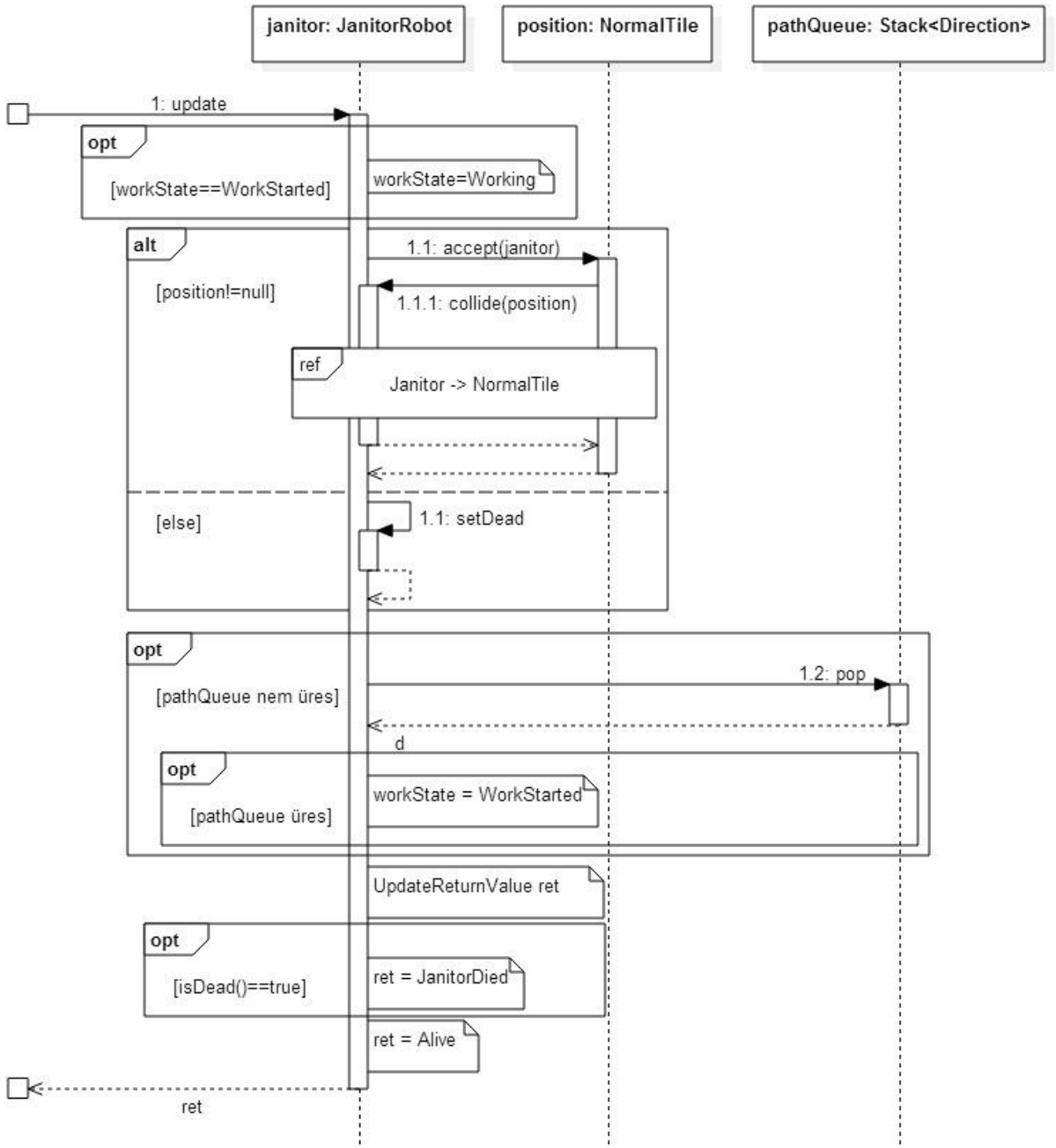
8-11. ábra Robot mozgatása



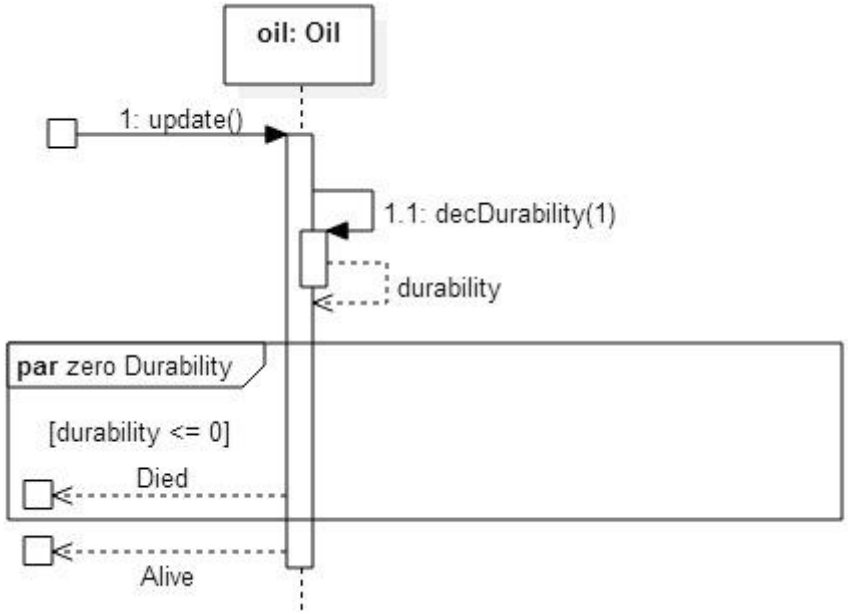
8-12. ábra Robot körének léptetése



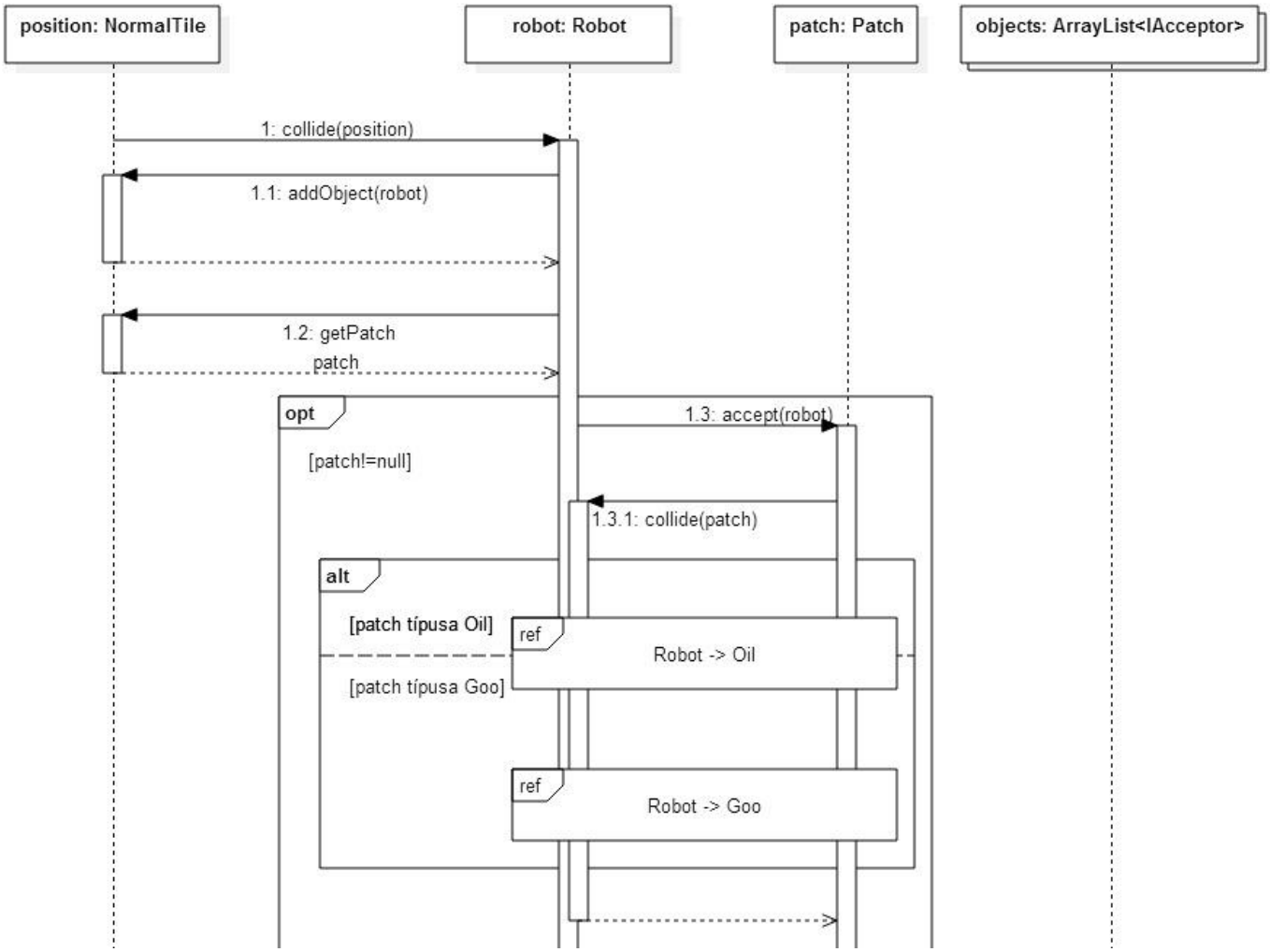
8-13. ábra Janitor mozgatása

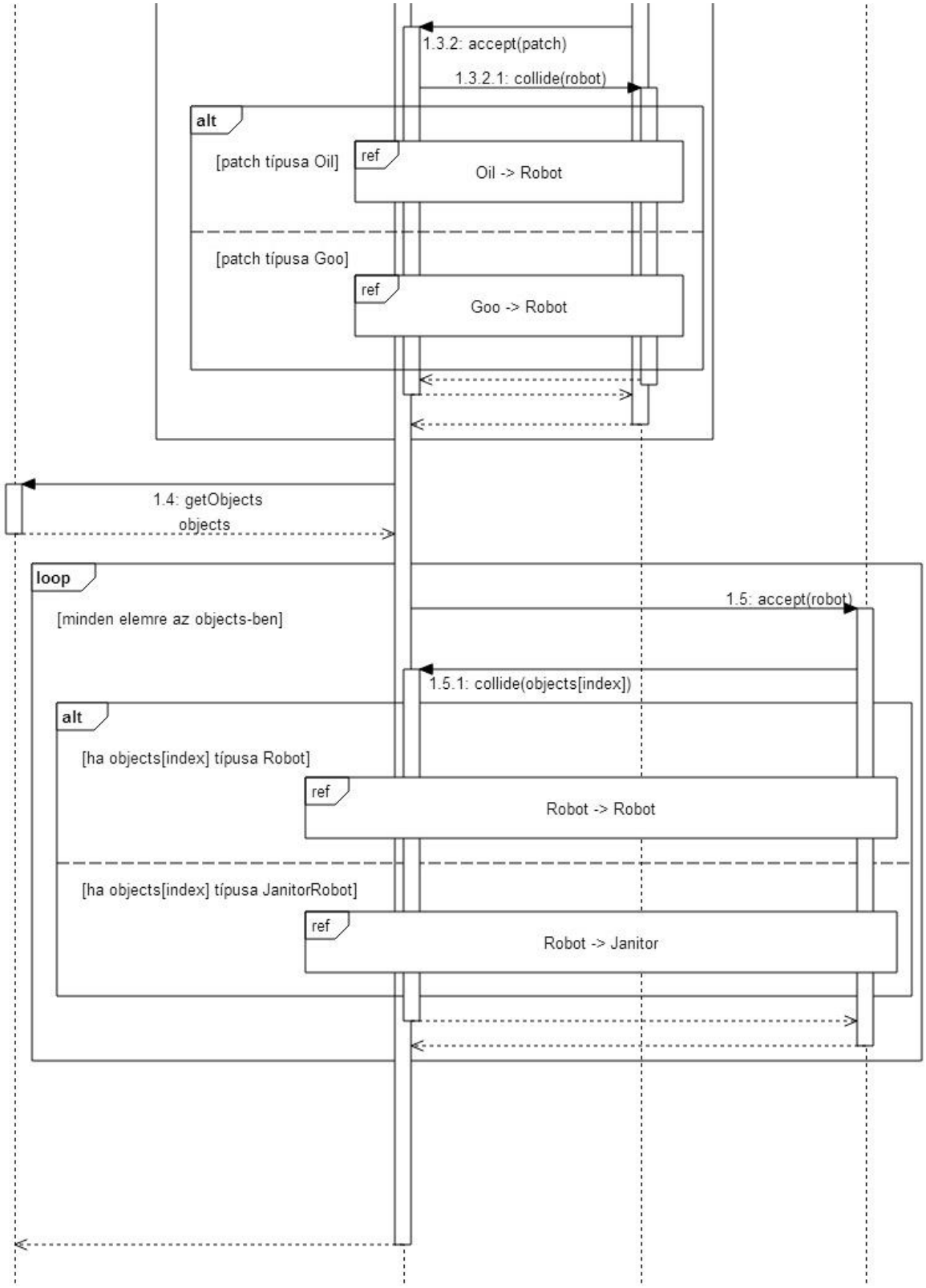


8-14. ábra Janitor körének léptetése

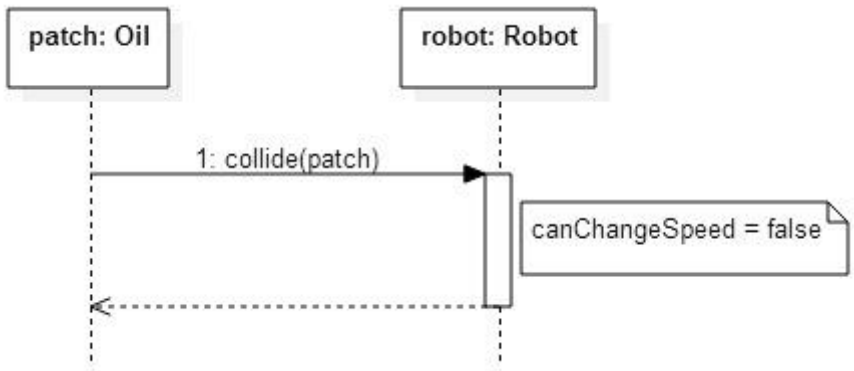


8-15. ábra Olajfolt körének léptetése

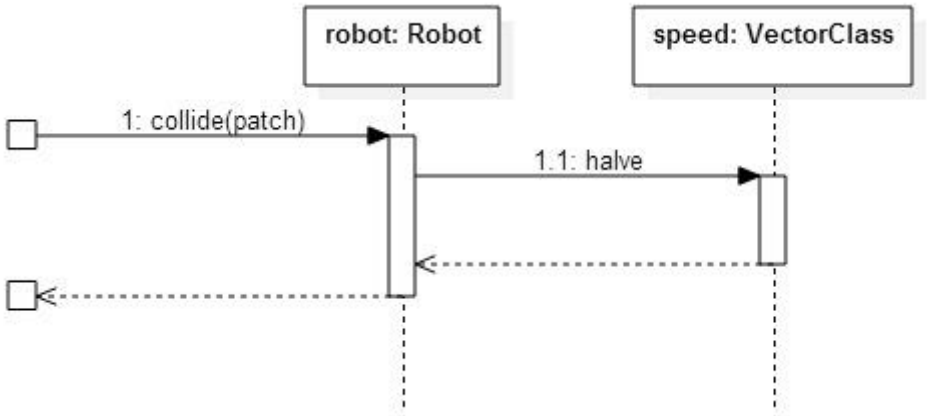




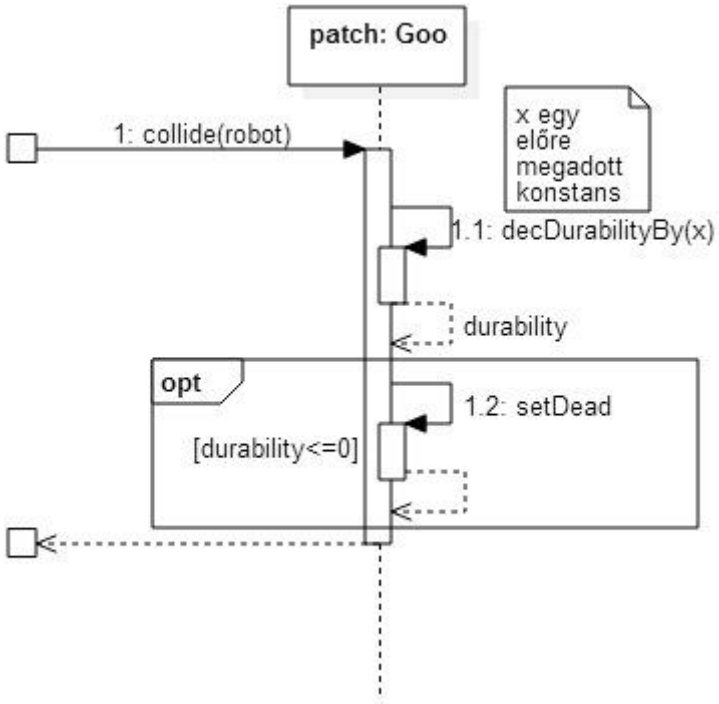
