# Követelmény, projekt, funkcionalitás

## Bevezetés

### Cél

A Phoebe nevű játék specifikációnak[[1]](#footnote-1) megfelelő elkészítése a cél. A játékban robotok versenyeznek, az nyer, aki a leghosszabb távot tudja megtenni adott idő alatt.

### Szakterület

Általános használat, játék

### Definíciók, rövidítések

MVC – Model-view-controller: háromrétegű szoftver architektúrális pattern[[2]](#footnote-2)

IIT – Irányítástechnikai és Informatikai Tanszék

JRE – Java Runtime Environment: a java programokat futtató környezet

### Hivatkozások

<https://www.iit.bme.hu/~szoftlab4/>

<http://docs.oracle.com/javase/7/docs/api/>

### Összefoglalás

**Általános áttekintés**: MVC modell használata a projektben, a pálya txt fájlban tárolása

**Funkciók:** A robotok mezőkre osztott pályán egység kezdeti sebességgel mozognak. A cél a startvonaltól minél messzebb kerülni, mielőtt lejár az idő. A robot irányítható, gyorsítható, lassítható, illetve akadályokat tud elhelyezni. A játék véget ér az idő lejártával, illetve véget érhet még robotok ütközése miatt is.

**Felhasználók:** Természetes személyek

**Korlátozások:** JRE (1.7-es Java Runtime Environment) telepítve kell legyen

**Feltételezések, kapcsolatok:** A dokumentum megértését segítő anyagok

**Funkcionális követelmények:** A játék használati funkcióinak áttekintése

**Erőforrásokkal kapcsolatos követelmények:** A használat előfeltételei

**Átadással kapcsolatos követelmények:** Szükséges tevékenységek a használathoz

**Egyéb nem funkcionális követelmények:** Lényeges, de nem kifejezetten használati funkcionalitások. Közvetett-, vagy nincs kapcsolat a felhasználó tevékenységeivel.

**Use-case leírások:** A funkcionalitásokhoz tartozó use-case-ek leírásai

**Use-case diagram:** A játék használati esetének diagramja

**Szótár:** A használt kifejezések magyarázata

**Projekt:** Leadási határidők táblázatos formában

## Áttekintés

### Általános áttekintés

A program többrétegű architektúrát használ, vagyis MVC modellt. Ez a modell alapvetően 3 különböző részre bontja a szoftvert.

Modell réteg: a rendszer viselkedéséért, logikájáért felel. Ez a réteg számolja például az életerő csökkenését, és az ellenfelek mozgatását.

View réteg: Megjeleníti a modellt egy megfelelő alakban. Azért a kép kialakításáért felelős, amit a felhasználó a monitorán keresztül látni fog

Controller: a felhasználói műveleteket dolgozza fel.

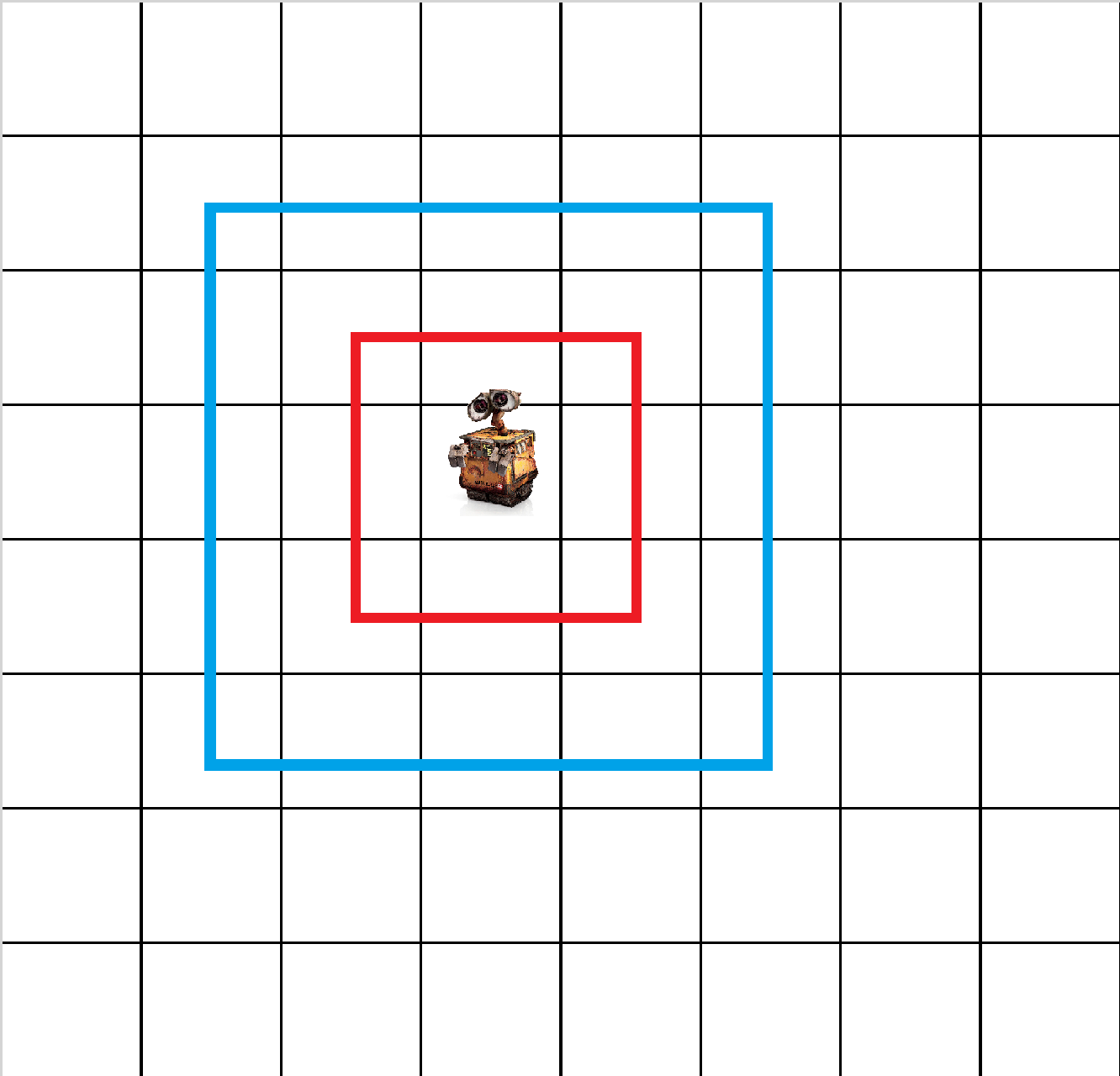
A pályákat txt fájlokban fogjuk tárolni, a program innen fogja betölteni a játék elején.

### Funkciók

A megrendelő, az Irányítástechnika és Informatikai Tanszék (IIT), egy „Phoebe” nevű játékot rendelt meg. Az általuk megadott leírásban nem szerepel a robotok pontos irányítása, így ennek meghatározása a csapat feladata.

A játék egy ügyességi típusú, erősen kompetitív játék, melyben a MarsOne telepeseit személyesítheti meg a játékos. Maximum négy versenyző mérheti össze képességeit. A játékos célja a lehető legnagyobb távolságot megtenni adott idő alatt. A robotot irányíthatja, valamint olajfoltot, vagy ragacsot hagyhat maga mögött a pályán.

Kezdetben a robot nulla sebességről, tehát álló helyzetből indul, egységnyi sebességvektorral. Ezzel a sebességgel át tud ugrani, a tartózkodási helyével szomszédos mezők egyikére. Kétegységnyi sebességgel a kétegységnyi sugarú környezetébe tud ugrani. A képen egy példa látható az egységnyi ugrás lehetséges helyeire (piros/kisebb négyzet), ha a pálya négyzetrácsos, illetve a kétegységnyi ugrással elérhető mezőkre (kék/nagyobb négyzet). A kép csak illusztráció.



A pálya tehát mezőkre van osztva, ezek a mezők belül homogének, vagyis nem tud egymás mellett két robot lenni egy mezőn, illetve egy lerakott olajfoltot vagy ragacsot kikerülni, ha az adott mezőre érkezik.

Az irányítás során a sebességvektort két ugrás között lehet csak változtatni, bármely irányba, de csak egy egységgel.  Ilyenformán a már elért sebességről értelemszerűen nem tud a robot azonnal egy sokkal kisebb sebességre átváltani. Ez tekinthető úgy, minthogy a robot nem képes a két ugrás közötti időben elvégezni csak egy sebességváltoztató utasítást és egy utasítás csak eggyel növelés, vagy csökkentés lehet. Mindez azt is jelenti, hogy az irányításhoz plusz nehézségként adódik, hogy megeshet, hogy a pályának nincs egyik irányban sem akkora kiterjedése, mint amekkorát a következő ugráskor a robot minimálisan át fog repülni, így akár több lépéssel előre meg kell tervezni minden lépést.

Ha a rendelkezésre álló idő alatt nincs sebességmódosítás, akkor a sebességvektor változatlan marad. Két ugrás között tehát korlátos idő áll rendelkezésre a felhasználónak, hogy irányíthassa a robotot.

A játék folyamán a robot akadályokat hagyhat maga mögött, ragacs, vagy olaj formájában. Ha egy játékos ragacsra érkezik, a sebessége megfeleződik, a vektor minden komponense fele annyi lesz, mint előtte, de a lehetőség megmarad, hogy változtathasson a robot irányán. Ha az eredeti sebesség páratlan Ragacsot két ugrás közötti időben hagyhat maga mögött a robot, és ez ott marad azon a mezőn, amin a robot éppen tartózkodik. A ragacsot lerakó robotot a lerakás utáni ugrásban nem befolyásolja semmi módon az alatta lévő ragacs. Ha valaki arra a mezőre lép ezután, az ragacsra lépett, a roboton ez kifejti hatását, de a ragacs ott marad a mezőn. Az olajt hasonlóképpen lehet lerakni, valamint az is ott marad a mezőn, miután belelépnek. Az olaj hatása a robotra, aki belelépett, hogy nem tudja megváltoztatni a sebességvektort, ez a hatás csak az adott mezőn érvényes, ha onnan másik mezőre lép, ott már változtathatja ismét. Az olajkészletek végesek, pontosan, ahogy a ragacskészletek. Mindkét féle akadályból három van a robotok fegyvertárában a játék indulásakor.

Egy mezőn akkor is lehet akadály, hogyha előzőleg oda nem tett robot. Ezek az akadályok előre jól látható pozíciójúak, nincsenek elrejtve. Ha több robot rakna egymás után akadályt egy mezőre, az utolsó akadály fog érvényre jutni a következő robot érkezésekor.

A pályán a robotok, ha ugyanabban az időben ugyanarra a mezőre lépnének, akkor az ütközésnek számít. Pontosan akkor lép fel ütközés, ha a robotok sebességvektora ugyanarra a mezőre mutat a következő ugrásnál. Ebből következően a robotok minden esetben elkerülik egymást, míg a levegőben tartózkodnak. Ütközéskor az összes résztvevő robot tönkremegy, roncsuk olajfoltot hagy a mezőn.

Ha a robot olyan mezőre érkezik, ami nem a pálya része, akkor kiesett a játékból, nem tud továbbhaladni. A pályán kívüli robotok eltűnnek, tehát roncsuk nem marad a pálya mellett. Ha több robot egyszerre érkezik ugyanarra a mezőre a pályán kívül, nincs ütközés, mind eltűnik.

### Felhasználók

Bármely természetes személy

### Korlátozások

Csak olyan gépeken használható, melyek támogatják a java használatát és telepítve van rajtuk a környezet (JRE, Java 1.7). A támogatott operációs rendszerek listája megtalálható az oracle honlapján[[3]](#footnote-3).

### Feltételezések, kapcsolatok

<http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

<http://www.oracle.com/technetwork/java/javase/config-417990.html#os>

## Követelmények

### Funkcionális követelmények

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Azonosító** | **Leírás** | **Ellenőrzés** | **Prioritás** | **Forrás** | **Use-case** | **Komment** |
| Ragacs | A robot leteszi a ragacsot a mezőre | kiértékelés | alapvető | specifikáció | URagacs |  |
| Olaj | A robot olajt hagy a mezőn maga mögött | kiértékelés | alapvető | specifikáció | UOlaj |  |
| Indítás | A játék elindítása | bemutatás | fontos | specifikáció | UIndítás |  |
| Halál | A robotot a pályán kívülre kerül vagy ütközik | bemutatás | alapvető | specifikáció/  csapat | UHalál |  |
| Befejezés | A versenynek vége, letelik az idő | bemutatás | fontos | specifikáció | UBefejezés |  |
| Grás | Helyváltoztatás | bemutatás  /kiértékelés | alapvető | specifikáció | UGrás |  |
| Vector | Sebességvektor módosítása | bemutatás/  kiértékelés | alapvető | specifikáció/  csapat | UVector |  |

### Erőforrásokkal kapcsolatos követelmények

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Azonosító** | **Leírás** | **Ellenőrzés** | **Prioritás** | **Forrás** | **Komment** |
| Java | JRE telepítése előre | applikáció futtatása | fontos | internet |  |
| OS | java-t futtató operációs rendszer | JRE telepítés | fontos | internet/  adathordozó |  |

### Átadással kapcsolatos követelmények

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Azonosító** | **Leírás** | **Ellenőrzés** | **Prioritás** | **Forrás** | **Komment** |
| Deployment | A program mappájának a hard diskre másolása. | A program futtatása | fontos | a jogtulajdonos | Nem igényel telepítést |

### Egyéb nem funkcionális követelmények

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Azonosító** | **Leírás** | **Ellenőrzés** | **Prioritás** | **Forrás** | **Komment** |
| Ugrásidő | A user utasításától az új mezőbeérkezésig eltelő idő | Logic/Application Test | opcionális | csapat | E kettő esemény között lehet a legtöbb számítás |
| Időmérés | Az idő mérése és a felhasználó értesítése erről | Application Test | alapvető | specifiká-ció | Fontos például, hogy mindig ugyanannyi idő legyen a döntésre |
| Pálya | Versenykörnyezet definiálása | Application Test | alapvető | specifiká-ció |  |

## Lényeges use-case-ek

### Use-case leírások

|  |  |
| --- | --- |
| **Use-case neve** | URagacs |
| **Rövid leírás** | A robot leteszi a ragacsot a mezőre |
| **Aktorok** | Felhasználó, Robot |
| **Forgatókönyv** | A felhasználó kiadja a parancsot a ragacs lerakására, a robot pedig lerakja a ragacsot. |

|  |  |
| --- | --- |
| **Use-case neve** | UOlaj |
| **Rövid leírás** | A robot olajt hagy a mezőn maga mögött |
| **Aktorok** | Felhasználó, Robot |
| **Forgatókönyv** | A felhasználó kiadja a parancsot az olaj lerakására, a robot pedig lerakja a ragacsot. |

|  |  |
| --- | --- |
| **Use-case neve** | UIndítás |
| **Rövid leírás** | A játék elindítása |
| **Aktorok** | Felhasználó |
| **Forgatókönyv** | A felhasználó elindítja a játékot, a játék elindul |

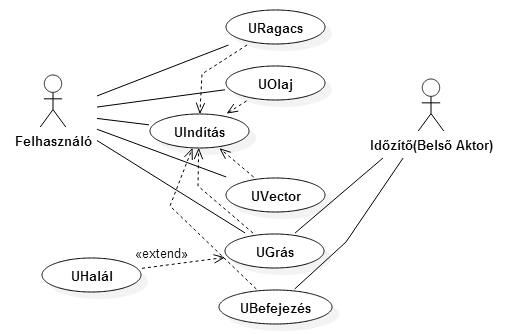
|  |  |
| --- | --- |
| **Use-case neve** | UHalál |
| **Rövid leírás** | A robotot a pályán kívülre kerül vagy ütközik |
| **Aktorok** | Robotok |
| **Forgatókönyv** | Az egyik robot landol, jelez a másiknak, ha ugyanoda érkezett ahol a másik tartózkodik, mindketten megsemmisülnek, olajt hagynak. Ha pedig pályán kívül landol, csak simán megsemmisül. |

|  |  |
| --- | --- |
| **Use-case neve** | UBefejezés |
| **Rövid leírás** | A versenynek vége, letelik az idő |
| **Aktorok** | Belső aktor, vagy Robot |
| **Forgatókönyv** | Lejár az idő, a játék megáll, vagy meghal az utolsó robot aki még versenyt jelentett, így a játék megáll |

|  |  |
| --- | --- |
| **Use-case neve** | UGrás |
| **Rövid leírás** | Helyváltoztatás |
| **Aktorok** | Felhasználó, Robot, Belső aktor |
| **Forgatókönyv** | A felhasználó befejezi a „körét”, vagy a Belső aktor fejezteti be vele. A kör befejezéséről a Robot értesül és ugrik. |

|  |  |
| --- | --- |
| **Use-case neve** | UVector |
| **Rövid leírás** | Sebességvektor módosítása |
| **Aktorok** | Felhasználó |
| **Forgatókönyv** | A felhasználó módosítja a sebességvektort, irányt és értéket ad neki. |

### Use-case diagram



## Szótár

***robot*** – játék szereplője, akit felhasználó irányít.

***pálya***- a robotok versenyének színtere.

***mező***- a pálya ezekből épül fel, a robotok, mezőről mezőre ugrálnak.

***kör***-egy teljes út megtétele a pályán startvonaltól startvonalig, melyet a versenyző robotok a játék során többször is megismételhetnek

***startvonal*** – pálya kezdőpontja.

***megtett távolság*** – eltelt körök száma és a startvonaltól mért távolság (mezőkben) összege

***sebességvektor***- minden robotnak van egy sebessége, ami meghatározza, hogy mekkorát ugrik

***ugrás*** – megadott irányvektorral történő előrehaladás, csak a kezdőpont és a végponton van a robot a pályán.

***ragacsfolt***-a pálya egy mezőjén elhelyezkedő zöld tárgy,ha a robot egy ilyen mezőre lép megfeleződik a sebessége.

***olajfolt***-a pálya egy mezőjén elhelyezkedő fekete tárgy,ha a robot egy ilyen mezőre ugrik

a sebességét nem tudja módosítani.

***kiesik***-akkor történik ha a robot olyan mezőre lép mely nem része a pályának ekkor a játékos kiesik a pályáról és előröl kell kezdenie a játékot.

