**14. Összefoglalás**

26 – five\_guys

Konzulens:

Huszerl Gábor

Csapattagok

| Lekli Zsolt | JENKHQ | zsoltlekli@gmail.com |
| --- | --- | --- |
| Fejes Ábel | AF1Y1C | fejesabel@gmail.com |
| Zelch Csaba | LK0617 | zch03po@gmail.com |
| Szolár Soma | E05SC6 | somaszolar1@gmail.com |
| Toronyi Zsombor | S8F7DV | toronyizsombi@gmail.com |

2025. 05. 23

1. **Összefoglalás**
   1. ***A projektre fordított összes munkaidő***

| **Tag neve** | **Munkaidő (óra)** |
| --- | --- |
| Lekli Zsolt | 71,5 |
| Fejes Ábel | 99 |
| Zelch Csaba | 131,5 |
| Szolár Soma | 102,75 |
| Toronyi Zsombor | 76 |
| **Összesen** | 480,75 |

***• A feltöltött programok forrássorainak száma***

| **Fázis** | **Kódsorok száma** |
| --- | --- |
| Szkeleton | 2072 |
| Prototípus | 2566 |
| Grafikus változat | 3512 |
| **Összesen** | 8150 |

* 1. ***Projekt összegzés***
     1. **Mit tanultak a projektből konkrétan és általában?**

Általában azt tanultuk meg, hogy hogyan kell csapatban dolgozni úgy, hogy minél gördülékenyebben menjen a munka. Továbbá megtanultuk, hogy hogyan érdemes elkezdeni egy szoftver megvalósítását a nulláról, hogyan kerülhetjük el a fatális hibákat megfelelő tervezéssel.

Konkrétan megtanultuk a model-view-controller tervezési mintát, és megtanultuk, hogyan kell pontos szekvencia, osztály és kommunikációs diagramokat csinálni.

* + 1. **Mi volt a legnehezebb és a legkönnyebb?**

A legnehezebb a csapattagok közötti feladatkiosztás volt, és a megfelelő időbeosztás, hogy el tudjuk végezni a részfeladatokat.

A legkönnyebb része a projektnek a megfelelő tervezés utáni kódolás volt, mivel ezeket a feladatokat gyorsan fel tudtuk osztani és a dokumentációk alapján gyorsan el tudtuk végezni.

* + 1. **Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?**

Nem.

* + 1. **Ha nem, akkor hol okozott ez nehézséget?**

Pszeudokód írás.

* + 1. **Milyen változtatási javaslatuk van?**

-

* + 1. **Milyen feladatot ajánlanának a projektre?**

Szerintünk hasonló játék elkészítése élvezetes feladat lenne.

* + 1. **Egyéb kritika és javaslat**

-

**2.Követelmény, projekt, funkcionalitás**

26 – five\_guys

Konzulens:

Huszerl Gábor

Csapattagok

| Lekli Zsolt | JENKHQ | zsoltlekli@gmail.com |
| --- | --- | --- |
| Fejes Ábel | AF1Y1C | fejesabel@gmail.com |
| Zelch Csaba | LK0617 | zch03po@gmail.com |
| Szolár Soma | E05SC6 | somaszolar1@gmail.com |
| Toronyi Zsombor | S8F7DV | toronyizsombi@gmail.com |

# Követelmény, projekt, funkcionalitás

## *Bevezetés*

### Cél

* A projekt kereteinek, a csapat működésének definiálása.
* Annak leírása, hogy a megrendelőnek milyen elvárásai lehetnek az eredményről a specifikáció alapján.
* A specifikáció pontosítása.

### Szakterület

A kialakítandó szoftver egy játék program, amit egyszerre többen használnak.

A játékosok egymást közt versengenek.

A játékot szinte bárki használhatja, játszás, kikapcsolódás és egymás közötti versengés céljára.

A felhasználóknak csak minimális előképzettséggel kell rendelkezniük, hogy tudják használni a programot. Tehát speciális tudás nem szükséges a használatra.

### Definíciók, rövidítések

**Funkcionális követelmény** – A szoftverrel szemben támasztott funkcionális elvárást írja le.

**Nemfunkcionális követelmény** - A szofvterrel szemben támasztott minőségi, teljesítménybeli, vagy egyéb elvárást írja le.

**Projekt** – Egy adott cél érdekében végzett szervezett munka, amelynek van kezdete és vége.

**Funkcionalitás** – Egy szoftver azon képességei, amelyekkel bizonyos feladatokat el tud látni.

**Specifikáció** – A rendszer követelményeire vonatkozó részletes leírás.

**Játék inicializálása** – A játék kezdő állapotának beállítása, például szereplők kiválasztása és pálya létrehozása.

**Játék befejezése** – Az a folyamat, amely során a játék véget ér és a győztesek meghatározásra kerülnek.

### Hivatkozások

-

### Összefoglalás

A dokumentum lépésről lépésre vezeti be az olvasót a szoftver részleteibe. A szerkezete átgondolt, az egymást követő alpontok logikailag összefüggenek és egymásra épülnek.

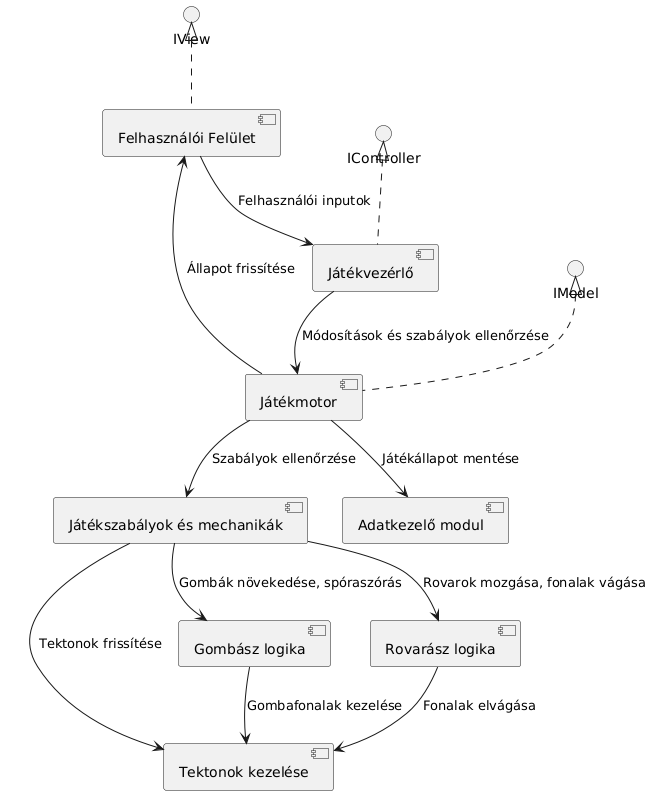
A bevezetés ismerteti a szoftver céljait és, hogy kiknek készül.

A következőkben megismerjük:

* A szoftver legmagasabb szintű architekturális képét.
* Az elvárt funkciókat, a várható felhasználókat, korlátozásokat.
  + Ezekből adódó követelményeket.
    - Az így kialakult követelményekhez kapcsolódó használati eseteket.
* A dokumentumban szereplő kevésbé triviális szavak definíciójait a szótárban.
* Végül bemutatjuk a feladat megoldásának tervezett módját.

## *Áttekintés*

### Általános áttekintés

**

### Funkciók

**Játék inicializálása:**

A program egy játékot valósít meg, amelyben egy vagy több gombász, illetve egy vagy több rovarász vehet részt. A játék kezdetben egy kezdőképernyővel indul, itt kiválaszthatjuk, hogy hány gombász, illetve hány rovarász vesz részt a játékban. Ezután sorban meg kell adni a gombászok, majd a rovarászok nevét, a játék során ebben a sorrendben fognak következni majd a játékosok egymás után.

A program létre kell hozza a Fungorium bolygó felületét, és azon el kell tudja helyezni a tektonokat. Minden tekton egy konvex sokszögként jelenik meg, közöttük pedig egy bizonyos méretű rés van. Mivel a tektonok változatosak lehetnek, a program a tektonok elhelyezése során random dönti el, hogy ez milyen fajta tekton legyen. A játékban lehetnek olyan tektonok, amelyeken a fonalak egy idő után felszívódnak, olyanok, amelyeken több gomba fonaljai is kereszteződni tudnak, olyanok, amelyeken legfeljebb egy fonal tud nőni és olyan tektonok, amiken bár fonal nőhet, de nem tud gombatest fejlődni. A tektonokat úgy különböztetjük meg egymástól, hogy a program fajtájuk szerint különböző színűre színezi ki.

Miután a program létrehozta a tektonokat, először a gombászok sorban kiválasztják, hogy melyik tektonra helyezik el a gombájukat, kezdetben minden gombásznak csak egy gombatestje lehet, majd a rovarászok is kiválasztják azt a tektont, amelyre elhelyezik a rovarukat. Ez lesz a játék 0. köre. A játék során minden körben először a gombászok kezdenek, majd csak azután jönnek a rovarászok, miután minden gombász befejezte a lépését.

**Gombászok funkciói:**

A gombászok a következő funkciókat hajthatják végre egy lépés során: a fonalak növekedési irányát szabályozhatják, a hozzájuk tartozó gombatestek spóraszórását kezdeményezhetik, illetve eldönthetik, hogy egy adott helyen nőjön-e gombatest. Ezek a funkciók a gombászok számára három gombként jelennek meg, melyeket annak függvényében választ ki, hogy milyen sorrendben akarja őket végrehajtani. A gombász akkor fejezte be a lépést, ha a befejezés gombra kattint, ez független attól, hogy tud-e még valamilyen funkciót végrehajtani vagy sem.

Spórákat csak akkor tudnak szórni, ha van egy olyan gombájuk, amelynek gombatestje kilőhető mennyiségű spórát termelt. A gombatestek minden párosadik körben termelnek egy egységnyi spórát, amelyről a gombász dönt, hogy amikor sorra jön kilövi-e egy tektonra, vagy nem. Ha a gombatest nem fejlett, akkor spórákat csak a gombatest tektonjával szomszédos tektonokra lehet lőni, ha azonban fejlett, akkor a gombatest tektonjával szomszédos tektonoknak a szomszédjára is lehet lőni. Egy gombatest akkor fejlett, ha keletkezése óta 5 kör eltelt. Mikor a spórák a tektonra kerülnek, akkor a program random dönti el, hogy ez milyen fajtájú lesz, azaz, ha egy bogár megeszi, akkor milyen hatással lesz ez rá. Lehetnek gyorsító, lassító spórák, olyanok, amelyektől a rovar egy időre megbénul, illetve van olyan spóra, amelytől a rovar egy időre nem tud gombafonalat vágni. Mindegyik spóra a tektonon ugyanúgy jelenik meg, azonban amikor egy rovar megeszi, akkor a rovarásznak látnia kell, hogy milyen spórát evett meg. A gombák összesen 10 egységnyi spórát tudnak kilőni, ez után ez a gombatest elpusztul. Abban az esetben, ha a gombatest tektonján elágazó fonál nincs összeköttetésben egy ugyanolyan fajból származó gombatesttel, akkor a gombafonál is elpusztul. A gombászok úgy szórhatnak spórákat, hogy a képernyőn kiválasztják a spórák szórása gombot, majd kiválasztják a gombatestet, amelynek a spóráit el akarják szórni, és kiválasztják azt a tektont, amelyre a spórákat szórni szeretnék, ekkor ha a lépés helyes, a spórák el lesznek szórva.

A gombászok minden egyes lépés során meg tudják mondani, hogy a gombafajukhoz tartozó gombafonál melyik tektonra nőjjön tovább. Egy gombafonál csak olyan tektonra tud tovább nőni, amely szomszédos egy olyan tektonnal, amelyen már elágazott az adott gombafaj fonala. Abban az esetben, ha a gombász egy olyan tektont jelölt ki, ami egy olyan tektonnal szomszédos, amelyen már elágazott az adott gombafaj fonala, illetve ugyanannak a gombafajnak valamelyik testje spórákat is helyezett el a kijelölt tektonon, akkor miután a gombafonál elágazott a kiválasztott tektonra, akkor a gombász kiválaszthat egy újabb tektont, amely szomszédos az előbb kiválasztottal, és erre a tektonra is tovább fog ágazni a gombafonál. Ez azért történik, mivel a spórák meggyorsítják a gombafonalak növekedését. A gombászok egy lépés során a gombafajhoz tartozó gombafonalat csak egy tektonnal növelhetik tovább, vagy kettővel, ha a spórák felgyorsítják a növekedést. A gombászok úgy tudják megmondani, hogy a gombájukhoz tartozó gombafonál merre nőjjön, hogy a képernyőn kiválasztják a fonál növelése gombot, majd kiválasztják azt a tektont, amelyre szeretnék, hogy a fonál nőjjön.

Ha egy tektonon egy adott gombafajnak elágazott már a fonala, és a gombafajhoz tartozó gombatest spóráiból található már annyi rajta, hogy új gombatestet lehessen növeszteni azaz három egység, akkor a gombász a gomba növesztése funkcióra kattintva egy új gombatestet növeszthet ezen a tektonon.

**Rovarászok funkciói:**

A rovarászok a következő funkciókat hajthatják végre egy lépés során: a rovarok irányát befolyásolhatják és eldönthetik, hogy egy adott fonalat egy rovar elvágjon-e. Akárcsak a gombászoknál náluk is megjelenik ez a két funkció gombként a képernyőn, azonban egy lépés során ők egyszerre csak egyet hajthatnak végre.

A rovarok csak akkor képesek egy résen átlépni egy másik tektonra, ha a két tekton között gombafonal vezet, abban az esetben, ha egy olyan tektonra lépnek, amelyen spóra is található, akkor a tektonon levő összes spórát megeszik, a spóra tápanyagtartalma megegyezik a tektonon levő spóraszámmal. A spórák különböző hatással vannak a rovarokra. Ha a rovar egy gyorsító spórát evett meg, akkor ez után még léphet egyet egy tektonra, ha lassító spórát evett, akkor a következő körben nem léphet, azonban vághat fonalat, ha bénító spórát evett, akkor a következő körben nem csinálhat semmit. Ehet olyan spórát is, amelytől a következő körben nem tud fonalat vágni, de lépni tud. A rovarászok úgy léphetnek rá egy másik tektonra, hogy a képernyőn kiválasztják a lépés gombot, majd kiválasztanak egy olyan tektont, amely szomszédos azzal a tektonnal, amelyen a rovar jelenleg elhelyezkedik, illetve a két tekton közötti résen van fonál.

A rovarászok csak olyan fonalakat tudnak elvágni, amelyek abból a tektonból indulnak ki, vagy haladnak tovább, amelyen a rovarjuk rajta van, azaz azokat a fonalakat, amelyek összekötik azt a tektont, amin a rovar rajta van egy vele szomszédos tektonnal. A vágás úgy valósul meg, hogy a rovarász kiválaszt egy ilyen tektont, ekkor a két tekton közötti résen áthaladó összes fonalat elvágja a rovar.

**Tektonok kettétörése:**

A tektonok mindig két szemközti csúcsnál törnek szét, ekkor a tektonon elágazó fonalak megszakadnak. Minden lépés után nagyon kis valószínűséggel kettétörhet egy random kiválasztott tekton, azonban azok a tektonok, amelyek csak három csúcsból állnak, illetve azok, amelyeken van gombatest nem törhetnek ketté. Abban az esetben, ha a fonál megszakad, és ennek során megszakad a kapcsolata is az őt tápláló gombatestekkel, akkor a fonál elpusztul.

**Játék befejezése:**

A játék egy adott számú kör után fejeződik be, ekkor két oszlopban, az egyikben a gombászok, a másikban a rovarászok jelennek meg, legfelül a nyertessel. A gombászok neve mellett megjelenik, hogy hány gombatestje fejlődött, a rovarászok neve mellett pedig az, hogy mennyi tápanyagot gyűjtött be a megevett spórákból.

### Felhasználók

A program használatára nincsenek különösebb megkötések, életkortól és előképzettségtől függetlenül bárki játszhat vele. A program kezdetekor meg kell adni a játékosok számát, valamint azt, hogy gombászként vagy rovarászként kívánnak szerepelni. A játék elsősorban azok számára ajánlott, akik kedvelik a stratégiai és szerencsén alapuló játékokat.

### Korlátozások

A programot a BME kari felhőjében biztosított, Windows 10 20H2 – JDK-Eclipse-WSU sablonnal ellátott virtuális gépen kell fordítani és futtatni. A játék fordítására és futtatására kizárólag a JDK szükséges, amely biztosítja a java és javac parancsok elérhetőségét. A grafikus felület megvalósításához Java Swing-et használunk.

### Feltételezések, kapcsolatok

-

## *Követelmények*

### Funkcionális követelmények

| ***Azonosító*** | ***Leírás*** | ***Ellenőrzés*** | ***Prioritás*** | ***Forrás*** | ***Use-case*** | ***Komment*** |
| --- | --- | --- | --- | --- | --- | --- |
| *TEKT1* | *A tektonok konvex sokszögek.* | *Sokszögekként jelennek meg a grafikus felületen.* | *MUST* | *five\_guys csapat* | *-* | *-* |
| *TEKT2* | *A tektonok ketté tudnak törni egy meghatározott valószínűséggel.* | *A grafikus felületen ez megjelenik.* | *SHOULD* | *five\_guys csapat* | *-* | *Amint egy tekton háromszöggé válik egy törés után, akkor az a tekton már nem képes tovább törni. Továbbá azok a tektonok sem tudnak tovább törni, amelyeken van gombatest.* |
| *TEKT3* | *Egy tektonon csak egy gombatest lehet.* | *Két gombatest nem jelenhet meg grafikusan egy tektonon.* | *MUST* | *five\_guys csapat* | *Gombatest növesztése* | *-* |
| *TEKT4* | *A különböző fajtájú tektonok más-más színnel jelennek meg.* | *Grafikusan látható.* | *MAY* | *five\_guys csapat* | *-* | *-* |
| *TEKT5* | *Egy tektonra nem lehet annál több spórát lőni, mint amennyi egy új gombatest növesztéséhez szükséges.* | *Grafikusan látható.* | *SHOULD* | *five\_guys csapat* | *Gombatest spórázás* | *-* |
| *GOMB1* | *Minden gombásznak a 0. körben le kell raknia egy gombatestet egy általa kiválasztott tektonra.* | *A kiválasztott tektonon megjelenik a játékos gombatestje a játékos színében.* | *MUST* | *five\_guys csapat* | *Gombatest növesztése* | *Egy tektonon csak egy gomatest lehet.* |
| *ROV1* | *Egy rovarásznak csak egy rovara lehet az egész játék során* | *Nincs két ugyanolyan színű rovar a grafikus felületen.* | *MUST* | *five\_guys csapat* | *Rovar kelesztése* | *-* |
| *ROV2* | *Minden rovarásznak a 0. körben le kell raknia egy rovart egy általa kiválasztott tektonra.* | *A kiválasztott tektonon megjelenik a játékos rovara a játékos színében.* | *MUST* | *five\_guys csapat* | *Rovar kelesztése* | *Egy rovarásznak csak egy rovara lehet az egész játék során* |
| *SPÓRA1* | *Minden párosadik körben kap 1 egység spórát gombatestenként egy gombász.* | *A grafikus felületen egy gombatestre kattintva látszani fog a gombatest által kilőhető spórák száma.* | *SHOULD* | *five\_guys csapat* | *Gombatest spórázás* | *A 2. körtől kezdődően kapnak a gombászok spórákat a gombatestjeik függvényében. (0. kör nem számít körnek)* |
| *SPÓRA2* | *Többfajta spóra létezik, amely más-más hatással vannak az őket elfogyasztó rovarokra.* | *-* | *SHOULD* | *five\_guys csapat* | *Gombatest spórázás, Spóra elfogyasztása* | *-* |
| *GOMBTEST1* | *5 kör után fejlődik egy gombatest kifejlett gombatestté.* | *Grafikusan látható. (Messzebbre tud spórát lőni)* | *SHOULD* | *five\_guys csapat* | *Gombatest fejlődése* | *Miután egy gombatest kifejletté válik, akkor a szomszédos tektonjának a szomszédjára is tud spórát lövelni.* |
| *GOMBTEST2* | *Ha egy tektonon összegyűlik 3 egység spóra egy adott gombásznak, akkor azon a tektonon lehet gombatestet növeszteni.* | *Grafikusan látható.* | *SHOULD* | *five\_guys csapat* | *Gombatest növesztése* | *Nem muszáj a gombásznak ezen a tektonon mindenképpen gombát növesztenie.* |
| *GOMBTEST3* | *10 spóra lövés után elpusztul egy gombatest.* | *Grafikusan látható.* | *SHOULD* | *five\_guys csapat* | *Gombatest elpusztulása* | *Ha egy gombatest elpusztul, akkor a hozzá tartozó összes fonál is elpusztul (a spóra nem).* |
| *GOMB2* | *Minden gombász a saját körében egy adott akciót csak egyszer játszhat ki, de nem muszáj kijátszania az összes akcióját.*  *Akciók:*  *- fonalak növekedésének irányítása*  *- gombatest spóra szórása*  *- gombatest növesztése* | *Gombokkal jelennek meg grafikusan a kiválasztható akciók.* | *MUST* | *five\_guys csapat* | *-* | *Ha egy gombász nem szeretné kijátszani az összes akcióját, akkor a „jöhet a következő” gombbal tudja jelezni, hogy végzett a körével.* |
| *ROV3* | *A rovarászok a két akciójuk közül csak az egyiket játszhatják ki a körükben.*  *Akciók:*  *- fonal vágása*  *- fonal mentén rovar léptetése* | *Gombokkal jelennek meg grafikusan a kiválasztható akciók.* | *MUST* | *five\_guys csapat* | *-* | *Ha egy rovarász nem szeretné kijátszani az akcióját, akkor a „jöhet a következő” gombbal tudja jelezni, hogy végzett a körével.* |
| *GOMBFON1* | *Ha a gombász olyan tekton felé irányítja a gombafonalat, amin van saját színű spórája, akkor mégegyszer tehet az adott irányba fonalat.* | *-* | *MAY* | *five\_guys csapat* | *Fonál növekedésének irányítása* | *-* |
| *ROV4* | *Ha a rovar olyan tektonra lép, ahol található spóra, akkor az adott tektonon az összes spórát megeszi a rovar.* | *Grafikusan látható.* | *SHOULD* | *five\_guys csapat* | *Spóra elfogyasztása* | *-* |

### Erőforrásokkal kapcsolatos követelmények

| ***Azonosító*** | ***Leírás*** | ***Ellenőrzés*** | ***Prioritás*** | ***Forrás*** | ***Komment*** |
| --- | --- | --- | --- | --- | --- |
| *SWING* | *A játék grafikus megjelenítésére Java Swing könyvtárt használjuk.* | *-* | *SHOULD* | *five\_guys csapat* | *-* |
| *FUT* | *A forrásprogramnak a kari felhőben biztosított környezet alatt kell futnia.* | *-* | *MUST* | *Tárgy általi feladatleírás* | *-* |

### Átadással kapcsolatos követelmények

| ***Azonosító*** | ***Leírás*** | ***Ellenőrzés*** | ***Prioritás*** | ***Forrás*** | ***Komment*** |
| --- | --- | --- | --- | --- | --- |
| *BEADÁS* | *A hercules beadandó rendszerbe feltöltendő anyagot egyetlen zip fájlba kell becsomagolni.* | *-* | *MUST* | *Tárgy általi feladatleírás* | *-* |

### Egyéb nem funkcionális követelmények

| ***Azonosító*** | ***Leírás*** | ***Ellenőrzés*** | ***Prioritás*** | ***Forrás*** | ***Komment*** |
| --- | --- | --- | --- | --- | --- |
| *-* | *-* | *-* | *-* | *-* | *-* |

## *Lényeges use-case-ek*

### Use-case leírások

| **Use-case neve** | Gombatest növesztés |
| --- | --- |
| **Rövid leírás** | Gombász választhatja, hogy adott helyen nőjön gombatest. |
| **Aktorok** | Gombász |
| **Forgatókönyv** | 1. Gombász kiválasztja a gombatest növesztése funkciót 2. Gombász kiválasztja a tektont, ahol testet akar növeszteni 3. A játék ellenőrzi, hogy van-e elég spóra az adott tektonon, van-e már ott fonal és nőhet-e itt gombatest 4. Megfelel a tekton 5. Kinő a gombatest   Alternatív:   1. Gombász kiválasztja a gombatest növesztése funkciót 2. Gombász kiválasztja a tektont, ahol testet akar növeszteni 3. A játék ellenőrzi, hogy van-e elég spóra az adott tektonon, van-e már ott fonal és nőhet-e itt gombatest 4. A tekton nem felel meg 5. A lépés során nem történik változás |

| **Use-case neve** | Gombatest spórázás |
| --- | --- |
| **Rövid leírás** | Gombász választhatja, hogy spórázzanak a gombatestjei |
| **Aktorok** | Gombász |
| **Forgatókönyv** | 1. Gombász kiválasztja a spórázás funkciót 2. Gombász kiválasztja a gombatestet, amit spóráztatni akar 3. A játék ellenőrzi, hogy van-e elég spórája a gombatestnek 4. Gombász kiválasztja a tektont amire spórázni akar 5. A játék ellenőrzi, hogy lehet e arra a tektonra spórázni 6. Ha minden megfelel, kilövi a spórát a tektonra   Alternatív   1. Gombász kiválasztja a spórázás funkciót 2. Gombász kiválasztja a gombatestet, amit spóráztatni akar 3. Nincs elég spórája, vagy nem jó tektont választ 4. A lépés során nem történik változás |

| **Use-case neve** | Gombatest elpusztulása |
| --- | --- |
| **Rövid leírás** | Gombatest elpusztul 10 spórázás után |
| **Aktorok** | Gombász |
| **Forgatókönyv** | 1. Gombász sikeresen spóráztatja a gombatestét 2. A játék ellenőrzi, hogy az adott gombatest spórázott-e már 10-szer 3. Ha igen, a gombatest rövid időn belül elpusztul, ha nem akkor tovább él |

| **Use-case neve** | Fonál növekedésének irányítása |
| --- | --- |
| **Rövid leírás** | Gombász eldönti, hogy merre nőjön tovább a gombafonala |
| **Aktorok** | Gombász |
| **Forgatókönyv** | 1. Gombász kiválasztja a fonal irányítása funkciót 2. Gombász kiválaszt egy tektont, amerre irányítani akarja 3. A játék ellenőrzi, hogy nőhet-e ide gombafonal 4. Ha igen, akkor kinő a gombafonal 5. A játék ellenőrzi, hogy volt-e már itt gombafonala és spórája az adott gombafajnak 6. Ha volt, akkor Gombász növeszthet gombafonalat még egyszer |

| **Use-case neve** | Rovar kelesztés |
| --- | --- |
| **Rövid leírás** | A játék elején amikor a rovarászok jönnek, mindegyik kiválaszthatja, hogy hova szeretnék letenni a rovarukat. |
| **Aktorok** | Rovarász |
| **Forgatókönyv** | 1. A játék elején, amikor először jönnek a rovarászok, ki kell választaniuk egy tektont, ahova leteszik a rovarukat 2. A játék ellenőrzi, hogy a tekton nem foglalt-e már 3. Ha nem, akkor megjelenik rajta egy rovar az adott játékos színeiben |

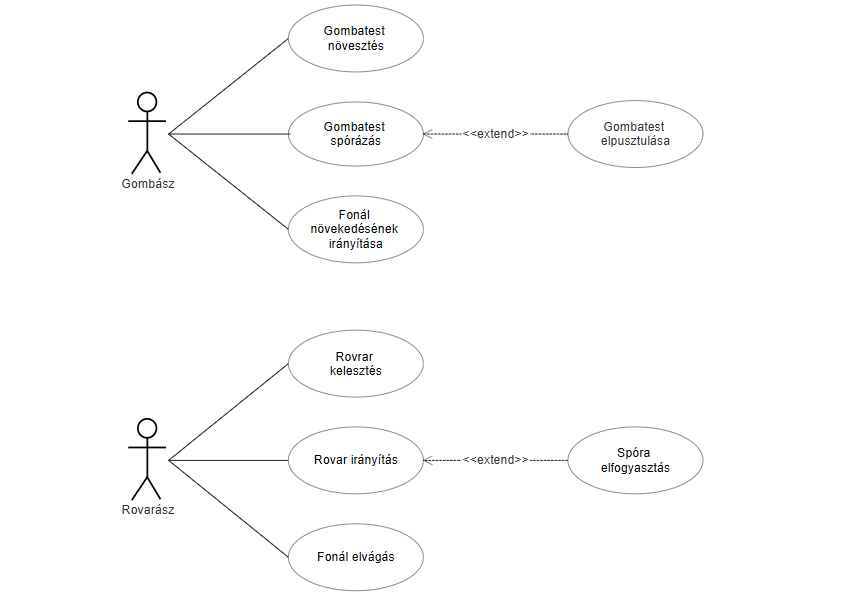
| **Use-case neve** | Rovar irányítás |
| --- | --- |
| **Rövid leírás** | Rovarász választhat két lépés közül, hogy irányítja a rovarát |
| **Aktorok** | Rovarász |
| **Forgatókönyv** | 1. Rovarász kiválasztja a rovar irányítása funkciót 2. Rovarász kiválasztja, hogy melyik tektonra akarja irányítani a rovarát 3. A játék ellenőrzi, hogy van-e gombafonal a két tekton között és szomszédos-e 4. Ha megfelel, akkor a rovar átmegy a választott tektonra   Alternatív:   1. Rovarász kiválasztja a rovar irányítása funkciót 2. Rovarász kiválasztja, hogy melyik tektonra akarja irányítani a rovarát 3. A játék ellenőrzi, hogy van-e gombafonal a két tekton között és szomszédos-e 4. A tekton nem felel meg, nem történik változás, a Rovarásznak más lépést kell választania |

| **Use-case neve** | Spóra elfogyasztás |
| --- | --- |
| **Rövid leírás** | Ha adott tektonra mozog egy rovar, akkor az ott lévő spórákat elfogyasztja |
| **Aktorok** | Rovarász |
| **Forgatókönyv** | 1. A Rovarász sikeresen használja a Rovar irányítás funkciót 2. A játék ellenőrzi, hogy a tektonon, amire lépett a rovar van-e spóra 3. Ha igen, akkor ezeket a rovar automatikusan elfogyasztja 4. Életbe lép a rovaron a spóra hatása |

| **Use-case neve** | Fonál elvágása |
| --- | --- |
| **Rövid leírás** | Rovarász a két lépés közül a fonál elvágását választja |
| **Aktorok** | Rovarász |
| **Forgatókönyv** | 1. Rovarász kiválasztja a fonál elvágása funkciót 2. Rovarász kiválasztja a tekton egyik oldalát, amelyiken a rovara van 3. Azon az oldalon futó gombafonalakat mind elvágja, ezzel a két tekton tovább nem lesz összekötve gombafonallal |

### 

### Use-case diagram



## *Szótár*

| **Fogalom** | **Definíció.** |
| --- | --- |
| **Architekturális kép** | A szoftver legfelsőbb szintű szerkezeti terve, amely bemutatja a főbb részeket és azok kapcsolatait. |
| **Extend kapcsolat** | Kiegészít egy use-case-t, nem mindig fut le. |
| **Fejlett gombatest** | Egy olyan gombatest, amely elérte a szükséges növekedési szintet és további funkciókkal bír. |
| **Gombafonal** | Egy gombák által növesztett kapcsolat. |
| **Gombász** | A játék egyik szereplőtípusa, aki a gombák növesztésével foglalkozik. |
| **Gombatest** | A gombafonalak innen indulnak, a spórák innen bocsátódnak ki. |
| **Interfész** | Két rendszer, modul vagy felhasználó és program közötti kommunikációs felület. |
| **Játék befejezése** | Az a folyamat, amely során a játék véget ér és a győztesek meghatározásra kerülnek. |
| **Java** | A programozási nyelv, amin a programot írjuk. |
| **Java Swing** | Egy Java alapú grafikus felhasználói felület létrehozására szolgáló keretrendszer. |
| **MAY** | Opcionális követelmény, amely jó, ha megvalósul, de nem szükséges. |
| **MUST** | Kötelező követelmény, enélkül a rendszer nem működik megfelelően. |
| **Prioritás (MUST, SHOULD, MAY)** | A követelmények fontossági besorolása az RFC 2119 szabvány szerint: |
| **Random** | Véletlenszerűen generált. |
| **Rovarász** | A játék másik szereplőtípusa, aki a rovarokat irányítja. |
| **SHOULD** | Fontos, de nem feltétlenül kötelező követelmény. |
| **Spóra** | Rovarok számára tápanyag. Gombászok számára építőanyag gombatest és gombafonal építésére. |
| **Spóraszórás** | A gombák szaporodásának módja, amelynek hatása van a játékmenetre. |
| **Tápanyagtartalom** | A játékban a rovarok által elfogyasztott spórák értéke. |
| **Tekton** | A játékban használt területegység. |
| **Use-case (használati eset)** | A felhasználó és a rendszer közötti interakciót reprezentálja a felhasználó szemszögéből. Segítenek megérteni a funkcionális követelményeket. |
| **Virtuális gép** | Egy számítógépes környezet, amelyen le kell futnia a játéknak. |

## *Projekt terv*

**Szereposztás:**

Egységesek a szerepek a csapatban, tehát mindenki dokumentál és kódol is. Nincs olyan csapattag, aki kifejezetten csak kódol, vagy dokumentál. Mindenkinek minden héten ki lesz osztva az adott heti feladata.

**Végrehajtási lépések, eredmények határideje:**

A tárgynak az ütemezése szerint hajtja végre a csapat a feladatot. Minden héten szerdáig a csapattagok átnézik a heti feladatot, a szerdai konzultáción, ha valami nem egyértelmű, akkor megbeszéli a csapat a pontosítást a konzulenssel. Ezek után szerdán egy közös meetingen kiosztja a csapat az adott heti feladatot a csapattagoknak megegyezés alapján. Vasárnap szintén tart a csapat egy összegző meetinget a heti feladatról, eddigre minden csapattársnak el kell végeznie a saját részfeladatát. Egy csapattag elakadása esetén rendkívüli meeting alkalmakat tervezünk segítségnyújtás céljából.

Az egyes feladatok elvégzéséért felelős személyek nevét és beosztását a fentiek szerint minden héten a szerdai meetingen határozzuk meg a heti feladat függvényében.

**Szükséges erőforrások, csoportmunkát támogató eszközök, választott technikák**

A meetingeket és konzultációkat a csapattagok egymás között Discordon és személyesen tartják. A szöveges dokumentumokat Google Drive-on keresztül osztjuk meg egymással. Az elkészített kódokat a Git és GitHub verziókezelő rendszerrel integráljuk és osztjuk meg egymás között. Mainline integration technikát használunk a verziókezelés során. Diagramok elkészítéséhez a PlantUML alkalmazást fogja használni a csapat. Továbbá a szoftver fejlesztése során a model-view-controller tervezési mintát tervezzük használni.

## *Napló*

| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| --- | --- | --- | --- |
| 2025.02.19. 14:00 | 2 óra | Fejes  Szolár  Zelch | Értekezlet. Feladat értelmezése.  Döntés: Zelch csinálja a  2.2.2.-es fejezetet 23-ára. |
| 2025.02.19. 16:00 | 1 óra | Fejes  Szolár  Zelch  Toronyi  Lekli | Értekezlet.  Toronyi és Lekli tájékoztatása, az előző meetingen történtekről. További ötletelések.  Döntések:  Szolár: a fedlapot és az 2.1. fejezetet,  Toronyi: a 2.2. fejezetet, kivéve a 2.2.2.-t (ezt Zelch csinálja az előző meeting alapján)  Fejes: a 2.3. fejezetet  és  Lekli: a 2.4. fejezetet fogja  elkészíteni 23-ára.  Az utolsó kettő téma késöbbi értekezletek tárgya. |
| 2025.02.19. 18:00 | 45 perc | Szolár | Fedlap elkészítése |
| 2025.02.20-23:00 | 2 óra | Szolár | Bevezetés és Szótár elkészítése. |
| 2025.02.21 | 3,5 óra | Zelch | Funkciók leírása |
| 2025.02.21. 10:00 | 3,5 óra | Fejes | Követelmények kidolgozása, és dokumentálása, 2.3-as fejezet megírása. |
| 2025.02.22. 11:30 | 1,5 óra | Fejes | Követelmények pontosítása, 2.3-as fejezet finomítása. |
| 2025.02.22. | 3 óra | Lekli | Use-case diagram készítése, forgatókönyvek megírása |
| 2025.02.22. | 3 óra | Toronyi | Általános áttekintés, illetve a felhasználókra és korlátozásokra vonatkozó leírás elkészítése, komponens diagram megtervezése és megvalósítása. |
| 2025.02.23 | 1 óra | Fejes  Lekli  Szolár  Toronyi  Zelch | Értekezlet, megbeszéltük, hogy ki mit csinált a hét során, végignéztük ezeket, ha volt valami hiányosság kiegészítettük, és együtt leraktuk a projekt terv alapjait.  Döntés: A megoldás beadható. |

**4. Analízis modell II.**

26 – five\_guys

Konzulens:

Huszerl Gábor

Csapattagok

| Lekli Zsolt | JENKHQ | zsoltlekli@gmail.com |
| --- | --- | --- |
| Fejes Ábel | AF1Y1C | fejesabel@gmail.com |
| Zelch Csaba | LK0617 | zch03po@gmail.com |
| Szolár Soma | E05SC6 | somaszolar1@gmail.com |
| Toronyi Zsombor | S8F7DV | toronyizsombi@gmail.com |

2025. március 10.