8-16. ábra Robot -> NormalTile



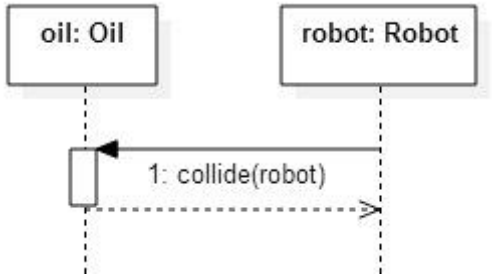
8-17. ábra Robot -> Oil



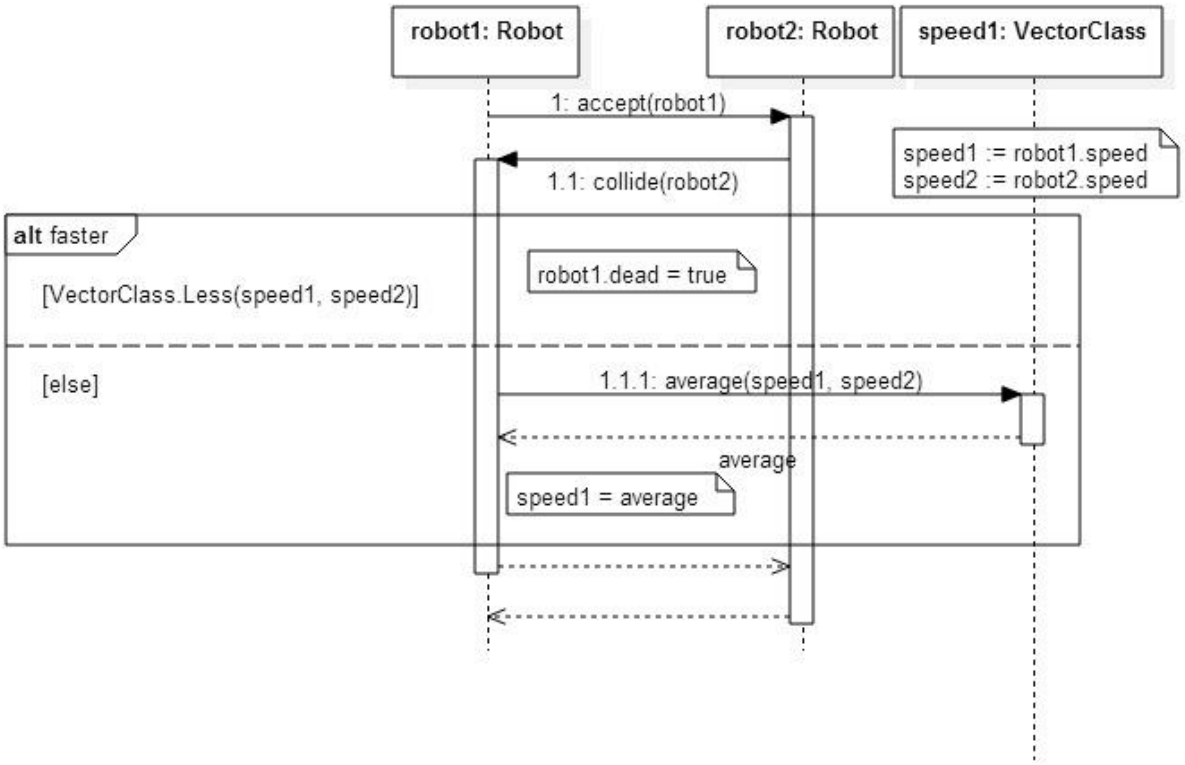
8-18. ábra Robot -> Goo



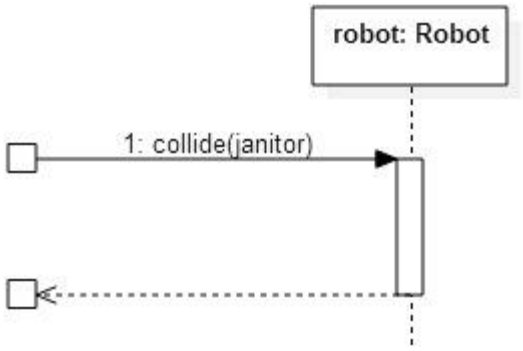
8-19. ábra Goo -> Robot



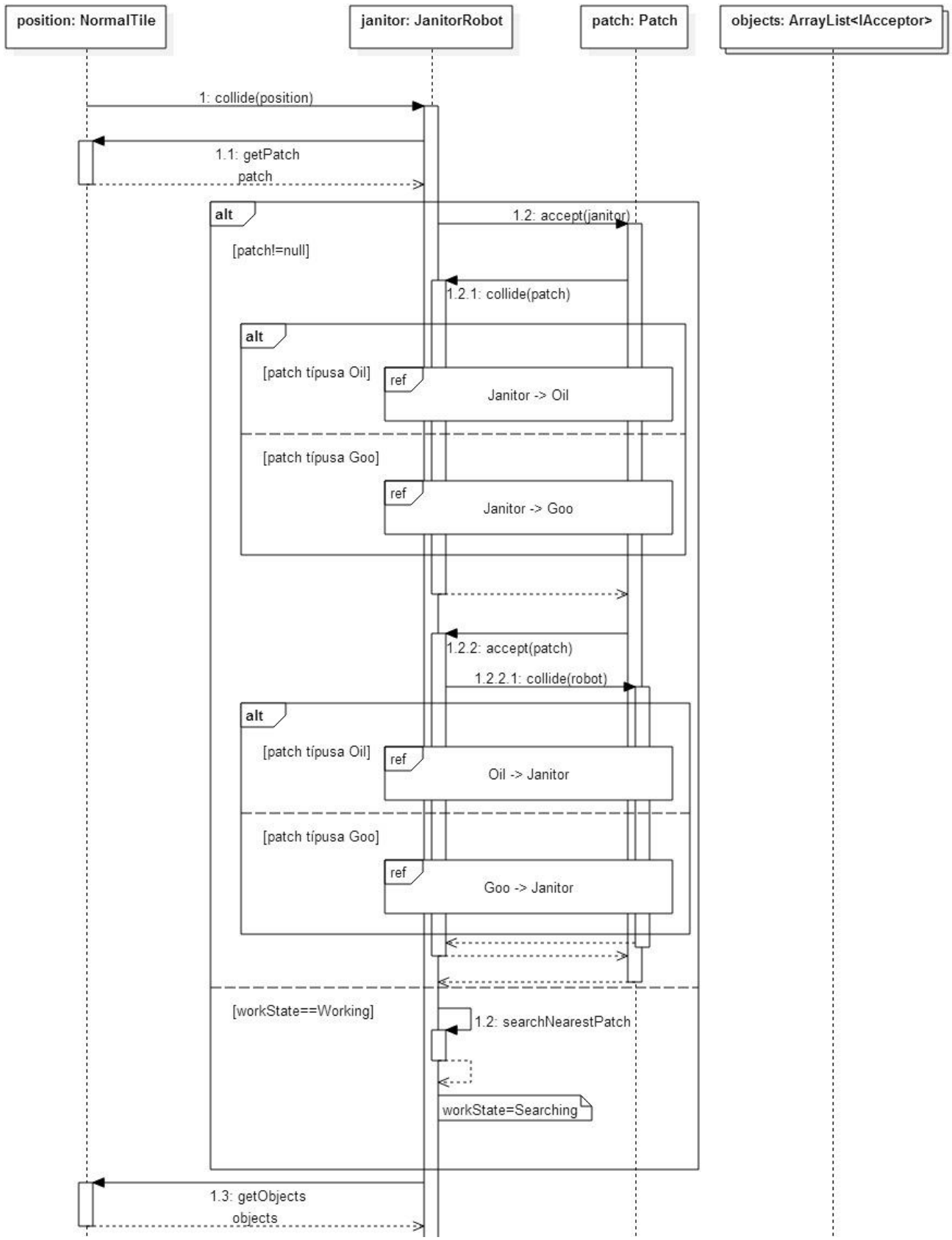
8-20. ábra Oil -> Robot

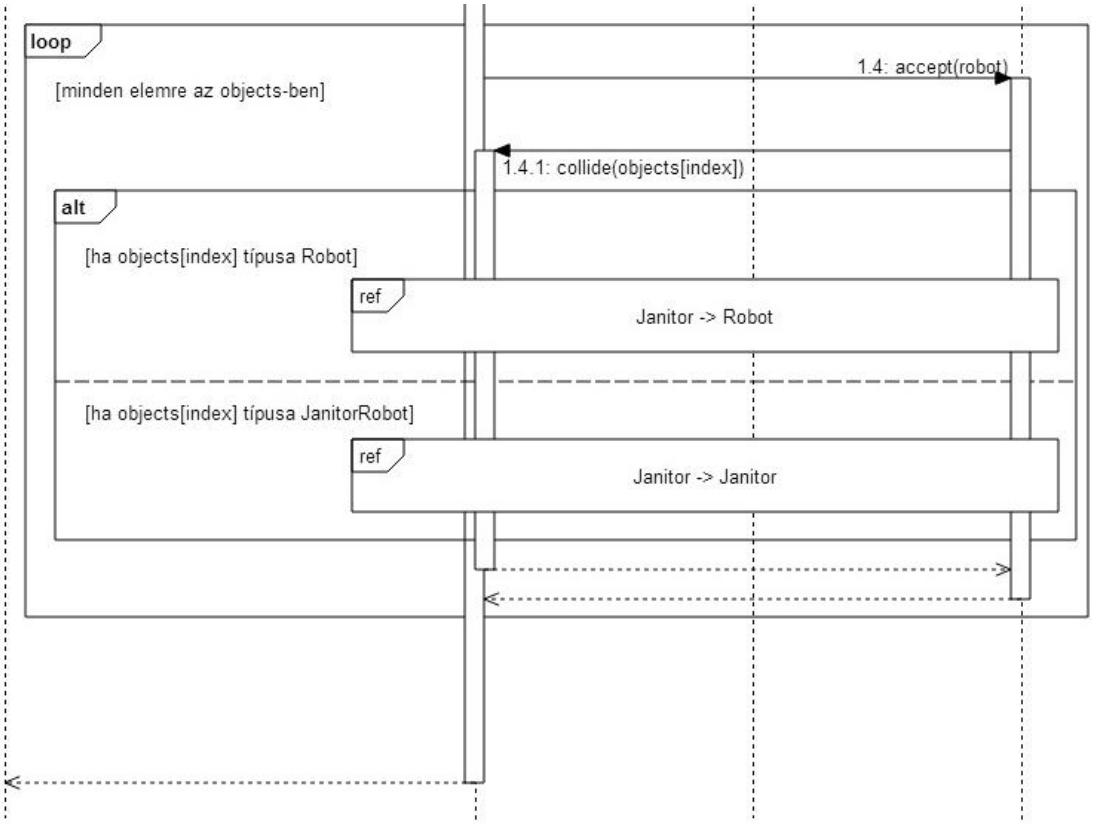


8-21. ábra Robot -> Robot

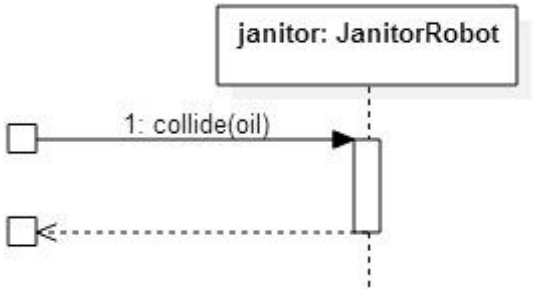


8-22. ábra Robot -> Janitor

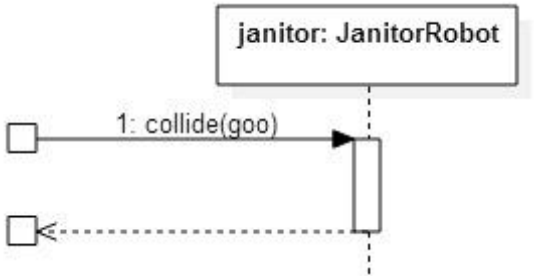

position: NormalTile
janitor: JanitorRobot
patch: Patch
objects: ArrayList<IAcceptor>
1: collide(position)
1.1: getPatch
patch
alt
[patch!=null]
1.2: accept(janitor)
1.2.1: collide(patch)
alt
[patch típusa Oil]
ref
Janitor -> Oil
[patch típusa Goo]
ref
Janitor -> Goo
1.2.2: accept(patch)
1.2.2.1: collide(robot)
alt
[patch típusa Oil]
ref
Oil -> Janitor
[patch típusa Goo]
ref
Goo -> Janitor
[workState==Working]
1.2: searchNearestPatch
workState=Searching
1.3: getObjects
objects