***ütközés***-két robot ütközik ha azonos mezőre lép

***menü***-a játék inditásakor megjelenő képernyőfelület. Itt különböző menüpontokból válaszhatunk mint pl. Új játék

***játék megnyerése***-a játékot az játékos nyeri meg ,amelyik adott idő alatt a legtöbb tette meg

## Projekt terv

## Határidők

|  |  |  |
| --- | --- | --- |
|  | **Dátum** | **Feladat** |
| 1 | febr. 13. | 14 h - csapatok regisztrációja |
| 2 | febr. 23. | Követelmény, projekt, funkcionalitás - beadás |
| 3 | márc. 2. | Analízis modell kidolgozása 1. - beadás |
| 4 | márc. 9. | Analízis modell kidolgozása 2. - beadás |
| 5 | márc. 16. | Szkeleton tervezése - beadás |
| 6 | márc. 23. | Szkeleton - beadás |
| 7 | márc. 30. | Prototípus koncepciója - beadás |
| 8 | ápr. 7. | Részletes tervek - beadás |
| 9 | ápr. 13. |  |
| 10 | ápr. 20. | Prototípus - beadás |
| 11 | ápr. 27. | Grafikus felület specifikációja - beadás |
| 12 | máj. 4. |  |
| 13 | máj. 11. | Grafikus változat - beadás |
| 14 | máj. 15. | Összefoglalás - beadás |

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2015.02.20. 14:00 | 2.5 óra | Mitterer Márton  Poór Bálint  Velenyák János  Pogátsa Áron | Tevékenység:  Értekezlet, a Játék elképzelése,megbeszélése,  2.1, 2.2, 2.3 írása  Döntés:  játék koncepciója, robot viselkedése, irányítása ütközés hatása |
| 2015.02.20. 19:41 | 0.3 óra | Deák Zsolt | Tevékenység: gerit repo készítés, 2.1 módosítás |
| 2015.02.20. 20:24 | 4,5 óra | Deák Zsolt | Tevékenység: 2.2, 2.3, 2.4.1, 2.6 módosítása/megírása |
| 2015.02.21. 11:00 | 1 óra | Velenyák János | Szótár kiegészítése |
| 2015.02.22 12:00 | 1 óra | Pogátsa Áron | Tevékenységek:  Szótár,Use case diagram |
| 2015.02.22 17:53 | 1 óra | Deák Zsolt | Tevékenység:  Dokumentumok merge-ölése, use-case diagram kiegészítés, 2.2.2 kiegészítés, összefoglalás |

# Analízis modell kidolgozása

## Objektum katalógus

### Bot

Kezelnie kell a magánál hordott ragacsokat és olajfoltokat, ami feladatkör kiterjed ezek lerakására és nyilvántartására, illetve a felhasználó informálására is, például, ha elfogyna valamelyik.

El kell tudja végezni az ugrásokat, amiből adódik, hogy a sebesség-/irányállítások követése, szabályozása is az ő feladata, ezen felül kapcsolatban kell legyen a pályával, hogy meg tudjon érkezni egy mezőre az ugrás után.

### Map

Ismernie kell az összes mezőt, tehát a versenyzésre használható pályarészeket és a halált jelentőeket is.

A pálya értesül az idő lejártáról és érvényteleníti a mezőket.

A robot informálása a pályáról, van-e akadály, lehet-e oda lépni. Ezeket az információkat a robotnak kell elkérnie.

### Field

Kezelnie kell az esetlegesen rajta levő akadályt.

Tudnia kell, hogy érvényes-e, tehát meghal-e a robot, amelyik rátéved, vagy sem.

Végül tudnia kell, hogy van-e rajta robot, illetve kezelnie kell az új robot landolását.

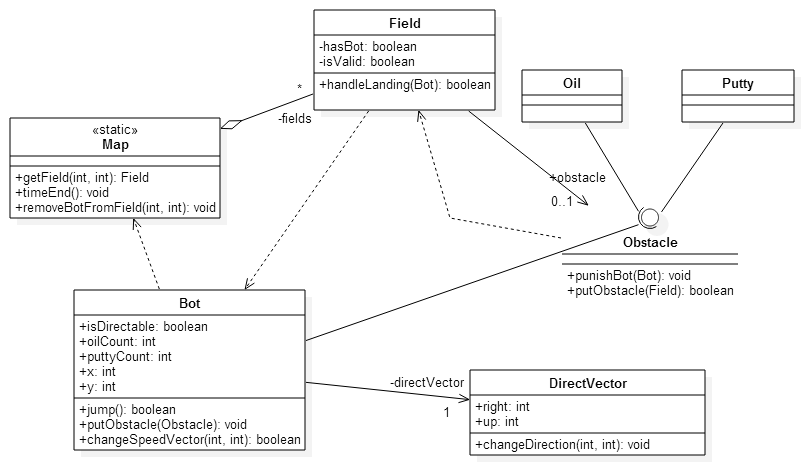
### Obstacle

A belelépő roboton el kell tudja végezni a megfelelő büntetést, attól függően, hogy olajról, vagy ragacsról van épp szó.

### DirectVector

Tárolja a robot irányvektorát (sebességvektorát), kapott utasítások szerint képes megváltoztatni 2 ugrás között. Az ugrásokhoz a paramétereket ad.

## Statikus struktúra diagramok



## Osztályok leírása

### Bot

#### Felelősség

Akadályok nyilvántartása, mennyiségükről a felhasználó tájékoztatása. Ugrás végrehajtása, annak kimenetelének ellenőrzése. Sebességvektor megváltoztatása

#### Ősosztályok

Nincs

#### Interfészek

Nincs

#### Attribútumok

* **isDirectable**: boolean: Ha olajra léptünk false lesz az értéke, nem lehet irányítani a robotot, egyébként true, a robot sebességvektora változtatható
* **directVector**: DirectVector: Ebben tároljuk el a sebességvektor értékét, hova kell ugrania a robotnak(következő field)
* **oilCount**: int: Számon tartja mennyi olaj akadályt rakhat még le a robot
* **puttyCount**: int: Számon tartja mennyi ragacsot rakhat még le a robot
* **x**: int: tárolja a robot jelenlegi pozícióját a pályán
* **y**: int: tárolja a robot jelenlegi pozícióját a pályán

#### Metódusok

* **boolean jump():** A jelenlegi sebességvektor felhasználásával elvégzendő ugrásra utasítja a robotot, illetve visszadja, hogy sikerült-e odaugrani
* **void putObstacle(Obstacle):** Átad a robotnak egy Obstacle-t megvalósító objektumot, amit az aktuális mezőre letesz a robot.
* **boolean changeSpeedVector(int, int):** Az adott paraméterekre módosítja robot sebességvektorát, ha ez lehetséges, ha valamiért nem, akkor hamis értékkel tér vissza.

### DirectVector

#### Felelősség

Irányvektort (ld Funkcionális leírás: sebességvektor) tárol, illetve módosít.

#### Ősosztályok

Nincs

#### Interfészek

Nem

#### Attribútumok

* **right**: int: vízszintes koordináta
* **up**: int: függőleges koordináta

#### Metódusok

* **void changeDirection(int, int)**: megváltoztatja az irányvektort a kapott inputra

### Field

#### Felelősség

Akadályok kezelése (rajta lévőké), mi van rajta, tudnia kell magáról, hogy érvényes-e, azaz meghal-e a robot, ha rálép, valamint tudnia kell, ha robot van rajta.

#### Ősosztályok

Nincs

#### Interfészek

Nincs

#### Attribútumok

* **isValid:** true, ha rá lehet lépni a mezőre, false, ha a mező nem a pálya rásze, ha ilyenkor rálép a robot, meghal
* **hasBot**: boolean
* **obstacle**:A fielden lévő, vagy nem lévő akadály.

#### Metódusok

* **boolean handleLanding(Bot):** A rálépő roboton elvégzi a megfelelő műveleteket, amik a mezőre lépés kapcsán szükségesek (pl akadály érvényesítése), illetve visszaadja, hogy sikerült-e a robotnak landolnia.

### Map

#### Felelősség

Idő kezelése, mezők nyilvántartása, információ szolgáltatása a robotnak a mezőkről.

#### Ősosztályok

Nincs

#### Interfészek

Nincs

#### Attribútumok

* **fields:** Field: Mezők eltárolása.

#### Metódusok

* **Field getField(int x, int y)**: Lekérdezi az x, y koordinátájú mezőt
* **void timeEnd()**: Kezeli az idő lejártát
* **void removeBotFromField()**:Robot eltávolítása a fieldről.

### Obstacle

#### Felelősség

Adott büntetés elvégzése a rálépő roboton. Akadály elhelyezése a fielden.

#### Ősosztályok

Nincs

#### Interfészek

Nincs

#### Attribútumok

Nincs

#### Metódusok

* **void punishBot(Bot bot)**: Elvégzi a büntetést.
* **putObstacle(Obstacle o):** Lerakja az akadályt, csökkenti a lerakható akadályok darabszámát

### Oil

#### Felelősség

Obstacle inteface megvalósítása. Adott büntetés elvégzése a rálépő roboton. Akadály elhelyezése.

#### Ősosztályok

Nincs

#### Interfészek

**Obstacle**

#### Attribútumok

Nincs

#### Metódusok

* **void punishBot(Bot bot)**: Elvégzi a büntetést.
* **putObstacle(Obstacle o):** lerakja az akadályt, csökkenti a lerakható akadályok darabszámát

### Putty

#### Felelősség

Obstacle inteface megvalósítása. Adott büntetés elvégzése a rálépő roboton. Akadály elhelyezése.

#### Ősosztályok

Nincs

#### Interfészek

**Obstacle**

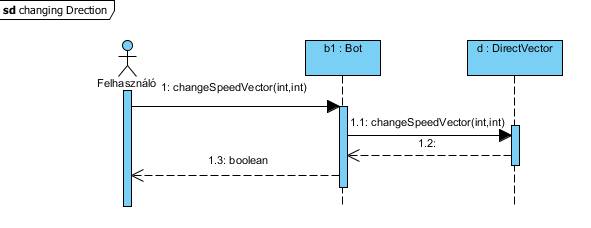
#### Attribútumok

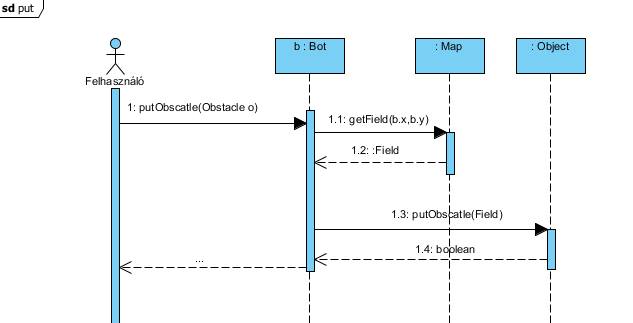
Nincs

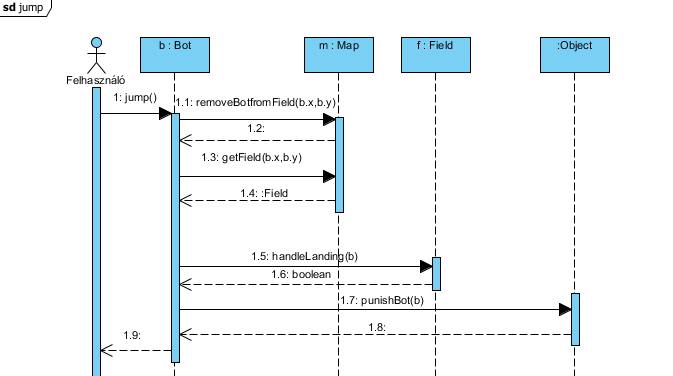
#### Metódusok

* **void punishBot(Bot bot)**: Elvégzi a büntetést.
* **putObstacle(Obstacle o):** lerakja az akadályt, csökkenti a lerakható akadályok darabszámát

## Szekvencia diagramok

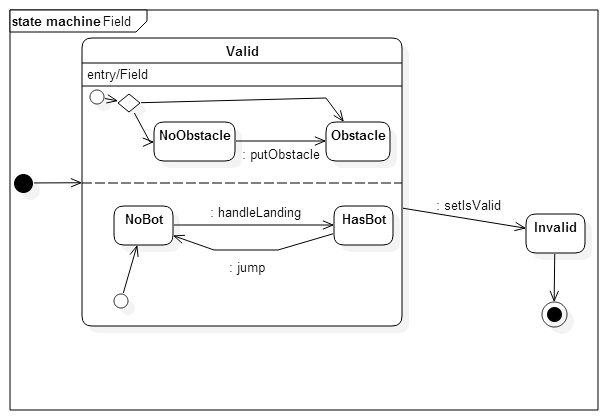


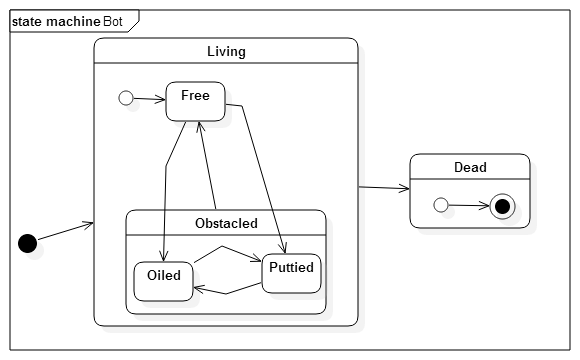




## State-chartok

Feltéve, hogy egy induláskor valid a field





## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2015.02.23. 16:22 | 2 óra | Deák Zsolt | Tevékenység: 3.1, 3.2 írása |
| 2015.02.24. 15:15 | 1,5 óra | Poór Bálint  Velenyák János | Tevékenység: javítások, 3.3 írása |
| 2015.02.27 | 1 óra | Poór Bálint  Velenyák János | Tevékenység:  3.1, 3.2, 3.3 |
| 2015.03.01 9:32 | 0,5 óra | Deák Zsolt | Class diagram változtatás |
| 2015.03.01 16:02 | 4 óra | Deák Zsolt | State chartok, Class diagram módosítás, dokumentáció |

# Analízis modell kidolgozása 2

## Objektum katalógus

### Bot

Kezelnie kell a magánál hordott ragacsokat és olajfoltokat, ami feladatkör kiterjed ezek lerakására és nyilvántartására, illetve a felhasználó informálására is, például, ha elfogyna valamelyik.

El kell tudja végezni az ugrásokat, amiből adódik, hogy a sebesség-/irányállítások követése, szabályozása is az ő feladata, ezen felül kapcsolatban kell legyen a pályával, hogy meg tudjon érkezni egy mezőre az ugrás után.

### Map

Ismernie kell az összes mezőt, tehát a versenyzésre használható pályarészeket és a halált jelentőeket is.

A pálya értesül az idő lejártáról és érvényteleníti a mezőket.

A robot informálása a pályáról, van-e akadály, lehet-e oda lépni. Ezeket az információkat a robotnak kell elkérnie.

### Field

Kezelnie kell az esetlegesen rajta levő akadályt.

Tudnia kell, hogy érvényes-e, tehát meghal-e a robot, amelyik rátéved, vagy sem.

Végül tudnia kell, hogy van-e rajta robot, illetve kezelnie kell az új robot landolását.

Ha a robot utasítja le kell tudni rakni magára az adott akadályt.

### Obstacle

A belelépő roboton el kell tudja végezni a megfelelő büntetést, attól függően, hogy olajról, vagy ragacsról van épp szó.

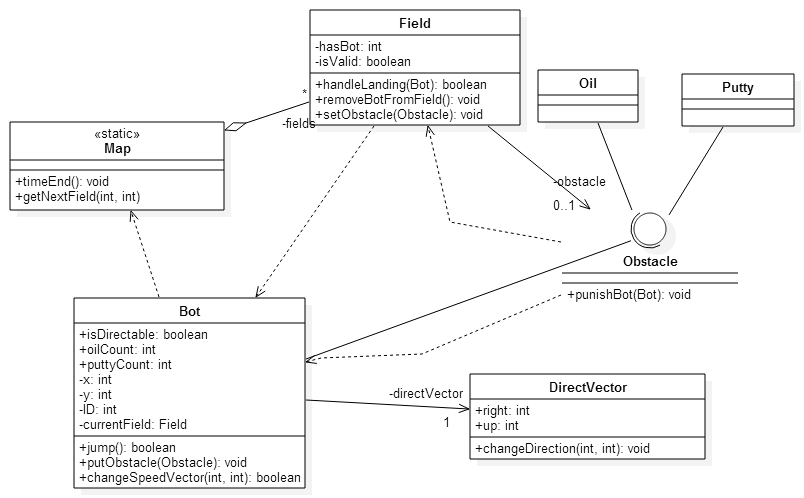
### Oil

Az Obstacle interface-t valósítja meg, feladata hogy a sebessség vektor megváltoztatását tiltsa le,ha olyan mezőre lép a robot amelyen Oil van.

### Putty

Az Obstacle interface-t valósítja meg, feladata hogy a sebessség vektort felezze,ha olyan mezőre lép a robot amelyen Putty van.

## Statikus struktúra diagramok



## Osztályok leírása

### Bot

#### Felelősség

Akadályok nyilvántartása, mennyiségükről a felhasználó tájékoztatása. Ugrás végrehajtása, annak kimenetelének ellenőrzése. Sebességvektor megváltoztatása

#### Ősosztályok

Nincs

#### Interfészek

Nincs

#### Attribútumok

* **isDirectable**: boolean: Ha olajra léptünk false lesz az értéke, nem lehet irányítani a robotot, egyébként true, a robot sebességvektora változtatható
* **directVector**: DirectVector: Ebben tároljuk el a sebességvektor értékét, hova kell ugrania a robotnak(következő field)
* **oilCount**: int: Számon tartja mennyi olaj akadályt rakhat még le a robot
* **puttyCount**: int: Számon tartja mennyi ragacsot rakhat még le a robot
* **x**: int: tárolja a robot jelenlegi pozícióját a pályán
* **y**: int: tárolja a robot jelenlegi pozícióját a pályán
* **currentField**: Field: A robot tartózkodási helye, a Field amin éppen áll.
* **ID**: int: A robot azonosítója

#### Metódusok

* **boolean jump():** A jelenlegi sebességvektor felhasználásával elvégzendő ugrásra utasítja a robotot, illetve visszadja, hogy sikerült-e odaugrani
* **void putObstacle(Obstacle):** Lerak egy obstacle-t a fieldre, amin éppen van.
* **boolean changeSpeedVector(int, int):** Az adott paraméterekre módosítja robot sebességvektorát, ha ez lehetséges, ha valamiért nem, akkor hamis értékkel tér vissza.

### DirectVector

#### Felelősség

Irányvektort (ld Funkcionális leírás: sebességvektor) tárol, illetve módosít.

#### Ősosztályok

Nincs

#### Interfészek

Nem

#### Attribútumok

* **right**: int: vízszintes koordináta
* **up**: int: függőleges koordináta

#### Metódusok

* **void changeDirection(int, int)**: megváltoztatja az irányvektort a kapott inputra

### Field

#### Felelősség

Akadályok kezelése (rajta lévőké), mi van rajta, tudnia kell magáról, hogy érvényes-e, azaz meghal-e a robot, ha rálép, valamint tudnia kell, ha robot van rajta.