# Analízis modell kidolgozása

## *Objektum katalógus*

* + 1. ***Rovar***

Minden rovarásznak van egy rovara. A rovar a különböző tekonokra léphet úgy, hogy áthalad a tekonok közötti résen található gombafonalakon. Ezenkívűl el tudja vágni azokat a gombafonalakat, amelyek abból a tektonból indulnak ki, vagy haladnak tovább, amelyen a rovar rajta van.

* + 1. ***Gombafonál***

Minden gombásznak van egy gombafonala, amely különböző tektonokon ágazhat el. A gombafonál megmutatja, hogy melyik tekonokon van elágazva, illetve feladata, hogy azokat a részeit eltávolítsa, amelyek már nincsenek összeköttetésben egy ugyanolyan fajból származó gombatesttel. Ezenkívül szerepe van a fonál elágaztatásában egy másik tektonra.

* + 1. ***Gombatest***

Minden gombásznak vannak gombatestjei, ezek kezdetben fejletlen állapotban vannak. Ezek a gombatestek tudnak spórákat szórni, azonban csak is szomszédos tektonra. 5 kör elteltével a gombatest fejlett állapotba kerül. Ezek a gombatestek már a szomszédos tektonok szomszédjaira is tudnak spórát szórni, tehát eggyel nagyobb távolságra mint a fejletlen párjuk. 10 db spóraszórás után, beleértve fejlődése előtti szórásokat is, a gombatest elpusztul.

* + 1. ***Tektonok***

A bolygó felületét lefedő sokszögek. A rovarászok rovarjaikat egyikről a másikra tudják mozgatni. A gombászok gombatesteket és gombafonalakat tudnak rájuk tenni, illetve a testekből spórát tudnak lövetni. Mindezt a már ismertetett szabályok alapján.

Mindegyik körben ketté törik az egyik. Csak az tud ketté törni amelyiken nincsen gombatest és a szögeinek a száma nagyobb mint három.

* + 1. ***SzívóTekton***

Egy speciális tekton. Minden 5. körben a rajta levő összes fonál felszívódik. Ez indikálhatja azt, hogy több fonálrész is eltűnik, ammenyiben azok ehhez a gombafajhoz tartoznak és nem kötődnek a fajukhoz tartozó testhez.

SzívóTektonra nem lehet tenni gombatestet.

* + 1. ***EgyfonalasTekton***

Egy olyan különleges tekton, amelyen csak egy fajta fonál hajthat át, vagy egy gombatest.

* + 1. ***TöbbfonalasTekton***

Olyan tekton, amelyen akár töbfajta fajta fonál is lehet, és akár egy gombatest is.

* + 1. ***Spóra***

A játék nagyon fontos eleme. A gombatest termeli ezeket az objektumokat. A játék során minden második körben termelődik egy egység belőlük minden gombatestben. A gombászok a gombatestjük fejlettségi szintjétől függően lőhetik ki őket tektonokra. A fejletlen gombatestből csak a szomszédos tektonokra lehet lőni őket, a fejlettből viszont a szomszédos tekton szomszédjára is lehet lőni. Egy tektonon maximum 5 egység spóra gyűlhet össze. Ha egy tektonon 5 egység összegyűlt, akkor arra a tektonra a gombász tud gombatestet növeszteni (feltéve persze ha lehet arra a tekton fajtára). Továbbá a rovarok táplálékaként is szolgálnak ezek az objektumok. Többféle spóra fajta létezik, ezek mind más hatással vannak az őket elfogyasztó rovarokra. Ezeket a spóra fajtákat a következő osztályok leírásában fejti ki a dokumentáció.

* + - 1. ***Gyorsítóspóra***

Olyan spóra, amely elfogyasztásával a rovar mégegyszer jöhet.

* + 1. ***Vágástalanítóspóra (NoCutspóra)***

Olyan spóra, amely elfogyasztásával a rovar nem tud fonalat vágni a körében.

* + - 1. ***Lassítóspóra***

Olyan spóra, amely elfogyasztásával a rovar a következő körében nem tud lépni, de vágni viszont tud.

* + 1. ***BénítóSpóra***

Olyan spóra, amely elfogyasztásával a rovar a következő körében se lépni, se vágni nem tud, tehát kimarad egy körből.

## *Statikus struktúra diagramok*

## *Osztályok leírása*

A template-től eltérve nem ABC sorrendben, hanem logikai sorrendben írjuk le az osztályokat. Szerintünk így sokkal érthetőbb.

### Insect(rovar)

#### Felelősség

Az Insect osztály felel a rovar lépésének megvalósításáért, vagyis azért, hogy egy másik tektonra lépjen a tektonok között található résen elhelyezkedő gombafonalakon át. Ezenkívül a fonalak elvágásáért is felelős, elvághat olyan gombafonalakat, amelyek a saját tektonja, azaz amelyiken elhelyezkedik és a vele szomszédos tektonok közötti réseken haladnak át. Az Insect osztálynak még meg kell tudnia mutatni, hogy melyik tektonon helyezkedik el, illetve tárolnia kell az állapotát.

#### Ősosztályok

Nincs ősosztálya.

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

* **asszociáció(is on)**: A túloldali szereplő a Tecton, amelyen maximum egy rovar helyezkedhet el, egy rovar pedig biztosan egy tektonon helyezkedik el.

#### Attribútumok

* **position** : Az a tekton, amelyen a rovar elhelyezkedik
* **state** : Ez egy enum típus, amely tartalmazza a rovar állapotát, ez átlagos esetben normal, de változik, miután a rovar megeszik egy spórát

#### Metódusok

* **boolean move(Tecton t)** : Ez a metódus felel a rovar lépéséért egy másik tektonra. Paraméterként egy tektont kap, ez lesz a cél, ahova a rovar lépni szeretne. A metódus meghívja a tekton putInsect(i: Insect, t: Tecton) metódusát, amely leellenőrzi, hogyha a rovar odaléphet-e arra a tektonra, majd ha igen, akkor végrehajtja a lépést, és true értékkel tér vissza, ezáltal frissül a rovar pozíciója, és sikeres a lépés
* **boolean cut(Tecton t)** : Ez a metódus felel a fonalak elvágásáért. Paraméterként egy tektont kap, és először leellenőrzi, hogy végre tudja-e hajtani a vágást, azaz a kapott tekton és az a tekton amin elhelyezkedik szomszédosak, illetve a közöttük levő résen van-e gombafonál. Ha a fonalvágás végrehajtható, akkor elvágja a fonalakat, és true értékkel tér vissza, ellenkező esetben pedig false értékkel tér vissza.

### FungalThread(gombafonál)

#### Felelősség

A FungalThread osztály felelős azoknak a tektonoknak a tárolására, amelyeken a gombafonál elágazik, azaz meg kell mutassa, hogy mely tektonokon van elágazva. Ezenkívűl el kell távolítsa a fonál azon részeit, melyek már nincsenek összeköttetésben egy ugyanolyan fajból származó gombatesttel, például, ha egy gombafonál valahol elszakad, vagy meghal egy gobmatest, gondoskodnia kell ezen fonálrészek eltávolításáról. A GombaFonál felel a fonál növesztéséért, azaz biztosítania kell, hogy a kijelölt tektonon elágazzon a fonál, ha ez lehetséges. Ezenkívül a fonál felel a gombatest növesztéséért is

#### Ősosztályok

Nincs ősosztálya

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

* **asszociáció1(branches)**: a túloldali szereplő a SingleThreadTecton, amelyen pontosan egy fonál lehet rajta, azonban egy fonál több ilyen fajta tektonon elágazhat
* **asszociáció2(branches)**: a túloldali szereplő a MultiThreadTecton, amelyen egyszerre több fonál is lehet, és egy fonál is több ilyen fajta tektonon elágazhat
* **asszociáció3(branches)**: a túloldali szereplő a AbsorbingTecton, amelyen egyszerre több fonál is lehet, és egy fonál is több ilyen fajta tektonon elágazhat
* **aggregáció(belongs)**: a túloldali szereplő a Mushroom, minden gombatesthez egy gombafonál tartozik, de egy gombafonálhoz tartozhat több gombatest
* **asszociáció4(use)**: a túloldali szereplő a spóra, ezek közt asszociáció van, mivel gombatest növesztésekor kell tudja a gombafonál azokat a spórákat, amelyekkel növeszteni tud gombatestet

#### Attribútumok

* **tectons :** Ez egy tektonokból álló lista lesz, segítségével eltároljuk azokat a tektonokat, amelyeken a gombafonál elágazik

#### Metódusok

* **void deleteUnnecessaryThreads()** : A metódus meghívása során leellenőrzi, hogy a fonálnak van-e olyan része, amely már nincsenek összeköttetésben egy ugyanolyan fajból származó gombatesttel. Ha talál ilyet, akkor ezt kitörli.
* **boolean branchThread(Tecton t)** : A metódus a fonál elágazásáért felel. Paraméterként egy tektont kap, amely a cél tekton lesz, vagyis amelyre a fonalat el akarjuk ágaztatni. Ekkor leellenőrzi, hogy a fonál tovább tud-e ágazni arra a tektonra. Ha igen, akkor kapott tektonon is elágaztatja a fonalat, és true értékkel tér vissza, ellenkező esetben false értékkel fog visszatérni.
* **boolean removeTecton(Tecton t) :** A metódus arról felel, hogy a paraméterként kapott tektonnal megegyező tektont kivegye a tectons listából
* **boolean addTecton(Tecton t) :** A paraméterként kapott Tectont hozzáveszi a tectons listához
* **boolean growMushroom(Tecton t)**: A metódus azért felel, hogy a paraméterként kapott tektonra gombatestet növesszen, ehhez meg kell néznie, hogy a kapott tektonon elágazik-e fonál, illetve van-e elég spóra

### Mushroom(gombatest)

#### Felelősség

A gombatest osztály, felelős a gombatest pozíciójának eltárolásáért, illetve a spóráinak listájának eltárolásáért is. Ezenkívül eltárolja a hozzá tartozó gombafonalat, a gombatest állapotát, és azt, hogy eddig hány spórát lőtt ki.

* + - * **Ősosztályok**

Nincs ősosztálya

#### Interfészek

*Nem valósít meg interface-t*

#### Asszociációk

* **asszociáció1:** Túloldali szereplő a SingleThreadTecton, ezen a tektonon nőhet gombatest
* **asszociáció2:** Túloldali szereplő a MultiThreadTecton, ezen a tektonon nőhet gombatest
* **asszociáció3:** Túloldali szereplő a spóra, a Gombatest termeli a spórákat
* **aggregáció:** Túloldali szereplő a gombafonál, minden gombatesthez egy gombafonál tartozik, egy gombafonálhoz tartozhat több gombatest

#### Attribútumok

* **position -** Az a tekton, amin a gombatest található
* **spores-** A testhez tartozó spórák listája
* **thread -** A testhez tartozó gombafonál
* **state -** A gombatest állapota
* **shootedSporesCount -** eddig hány spórát lőtt ki

#### Metódusok

* **boolean shootSpore(Tekton t) -** A metódus a paraméterben megadott tektonra kilő egy spórát
* **evolve() -** a gombatest a metódus során fejletté válik
* **generateSpore(SporeEffect) -** a paraméterként kapott spóra fajta alapján egy vele megegyező fajtájú spórát hoz létre, amit ki tud majd lőni

### Tecton(tekton)

#### Felelősség

Az absztrakt Tecton osztályból származnak le a példányosítható tekton osztályok. A put-, removeThread(t : Thread ), a put-,removeMushroom(m : Mushroom), a putFirstMushroom() és a break() függvények implementálását a leszármazottakra hagyja.Az osztálydiagramon látható többi függvényt viszont megvalósítja, ez egy közös funkcionalitás.

#### Ősosztályok

Nincs ősosztálya

#### Interfészek

Nem valósít meg interface-t.

#### Asszociációk

* **asszociáció (is on)**: Másik oldal: Spore. A Tecton Spore-okat tartalmaz. Egy Tecton tartalmazhat több Spore-t, egy Spore csak egy Tecton-hoz tartozhat. Ezen kapcsolat mentén tartja nyilván a rajta lévő spórákat, amiket kilőttek rá.
* **asszociáció (is on):** Másik oldal: Insect. A tekton tárolja a rajta lévő rovart, továbbá a rovar is tárolja, hogy melyik tektonon van éppen. Egy rovar egyszerre csak egy tektonon lehet. Továbbá egy tektonon csak egy rovar lehet.
* **asszociáció (neighbor):** Másik oldal: Tecton (önmaga). A tekton tárolja a szomszédos tektonjait. Egy tektonnak több szomszédja is lehet.

#### Attribútumok

* **spores : List<Spore>**. A tektonon lévő (tektonra kilőtt) spórák.
* **neighbors : List<Tecton>**. A szomszédos Tecton-ok.
* **i : Insect**: Éppen a tektonon lévő rovar.

#### Metódusok

* **putThread(f : FungalThread) : boolean**. Absztrakt metódus.
* **putMushroom(m : Mushroom) : boolean**. Absztrakt metódus.
* **removeThread(f : FungalThread) : boolean**. Absztrakt metódus.
* **removeMushroom() : boolean**. Absztrakt metódus.
* **getThreadSection(f : FungalThread) : ArrayList<Tecton>**. Visszatér egy ArrayList-el, amiben olyan szomszédos tektonok vannak, amiken rajta van a paraméterként kapott fonál egy része.
* **putSpore(sp: Spore, t: Tecton) : boolean.** Az átvett sporát hozzávesszi spores listájába a tektonnak. A paraméterül kapott tekton az a tekton, ahonnan lőtték a spórát.
* **isNeighbor(t : Tecton) : boolean**. Átvesz egy tektont és megnézi, hogy benne van-e neighbor listájában.
* **break(): boolean**: boolean: Tekton kettétörését valósítja meg(absztrakt metódus)
* **putFirstMushroom(): boolean**: Játékos első gombatestjének lehelyezése(absztrakt metódus)
* **putFirstInsect(): boolean**: Jatékos első rovarának lehelyezése
* **putInsect(i: Insect, t: Tecton): boolean** :rovar lehelyezése
* **removeInsect() boolean**: rovar eltávolítása tektonról
* **addNeighbor(tlist: List<Tecton>) : boolean** : sszomszédsági lista bővítése a kapott tektonokkal
* **putEvolvedSpore(sp:Spore, t: Tecton): boolean** : az átvett spórát hozzáveszi a spores listájához
* **removeSpores(slist: List<Spore>): boolean** : a kapott spórákat kitörli a spores listájából

### AbsorbingTecton(*SzívóTekton*)

#### Felelősség

Ő felel a Tecton példányosítható, szívó változatának a reprezentálásáért.

#### Ősosztályok

Tecton -> AbsorbingTecton

#### Interfészek

Nem valósít meg interface-t.

#### Asszociációk

* **(branches) asszociáció**: Másik oldal:FungalThread. Az AbsorbingTecton rendelkezik egy listával, amelyben tárolja FungalThread példányokat, és végrehajtja a módosításokat. Egy FungalThread létezhet AbsorbingTecton nélkül is, amennyiben az egy másfajta tektonon van.

#### Attribútumok

* **threads : List<FungalThread>**. Az asszociációnak megfelelően, lehet rajta akár több fonál is.

#### Metódusok

* **putThread(f:FungalThread) : boolean**. Megpróbál felvenni egy fonalat a listájába, a művelet sikerességéval tér vissza.
* **putMushroom(m : Mushroom) : boolean**. Mindig hamissal tér vissza mert a szívó tektonra nem lehet tenni gombatestet.
* **removelThread(f : FungalThread) : boolean.** A paraméterként megkapott fonalat kitörli a listájából és visszatér a művelet sikerességével.
* **removeMushroom() : boolean**. Mindig false-al tér vissza mert a szívó tektonra nem lehet tenni gombatestet.
* **break(): boolean**: boolean: Tekton kettétörését valósítja meg
* **putFirstMushroom(): boolean**: Játékos első gombatestjének lehelyezése
* **absorb(): void** : leveszi a tektonról a rajta elágazó fonalakat

### SingleThreadTecton(*EgyfonalasTekton*)

#### Felelősség

Ő felel a Tecton példányosítható, egyfonalas változatának a reprezentálásáért.

#### Ősosztályok

Tecton -> SingleThreadTecton

#### Interfészek

Nem valósít meg interface-t.

#### Asszociációk

* **asszociáció(growing)**: Másik oldal: Mushroom. Van egy Mushroom attribútuma.
* **asszociáció(branches)**: Másik oldal:FungalThread. Van egy FungalThread attribútuma.

#### Attribútumok

* **thread : FungalThread**. A asszociáció-nak megfelelően, max 1 fonala lehet.
* **mushroom : Mushroom**. A aggregáció-nak megfelelően, maxiumum egy gomba lehet rajta.

#### Metódusok

* **putThread(f:FungalThread) : boolean**. Csak akkor vesz fel fonalat és tér vissza igazzal, ha még nincsen rajta és valid a lépés, különben hamissal tér vissza.
* **putMushroom(m : Mushroom) : boolean**. Megpróbál felvenni egy gombát, igazzal tér vissza, ha van legalább 5 spóra a tektonon és a még van lépése a gombásznak.
* **removeThread(f : FungalThread) : boolean.** Megpróbálja kitörölni a megadott fonalat magáról. A művelet sikerességével tér vissza.
* **removeMushroom() : boolean**. Megpróbálja kitörölni a gombát magáról. A művelet sikerességével tér vissza.
* **break(): boolean**: boolean: Tekton kettétörését valósítja meg
* **putFirstMushroom(): boolean**: Játékos első gombatestjének lehelyezése

### MultiThreadTecton(*TöbbfonalasTekton*)

#### Felelősség

Ő felel a Tecton példányosítható, többfonalas változatának a reprezentálásáért.

#### Ősosztályok

Tecton -> MultiThreadTecton

#### Interfészek

Nem valósít meg interface-t.

#### Asszociációk

* **asszociáció(branches)**: Másik oldal:FungalThread. Mint az AbsorbingTecton-nál.
* **asszociáció(growing)**: Másik oldal: Mushroom. Van egy Mushroom attribútuma.

#### Attribútumok

* **threads : List<FungalThread>**. Az asszociációnak megfelelően, lehet rajta akár több fonál is.
* **mushroom : Mushroom**. Az aggregációnak megfelelően, maxiumum egy gomba lehet rajta.

#### Metódusok

* **putThread(f : FungalThread) : boolean**. Mint az AbsorbingTecton-nál.
* **putMushroom(m : Mushroom) : boolean**. Mint az AbsorbingTecton-nál.
* **removeThread(f : FungalThread) : boolean.** Mint az AbsorbingTecton-nál.
* **removeMushroom() : boolean.** Mint az AbsorbingTecton-nál.
* **break(): boolean**: boolean: Tekton kettétörését valósítja meg
* **putFirstMushroom(): boolean**: Játékos első gombatestjének lehelyezése

### Spore (Spóra)

#### Felelősség

Ez az absztrakt osztály reprezentálja a spórákat és a hozzájuk tartozó tulajdonságokat. A spórákról tároljuk azt, hogy melyik gombafonálhoz tartoznak. (Egy gombafonál kizárólag egy játékoshoz tartozik.) Ez az osztály azért absztrakt, mert több fajta spóra létezik a játék során, amelyeket leszármazással valósít meg a program. Továbbá az osztály azért is felel, hogy az őt elfogyasztó rovar objektum state attribútumát be tudja állítani az alapján, hogy milyen fajta a példánya. Ez egy absztrakt függvényként jelenik meg ebben az osztályban, amit a leszármazottak fognak megvalósítani.

#### Ősosztályok

Nincs ősosztálya.

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

* **asszociáció (made)**: A túloldali szereplő a Mushroom (gombatest) osztály, amely tartalmazza az általa termelt spóra objektumait, melyeket még nem lőtt ki a játékos kérésére. A kilőtt spórák már nem tartoznak a gombatesthez, csak az adott gombafonálhoz. Miután egy Mushroom objektum elpusztul a hozzá tartozó (általa termelt, de még nem kilőtt) spóra objektumok is elpusztulnak. Azonban a már kilőtt spórák nem pusztulnak ekkor el.
* **asszociáció (use)**: A túloldali szereplő a FungalThread (gombafonál) osztály. A spóra tárolja, hogy melyik gombafonálhoz tartozik. Egy spóra csak egy gombafonálhoz tartozhat.
* **asszociáció (is on):** A túloldali szereplő a Tecton osztály. A tekton tárolja, hogy melyik spórák helyezkednek el rajta. Egy spóra csak egy tektonon lehet.

#### Attribútumok

* **thread**: Az a gombafonál, amihez a spóra tartozik. Fontos, hogy egy gombász játékosnak egy gombafonala van. Így egyértelműen meghatározható, hogy egy játékos fonalához mely spórák tartoznak.

#### Metódusok

* **void applyEffect(i: Insect)**: Absztrakt függvény. Tehát csak a leszármazott osztályok fogják megvalósítani. A paraméterül kapott rovar state attribútumát beállítja a spóra típusa szerint.

### SpeedSpore (Gyorsítóspóra)

#### Felelősség

Azokért a spórákért felelős, amelyek elfogyasztása során a rovar gyorsabb lesz. Gyorsított állapotban a rovar még egyszer léphet.

#### Ősosztályok

Spore (Spóra) -> SpeedSpore (Gyorsítóspóra)

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

Csak olyanok, amelyet az ősosztályától örököl.

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl.

#### Metódusok

* **void applyEffect(i: Insect)**: Az ősosztály absztrakt függvénye, amit az ősosztálynál leírtak szerint valósít meg ez az osztály. Az osztály példányainak a spóratípusa (SporeEffect) SpeedBoost.

### NoCutSpore

#### Felelősség

Azokért a spórákért felelős, amelyek elfogyasztása után a rovar nem tud fonalat vágni. (Ha a körében már vágott fonalat, akkor a következő körben nem fog tudni.)

#### Ősosztályok

Spore (Spóra) -> NoCutSpore

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

Csak olyanok, amelyet az ősosztályától örököl.

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl.

#### Metódusok

* **void applyEffect(i: Insect)**: Az ősosztály absztrakt függvénye, amit az ősosztálynál leírtak szerint valósít meg ez az osztály. Az osztály példányainak a spóratípusa (SporeEffect) NoCutting.

### SlowingSpore (LassítóSpóra)

#### Felelősség

Azokért a spórákért felelős, amelyek elfogyasztása után a rovar lassult állapotba kerül. Ekkor a rovar a következő körben nem léphet, de vághat fonalat.

#### Ősosztályok

Spore (Spóra) -> SlowingSpore

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

Csak olyanok, amelyet az ősosztályától örököl.

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl.

#### Metódusok

* **void applyEffect(i: Insect)**: Az ősosztály absztrakt függvénye, amit az ősosztálynál leírtak szerint valósít meg ez az osztály. Az osztály példányainak a spóratípusa (SporeEffect) Slow.

### ParalyserSpore (BénítóSpóra)

#### Felelősség

Azokért a spórákért felelős, amelyek elfogyasztása során a rovar bénult állapotba kerül. Ekkor a rovar a következő körben nem léphet és nem is vághat fonalat, tehát kimarad egy körből.

#### Ősosztályok

Spore (Spóra) -> ParalyserSpore

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

Csak olyanok, amelyet az ősosztályától örököl.

#### Attribútumok

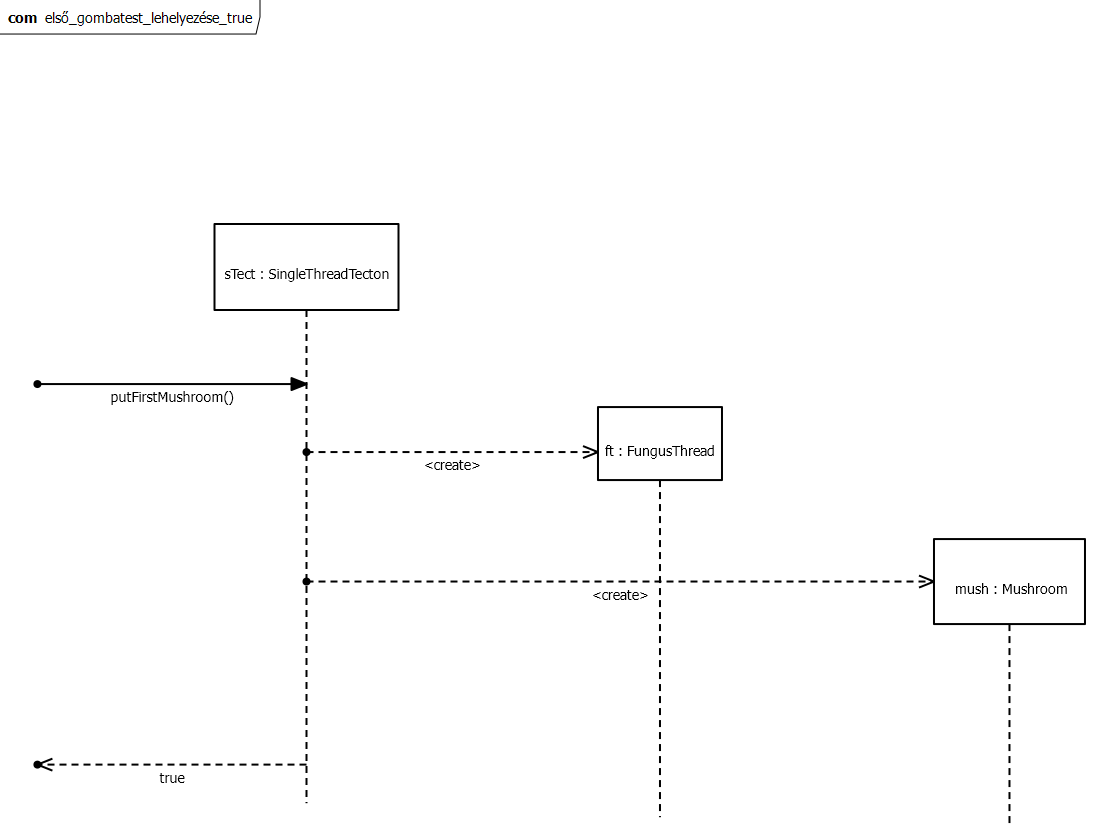
Csak olyanok, amelyet az ősosztályától örököl.

#### Metódusok

* **void applyEffect(i: Insect)**: Az ősosztály absztrakt függvénye, amit az ősosztálynál leírtak szerint valósít meg ez az osztály. Az osztály példányainak a spóratípusa (SporeEffect) Paralysis.

## *Szekvencia diagramok*

**Első gombatest lehelyezése egy tektonra, amelyre le lehet helyezni**

****

A folyamatot a Controller külső komponens indítja.

Ezen a diagramon három objektum található:

* sTect: A tekton, amire le szeretné rakni a gombász a gombatestjét.
* mush: A gombatest, amelyet létrehoz a gombász és le szeretné helyezni az általa kiválasztott tektonon.
* ft: A gombafonál, amelyet létrehoz a gombász és le szeretné helyezni az általa kiválasztott tektonon.

Ezen a szekvenciadiagramon egy külső komponens (ez a Controller lesz, de ezt még ebben a feladatrészben nem kell megjeleníteni) meghívja a **putFirstMushroom** függvényt az sTect objektumon. Ez létrehoz először egy FungusThread objektumot ft néven. Az ft fonál tecton listájába bekerül az sTect objektum a konstruktor által. Ezek után a függvény létrehoz egy Mushroom objektumot mush néven. A konstruktora neki is inicializálja az attribútumait (position = sTect, thread = ft, state = unevolved, shootedSporesCount = 0, spores<Spore>: üres lista). Miután az objektumok megfelelően létrejöttek a **putFirstMushroom** true-val tér vissza.

.**Első gombatest lehelyezése egy tektonra, amelyre nem lehet lehelyezni**

****

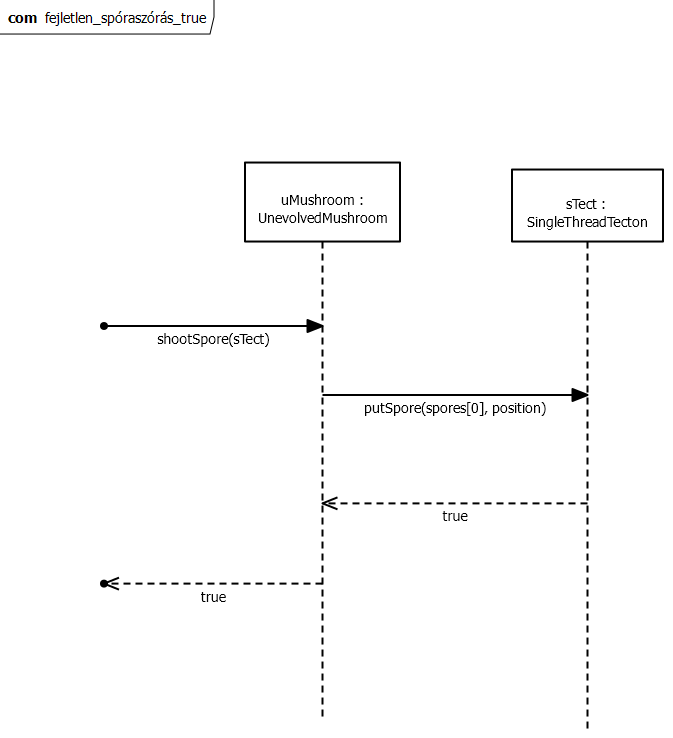
A folyamatot a Controller külső komponens indítja.

Ezen a diagramon egy objektum található:

* aTect: A tekton, amire le szeretné rakni a gombász a gombatestjét.

Az előző szekvenciadiagramhoz hasonlóan itt is a külső komponens meghívja a **putFirstMushroom** függvényt az aTect objektumon, amely false-t ad vissza, mert az AbsorbingTecton típusú tekton objektumokon nem lehet gombatest. Így végül a **putFirstMushroom** függvény false értékkel tér vissza.

**Fejletlen gombatest spóra szórása olyan tektonra, amelyre tud**

****

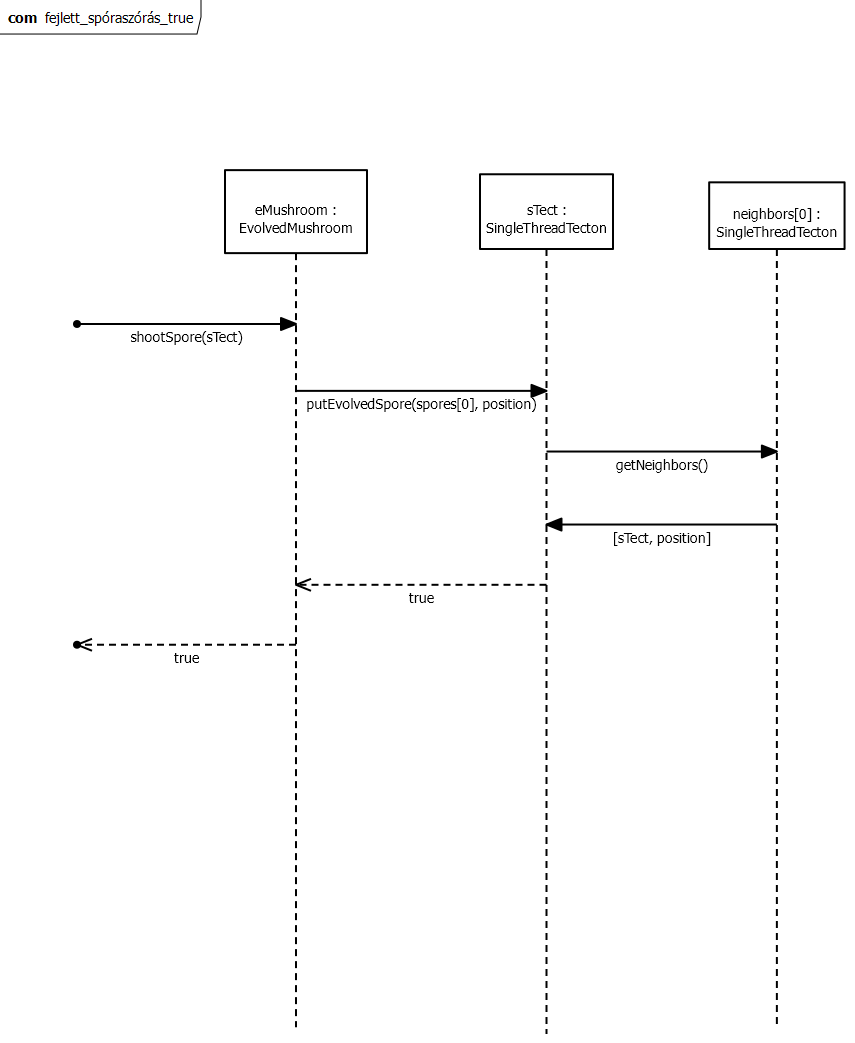
A folyamatot a Controller külső komponens indítja.

Ezen a diagramon két objektum található:

* uMushroom: A fejletlen gombatest, amely a spórát kilövi a játékos kérésére.
* sTect: A tekton, amire kilövi az uMushroom a spórát a játékos kérésére.

A külső komponens meghívja a **shootSpore** metódust az uMushroom objektumon, amely paraméterül az sTect objektumot kapja. Ezután az sTect objektumon meghívódik az uMushroom hatására a **putSpore** függvény, amely paraméterül megkapja az uMushroom által termelt spórák közül a legkorábban termeltet (spores[0]) és azt a tektont, amelyen az uMushroom rajta van (position). A függvény megnézi, hogy az sTectnek szomszédja-e position. Ebben a szekvenciában ez igaz, ekkor ráhelyezi a függvény a tektonra a spores[0] spórát. Egy ilyen szekvenciában csak egy egység spóra kilőhető egy tektonra. Miután ez megtörtént a függvény true-val tér vissza. Ezek után a **shootSpore** függvény is true-val tér vissza.

**Fejlett gombatest spóra szórása olyan tektonra, amelyre tud**



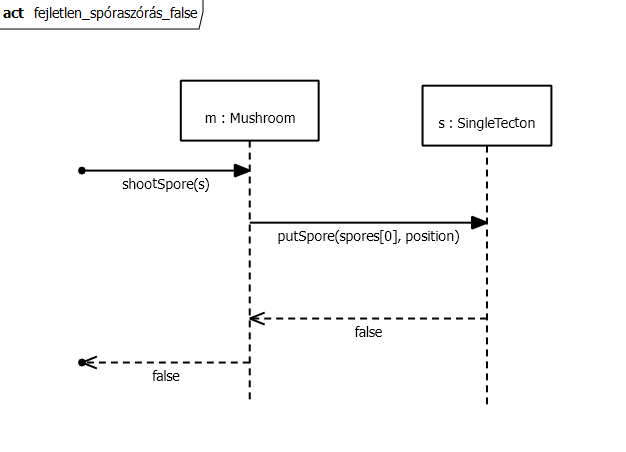
A folyamatot a Controller külső komponens indítja.

Ezen a diagramon három objektum látható:

* eMushroom: A fejlett gomabatest, amely a spórát kilövi a játékos kérésére.
* sTect: A tekton, amire kilövi az eMushroom a spórát a játékos kérésére.
* neighbors[0]: Az a tekton, amely szomszédja az sTectnek és annak a tektonnak, amin a gombatest helyezkedik el. (Az sTect neighbors (szomszédos tektonok listája) attribútuma. Ebben a szekvenciában feltételezem, hogy az sTectnek csak ez az egy szomszédja van, ezért a neighbors Lista 1 elemű).