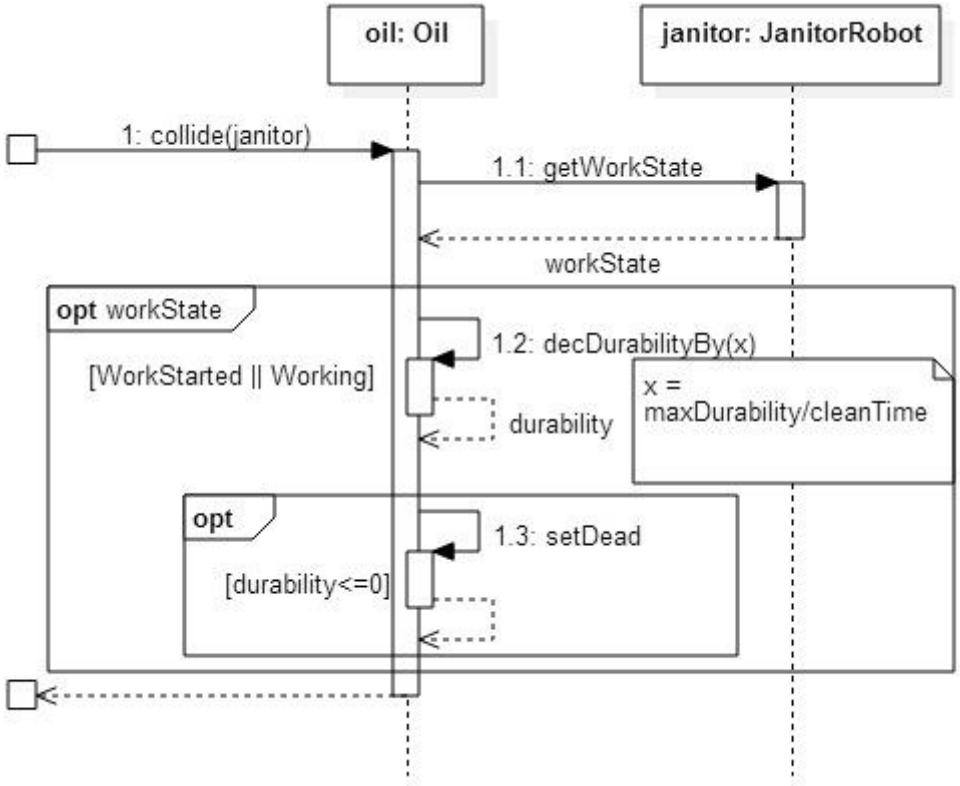
8-23. ábra Janitor -> NormalTile



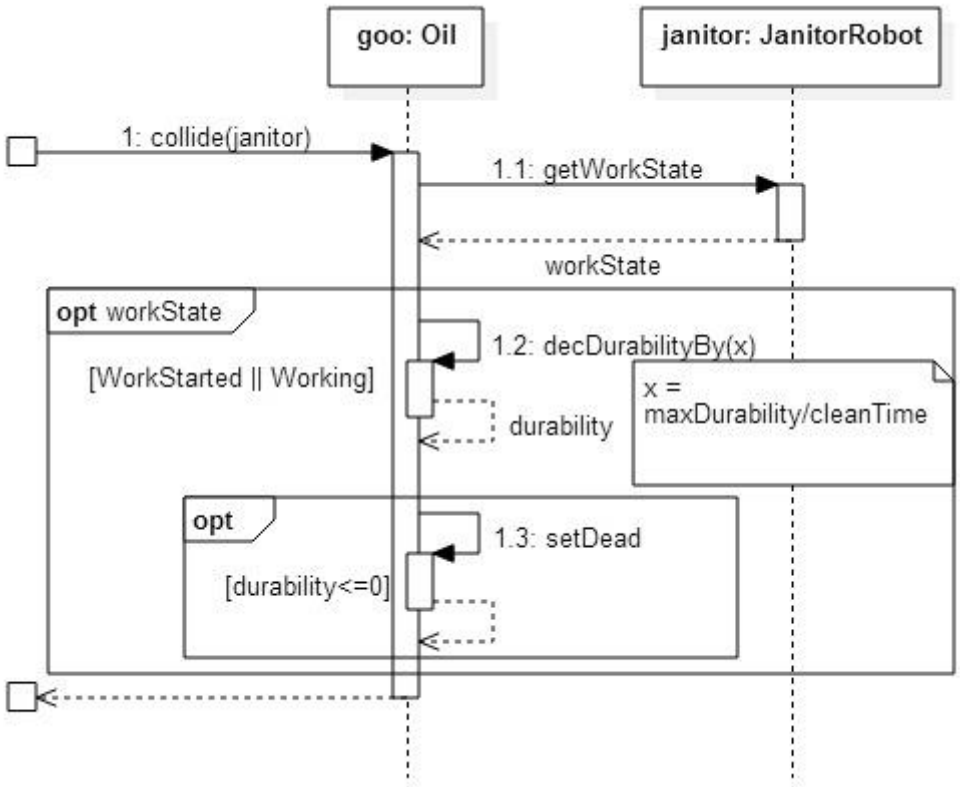
8-24. ábra Janitor -> Oil



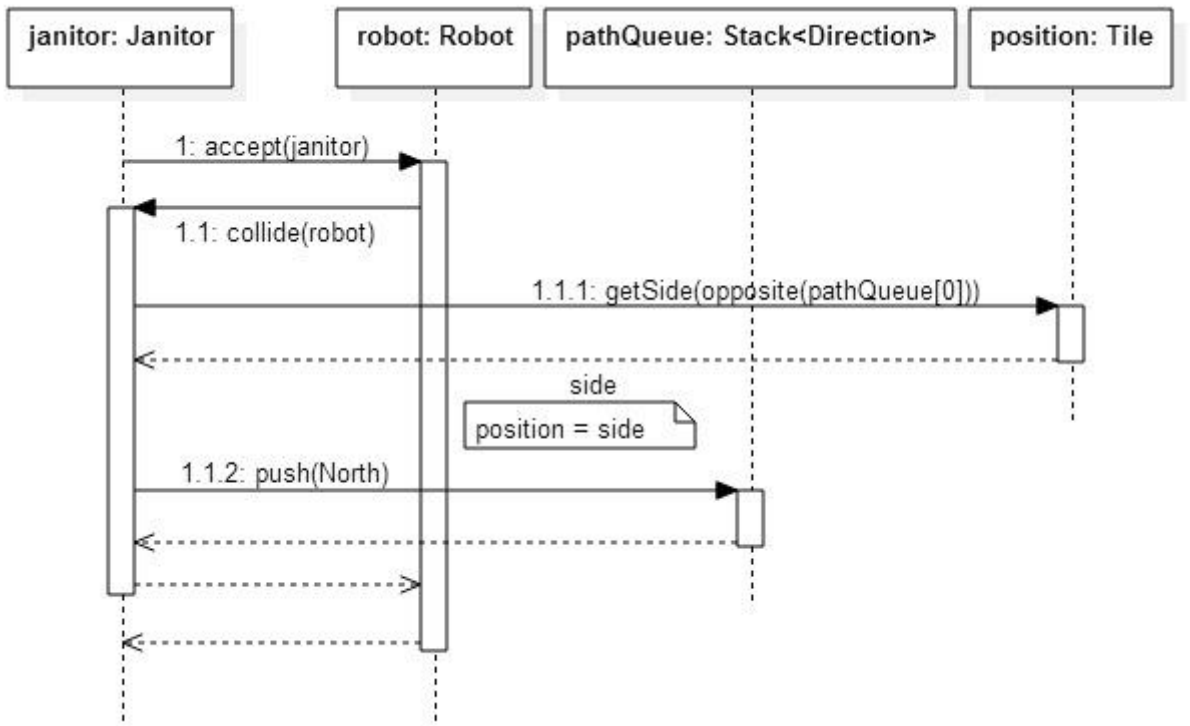
8-25. ábra Janitor -> Goo



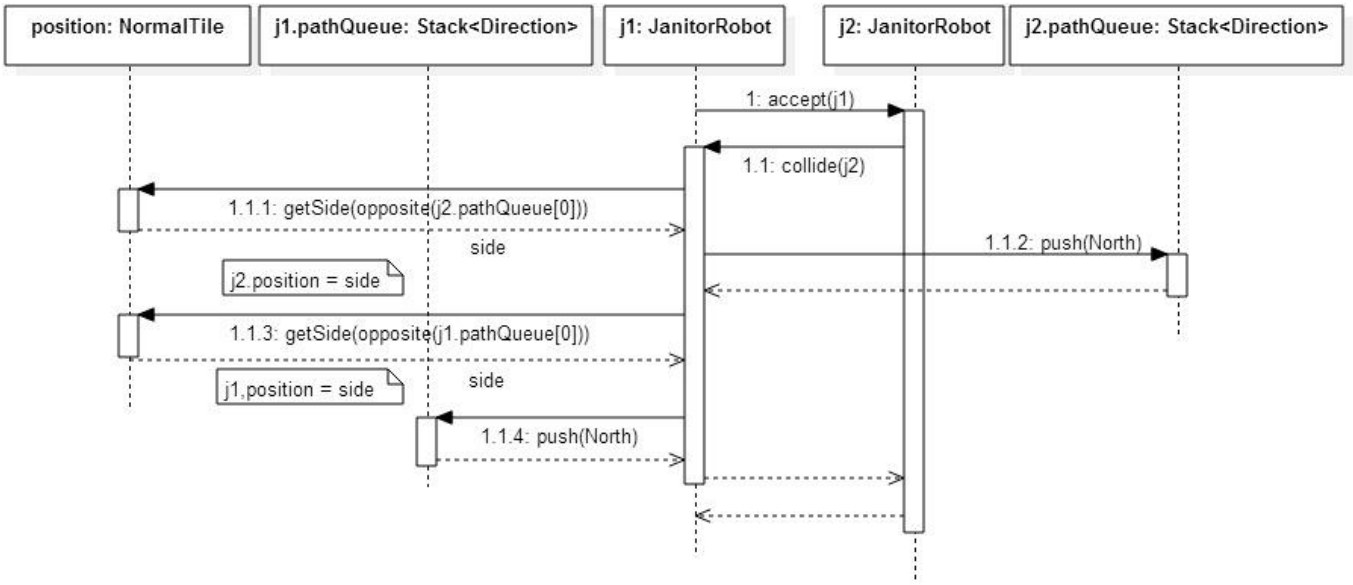
8-26. ábra Oil -> Janitor



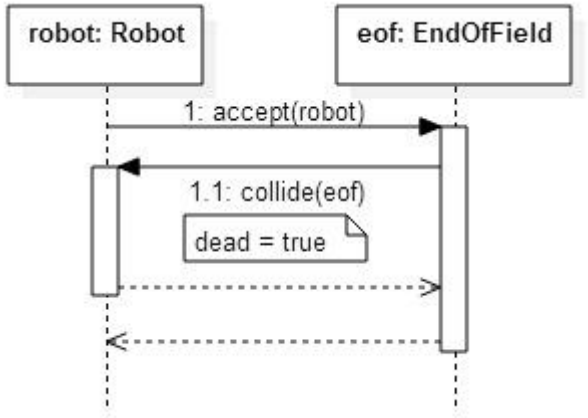
8-27. ábra Goo -> Janitor



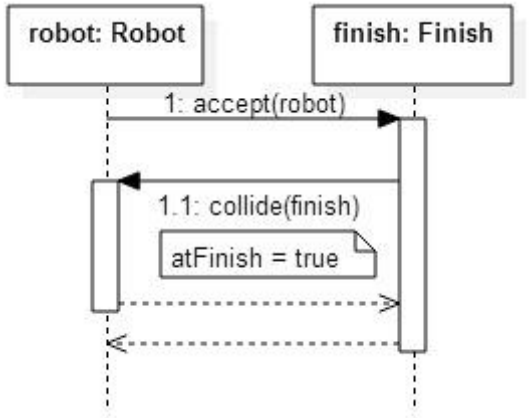
8-28. ábra Janitor -> Robot



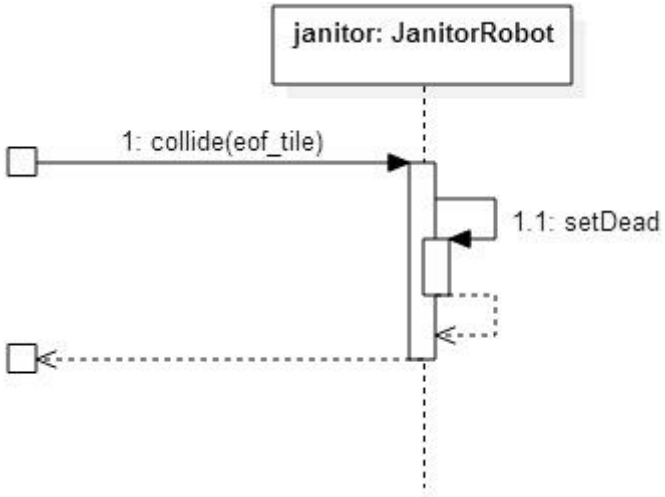
8-29. ábra Janitor -> Janitor



8-30. ábra Robot -> EndOfField



8-31. ábra Robot -> Finish



8-32. ábra Janitor -> EndOfField

8.2 A tesztek részletes tervei, leírásuk a teszt nyelvén

8.2.1 Új játék kezdése

- **Leírás:**
  Cél a játék helyes inicializációjának tesztelése, a robotok illetve azok tulajdonságainak helyes létrejöttének vizsgálata.

- **Ellenőrzött funkcionalitás, várható hibahelyek:**

A játék kezdete szempontjából a robotok fontos tulajdonságainak körébe tartozik hogy hány ragacs/olaj folt készlettel rendelkeznek, mekkora a sebességük illetve mi a kezdő pozíciójuk, ezeket fogjuk ellenőrizni. És természetesen azt is hogy maguk a robotok helyesen jönnek-e létre.
A legegyszerűbb de mégis kritikus hibalehetőség, hogy egyes robotok nem jönnek létre. Könnyebben előforduló de mégis súlyos hiba lehet, hogy egy robot rossz helyen éled, például a pályán kívül, ahol egyből meghal vagy épp egy ragacsfolton ami által egy robot hátrányba kerül. Ezen kívül természetesen előfordulhat az is hogy egyes robotok nem, vagy nem helyes mennyiségű olaj/ragacs készletet kapnak.