#### Ősosztályok

Nincs

#### Interfészek

Nincs

#### Attribútumok

* **isValid:** true, ha rá lehet lépni a mezőre, false, ha a mező nem a pálya rásze, ha ilyenkor rálép a robot, meghal
* **hasBot**: boolean
* **obstacle**:A fielden lévő, vagy nem lévő akadály.

#### Metódusok

* **boolean handleLanding(Bot):** A rálépő roboton elvégzi a megfelelő műveleteket, amik a mezőre lépés kapcsán szükségesek (pl akadály érvényesítése), illetve visszaadja, hogy sikerült-e a robotnak landolnia.
* **void removeBotFromField()**:Robot eltávolítása a fieldről.
* **void setObstacle**: Kitüntetett jelentőségű setter metódus. Beállítja az obstacle attribútumot.

### Map

#### Felelősség

Idő kezelése, mezők nyilvántartása.

#### Ősosztályok

Nincs

#### Interfészek

Nincs

#### Attribútumok

* **fields:** Field: Mezők eltárolása.

#### Metódusok

* **Field getNextField(int x, int y)**: Lekérdezi az x, y koordinátájú mezőt, ez egy adott mező elérésére is alkalmas függvény, de a nevéből is adódik, hogy a megfelelő használata a következő mező elkérése.
* **void timeEnd()**: Kezeli az idő lejártát

### Obstacle

#### Felelősség

Adott büntetés elvégzése a rálépő roboton.

#### Ősosztályok

Nincs

#### Interfészek

Nincs

#### Attribútumok

Nincs

#### Metódusok

* **void punishBot(Bot bot)**: Elvégzi a büntetést.

### Oil

#### Felelősség

Obstacle inteface megvalósítása. Adott büntetés elvégzése a rálépő roboton: sebessség vektor megváltoztatását tiltja le.

.

#### Ősosztályok

Nincs

#### Interfészek

**Obstacle**

#### Attribútumok

Nincs

#### Metódusok

* **void punishBot(Bot bot)**: Elvégzi a büntetést.

### Putty

#### Felelősség

Obstacle inteface megvalósítása. Adott büntetés elvégzése a rálépő roboton: sebessség vektor nagyságát megfelezi.

#### Ősosztályok

Nincs

#### Interfészek

**Obstacle**

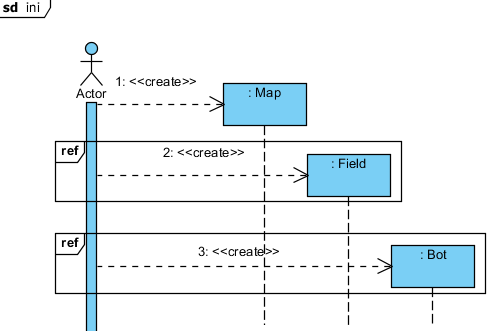
#### Attribútumok

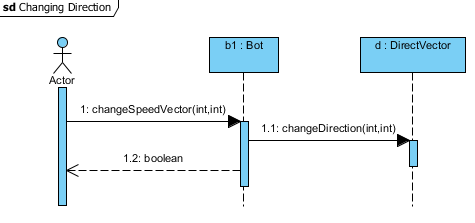
Nincs

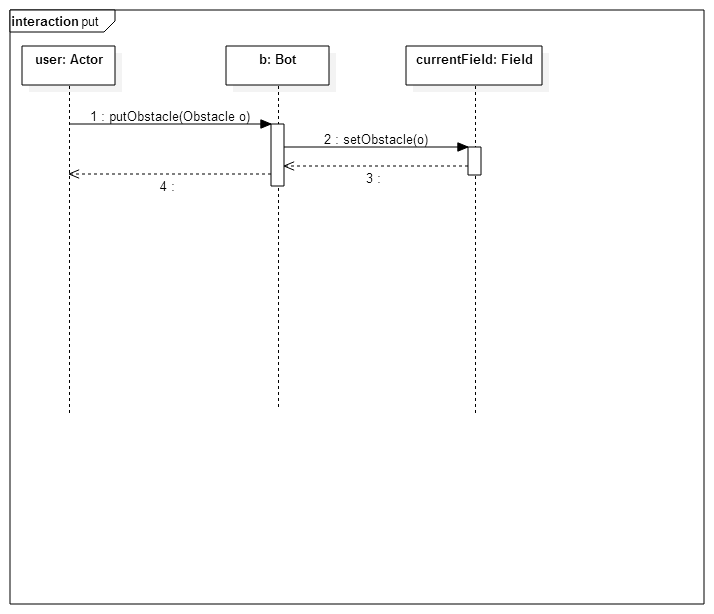
#### Metódusok

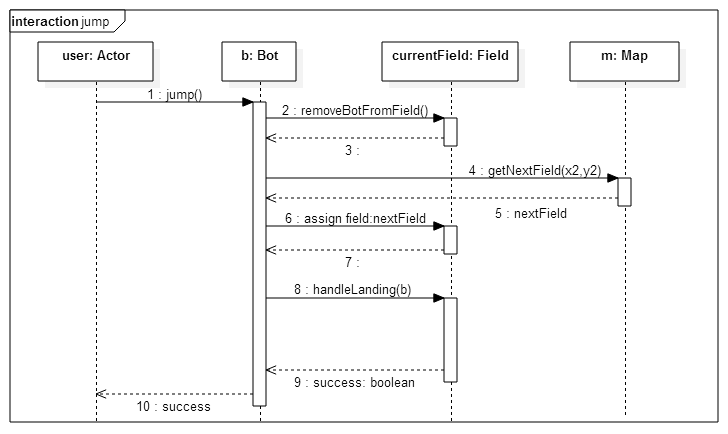
* **void punishBot(Bot bot)**: Elvégzi a büntetést.

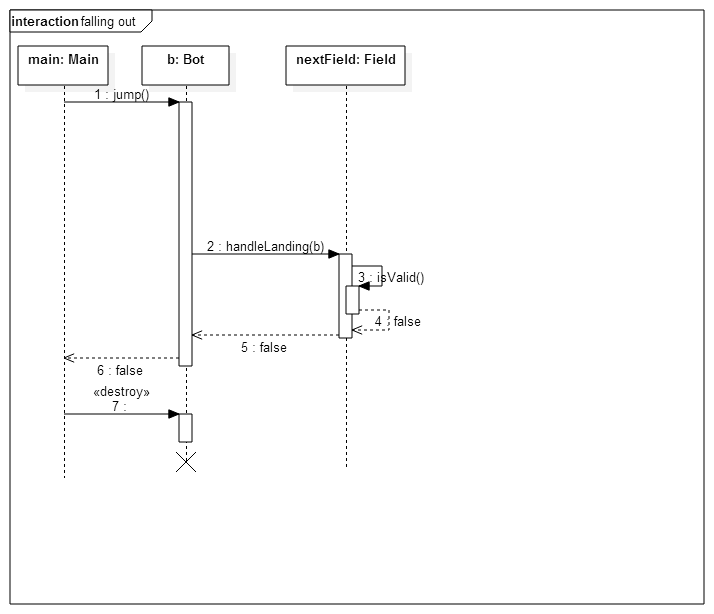
## Szekvencia diagramok

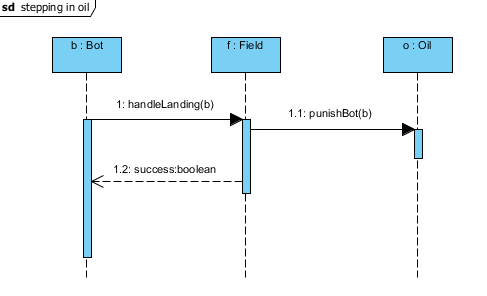


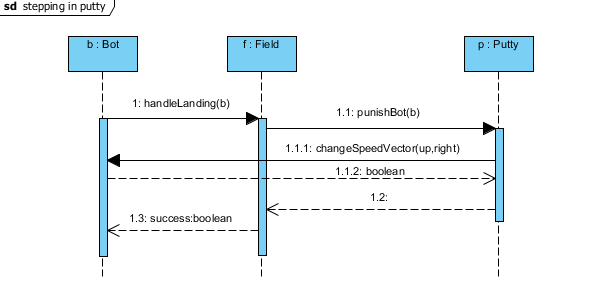


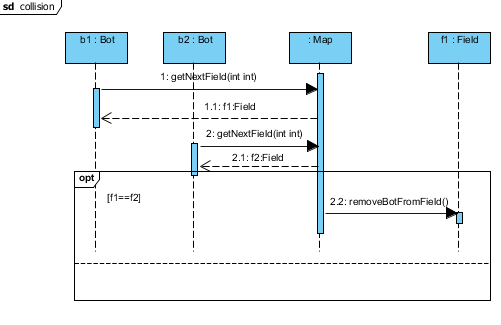






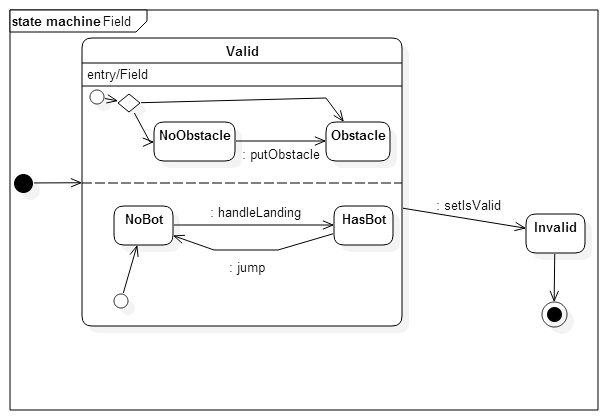


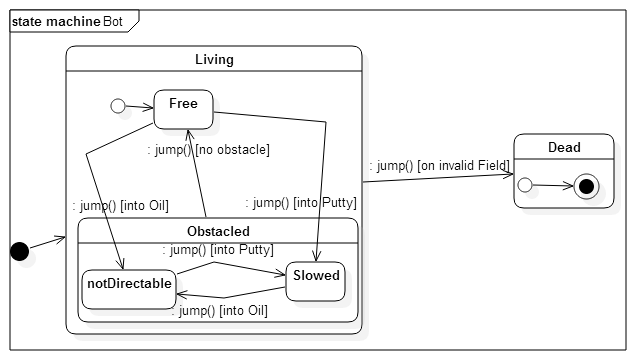




## State-chartok

Feltéve, hogy egy induláskor valid a field



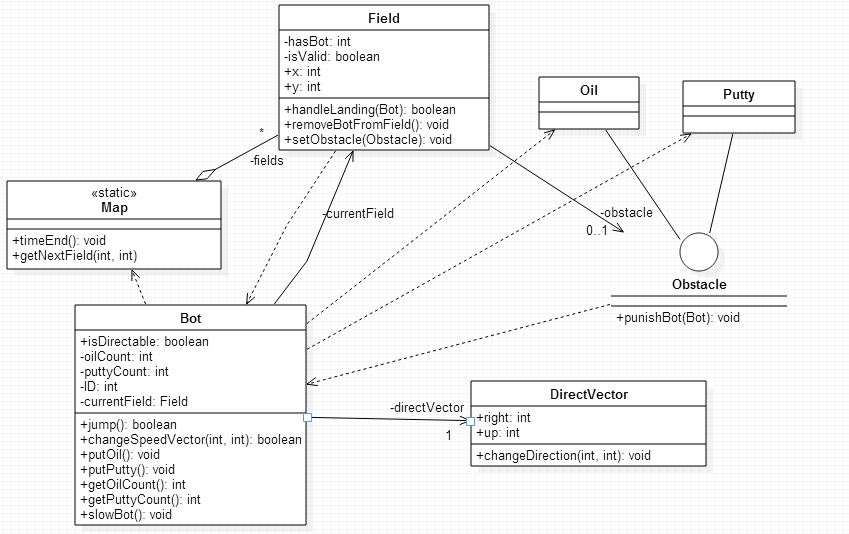


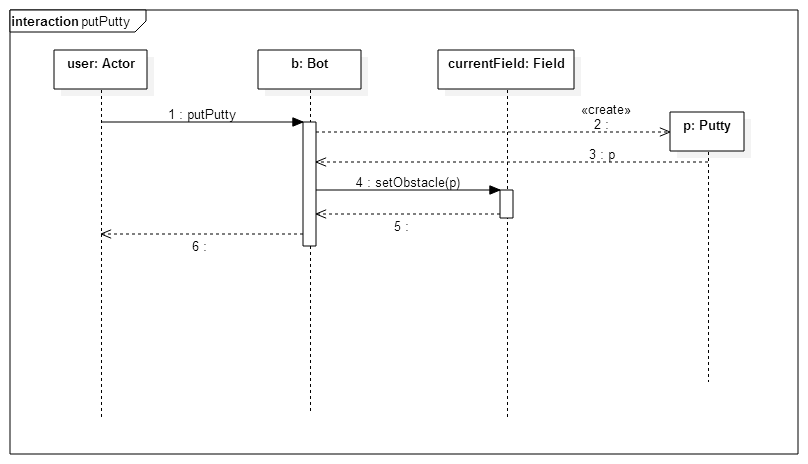
## Napló

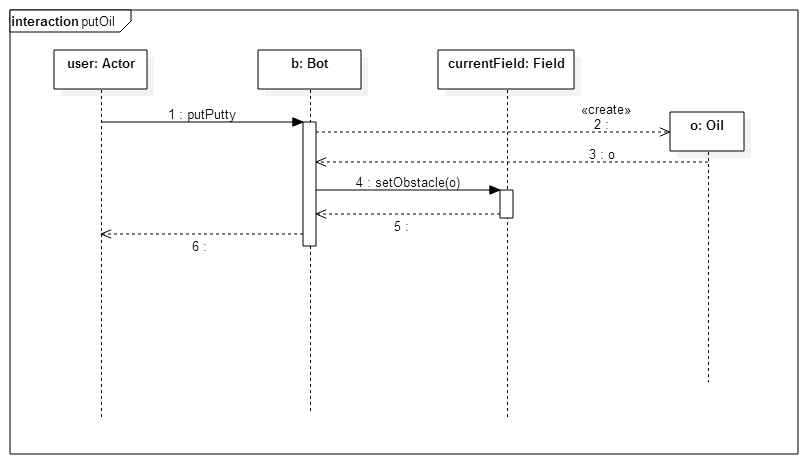
|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2015.03.08 09:02 | 2,5 óra | Deák Zsolt | State chart, szekvencia diagramok, class diagram módosítása |
| 2015.03.06 09:20 | 2 óra | Pogátsa Áron,  Mitterer Márton,  Poór Bálint Imre,  Velenyák János | Megbeszélés: 4.2 |
| 2015.03.08 17:00 | 1 óra | Pogátsa Áron | 4.4 |
| 2015.03.08 21:30 | 30 perc | Mitterer Márton | 4.4 |

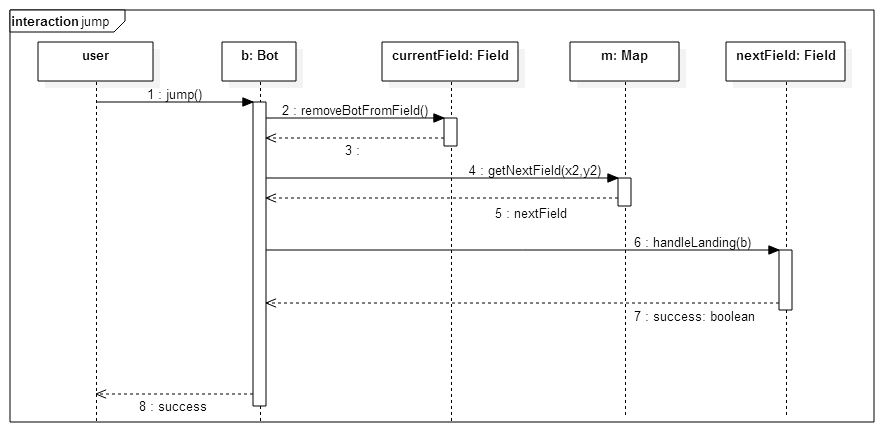
# Szkeleton tervezése

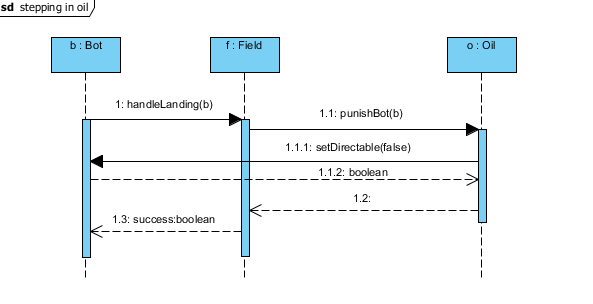
***5.0* Módosítások az Analízis modellhez képes**

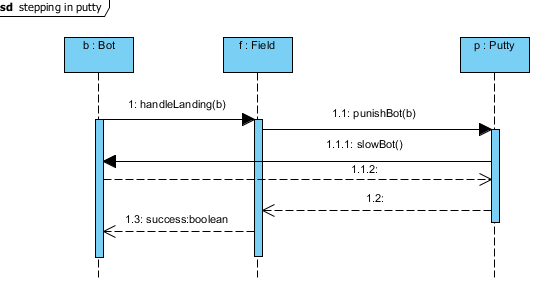


****

****

****



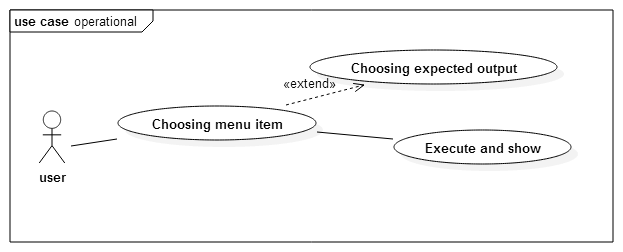


## A szkeleton modell valóságos use-case-ei

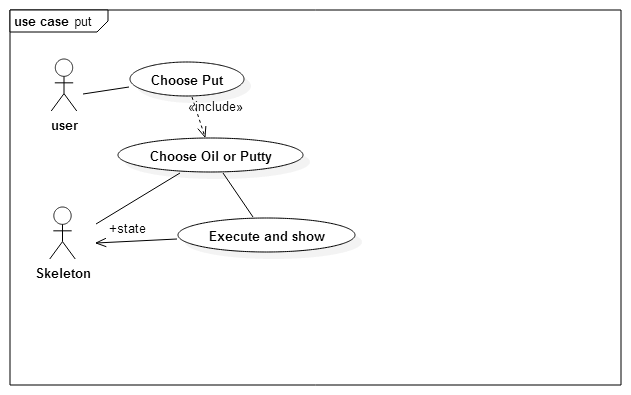
[A szkeletonnak, mint önálló programnak a működésével kapcsolatos use-case-ek. ]

### Use-case diagram

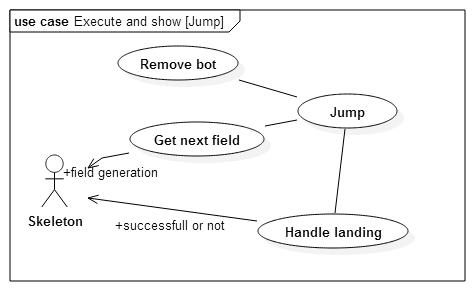
Legfelső szintű működés:

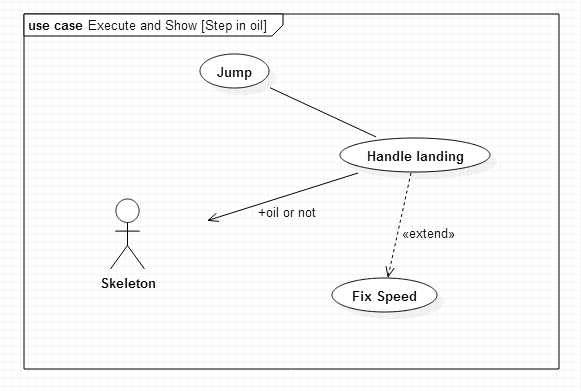


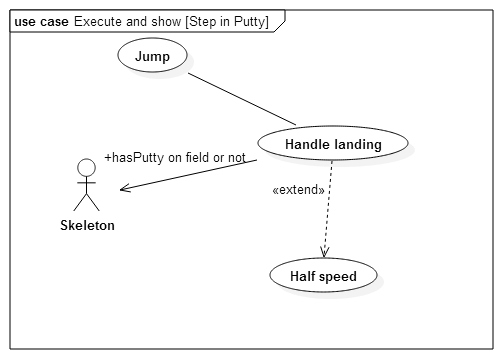
Konkrét működésre példa use-case, ebben az esetben a state a rendelekzésre álló putty-k és oil-ok, ezeket az állapot jellemzőket a skeleton mondja.