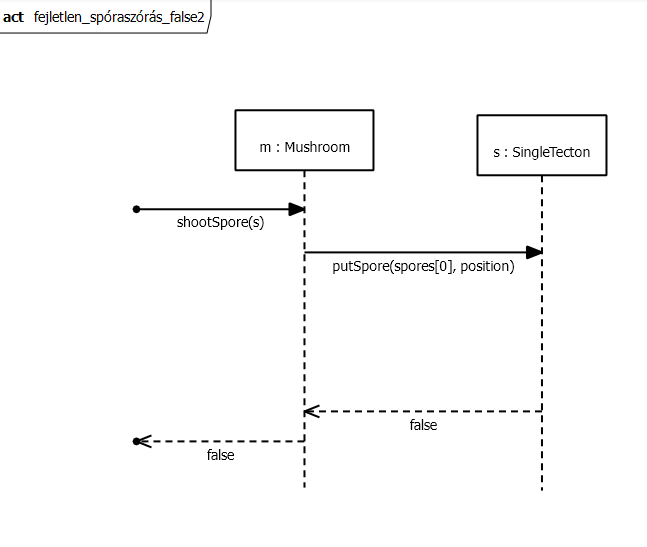
Egy külső komponens meghívja az eMushroom **shootSpore** metódusát, aminek paraméterül adja az sTect tektont. Ezután ez a függvény meghívja az sTect **putEvolvedSpore** függvényét, amely paraméterül kapja az eMushroom által termelt spórák közül a legkorábban termeltet (spores[0]) és azt a tektont, amelyen az eMushroom rajta van (position). A függvény megnézi, hogy az sTectnek szomszédja-e position. Ebben a szekvenciában ez hamis, de mivel fejlett gomabtestről van szó, ezért a függvény az sTect neighbors (szomszédos tektonok listája) attribútumán végig iterál, és mindegyik szomszédos tektonon meghívja a **getNeighbors** metódust. Ebben a szekvenciában az egyszerűség kedvéért a neighbors 1 elemű, ezért csak a neighbors[0] tektonon hívódik meg a **getNeighbors**, amely visszaadja a szomszédait. Ez ebben a szekvenciában egy olyan lista lesz, amiben az sTect és a position szerepel. Így már tudjuk, hogy az sTect a position (a tekton, amin a gombatest van) szomszédjának a szomszédja, így fejlett gombatest esetén szórhatunk rá spórát. Egy ilyen szekvenciában csak egy egység spóra kilőhető egy tektonra. A **putEvolvedSpore** ekkor ráteszi a spórát az sTect-re és true-val tér vissza. Ezek után a **shootSpore** is true-val tér vissza.

**Fejletlen gombatest spóra szórása olyan tektonra, amelyre nem tud, mert nem szomszédos**



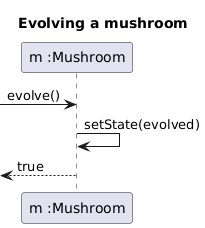
* A folyamatot a controller indítja meg.
* A diagrammon szereplő objektumok:
  + **m**:**Mushroom:** Ő lövi ki a spórát
    - **position:SingleTecton**: A tekton, amin a gomba van.
    - state:MushroomState=UnevolvedMushroom (Csak a szomszédos tektonokra tud spórát tenni.)
    - spores:List<Spore>: A gombában lévő spórák. Feltételezzük, hogy nem üres.
  + **s**:**SingleTecton:** Tekton, amire kilövi az m a spórát.
* A külső komponens meghívja a **shootSpore** metódust az mobjektumon, amely paraméterül az **s** objektumot kapja. - m-nek s-re kell lőnie.
  + A függvény törzsében ellenőrizzük, hogy nem üres a spores. Ha üres lenne visszatérne false-al. Most tfh. nem üres.
* Az sobjektumon meghívódik a **putSpore** függvény, paraméterül megkapja az m spores listájának első elemét és az m position-ét - s tudja, hogy mit és honnan kap.
  + A függvény elleneőrzi, hogy a paraméterül kapott position benne van-e az s neighbors listájában. Most tfh. nincs benne, ezért false-al tér vissza.
* putSpore=false -> nem lehet s-re spórát tenni m-nek.
* shootSpore=false->sikertelen volt a spóra lövés.

**Fejletlen gombatest spóra szórása olyan tektonra, amelyre nem tud, mert már túl sok spóra van rajta.**



* A folyamatot a controller indítja meg.
* A diagrammon szereplő objektumok megegyeznek az előzőével.
* A függvényhívások megegyeznek az előzőével, de itt most s és position szomszédos.
  + Ezután a putSpore függvényben ellenőrizzük: spores.size()<5? - lehet-e még tenni a tektonra spórát?
  + spores.size()<5=false-> nincsen hely már spórának s-en.
* putSpore=false -> nem lehet s-re spórát tenni m-nek.
* shootSpore=false->sikertelen volt a spóra lövés.

**Gombatest fejlődése**

****

Ez a szekvencia diagram a gombatest fejlődését mutatja be. **A folyamat a Controller rétegből indul**, amely meghívja a Mushroom objektum evolve() metódusát. A Mushroom belső állapotot frissíti a setState(evolved) metódus segítségével, ezzel jelezve, hogy a gomba kifejlett állapotba került. A művelet eredményeként true érték tér vissza, amely azt jelzi, hogy a gombatest sikeresen fejlődött.

***Gombatest elpusztulása***

Az e objektum shotSpore(t1) metódusát egy külső kontroller fogja meghívni

Ezen a diagramon a következő objektumok találhatók:

* e: gombatest, amely a 10. spóráját is elszórja, ezért el kell pusztulnia
* t1 : erre a tektonra szórja az e gombatest a spóráját
* position : ezen a tektonon helyezkedik el a gombatest
* f : gombafonál, amely ugyanaból a gombafajból van, mint az e gombatest

**shotSpore(t1)**

egy külső kontroller hívja meg az e objektum ezen metódusát

**putSpore(spores[0], position)**

kilőjük a spórát, ez a gombatest által tárolt spóratömb első eleme lesz, és paraméterül küldjük azt a tektont is, amelyiken a gombatest található, azaz amelyikről érkeznek a spórák, ezáltal a t1 tekton eldöntheti, hogy valóban végre lehet hajtani a spórakilövést

**setshootedSporesCount(shootedSoresCount+1)**

növeljük a kilőtt spórák számát 1-el ha sikeres a kilövés

leellenőrizzük, hogy kilőttük-e a 10. spórát

**removeMushroom()**

mivel kilőttük a 10. spórát törölnünk kell a gombatestet, ezt először a tektonról vesszük le

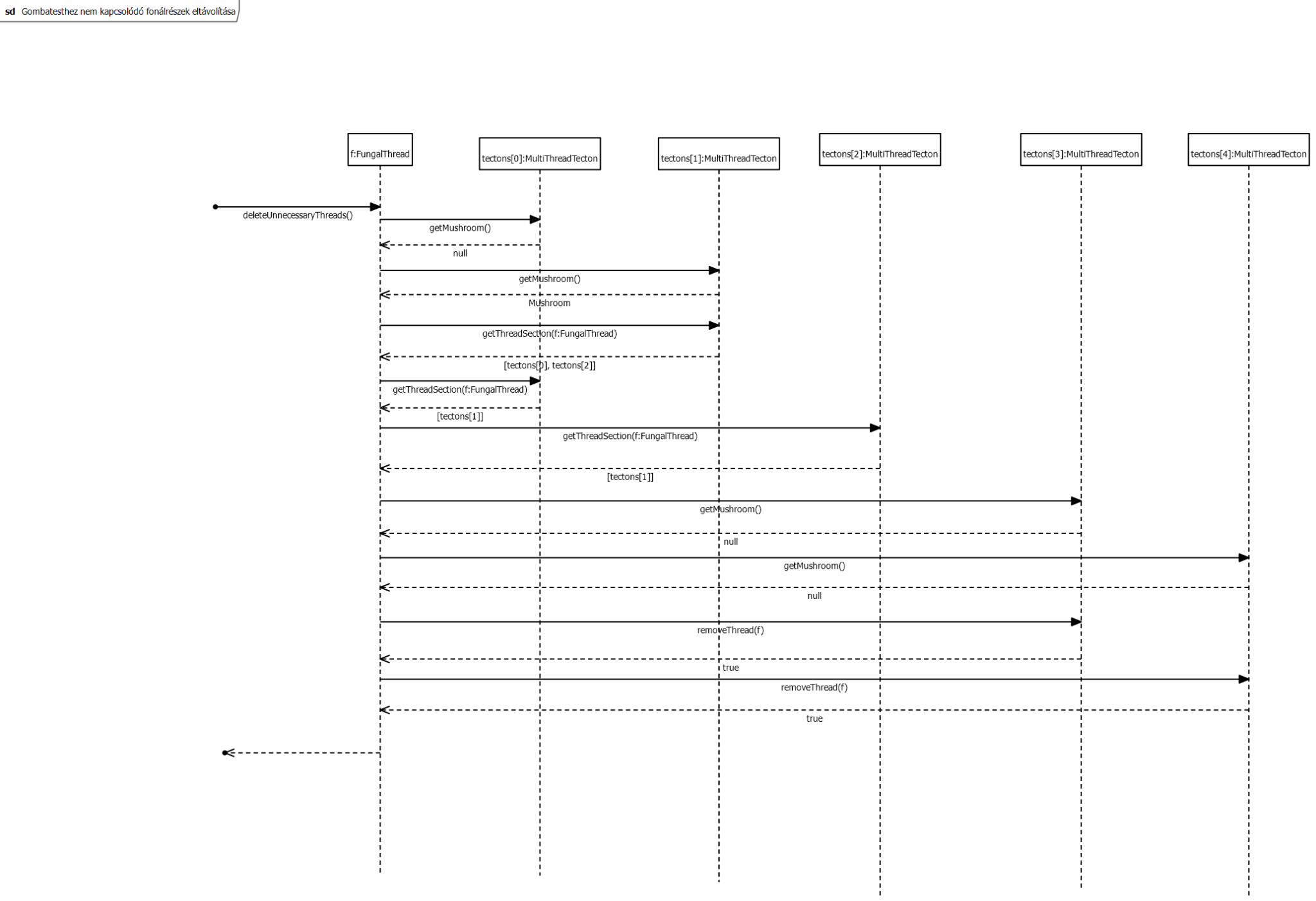
**deleteUnnecessaryThreads()**

a fajhoz tartózó gombafonálhoz töröljük azokat a részeket, amelyek nem kapcsolódnak azonos fajhoz tartozó gombatesthez

**getshootedSporesCount()**

a kontroller lekérdi, hogy hány spórát lőttünk ki eddig

ha ez megegyezik 10-el, akkor töröljük a gombatestet



***Gombatesthez nem kapcsolódó fonálrészek eltávolítása***

A deleteUnnecessaryThread függvényhívás érkezhet egy tektontól, egy gombatesttől, vagy akár egy külső kontrollertől

Ezen a diagramon a következő objektumok találhatók:

* f : gombafonál
* tectons[0] : tekton, amelyen az f gombafonál elágazik
* tectons[1] : tekton, amelyen az f gombafonál elágazik
* tectons[2] : tekton, amelyen az f gombafonál elágazik
* tectons[3] : tekton, amelyen az f gombafonál elágazik
* tectons[4] : tekton, amelyen az f gombafonál elágazik

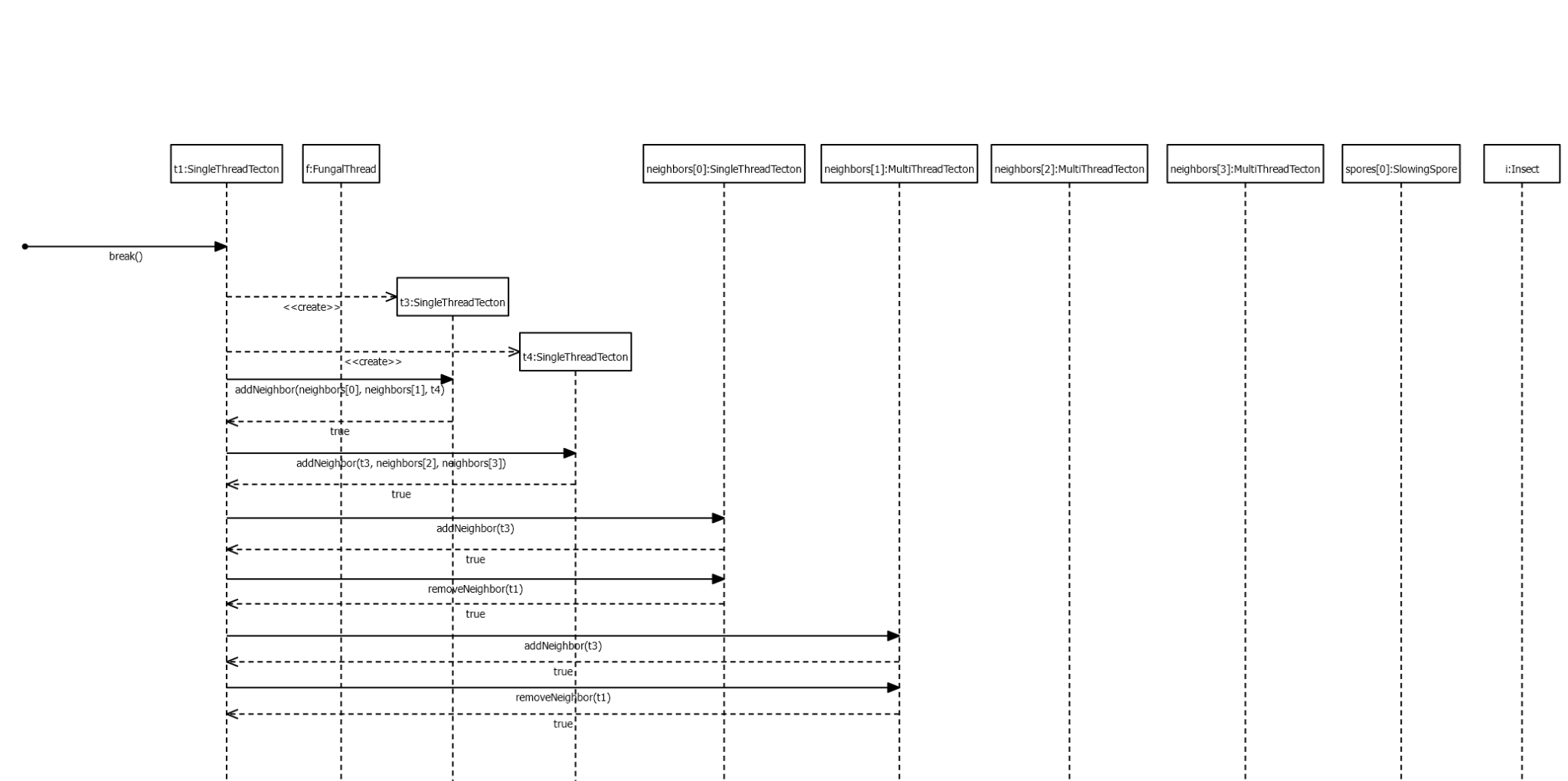
A tectons az f gombafonál egy tömbje, amelyben azokat a tektonokat tárolja, amelyen elágazik.

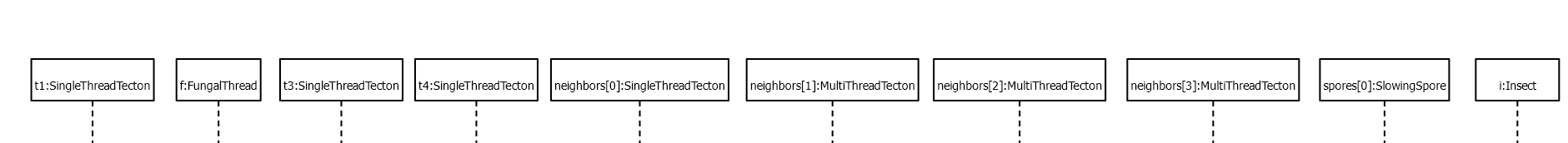
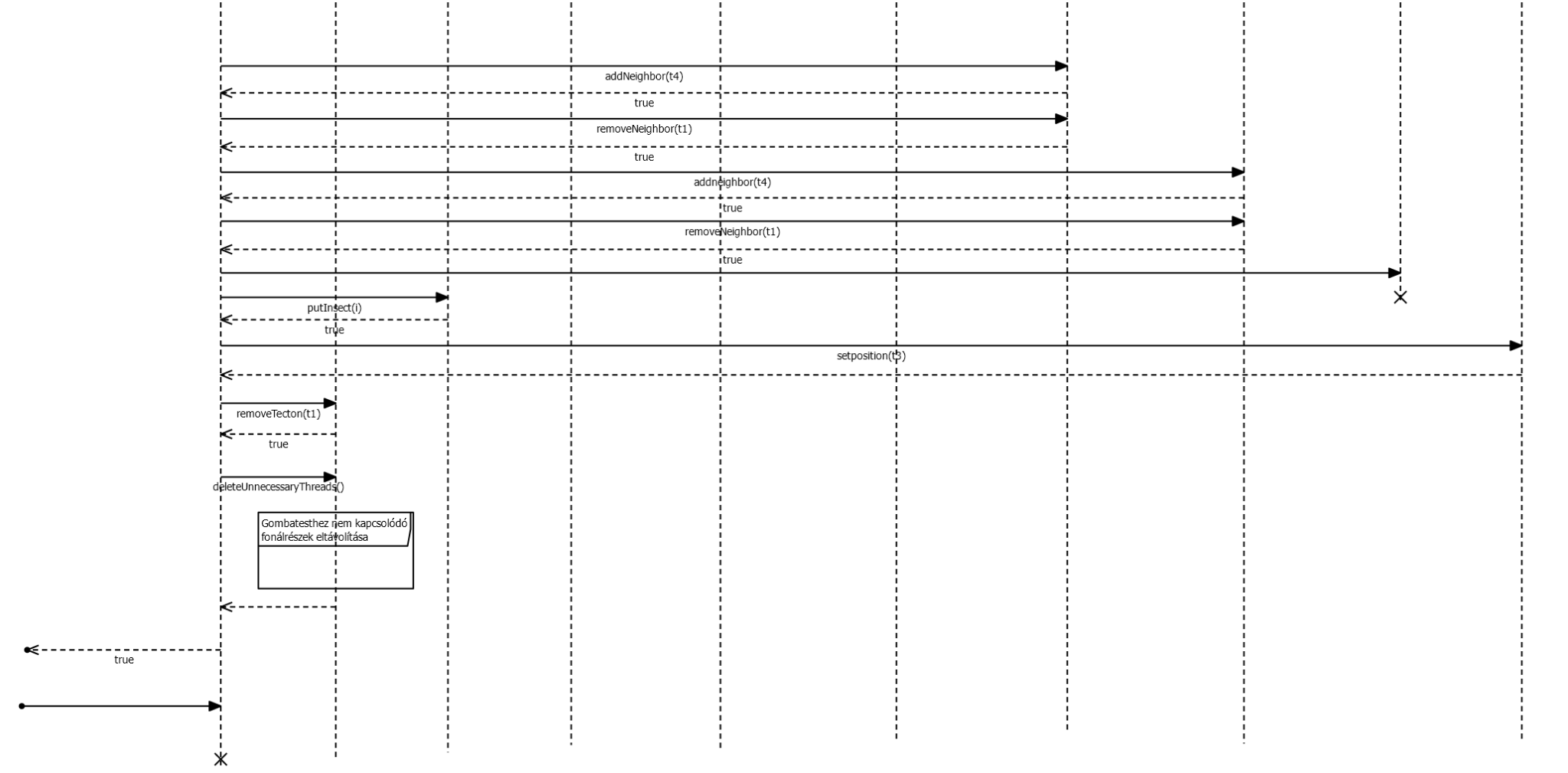
Ezeken a tektonokon sorban végigiterál, és megkérdezi, ha van-e rajtuk gombatest. Ha nincs megnézi a sorban következő tektonra is, ha van, és ez a gombatest ugyanahhoz a gombafajhoz származik, akkor megkeresi azt a fonálrészt, amely ezzel a gombatesttel összeköttetésben van.

Ezt a következő képpen teszi: meghívja a **getThreadSection(f: FungalThread)** metódust arra a tektonra, amelyen a gombatest rajta van, ez visszaadja azokat a tektonokat, amelyek a gombatest tektonjával szomszédosak, és elágazik rajtuk az f gombafonál. Ezután meghívja ezt a metódust az így kapott tektonokra is, és ezt addig csinálja, amig talál új tektont. Ezeket a tektonokat behelyezi egy listába.

Ha már nem talált új tektont, akkor tovább megy a gombafonál tectons listáján, és kérdezi, ha van-e rajtuk gombatest.

Mikor ezt befejezte, azokra a tektonokra, amelyek nem kerültek be az algoritmus által elkészített listába, de benne vannak a gombafonál által tárolt tectons listába, meghívja a **removeThread(f)** metódusukat, ezzel leveszi róluk a fonalat, majd kiveszi őket a tectons listából is, ezzel biztosítva, hogy a fonál nem lesz rajtuk elágazva.

***Tekton kettétörése***

***Folytatás***

Egy külső kontroller meghívja a t1 tekton break() metódusát

Ezen a diagramon a következő objektumok találhatók:

* t1 : ez a tekton fog kettétörni
* f: ez a gombafonál ágazik el a tektonon
* t3: a kettétört tekton egyik része
* t4 : a kettétört tekton másik része
* neighbors[0], neighbors[1],neighbors[2],neighbors[3]: a t1 tektonnal szomszédos tektonok
* spores[0]: a tektonon található spóra
* i: a tektonon található rovar

A t1 tekton **break()** metódusa először két új tektont hoz létre. Ezeknek először be kell állítani a szomszédait. Mivel a szomszédok sorban következnek a neighbors listában, és a tekton a listában tárolt első és az n/2 (felkerekített) szomszédjánál tört el, így tudni fogjuk, hogy a két új tekton melyik tektonokkal lesz szomszédos. Ezután a t1 tektonnal szomszédos tektonoknak is beállítjuk a szomszédsági listáját, valamint kivesszük belőle a t1tektont.

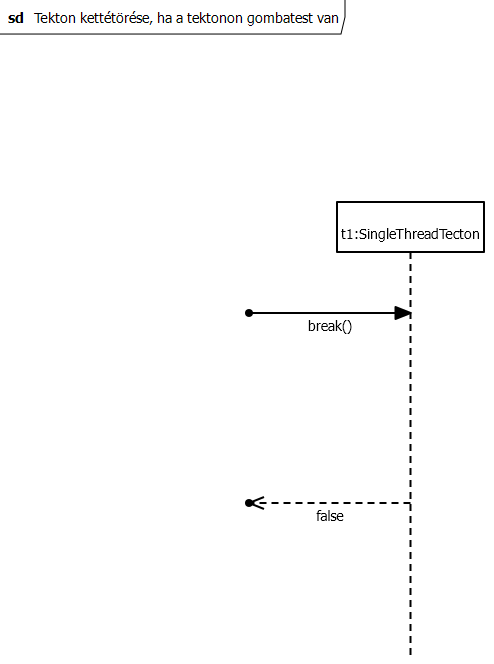
Ha van a t1 tektonon spóra, akkor ezt töröljük.

Ha van a tektonon rovar, ezt ráhelyezzük az elsőnek létrehozott tektonra, és frissítjük a rovar pozícióját.

Ha a t1 tektonon elágazott fonál, akkor erről levesszük a fonalat, majd töröljük azokat a fonálrészeket, amelyek nincsenek összeköttetésben ugyanolyan gombafajból származó tektonnal.

Végül a egy külső kontroller törli a tectont, amely kettétört.

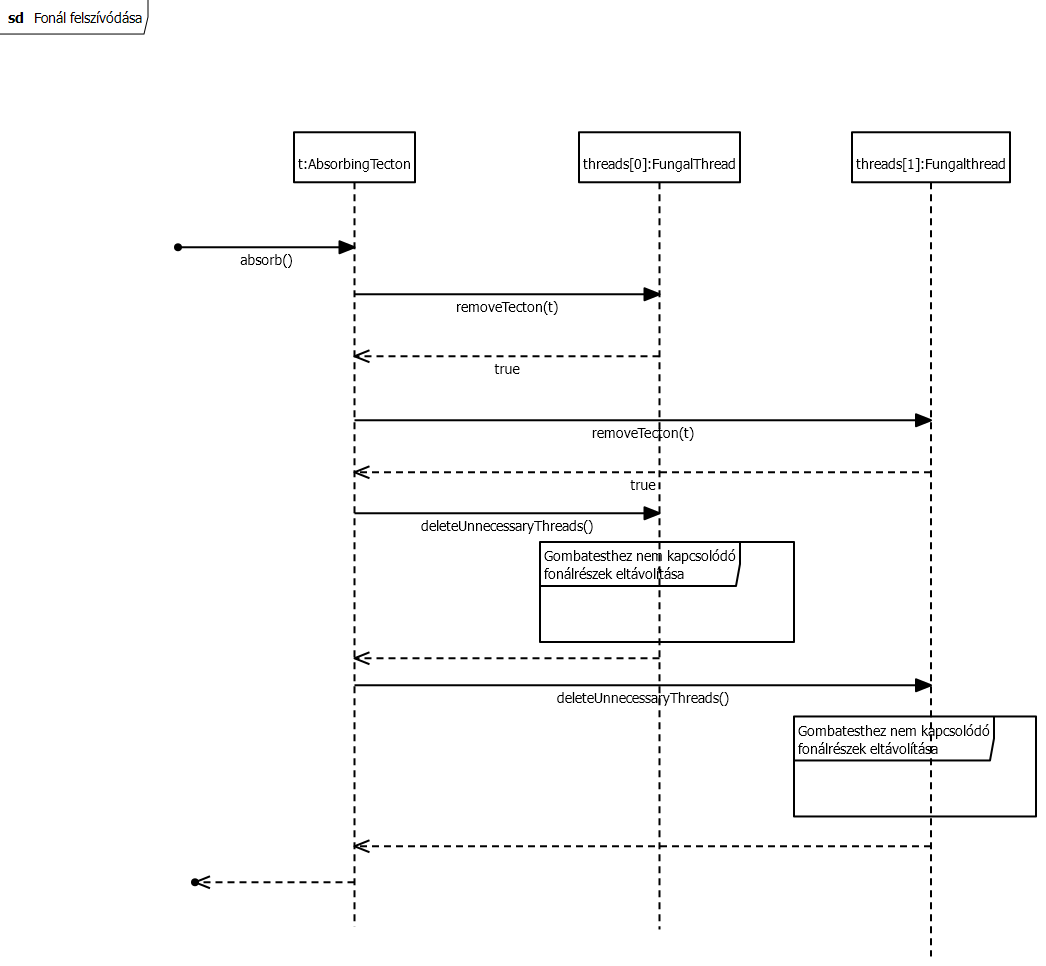
***Tekton kettétörése, ha a tektonon gombatest van***

**

Egy külső kontroller meghívja a t1 tekton break() metódusát

Mivel a tektonon gombatest van, ezért nem törhet ketté, így false-val tér vissza

***Fonál felszívódása tektonról***

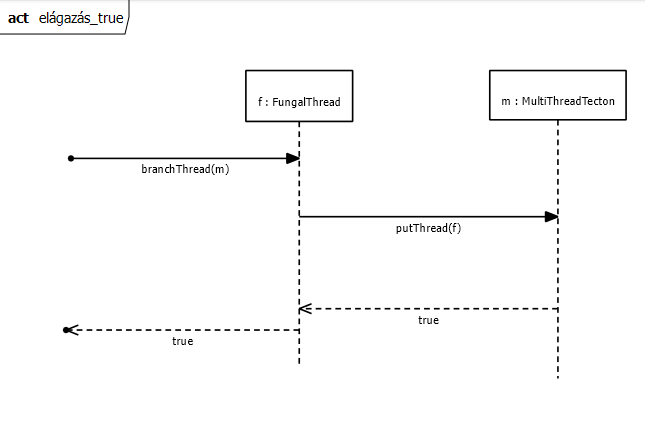
Egy külső kontroller meghívja a t tekton absorb() metódusát

Ezen a diagramon a következő objektumok találhatók:

* t: felszívótekton, amelyről felszívódnak a fonalak
* threads[0]: az egyik gombafonál, amelynek egy része rajta van a tektonon
* threads[1]: egy másik gombafonál, amelynek egy része rajta van a tektonon

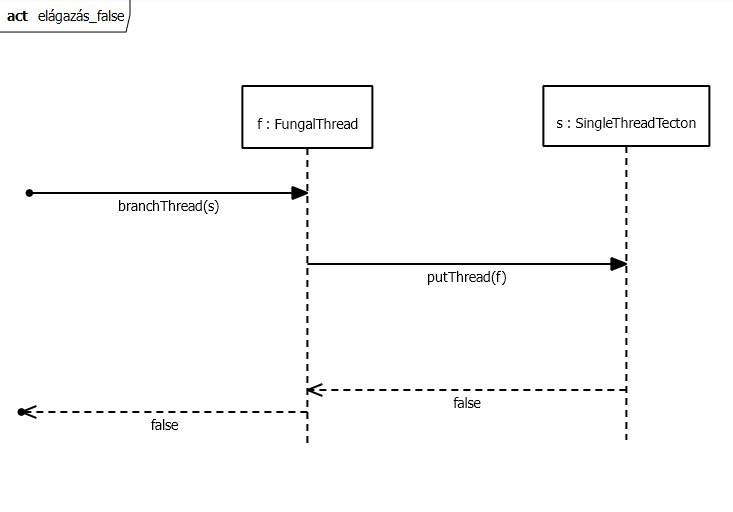
Ezen a szekvencia diagramon szemléltetem a fonál felszívódását egy AbsorbingTecton objektumon, amelyen 5 kör után felszívódnak a gombafonalak. A t tektonon két gombafonál ágazik el, az f1 és az f2. Mivel a t tektonról felszívódtak a fonalak törölni kell, hogy az f1 és az f2 fonál elágazik a t tektonon, ezért meghívódik először az f1 **removeTecton(t)** metódusa, majd az f2 **removeTecton(t)** metódusa, amelyek kitörlik a gombafonalakat a t tektonról, azaz a gombafonalak által tárolt tectons listából kiveszik a paraméterként kapott Tectont, ezután pedig a **deleteUnnecessaryThreads()** függvények segítségével kitöröljük azokat a gombafonál részeket, amelyek már nincsenek összeköttetésben egy ugyanolyan fajból származó gombatesttel.

**Gombafonál elágaztatása Tektonra, és sikeres az elágazás**



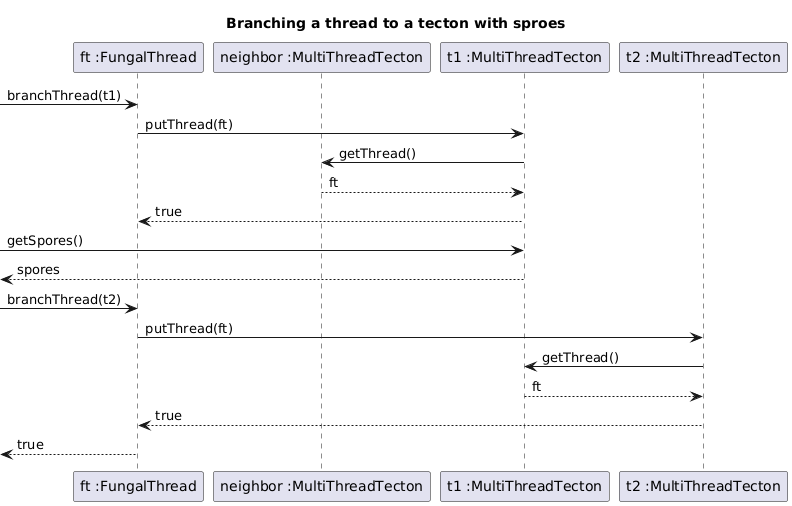
* A folyamatot a controller indítja el.
* Objektumok:
  + **f** : **FungalThread** (fonal)
  + **m** : **MultiThread** (többfonalas tekton)
    - neighbors : List<Tecton>
    - threads : List<FungalThread>
* A külső objektum meghívja az f **branchThread** függvényét m-et átadva paraméterül.
* A **branchThread**-en belül meghívódik a **putThread** függvény m-en, f átadja saját magát.
  + Az m tekton meg nézi, hogy a neigbors-ben lévő tektonokon milyen fonalak vannak. A szomszédaitól le kéri a fonál tömbjüket/fonalukat, és megnézi, hogy talál-e benne f fonállal megegyezőt. Most tfh. talál.
  + A **putThread**-en belül: m felveszi a listájába f-et
* putThread=true->felehet tenni m-re f fonalat.
* Visszatérve a **branchThread**-be f felveszi a listájába m-et és visszatér true-val.

**Gombafonál elágaztatása Tektonra, és nem sikeres az elágazás**



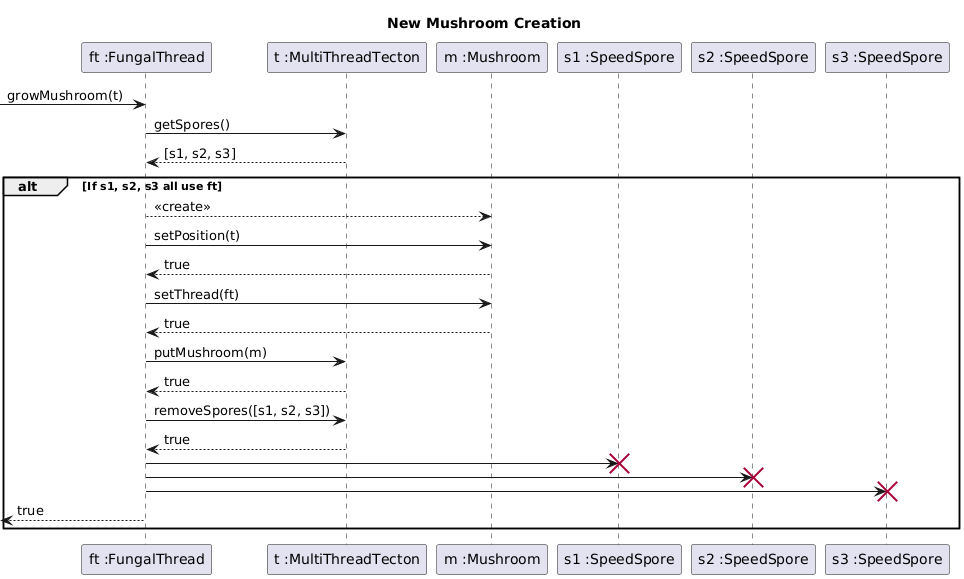
* A folyamatot a controller indítja el.
* Kettő objektum:
  + **f** : **FungalThread** (fonal)
  + **s** : **SingleThreadTecton** (egyfonalas tekton)
* A külső objektum meghívja az **f branchThread** függvényét **s**-et átadva paraméterül.
* A **branchThread**-en belül meghívódik a **putThread** függvény **s**-en, **f** átadja saját magát.
  + A **putThread**-en belül: **s** először megnézi, hogy van-e már rajta fonál. Most tfh.
* Mivel van már rajta fonál és egyfonalas tektonokon csak egy fonál ágazhat el, ezért **putThread=**false->s nem tudja elraktározni f-et.
* **branchThread=**false->f nem tehető s-re.

**Gombafonal elágazása**



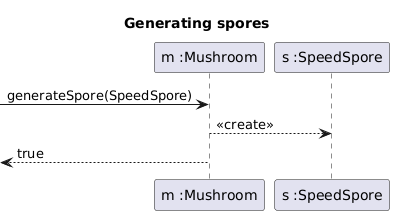
Ez a szekvencia diagram azt mutatja be, hogy egy tektonon található spóra hogyan befolyásolja a gombafonal elágazását. Ha a kiválasztott tektonon van spóra, akkor még egy további tekton kiválasztható a fonal terjesztésére. A folyamat során a rendszer ellenőrzi a szomszédos tektonokat és a spórák jelenlétét, mielőtt a fonal továbbágazna. **A Controller meghívja a branchThread(ft) metódust a FungalThread objektumon.** A FungalThread kezdeményezi a fonál hozzáadását (putThread(ft)) a szomszédos Tecton-on. A szomszédos Tecton lekéri a jelenlegi fonalakat (getThread()) és visszaküldi az ft fonalat. Ha a fonal sikeresen hozzáadódik, true érték tér vissza. A FungalThread ellenőrzi, hogy vannak-e spórák az adott tektonon (getSpores()). Ha megfelelő mennyiségű spóra van jelen, a második tektonon is megismétlődik a folyamat, és egy újabb fonalat növeszt (putThread(ft)). A második tekton is lekéri a meglévő fonalat (getThread()) és visszaküldi az ft értéket. Ha minden feltétel teljesült, az ágaztatás sikeres, és true érték tér vissza.