- **További információk / Megjegyzések**
- Az induló ragacs/olaj készletek száma fixen 3-3.

- **Bemenet:**
  - `newGame(30, 'new_game_test_map')`
  - `listRobot()`

- **Elvárt kimenet:**
  - `<0><1,1><3><3><0><0>`
  - `<1><2,2><3><3><0><0>`
  - `<2><3,3><3><3><0><0>`

- **Pálya [new_game_test_map] :**

```
<6><5>
<3>
<E><E><E><E><E><E>
<E><S><N><G><N><E>
<E><N><S><N><O><E>
<E><G><O><S><N><E>
<E><E><E><E><E><E>
```

8.2.2 Győztes robot tesztelése

- **Leírás:**
  Cél a nyerési feltételek létrejötte esetén, a győzelem helyes megvalósulásának letesztelése.

- **Ellenőrzött funkcionalítás, várható hibahelyek:**
  Fontos hogy amennyiben egy robot célba ér, akkor nyerje is meg a játékot, illetve ha letelik a játék maximális köreinek száma akkor az a robot nyerjen amelyik a legnagyobb távolságot tette meg. Kritikus hiba ha bármely győzedelmi feltétel teljesülése esetén az adott robot nem nyer, vagy esetlegesen másik robot nyeri meg a játékot. Nem szabad azt sem elfelejtenünk hogy győztes robot csak élő robot lehet hiába tett meg egy halott robot a legnagyobb távolságot.

- **További információk / Megjegyzések**
- Azonos körben való célba érés esetén a kisebb indexű (“fiatalabb”) robot nyer.