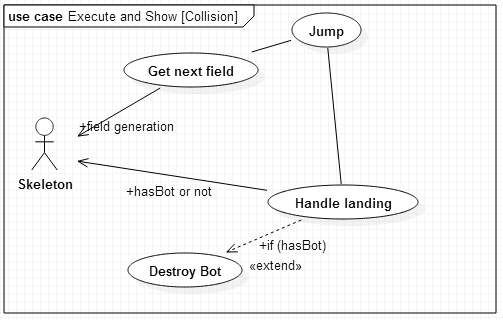


Az Execute and show az analízis tervben sequence diagramon megmutatott megfelelő működést tartalmazza.









### Use-case leírások

|  |  |
| --- | --- |
| **Use-case neve** | operational |
| **Rövid leírás** | A program alapvető használati esete, a felhasználó lehetőségeit mutatja meg. |
| **Aktorok** | A felhasználó |
| **Forgatókönyv** | A felhasználó választ a menüpontok, illetve a lehetséges almenük közül, majd a program ennek megfelelően lefuttatja a kiválasztott működést és közben az egyes lépésekről a felhasználót is tájékoztatja. |

|  |  |
| --- | --- |
| **Use-case neve** | put |
| **Rövid leírás** | Ez a use-case bemutatja a put menüpont választásakor a skeleton program működését. |
| **Aktorok** | A felhasználó, a skeleton (segédosztály) |
| **Forgatókönyv** | A felhasználó kiválasztja a Put menüpontot, majd eldönti, hogy mit rakjon le a robot (igazából a skeleton mivolt miatt nem egy tulajdonképpeni robot, csak egy mock-ja). Ezt követően a skeleton program lefut, a segédosztály állapotait figyelembe véve és futás közben a felhasználónak az egyes lépéseket jelezve. |

|  |  |
| --- | --- |
| **Use-case neve** | Execute and show [Jump] |
| **Rövid leírás** | Az ugrás elvégzése |
| **Aktorok** | Skeleton |
| **Forgatókönyv** | A specifikációknak megfelelő szekvencia futása. Az ugrás hatására eltávolítjuk a robotot az aktuális mezőjéről, kikeressük a következő mezőt és landolunk rajta. |

|  |  |
| --- | --- |
| **Use-case neve** | Execute and show [Step in oil] |
| **Rövid leírás** | A robot olajba lép |
| **Aktorok** | Skeleton |
| **Forgatókönyv** | A specifikációknak megfelelő szekvencia futása. Az ugrás után annak vizsgálata, hogy a mezőn van-e olaj, ha igen, a büntetést végrehajtjuk a roboton, azaz a sebességét fixeljük, nem lehet majd megváltoztatni |

|  |  |
| --- | --- |
| **Use-case neve** | Execute and show [Step in putty] |
| **Rövid leírás** | A robot ragacsba lép |
| **Aktorok** | Skeleton |
| **Forgatókönyv** | A specifikációknak megfelelő szekvencia futása. Az ugrás után a Skeleton megmondja, hogy van-e ragacs a mezőn, ha igen, a sebességet megfelezzük |

|  |  |
| --- | --- |
| **Use-case neve** | Execute and show [collison] |
| **Rövid leírás** | Robotok összeütköznek |
| **Aktorok** | Skeleton |
| **Forgatókönyv** | A specifikációknak megfelelő szekvencia futása. Az ugrás után a Skeleton megmondja, hogy a mező, ahova egy robot lép, vannak-e már ott, ha igen, akkor megsemmisülnek a robotok azon a mezőn |

## A szkeleton kezelői felületének terve, dialógusok

A felhasználó minden esetben, mikor választhat a funkcionalitások közül, az a, b, c, d,… bemenetekkel tudja ezt megtenni. A kezelő felületen egyértelműen eldönthető kell legyen, melyik választáshoz melyik betű tartozik. A válasz véglegesítése az enter billentyűvel lehetséges, ennek lenyomását követően a skeleton vagy továbblép (lefut az egyik funkció, esetleg további bemeneteket kér), vagy nem megfelelő bemenet esetén jelez a felhasználónak.

Minden függvény a következő formátumban kell információt kiírjon: Function called: Class: function(param types), Function ended: function().

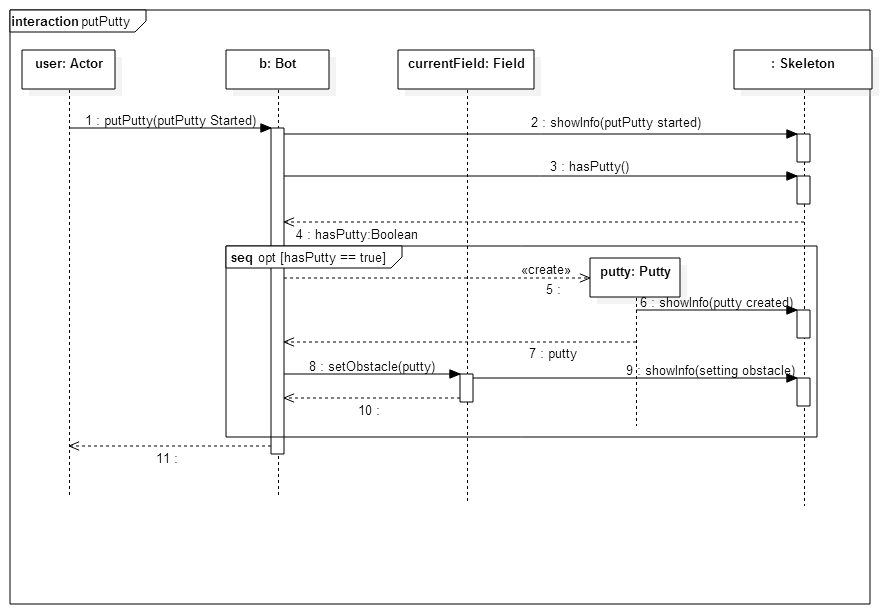
Ezen kívül minden függvény szolgálhat egyéb információval is a működéséről.

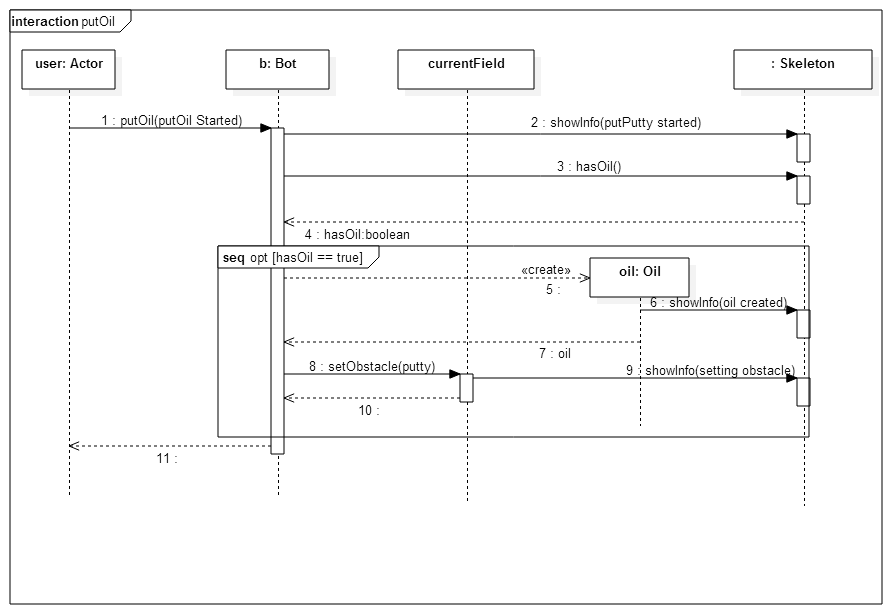
## Szekvencia diagramok a belső működésre

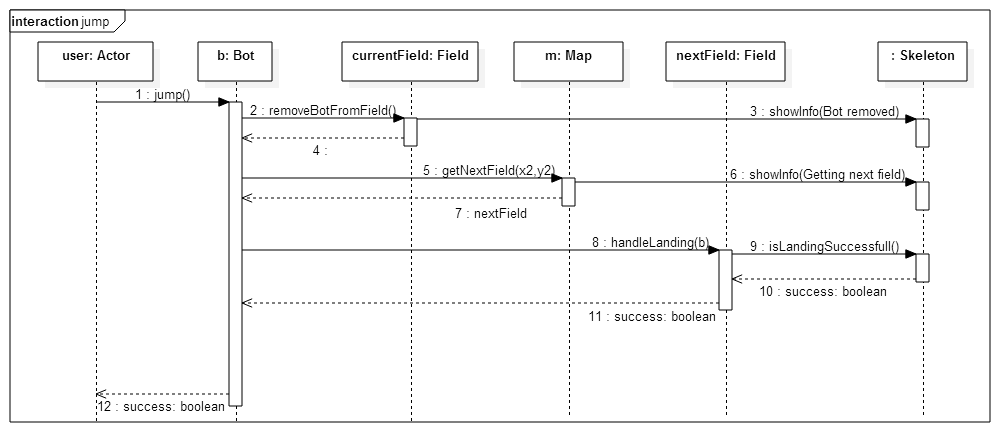
Praktikusan itt most a robotok használatának bemutatása a cél, így elég, ha statikusan kommunikálunk a Skeletonnal, nem kell ismerniük egymás példányait.

Minden függvény loggol a meghívódása után.(showInfo()).

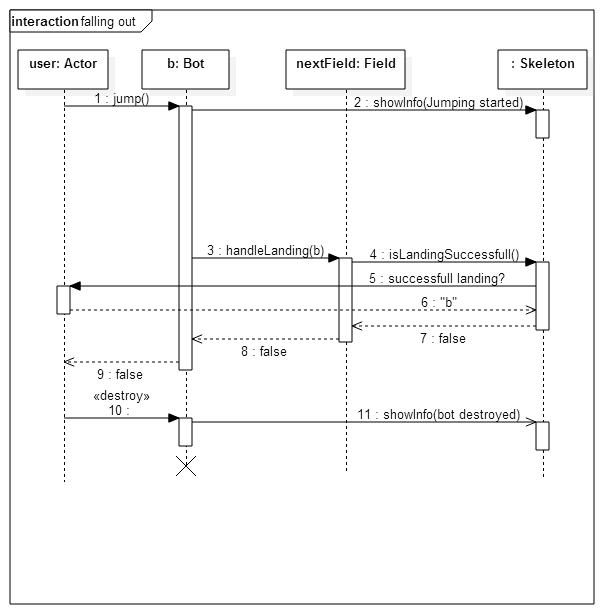
A Skeleton egyéb függvényei maguk is loggolnak.

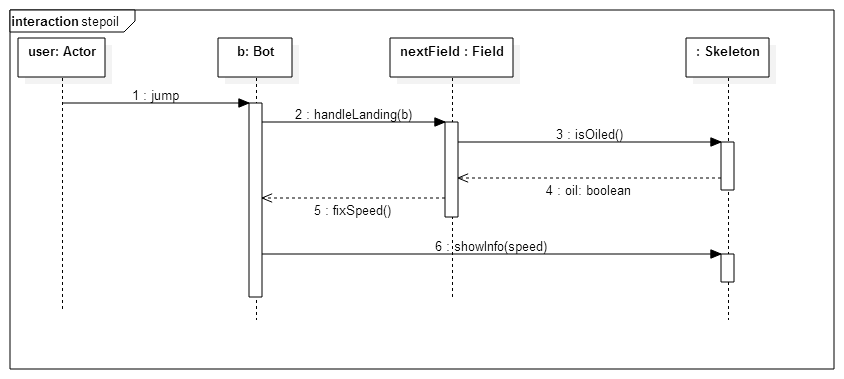


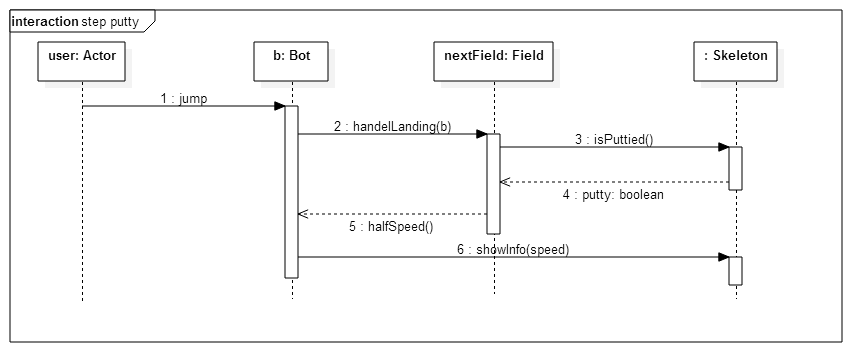


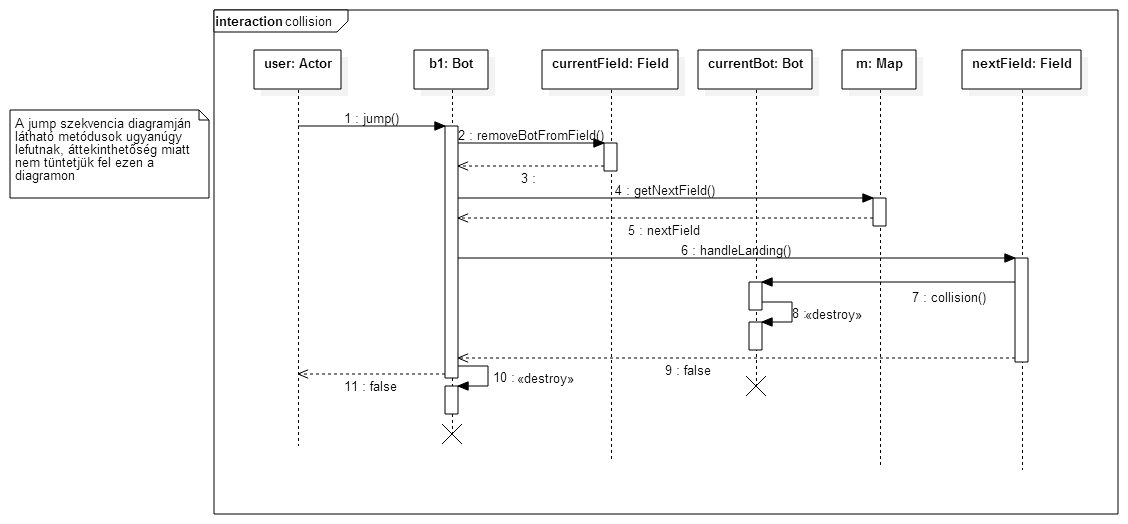


Az ugrás egy részesetén bamutatva a Skeleton döntéshozását:

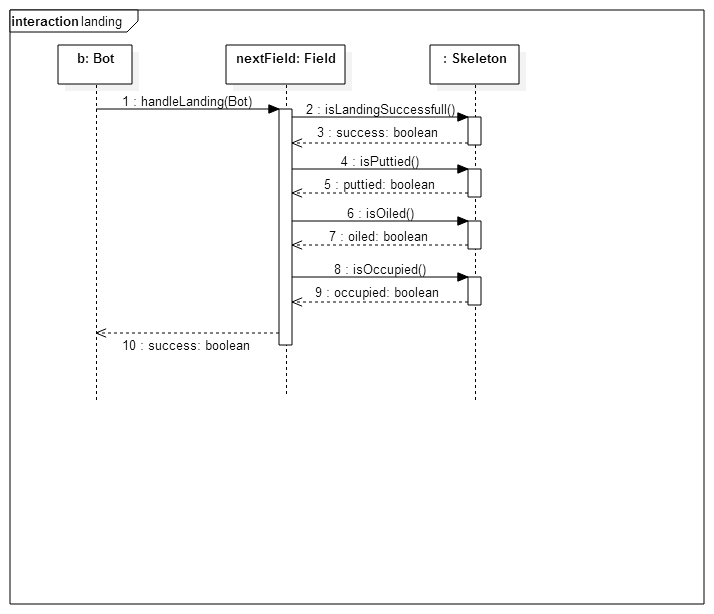




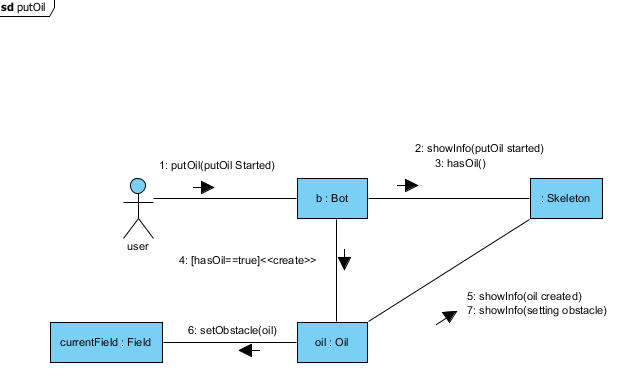


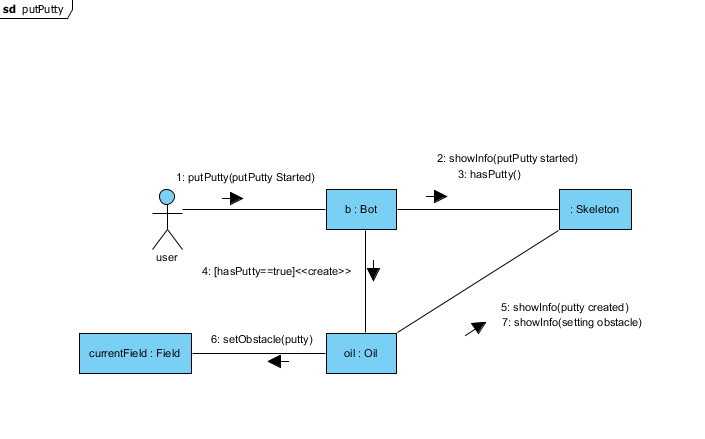


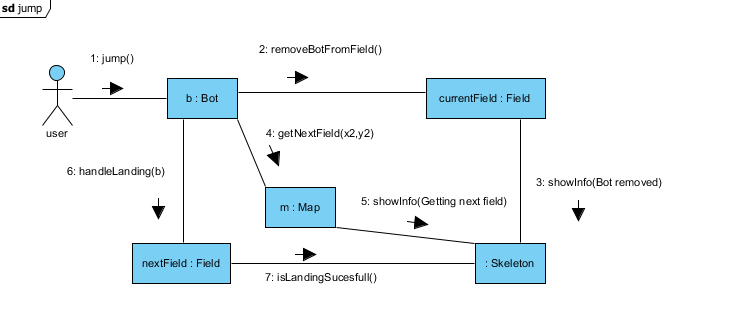
A következő ábra mutatja be a teljes handleLanding (belső) folyamatot, amik hatására a fent megismert hívások megvalósulnak.