**Új gombatest lerakása**

****

Ez a szekvencia diagram azt szemlélteti, hogyan jön létre egy új gombatest egy tektonon, ha megfelelő feltételek teljesülnek. A folyamat során a rendszer ellenőrzi a spórák jelenlétét és a fonalak állapotát, majd végrehajtja a szükséges módosításokat. **A Controller meghívja a growMushroom(t) metódust a FungalThread objektumon,** amely elindítja a folyamatot. A FungalThread lekéri a spórák listáját a célzott tektonról a getSpores() metódussal. A tekton visszaküldi az elérhető spórák listáját (s1, s2, s3). Ha mindhárom spóra ugyanazt a fonalat használja, akkor, létrejön a Mushroom objektum. A Mushroom beállítja a pozícióját a tektonon. A fonál beállítása a tektonon megtörténik. A Mushroom elhelyezése a tektonon végbemegy. A spórák eltávolítása. A művelet befejeződik, és a három spóra törlődik. A folyamat végén a growMushroom(t) metódus visszatér true értékkel, amely megerősíti a sikeres gombatest létrehozást.

**Új spóra termelődése**

****

Ez a szekvencia diagram azt mutatja be, hogyan generál egy fejlődő gombatest egy spórát. A folyamat során a rendszer létrehozza a megfelelő spóra objektumot, és visszajelzi a sikeres műveletet. **A Controller meghívja a Mushroom objektum generateSpore(SpeedBoost) metódusát**, amely elindítja a spóra létrehozását. A SpeedSpore objektum létrejön és a metóduson belül hozzáadódik az eltárolt spórák listájához. A Mushroom objektum is visszaküldi a true értéket, jelezve, hogy a folyamat befejeződött.

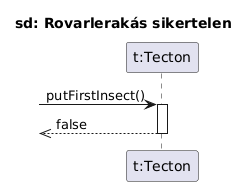
**Rovar elhelyezése játék elején:**

**-Megfelelő tektonra sikerül:**



A diagramon 2 objektum található:Tekton, Rovar. A játék elején a kontrollerből érkezik egy putFirstInsect() fűggvényhivás a tektonra, amennyiben a tetkon insect adattagja null értékű, azaz nem foglalt, létrehozza a rovart. A rovaron beállítódik a pozícióra az adott tektonra, majd a tektonon is beállítódik a rovar mező az adott rovarra. Ez után a putFirstInsect() visszatér igaz értékkel a controller felé.

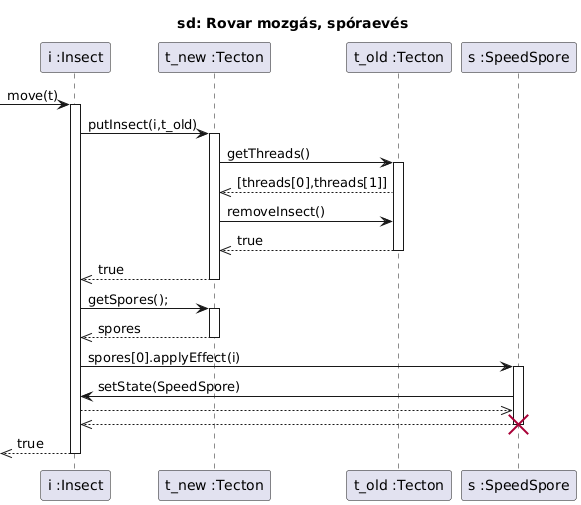
**-Nem megfelelő tektonra nem sikerül**



A diagramon az az eset szemléltetett, amikor a kontroller felől érkező putFirstInsect() függvény futása során az derül ki, hogy a tektonon már van bogár, mivel az insect adattagja nem null értékű, ezért a művelet nem megengedett, hamissal tér vissza a függvény a kontroller felé.

**Rovar mozgása és spóra evés:**

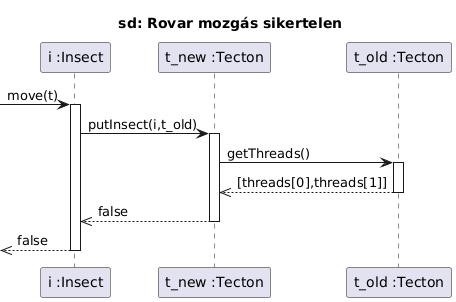
**-Sikeres mozgás és spóraevés**



A diagramon szereplő 4 objektum a következők: Rovar,Régi és új tekton, Spóra. Azt mutatja be az ábra, amikor a Rovar sikeresen mozog egy tektonra, a mozgás akkor sikeres, ha a két tekton között van gombafonal és szomszédosak(a függvényen belül megnézi, hogy a tekton szomszéd listájába szerepel-e a másik). Ekkor az ott lévő spórák közül megesz egyet és megkapja annak a hatását, majd a spóra megszűnik létezni. Alternatívan, ha a tekton spóra listája üres, akkor csak a mozgás történik, és nem kap effektet.

A diagram elején az i:Insect position adattagja egyenlő t\_old tektonnal, ezt adjuk át a putInsect-ben.

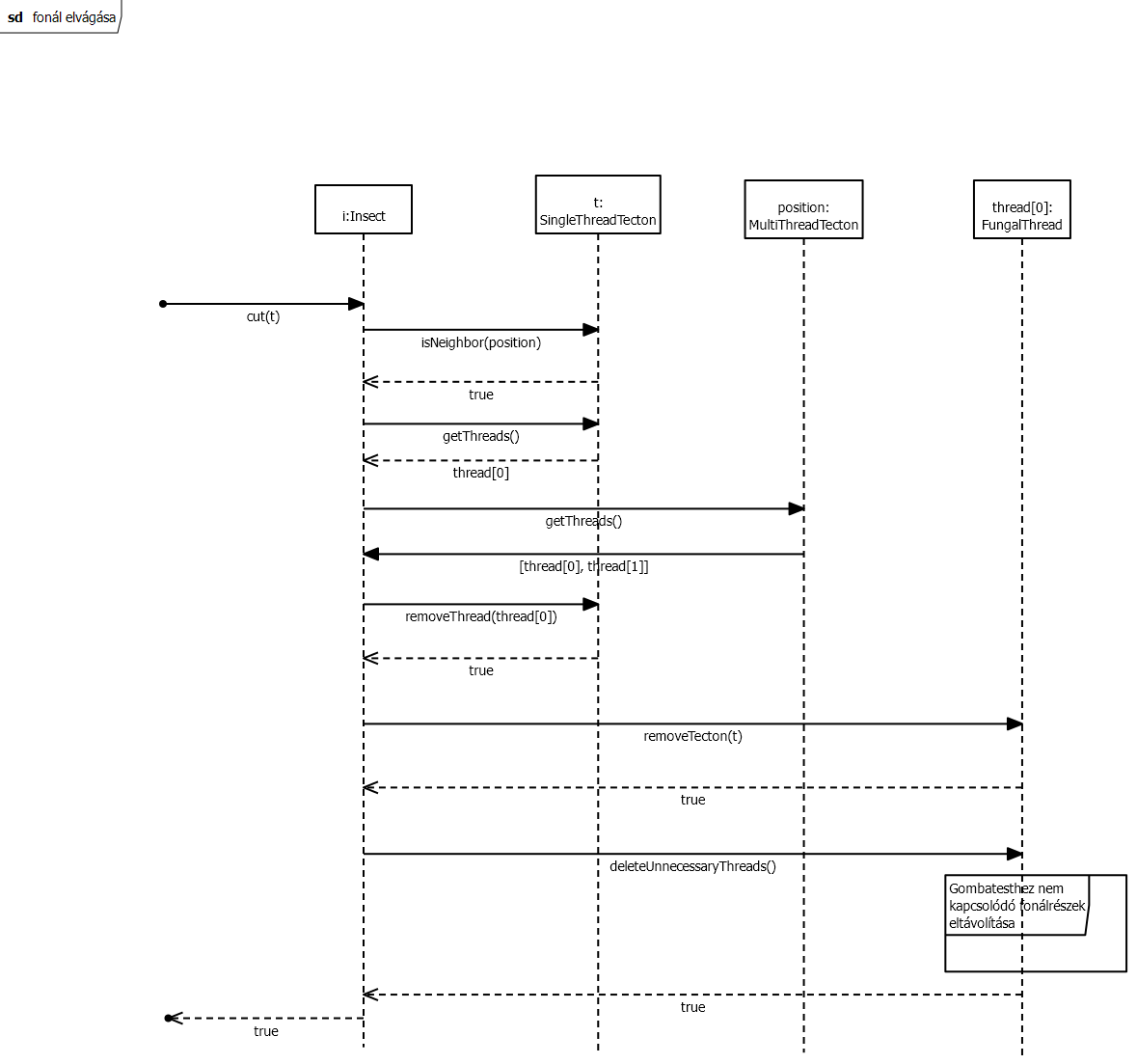
**-Sikertelen mozgás**



Ez a diagram a rovar sikertelen mozgását mutatja be, akkor következik be, ha a szomszédosság és a fonallal való összeköttetés feltételek közül legalább az egyik sérül. Ebben az esetben a getThreads() függvény visszatérési értékét és a saját threads adattagját összehasonlítva nem talál egyezést, tehát nincs fonal összeköttetés így a feladat sikertelen, hamissal tér vissza a függvény.

A diagram elején az i:Insect position adattagja egyenlő t\_old tektonnal ezt adjuk át a putInsect-ben.

**Gombafonál elvágása**



Egy külső kontroller meghívja az i rovar cut metódusát.

Ezen a diagramon a következő objektumok találhatók:

* i: Rovar, amely a vágást végzi
* t: ezt a tektont jelöltük ki
* position: ezen a tektonon található a rovar
* thread[0]: gombafonál, amely a t és a position tektonon is elágazik

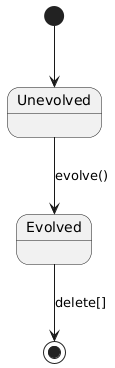
A **cut(t)** metódushívás után leellenőrizzük, hogy a paraméterül kapott t tekton szomszédos-e a position tektonnal, azaz azzal a tektonnal, amelyen a rovar rajta van.

Ha ez teljesül, akkor mindkét tektonról lekérjük a gombafonalakat mindkettőről külön-külön. Ha találunk olyat, amely mindkettőn elágazik, akkor ezt levesszük a t tektonról, majd a t tektont kivesszük a fonál listájából, végül pedig a **deleteUnnecessaryThreads()** függvényhívással kitöröljük azokat a fonálrészeket, amelyek nincsenek összeköttetésben azonos fajból származó tektonnal.

.

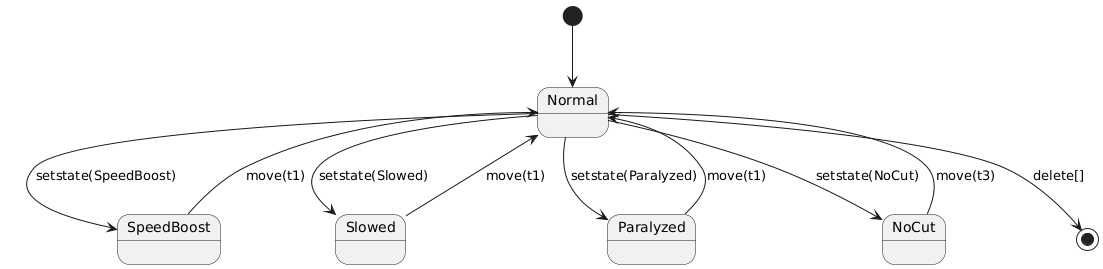
## *State-chartok*

**Gombatest állapot diagramja**



Az állapotdiagram a gombatest állapotait szemlélteti. Minden gombatest létrejötte után fejletlen, azaz Unevolved állapotban van. A gombatest létrejötte után 5 kör elteltével az evolve() hívás hatására a gombatest fejletté válik. Mindaddig élni fog a gombatest, amíg ki nem lövi a 10. spóráját, ekkor a játék automatikusan törli meghívva a destruktorát.

**Rovar állapot diagramja**



Az állapotdiagram a rovar állapotait szemlélteti. Minden rovar létrejöttekor Normal állapotba kerül. A rovarnak akkor változik meg az állapota, ha egy spórát megeszik, majd a megevett spóra annak függvényében, hogy milyen típusú, beállítja a rovar state attribútumát, és ennek függvényében a következő állapotokba kerülhet: Slowed, Paralyzed, noCut, SpeedBoost. A Slowed, NoCut, Paralyzed állapotból a rovar a játék következő körében lép át Normal-ba, mikor ezeknek az állapotoknak megfelelően végrehajtja a lépését. A SpeedBoost állapotból akkor kerül vissza Normal-ba, miután még lép egy újabb tektonra. A rovar a játék befejeztével szűnik meg létezni. A delete[] azaz a destruktorívás hatására törlődik az objektum.

## *Napló*

| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| --- | --- | --- | --- |
| 2025.03.08 11:00 | 5 óra | Toronyi  Lekli  Zelch  Szolár  Fejes | Konzultáció  Döntések:  Megbeszéltük az osztálydiagram változtatásával kapcsolatos kérdéseket, és elkészült az osztálydiagram legfrissebb verziója, ezután minden szekvenciadiagramot végigneztünk külön-külön és megbeszéltük a változatásokat |
| 2025.03.08. 17:00 | 4 óra | Lekli | Szekvencia diagramok finomítása |
| 2025.03.08 18:00 | 3 óra | Zelch | Szekvencia diagramok finomítása |
| 2025.03.08. 18:00 | 3 óra | Toronyi | Osztályok változásának feltérképezése, szekvencia diagramok átszervezése |
| 2025.03.09. 11:00 | 3 óra | Toronyi | Szekvencia diagramok befejezése |
| 2025.03.09. 11:00 | 2 óra | Fejes | Szekvencia diagramok módosítása |
| 2025.03.08. 12:00 | 4 óra | Szolár | Szekvencia diagramok módosítása és osztály leírások frissítése. |
| 2025.03.09 19:00 | 3 óra | Zelch | Szekvencia diagramok finomítása, osztálydiagram finomítása |
| 2025.03.09 20:00 | 2 óra | Fejes | Osztályleírás módosítása: Spóra és leszármazottai, Tekton |
| 2025.03.10 8:00 | 2 óra | Zelch | Utolsó finomítások szekvenciadiagramokon, illetve a kiosztott osztályok leírásán |
| 2025.03.10 9:00 | 2 óra | Fejes | Szekvencia diagramok átnézése, dokumentum formázása |

**5. Szkeleton tervezése**

26 – five\_guys

Konzulens:

Huszerl Gábor

Csapattagok

| Lekli Zsolt | JENKHQ | zsoltlekli@gmail.com |
| --- | --- | --- |
| Fejes Ábel | AF1Y1C | fejesabel@gmail.com |
| Zelch Csaba | LK0617 | zch03po@gmail.com |
| Szolár Soma | E05SC6 | somaszolar1@gmail.com |
| Toronyi Zsombor | S8F7DV | toronyizsombi@gmail.com |

2025. 03. 17.

1. **Szkeleton tervezése**
   1. ***A szkeleton modell valóságos use-case-ei***
      1. **Use-case diagram**

-

* + 1. **Use-case leírások**

Aktor: mindig a tesztelő

| **Név** | Első gombatest lehelyezése olyan tektonra, amelyre le lehet helyezni |
| --- | --- |
| **Leírás, cél** | Azt az esetet teszteli, amikor egy gombász le szeretné tenni az első gombatestjét a játék kezdetekor egy olyan tektonra, amelyre le tudja a szabályok szerint. Ez egy olyan tekton lehet, amin nincs másik gombatest és a fajtája nem AbsorbingTecton. |
| **Előfeltétel** | Lennie kell:   * t4: MultiThreadTecton - A tekton, amire le szeretné rakni a gombász a gombatestjét.   Teljesülnie kell:   * t4 SingleThreadTecton vagy MultiThreadTecton * t4-en nincs gombatest * Ha t4 SingleThreadTecton, akkor t4-en nincs gombafonál (FungalThread) |
| **Hivatkozott kommunikációs diagram** | Diagram 1 |
| **Bemenet** | A tester meghívja a t4-en a **putFirstMushroom** metódust. |
| **Forgatókönyv** | * Mivel a t4 MultiThreadTecton, ezért a **putFirstMushroom** függvény csak azt ellenőrzi le, hogy van-e tektonon gombatest. (Ha SingleThreadTecton lenne, akkor azt is le kéne ellenőrizni, hogy van-e a tektonon gombafonál. AbsorbingTecton esetén a függvény szimplán false-al tér vissza). * Ha nincs, akkor létrehoz először egy fonalat, utána egy gombatestet. * A gombatest fonalának (thread attribútum) a létrehozott gombafonalat állítja be a függvény. * A fonálnak és a gombatestnek is beállítja az attribútumait a tektonnak megfelelően. * Ezek után a **putFirstMushroom** függvény true-val tér vissza. |
| **Elvárt kimenet** | A **putFirstMushroom** függvény true-val tér vissza, tehát t4 tektonon lesz egy új gombafonál és egy új gombatest. |
| **Hivatkozott szekvencia diagram** | első\_gombatest\_lehelyezése\_true |

| **Név** | Első gombatest lehelyezése egy tektonra, amelyre nem lehet lehelyezni (AbsorbingTecton) |
| --- | --- |
| **Leírás, cél** | Azt az esetet teszteli, amikor egy gombász le szeretné tenni az első gombatestjét a játék kezdetekor egy olyan tektonra, amelyre nem tudja a szabályok szerint. Ez ebben az esetben egy AbsorbingTecton. |
| **Előfeltétel** | Lennie kell:   * t5: AbsorbingTecton - A tekton, amire le szeretné rakni a gombász a gombatestjét.   Teljesülnie kell:   * t5 egy AbsorbingTecton. |
| **Hivatkozott kommunikációs diagram** | Diagram 1 |
| **Bemenet** | A tester meghívja a t5-ön a **putFirstMushroom** metódust. |
| **Forgatókönyv** | Mivel a t5 tekton egy AbsorbingTecton, ezért a **putFirstMushroom** false-al tér vissza. |
| **Elvárt kimenet** | A t5 tekton változatlan marad, tehát nem kerül rá se új gombafonál, se új gombatest. |
| **Hivatkozott szekvencia diagram** | első\_gombatest\_lehelyezése\_false\_absorb |

| **Név** | Első gombatest lehelyezése egy tektonra, amelyre nem lehet lehelyezni (van gombatest a tektonon) |
| --- | --- |
| **Leírás, cél** | Azt az esetet teszteli, amikor egy gombász le szeretné tenni az első gombatestjét a játék kezdetekor egy olyan tektonra, amelyre nem tudja a szabályok szerint. Ez ebben az esetben egy olyan tekton, amelyen már van gombatest. |
| **Előfeltétel** | Lennie kell:   * t1: MultiThreadTecton - A tekton, amire le szeretné rakni a gombász a gombatestjét. * f: FungalThread - A gombafonál, amely elágazik t1-en. * m: Mushroom - A gombatest, amely a t1 tektonon helyezkedik el.   Teljesülnie kell:   * m gombatest és f fonál a t1 tektonon helyezkedik el. |
| **Hivatkozott kommunikációs diagram** | Diagram 1 |
| **Bemenet** | A tester meghívja a t1-en a **putFirstMushroom** metódust. |
| **Forgatókönyv** | * Mivel a t1 MultiThreadTecton, ezért a **putFirstMushroom** függvény csak azt ellenőrzi le, hogy van-e tektonon gombatest. * Mivel van, ezért a függvény visszatérési értéke false. |
| **Elvárt kimenet** | A t1 tekton változatlan marad. |
| **Hivatkozott szekvencia diagram** | első\_gombatest\_lehelyezése\_false\_vangomba |

| **Név** | Új gombatest lerakása |
| --- | --- |
| **Leírás, cél** | Azt az esetet teszteli, amikor egy gombász le szeretne tenni egy új gombatestet egy olyan tektonra, amelyre le tudja a szabályok szerint. |
| **Előfeltétel** | Lennie kell:   * f: FungalThread - A gombászhoz tartozó gombafonál. * t3: SingleThreadTecton - A tekton, amire a gombász le szeretné helyezni egy új gombatestet. * s1: SlowingSpore, s2: NoCutSpore, s3: ParalysedSpore - A t3 tektonon lévő spórák.   Teljesülnie kell:   * t3 tektonon el kell lennie ágazva az f fonálnak. * t3 tektonon kell, hogy legyen legalább 3 olyan spóra, amely az f fonálhoz tartozik. Tehát ez esetben s1, s2 és s3-nak is az f-hez kell tartoznia. * t3 tektonon nem lehet alapból gombatest. |
| **Hivatkozott kommunikációs diagram** | Diagram 5 |
| **Bemenet** | A tester meghívja a **growMushroom**(t3) metódust f a FungalThread objektumon t3 paraméterrel. |
| **Forgatókönyv** | * A **growMushroom** először ellenőrzi, hogy t3 tektonon van-e gombatest. * Ha nincs, akkor a FungalThread lekéri a spórák listáját a célzott tektonról a **getSpores**() metódussal. * A tekton visszaküldi az elérhető spórák listáját (s1, s2, s3). * Ha mindhárom spóra ugyanazt a fonalat használja, akkor, létrejön a Mushroom objektum. * A Mushroom beállítja a pozícióját a tektonon (**setPosition**). * Továbbá beállítja a fonalát is (**setThread**). * A Mushroom elhelyezése a tektonon végbemegy a **putMushroom** függvény segítségével. * Ezek után a spórákat eltávolítja a **growMushroom** függvény (**removeSpores**). * A folyamat végén a **growMushroom** metódus visszatér true értékkel, amely megerősíti a sikeres gombatest létrehozást. |
| **Elvárt kimenet** | A t3 tektonon megjelenik az új gombatest a megfelelő attribútumokkal. |
| **Hivatkozott szekvencia diagram** | New Mushroom Creation |

| **Név** | Fejletlen gombatest spóra szórása olyan tektonra, amelyre tud |
| --- | --- |
| **Leírás, cél** | Azt az esetet teszteli, amikor egy gombász hatására egy mushroom objektum (ami fejletlen) spórát szór egy szomszédos tektonra. Ebben az esetben ez megtehető a játék szabályai szerint. |
| **Előfeltétel** | Lennie kell:   * t3:SingleThreadTecton - A cél tekton, amire a spórát a gombatest lőni szeretné. * t1:MultiThreadTecton - A tekton amin a gomba van. * m:Mushroom - A gomba ami lő. * s2:Spore - Spóra, amit a gomba kilő.   Teljesülnie kell:   * t1 és t3 szomszédosak: Benne vannak egymás neighbors listájában. * m rajta van t1-en:   + t1.mushroom=m   + m.position=t1 * m-nek van spórája, amit kilő, ez most s2:   + m.spores[0]=s2 * m fejletlen gombatest: m.state=Unevolved (alapértelmezett állapot létrehozás után) |
| **Hivatkozott kommunikációs diagram** | Diagram 1 |
| **Bemenet** | A Tester meghívja az m shootSpore függvényét t3-at átadva paraméterül . |
| **Forgatókönyv** | * A **shootspore** függvény törzsében ellenőrizzük, hogy nem üres a spores. Ha üres lenne visszatérne false-al. Most nem üres, benne van s2. * A t3objektumon meghívódik a **putSpore** függvény, paraméterül megkapja az m spores listájának első elemét(s2) és az m position-ét (t1). * A **putSpore** függvény ellenőrzi, hogy a paraméterül kapott position benne van-e a t3 neighbors listájában. Most benne, ezért a függvény ráhelyezi t3 tektonra az s2 spórát, és true-val tér vissza. * Ezek után a **shootSpore** függvény is true-val tér vissza. |
| **Elvárt kimenet** | A t3 tektonon rajta lesz s2 spóra. Továbbá s2 spóra kikerült az m Mushroom spores listájából. |
| **Hivatkozott szekvencia diagram** | fejletlen\_spóraszórás\_true |

| **Név** | Fejlett gombatest spóra szórása olyan tektonra, amelyre tud |
| --- | --- |
| **Leírás, cél** | Azt az esetet teszteli, amikor egy gombász hatására egy mushroom objektum (ami fejlett) spórát szór egy olyan tektonra, ami a szomszédjának a szomszédja. Ebben az esetben ez megtehető a játék szabályai szerint. |
| **Előfeltétel** | Lennie kell:   * t2: SingleThreadTecton - A cél tekton, amire a spórát a gombatest lőni szeretné. * t1:MultiThreadTecton - A tekton amin a gomba van. * m:Mushroom - A gomba ami lő. * s4: SpeedSpore - Spóra, amit a gomba kilő. * t3: SingleThreadTecton - A tekton, ami szomszédja t2-nek és t1-nek is.   Teljesülnie kell:   * t1 és t2 nem szomszédosak, de t1 szomszédjának (t3) a szomszédja t2. * m rajta van t1-en:   + t1.mushroom=m   + m.position=t1 * m-nek van spórája, amit kilő, ez most s4:   + m.spores[0]=s4 * m fejlett gombatest: m.state=Evolved * t3-nak szomszédja t1 és t2 is. |
| **Hivatkozott kommunikációs diagram** | Diagram 5 |
| **Bemenet** | A Tester meghívja az m shootSpore függvényét t2-őt átadva paraméterül . |
| **Forgatókönyv** | * A **shootspore** függvény törzsében ellenőrizzük, hogy nem üres a spores. Ha üres lenne visszatérne false-al. Most nem üres, benne van s4. * A t2objektumon meghívódik a **putEvolvedSpore** függvény, paraméterül megkapja az m spores listájának első elemét (s4) és az m position-ét (t1). * A **putEvolvedSpore** függvény először ellenőrzi, hogy a paraméterül kapott position benne van-e a t2 neighbors listájában. Most nincs benne. * Ezek után a **putEvolvedSpore** függvény az t2 neighbors (szomszédos tektonok listája) attribútumán végig iterál, és mindegyik szomszédos tektonon meghívja a **getNeighbors** metódust. * Ellenőrzi, hogy van-e olyan tekton a t2 szomszédjai között, aminek a getNeighbors által visszaadott tekton listában benne van t1. Most t3 egy ilyen tekton. * Ekkor tudjuk, hogy t2 t1 szomszédjának a szomszédja. Így a **putEvolvedSpore** függvény ráhelyezi az s4-et t2 tektonra és true-val tér vissza. * Ezek után a **shootSpore** függvény is true-val tér vissza. |
| **Elvárt kimenet** | A t2 tektonon rajta lesz s4 spóra. Továbbá s4 spóra kikerült az m Mushroom spores listájából. |
| **Hivatkozott szekvencia diagram** | fejlett\_spóraszórás\_true |

| **Név** | Sikertelen spóraszórás, mert nem szomszédos a céltekton. |
| --- | --- |
| **Leírás, cél** | Egy Mushroom objektum megpróbál spórát lőni egy SingleThreadTecton célpontra. Mivel nem szomszédos a Mushroom tektonja a SingleThreadTecton-al, a művelet sikertelen. |
| **Előfeltétel** | Lennie kell:   * t2:SingleThreadTecton - a cél tekton * t1:MultiThreadTecton - a tekton amin a gomba van * m:Mushroom - a gomba ami lő * s2:Spore - spóra, amit a gomba kilő   Teljesülnie kell:   * t1 és t2 nem szomszédosak: Nincsenek egymás neighbors listájában. * m rajta van t1-en:   + t1.mushroom=m   + m.position=t1 * m-nek van spórája, amit kilő, ez most s2:   + m.spores[0]=s2 * m fejletlen gombatest: m.state=Unevolved (alapértelmezett állapot létrehozás után) |
| **Hivatkozott kommunikációs diagram** | Diagram 1 |
| **Bemenet** | A Tester meghívja az m shootSpore függvényét t2-őt átadva paraméterül |
| **Forgatókönyv** | * A **shootspore** függvény törzsében ellenőrizzük, hogy nem üres a spores. Ha üres lenne visszatérne false-al. Most nem üres, benne van s2. * Az t2objektumon meghívódik a **putSpore** függvény, paraméterül megkapja az m spores listájának első elemét(s2) és az m position-ét - t2 tudja, hogy mit és honnan kap.   + A függvény elleneőrzi, hogy a paraméterül kapott position benne van-e az t2 neighbors listájában. Most nincs benne, ezért false-al tér vissza. * putSpore=false -> nem lehet t2-re spórát tenni m-nek. * shootSpore=false->sikertelen volt a spóra lövés. |
| **Elvárt kimenet** | A shootSpore false-al tér vissza, sikertelen volt a spóra lövés.  Nem került rá s2 t2-re. |
| **Hivatkozott szekvencia diagram** | fejletlen\_spóraszórás\_false |

| **Név** | Gombafonál sikeres elágaztatása tektonra. |
| --- | --- |
| **Leírás, cél** | Egy FungalThread ágat növeszt egy MultiThread felé. Amennyiben a MultThread egyik szomszédján van már azonos fajú FungalThread. A művelet sikeres. |
| **Előfeltétel** | Lennie kell:   * t3:SingleThreadTecton - róla ágaztatunk el * t4:MultiThreadTecton - céltekton * f:FungalThread - növeszteni kívánt fonál   Teljesülnie kell:   * t3 és t4 szomszédosak: Benne vannak egymás neigbors listáiban. * t3-on lévő fonalak közt van olyan, amilyet szeretnénk elágaztatni t4-re   + t3.thread=f   + t4-re f-et akarunk ágaztatni. |
| **Hivatkozott kommunikációs diagram** | Diagram 1 |
| **Bemenet** | A Tester meghívja az f branchThread függvényét t4-et átadva paraméterül. |
| **Forgatókönyv** | * A **branchThread**-en belül meghívódik a **putThread** függvény t4-en, f átadja saját magát.   + A t4 tekton meg nézi, hogy a neigbors-ben lévő tektonokon milyen fonalak vannak. A szomszédaitól le kéri a fonál tömbjüket/fonalukat, és megnézi, hogy talál-e benne f fonállal megegyezőt. Most talál, t3 szomszédján rajta van f.   + t4 felveszi a listájába f-et * putThread=true->fel lehet tenni t4-re az f fonalat. * Visszatérve a branchThread-be f felveszi a listájába t4-et és visszatér true-val. |
| **Elvárt kimenet** | * A branchThread igazzal tér vissza. * t4 felvette a listájába f-et . * f felvette a listájába t4-et. |
| **Hivatkozott szekvencia diagram** | elágazás\_true |

| **Név** | Gombafonál sikertelen elágaztatása egyfonalas tektonra, mert már van rajta. |
| --- | --- |
| **Leírás, cél** | Egy FungalThread megpróbál ágat növeszteni egy SingleThreadTecton felé. Mivel már tartalmaz egy fonalat, és csak egyet fogadhat, a művelet sikertelen. |
| **Előfeltétel** | Lennie kell:   * f2: FungalThread - növeszteni kívánt fonál * t3:AbsorbingTecton - erről a tektonról menne a terjesztés * t2:SingleThreadTecton - céltekton * f1: FungalThread - fonál ami már van t2-őn   Teljesülnie kell:   * t3 és t2 szomszédosak: Benne vannak egymás neighbors listáiban. * t3-on rajta van f2 és f1:   + t3.threads-ben benne van f2 és f1   + f2 és f1.tectons tartalmazza t3-at * t2-őn csak f1 van rajta: hasonló mint az előbbi |
| **Hivatkozott kommunikációs diagram** | Diagram 4 |
| **Bemenet** | A tesztelő meghívja az f2 branchThread függvényét t2-őt átadva paraméterül. |
| **Forgatókönyv** | * A **branchThread**-en belül meghívódik a **putThread** függvény t2-őn, f2 átadja saját magát.   + A putThread-en belül: t2 először megnézi, hogy van-e már rajta fonál. Most f1 van rajta.   + Mivel van már rajta fonál és egyfonalas tektonokon csak egy fonál ágazhat el, ezért * putThread**=**false->t2 nem tudja elraktározni f2-őt. * branchThread**=**false->f2 nem tehető t2-re. |
| **Elvárt kimenet** | A branchThread false-al tér vissza.   * t2.thread-e nem lesz f2. (továbbra is f1) * f2.tectons nem tartalmazza t2-őt. |
| **Hivatkozott szekvencia diagram** | elágazás\_false |

| **Név** | Új spóra termelődése. |
| --- | --- |
| **Leírás, cél** | Egy gombatest generál egy spórát minden második körben.  Most ezt a viselkedést a Tester idézi elő. |
| **Előfeltétel** | Lennie kell:   * m:Mushroom - termelő gomba |
| **Hivatkozott kommunikációs diagram** | Diagram 1 |
| **Bemenet** | A Tester meghívja az m generateSpore függvényét egy SpeedSpore enumot átadva paraméterként. |
| **Forgatókönyv** | * A **generateSpore** belsejében   + m létrehoz egy spórát, amelyre:     - s1:SpeedSpore - a kapott enum-nak megfelelő a Spore     - s1.thread=m.thread - s1 ctor-a megkapja m.thread-et és ezt állítja be s1.thread-nek - fonala megegyezik m fonalával   + s1-et eltárolja a spores listájában. * generateSpore=true->sikeres a spóra generálás |
| **Elvárt kimenet** | * A generateSpore igazzal tér vissza. * Létrejött s1 és ezt eltárolta m. |
| **Hivatkozott szekvencia diagram** | generate\_spore |

| **Név** | **Gombatest elpusztulása** |
| --- | --- |
| **Leírás, cél** | Egy Mushroom objektum kilövi a 10. spóráját is, ekkor a gombatestnek el kell pusztulnia |
| **Előfeltétel** | Lennie kell:   * m: gombatest, amely a 9. spóráját is már elszórta, spores tömbjében lennie kell még legalább egy spórának, amit még el tud szórni * t1 : erre a tektonra szórja az m gombatest a spóráját * t2: ezen a tektonon helyezkedik el a gombatest * f : gombafonál, amely ugyanabból a gombafajból van, mint az m gombatest * s: spóra, amelyet az m gombatest elszórhat |
| **Hivatkozott kommunikációs diagram** | Diagram 3 |
| **Bemenet** | A Tester meghívja az m gombatest shootSpore függvényét t1-et átadva paraméterül |
| **Forgatókönyv** | A shootSpore metódushívás hatására a  kilőjük az s spórát a t1 tektonra meghívva a tekton putSpore metódusát, paraméterül küldjük az s spórát, és azt a tektont is, amelyiken a gombatest található, azaz a t2-t.  Ezután növeljük a kilőtt spórák számát 1-el ha sikeres a kilövés, leellenőrizzük, hogy kilőttük-e a 10. spórát  Mivel kilőttük a 10. spórát törölnünk kell a gombatestet, ezt először a tektonról vesszük le a removeMushroom() metódushívás segítségével, majd a gombafajhoz tartózó gombafonálnál töröljük azokat a részeket a deleteUnnecessaryThreads() metódushívás segítségével, amelyek nem kapcsolódnak azonos fajhoz tartozó gombatesthez.  Egy külső tesztelő objektum lekérdi, hogy hány spórát lőttünk ki eddig a getshootedSporesCount() segítségével,  ha ez megegyezik 10-el, akkor meghívja a gombatest destruktorát. |
| **Elvárt kimenet** | A t2 tektonon az m gombatestnek el kell tűnnie, illetve meg kell szűnnie létezni. A t1 tektonon rajta kell lennie az s spórának. |
| **Hivatkozott szekvencia diagram** | gombatest\_elpusztulása |