- **Bemenet:**
  - `newGame(30, 'next_turn_1_test_map')`
  - `setTurn(E,0)`

- `setTurn(E,0)`
- `nextTurn()`
- `getWinner()`
- `newGame(2, 'next_turn_2_test_map')`
- `setTurn(0,0)`
- `setTurn(E,0)`
- `nextTurn()`
- `nextTurn()`
- `getWinner()`
- `newGame(3, 'next_turn_3_test_map')`
- `setTurn(E,0)`
- `setTurn(0,0)`
- `nextTurn()`
- `setTurn(0,0)`
- `setTurn(E,0)`
- `nextTurn()`
- `nextTurn()`
- `getWinner()`

- **Elvárt kimenet:**
  - `-1`
  - `-1`
  - `-1`

- **Pálya 1 [next_turn_1_ test_map] :**

```
<2><2>
<2>
<S><F>
<S><F>
```

- **Pálya 2 [next_turn_2_ test_map] :**

```
<3><2>
<2>
<S><N><E>
<S><N><E>
```

- **Pálya 3 [next_turn_3_ test_map] :**

```
<5><2>
<2>
<S><N><N><N><E>
<S><N><N><N><E>
```

8.2.3 Kör léptetése

- **Leírás:**
  Cél a kör léptetés helyességének tesztelése. A sima/takarító robotok lépésének tesztelése, illetve az idő hatású eseményeké mint pl egy olajfolt felszáradása.

- **Ellenőrzött funkcionalítás, várható hibahelyek:**
  Alapvető funkcionális elvárás hogy a robotok illetve a takarító robotok csinálják meg a dolgukat (lépjenek) a kör léptetése során. Mivel ezt egy másik teszteset bővebben

lefedi itt nem foglalkozunk vele. Így a teszteset célja az olajfoltok felszáradásának a tesztelése lesz

- **További információk / Megjegyzések**
- Az olajfoltok 4 kör alatt száradnak fel.

- **Bemenet:**
  - `newGame(30, 'next_turn_test_map')`
  - `nextTurn()`
  - `nextTurn()`
  - `nextTurn()`
  - `nextTurn()`
  - `getMapTile(1,3)`
  - `getMapTile(3,1)`

- **Elvárt kimenet:**
  - `<N>`
  - `<N>`

- **Pálya [next_turn_test_map] :**

```
<6><5>
<0>
<E><E><E><E><E><E>
<E><S><N><O><S><E>
<E><N><S><N><G><E>
<E><O><G><S><N><E>
<E><E><E><E><E><E>
```

8.2.4 Ragacsfolt/Olajfolt hagyása

- **Leírás:**
  Cél a a robotok általi folthagyás helyességének letesztelése.

- **Ellenőrzött funkcionalítás, várható hibahelyek:**
  Alapvető funkcionalítási elvárás hogy a robotok tudjanak foltot hagyni, illetve azt a foltot (olaj/ragacs) hagyják maguk után amit szeretnének. Hibalehetőség lehet hogy akkor is tudnak foltot hagyni a robotok ha nem rendelkeznek megfelelő készlettel (0 a készletük az adott típusú foltból).

- **Bemenet:**
  - `newGame(30, 'patch_test_map')`
  - `setTurn(E,G)`
  - `nextTurn()`
  - `setTurn(0,O)`
  - `nextTurn()`
  - `setTurn(0,O)`
  - `nextTurn()`
  - `setTurn(0,G)`
  - `nextTurn()`
  - `setTurn(0,G)`
  - `nextTurn()`

- `setTurn(0,O)`
- `nextTurn()`
- `setTurn(0,G)`
- `nextTurn()`
- `setTurn(0,O)`
- `nextTurn()`
- `getMapTile(0,0)`
- `getMapTile(1,0)`
- `getMapTile(0,6)`
- `getMapTile(0,7)`
- `getMapTile(0,8)`

- **Elvárt kimenet:**
  - `<G>`
  - `<N>`
  - `<O>`
  - `<N>`
  - `<N>`

- **Pálya [patch_test_map] :**

```
<10><1>
<1>
<S><N><N><N><N><N><N><N><N><N>
```

8.2.5 Robotok mozgatása

- **Leírás:**
  Cél a robot lépésének (ugrás/jump) a helyes mükődésének a letesztelése

- **Ellenőrzött funkcionalitás, várható hibahelyek:**
  Alapvető elvárás hogy a robotok az elvártaknak megfelelően lépjenek a kör léptetésekor. Egy robot 4 Irányba tud irányt/sebességet változtatni : Észak, Kelet, Dél, Nyugat. Hibás működés lehet hogyha rossz irányba megy egy robot, vagy nem megfelelő sebességgel teszi azt.

- **Bemenet:**
  - `newGame(30, 'jump_test_map')`
  - `listRobot()`
  - `setTurn(N,0)`
  - `nextTurn()`
  - `listRobot()`
  - `setTurn(S,0)`
  - `nextTurn()`
  - `setTurn(S,0)`
  - `nextTurn()`
  - `listRobot()`
  - `setTurn(N,0)`
  - `nextTurn()`
  - `setTurn(E,0)`
  - `nextTurn()`
  - `listRobot()`

- `setTurn(W,0)`
- `nextTurn()`
- `setTurn(W,0)`
- `nextTurn()`
- `listRobot()`

- **Elvárt kimenet:**
  - `<0><0,1><3><3><0><0>`
  - `<0><0,0><3><3><1><1>`
  - `<0><0,1><3><3><2><1>`
  - `<0><1,1><3><3><3><1>`
  - `<0><0,1><3><3><4><1>`

- **Pálya [jump_test_map] :**
  ```
  <2><2>
  <1>
  <N><N>
  <S><N>
  ```

8.2.6 Takarító robotok mozgatása

- **Leírás:**
  Cél a takarítórobot lépésének a helyes mükődésének a leteszetelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek:**
  Egy takarítórobot a legközelebbi folt felé mozog. Ezért hibalehetőség lehet hogy rossz folt felé indul el, vagy nem is folt felé megy.

- **Bemenet:**
  - `newGame(30, 'janitor_jump_test_map')`
  - `spawnJanitor(3,1)`
  - `listJanitor()`
  - `nextTurn()`
  - `listJanitor()`
  - `nextTurn()`
  - `listJanitor()`

- **Elvárt kimenet:**
  - `<0><3,1><0><5,1>`
  - `<0><4,1><0><5,1>`
  - `<0><5,1><4><0,0>`

- **Pálya [janitor_jump_test_map] :**
  ```
  <6><2>
  <0>
  <N><N><N><S><N><N>
  <O><N><N><S><N><O>
  ```

8.2.7 Leesés

- **Leírás:**
  Cél annak tesztelése hogy a robot meghal-e ha leesik a pálya széléről.

- **Ellenőrzött funkcionalitás, várható hibahelyek:**
  Alapvető funkcionalítási elvárás hogy a robot meghaljon. Egyszerű hibalehetőség hogy a robot nem hal meg, és tovább tud közlekedni. Vagy csak életben marad, de irányíthatatlanná válik.

- **További információk / Megjegyzések**
- Egy robot már nem létezik ha meghalt, így a listRobot nem fogja listázni már.

- **Bemenet:**
  - `newGame(30, 'fall_down_test_map')`
  - `listRobot()`
  - `setTurn(E,0)`
  - `nextTurn()`
  - `nextTurn()`
  - `listRobot()`

- **Elvárt kimenet:**
  - `<0><0,0><3><3><0>`
  -

- **Pálya [fall_down_test_map] :**
  `<3><1>`
  `<1>`
  `<S><S><E>`

8.2.8 Folt feltakarítás

- **Leírás:**
  A takarítórobot feltakarít egy olajfoltot. A foltos mező normális mezővé alakul. Ezután a robot továbbhalad a következő takarítandó folt felé.

- **Ellenőrzött funkcionalitás, várható hibahelyek:**
  Hibalehetőségek: Nem alakul a mező rendes mezővé a takarítás elkészülte után, illetve lassabban, vagy gyorsabban takarítja fel a takarítórobot a foltot a kelleténél.

- **Bemenet:**
  - `newGame("janitor_clean_test_map)`
  - `spawnJanitor(0,0)`
  - `listJanitor`
  - `nextTurn`
  - `listJanitor`
  - `nextTurn`
  - `nextTurn`
  - `nextTurn`

  - `nextTurn`
  - `listJanitor`

- **Elvárt kimenet:**
  - `<0><0,0><0><1,0>`
  - `<0><1,0><4><1,0>`
  - `<0><1,0><0><2,0>`

- **Pálya [janitor_clean_test_map] :**
  ```
  <3><1>
  <0>
  <S><0><0>
  ```

8.2.9 Olajfoltra lépés

- **Leírás:**
  A Robot egy olajfoltra lép, és a következő körben nem tudja változtatni a sebességét.

- **Ellenőrzött funkcionalitás, várható hibahelyek:**
  Elvárt funkcionalítás hogy hiába próbáljuk a Robot sebességét módosítani, az ugyanolyan maradjon. Hibalehetőség: A Robot sebességét mégis meg tudjuk változtatni

- **Bemenet:**
  - `newGame(”oil_test_map)`
  - `listRobot`
  - `setTurn(E,0)`
  - `nextTurn`
  - `listRobot`
  - `setTurn(N,0)`
  - `nextTurn`
  - `listRobot`

- **Elvárt kimenet:**
  - `<0><0,1><3><3><0><0>`
  - `<0><1,1><3><3><1><1>`
  - `<0><2,1><3><3><2><1>`

- **Pálya [oil_test_map] :**
  ```
  <3><2>
  <1>
  <E><N><E>
  <S><O><N>
  ```

8.2.10 Ragacsfoltra lépés

- **Leírás:**
  A Robot ragacsfoltra lép és emiatt elveszíti a sebességének a felét.