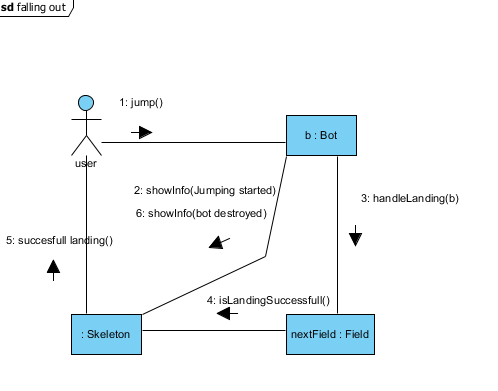


## Kommunikációs diagramok









## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2015.03.15 08:46 | 5 óra | Deák Zsolt | Jump, put, fallout use-case-ei, frissített Analízis modell diagramjai, sd-jai |
| 2015.03.15 17:45 | 2,5 óra | Velenyák János | ragacsba, és olajba lépés use case és szekvencia diagramjai |
| 2015.03.15 23:30 | 2 óra | Velenyák János | az ütközés javítása, valamint use case, szekvencia diagramjai elkészítése a skeleton-hoz |
| 2015.03.15 18:22 | 3,5 óra | Pogátsa Áron | Use-case és sequence diagramok írása/javítása |

# Szkeleton beadás

## Fordítási és futtatási útmutató

### Fájllista

|  |  |  |  |
| --- | --- | --- | --- |
| **Fájl neve** | **Méret** | **Keletkezés ideje** | **Tartalom** |
| Bot.java | 3KB | 2015.03.22 | Bot osztály |
| DirectVector.java | 1KB | 2015.03.22 | DirectVector osztály |
| Field.java | 2KB | 2015.03.22 | Field osztály |
| Map.java | 1KB | 2015.03.22 | Map osztály |
| Obstacle.java | 1KB | 2015.03.22 | Obstacle osztály |
| Oil.java | 1KB | 2015.03.22 | Oil osztály |
| Putty.java | 1KB | 2015.03.22 | Putty osztály |
| Skeleton.java | 6KB | 2015.03.22 | Skeleton osztály |

### Fordítás

Egy (jre 1.7-tel kompatibilis) java project-be be kell importálni, és lefordítani, illetőleg bármely ezzel analóg eljárás megfelel. Ilyen java project-et tud létrehozni például az Eclipse.

### Futtatás

A fordított állományokat futtatni tudó környezet alkalmazása szükséges, vagy szintén ezzel analóg mód. Az Eclipse fejlesztőkörnyezet ezzel a funkcióval is bír.

## Értékelés

|  |  |
| --- | --- |
| **Tag neve** | **Munka százalékban** |
| Deák Zsolt | 36,57 |
| Mitterer Márton | 7,46 |
| Pogátsa Áron | 19,4 |
| Poór Bálint Imre | 14,93 |
| Velenyák János | 21,64 |

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2015.03.21 11:00 | 1 óra | Poór Bálint | osztályok elkészítése |
| 2015.03.21. 12:10 | 5 óra | Deák Zsolt | Skeleton, javítások |
| 2010.03.21. 19:00 | 2 óra | Poór Bálint  Velenyák János | olajba lépés, ragacsba lépés,  pályáról kiesés |

# Prototípus koncepciója

## Specifikáció változás

A megrendelő a következő változásokat kérte a programmal kapcsolatban:

 a ragacs eltűnik a pályáról, miután négy robot ráugrott (elkopik)

 egy meghatározott idő letelte után az olajfolt eltűnik a pályáról (felszárad)

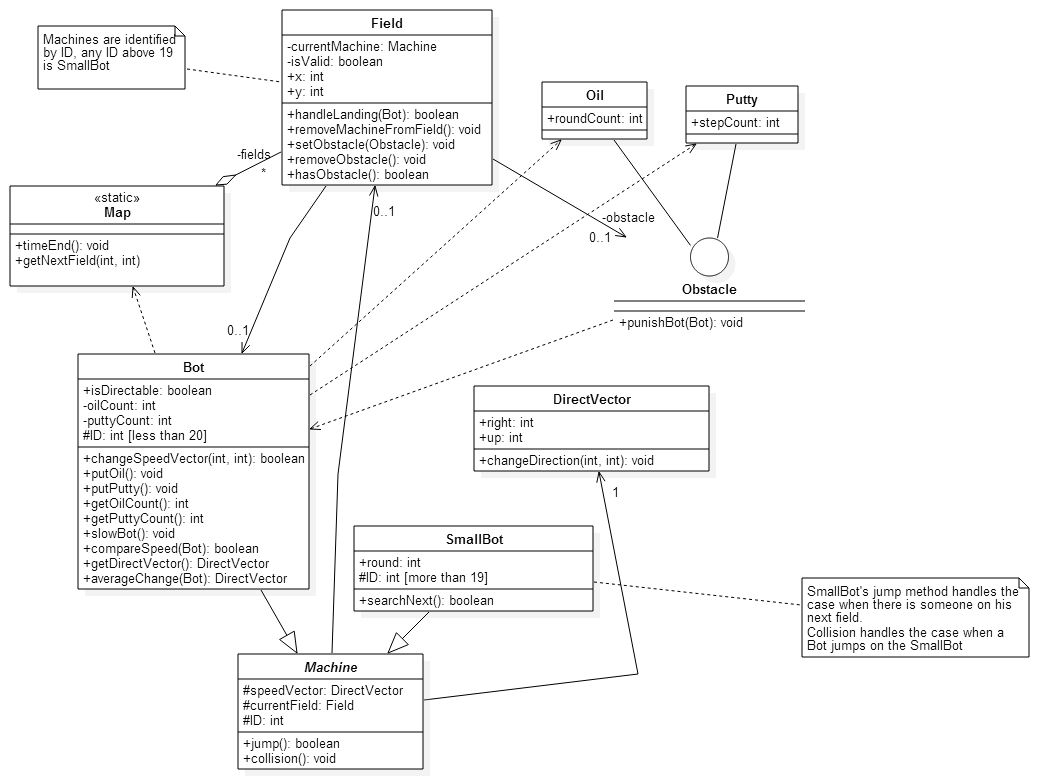
 a pályára időnként keményen dolgozó kisrobotok jutnak be, akik szépen sorban feltakarítják a foltokat. A kisrobot egységnyi sebességgel halad, adott ideig (pl. két kör) takarítja a foltot. A folt feltakarítása után a legközelebbi folthoz indul. Ha egy robot ráugrik, akkor a kisrobot megsemmisül és olajfolt kerül a helyére; a ráugró robot nem szenved sérülést. Ha a kisrobot másik kisrobotnak vagy robotnak ütközik, irányt vált.

 a robotok képesek ütközni, ha azonos helyre érkeznek ugrásuk végén. Ilyenkor a gyorsabb robot összetöri (megsemmisíti) a lassabbat, és kettejük átlagsebességével halad tovább (vektorátlag!).

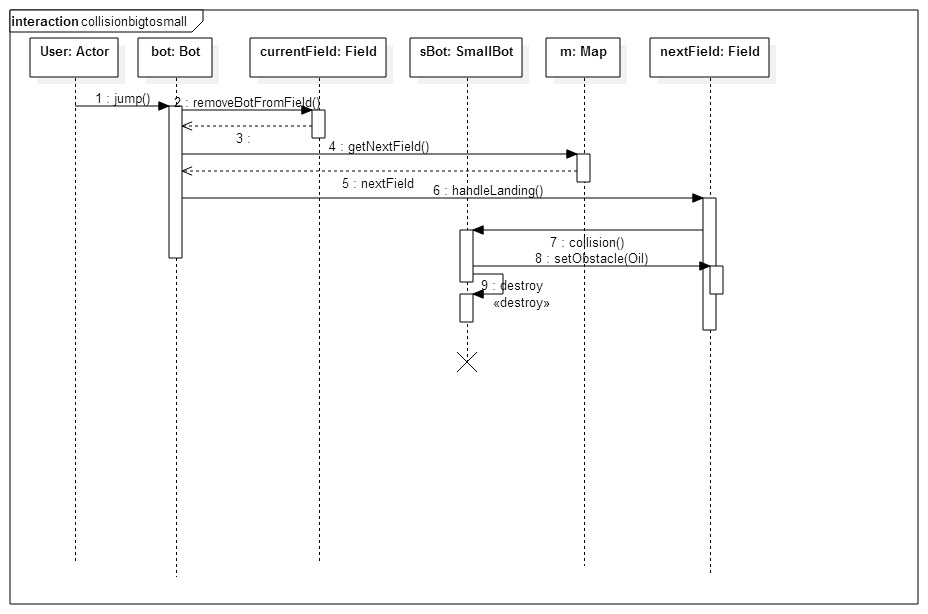
Az első ponthoz létrehoztunk egy removePuttyFromField() nevű metódust, valamint felvettünk egy stepCount nevű változót, melynek a 4 kezdőértéket adjuk, mely csökken eggyel, ha arra a mezőre lépnek, amelyen az elhelyezkedik. Ha eléri a 0-t, a removePuttyFromField() meghívódik, és eltávolítja a ragacsot a mezőről.  
A második ponthoz létrehoztunk egy removeOilFromField() nevű metódust, valamint felvettünk egy roundCount nevű változót, melynek a 0 kezdőértéket adtuk, és eggyel nő, ha minden robot ugrott egyet. Ha eléri a húszat, meghívódik a a removeOilFromField(), és eltávolítja az olajfoltot a mezőről.  
A harmadik ponthoz létrehoztunk egy SmallBot osztályt, melyet a Bot osztályból örököltetünk. A Bot osztályban létrehoztunk egy comapreSpeed(Bot) metódust, mely eldönti, hogy az adott robottól nagyobb sebbességű-e a paraméterként kapott robot. A metódus true-val tér vissza, ha a paraméterül kapott robot sebessége nagyobb, és false-szal, ha nem. A nagyobb sebességűn meghívjuk az aveorageChange() metódust, mely megadja az új irányvektort, a kisebb sebességű pedig megsemmisül. Az átlagsebesség meghatározásához szükség van mindkét robot irányvektorára, így létre kellett hoznunk egy getDirection() nevű metódust is, mely visszaadja a directVector értékét. Ahhoz, hogy a kisrobot megtalálja a hozzá legközelebbi foltot, létre kell hoznunk egy searchNext() nevű függvényt. A searchNext() metódust minden körben meg kell hívnunk, mely miután megkereste a legközelebbi foltot, afelé irányítja a robotot. Ha elértük a keresett mezőt, meghívjuk rajta a removeObstacle() metódust, mely feltakarítja a foltot.

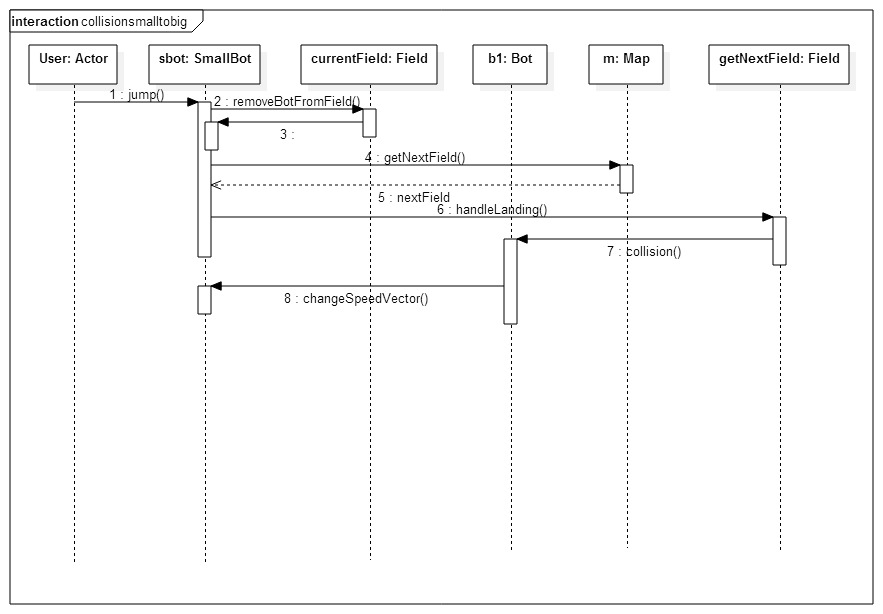
A negyedik ponthoz felvettünk a Bot osztályban egy speed nevű változót, mely a robot sebességét rögzíti(vektorhossz). Továbbá le kell kérdeznünk a currentBot-ot, hogy nagyobb-e a sebessége, és ez alapján dől el, hogy melyik robot semmisül meg.

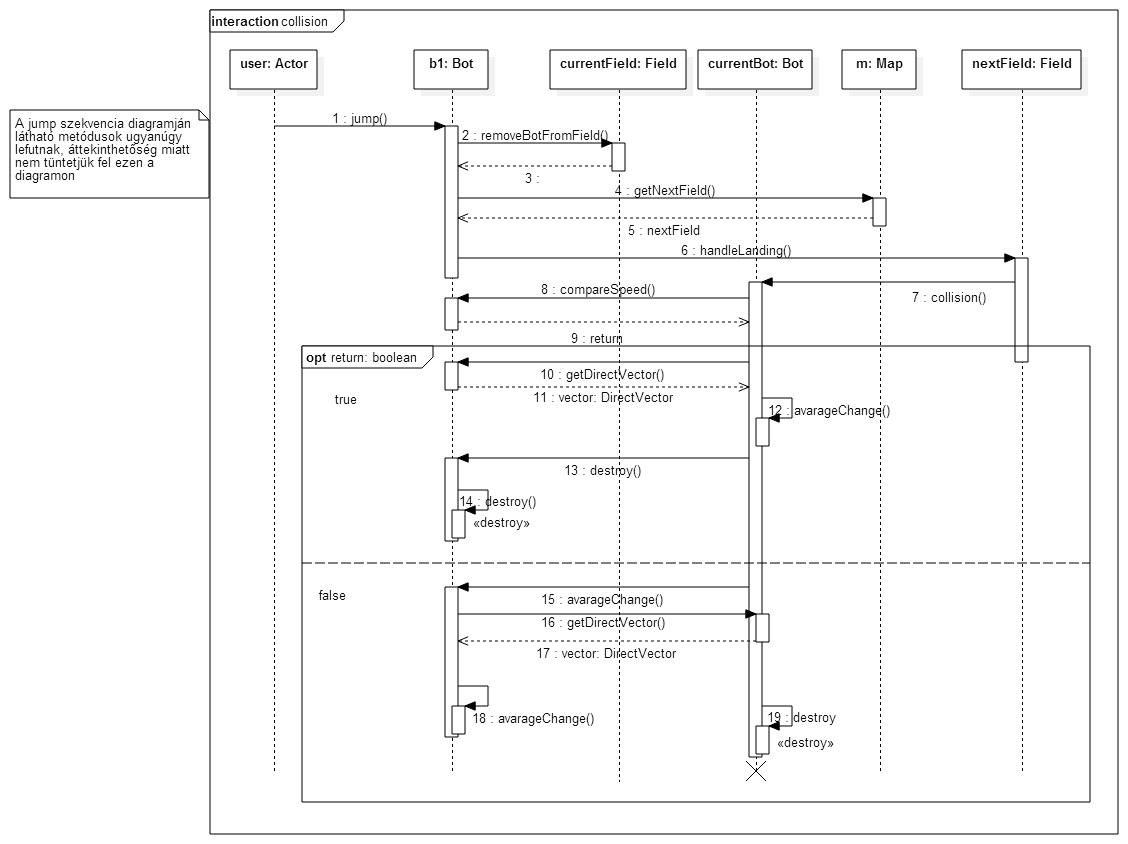
A módosult osztálydiagram:



Az új és a módosult szekvencia diagramok:







## Prototípus interface-definíciója

### Az interfész általános leírása

Az interfész szöveges alapú, parancsait a standard bemenetről várja, kimenete a standard kimenet, ezáltal tesztelése könnyen automatizálható, ha a standard i/o helyett fájlokba irányítjuk a forgalmat egy tesztállomány segítségével. A parancsokat új sor karakterek választják el egymástól, tehát standard bemeneten minden parancsot egy enter kell kövessen. Minden teszteset lefutásának eredménye az elvárt válaszok alapján ellenőrizhető. A kimenet egyrészt tartalmazza, hogy melyik funkcióra adott választ látja a felhasználó/tesztelő, másrészt mutatja az állapotváltozásokat, tehát nagyon hasonló a skeleton kimenetéhez, csak konkretizált és az új bemenetekhez igazított.