| **Név** | **Gombatesthez nem kapcsolódó fonálrész eltávolítása** |
| --- | --- |
| **Leírás, cél** | Gombatest megszűnése, fonál felszívódása, tekton kettétörése és rovar fonálvágása során keletkezhetnek olyan fonálrészek, melyek nincsenek már kapcsolatban ugyanolyan gombafajból származó gombatesttel. Ezeket törölni kell. |
| **Előfeltétel** | Lennie kell:  t1, t2, t3, t4, t5, t6 tektonok  f gombafonál, amely elágazik a t1, t2, t3, t5, t6 tektonokon  m gombatest, amely a t2 tektonon van |
| **Hivatkozott kommunikációs diagram** | Diagram 2 |
| **Bemenet** | A Tester meghívja a gombafonál deleteUnnecessaryThreads() metódusát |
| **Forgatókönyv** | A metódushívás után sorban végigmegyünk azokon a tektonokon, amelyeken a gombafonál elágazik, és megkérdezzük, ha van-e rajtuk gombatest. Ha nincs megnézzük a sorban következő tektonra is, ha van, és ez a gombatest ugyanahhoz a gombafajhoz származik, akkor megkeressük azt a fonálrészt, amely ezzel a gombatesttel összeköttetésben van.  Ez a következő képpen történik: meghívjuk a getThreadSection(f: FungalThread) metódust arra a tektonra, amelyen a gombatest rajta van, ez visszaadja azokat a tektonokat, amelyek a gombatest tektonjával szomszédosak, és elágazik rajtuk az f gombafonál. Ezután meghívjuk ezt a metódust az így kapott tektonokra is, és ezt addig csináljuk, amig találunk új tektont. Ezeket a tektonokat behelyezzük egy listába.  Ha már nem találunk új tektonokat, akkor tovább megyünk azokon a tektonokon, amelyen a gombafonál elágazik, és megkérdezzük, ha van-e rajtuk gombatest.  Mikor ezt befejeztük, azokra a tektonokra, amelyek nem kerültek be az algoritmus által elkészített listába, de benne vannak a gombafonál által tárolt tectons listába, meghívjuk a removeThread(f) metódusukat, ezzel levesszük róluk a fonalat, majd kivesszük őket a tectons listából is, ezzel biztosítva, hogy a fonál nem lesz rajtuk elágazva. |
| **Elvárt kimenet** | A t5 és t6 tektonokon nincs az f gombafonál elágazva, mivel ez egy olyan fonálrész, amely nem kapcsolódik egy ugyanolyan fajból származó gombatesthez.  A metódushívás true értékkel térjen vissza. |
| **Hivatkozott szekvencia diagram** | gombatesthez\_nem\_kapcsolódó\_fonálrész\_eltávolítása |

| **Név** | **Tekton kettétörése** |
| --- | --- |
| **Leírás, cél** | A t3 tektonnak meghívjuk a break() metódusát, ennek hatására két új tekton kell keletkezzen. |
| **Előfeltétel** | Lennie kell:  t1, t2, t3, t4, t5 tektonok, a t3 tekton szomszédos az összes többivel  m: gombatest, amely a t1 tektonon található  f: gombafonál, amely a t1, t2, t3 tektonokon ágazik el  s: spóra, amely a t3 tektonon található  i: rovar, amely a t3 tetkonon található |
| **Hivatkozott kommunikációs diagram** | Diagram 1 |
| **Bemenet** | A t3 tektonnak a Tester meghívja a break() metódusát |
| **Forgatókönyv** | A t3 tekton break() metódusa először két új tektont hoz létre. Ezeknek először be kell állítani a szomszédait. Mivel a szomszédok sorban következnek a neighbors listában, és a tekton a listában tárolt első és az n/2 (felkerekített) szomszédjánál tört el, így tudni fogjuk, hogy a két új tekton melyik tektonokkal lesz szomszédos. Ezután a t3 tektonnal szomszédos tektonoknak is beállítjuk a szomszédsági listáját, valamint kivesszük belőle a t3 tektont.  Ha van a t3 tektonon spóra, akkor ezt töröljük.  Ha van a tektonon rovar, ezt ráhelyezzük az elsőnek létrehozott tektonra, és frissítjük a rovar pozícióját.  Ha a t3 tektonon elágazott fonál, akkor erről levesszük a fonalat, majd töröljük azokat a fonálrészeket, amelyek nincsenek összeköttetésben ugyanolyan gombafajból származó tektonnal.  Végül egy külső kontroller törli a tectont, amely kettétört. |
| **Elvárt kimenet** | Létrejött a t6 és t7 tekton, illetve a t3 tekton megszűnt létezni, a bogár a t6 tektonra került, a spóra, ami a t3 tektonon volt megszűnt létezni, a keletkezett tektonoknak és a megmaradt tektonoknak szomszédsági listájuk helyesen beállítódott.  A break() metódus true-val tért vissza |
| **Hivatkozott szekvencia diagram** | tekton\_kettétörése |

| **Név** | **Tekton kettétörése, ha a tektonon gombatest van** |
| --- | --- |
| **Leírás, cél** | Egy olyan tektonnak hívjuk meg a break() metódusát, amelyen gombatest található, ekkor a tekton nem törhet ketté |
| **Előfeltétel** | Lennie kell:  t1: tekton  m: gombatest, amely a t1 tektonon van |
| **Hivatkozott kommunikációs diagram** | Diagram 1 |
| **Bemenet** | A t1 tektonnak a Tester meghívja a break() metódusát |
| **Forgatókönyv** | A metódushívás után nem történik semmi, mivel a tektonon gombatest van. |
| **Elvárt kimenet** | A break() metódus false-val tér vissza |
| **Hivatkozott szekvencia diagram** | tekton\_kettétörése\_ha\_a\_tektonon\_gombatest\_van |

| **Név** | **Fonál felszívódása tektonról** |
| --- | --- |
| **Leírás, cél** | Egy AbsorbingTecton objektumnak meghívtuk az absorb metódusát, ezáltal a rajta levő fonalak fell kell szívódjanak |
| **Előfeltétel** | Lennie kell:  t3: AbsorbingTecton  f1, f2 gombafonál, amelyek elágaznak a t3 tektonon |
| **Hivatkozott kommunikációs diagram** | Diagram 4 |
| **Bemenet** | A Tester meghívja a t3 tekton absorb() metódusát |
| **Forgatókönyv** | A t3 tektonon két gombafonál ágazik el, az f1 és az f2. Mivel a t3 tektonról felszívódnak a fonalak az absorb() metódushívás során törölni kell, az f1 és az f2 fonalat a t3 tektonról, azaz, a két fonál ennek során már nem lesz elágazva a tektonon, ezért meghívódik először az f1 removeTecton(t3) metódusa, majd az f2 removeTecton(t3) metódusa, amelyek kitörlik a t3 tektont a gombafonalak tecons listájából. Ezután a deleteUnnecessaryThreads() függvények segítségével kitöröljük azokat a gombafonál részeket, amelyek már nincsenek összeköttetésben egy ugyanolyan fajból származó gombatesttel. |
| **Elvárt kimenet** | A t3 tektonon nincsenek gombafonalak elágazva, azaz a t3 threads listája üres, és az f1 és f2 tectons listája sem tartalmazza a t3-at.  Az absorb() metódus true értékkel tér vissza. |
| **Hivatkozott szekvencia diagram** | fonál\_felszívódása\_tektonról |

| **Név** | **Gombafonál elvágása** |
| --- | --- |
| **Leírás, cél** | A rovar elvágja a gombafonalat a saját tektonja, és a kiválasztott tekton közötti résen, ezáltal az összes olyan gombafonalat, amely a résen átmegy a kiválasztott tektonról töröljük. |
| **Előfeltétel** | Lennie kell:  t2, t3 tekton  f1, f2 gombafonál  f1 és f2 gombafonál is elágazik a t3 tektonon, azonban t2 tektonon csak az f1 ágazik el  i: rovar, amely a t2 tektonon található |
| **Hivatkozott kommunikációs diagram** | Diagram 4 |
| **Bemenet** | A Tester az i rovarnak meghívja a cut metódusát paraméterként a t2 tektont adva át |
| **Forgatókönyv** | A cut(t2) metódushívás után leellenőrizzük, hogy a paraméterül kapott t2 tekton szomszédos-e a t3 tektonnal, azaz azzal a tektonnal, amelyen a rovar rajta van.  Ha ez teljesül, akkor mindkét tektonról lekérjük a gombafonalakat mindkettőről külön-külön. Ha találunk olyat, amely mindkettőn elágazik, akkor ezt levesszük a t2 tektonról, majd a t2 tektont kivesszük a fonál listájából, végül pedig a deleteUnnecessaryThreads() függvényhívással kitöröljük azokat a fonálrészeket, amelyek nincsenek összeköttetésben azonos fajból származó tektonnal. |
| **Elvárt kimenet** | A t2 tektonon már nincs elágazva az f1 gombafonál.  A rovar cut metódusa true értékkel tér vissza. |
| **Hivatkozott szekvencia diagram** | gombafonál\_elvágása |

| **Név** | **Rovar lehelyezése tektonra sikeres** |
| --- | --- |
| **Leírás, cél** | A játék elején egy rovarász első lépése az lesz, hogy letegye a rovarját, amit irányítani fog a játék során egy tektonra. |
| **Előfeltétel** | Lennie kell :  t1: tecton |
| **Hivatkozott kommunikációs diagram** | Diagram 1 |
| **Bemenet** | Tester meghívja a t1 tekton putFirstInsect() metódusát. |
| **Forgatókönyv** | A putFirstInstect() metódushívás után a tekton leellenőrzi magán, hogy nem foglalt, azaz nincs rajta már rovar. Ez után létrehoz egy új Rovar objektumot, aminek beállítja a pozícióját magára, illetve beállítja magán hogy az a rovar rajta van. Végül true értékkel tér vissza a hívó felé, jelezve a művelet sikerességét. |
| **Elvárt kimenet** | A t1 tektonon megjelenik az új rovar, megfelelő pozícióval. |
| **Hivatkozott szekvencia diagram** | rovarlerakás\_sikeres |

| **Név** | Rovar lehelyezése tektonra sikertelen |
| --- | --- |
| **Leírás, cél** | A játék elején egy rovarász első lépése az lesz, hogy letegye a rovarját, amit irányítani fog a játék során egy tektonra. |
| **Előfeltétel** | Lennie kell :  t2: tecton |
| **Hivatkozott kommunikációs diagram** | Diagram 1 |
| **Bemenet** | Tester meghívja a t1 tekton putFirstInsect() metódusát. |
| **Forgatókönyv** | A putFirstInsect() metódushívás után a tekton önellenőrzése során arra jut, hogy már foglalt, azaz az Insect mezője nem üres. Ekkor fasle értékker tér vissza a hívó felé jelezve azt, hogy a művelet sikertelen. |
| **Elvárt kimenet** | Nem történik változás. |
| **Hivatkozott szekvencia diagram** | rovarlerakás\_sikertelen |

| **Név** | Rovar mozgása és spóraevése |
| --- | --- |
| **Leírás, cél** | A játék során egy rovarász lépései közé tartozik a mozgás lépése, ekkor a rovarász választ egy tektont amire mozogni szeretne, és a rovara átmegy oda, ha ez lehetséges.  Továbbá a már átment rovar ekkor az új tekton spórái közül megesz egyet és megkapja annak a hatását. |
| **Előfeltétel** | Kell legyen:  t2: tecton  t3: tecton  s: speedSpore  t: thread  i: insect  i Rovar t2 tektonon, s gyórsítóspóra t3 tektonon, köztük f gombafonál. |
| **Hivatkozott kommunikációs diagram** | Diagram 1 |
| **Bemenet** | Tester meghívja a Rovar move(t3) metódusát. |
| **Forgatókönyv** | A move(t3) függvényhívás után a rovar meghívja t3 putInsect metódusát. A t3 megnézi, hogy t2 szomszédja és létezik -e köztük spóra. t2 igazzal való visszatérése után t3 meghívja a removeInsect metódust t2-n és beállítja magának i rovart. Igazzal tér vissza a rovar felé.  Ekkor a rovar lekérdezi t3 spóráit és a 0. elemén, azaz s-en meghívja az applyEffect metódust, aminek hatására a spóra meghívja a rovar setState metódusát és beállítódik a hatás.  A rovar igazzal tér vissza a hívó felé. |
| **Elvárt kimenet** | i Rovar t3 tektonon, gyorsított hatás a state mezőben, s már nem eleme t3 tekon spóráinak. |
| **Hivatkozott szekvencia diagram** | rovar\_mozgas\_sporaeves\_sikertelen |

| **Név** | Rovar mozgása sikertelen |
| --- | --- |
| **Leírás, cél** | A játék során egy rovarász lépései közé tartozik a mozgás lépése, ekkor a rovarász választ egy tektont amire mozogni szeretne, és a rovara átmegy oda, ha ez lehetséges.  Ebben az esetben nem lesz lehetséges fonal hiánya miatt. |
| **Előfeltétel** | Kell lennie:  t3: tecton  t4: tecton  i: Insect  i Rovar t4 tektonon, t3 szomszédos tekton t4 -gyel |
| **Hivatkozott kommunikációs diagram** | Diagram 1 |
| **Bemenet** | Tester meghívja a Rovar move(t3) metódusát. |
| **Forgatókönyv** | A move(t3) függvényhívás után a rovar meghívja t3 putInsect metódusát. t3 megnézi, hogy szomszédja -e t4 illetve, hogy van -e köztük fonal. Ebben az esetben a két tekton Threads mezőiben nincs azonos elem, ezért false értékkel tér vissza i Rovar felé, majd az szintén false értékkel tér vissza a hívó felé. |
| **Elvárt kimenet** | Nincs változás |
| **Hivatkozott szekvencia diagram** | rovar\_mozgas\_sikertelen |

| **Név** | Gombafonal elágazása olyan tektonra, ahol van spóra |
| --- | --- |
| **Leírás, cél** | Amikor a gombász egy olyan tektonra szeretné fonalát elágaztatni, ahol található spóra, akkor még egy további tektonra is onnan el tudja ágaztatni a fonalát. |
| **Előfeltétel** | Kell lennie:  t1: tekton  t2: tekton  t3: tekton  f: gombafonál |
| **Hivatkozott kommunikációs diagram** | Diagram 3 |
| **Bemenet** | Tester hívja a fonál branchThread(t1) metódusát |
| **Forgatókönyv** | A t1 tekton mivel szomszédos a t2 tektonnal, amin megtalálható az f gombafonál már, így sikeresen elágazik f gombafonál t1 tektonra. Majd lekérdezzük t1 tekton spóráit, és ha található rajta, akkor a branchThread() metódus meghívható még egy alkalommal. |
| **Elvárt kimenet** | Sikeres fonal elágazás, azaz logikai igaz visszatérési érték. |
| **Hivatkozott szekvencia diagram** | Gombafonal elágazása |

* 1. ***A szkeleton kezelői felületének terve, dialógusok***

**5.2.1 Statikus naplózó**

A kezelői felület a következő módon épül fel:

* Ahány use-case, annyi menüpont lesz a program indításakor
* A felhasználó kiválaszthatja a menüpontok közül, hogy melyik teszt-szekvencia fusson.
* A kimenet a konzolon egy általunk definiált nyelv alapján íródik ki, amely bemutatja a szekvencia hívásait, a hívási paramétereit és a szekvenciában résztvevő függvények visszatérési értékeit.
* Ez a nyelv PlantUML alapú.

A nyelv felépítése:

* **->** : metódushívás
* **- - >**: visszatérési érték
* A nyíl bal oldalán áll az az objektum, amelyik a hívó
* A nyíl jobb oldalán áll az az objektum, amelyiken a függvény meghívódik
* **:** után függvény neve metódushívás esetén (->)
* **:** után visszatérési érték visszatérés esetén (- - >)
* **participant** : a szekvencia során résztvevő objektum
  + **“ “** között az objetum neve és típusa
  + **as** után referencia erre az objektumra

**Példa**:

**Gombafonal elágazása olyan tektonra, ahol van spóra** use-case kiválasztása esetén a a konzolos kimenet:

participant "f :FungalThread" as f

participant "t1 :MultiThreadTecton" as t1

participant "t2 :MultiThreadTecton" as t2

participant "t3 :MultiThreadTecton" as t3

-> f : branchThread(t1)

f -> t1 : putThread(f)

t1 -> t2 : getThreads()

t2 --> t1 : f

t1 --> f : true

-> t1 : getSpores()

<-- t1 : spores

-> f : branchThread(t3)

f -> t3 : putThread(f)

t3 -> t1 : getThread()

t1 --> t3 : f

t3 --> f : true

<-- f : true

**5.2.2 Kérdező**

Abban az esetben, ha az adott use-case teszt esetben van állapot lekérdezés a Tester felé, amelyre a válasz Tester aktortól érkezik, akkor a program először a konzolon felteszi a kérdést (az use-case-től függő formában), és kéri a választ. Ezután írja ki a konzolra statikus naplózó a szekvencia lefutását a fent leírt formában.

**Példa:**

**(Fejletlen) gombatest spóra szórása olyan tektonra, amelyre tud** esetén a konzolos kimenet:

[Gomba állapota (e [evolved] / u [unevolved)?]

[Válasz] : u

participant "m: Mushroom" as m

participant "Tester" as t

participant "t3 :SingleThreadTecton" as t3

-> m : shootSpore(t3)

m -> t : getMushroomState()

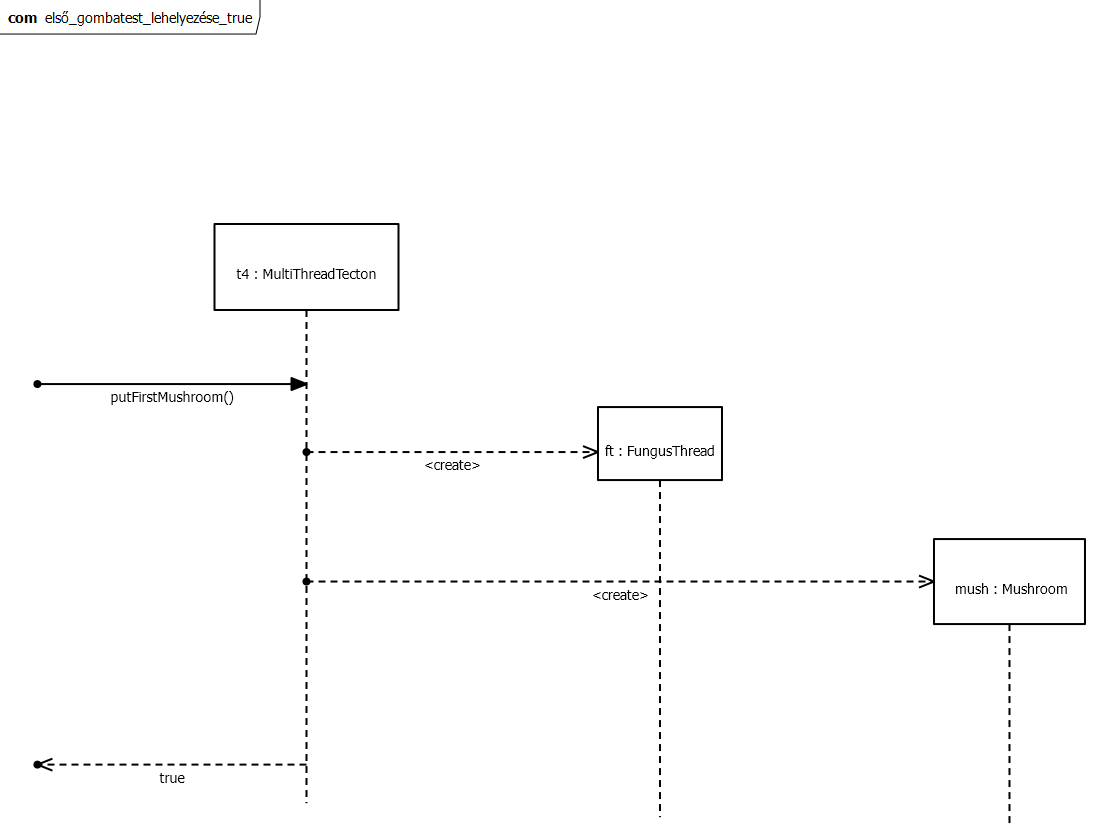
t --> m : unevolved

m -> t3 : putSpore(spores[0], position)

t3 --> m : true

<-- m : true

* 1. ***Szekvencia diagramok a belső működésre***
     1. ***első\_gombatest\_lehelyezése\_true***

******

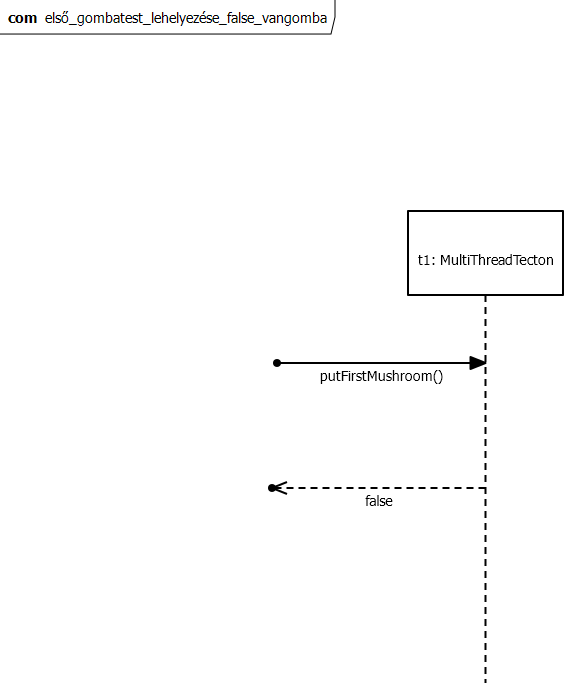
Tesztesetek amik felhasználják:

* Első gombatest lehelyezése olyan tektonra, amelyre le lehet helyezni
  + 1. **első\_gombatest\_lehelyezése\_false\_absorb**

****

Tesztesetek amik felhasználják:

* Első gombatest lehelyezése egy tektonra, amelyre nem lehet lehelyezni (AbsorbingTecton)
  + 1. **első\_gombatest\_lehelyezése\_false\_vangomba**

****

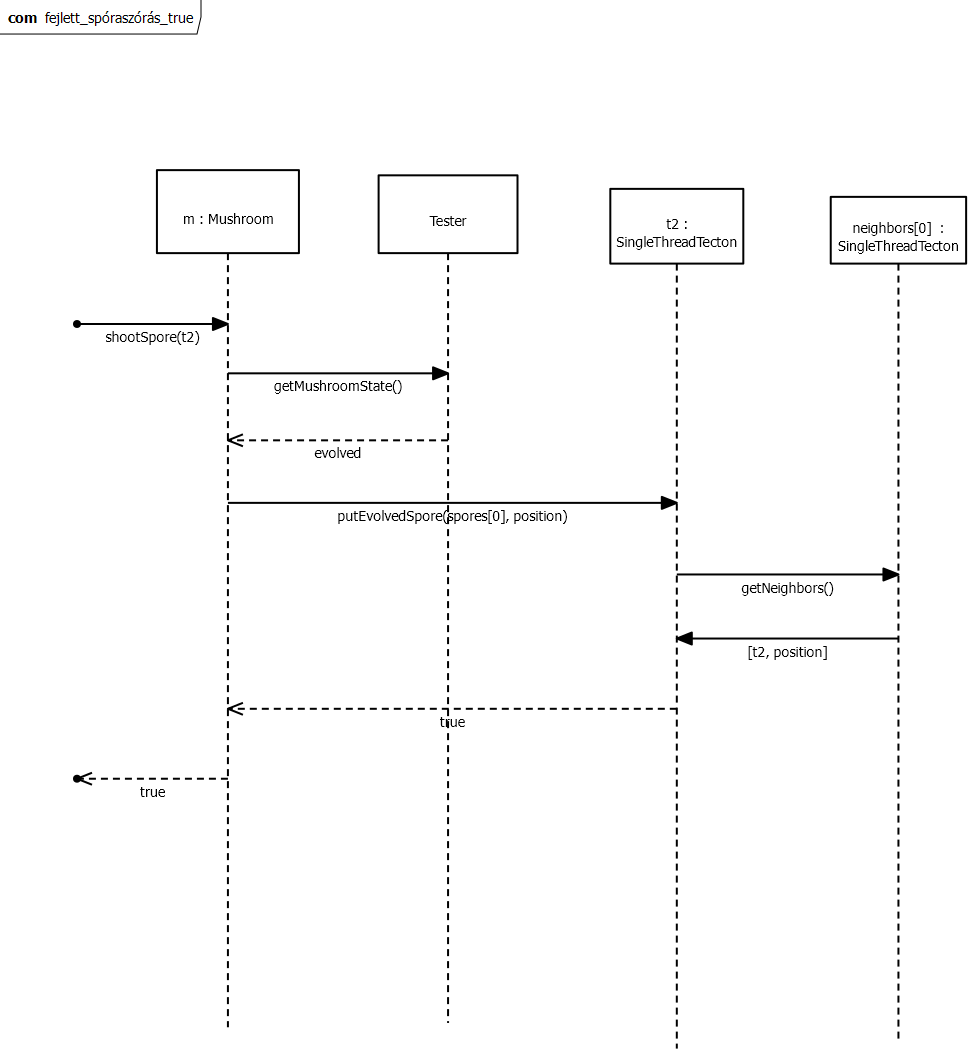
Tesztesetek amik felhasználják:

* Első gombatest lehelyezése egy tektonra, amelyre nem lehet lehelyezni (van gombatest a tektonon)
  + 1. ***fejletlen\_spóraszórás\_true***

******

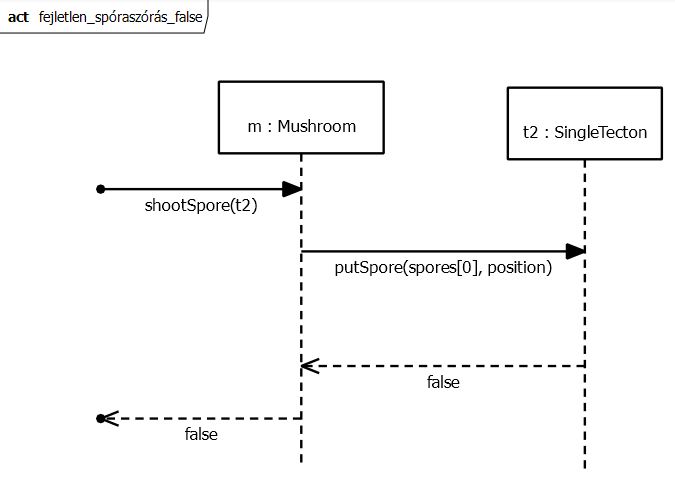
Tesztesetek amik felhasználják:

* Fejletlen gombatest spóra szórása olyan tektonra, amelyre tud.
  + 1. ***fejlett\_spóraszórás\_true***

******

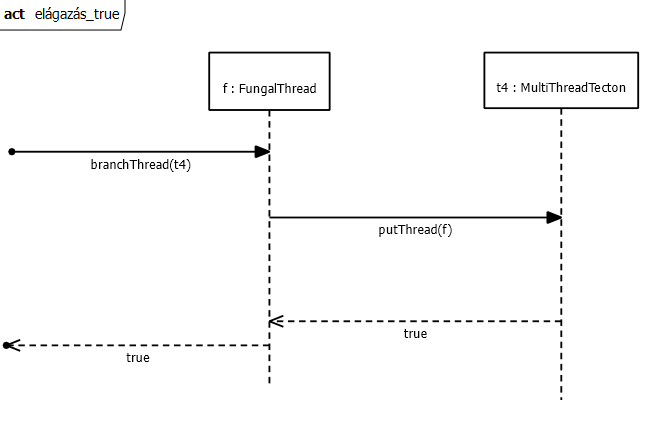
Tesztesetek amik felhasználják:

* Fejlett gombatest spóra szórása olyan tektonra, amelyre tud.
  + 1. ***fejletlen\_spóraszórás\_false***

**

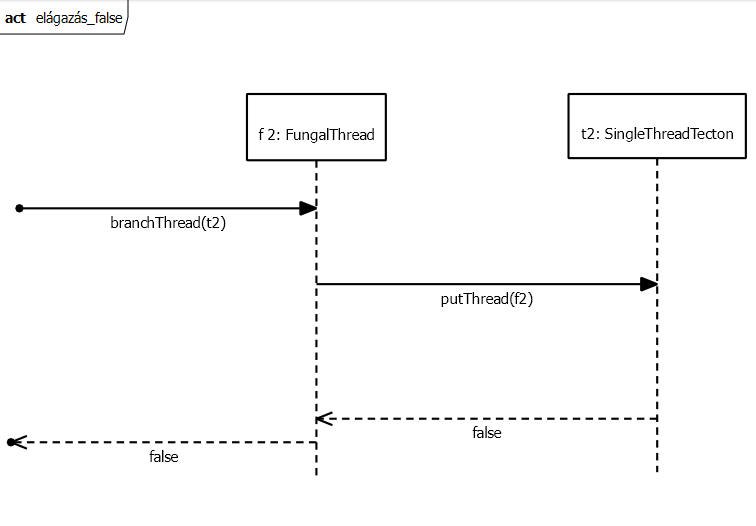
Tesztesetek amik felhasználják:

* Sikertelen spóraszórás, mert nem szomszédos a céltekton.
  + 1. ***elágazás\_true***

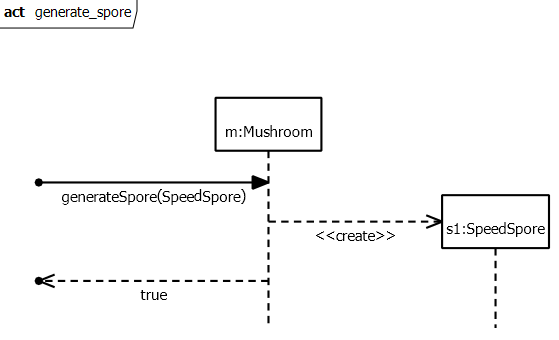


Tesztesetek amik felhasználják:

* Gombafonál sikeres elágaztatása tektonra.
  + 1. ***elágazás\_false***



Tesztesetek amik felhasználják:

* Gombafonál sikertelen elágaztatása egyfonalas tektonra, mert már van rajta.
  + 1. ***generate\_spore***

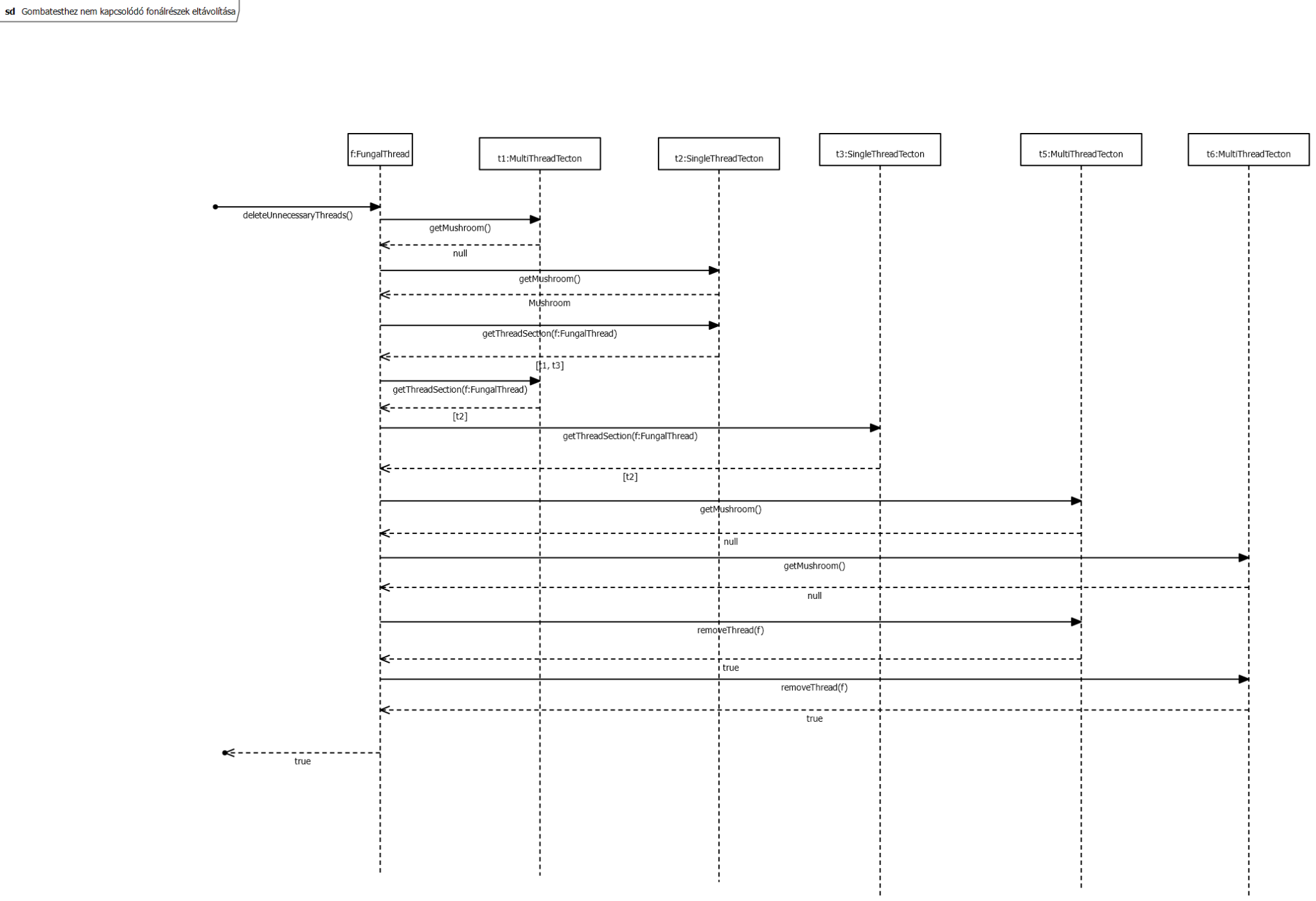
Tesztesetek amik felhasználják:

* Új spóra termelődése.
  + 1. ***gombatest\_elpusztulása***



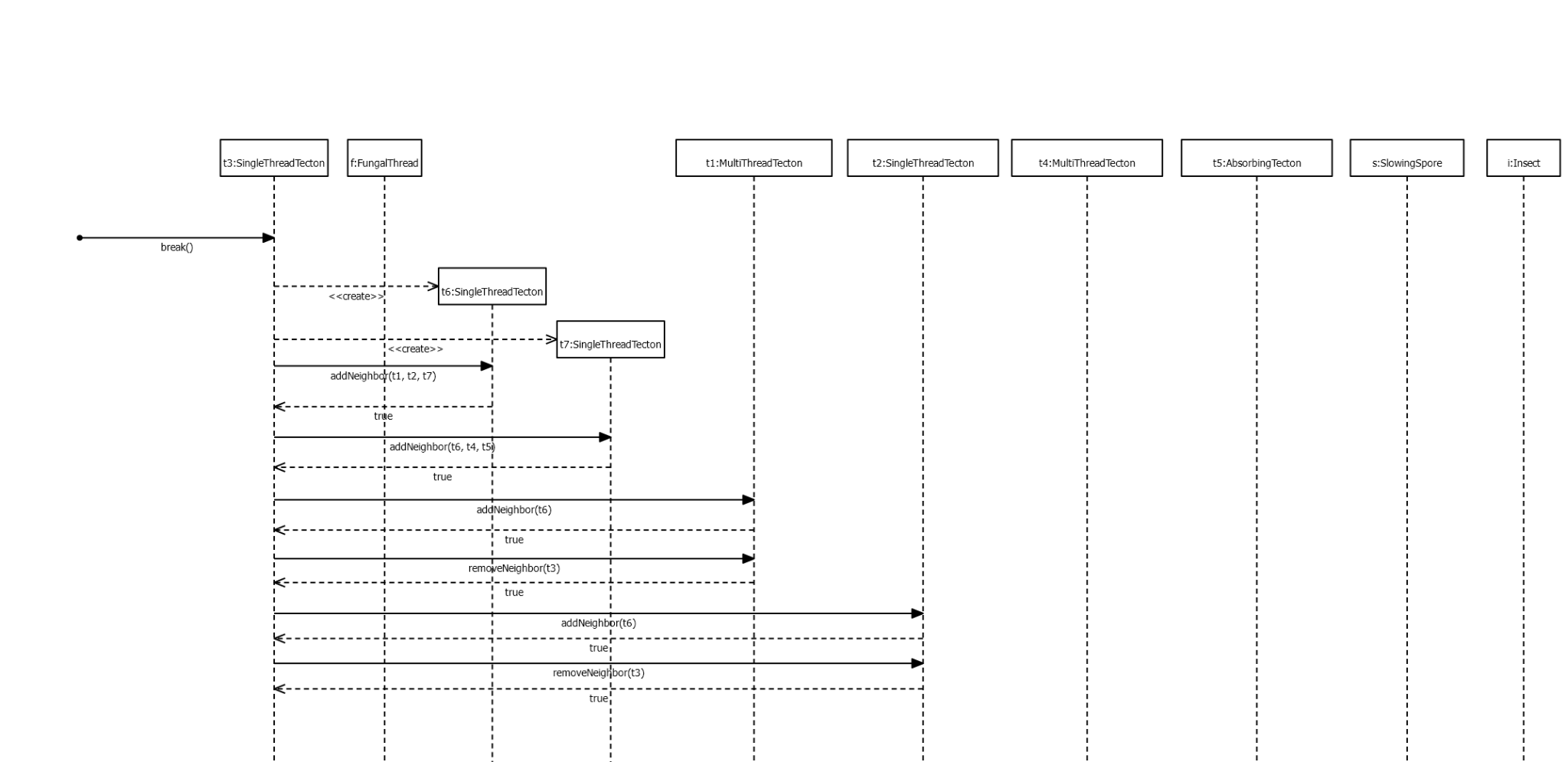
Tesztesetek, amik felhasználják:

* Gombatest elpusztulása
  + 1. ***gombatesthez\_nem\_kapcsolódó\_fonálrész\_eltávolítása***

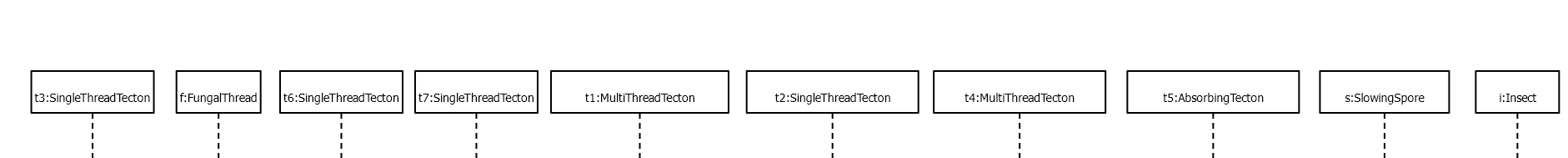
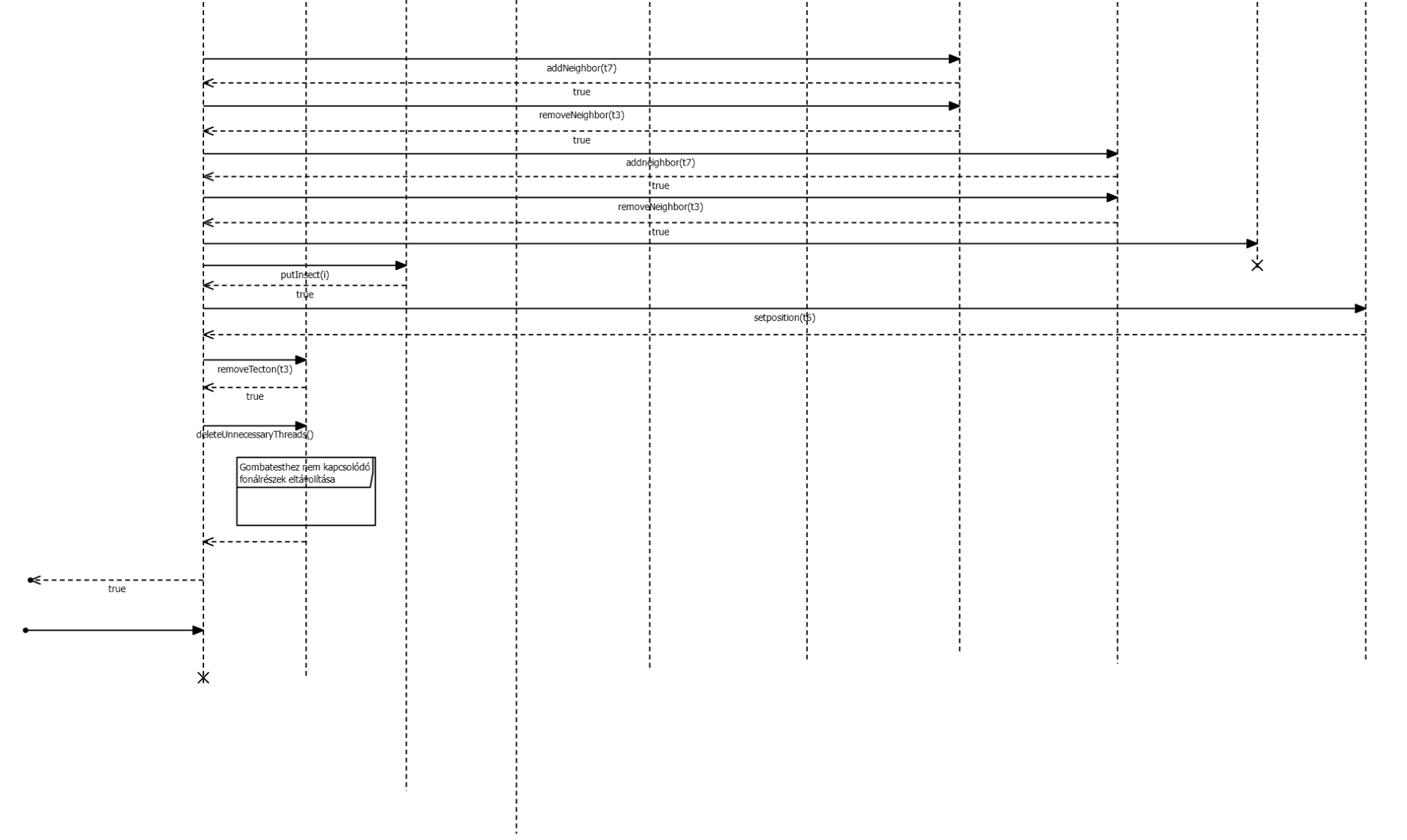


Tesztesetek, amik felhasználják:

* Gombatesthez nem kapcsolódó fonálrész eltávolítása
  + 1. ***tekton\_kettétörése***

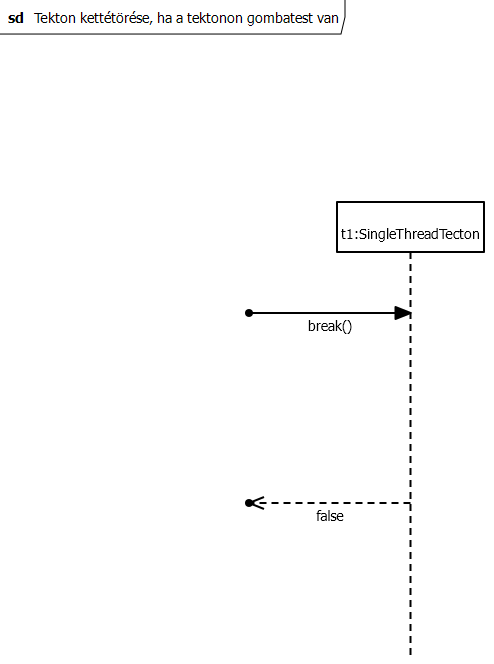


**Folytatás:**



Tesztesetek, amik felhasználják:

* Tekton kettétörése
  + 1. ***tekton\_kettétörése\_ha\_a\_tektonon\_gombatest\_van***



Tesztesetek, amik felhasználják:

* Tekton kettétörése, ha a tektonon gombatest van
  + 1. ***fonál\_felszívódása\_tektonról***



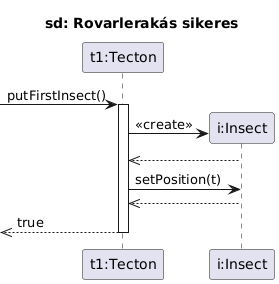
Tesztesetek, amik felhasználják:

* Fonál felszívódása tektonról
  + 1. ***gombafonál\_elvágása***

Tesztesetek, amik felhasználják:

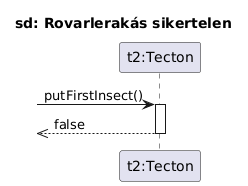
* Gombafonál elvágása

***5.3.16 rovarlerakás\_sikeres***

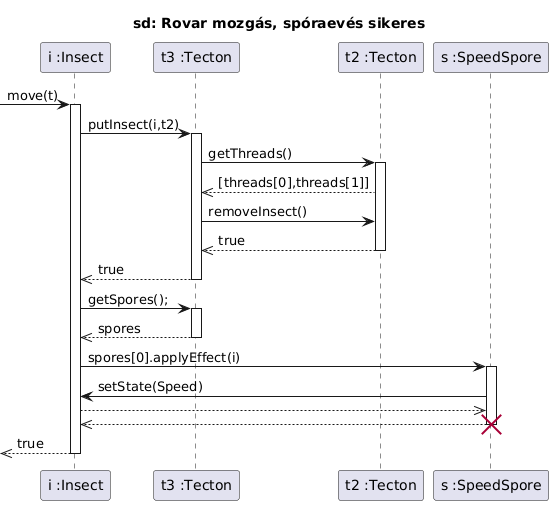
Tesztesetek, amik felhasználják:

* Rovar lehelyezése tektonra sikeres

***5.3.17 rovarlerakás\_sikertelen***

Tesztesetek, amik felhasználják:

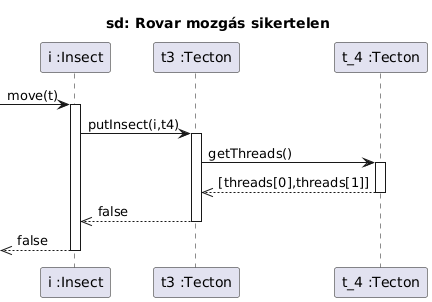
* Rovar lehelyezése tektonra sikertelen

***5.3.18 rovar\_mozgas\_sporaeves\_sikeres***

Tesztesetek, amik felhasználják:

* Rovar mozgása és spóraevése

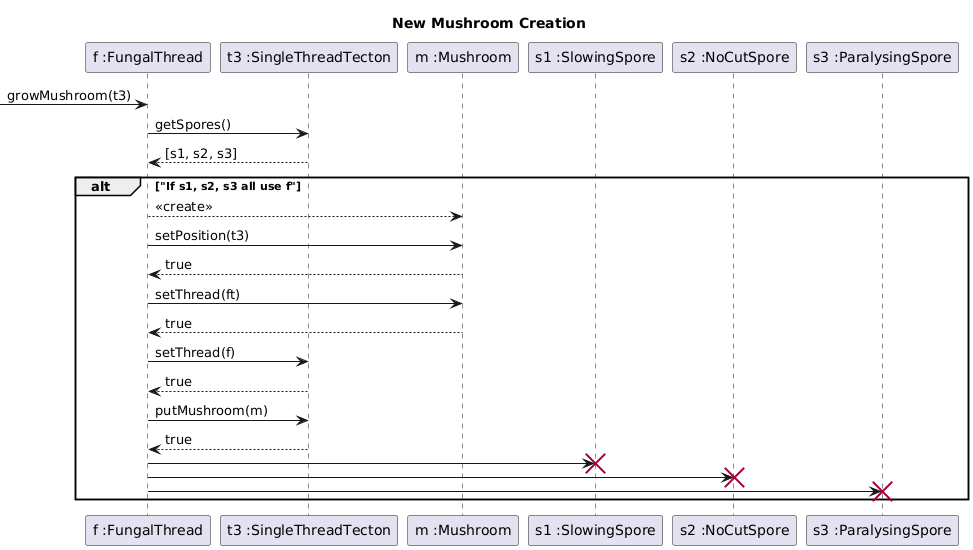
***5.3.19 rovar\_mozgas\_sikertelen***



Tesztesetek, amik felhasználják:

* Rovar mozgása sikertelen

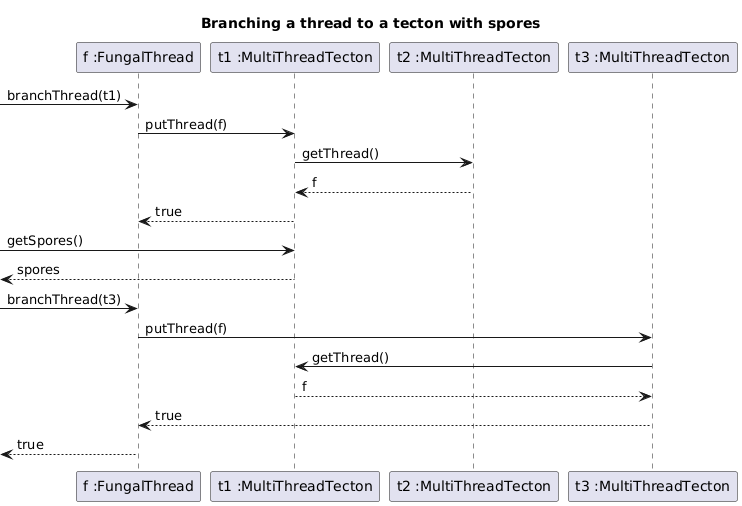
***5.3.20 Új gombatest lerakása***

******

Tesztesetek, amik felhasználják:

* Új gombatest lerakása

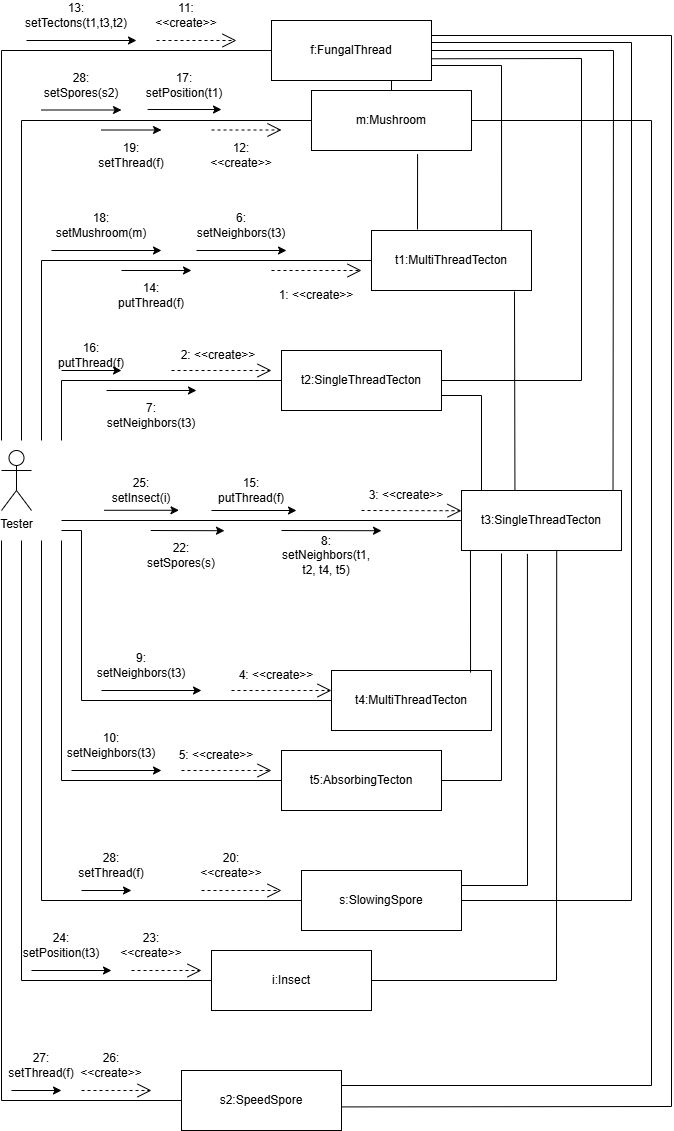
***5.3.21 Gombafonal elágazása***

******

Tesztesetek, amik felhasználják:

* Gombafonal elágazása olyan tektonra, ahol van spóra
  1. ***Kommunikációs diagramok***

**Diagram 1**

****

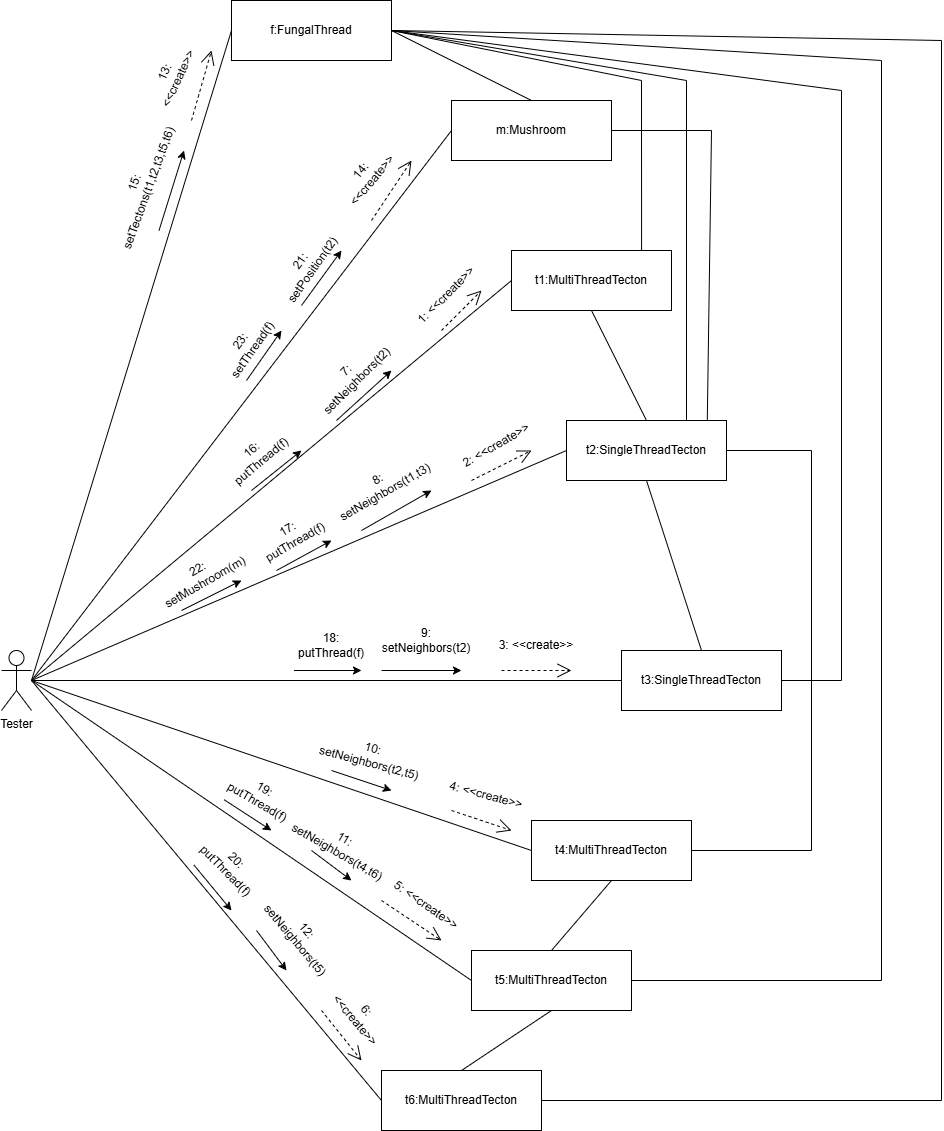
A Tester létrehozze a t1, t2, t3 ,t4 és t5 tektonokat. A t3 tekton szomszédos az összes többivel, a t1, t2, t4, t5 szomszédos a t3-al. A szomszédságokat a setNeighbors függvény segítségével állítja be.

Ezután a Tester létrehoz egy gombafonalat, majd egy gombát, beállítja, hogy a létrehozott f gombafonál a t1, t2 és t3 tektonokon ágazzon el, a gombát a t1 tektonra helyezi le, a gombának az f gombafonalat adja át, azaz ez lesz a gomba gombafonala.

A Tester létrehoz egy s gyorsítóspórát, melyet a t3 tektonra helyez le, és a gombafonál attribútumát beállítja f-re, azaz csak az f gombafonál tudja majd felhasználni újabb gombatest növesztésére.

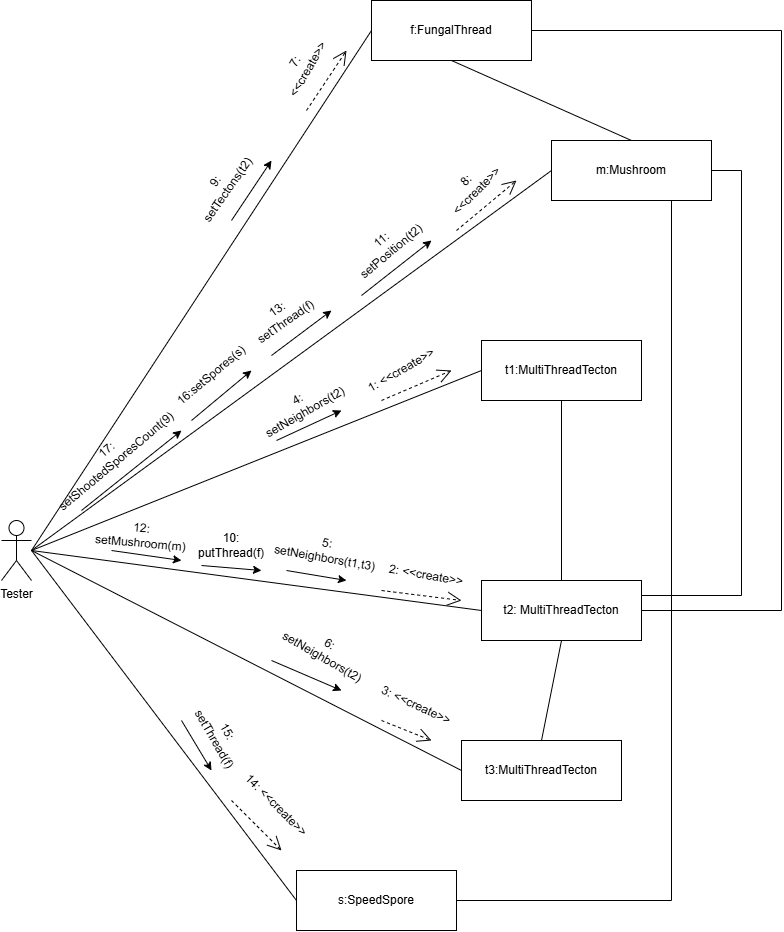
Ezután a Tester létrehozza az i rovart, ezt a t2 tektonra helyezi le.

A Tester létrehoz egy s2 gyorsítóspórát, melynek a gombafonál attribútumát beállítja f-re, azaz csak az f gombafonál tudja majd felhasználni újabb gombatest növesztésére. Ezt a spórát odaadja az m gombatestnek, azaz beleteszi a spores tömbjébe.

**Diagram 2**

A Tester létrehozza a t1, t2, t3, t4, t5 és t6 tektonokat. A t1 tekton szomszédos a t2-vel, a t2 szomszédos a t1-el és t3-al, a t3 szomszédos a t2-vel, a t4 szomszédos a t2-vel és t5-el, a t5 szomszédos a t4-el és t6-al, a t6 szomszédos a t5-el.

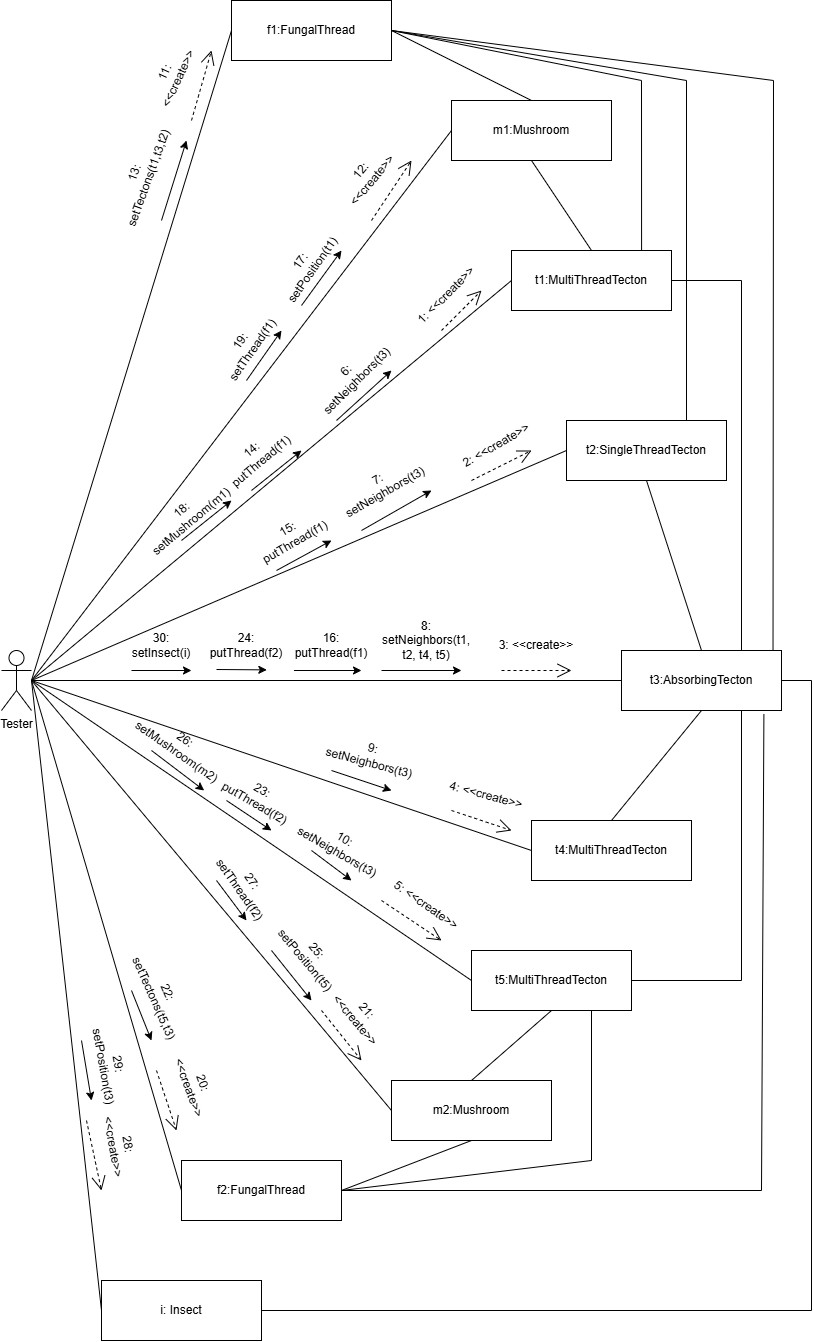
Ezután létrehoz egy gombafonalat, meg egy gombatestet. A gombafonál a t1, t2, t3, t5 és t6 tektonokon ágazik el, a gombatestet a t2 tektonra helyezi le. A gombának az f gombafonalat adja át, azaz ez lesz a gomba gombafonala.

**Diagram 3**

A Tester létrehozza a t1,t2 és t3 tektonokat, a t1 szomszédos a t2-vel, a t2 szomszédos a t1-el és t3-al, a t3 szomszédos a t2-vel.

Ezután létrehozza az m gombatestet és f gombafonalat. A gombafonalat a t2 tektonon ágaztatja el, a gombatestet is a t2 tektonra helyezi.

A Tester létrehoz egy s gyorsítóspórát, melynek a gombafonál attribútumát beállítja f-re, azaz csak az f gombafonál tudja majd felhasználni újabb gombatest növesztésére. Ezt a spórát odaadja az m gombatestnek, azaz beleteszi a spores tömbjébe. A gombatest shootedSporesCount attribútumát 9-re állítja, azaz a gombatest már kilőtt 9 spórát.

**Diagram 4**

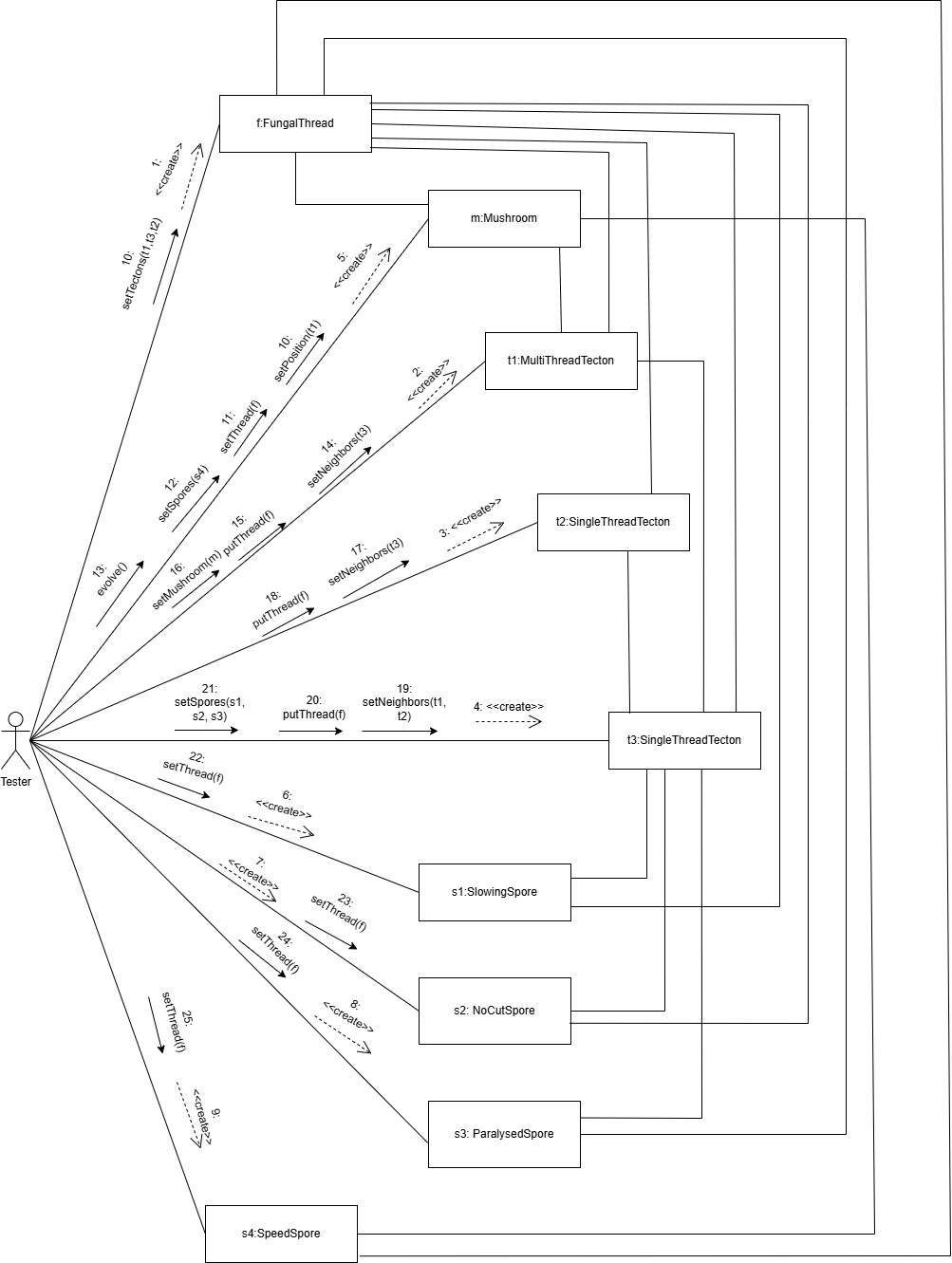
A Tester létrehozze a t1, t2, t3 ,t4 és t5 tektonokat. A t3 tekton szomszédos az összes többivel, a t1, t2, t4, t5 szomszédos a t3-al. A szomszédságokat a setNeighbors függvény segítségével állítja be.

Ezután a Tester létrehoz egy f1 gombafonalat, majd egy m1 gombát, beállítja, hogy a létrehozott f1 gombafonál a t1, t2 és t3 tektonokon ágazzon el, a gombát a t1 tektonra helyezi le, a gombának az f gombafonalat adja át, azaz ez lesz a gomba gombafonala.

Létrehoz egy f2 gombafonalat, majd egy m2 gombát is, beállítja, hogy a létrehozott f2 gombafonál a t5 és t4 tektonokon ágazzon el, a gombát a t5 tektonra helyezi le, a gombának az f2 gombafonalat adja át, azaz ez lesz a gomba gombafonala.

A Tester létrehoz egy i rovart, amelyet a t3 tektonra helyez le.

**Diagram 5**

****

A Tester létrehozza az f fonalat, az m gombatestet, a t1 MultiThreadTecton-t, a t2 SingleThreadTecton-t, a t3 SingleThreadTecton-t, az s1 SlowingSpore-t, az s2 NoCutSpore-t, az s3 ParalysedSpore-t és végül az s4 SpeedSpore-t.

Az f fonálon beállítja a Tester, hogy a t1, t2, t3 tektonokon ágazzon el.

Az m Mushroomon beállítja:

* A t1-en legyen
* Az f fonálhoz tartozzon
* A termelt spórái közé felveszi az s4-et
* Fejlett gombává teszi (evolve)

A t1-en beállítja:

* t3 szomszédja legyen
* f fonál elágazzon rajta
* m gombatest rajta legyen

A t2-őn beállítja:

* t3 szomszédja legyen
* f fonál elágazzon rajta

A t3-on beállítja:

* t1, t2 is szomszédja legyen
* f fonál elágazzon rajta
* Az s1, s2 és s3 spórák ki legyenek lőve rá

Spórákon beállítja (s1, s2, s3 és s4):

* Az f fonálhoz tartozzanak
  1. ***Napló***

| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| --- | --- | --- | --- |
| 2025.03.13. 20.30 | 2 óra | Lekli, Fejes, Zelch, Szolár, Toronyi | Meeting, feladat átbeszélése. Döntés: Mindenki a saját szekvencia diagrammjának megfelelő teszteseteket készíti el. |
| 2025.03.15. 12:00 | 8 óra | Zelch | A kommunikációs diagramok közül: diagram 1, diagram 2, diagram 3, diagram 4 megcsinálása, ezután a következő tesztesetek és szekvencia diagramok megcsinálása: Gombatest elpusztulása  ,Gombatesthez nem kapcsolódó fonálrész eltávolítása,Tekton kettétörése, Tekton kettétörése, ha a tektonon gombatest van, Fonál felszívódása tektonról, Gombafonál elvágása |
| 2025.03.15. 11:00 | 7,5 óra | Szolár | * Sikertelen spóraszórás, mert nem szomszédos a céltekton. * Gombafonál sikeres elágaztatása tektonra. * Gombafonál sikertelen elágaztatása egyfonalas tektonra, mert már van rajta. * Új spóra termelődése.   Tesztesetek megalkotása.  Diagram 1 refaktorálása. |
| 2025. 03. 15. 18:00 | 3 óra | Fejes | Use case leírások és a hozzá tartozó szekvencia diagram elkészítése:   * első\_gombatest\_lehelyezése\_true * első\_gombatest\_lehelyezése\_false\_absorb * első\_gombatest\_lehelyezése\_false\_vangomba |
| 2025. 03. 16. 10:00 | 4 óra | Fejes | Use case leírások és a hozzá tartozó szekvencia diagram elkészítése:   * fejletlen\_spóraszórás\_true * fejlett\_spóraszórás\_true   Diagram 5 kommunikációs diagram elkészítése |
| 2025.03.16. 14:00 | 4 óra | Toronyi | Tesztesetek megtervezése, azok leírása, megfelelő szekvencia diagramok átdolgozása. |
| 2025.03.16. 15:00 | 3 óra | Lekli | Tesztesetek megtervezése, szekvenciadiagramok javítása. |
| 2025. 03. 17. 10:00 | 1,5 óra | Fejes, Toronyi | Meeting: szkeleton kezelői felület tervének átbeszélése, pontos leírás elkészítése. |
|  |  |  |  |

**6.Szkeleton beadás**

26 – five\_guys

Konzulens:

Huszerl Gábor

Csapattagok

| Lekli Zsolt | JENKHQ | zsoltlekli@gmail.com |
| --- | --- | --- |
| Fejes Ábel | AF1Y1C | fejesabel@gmail.com |
| Zelch Csaba | LK0617 | zch03po@gmail.com |
| Szolár Soma | E05SC6 | somaszolar1@gmail.com |
| Toronyi Zsombor | S8F7DV | toronyizsombi@gmail.com |

1. **Szkeleton beadás**
   1. ***Fordítási és futtatási útmutató***
      1. **Fájllista**

| **Fájl neve** | **Méret** | **Keletkezés ideje** | **Tartalom** |
| --- | --- | --- | --- |
| AbsorbingTecton.java | 12 617 bájt | 2025.03.20. 21:14 | Az AbsorbingTecton osztály a Tecton leszármazottja, amely olyan speciális viselkedéssel rendelkezik, hogy nem lehet rá Mushroom objektumot helyezni. Ez az osztály a rajta lévő FungalThread fonalakkal való műveletek kezelésére összpontosul. |
| FungalThread.java | 12 288 bájt | 2025.03.20. 21:14 | A FungalThread osztály egy fonál modellt definiál, amely gombák növekedését és tektonok közötti kapcsolatot reprezentálja. Ez az osztály számos funkciót biztosít a fonál részeinek kezelésére, például hozzáadására, eltávolítására, elágazására, és a nem szükséges fonálrészek eltávolítására. |
| Fungorium.java | 5 641 bájt | 2025.03.20. 21:14 | Ez a fájl az egyetlen futtatható állomány, itt található a main függvény, amely egy menüt jelenít meg a user számára és egy számérték bekérését követően a megfelelő menüponthoz tartozó tesztesetet futtatja le, amely naplózva megjelenik a konzolon. |
| Insect.java | 6 073 bájt | 2025.03.20. 21:14 | Az Insect osztály egy rovart reprezentál, amely különböző műveleteket tud végrehajtani egy "Tecton" típusú pozíción és gombafonalakon. Az osztály a mozgásra, pozíció megváltoztatására és fonál vágására vonatkozó függvényeket valósít meg. |
| MultiThreadTecton.java | 11 966 bájt | 2025.03.20. 21:14 | A MultiThreadTecton osztály a Tecton osztályból származik. Ez az osztály több szál kezelését, valamint a különféle gombafonalakhoz és gombatesthez kapcsolódó műveleteket valósítja meg. |
| Mushroom.java | 7 287 bájt | 2025.03.20. 21:14 | A Mushroom osztály egy gombát modellez a játékban, amely spórákat képes generálni és kilőni egy adott tektonra. A gomba két különböző állapotban lehet: fejlett vagy fejletlen. |
| MushroomState.java | 490 báj | 2025.03.20. 21:14 | A MushroomState a gomba fejlődési állapotainak reprezentálására szolgál. Ezt az osztályt különböző gomba-alapú logikákhoz lehet felhasználni. |
| NoCutSpore.java | 979 bájt | 2025.03.20. 21:14 | A NoCutSpore osztály a Spore absztrakt osztályból származik, és egy specifikus spóra effektust valósít meg, amely megakadályozza, hogy a célzott rovar vágási képességet használhasson. |
| ParalysingSpore.java | 995 bájt | 2025.03.20. 21:14 | A ParalysingSpore osztály egy konkrét megvalósítása a Spore absztrakt osztálynak. Ez az osztály reprezentál egy speciális spórát, amely alkalmazásakor megbénítja a célul választott rovart. |
| SingleThreadTecton.java | 12 524 bájt | 2025.03.20. 21:14 | A SingleThreadTecton osztály a Tecton osztályból származik, és egy olyan különleges tektont reprezentál, amelyen legfeljebb egy gomba és egy fonál lehet jelen. |
| SlowingSpore.java | 759 bájt | 2025.03.20. 21:14 | A SlowingSpore osztály egy konkrét megvalósítása a Spore absztrakt osztálynak. Ez az osztály egy lassító hatást alkalmaz az érintett rovarokra. |
| SpeedSpore.java | 1 090 bájt | 2025.03.20. 21:14 | A SpeedSpore osztály a Spore absztrakt osztály leszármazottja. Ez az osztály egy hatást alkalmaz egy rovarra, amely sebességnövekedést idéz elő. |
| Spore.java | 1 531 bájt | 2025.03.20. 21:14 | A Spore osztály egy absztrakt osztály, amely egy általános spóra mechanizmust definiál. Ez az osztály alapvető funkcionalitást és interfészt biztosít különféle specifikus spóra implementációk számára, amelyek különböző hatásokat alkalmazhatnak rovarokra. |
| SporeEffect.java | 664 bájt | 2025.03.20. 21:14 | A SporeEffect enum a különböző spóra hatásokat reprezentálja, amelyek alkalmazhatók rovarokra. Ezek az állapotok meghatározzák, hogy a spóra milyen módon befolyásolja a rovart. |
| Tecton.java | 13 199 bájt | 2025.03.20. 21:14 | A Tecton osztály egy absztrakt osztály, amely a játék tektonjait reprezentálja, ezek a játékterület alapvető egységei. |
| Tester.java | 48 309 bájt | 2025.03.22. 17:30 | A Tester osztály célja szimulált műveletek végrehajtása objektumok és függvények között, kifejezetten szekvencia diagramokban és belső kezelési logikákban.    Az osztály többek között tartalmaz metódusokat függvényhívások szimulálására, objektumok létrehozására és visszatérési értékek megjelenítésére.  Folyamatok vizualizációját végzi tesztelési forgatókönyvekben. |

* + 1. **Fordítás**

A programot a BME kari felhőjében biztosított, Windows 10 20H2 – JDK-Eclipse-WSU sablonnal ellátott virtuális gépen kell tudni fordítan. A játék fordítására kizárólag a JDK szükséges, amely biztosítja a java és javac parancsok elérhetőségét.