- **Ellenőrzött funkcionalitás, várható hibahelyek:**

Hibalehetőség: A Robot „nem ragad be” azaz nem feleződik a sebessége, miután rálépett a ragacsfoltra, vagy rosszul módosul a sebessége.

- **Bemenet:**
  - `newGame(”goo_test_map”)`
  - `listRobot`
  - `setTurn(E,0)`
  - `nextTurn`
  - `setTurn(E,0)`
  - `nextTurn`
  - `listRobot`
- **Elvárt kimenet:**
  - `<0><0,0><3><3><0><0>`
  - `<0><3,0><3><3><3><1>`
- **Pálya [oil_test_map] :**

```
<4><1>
<1>
<S><N><N><G>
```

8.2.11 Robot-Robot ütközés

- **Leírás:**
  Két robot lesz a pályán, és az egyik nekiütközik a másiknak. A várt esemény az, hogy a lassabb robot meghal, és csak a gyorsabb marad a mezőn. Aminek sebessége pedig a két robot sebességének vektorátlagára változik.
- **Ellenőrzött funkcionalitás, várható hibahelyek:**
  Alapvető funkcionalítási elvárás hogy a gyorsabb robot élje túl, a lassabb pedig semmisüljön meg. A legalapvetőbb hiba lehetőség az ha egyik robot sem semmisül meg, vagy akár mindkettő, vagy egyiksem. További hibalehetőség még, a megmaradt robot hibás sebességmódosúlása.
- **Bemenet:**
  - `newGame(”robot_robot_interaction_test_map”)`
  - `listRobot`
  - `setTurn(0, 0)`
  - `setTurn(W, 0)`
  - `nextTurn`
  - `nextTurn`
  - `listRobot`
- **Elvárt kimenet:**
  - `<0><0,0><3><3><0><0>`
  - `<1><2,0><3><3><0><0>`
  - `<1><0,0><3><3><2><1>`
- **Pálya [robot_robot_interaction_test_map] :**

```
<2><1>
<2>
<S><N><S>
```

8.2.12 Robot -> takarítórobot interakció

- **Leírás:**
  A robot összeütközik a takarítórobottal, ami ennek hatására megsemmisül.

- **Ellenőrzött funkcionalitás, várható hibahelyek:**
  Alapvető elvárás hogy a takarítórobot megsemmisüljön, a robotra pedig ne keltsen befolyást az interakció. A mezőn pedig olajfolt keletkezzen az ütközés után.

- **Bemenet:**
  - `newGame(”robot_janitor_interaction_test_map”)`
  - `spawnJanitor(1,1)`
  - `listJanitor`
  - `listRobot`
  - `setTurn(S,0)`
  - `nextTurn`
  - `listJanitor`
  - `listRobot`
  - `getMapTile(1,1)`

- **Elvárt kimenet:**
  - `<0><1,1><0><0,0>`
  - `<0><1,0><3><3><0><0>`
  - `<0><1,1><3><3><1><1>`
  - `<O>`

- **Pálya [robot_janitor_interaction_test_map] :**

```
<3><2>
<1>
<N><S><N>
<S><N><O>
```

8.2.13 Takarító robot - takarító robot, takarító robot -> robot interakció

- **Leírás:**
  A takarító robot nekiütközik egy robotnak, irányt vált és más irányba indul tovább.

- **Ellenőrzött funkcionalitás, várható hibahelyek:**
  Hibalehetőség: A takarítórobot nem vált irányt és a Robot mellett ugyanazon a mezőn fog állni.

- **Bemenet:**
  - `newGame(”janitor_robot_interaction_test_map”)`
  - `spawnJanitor(2,0)`
  - `listJanitor`
  - `listRobot`

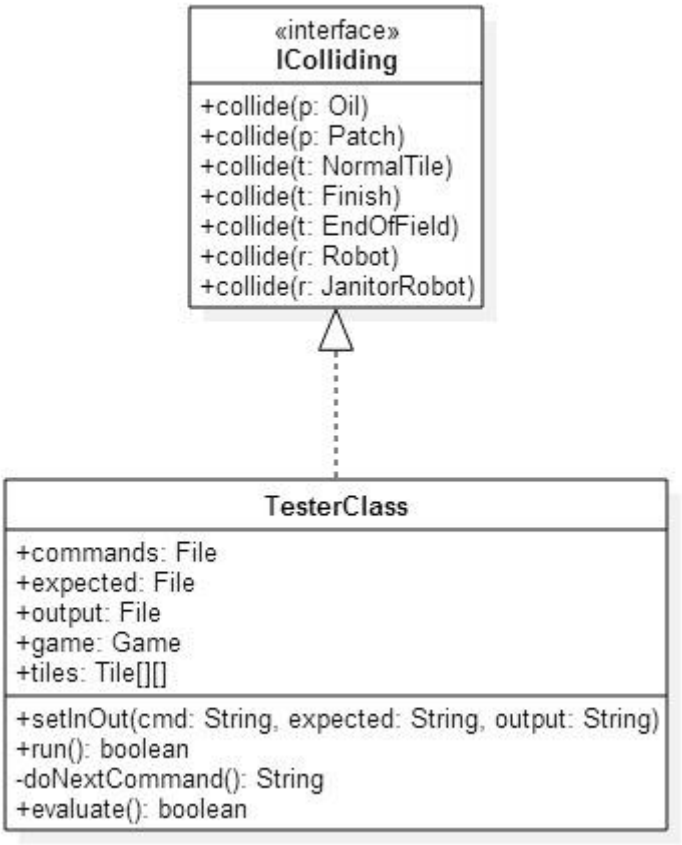
    - `setTurn(0,0)`
    - `nextTurn`
    - `listJanitor`
    - `listRobot`

- **Elvárt kimenet:**
    - `<0><2,0><0><0,0>`
    - `<0><1,0><3><3><0><0>`
    - `<0><2,1><0><0,0>`
    - `<0><1,0><3><3><0><0>`

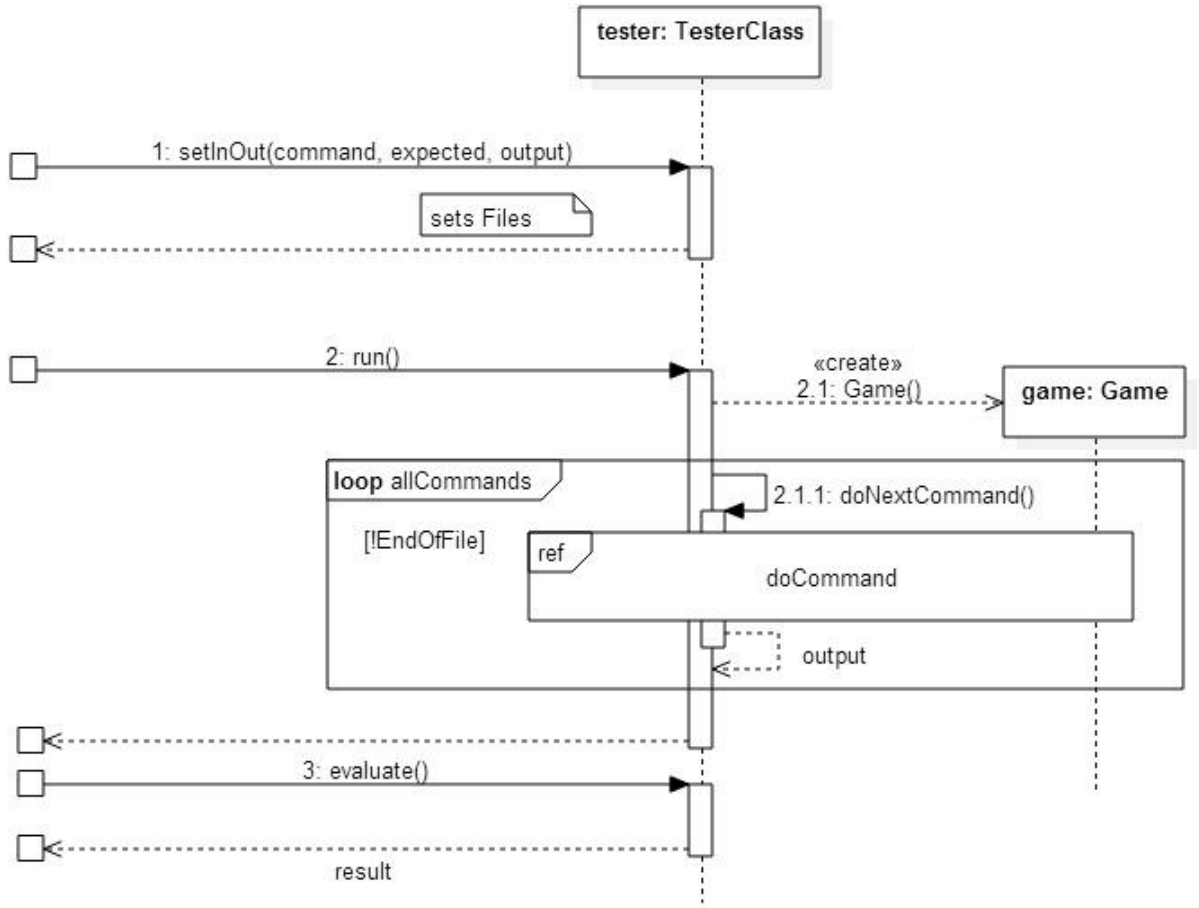
- **Pálya [janitor_robot_interaction_test_map] :**

```
<3><2>
<1>
<O><S><S>
<N><N><N>
```