A pálya a rajta levő akadályokkal együtt egy fájlban definiálható, de futás közben is a bemeneti nyelv szabályai szerint változtatható.

A bemenet formája: **parancs**[<*space*>**opció1**<*space*>**opció2**….**opcióN**]<*enter*>

Szövegesen megfogalmazva: minden utasítás úgy épül fel, hogy az első szava a parancs, majd ezután minden opciója a parancsnak (amennyiben van neki) egy space-szel elválasztva következik, végül az opciók enumerálása után egy enterrel lehet véglegesíteni az utasítás kiadását.

A teszt jellegéből fakadóan az időt a körök adják, a köröket pedig a felhasználó jump utasításai határozzák meg. Erre azért is szükség van, hogy a tesztelő figyelmét ne kerüljék el esetleges események, amik ha nem lenne a játék körökre osztva addig történnének, amíg eldönti/beírja, hogy hogy nézzen ki a következő tesztelendő állapot, illetve így nem fordulhat elő inkonzisztencia olyan téren, hogy a tesztelő valamit meg akar változtatni, de az már magától más lett közben.

Egy robot irányítására van csak lehetőség, mint a játékban is lesz ez, viszont lehet robotokat elhelyezni a mapen.

### Bemeneti nyelv

/\* Robottal kapcsolatos parancsok \*/

**jump**

**Leírás:** Lényegében ezzel a paranccsal lehet kipróbálni mi fog történni 1 időegység múlva. Ugyanakkor az irányított robot sebességvektornak megfelelő mezőre ugrási kísérletét is ez indítja el.

**Opciók: -**

**changeSpeed**

**Leírás:** Sebességvektor megváltoztatása. A tesztkedvéért nincs egyre lekorlátozva a változtatás.

**Opciók:** x és y irányút vektorkomponensek

**putOil**

**Leírás:** Olaj lerakása egy mezőre

**Opciók:** A mező koordinátái, vagy ha a robot olajlerakását tesztelné a tesztelő, akkor nincs paraméter.

**putPutty**

**Leírás:** Ragacs lerakása egy mezőre.

**Opciók:** A mező koordinátái, vagy ha a robot ragacslerakását tesztelné a tesztelő, akkor nincs paraméter.

**getBotState**

**Leírás:** Ezzel az utasítással lekérdezhető a robot állapota (irányítható-e, mennyi olaja/ragacsa van, az ID-ja, és hogy hol tartózkodik)

**Opciók:** A kérdéses robot ID-ja, vagy üres is lehet és ekkor az épp irányított robot adatai jelennek meg.

**setBotState**

**Leírás:** Robot állapotának megváltoztatása

**Opciók:** isDirectable, oilCount, puttyCount állítása, ebben az adott sorrendben. A mező amin áll csak ugrással módosítható.

/\*Környezettel kapcsolatosak (putPutty és putOil össze van vonva, nincs külön környezeti és roboti)\*/

**putBot**

**Leírás:** Egy robot elhelyezése a mapen. A robot az indulási értékekkel fog rendelkezni. Ha van már valaki a mezőn, akkor sikertelen a lerakás. (Kis robotra se lehet rárakni.)

**Opciók:** x és y koordinátája a field-nek, ahova a robotnak kerülnie kell.

**putCleanerBot**

**Leírás:** Takarító robot lerakása egy mezőre. Ha a mező már foglalt egy másik robot, vagy takarító robot által, akkor sikertelen a parancs végrahajtása.

**Opciók:** x és y koordinátája a field - nek amire kerülnie kell.

**modifyMapWithField**

**Leírás:** Map modifikáció: ha az adott field létezik, megváltoztatja az állapotát, ha nem létezik, létrehozza.

**Opciók:** Vagy a Field class attribútumainak megfelelő típusú értékek felsorolva sorrendben (x, y, currentBotID, isValid, obstacle = Putty/Oil/None), vagy csak az x, y, ekkor default field jön létre.

**advanceTime**

**Leírás:** Idő (körök) pergetése, ha valamilyen teszteset szempontjából a magára hagyott rendszer működése lenne releváns. Ekkor nem történik más (a robotok szemszögéből), mint a robotok átváltanak önműködő üzembe és az aktuális sebességükkel ugranak n-et. Egyenértékű az n db jump utasítással.

**Opciók:** Körök száma (n).

/\*Interaktív funkciók (getBotState-en kívül\*/

**getMap**

**Leírás:** Fieldek felsorolása.

**Opciók:** -

**getField**

**Leírás:** Egy konkrét field állapotának lekérdezése

**Opciók:** x és y koordinátája a field - nek

**getBots**

**Leírás:** Takarító és sima robotok felsorolása.

**Opciók:** -

A Map definiálására egy txt állományból van lehetőség a játék előtt. Ennek formátuma:

**hasBot isValid x y**

Ez a sor ismétlődik a fájlban (mezőnként 1 sor), a következő képpen: **hasBot** 0 vagy 1 lehet, értelem szerűen 0 jelenti, hogy nincs az adott fielden robot, 1, hogy van. **isValid** indikálja, hogy a mezőre lépés zuhanással jár-e a robot számára, 0 ha nem, 1 ha igen.  **x** és **y** egy-egy egész szám, a mező helyzetét határozzák meg.

Ha az adott field, amiről a tesztelő indul a játék elején **hasBot** = 1 értékű, ez a default robot lesz amit irányíthat.

A többszörösen definiált mezők felülíródnak, a legutolsó értékei lesznek érvényesek.

Az egyes értékek space karakterrel kell elválasztva legyenek.

### Kimeneti nyelv

A get metódusok adnak kimenetet, ezek a következők:

*getBotState* **idDirectable oilCount puttyCount ID x y**

*getMap* **x1 y1***,* **x2 y2***,* **x3 y3***,***…**

*getField* **currentBotID isValid obstacle**

*getBots* **bot1ID***,* **bot2ID***,* **... cleanerBot1.x cleanerBot1.y***,* **...**

A félkövér értékek értelemszerűen a már feljebb ismertetett módon ki vannak töltve, tehát az érték fog megjelenni, míg a dőlttel szedett részek konkrét kiírások (a vesszők is).

## Összes részletes use-case

[A use-case-eknek a részletezettsége feleljen meg a kezelői felületnek, azaz a felület elemeire kell hivatkozniuk.

Alábbi táblázat minden use-case-hez külön-külön.]

|  |  |
| --- | --- |
| **Use-case neve** | jump |
| **Rövid leírás** | Ugrik egyet a robottal. |
| **Aktorok** | Tesztelő |
| **Forgatókönyv** | A robotot törli az aktuális mezőről, majd elhelyezi a következő mezőn. A következő mező állapotától függően változtatja a robot állapotát. |

|  |  |
| --- | --- |
| **Use-case neve** | changeSpeed |
| **Rövid leírás** | Megváltoztatja a robot sebességvektorát. |
| **Aktorok** | Tesztelő |
| **Forgatókönyv** | A robot aktuális sebességvektorához adja a kapott x és y irányú vektorkomponenseket. |

|  |  |
| --- | --- |
| **Use-case neve** | putOil |
| **Rövid leírás** | Olajt rak egy mezőre. |
| **Aktorok** | Tesztelő |
| **Forgatókönyv** | Elhelyez egy olajt az aktuális vagy a koordinátáival megadott mezőn. |

|  |  |
| --- | --- |
| **Use-case neve** | putPutty |
| **Rövid leírás** | Ragacsot rak egy mezőre. |
| **Aktorok** | Tesztelő |
| **Forgatókönyv** | Elhelyez egy ragacsot az aktuális vagy a koordinátáival megadott mezőn. |

|  |  |
| --- | --- |
| **Use-case neve** | getBotState |
| **Rövid leírás** | Kiírja a robot állapotát. |
| **Aktorok** | Tesztelő |
| **Forgatókönyv** | A kimenetre írja a robot állapotát (irányíthatóság, olaj/ragacs száma, ID, tartózkodási hely) |

|  |  |
| --- | --- |
| **Use-case neve** | setBotState |
| **Rövid leírás** | Beállítja a robot állapotát. |
| **Aktorok** | Tesztelő |
| **Forgatókönyv** | Az általunk megadott értékre állítja a robot irányíthatóságát valamint az olaj és ragacs számát. |

|  |  |
| --- | --- |
| **Use-case neve** | putBot |
| **Rövid leírás** | Lerak egy robotot a map-re. |
| **Aktorok** | Tesztelő |
| **Forgatókönyv** | A kapott koordinátákra megnézi, hogy elhelyezhető-e a robot. Ha igen, elhelyezi robotot és beállítja a robot valamint a mező állapotát. |

|  |  |
| --- | --- |
| **Use-case neve** | putCleanerBot |
| **Rövid leírás** | Lerak egy kisrobotot a mapre. |
| **Aktorok** | Tesztelő |
| **Forgatókönyv** | A kapott koordinátákra megnézi, hogy elhelyezhető-e a kisrobot. Ha igen, elhelyezi kisrobotot és beállítja a kisrobot valamint a mező állapotát. |

|  |  |
| --- | --- |
| **Use-case neve** | modifyMapWithField |
| **Rövid leírás** | Módosítja a map egy mezőjének az állapotát. |
| **Aktorok** | Tesztelő |
| **Forgatókönyv** | Ha az adott field létezik, megváltoztatja az állapotát, ha nem létezik, létrehoz egy default fieldet. |

|  |  |
| --- | --- |
| **Use-case neve** | advanceTime |
| **Rövid leírás** | Végrehajt megadott számú ugrást. |
| **Aktorok** | Tesztelő |
| **Forgatókönyv** | A robotok átváltanak önműködő üzembe és az aktuális sebességükkel ugranak n-et. Egyenértékű az n db jump utasítással. |

|  |  |
| --- | --- |
| **Use-case neve** | getMap |
| **Rövid leírás** | Felsorolja a field-eket. |
| **Aktorok** | Tesztelő |
| **Forgatókönyv** | A kimenetre írja a map összes field-jét. |

|  |  |
| --- | --- |
| **Use-case neve** | getField |
| **Rövid leírás** | Kiírja egy adott field állapotát. |
| **Aktorok** | Tesztelő |
| **Forgatókönyv** | A kimenetre írja a koordinátáival megadott field állapotát. |

|  |  |
| --- | --- |
| **Use-case neve** | getBots |
| **Rövid leírás** | Felsorolja a map-en lévő robotokat. |
| **Aktorok** | Tesztelő |
| **Forgatókönyv** | A kimenetre írja a robotok és a kisrobotok állapotát/adatait. |

## Tesztelési terv

[A tesztelési tervben definiálni kell, hogy a be- és kimeneti fájlok egybevetésével miként végezhető el a program tesztelése. Meg kell adni teszt forgatókönyveket. Az egyes teszteket elég informálisan, szabad szövegként leírni. Teszt-esetenként egy-öt mondatban. Minden teszthez meg kell adni, hogy mi a célja, a proto mely funkcionalitását, osztályait stb. teszteli. Az alábbi táblázat minden teszt-esethez külön-külön elkészítendő.]

Minden tesztesethez készítünk egy bemeneti fájlt (szövegfájl a kiinduló állapottal), valamint egy hozzá tartozó elvárt kimeneti fájlt. Az egyes tesztek lefuttatása után a kapott kimenetet egyszerűen összehasonlíthatjuk az elvárt kimenettel. Egyezés esetén a teszt sikeres.

|  |  |
| --- | --- |
| **Teszt-eset neve** | Ugrás |
| **Rövid leírás** | A robot ugrását teszteli. |
| **Teszt célja** | Annak ellenőrzése, hogy a robot a megfelelő mezőre érkezik-e. |

|  |  |
| --- | --- |
| **Teszt-eset neve** | Irányváltás |
| **Rövid leírás** | A robot irányítását teszteli. |
| **Teszt célja** | Annak ellenőrzése, hogy az irányvektor módosításakor a robot a megfelelő mezőre ugrik-e. |

|  |  |
| --- | --- |
| **Teszt-eset neve** | Akadály lerakása |
| **Rövid leírás** | Az akadályok elhelyezését teszteli. |
| **Teszt célja** | Annak ellenőrzése, hogy a lerakott akadályok valóban a helyükre kerülnek-e. |

|  |  |
| --- | --- |
| **Teszt-eset neve** | Olajba lépés |
| **Rövid leírás** | A robot olajba való lépésének hatását teszteli. |
| **Teszt célja** | Annak ellenőrzése, hogy ha a robot olajba lép valóban irányíthatatlan állapotba kerül. |

|  |  |
| --- | --- |
| **Teszt-eset neve** | Ragacsba lépés |
| **Rövid leírás** | A robot ragacsba való lépésének hatását teszteli. |
| **Teszt célja** | Annak ellenőrzése, hogy ha a robot ragacsba lép valóban feleződik-e a sebességvektora. |

|  |  |
| --- | --- |
| **Teszt-eset neve** | Kisrobotok ütközése |
| **Rövid leírás** | A kisrobotok ütközését teszteli. |
| **Teszt célja** | Annak ellenőrzése, hogy a kisrobotok ütközése során megsemmisülnek-e és olajfoltot hagynak-e maguk után. |

|  |  |
| --- | --- |
| **Teszt-eset neve** | Robotok ütközése |
| **Rövid leírás** | 2 vagy több robot ütközését teszteli. |
| **Teszt célja** | Annak ellenőrzése, hogy robotok ütközésekor a lassabb összetörik-e, a gyorsabbnak megfelelően módosul-e az irányvektora. |

|  |  |
| --- | --- |
| **Teszt-eset neve** | Robot-kisrobot ütközése |
| **Rövid leírás** | Egy robot és egy kisrobot ütközésének tesztelése. |
| **Teszt célja** | Annak ellenőrzése, hogy ha egy robot egy kisrobottal ütközik, a kisrobot megsemmisül-e és olajfolt lesz-e belőle. |

|  |  |
| --- | --- |
| **Teszt-eset neve** | Kisrobot mozog |
| **Rövid leírás** | A kisrobotok mozgását teszteli. |
| **Teszt célja** | Annak ellenőrzése, hogy a kisrobotok megtalálják-e a legközelebbi akadályokat (ragacs/olajfolt). |

|  |  |
| --- | --- |
| **Teszt-eset neve** | Kisrobot takarít |
| **Rövid leírás** | A kisrobotok takarításának tesztelés. |
| **Teszt célja** | Annak ellenőrzése, hogy a kisrobotok felszedik-e az alattuk lévő foltot. |

## Tesztelést támogató segéd- és fordítóprogramok specifikálása

[Specifikálni kell a tesztelést támogató segédprogramokat.]

A tesztelés megkönnyítéséhez egy segédprogramot hozunk létre, amely összehasonlítja soronként a várt kimeneteleket a programunk által adott kimenetelekkel. Ha valami nem egyezik meg, a program kiírja ezeket a sorokat. Ha nincs eltérés, a „Sikeres teszt” üzenetet kapjuk. A segédprogramot java nyelven fogjuk elkészíteni.

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2015.03.29 10:02 | 3,5 óra | Deák Zsolt | Interface |
| 2015.03.29 14:05 | 4 óra | Poór Bálint | Módosítások |
| 2015.03.30 10:00 | 1 óra | Velenyák János, Poór Bálint | Tesztelés specifikálása,  Módosítás kiegészítés(3. pont) |

# Részletes tervek

# *8.0* Változtatások

A bemeneti nyelvben:

**changeSpeed**

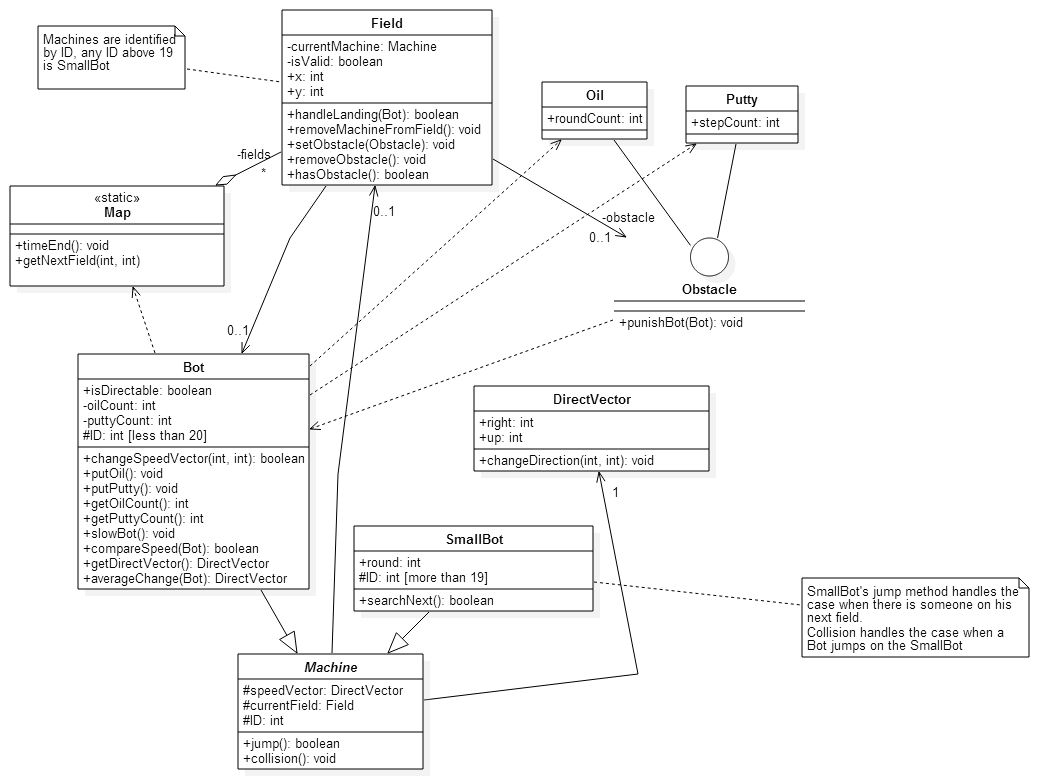
**Leírás:** Sebességvektor megváltoztatása. A tesztkedvéért nincs egyre lekorlátozva a változtatás. A kapott értéket direktben állítja be az irányított robotnak, így a robotot egyhelyben állásra is lehet kényszeríteni.