A következő parancs segítségével a fungorium package-ben található program fordításra kerül (a package-t tartalmazó mappából kiadva):

**javac -d . fungorium/Fungorium.java**

* + 1. **Futtatás**

A programot a BME kari felhőjében biztosított, Windows 10 20H2 – JDK-Eclipse-WSU sablonnal ellátott virtuális gépen kell tudni futtatni. A játék futtatásához kizárólag a JDK szükséges, amely biztosítja a java és javac parancsok elérhetőségét.

A fordítást követően az alábbi parancs segítségével a program futtatható:

**java fungorium.Fungorium**

* 1. ***Értékelés***

| **Tag neve** | **Tag neptun** | **Munka százalékban** |
| --- | --- | --- |
| Zelch Csaba | LK0617 | 20 |
| Fejes Ábel | AF1Y1C | 20 |
| Lekli Zsolt | JENKHQ | 20 |
| Soma Szolár | E05SC6 | 20 |
| Toronyi Zsombor | S8F7DV | 20 |

* 1. ***Napló***

| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| --- | --- | --- | --- |
| 2025.03.21 15:00 | 2 óra | Fejes  Szolár  Zelch  Toronyi  Lekli | Meeting, feladat átbeszélése. Döntés: Mindenki a saját szekvencia diagramjának megfelelő teszteseteket készíti el. |
| 2025.03.22. 12:00 | 10 óra | Zelch | A teszt osztály logika kialakításának segítése, kommunikációs diagramokból inicializáció készítése, osztályok getter setter függvényeinek megírása, a saját teszteseteim megvalósítása |
| 2025.03.22. 12:00 | 10 óra | Szolár | A teszt osztály és kiíratások logikájának kitalálásában részvétel.  init1 elkészítése.  A saját teszteseteim elkezdése. |
| 2025.03.22. 16:00 | 4 óra | Fejes | A kiosztott test case-ek és a hozzájuk tartozó metódusok implementálása |
| 2025.03.22. 16:00 | 4 óra | Toronyi | Kiosztott függvények, tesztesetek megvalósítása. |
| 2025.03.23 10:00 | 3 óra | Zelch | A saját teszteseteim befejezése |
| 2025.03.23. 13:00 | 3 óra | Szolár | Saját tesztestek és hozzájuk tartozó fg-k befejezése, kommentek, kiíratások finomítása. |
| 2025.03.23. 15:00 | 5 óra | Fejes | Implementáció befejezése, futtatható állomány menüjének elkészítése |
| 2025.03.23. 15:00 | 6 óra | Toronyi | Maradék tesztesetek implementálása, dokumentáció és javadoc kommentek elkészítése. |
| 2025.03.24 14:00 | 5 óra | Lekli | Szekvencia diagramoknak megfelelő tesztfüggvények megírása, szükséges függvények megvalósítása |

**7. Prototípus koncepciója**

26 – five\_guys

Konzulens:

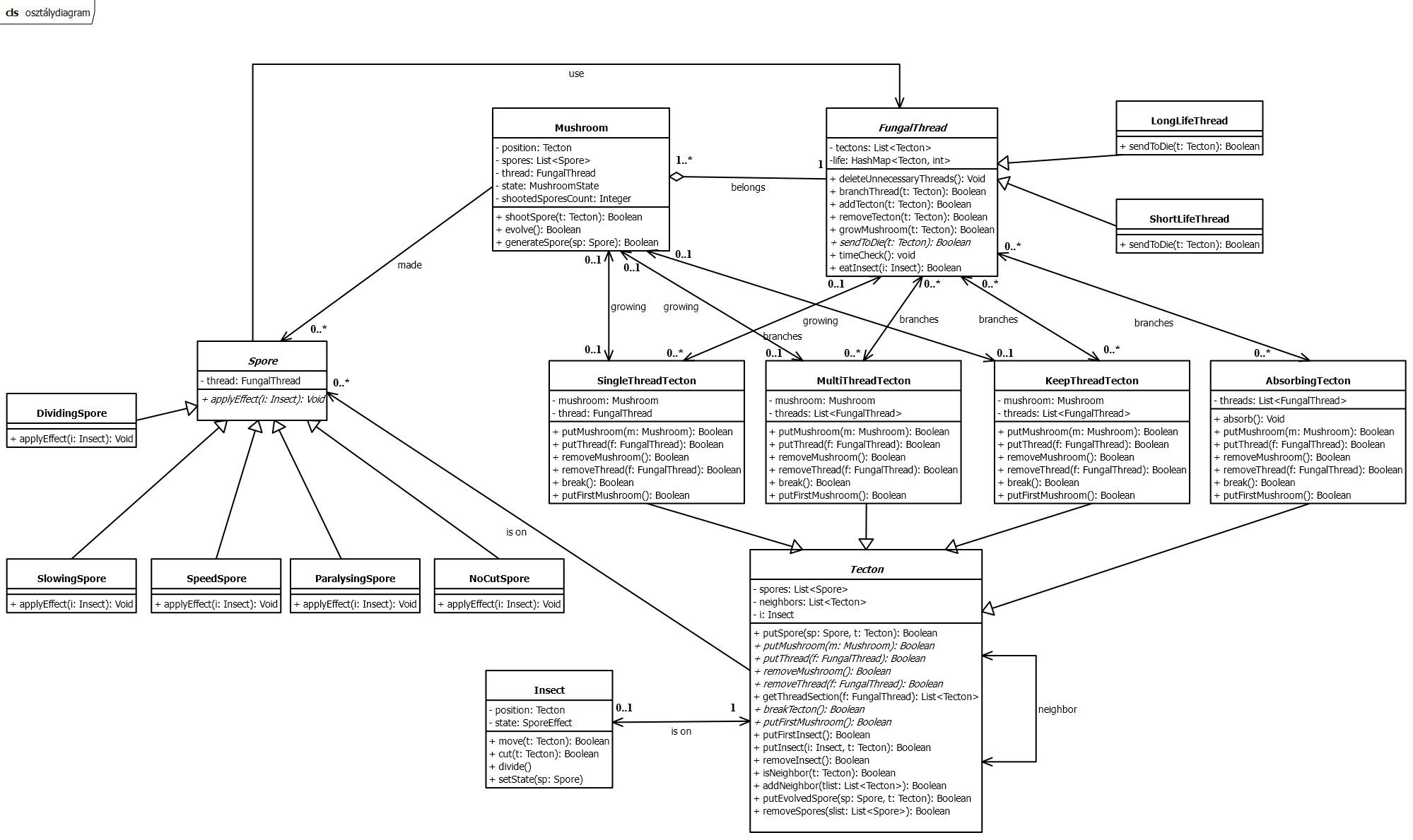
Huszerl Gábor

Csapattagok

| Lekli Zsolt | JENKHQ | zsoltlekli@gmail.com |
| --- | --- | --- |
| Fejes Ábel | AF1Y1C | fejesabel@gmail.com |
| Zelch Csaba | LK0617 | zch03po@gmail.com |
| Szolár Soma | E05SC6 | somaszolar1@gmail.com |
| Toronyi Zsombor | S8F7DV | toronyizsombi@gmail.com |

2025. 03. 17.

**Prototípus koncepciója**

* 1. **Változás hatása a modellre**
     1. **Módosult osztálydiagram** ****

#### 

* + 1. **Új vagy megváltozó metódusok**

***Új osztályok:***

* + - 1. **DividingSpore(Rovarosztó spóra)**

#### Felelősség

Azokért a spórákért felelős, amelyek elfogyasztása után a rovart osztódásra készteti . Ekkor a keletkezik egy újabb rovar, amelynek a rovarásza megegyezik az eredeti rovar rovarászával.

#### Ősosztályok

Spore (Spóra) -> *DividingSpore*

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

Csak olyanok, amelyet az ősosztályától örököl.

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl.

#### Metódusok

* + **void applyEffect(i: Insect)**: A metódus meghívja a rovar divide() függvényét
    - 1. **KeepThreadTecton(fonalat megtartó tekton)**

#### Felelősség

Ez a tektonfajta a MultiThreadTektonhoz hasonló, azonban életben tartja azokat a fonalakat, amelyek nincsenek közvetve vagy közvetlenül gombatesthez kötve

#### Ősosztályok

Tecton -> KeepThreadTecton

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

* + **asszociáció(branches)**: Másik oldal:FungalThread. Mint a MultiThreadTecton-nál.
  + **asszociáció(growing)**: Másik oldal: Mushroom. Van egy Mushroom attribútuma.

#### Attribútumok

* + **threads : List<FungalThread>**. Az asszociációnak megfelelően, lehet rajta akár több fonál is.
  + **mushroom : Mushroom**. Az asszociációnak megfelelően, maximum egy gombatest lehet rajta.

#### Metódusok

* + **putThread(f : FungalThread) : boolean**. Mint a MultiThreadTecton-nál.
  + **putMushroom(m : Mushroom) : boolean**. Mint a MultiThreadTecton-nál.
  + **removeThread(f : FungalThread) : boolean.** Mint a MultiThreadTecton-nál.
  + **removeMushroom() : boolean.** Mint a MultiThreadTecton-nál.
  + **break(): boolean**: boolean: Tekton kettétörését valósítja meg, két KeepThreadTecton keletkezik
  + **putFirstMushroom(): boolean**: Játékos első gombatestjének lehelyezése
    - 1. **ShortLifeThread**

#### Felelősség

Felelőssége hasonló a FungalThread felelősségeihez, azonban fonálvágás esetén nem egyből pusztulnak el róla a fonalak, hanem csak a következő kör elején

#### Ősosztályok

FungalThread -> ShortLifeThread

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

Csak olyanok, amelyet az ősosztályától örököl.

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl.

#### Metódusok

* + **sendToDie(t: Tecton): boolean**: A metódus a kapott tektont, ha a fonál rajta van a tektonon, behelyezi a life<Tecton, int> HashMap-be kulcsként, értékként pedig egyet állít be. A gombafonálnak van egy timeCheck() metódusa, amely minden kör elején meghívódik, célja, hogyha a HashMap-ben van elem, akkor ezek értékét csökkentse mindig 1-el. Ha az érték 0 lesz, ekkor a tárolt tektonról leveszi a fonalat, majd az elemet kitörli a HashMap-ből
    - 1. **LongLifeThread**

#### Felelősség

Felelőssége hasonló a FungalThread felelősségeihez, azonban fonálvágás esetén nem egyből pusztulnak el róla a fonalak, hanem csak a következő kör utáni körben

#### Ősosztályok

FungalThread -> LongLifeThread

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

Csak olyanok, amelyet az ősosztályától örököl.

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl.

#### Metódusok

* + **sendToDie(t: Tecton): boolean**: A metódus hasonló, mint a ShortLifeThread-nél, azonban a HashMap-ben az elem értékét kettőre állítja be

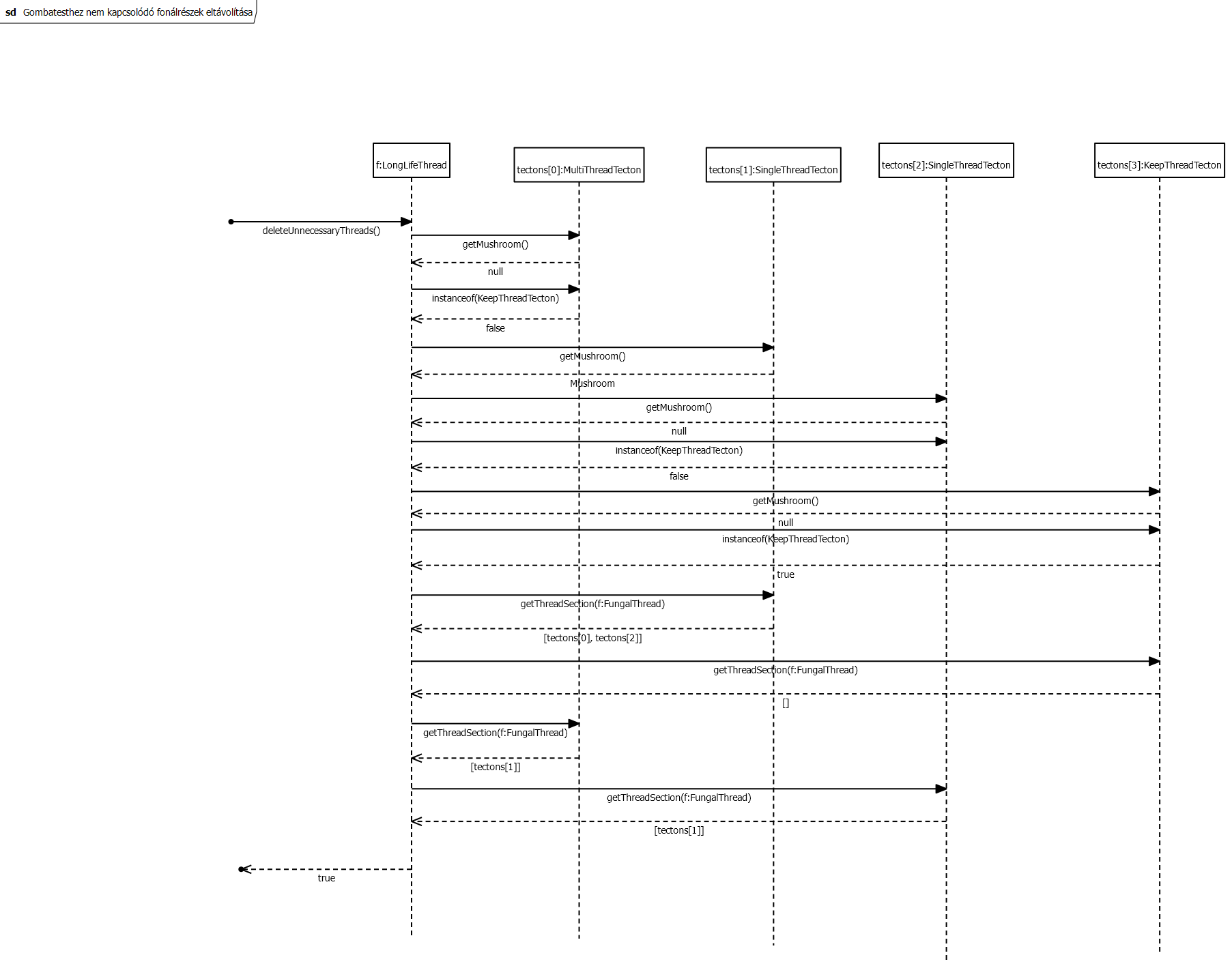
***Változások a már meglévő osztályokban***

Az **Insect** osztályban történt változtatások:

* az Insect osztálynak lett egy új divide() metódusa
  + **divide(): boolean**: A metódus egy új rovart hoz létre, melyet az eredeti rovar tektonjának az első szomszédjára helyez le, ha ez nem lehetséges, mert azon a tektonon már van rovar, tovább lépeget a szomszédain, ha egyetlen szomszédjára se tudta lehelyezni, sorban végigmegy a tektonokon és az első olyan tektonra helyezi le, amire ez lehetséges
* megváltozott a **cut(t: Tecton)** metódusa, amely a gombafonálnak a sendToDie(t: Tecton) metódusát fogja hívni és nem a removeTecton(t: Tecton) metódusát, valamint a tektonon, amelyen elvágta a fonalat nem hívja többet meg a removeThread(f: FungalThread) metódust

A **FungalThread** osztályban történt változtatások:

* az osztály absztrakt osztállyá vált
* lett egy új attribútuma
  + **life: HashMap<Tecton, int>**: ebben az attribútumban tároljuk el azokat a tektonokat, amelyeken a gombafonalak el kell pusztuljanak, mivel valamelyik bogarász elvágta, ez lesz a kulcs, értékként pedig azt tároljuk el, hogy hány kör múlva kell a fonalak elpusztuljanak a tetkonról
* új metódusokkal bővült
  + **sendToDie(t: Tecton): boolean** :absztrakt metódus
  + **timeCheck(): void**: a metódust minden kör elején a kontroller meghívja, és ha a time HashMap nem üres, akkor csökkenti 1-el a benne levő elemek értékét. Ha az érték elérte a 0-át, akkor a tárolt tektonról leveszi a fonalat meghívva a removeThread(f: FungalThread) metódusát, majd törli az elemet a HashMapből, illetve a gombafonál tectons listájából törli a tectont, és lefuttatja a deleteUnnecessaryThreads() metódust
  + **eatInsect(i: Insect):boolean**: leveszi a rovart a tektonról, majd törli a rovart. Abban az esetben, ha a rovar nem AbsorbingTectonon, vagy nem egy olyan tektonon helyezkedett el, ahol már volt gombatest, akkor növeszt egy új gombatestet
* változott a **deleteUnnecessaryThreads()** metódusa: a metódus során sorban végigmegy a tectons tömbjén, megnézi, hogy mely tektonokon van ugyanolyan fajból származó gombatest, ezeket eltárolja, majd ha nincs ilyen gombatest rajtuk, akkor megnézi, ha nem-e KeepThreadTecton típusúak, ha ez teljesül rájuk, akkor is eltárolja, végül megkeresi az összes olyan fonalat, amelyek elérhetők az elmentett tektonokról haladva. Ezek a fonálrészek lesznek azok a részek, amelyeket meg kell tartani, a többit pedig töröljük.
  + 1. **Szekvencia-diagramok**

**Gombatesthez nem kapcsolódó fonálrész eltávolítása**

A deleteUnnecessaryThread függvényhívás érkezhet egy tektontól, egy gombatesttől, vagy akár egy külső kontrollertől

Ezen a diagramon a következő objektumok találhatók:

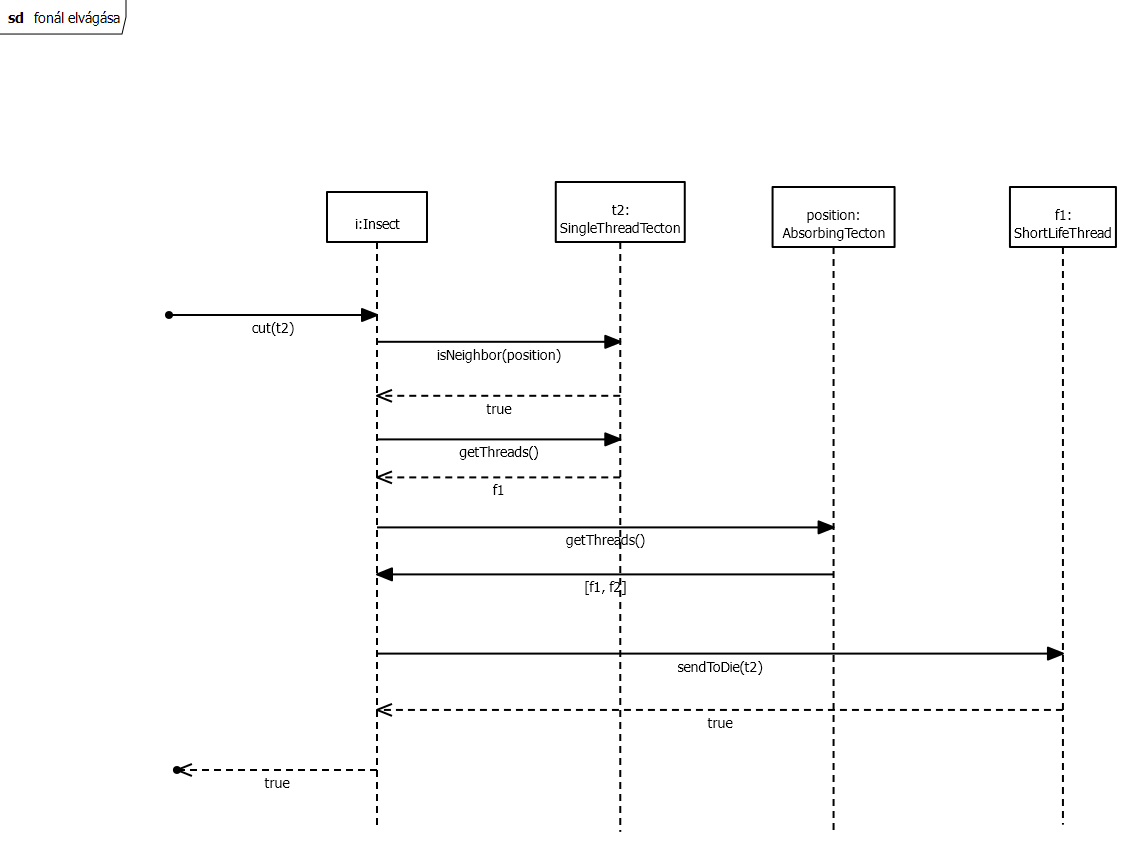
* f : gombafonál
* tectons[0] : tekton, amelyen az f gombafonál elágazik
* tectons[1] : tekton, amelyen az f gombafonál elágazik
* tectons[2] : tekton, amelyen az f gombafonál elágazik
* tectons[3] : tekton, amelyen az f gombafonál elágazik

A tectons az f gombafonál egy tömbje, amelyben azokat a tektonokat tárolja, amelyen elágazik.

A metódus sorban végigmegy a tectons tömb tektonjain, megnézi melyiken van ugyanabból a fajból származó gombatest, ezt eltárolja. Ha nincs rajta ilyen gombatest, megnézi, hogy KeepThreadTecton fajtájú-e, ha igen, ugyancsak eltárolja.

Miután ezt megcsinálta az eltárolt tektonokon végigmegy, és meghívja rájuk a getThreadSection(f: FungalThread) metódusukat, amely visszaadja azokat a szomszédos tektonokat, amelyeken elágazik az f gombafonál, és ezeket a tektonokat is eltárolja. A kapott tektonokon újra meghívja ezt a metódust, és addig folytatja, amig talál új tektont.

Mikor már nem talál új tektont, azt jelenti, hogy megtatált az összes olyan fonálrészt, amely közvetve vagy közvetlenül gombatesthez vagy KeepThreadTecton-hoz van kötve, tehát ezeken a gombafonalakat meg kell tartani, és azokon a tektonokon kell törölni, amelyek nincsenek a megtaláltak között.

**Gombafonál elvágása**

Egy külső kontroller meghívja az i rovar cut metódusát.

Ezen a diagramon a következő objektumok találhatók:

* i: Rovar, amely a vágást végzi
* t2: ezt a tektont jelöltük ki
* position: ezen a tektonon található a rovar
* f1: gombafonál, amely a t2 és a position tektonon is elágazik

A **cut(t)** metódushívás után leellenőrizzük, hogy a paraméterül kapott t tekton szomszédos-e a position tektonnal, azaz azzal a tektonnal, amelyen a rovar rajta van.

Ha ez teljesül, akkor mindkét tektonról lekérjük a gombafonalakat mindkettőről külön-külön. Ha találunk olyat, amely mindkettőn elágazik, akkor meghívjuk a fonál sendToDie metódusát a t2 tektonra, amely felveszi a fonál time HashMap-jébe a t2 tektont kulcsként, és hozzá értékként 1-et.

**Fonál elpusztulása a tektonról vágás után**



Egy külső kontroller meghívja az f fonál timecheck() metódusát.

Ezen a diagramon a következő objektumok találhatók:

* f: gombafonál
* t: ezen a tektonon pusztul el a gombafonál

A timeCheck() metódust minden kör elején a kontroller meghívja, és ha a time HashMap nem üres, akkor csökkenti 1-el a benne levő elemek értékét. Ha az érték elérte a 0-át, akkor a tárolt tektonról leveszi a fonalat meghívva a removeThread(f: FungalThread) metódusát, majd törli az elemet a HashMapből, illetve a gombafonál tectons listájából törli a tectont, és lefuttatja a deleteUnnecessaryThreads() metódust.

**Bogár megevése**

Egy külső kontroller meghívja az f gombafonál eatInnsect(i)metódusát.

Ezen a diagramon a következő objektumok találhatók:

* f: gombafonál, amely megeszi a rovarat
* i: Rovar
* position: ezen a tektonon található az i rovar
* m: keletkező gombatest

Az eatInsect() metódus leveszi a rovart a tektonról, majd törli a rovart. ezután megnézi, hogy a tektonon, amelyen a rovar volt van-e gombatest, illetve hogy nem-e AbsorbingTecton típusú. Ha ebből egyik se teljesül, létrehoz egy új gombatestet, amelyet elhelyez a tektonra, ahonnan a rovart megette.

**Rovar mozgása során DividingSpore-t evett meg**



Egy külső kontroller meghívja az i rovar move metódusát.

Ezen a diagramon a következő objektumok találhatók:

* i: Rovar
* t2: erre a tektonra lépünk a rovarral
* t1: ezen a tektonon van rajta a rovar
* spores[0]: ezt a spórát eszi meg a rovar
* neighbors[0]: a t2 tekton első szomszédja

Azt mutatja be az ábra, amikor a Rovar sikeresen mozog egy tektonra, a mozgás akkor sikeres, ha a két tekton között van gombafonál és szomszédosak(a függvényen belül megnézi, hogy a tekton szomszéd listájába szerepel-e a másik). Ekkor az ott lévő spórák közül megesz egyet és megkapja annak a hatását, majd a spóra megszűnik létezni.

Jelen esetben egy DividingSpore típusú spórát evett meg, amely meghívja a bogár divide függvényét, ennek hatására keletkezik egy új bogár, amely a régi bogár tektonjának az első szomszédjára kerül.

A diagram elején az i:Insect position adattagja egyenlő t1 tektonnal, ezt adjuk át a putInsect-ben.

* 1. ***Prototípus interface-definíciója***
     1. **Az interfész általános leírása**

A projekt teszteléséhez egy szöveges, konzol alapú interfészt alkalmazunk. Az interfész parancsalapú, a felhasználó a különböző fázisokban strukturált parancsokat adhat ki. Ezeket a rendszer egyéni szintaktikával értelmezi és hajtja végre.

MushroomController, MushroomView

TectonController, TectonView

FungalThreadController, FungalThreadView

InsectController, InsectView

SporeController, SporeView

* + 1. **Bemeneti nyelv**

arrange

Leírás: Arrange fázis elkezdését jelenti, innentől lehet kiadni az arrange fázisba tartozó parancsokat.

Opciók: -

end

Leírás: Aktuális fázis véget ér, ezután lehet másik fázis kezdése parancsot kiadni.

Opciók: -

act

Leírás: act fázis elkezdését jelenti, innentől lehet kiadni az act fázisba tartozó parancsokat.

Opciók: -

saveResult

Leírás: A result.txt fájlba menti az aktuális állapotot.

Opciók: -

loadResult

Leírás:Betölti a result.txt fájlból az állapotot.

Opciók: -

turnOffRandom

Leírás: Ha ez be van kapcsolva a program minden egyes alkalomkor, mikor valami random értéket szeretne a felhasználóhoz fordul, és kér egy értéket

Opciók: -

turnOnRandom

Leírás: Ha ez be van kapcsolva a program minden egyes alkalomkor, mikor valami random értéket szeretne a program generál egy random értéket

Opciók: -

loadInit <InitFájlneve>

Leírás: Ahhoz, hogy elkezdhessük a játékot, először fel kell építenünk a pályát, azaz a tektonokat és a szomszédságaikat, amelyet egy adott fájlból fogunk betölteni

Opciók: -

assert<Elvártkimenetfájlnév>

Leírás:A result.txt fájlban elmentett állapotát a rendszernek összehasonlítja a tesztesethez tartozó már megadott .txt fájllal, és dönt a helyes működésről

Opciók: -

runTest<Tesztesetfájlneve>

Leírás: A parancs során a program megadott .txt fájlból sorra olvassa be és futtatja le a parancsokat, amig az összes parancsot végre nem hajtja a fájlból, azaz a tesztesetet le nem futtatja

Opciók: -

**Arrange parancsok:**

createTecton <Tektonfajta>

Leírás: A parancs hatására az adott tektonfajta alapján létrejön egy új tekton, a nevét a kontroller generálja, az alapján, hogy eddig hány tekton volt a játékban, a tekton neveknek a következő a szintaxisa: t<játékban előfordult tektonok száma+1>

Opciók: -

setNeighbors <Tektonnév> [Tektonnév] [Tektonnév] [Tektonnév]

Leírás: Adott nevű tektonnak beállítja a szomszédjait.

Opciók: Meg lehet adni egy vagy több szomszédot is.

createShortLifeThread <Tektonnév> <Játékosnév>

Leírás: Létrehoz egy ShortLifethread fonalat a <Tektonnév> tectonon, illetve odaadja ezt a gombafonalat az adott nevű játékosnak, a fonal nevét a kontroller generálja, az alapján, hogy eddig hány tekton volt a játékban, a gombafonál neveknek a következő a szintaxisa: f<játékban előfordult fonalak száma+1>

Opciók: -

createLongLifeThread <Tektonnév> <Játékosnév>

Leírás: Létrehoz egy LongLifethread fonalat a Tektonnév tectonon, illetve odaadja ezt a gombafonalat az adott nevű játékosnak, a fonal nevét a kontroller generálja, az alapján, hogy eddig hány tekton volt a játékban

Opciók: -

setTectons <Fonálnév> [Tektonnév] [Tektonnév] [Tektonnév]

Leírás: A fonál tectons listájába beveszi a tektonokat, valamint a tektonokra is rárakja a fonalat.

Opciók: Megadható egy vagy több tekton is.

createMushroom <Tektonnév> <Fonálnév> <Játékosnév>

Leírás: Létrehoz egy gombatestet az adott tektonra, és beállítja, hogy a Fonálnév nevű fonálhoz tartozik, illetve odaadja az adott játékosnak, és beállítja az éltekorát nullára, a gombatest nevét a kontroller generálja, az alapján, hogy eddig hány gombatest volt a játékban, a gombatest neveknek a következő a szintaxisa: m<játékban előfordult gombatestek száma+1>. Ezenkívül növelődik eggyel a <Játékosnév> játékos points attribútuma

Opciók: -

createEvolvedMushroom <Tektonnév> <Fonálnév> <Játékosnév>

Leírás: Létrehoz egy fejlett gombatestet az adott tektonra, és beállítja, hogy a <Fonálnév> nevű fonálhoz tartozik, illetve odaadja az adott játékosnak, és beállítja az életkorát 6-ra, a gombatest nevét a kontroller generálja, az alapján, hogy eddig hány gombatest volt a játékban. Ezenkívül növelődik eggyel a <Játékosnév> játékos points attribútuma

Opciók: -

createInsect <Tektonnév> <Játékosnév>

Leírás: Létrehoz egy rovart az adott nevű tektonon, majd odaadja az adott nevű játékosnak, a rovar nevét a kontroller generálja, az alapján, hogy eddig hány rovar volt a játékban

Opciók: -

createFungusPlayers <Játékosnév> <Játékosnév> <Játékosnév> <Játékosnév>

Leírás: A parancs során létrejönnek a gombászok a megadott nevek szerint, maximum négy nevet adhatunk meg, mivel egyszerre maximum csak négy gombász játszhat a játékkal

Opciók: -

createInsectPlayers <Játékosnév> <Játékosnév> <Játékosnév> <Játékosnév>

Leírás: A parancs során létrejönnek a rovarászok a megadott nevek szerint, maximum négy nevet adhatunk meg, mivel egyszerre maximum csak négy rovarász játszhat a játékkal

Opciók: -

generateSpore <Gombatest név> <Spórafajta>

Leírás: az adott gombatest spores listájába egy <Spórafajta> fajtájú spórát helyez el, a spspóra nevét a kontroller generálja, az alapján, hogy eddig hány spóra volt a játékban

Opciók: -

setShotSpores <Gombatest név> <Nem negatív egész szám>

Leírás: beállítjuk, hogy a gombatest eddig hány spórát lőtt ki.

Opciók: -

createSpore <Spórafajta> <Tektonnév> <Fonálnév>

Leírás: Ennek során létrejön egy <Spórafajta> fajtájú spóra, amely az adott fonálhoz tartozik, és az adott tektonra kerül, a spóra nevét a kontroller generálja, az alapján, hogy eddig hány spóra volt a játékban

Opciók: -

setState <Rovarnév> <InsectState>

Leírás: Beállítja az adott rovar állapotát az adott spórahatásnak megfelelően.

Opciók: -

setMaxRound <Pozitív egész szám>

Leírás: Megmondhatjuk, hogy a játéknak mennyi lehet a maximum körszáma, ezután a játék befejeződik.