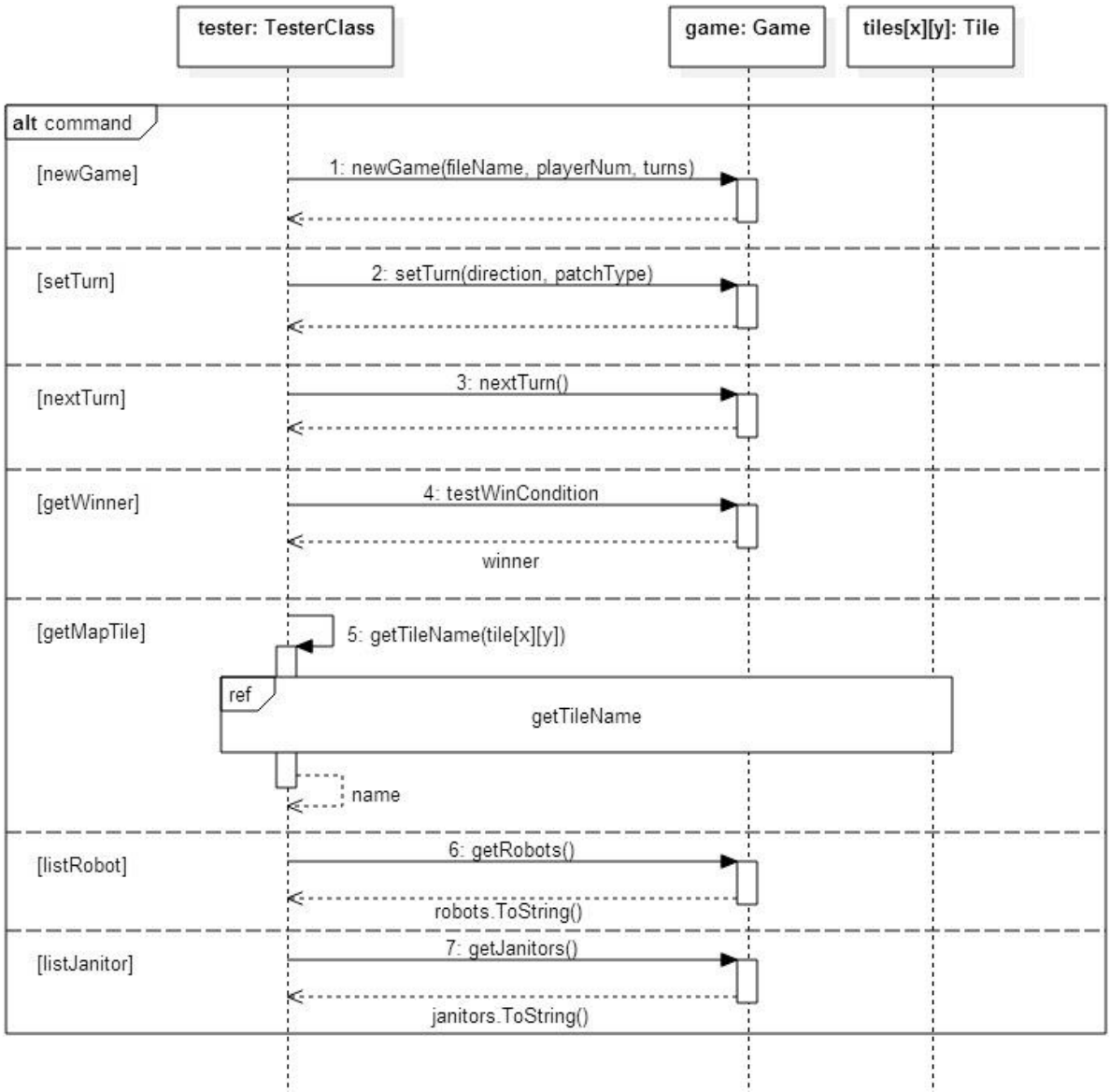
8.3 A tesztelést támogató programok tervei



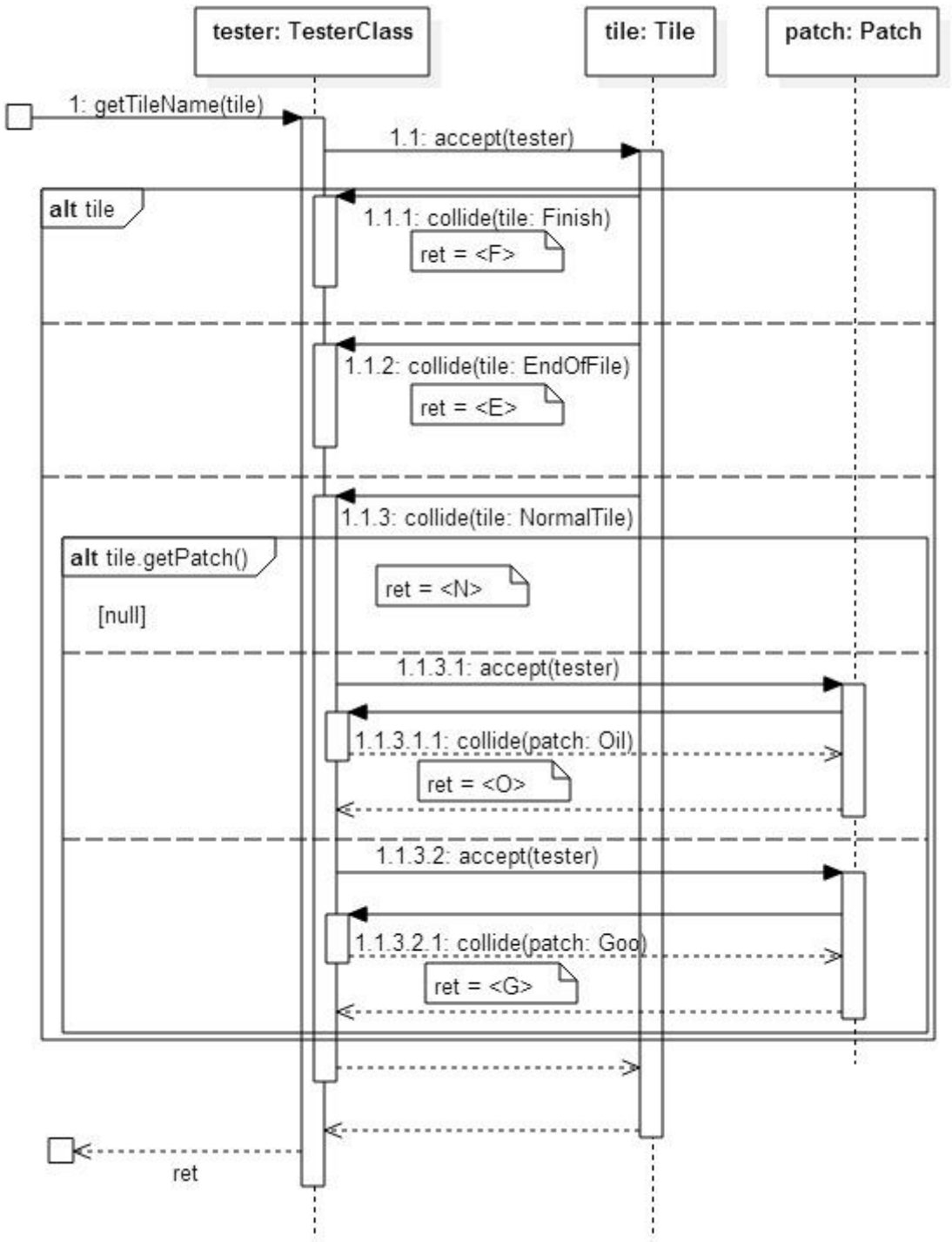
8-33. ábra TesterClass



8-34. ábra Test main



8-35. ábra Parancs végrehajtása



8-36. ábra Mező beazonosítása

8.4 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2015.04.02. (csütörtök) 18:30	1,5 óra	Király Dankó Dobosy Szabó Kovács	Feladatok kiosztása, feladat átbeszélése
2015.04.04 (szombat) 15:30	0.5 óra	Dankó	Tesztelést segítő program
2015.04.04 (szombat) 22:00	2 óra	Kovács	Osztályok leírása
2015.04.06 (hétfő) 15:00	1,5 óra	Kovács	Osztályok leírása
2015.04.06 (hétfő) 15:00	5 óra	Szabó	Osztálydiagram és szekvenciadiagramok módosítása, új szekvenciadiagramok készítése
2015.04.06 (hétfő) 12:00	4.5 óra	Király / Dobosy	Tesztesetek
2015.04.06 (hétfő) 15:00	5 óra	Dankó	Osztálydiagram és szekvenciadiagramok módosítása, új szekvenciadiagramok készítése