**Opciók:** x és y irányút vektorkomponensek

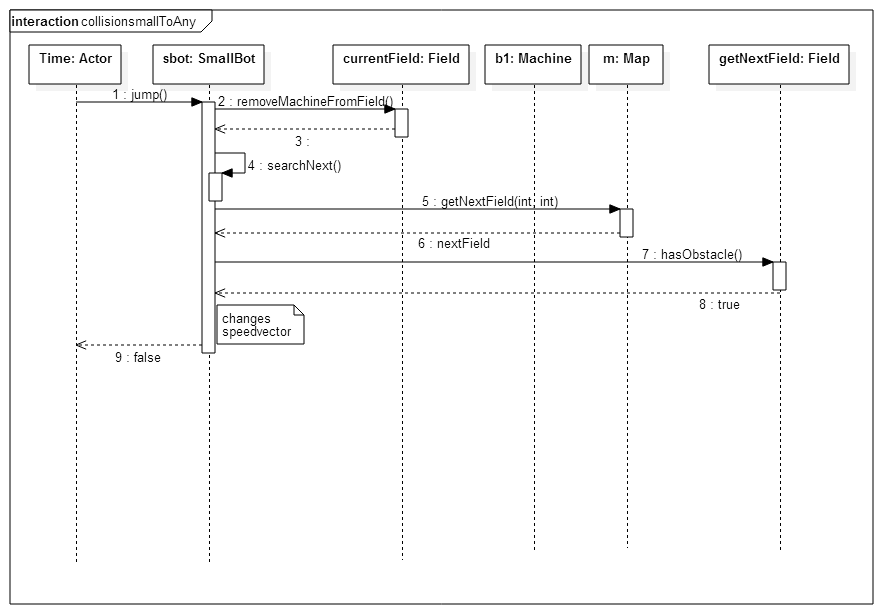
**acquireBot**

**Leírás:** Egy adott Bot elérése melyet irányítani szeretnénk, ezzel megteremtve a multiplayer lehetőséget és a robotok közti váltást a tesztelés megkönnyítése érdekében

**Opciók:** Annak a Botnak az ID-je melyet irányítani szeretnénk.



Változás még, hogy a SmallBot nem hív handleLandinget



## Osztályok és metódusok tervei.

### Bot

#### Felelősség

Akadályok nyilvántartása, mennyiségükről a felhasználó tájékoztatása. Ugrás végrehajtása. Sebességvektor megváltoztatása. Képes akadályokat letenni, valamint sebességét más robottal összehasonlítani.

#### Ősosztályok

Machine

#### Interfészek

Nincs

#### Attribútumok

* **+ boolean isDirectable**: Ha olajra léptünk false lesz az értéke, nem lehet irányítani a robotot, egyébként true, a robot sebességvektora változtatható
* **- int oilCount**: Számon tartja mennyi olaj akadályt rakhat még le a robot
* **- int puttyCount**: Számon tartja mennyi ragacsot rakhat még le a robot
* **# int ID**: A robot azonosítója, kisebb mint 20

#### Metódusok

[Milyen publikus, protected és privát metódusokkal rendelkezik. Metódusonként precíz leírás, ha szükséges, activity diagram is a metódusban megvalósítandó algoritmusról.]

* **+ boolean changeSpeedVector(int, int):** Az adott paraméterekre módosítja robot sebességvektorát, ha ez lehetséges, ha valamiért nem, akkor hamis értékkel tér vissza.
* **+ void putOil(Obstacle):** Lerak egy oil-t a fieldre, amin éppen van.
* **+ void putPutty(Obstacle):** Lerak egy ragacsot a fieldre, amin éppen van.
* + int getOilCount(): Az oilCount attribútum getter fügvénye.
* + int getPuttyCount(): A puttyCount attribútum getter fügvénye.
* + void slowBot(): A robot sebességét felezi.
* + boolean compareSpeed(Bot): összehasonlítja 2 robot sebességét, amik egy mezőn vannak. True-val tér vissza, ha a paraméterül kapott robot sebessége nagyobb, false-al, ha kisebb.
* + DirectVector getDirectVector(): a directVector getter függvénye.
* + DirectVector avarageChange(Bot): A paraméterül kapott robot, és saját magának a sebességvektorainak az átlagával tér vissza.

### DirectVector

#### Felelősség

Irányvektort tárol, illetve módosít.

#### Ősosztályok

Nincs

#### Interfészek

Nem

#### Attribútumok

* **+ int right**: vízszintes koordináta
* **+ int up**: függőleges koordináta

#### Metódusok

* **+ void changeDirection(int, int)**: megváltoztatja az irányvektort a kapott inputra

### Field

#### Felelősség

Akadályok kezelése (rajta lévőké), mi van rajta, tudnia kell magáról, hogy érvényes-e, azaz meghal-e a robot, ha rálép, kezeli, hogy melyik robot van rajta.

#### Ősosztályok

Nincs

#### Interfészek

Nincs

#### Attribútumok

* - **Machine currentMachine:** A robotot tárolja, ami éppen a field-en tartózkodik. Ha a robot leugrik a field-ről, törli az attribútumai közül.
* **- boolean isValid:** true, ha rá lehet lépni a mezőre, false, ha a mező nem a pálya része, ha ilyenkor rálép a robot, meghal
* **+ int x:** A mező horizontális koordinátája
* **+ int y:** A mező vertikális koordinátája

#### Metódusok

* **+ boolean handleLanding(Bot):** A rálépő roboton elvégzi a megfelelő műveleteket, amik a mezőre lépés kapcsán szükségesek (pl akadály érvényesítése), illetve visszaadja, hogy sikerült-e a robotnak landolnia. Jelez a robotnak ütközésnél, elmondja kivel ütközött.
* **+ void removeMachineFromField()**:Robot eltávolítása a fieldről.
* **+ void setObstacle**: Kitüntetett jelentőségű setter metódus. Beállítja az obstacle attribútumot.
* **+ void removeObstacle()**: Ha a kisbot rálép a mezőre, fel tudja takarítani az akadályt
* **+ boolean hasOil()**: True-val tér vissza, ha a mezőn van olaj, false-al, ha nincs
* **+ boolean hasPutty ()**: True-val tér vissza, ha a mezőn van ragacs, false-al, ha nincs

### Map

#### Felelősség

Idő kezelése, mezők nyilvántartása. Kezeli, hogy az ugrások a pályán honnan, hova mennek végbe.

#### Ősosztályok

Nincs

#### Interfészek

Nincs

#### Attribútumok

* **- Field fields**: Mezők eltárolása.

#### Metódusok

* **+ Field getNextField(int x, int y)**: Lekérdezi az x, y koordinátájú mezőt, ez egy adott mező elérésére is alkalmas függvény, de a nevéből is adódik, hogy a megfelelő használata a következő mező elkérése.
* **+ void timeEnd()**: Kezeli az idő lejártát

### Machine

#### Felelősség

A Bot és a SmallBot ősosztálya. Sebességvektort, és a jelenlegi mezőt is nyilvántartja. Kezeli az ugrást és az ütközést.

#### Ősosztályok

Nincs

#### Interfészek

Nincs

#### Attribútumok

* **# DirectVector speedVector:** A robot sebességvektora
* **# Field currentField:** az a mező, amin a robot éppen tartózkodik
* **# int ID:** a robot azonosítója

#### Metódusok

* **+ boolean jump():**A jelenlegi sebességvektor felhasználásával elvégzendő ugrásra utasítja a robotot, illetve visszaadja, hogy sikerült-e odaugrani.
* **+ void collision():** Az ütközésért vizsgálandó, valamint végrehajtandó dolgokért felel

### Obstacle

#### Felelősség

Interface az Oil, és Putty osztályok számára. Adott büntetés elvégzése a rálépő roboton.

#### Ősosztályok

Nincs

#### Interfészek

Nincs

#### Attribútumok

Nincs

#### Metódusok

* **void punishBot(Bot bot)**: Elvégzi a büntetést a paraméterben kapott roboton.

### Oil

#### Felelősség

Obstacle inteface megvalósítása. Adott büntetés elvégzése a rálépő roboton: sebessség vektor megváltoztatását tiltja le.

#### Ősosztályok

Nincs

#### Interfészek

**Obstacle**

#### Attribútumok

Nincs

#### Metódusok

* **void punishBot(Bot bot)**: Elvégzi a büntetést a paraméterként kapott roboton.

### Putty

#### Felelősség

Obstacle inteface megvalósítása. Adott büntetés elvégzése a rálépő roboton: sebesség vektor nagyságát megfelezi.

#### Ősosztályok

Nincs

#### Interfészek

**Obstacle**

#### Attribútumok

Nincs

#### Metódusok

* **void punishBot(Bot bot)**: Elvégzi a büntetést a paraméterként kapott roboton.

### SmallBot

#### Felelősség

A Machine ősosztály megvalósítása. Kis robotok, amik a pályát takarítják, valamint számon tartják, hogy mióta vannak a pályán. (egy adott idő után megsemmisülnek)

#### Ősosztályok

Machine

#### Interfészek

Nincs

#### Attribútumok

* **+ int round**: számon tartja, hogy hányadik körben van a robot.
* **# int ID**: A robot azonosítója, 19-től nagyobb szám

#### Metódusok

* **+ boolean searchNext():** Megkeresi a következő feltakarítandó akadályt, ha nem talál, false-al tér vissza

## A tesztek részletes tervei, leírásuk a teszt nyelvén

[A tesztek részletes tervei alatt meg kell adni azokat a bemeneti adatsorozatokat, 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.]

A tesztekben a pályán a megfelelő számú és pozíciójú robot a leírásnak megfelelő kezdőhelyzetben rendelkezésre fog állni, ami alól kivételek azok a robotok/akadályok, melyek elhelyezése explicite benne van a leírásban.

A tesztelés során a felhasználók mindig az 1-es ID-jú robotot irányítják.

### Ugrás

* **Leírás**

A játék legalapvetőbb funkciójának tesztelése. A robot egy üres pályán ugrik egyet az adott sebességvektorával. A pálya elég nagy kell legyen, hogy a robot ne essen le először. Másodikra kiugrunk a pályáról. Ez például úgy lehetséges, ha a pálya -1 < x < 12 például. Ha a robot a Map-ben nem tárolt mezőre ugrik, az invalidnak számít, így leesik.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Megfelelő default értékek, tehát helyesen működő Robot és Map inicializálás. A robot a kezdeti 1,1 sebességvektornak megfelelő x és y koordinátákra kell érkezzen az ugrást követően. Várható a sikertelen landolás, vagy rossz helyre érkezés, mivel ez az egyik leghosszabb szekvencia a játékban. Lehetséges még, ha rossz az irányváltás, hogy nem sikerül. Ez esetben az Irányváltás teszt lefutása után kiderül, hogy ott a hiba.

* **Bemenet**

getBotState

jump

getBotState

changeSpeed -2 0

jump

getBotState

ugrasteszt-map.txt:

//hasBot isValid x y

1 0 0 0

0 0 1 1

* **Elvárt kimenet**

getBotState 1 3 3 1 0 0

getBotState 1 3 3 1 1 1

getBotState 0 0 0 1 -1 1

### Irányváltás

* **Leírás**

Az ugrás a játék alapköve, viszont semmit sem ér az egész, ha nem irányítható a robot. Ennek a tesztelését fogja végezni ez a teszteset. Az előző esethez hasonlóan egy üres pályán váltunk irányt, majd megvizsgáljuk a hatását.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Irányváltás ellenőrzése. Maga az irányvektor átállítása csak egy sima vektor átírása, viszont a megfelelő működést egy némileg komplexebb logika garantálja. Ezen a ponton előfordulhat, hogy például valahol beragad egy régi érték és így nem lép életbe az irányváltás. Elméletileg az irányvektor kezelése is jól kell működjön, ha hibátlan az ugrási szekvencia.

* **Bemenet**

getBotState

changeSpeed 3 1

jump

getBotState

changeSpeed 0 0

jump

getBotState

iranyvaltasteszt-map.txt:

1 0 0 0

0 0 3 1

* **Elvárt kimenet**

getBotState 1 3 3 1 0 0

getBotState 1 3 3 1 3 1

getBotState 1 3 3 1 3 1

### Akadály lerakása (olaj, ragacs)

* **Leírás**

A robot a kezdőpozícióján elhelyez egy olajfoltot, lép egyet, majd lerak egy ragacsot.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Mindkét fajta akadály elhelyezésének képessége. A funkcionalitás relatív biztonsággal ellenőrizhető, ha a lerakás után elkérjük az adott mezők állapotát. Ezt a tesztesetet szintén lehet egy tetszőleges üres pályán végezni. Várható, hogy nem megfelelő implementáció esetén nem fog módosulni a field állapota. Azt is ellenőrizzük, hogy üres tarsollyal mit csinál.

* **Bemenet**

getBotState

getField 0 0

putOil

jump

putPutty

getField 0 0

getField 1 1

jump

setBotState 1 0 2

putOil

getField 2 2

akadalyteszt-map.txt:

1 0 0 0

0 0 1 1

0 0 2 2

* **Elvárt kimenet**

getBotState 1 3 3 1 0 0

getField 1 1 None

getField 0 1 Oil

getField 1 1 Putty

getField 1 1 None

### Olajba lépés, ragacsba lépés

* **Leírás**

Olajba lépéskor letiltódik a robot irányíthatósága, ragacsba lépéskor feleződik a speedvectora. Mindkettő egyszerűen tesztelhető a robot állapotának lekérdezésével ismét. A különböző akadályok esetén különböző büntetéseket kell elszenvedjen a robot, viszont ismét üres mezőre érve ezek hatása megszűnik. Az egyszerűség kedvéért a Map releváns része ebben az esetben egy 0,0 pozíciójú tiszta mező, egy 1,2-es olajos, egy 2,4-es ragacsos és egy 3,5-ös tiszta. (Felezéskor, mivel csak egész számnyi mezőt lehet átugrani, a matematika szabályainak megfelelő kerekítést alkalmazzuk így marad az 1 1.)

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A landolás után az adott mezőn lévő akadály el kell végezze hatását a roboton. Ez a két külön akadálytípus esetén más eredményt ad. A funkcionalitás az ugrás azon részesete, mikor a field, illetve az azon lévő obstacle is aktorrá válik és megváltoztat egy állapotot. Hibalehetőség, hogy rossz módosítás történik, illetve, hogy tiszta mezőre érkezéskor nem szűnik meg a hatás.

* **Bemenet**

putOil 1 2

putPutty 2 4

changeSpeed 1 2

jump

getBotState

changeSpeed 5 7

jump

getBotState

jump

getBotState

akadalybalep-map.txt:

1 0 0 0

0 0 1 2

0 0 2 4

0 0 3 5

* **Elvárt kimenet**

getBotState 0 3 3 1 1 2

getBotState 1 3 3 1 2 4

getBotState 1 3 3 1 3 5

### Kisrobotok ütközése

* **Leírás**

Mivel körökre van osztva a játék, az egymásnak ütközés minden esetben azt jelenti, hogy az egyik nekimegy a másiknak, ami pedig egyhelyben áll. Elindítunk először egy kisrobotot egy üres pályán 0,1 vectorral, a 0,0 pontból. Elé helyezünk egy nagy robotot, 0,0 sebességet adunk neki, lepergetjük az időt és figyeljük az eredményt.

(getField 0 0 ha jól működik megadja a kisrobot ID-ját, ami 20).

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A kisrobot irányváltása az ellenőrzés tárgya, amennyiben nekiütközik valaminek. Az ütközés körében a kisrobot a foglalt mezőn van, viszont nem ő foglalja a mezőt, tehát a mező szemszögéből olyan mintha már nem lenne ott. Hiba lehet, ha nincs lekezelve merre menjen a kisrobot, ha nincs akadály a pályán (elvileg ekkor a speedvectora irányába).

* **Bemenet**

getField 0 0

putBot 0 2

changeSpeed 0 0

advanceTime 2

getBotState 20

getField 0 2

advanceTime 1

getBotState 20

kisrobotok-map.txt:

1 0 0 0

0 0 0 1

0 0 0 2

* **Elvárt kimenet**

getField 20 1 None

getBotState 0 0 0 20 0 2

getField 1 1 None

getBotState 0 0 0 20 0 1

### Robot-kisrobot ütközése

* **Leírás**

A map (0,0) koordinátájú mezőjére teszünk egy robotot, (1,1) koordinátájú mezőjére pedig egy kisrobotot, a robot sebességvektorát 1,1-re állítjuk így ráugrunk a kis robotra, és megnézzük ennek a mezőnek az állapotát.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Azt ellenőrizzük ,hogy ha egy robot ugrik egy olyan mezőre amelyen van egy kisrobot, akkor megsemmisül-e a kisrobot,és olaj kerül-e erre a mezőre.

* **Bemenet**

getField 1 1

changeSpeed 1 1

jump

getBots

getField 1 1

getBotState

robot-kisrobot-map.txt:

1 0 0 0

1 0 1 1

* **Elvárt kimenet**

getField 20 1 None

getBots 1

getField 1 1 Oil

getBotState 1 3 3 1 1 1

### Kisrobot mozog

* **Leírás**

Elhelyezünk egy kisrobotot a pályán és mellé különbőző távolságokra ragacsot,majd megnézük hogy tényleg a helyes ragacsot találta e meg a kisrobot.

A (0,0) és a (0,5) koordinátájú mezőre teszünk egy ragacsot, a (2,0)-ra pedig egy kisrobotot

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Ellenőrizzük, hogy a kisrobot megtalálja az őházza legközelebbi akadályokat.

* **Bemenet**

putPutty 0 0

putPutty 0 5

getField 0 0

advanceTime 2

getField 0 0

kisrobot-mozog-map.txt:

0 0 0 0

0 0 1 0

1 0 2 0

0 0 0 5

* **Elvárt kimenet**

getField 0 1 Putty

getField 20 1 Putty

### Kisrobot takarít

* **Leírás**

A (0,0) koordinátájú Fieldre beállítunk egy kisrobotot az (1,1) re pedig egy ragacsot, ráugrunk a kisrobottal erre a mezőre és megnézük hogy a mező állapota tényleg megváltozik-e

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Ellenőrizük,hogy a kisrobot tényleg törli-e a mezőn lévő ragacsot, vagy olajat.

* **Bemenet**

putPutty 1 1

getField 1 1

changeSpeed 1 1

jump

changeSpeed 0 0

advance(2)

getField 1 1

kisrobot-takarit-map.txt:

1 0 0 0

0 0 1 1

* **Elvárt kimenet**

getField 0 1 Putty

getField 20 1 None

## A tesztelést támogató programok tervei

A tesztadatokat, és az elvárt kimeneteket egyszerű szövegszerkesztő programmal állítjuk elő (Wordpad). A tesztelés úgy zajlik majd, hogy egy batch fájl lefuttatja a prototípust az adott tesztesetnek megfelelő paraméterekkel. A futtatás során elkészült, a teszt kimenetét tartalmazó fájlt az általunk készített Java program segítségével összehasonlítjuk az elvárt kimenetet tartalmazó szövegfájllal. Ez a program paraméterként fogja átvenni a két állományt. Amennyiben a két fájl tartalma nem egyezik meg, a konzolon kiírja azt, hogy melyik sorban volt először eltérés.

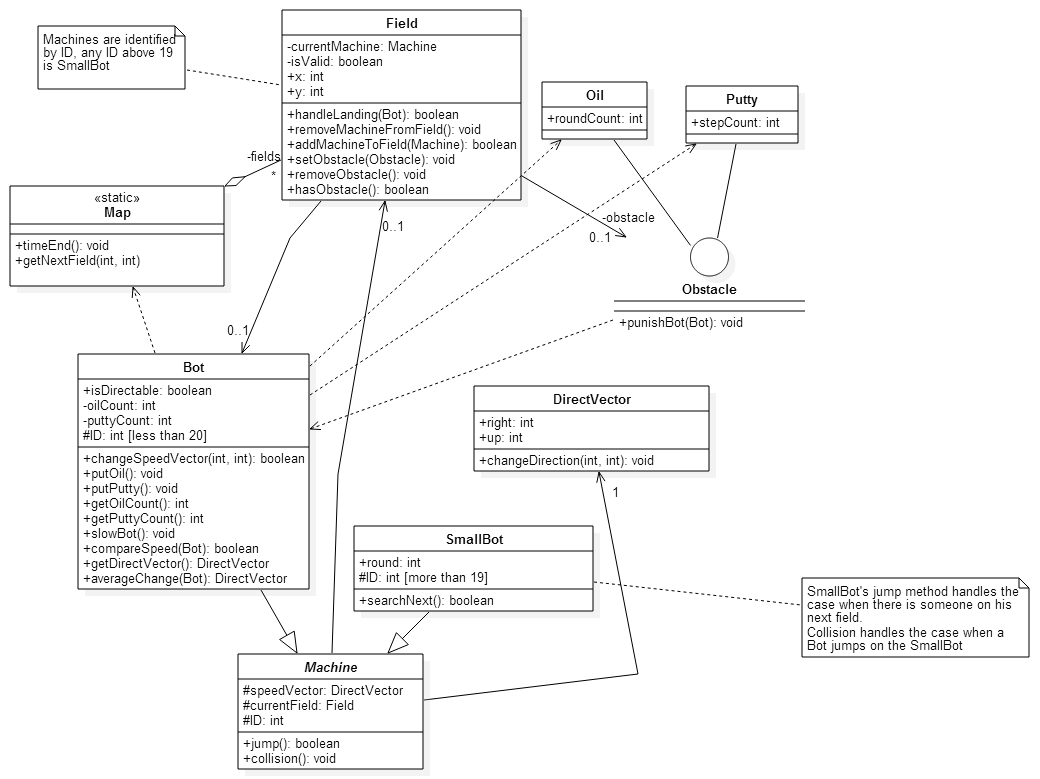
## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2015.03.31 19:12 | 2,5 óra | Deák Zsolt | Tesztesetek 1, 2, 3 |
| 2015.04.1 6:48 | 3,5 óra | Deák Zsolt | 0. Változtatások, Tesztesetek 4, 5 |
| 2015.04.02 02:00 | 1 óra | Poór Bálint  Velenyák János | 8.1.1 8.1.2 8.1.3 8.1.4  Osztályok leírása |
| 2015.04.03:20:32 | 1,5 óra | Pogátsa Áron | Tesztesetek |
| 2015.04.03 23:00 | 1 óra | Velenyák János | 8.1 befejezése |
| 2015.04.06 15:00 | 1 óra | Mitterer Márton | Testesetek kiegészítése |

# Prototípus beadása

***10.0 Változtatások***

Added setter to currentMachine



*getBots* **bot1ID***,* **bot2ID***,* **... cleanerBot1ID, cleanerBot2ID…**

A player kezdéskor a 0. robotot irányítja.

## Fordítási és futtatási útmutató

### Fájllista

|  |  |  |  |
| --- | --- | --- | --- |
| **Fájl neve** | **Méret** | **Keletkezés ideje** | **Tartalom** |
| Proto.java | 16KB | 2015.04.17 | Proto osztál(main) |
| Machine.Java |  | 2015.04.17 | Machine osztály |
| SmallBot.java | 6KB | 2015.04.17 | SmallBot osztály |
|  |  | 2015.04.17 |  |
| change\_speed.txt | 1KB | 2015.04.17 | első automatikus teszteset |
| change\_speed.txt | 1KB | 2015.04.17 | első teszteset várt kimenete |
| jump | 1KB | 2015.04.17 | második teszteset |
| jump\_output | 1KB | 2015.04.17 | második teszteset kimenete |
| map.txt | 1KB | 2015.04.17 | pályát definiáló fájl |
| output.txt | 0KB | 2015.04.17 | az automatikus tesztek kimeneti fájlja(csak az utolsó teszt kimenete marad benne) |
|  |  |  |  |
| Bot.java | 3KB | 2015.03.22 | Bot osztály |
| DirectVector.java | 1KB | 2015.03.22 | DirectVector osztály |
| Field.java | 2KB | 2015.03.22 | Field osztály |
| Map.java | 1KB | 2015.03.22 | Map osztály |
| Obstacle.java | 1KB | 2015.03.22 | Obstacle osztály |
| Oil.java | 1KB | 2015.03.22 | Oil osztály |
| Putty.java | 1KB | 2015.03.22 | Putty osztály |

### Fordítás

Egy (jre 1.7-tel kompatibilis) java project-be be kell importálni, és lefordítani, illetőleg bármely ezzel analóg eljárás megfelel. Ilyen java project-et tud létrehozni például az Eclipse.

Ezután hozzá kell adni a projekthez a test foldert is, ez lehetőleg az src mappa mellé kerüljön.

### Futtatás

A fordított állományokat futtatni tudó környezet alkalmazása szükséges, vagy szintén ezzel analóg mód. Az Eclipse fejlesztőkörnyezet ezzel a funkcióval is bír.

## Tesztek jegyzőkönyvei

### Change speed

|  |  |
| --- | --- |
| **Tesztelő neve** | Deák Zsolt |
| **Teszt időpontja** | 2015.04.20 10:00 |

|  |  |
| --- | --- |
| **Tesztelő neve** | Deák Zsolt |
| **Teszt időpontja** | 2015.04.20 9:55 |
| **Teszt eredménye** | Hiba az első automatikus tesztfuttatásban(changeSpeed) |
| **Lehetséges hibaok** | Rossz formátumú kimenet generálása |
| **Változtatások** | getBotState függvény megváltoztatása |

### Jump

|  |  |
| --- | --- |
| **Tesztelő neve** | Deák Zsolt |
| **Teszt időpontja** | 2015.04.20 10:18 |

|  |  |
| --- | --- |
| **Tesztelő neve** | Deák Zsolt |
| **Teszt időpontja** | 2015.04.20 10:08 |
| **Teszt eredménye** | jump\_output.txt is not identical to the output.txt's content  The problem occured at line: getBotState 1 3 3 1 0 0 |
| **Lehetséges hibaok** | Nem lett hozzáigazítva a kimeneti elvárás ID a változtatásokhoz |
| **Változtatások** | jump\_output.txt módosítása |

|  |  |
| --- | --- |
| **Tesztelő neve** | Deák Zsolt |
| **Teszt időpontja** | 2015.04.20 10:09 |
| **Teszt eredménye** | jump\_output.txt is not identical to the output.txt's content  The problem occured at line: getBotState 1 3 3 0 0 0 |
| **Lehetséges hibaok** | Rossz helyen van a bot |
| **Változtatások** | buildMapFromFile -hoz teardown hozzáadása |

## Értékelés

|  |  |
| --- | --- |
| **Tag neve** | **Munka százalékban** |
| Deák Zsolt | 38.43 |
| Mitterer Márton | 7.85 |
| Pogátsa Áron | 14.46 |
| Poór Bálint Imre | 19.01 |
| Velenyák János | 20.25 |

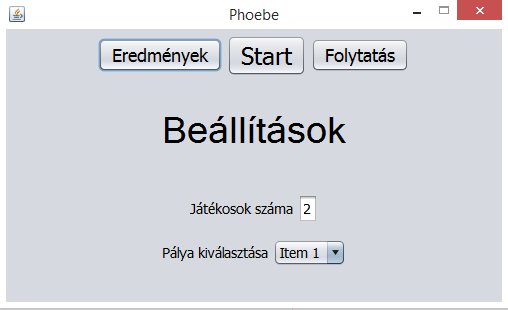
## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2015.04.18 08:12 | 4 óra | Deák Zsolt | Egyeztetés, specifikációhoz igazítás, implementáció |
| 2015.04.18 17:35 | 5 óra | Deák Zsolt | Proto implementáció |
| 2015.04.19 21:30 | 4 óra | Poór Bálint  Velenyák János | Proto  függvények implementálása |
| 2015.04.17 12:00 | 3 óra | Poór Bálint Velenyák János | Bot, SmallBot, Field, DirectVector függvényei implementálása |
| 2015.04.19 22:30 | 3 óra | Pogátsa Áron | Proto  függvények implementálása |
| 2015.04.19 16:00 | 3,5 óra | Mitterer Márton | Proto implementáció |
| 2015.04.20 8:00 | 3,5 óra | Deák Zsolt | Proto befejezése, dokumentáció, javítások |

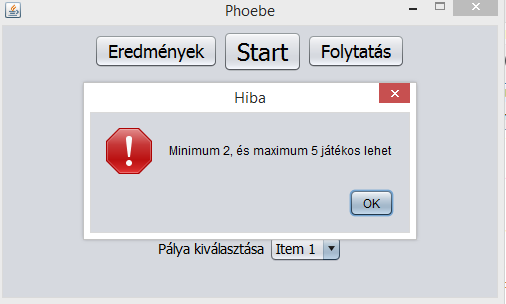
# Grafikus felület specifikációja

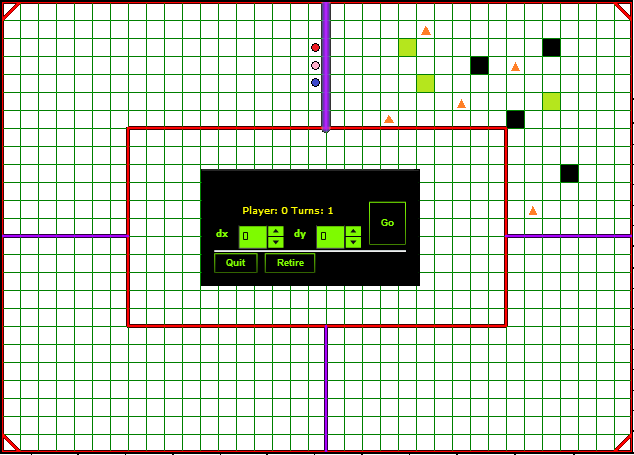
## A grafikus interfész

A program menüje egy ablakból fog állni. Az alapvető funkciók is innen indíthatók majd el, pl. Játék indítása, Mentett játék folytatása, Eredménytábla megtekintése. Ennek az ablaknak egy része fog szolgálni a játék beállításaira. Itt meg lehet majd adni, hogy milyen legyen a pálya, valamint hány játékos legyen a pályán. Ha itt nem megfelelő adatokat visz be a felhasználó, azt ellenőrizzük, valamint erről értesítjük, hogy mit lehet megfelelő bemenetként megadni.



A felugró ablak hibás bemenet esetén a következő lesz:





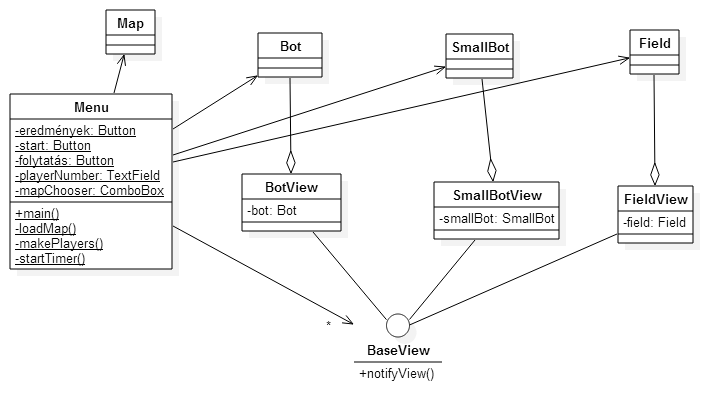
## A grafikus rendszer architektúrája

[A felület működésének elve, a grafikus rendszer architektúrája (struktúra diagramok). A struktúra diagramokon a prototípus azon és csak azon osztályainak is szerepelnie kell, amelyekhez a grafikus felületet létrehozó osztályok kapcsolódnak.]

### A felület működési elve

A grafikus felület megvalósításához push alapelvű modellt választottunk. Ennek az a lényege, hogy változás esetén a modellobjektumok értesíteni fogják a grafikus megjelenítésért felelős objektumokat, majd azok végrehajtják az újrarajzolást. Minden modellobjektumhoz tartozni fog egy megjelenítésért felelős objektum. Ezek, a megjelenítésért felelős objektumok, a BaseView interfészt fogják megvalósítani. A modell és a megjelenítés közti kommunikáció leegyszerűsítésének érdekében minden modell-megjelenítő pár fog egymásra tartalmazni referenciát. A bővíthetőséget szem előtt tartva azonban a modell objektumok BaseView típusú referenciát fognak tárolni, így bármikor egyszerűen lecserélhető lesz a grafikus alrendszer. A BaseView interfész a +notifyView(): void függvény fejlécét fogja tartalmazni, változás esetén ezen keresztül értesítheti a modell a megjelenítő osztályt, amely a getX(), getY() függvényekkel kérdezheti le a kirajzolandó elemek pályán elfoglalt helyét.

### A felület osztály-struktúrája



## A grafikus objektumok felsorolása

[Az új osztályok felsorolása. Az régi osztályok közül azoknak a felsorolása, ahol változás volt. Ezek esetén csak a változásokat kell leírni.]

### BaseView<<interface>>

#### Felelősség

A megjelenítő osztályok interface, minden osztály megvalósítja, aki a grafikus megjelenítésért felelős.

#### Ősosztályok

nincs

#### Interfészek

nincs

#### Attribútumok

nincs

#### Metódusok

* **+ void notifyView():** lekéri a kirajzolandó objektumoktól a kérdéses attribútumokat,majd kirajzolja az objektumot

### BotView

#### Felelősség

A Robotok kirajzolásáért felelős

#### Ősosztályok

nincs

#### Interfészek

BaseView

#### Attribútumok

* **-bot**:**Bot**

Referencia a Bot objektumra

#### Metódusok

* **+ void notifyView():** lekéri a kirajzolandó objektumoktól a kérdéses attribútumokat,majd kirajzolja az objektumot

### SmallBotView

#### Felelősség

A SmallBotok kirajzolásáért felelős

#### Ősosztályok

nincs

#### Interfészek

BaseView

#### Attribútumok

* **-smallBot**: **SmallBot**

Referencia a SmallBot objektumra

#### Metódusok

* **+ void notifyView():** lekéri a kirajzolandó objektumoktól a kérdéses attribútumokat,majd kirajzolja az objektumot

### FieldView

#### Felelősség

A Fieldek kirajzolásáért felelős

#### Ősosztályok

nincs

#### Interfészek

BaseView

#### Attribútumok

* **-field Field**

Referencia a field objektumra

#### Metódusok

* **+ void notifyView():** lekéri a kirajzolandó objektumoktól a kérdéses attribútumokat,majd kirajzolja az objektumot

### Menu

#### Felelősség

A grafika és a logika inicializálása, irányítása. A user bemenetei alapján összerakja a játékban szükséges objektumokat.

#### Ősosztályok

nincs

#### Interfészek

nincs

#### Attribútumok

* **-eredmények**: **Button**

A statisztikákat lehet megtekinteni erre a gombra kattintva

* **-start**: **Button**

A játék elindítása

* **-folytatás**: **Button**

Előzőleg abbahagyott játék folyatása, ha még nem ért véget.

* **-playerNumber**: **TextField**

Bemeneti mező amellyel meg lehet adni hány játékos fog játszani.

* **-mapChooser**: **ComboBox**

Az előre definiált pályák közül lehet választani.

#### Metódusok

* **+void main()**:

Az applikáció main függvénye, elindítja a programot, összerakja a Menu grafikus felületét.

* **-void loadMap()**:

A kiválasztott map betöltése (Map felépítése és kirajzolása).

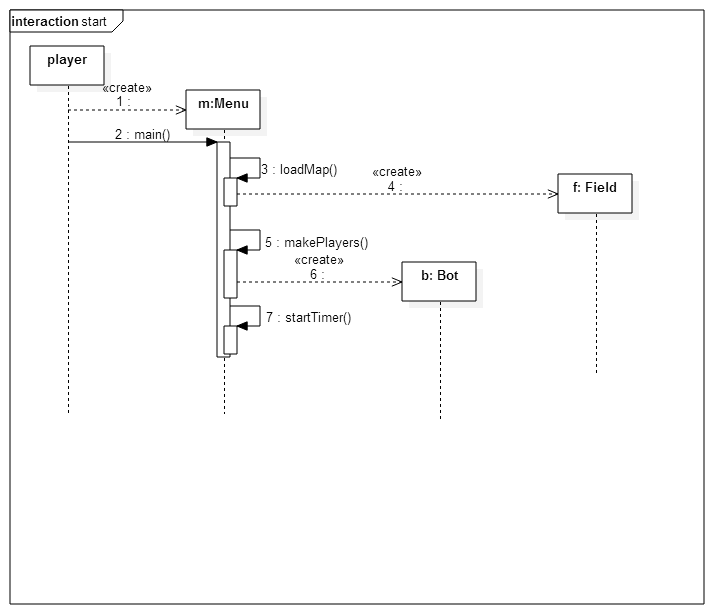
* **-void makePlayers()**:

Létrehozza a és Fieldekhez rendeli a Botokat.

* **-void startTimer()**:

Elindítja az időt.

## Kapcsolat az alkalmazói rendszerrel



## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2015.04.24 | 1 óra | Pogátsa Áron | 11.2 Grafikus rendszer architekrúrája, |
| 2015.04.25 | 2 óra | Velenyák János | 11.1 Menü megírása, felépítése |
| 2015.04.27 9:00 | 2 óra | Deák Zsolt | Diagramok, leírások kiegészítése |
| 2015.04.27 11:00 | 1 óra | Pogátsa Áron | 11.1 Grafikus felület |

1. Ld, 2.1.4 Hivatkozások [↑](#footnote-ref-1)
2. Ld. 2.2.5 Feltételezések, kapcsolatok [↑](#footnote-ref-2)
3. Ld. 2.2.5 Feltételezések ,kapcsolatok [↑](#footnote-ref-3)