Opciók:

setMushroomAge <Gombatestnév> <Nem negatív egész szám>

Leírás: Beállítja a gombatest életkorát, azaz beállítja, hogy hány köre léteziik a gombatest a játékban

Opciók: -

setFungalPlayerCount <Pozitív egész szám>

Leírás: Beállítja a gombászok számát.

Opciók: -

setInsectPlayerCount <Pozitív egész szám>

Leírás: Beállítja a rovarászok számát.

Opciók: -

stepGameRound <Nem negatív egész szám>

Leírás: A játék köreinek számát lépteti a parancs által megadott egész számmal

Opciók: -

break <Tektonnév>

Leírás: A parancs hatására a <Tektonnév>-ben megadott tekton kettétörik, ennek során két új tekton keletkezik, ezeknek neveit a kontroller generálja, az alapján, hogy eddig hány tekton volt a játékban

Opciók: -

absorb <Tektonnév>

Leírás: A parancs hatására a <Tektonnév>-ben megadott megadott tektonról felszívódnak a gombafonalak

Opciók: -

evolve <Gombatestnév>

Leírás: A parancs hatására a <Gombatestnév>-ben megadott megadott gombatest ha még nem fejlett, akkor fejletté válik

Opciók: -

deleteUnnecessaryThreads <Fonálnév>

Leírás: A parancs hatására a megadott fonálnak törlődnek azok a részei, melyek nem kapcsolódnak ugyanolyan fajból származó gombatesthez, vagy KeepThreadTecton-hoz

Opciók: -

timeCheck

Leírás: A parancs hatására az összes gombafonalon meghívódik a timeCheck() metódus

Opciók: -

divide <rovarnév>

Leírás: A parancs hatására a kapott rovar kettéválik, az új rovar nevét a kontroller generálja, az alapján, hogy eddig hány rovar volt a játékban

Opciók: -

View felőli interface parancsai:

getState <rovarnév>

Leírás: A parancs hatására a kapott rovarnak lekérjük az állapotát

Opciók: -

getInsect <Tectonnév>

Leírás: A parancs hatására a kapott tektonról lekérjük a bogarat

Opciók: -

getMushroom <Tectonnév>

Leírás: A parancs hatására a kapott tektonról lekérjük a gombatestet

Opciók: -

getFungalThread <Tectonnév>

Leírás: A parancs hatására a kapott tektonról lekérjük a rajta levő fonalakat

Opciók: -

getSporeTecton <Tecton>

Leírás: A parancs hatására a kapott tektonról lekérjük a spórákat

Opciók: -

**Act parancsok:**

closeStep

Leírás: A parancsot a játékosok adják ki, amikor le akarják zárni a lépésüket, ezt az első gombatest, illetve az első rovar lehelyezése kivételével minden lépésben ki kell adni

Opciók: -

putFirstMushroom <Fonáltípus> <Tektonnév>

Leírás: Gombász első lépése, lehelyez gombatestet adott tektonra, ezzel együtt egy gombafonál is létrejön a tektonon, melynek típusa <Fonáltípus>, a gombatest és a gombafonál nevét a kontroller generálja, az alapján, hogy eddig hány gombatest, illetve gombafonál volt a játékban. Ezenkívül az adott gombásznak növelődik eggyel a points attribútuma

Opciók: -

branchThread <Fonal név> <Tektonnév>

Leírás: Adott nevű fonalat átnöveszti az adott tektonra.

Opciók: -

shootSpore <Gombatest név> <Tektonnév>

Leírás: Adott nevű gombatest spórát lő adott tektonra.

Opciók: -

growMushroom <Fonal név> <Tektonnév>

Leírás: Adott nevű gombafonál gombatestet növeszt az adott tektonra, a gombatest nevét a kontroller generálja, az alapján, hogy eddig hány gombatest, volt a játékban. Ezenkívül az adott gombásznak növelődik eggyel a points attribútuma

Opciók: -

eatInsect <Fonal név> <Rovarnév>

Leírás: Adott nevű fonál megeszi az adott rovart. Ha lehetséges, akkor gombatest nő ezen a helyen, ekkor az adott gombásznak növelődik eggyel a points attribútuma.

Opciók: -

putFirstInsect <Tektonnév>

Leírás: A rovarász első lépése, lehelyez egy rovart az adott tektonra, a rovar nevét a kontroller generálja, az alapján, hogy eddig hány rovar, volt a játékban

Opciók: -

move <Rovarnév> <Tektonnév>

Leírás: Az adott nevű rovar átmozog az adott tektonra. Ha megeszik egy spórát növeljük az adott rovarász points attribútumát eggyel.

Opciók: -

cut <Rovarnév> <Tektonnév>

Leírás: Az adott nevű rovar elvágja a saját tektonjáról az adott tektonra vezető fonalat.

Opciók: -

* + 1. **Kimeneti nyelv**

## Általános leírás

A program képes elmenteni az objektumok állapotát két különböző fájlba:

* **init.txt**: Ez a fájl a teszteset kezdeti állapotát tartalmazza, amit az arrange utasításokkal hoztunk létre.
* **result.txt**: Ez a fájl az act parancsok végrehajtása utáni állapotot tartalmazza.
* A kettőt az assert fázisban betöltjük és összehasonlítjuk őket. Az alapján mondjuk meg a teszt sikerességét, hogy a result.txt tartalma az init.txt-ből kiíndulva, az act utasítás(oka)t alkalmazva az elvárásainknak megfelelő-e.

Az állapot mentésére és betöltésére szolgáló parancsok:

* **saveInit**: Az init.txt fájlba menti az aktuális állapotot. - az arrange fázis után esedékes
* **saveResult**: A result.txt fájlba menti az aktuális állapotot. - az act fázis után esedékes
* **loadInit**: Betölti az init.txt fájlból az állapotot. - assert fázis elején esedékes
* **loadResult**: Betölti a result.txt fájlból az állapotot. - assert fázis eleje esedékes

## Fájlformátum

A fájl szerkezete az alábbi módon épül fel:

-begin-

<Objektum típusa> <Objektum neve>

<Objektum típusa> <Objektum neve>

…

--- # Objektumok listájának vége

<Objektum neve>

<Objektum attribútuma> <Érték/Név> <Érték/Név> ...

<Objektum attribútuma> <Érték/Név>

<Objektum attribútuma> - # ilyen akkor lehet, ha ez az attribútuma még nincs beállítva pl.: null

…

#elválasztó üres sor

<Objektum neve>

…

-end-

A kiíratás kettő részre oszlik el:

* -begin- <-> — : Az szereplők felsorolása, innen tudjuk, hogy milyen objektumok, milyen névvel kerülnek be a HashMap-ünkbe.
* — <-> -end- : Az szereplők attribútumainak felsorolása. Az összeset felsoroljuk. Ha az értéke inicializálatlan pl.: nullptr a listájára, vagy a kilőtt spóráját -1-re állította ctor-a, akkor azt egy - jellel jelöljük. Az attribútum értékeken kívül innen tudjuk meg a szereplők közti asszociációkat.
  + Az objektumok felsorolásait üres sorok tagolják.

### 

### Példa egy konkrét fájlra:

-begin-

MultiThreadTecton t1

SingleThreadTecton t2

FungalThread f1

Mushroom m1

SpeedSpore s1

—

t1

spores -

neighbors t2

i -

mushroom m1

threads f1

t2

spores s1

neighbors t1

i -

mushroom -

thread -

f1

tectons t1

life -

s1

thread f1

-end-

* 1. ***Összes részletes use-case***

| **Use-case neve** | Első gombatest lehelyezése |
| --- | --- |
| **Rövid leírás** | A játék elején a gombász első lépése az első gombatestjének a lehelyezése egy tektonra. Ebben az esetben egy olyan tektonra próbálja lehelyezni a gombász a gombatestet, ahova a játék szabályok szerint le tudja. Ez egy olyan tekton lehet, amin nincs másik gombatest és a fajtája nem AbsorbingTecton. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * Gombász választ egy tektont * A tektonra lehelyezhető gombatest * A tektonra rákerül a gombatest |
| **bemenet** | A játékos hatására meghívódik a kiválasztott tektonon a putFirstMushroom. - ***<Gombatest név> putFirstMushroom <Tektonnév>*** |

| **Use-case neve** | Első gombatest lehelyezése olyan tektonra, amelyen már van gombatest |
| --- | --- |
| **Rövid leírás** | A játék elején a gombász első lépése az első gombatestjének a lehelyezése egy tektonra. Ebben az esetben egy olyan tektonra próbálja lehelyezni a gombász a gombatestet, ahol már van gombatest. Ekkor nem fog lehelyeződni a gombatest a tektonra. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * Gombász választ egy tektont * A tektonon van már gombatest * A tektonon nem történik változás, a gombatest nem kerül lehelyezésre |
| **bemenet** | A játékos hatására meghívódik a kiválasztott tektonon a putFirstMushroom. - ***<Gombatest név> putFirstMushroom <Tektonnév>*** |

| **Use-case neve** | Első gombatest lehelyezése absorbing tektonra |
| --- | --- |
| **Rövid leírás** | A játék elején a gombász első lépése az első gombatestjének a lehelyezése egy tektonra. Ebben az esetben egy olyan tektonra próbálja lehelyezni a gombász a gombatestet, ami Absorbing típusú. Az ilyen típusú tektonokon nem lehet gombatest. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * Gombász választ egy tektont * A tekton Absorbing típusú * A tektonon nem történik változás, a gombatest nem kerül lehelyezésre |
| **bemenet** | A játékos hatására meghívódik a kiválasztott tektonon a putFirstMushroom. - ***<Gombatest név> putFirstMushroom <Tektonnév>*** |

| **Use-case neve** | Gombafonál növesztése olyan tektonra, amelyre tud növeszteni |
| --- | --- |
| **Rövid leírás** | A gombász a körében tud gombafonalat növeszteni, elágaztatni egy adott tektonon. Ehhez ki kell választania azt a tektont, ahova a fonalát el szeretné ágaztatni. Ezt akkor tudja sikeresen véghez vinni, ha a kiválasztott tektonnak van legalább egy olyan szomszédja, ahol van azonos fajtájú FungalThread fonál az elágaztatni kívánttal. Továbbá fontos, hogy SingleThread fajtájú tektonon csak egy fonál ágazhat el. Ebben az esetben olyan tektonon ágaztatja el a fonalát a játékos, amelyen el lehet. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * Gombász választ egy tektont * A tektonon elágaztatható a játékos fonala * A tektonon elágazik a játékos fonala |
| **bemenet** | A játékos hatására meghívódik a fonalán a branchThread metódus, amely paraméterül a kiválasztott tektont kapja. - ***<Fonal név> branchThread <Tektonnév>*** |

| **Use-case neve** | Gombafonál növesztése olyan tektonra, amelyre nem tud növeszteni, mert a tekton messze van |
| --- | --- |
| **Rövid leírás** | A gombász a körében tud gombafonalat növeszteni, elágaztatni egy adott tektonon. Ehhez ki kell választania azt a tektont, ahova a fonalát el szeretné ágaztatni. Ezt akkor tudja sikeresen véghez vinni, ha a kiválasztott tektonnak van legalább egy olyan szomszédja, ahol van azonos fajtájú FungalThread fonál az elágaztatni kívánttal. Továbbá fontos, hogy SingleThread fajtájú tektonon csak egy fonál ágazhat el. Ebben az esetben olyan tektonon ágaztatja el a fonalát a játékos, amelyen nem tudja, mert a tektonnak nincs egy olyan szomszédja se, amelyen el van ágazva a játékos fonala. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * Gombász választ egy tektont * A tektonon nem ágazható el a játékos fonala, mert messze van (egy szomszédján sincs az elágaztatni kívánttal azonos fajtájú fonál) * A tektonon nem történik változás, a fonál nem ágazik rajta el |
| **bemenet** | A játékos hatására meghívódik a fonalán a branchThread metódus, amely paraméterül a kiválasztott tektont kapja. - ***<Fonal név> branchThread <Tektonnév>*** |

| **Use-case neve** | Gombafonál növesztése olyan SingleThread tektonra, amelyre nem tud növeszteni, mert a tektonon már ágazik el másik fajta fonál |
| --- | --- |
| **Rövid leírás** | A gombász a körében tud gombafonalat növeszteni, elágaztatni egy adott tektonon. Ehhez ki kell választania azt a tektont, ahova a fonalát el szeretné ágaztatni. Ezt akkor tudja sikeresen véghez vinni, ha a kiválasztott tektonnak van legalább egy olyan szomszédja, ahol van azonos fajtájú FungalThread fonál az elágaztatni kívánttal. Továbbá fontos, hogy SingleThread fajtájú tektonon csak egy fonál ágazhat el. Ebben az esetben olyan tektonon ágaztatja el a fonalát a játékos, amelyen nem tudja, mert a tekton típusa SingleThread, és már ágazik el rajta egy másik fajta fonál. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * Gombász választ egy tektont, ami típusa SingleThread * A tektonon nem ágazható el a játékos fonala, mert már elágazik rajta egy másik fajta fonál (másik játékos fonala) * A tektonon nem történik változás, a fonál nem ágazik rajta el |
| **bemenet** | A játékos hatására meghívódik a fonalán a branchThread metódus, amely paraméterül a kiválasztott tektont kapja. - ***<Fonal név> branchThread <Tektonnév>*** |

| **Use-case neve** | Gombafonál sikeres növesztése olyan tektonra, amelyen spóra van |
| --- | --- |
| **Rövid leírás** | A gombász a körében tud gombafonalat növeszteni, elágaztatni egy adott tektonon. Ehhez ki kell választania azt a tektont, ahova a fonalát el szeretné ágaztatni. Ezt akkor tudja sikeresen véghez vinni, ha a kiválasztott tektonnak van legalább egy olyan szomszédja, ahol van azonos fajtájú FungalThread fonál az elágaztatni kívánttal. Továbbá fontos, hogy SingleThread fajtájú tektonon csak egy fonál ágazhat el. Ebben az esetben olyan tektonon ágaztatja el a fonalát a játékos, amelyen tudja. Továbbá ezen a tektonon van olyan spóra, ami ugyanolyan fajtájú (ugyanahhoz a játékoshoz tartozik), mint a fonál. Ekkor a játékos újra elágaztathatja a fonalát egy olyan tektonra, amely szomszédos azzal a tektonnal, amelyen a spóra rajta van. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * A játékos választ egy tektont * A tektonon elágaztatható a játékos fonala * A tektonon van olyan spóra, amely a játékoshoz tartozik * A tektonon elágazik a játékos fonala * A játékos kiválaszt egy új tektont * A tekton szomszédos azzal a tektonnal, amelyen előbb elágazott a játékos fonala * A tektonon elágaztatható a játékos fonala * A tektonon elágazik a játékos fonala |
| **bemenet** | A játékos hatására meghívódik a fonalán a branchThread metódus, amely paraméterül a kiválasztott tektont kapja. - ***<Fonal név> branchThread <Tektonnév>***  Ezután a játékos hatására megint meghívódik a fonalán a branchThread metódus, amely paraméterül a kiválasztott tektont kapja, amely szomszédos az előzőleg kiválasztott tektonnal. - ***<Fonal név> branchThread <Tektonnév>*** |

| **Use-case neve** | Fejletlen gombatest spóraszórása olyan tektonra, amelyre tud |
| --- | --- |
| **Rövid leírás** | A gombász játékos a saját körében lőhet spórát az egyik gombatestjéből. Ekkor a játékosnak ki kell választania azt a gombatestet, amelyikből a spórát lőni szeretné. Ekkor ha a gombatestben van termelt spóra, akkor ki kell választania a játékosnak azt a tektont, ahova lőni szeretné a spórát. Ebben az esetben a kiválasztott gombatest fejletlen, így a céltektonnak muszáj szomszédosnak lennie a gombatest tektonjával. A kiválasztott céltekton most szomszédos a gombatest tektonjával. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * Gombász kiválaszt egy hozzá tartozó gombatestet, ami fejletlen * A gombász kiválaszt egy tektont, ahova a spórát lőni szeretné, ez a tekton szomszédos a gombatest tektonjával * A spóraszórás sikeresen végrehajtódik (a spóra rákerül a céltektonra, és törlődik a gombatestből) |
| **bemenet** | A játékos hatására meghívódik a kiválasztott gombatesten a shootSpore metódus, amely paraméterül a kiválasztott céltektont kapja. - ***<Gombatest név> shootSpore <Tektonnév>*** |

| **Use-case neve** | Fejlett gombatest spóra szórása tektonra, a céltekton nem közvetlen szomszéd |
| --- | --- |
| **Rövid leírás** | Azt az esetet teszteli, amikor egy gombász hatására egy mushroom objektum (ami fejlett) spórát szór egy olyan tektonra, ami a szomszédjának a szomszédja. Ebben az esetben ez megtehető a játék szabályai szerint. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * Gombász kiválaszt egy hozzá tartozó gombatestet, ami fejlett * A gombász kiválaszt egy tektont, ahova a spórát lőni szeretné, ez a tekton nem szomszédos, hanem a gombatest szomszédjának szomszédja. * A spóraszórás sikeresen végrehajtódik (a spóra rákerül a céltektonra, és törlődik a gombatestből) |
| **bemenet** | A játékos hatására meghívódik a kiválasztott gombatesten a shootSpore metódus, amely paraméterül a kiválasztott céltektont kapja. - ***<Gombatest név> shootSpore <Tektonnév>*** |

| **Use-case neve** | Fejlett gombatest spóra szórása tektonra, a céltekton közvetlen szomszéd |
| --- | --- |
| **Rövid leírás** | Ugyanaz mint az előzőnél, csak a céltekton közvetlen szomszéd. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | Ugyanaz mint az előzőnél, csak a céltekton közvetlen szomszéd. |
| **bemenet** | ***<Gombatest név> shootSpore <Tektonnév>*** |

| **Use-case neve** | Sikertelen fejletlen spóraszórás, mert nem szomszédos a céltekton. |
| --- | --- |
| **Rövid leírás** | Egy Mushroom (fejlettlen) objektum megpróbál spórát lőni egy tekton célpontra. Mivel nem szomszédos a Mushroom tektonja a céltekton-al, a művelet sikertelen. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * Gombász kiválaszt egy hozzá tartozó gombatestet, ami fejlettlen * A gombász kiválaszt egy tektont, ahova a spórát lőni szeretné, ez a tekton nem szomszédos. * Már tovább se nézzük. A spóraszórás sikertelen (a spóra nem kerül rá a céltektonra, és nem törlődik a gombatestből) |
| **bemenet** | A játékos hívja a fejletlen gombateseten a shootspore-t egy nem szomszédos tektont átadva paraméterül.  ***<Gombatest név> shootSpore <Tektonnév>*** |

| **Use-case neve** | Sikertelen fejletlett spóraszórás, mert se nem közvetetten se nem közvetlenül szomszédos a céltekton. |
| --- | --- |
| **Rövid leírás** | Egy Mushroom (fejlett) objektum megpróbál spórát lőni egy tekton célpontra. Mivel nem szomszédos (közvetlenül vagy közvetetten) a Mushroom tektonja a céltekton-al, a művelet sikertelen. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * Gombász kiválaszt egy hozzá tartozó gombatestet, ami fejlettlen * A gombász kiválaszt egy tektont, ahova a spórát lőni szeretné, ez a tekton nem szomszédos és szomszédjaival sem szomszédos * A spóraszórás sikertelen (a spóra nem kerül rá a céltektonra, és nem törlődik a gombatestből) |
| **bemenet** | A játékos hívja a fejlett gombateseten a shootspore-t egy elég távoli tektont átadva paraméterül.  ***<Gombatest név> shootSpore <Tektonnév>*** |

| **Use-case neve** | Gombatest 10. spórájának elszórása, a gombatest meghal. |
| --- | --- |
| **Rövid leírás** | A gombatest már 9 spórát ellőtt, ezért a következő lövés hatására meghal. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * Gombász kiválaszt egy hozzá tartozó gombatestet, ami már 9-szer lött. * Spórát lő sikeresen. * A gombatest meghal és lekerül a tektonról. |
| **bemenet** | ***<Gombatest név> shootSpore <Tektonnév>*** |

| **Use-case neve** | Gombatest növesztése sikeres. |
| --- | --- |
| **Rövid leírás** | A gombatest növesztéséhez szükséges előfeltételek teljesülnek. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * A gombász kiválaszt egy hozzá tartozó gombafonalat és egy tektont. * Meghívja a **growMushroom()** metódust a kiválasztott fonálon a megfelelő paraméterekkel. * A rendszer ellenőrzi, hogy a tektonon nincs-e már gombatest. * A rendszer lekéri a spórák listáját a tektonról. * A tekton visszaadja a spórák listáját. * Ha mindhárom spóra a kiválasztott fonalhoz tartozik, a rendszer létrehozza a gombatestet.   Eredmény:   * A céltektonon megjelenik az új gombatest a megfelelő attribútumokkal. * A gombatest beállítja a helyét és a fonalát. * A rendszer eltávolítja a spórákat a tektonról. * A growMushroom metódus sikeres. |
| **bemenet** | ***<Fonálneve> growMushroom <Tectonneve> <Mushroomneve>*** |

| **Use-case neve** | A gombafonál megeszi a bénult rovart és új gombatest nő. |
| --- | --- |
| **Rövid leírás** | A gombafonál megeszi a bénító spórát evett rovart és mivel ez nem egy szivó tektonon történt, ezért gombatest keletkezik a rovar helyén. |
| **Aktorok** | játékos (gombász) |
| **Forgatókönyv** | * A gombász kiválaszt egy hozzá tartozó gombafonalat és egy bénult rovart, ami olyan tektonon van, amin a fonala el van ágazva. * A gombász meghívja az eatInsect(bénult rovar) metódust a fonálon. * A rendszer eltávolítja a rovart a tektonról. * A rovar törlődik a rendszerből. * A rendszer ellenőrzi, hogy a tektonon már van-e gombatest. Most nincs. - Ha lenne, a folyamat véget érne. * A rendszer ellenőrzi, hogy a tekton AbsorbingTecton típusú-e. Most nem - Ha az lenne, a folyamat véget érne. * Mivel egyik feltétel sem teljesült, egy új gombatest jön létre. * Az új gombatest pozíciója beállításra kerül a tekton helyére. * A gombatest fonala beállításra kerül a kiválasztott gombafonalra. * A gombatest elhelyeződik a tektonon. * A eatInsect() metódus sikeresen végrehajtódott. |
| **bemenet** | ***<Fonálneve> eatInsect <Rovarneve>*** |

| **Use-case neve** | A gombafonál megeszi a bénult rovarat nem keletkezik új gombatest mert szívótektonon történt az evés. |
| --- | --- |
| **Rövid leírás** | A gombafonál megeszi a bénító spórát evett rovart. Mivel ez egy szívó tektonon történt, ezért nem keletkezik új gombatest a rovar helyén. |
| **Aktorok** | Játékos (gombász) |
| **Forgatókönyv** | A forgató könyv megegyezik a az előzőével, (a happy path-ével) amíg:   * A rendszer ellenőrzi, hogy a tekton AbsorbingTecton típusú-e. Most igen. - A folyamat véget ér, nem jön létre új gombatest. * Ettől még az eatInsect() metódus sikeresen végrehajtódik és true értékkel tér vissza |
| **bemenet** | ***<Fonálneve> eatInsect <Rovarneve>*** |

| **Use-case neve** | A gombafonál megeszi a bénult rovarat nem keletkezik új gombatest mert már van rajta gombatest. |
| --- | --- |
| **Rövid leírás** | A gombafonál megeszi a bénító spórát evett rovart. Mivel a tektonon már van egy gombatest, ezért nem keletkezik új gombatest a rovar helyén. |
| **Aktorok** | Játékos (gombász) |
| **Forgatókönyv** | A forgatókönyv megegyezik az előző (happy path) esettel, amíg:   * A rendszer ellenőrzi, hogy a tektonon már van-e gombatest. Most van. - A folyamat véget ér. * Ettől még az eatInsect() metódus sikeresen végrehajtódott. |
| **bemenet** | ***<Fonálneve> eatInsect <Rovarneve>*** |

| **Use-case neve** | Rovar lehelyezése tektonra sikeres |
| --- | --- |
| **Rövid leírás** | A játékos a játék elején egy rovart helyez le egy tektonra, amin még nincsen rovar. |
| **Aktorok** | Játékos (rovarász) |
| **Forgatókönyv** | * A játékos elvégzi a **putFirstInsect()** metódushívást * A tekton ellenőrzi, hogy nincs-e már rajta rovar. * Mivel a tekton szabad, létrehoz egy új **Rovar** objektumot. * Beállítja a rovar pozícióját saját magára. * Beállítja, hogy a rovar mostantól rajta található. * A tektonon megjelenik az új rovar a megfelelő pozícióval. * A rovarlehelyezés sikeres. |
| **bemenet** | ***<Rovarneve> putFirstInsect <Tectonneve>*** |

| **Use-case neve** | Rovar lehelyezése foglalt tektonra |
| --- | --- |
| **Rövid leírás** | A játék elején a rovarászok első lépése az első rovarjának lehelyezése egy tektonra. Ebben a tesztben azt teszteljük, amikor a lehelyezés egy foglalt tekonra próbálkozik. Ekkor nincs változás |
| **Aktorok** | játékos |
| **Forgatókönyv** | * Rovarász választ egy tektont * A tekton foglalt * Nem helyeződik le rovar |
| **bemenet** | myRovar putFirstInsect foglaltTekton |

| **Use-case neve** | Rovar mozgatása sikeresen |
| --- | --- |
| **Rövid leírás** | A játék során a rovarász lépése lehet az, hogy rovarát átmozgassa egy másik szomszédos tektonra, amire fonál vezet a saját tektonjáról. Sikeres mozgás után a rovar automatikusan megesz egy spórát az új tektonról, ha van rajta. |
| **Aktorok** | játékos |
| **Forgatókönyv** | * Rovarász választ egy tektont * A tekton szomszédos * A a saját és új tekton között van fonal * A rovar átmozog * Nincs spóra az új tektonon |
| **bemenet** | myRovar move üresTekton |

| **Use-case neve** | Rovar mozgása sikertelen |
| --- | --- |
| **Rövid leírás** | A játék során a rovarász lépése lehet az, hogy rovarát átmozgassa egy másik szomszédos tektonra, amire fonál vezet a saját tektonjáról. Sikeres mozgás után a rovar automatikusan megesz egy spórát az új tektonról, ha van rajta. |
| **Aktorok** | játékos |
| **Forgatókönyv** | * Rovarász választ egy tektont * A tekton szomszédos * A a saját és új tekton között nincs fonal * A rovar marad az eredeti helyén |
| **bemenet** | myRovar move fonáltalanTekton |

| **Use-case neve** | Rovar evése mozgás után - SpeedSpore |
| --- | --- |
| **Rövid leírás** | Sikeres mozgás után a rovar automatikusan megesz egy spórát az új tektonról, ha van rajta. Ebben az esetben volt gyorsítóspóra, ezért a rovarász még egyet léphet. |
| **Aktorok** | játékos |
| **Forgatókönyv** | * Rovarász választ egy tektont * A tekton szomszédos * A a saját és új tekton között van fonal * A rovar átmozog * Egy gyorsítóspórát megesz az új tektonról * Megkapja a hatását |
| **bemenet** | myRovar move speedSporeTekton |

| **Use-case neve** | Rovar evése mozgás után - SlowingSpore |
| --- | --- |
| **Rövid leírás** | Sikeres mozgás után a rovar automatikusan megesz egy spórát az új tektonról, ha van rajta. Ebben az esetben volt lassító, ezért a rovar a következő körben nem tud mozogni, csak vágni |
| **Aktorok** | játékos |
| **Forgatókönyv** | * Rovarász választ egy tektont * A tekton szomszédos * A a saját és új tekton között van fonal * A rovar átmozog * Egy lassítóspórát megesz az új tektonról * Megkapja a hatását |
| **bemenet** | myRovar move slowSporeTekton |

| **Use-case neve** | Rovar evése mozgás után - ParalyzingSpore |
| --- | --- |
| **Rövid leírás** | Sikeres mozgás után a rovar automatikusan megesz egy spórát az új tektonról, ha van rajta. Ebben az esetben volt bénítóspóra, ezért a rovar a következő körben nem tud se mozogni, se vágni, tehát teljesen kimarad. |
| **Aktorok** | játékos |
| **Forgatókönyv** | * Rovarász választ egy tektont * A tekton szomszédos * A a saját és új tekton között van fonal * A rovar átmozog * Egy bénítóspórát megesz az új tektonról * Megkapja a hatását |
| **bemenet** | myRovar move paralyzingSporeTekton |

| **Use-case neve** | Rovar evése mozgás után - NoCutSpore |
| --- | --- |
| **Rövid leírás** | Sikeres mozgás után a rovar automatikusan megesz egy spórát az új tektonról, ha van rajta. Ebben az esetben volt vágástalanító, ezért a rovar a következő körben nem vágni, de mozogni igen. |
| **Aktorok** | játékos |
| **Forgatókönyv** | * Rovarász választ egy tektont * A tekton szomszédos * A a saját és új tekton között van fonal * A rovar átmozog * Egy vágástalanító spórát megesz az új tektonról * Megkapja a hatását |
| **bemenet** | myRovar move nocutSporeTekton |

| **Use-case neve** | Tekton kettétörése kör elején |
| --- | --- |
| **Rövid leírás** | Minden kör elején egy tektonnak ketté kell törnie, tehát a kontroller meghívja egy tektonra a break() metódusát. A tesztelés során miután minden játékos elvégezte a lépését, az új kör legelején hogy tesztelni tudjuk a helyes működését a törés nem random fog történni, hanem a játékos adja meg, hogy melyik tektont szeretné kettétörni |
| **Aktorok** | játékos |
| **Forgatókönyv** | Miután minden játékos lépett az új kör kezdésekor a megadott tekton kettétörik, két új tekton keletkezik, az eredeti tekton pedig elpusztul. Ha volt a tektonon bogár az elsőnek létrejöttre kerül, a spórák, amelyek rajta voltak a tektonnal együtt eltűnnek, illetve a fonálrész, amely a tektonon volt elpusztul |
| **bemenet** | Meg kell adni melyik tekton törjön ketté |

| **Use-case neve** | Gombafonál felszívódása 4 kör eltelte után |
| --- | --- |
| **Rövid leírás** | Minden 4. kör eltelte után az AbsorbingTekton-on a gombafonalak fel kell szívódjanak, tehát a kontrollernek meg kell hívnia az AbsorbingTectonok absorb() függvényét |
| **Aktorok** | játékos |
| **Forgatókönyv** | A 3. kör végén, miután az utolsó játékos is befejezte a lépését, mikor a játék átmegy a 4. körbe, a kontroller meg kell hívja az AbsorbingTectonok-on az absorb() metódust |
| **bemenet** | A 3. körben az utolsó játékos be kell fejezze a lépését |

| **Use-case neve** | Gombatest fejletté válása 5 kör után |
| --- | --- |
| **Rövid leírás** | A gombatest lehelyezését követő 5. körben a gombatest fejletlen állapotból fejletté válik |
| **Aktorok** | - |
| **Forgatókönyv** | * A gombatest lehelyezésre kerül * 5 kör eltelik * Állapot átmenet |
| **bemenet** | 5 kör eltelik a lehelyezéstől számítva |

| **Use-case neve** | Játék befejezése egy megadott kör után |
| --- | --- |
| **Rövid leírás** | A játék legelején megadjuk, hogy hány kör után fejeződjön be a játék, ezután a játékosok már nem tudnak többet lépni |
| **Aktorok** | játékos |
| **Forgatókönyv** | A legutolsó kör végén, miután az utolsó játékos is lépett a program játék vége állapotba kell kerüljön |
| **bemenet** | Az utolsó körben az utolsó játékos be kell fejezze a lépését |

| **Use-case neve** | Rovar mozgatása olyan tektonra, amelyen DividingSpore van |
| --- | --- |
| **Rövid leírás** | A játék során a rovarász lépése lehet az, hogy rovarát átmozgassa egy másik szomszédos tektonra, amire fonál vezet a saját tektonjáról. Sikeres mozgás után a rovar automatikusan megesz egy spórát az új tektonról, ha van rajta. Ebben az esetben ez a spóra DividingSpore, ami miatt a rovar osztódni fog és az állapota normálra fog visszaállni. |
| **Aktorok** | játékos (Rovarász) |
| **Forgatókönyv** | * Rovarász választ egy tektont * A tekton szomszédos * A a saját és új tekton között van fonal * A rovar átmozog * DividingSpore van a tektonon |
| **bemenet** | ***<Rovarneve> move <Tectonneve>*** |

| **Use-case neve** | Rovar fonal vágása |
| --- | --- |
| **Rövid leírás** | A játék során a rovarász lépése lehet az, hogy rovarával elvág egy gombafonalat. Ez akkor lehetséges, ha a kiválasztott gombafonál a rovar pozícióját jelölő tektonnal szomszédos tektont választ ki, amin van gombafonal, de gombatest nincs. |
| **Aktorok** | játékos (Rovarász) |
| **Forgatókönyv** | * Rovarász választ egy tektont * A tekton szomszédos * A a saját és új tekton között van fonal * Meghívódik a gombafonál sendToDie metódusa |
| **bemenet** | ***<Rovarneve> cut <Tectonneve>*** |

| **Use-case neve** | Rovar fonal vágása olyan tektonon, amin van gombatest |
| --- | --- |
| **Rövid leírás** | A játék során a rovarász lépése lehet az, hogy rovarával elvág egy gombafonalat. Ez akkor lehetséges, ha a kiválasztott gombafonál a rovar pozícióját jelölő tektonnal szomszédos tektont választ ki, amin van gombafonal, de gombatest nincs. |
| **Aktorok** | játékos (Rovarász) |
| **Forgatókönyv** | * Rovarász választ egy tektont * A tekton szomszédos * A kiválasztott tektonon van fonal * A vágás nem megy végbe |
| **bemenet** | ***<Rovarneve> cut <Tectonneve>*** |

| **Use-case neve** | Körváltás, ha minden játékos befejezte a lépését |
| --- | --- |
| **Rövid leírás** | A játék során a körszámláló akkor lép eggyel, amikor minden játékos végrehajtotta a lépését az adott körben. Ez lehet például az első kör, amikor minden játékos lerakta az első gombatestjét és a hozzá tartozó gombafonalat. A teszt célja annak ellenőrzése, hogy ilyenkor valóban megtörténik-e a körváltás. |
| **Aktorok** | Játékosok |
| **Forgatókönyv** | * Minden játékos lerakja a gombafonalát * Minden játékos lerakja a gombatestét * A körszámláló növekszik |
| **bemenet** | *Megfelelő lépések elvégzése* |

| **Use-case neve** | Gombafonal tényleges elvágása ShortLife esetben |
| --- | --- |
| **Rövid leírás** | A játék során ha egy rovar vág egy gombafonalat, amely ShortLifeThread típusú, akkor az elvágás csak a következő körben történik meg ténylegesen. A teszteset célja annak ellenőrzése, hogy a sendToDie() hívás után csak a következő körben tűnik el a fonál. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | * Egy rovar cut metódusa sikeresen meghívódik * A kiválasztott tektonon ShortLifeThread van * A metódus a kapott tektont behelyezi a life<Tecton, int> HashMap-be kulcsként, értékként pedig egyet állít be * A gombafonalnak timeCheck() metódusa a következő kör elején meghívódik * A HashMap-ben van elem, így értékét csökkenti 1-el, azaz 0 lesz * Ekkor a tárolt tektonról leveszi a fonalat, majd az elemet kitörli a HashMap-ből |
| **bemenet** | ***<Rovarneve> cut <Tectonneve>*** |

| **Use-case neve** | Gombafonal tényleges elvágása LongLife esetben |
| --- | --- |
| **Rövid leírás** | A játék során ha egy rovar vág egy gombafonalat, amely LongLifeThread típusú, akkor az elvágás csak a következő utáni körben történik meg ténylegesen. A teszteset célja annak ellenőrzése, hogy a sendToDie() hívás után csak a következő utáni körben tűnik el a fonál. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | * Egy rovar cut metódusa sikeresen meghívódik * A kiválasztott tektonon LongLifeThread van * A metódus a kapott tektont behelyezi a life<Tecton, int> HashMap-be kulcsként, értékként pedig kettőt állít be * A gombafonalnak timeCheck() metódusa a következő kör elején meghívódik * A HashMap-ben van elem, így értékét csökkenti 1-el, azaz 1 lesz * A gombafonalnak timeCheck() metódusa a következő kör elején ismét meghívódik * A HashMap-ben van elem, így értékét csökkenti 1-el, azaz 0 lesz * Ekkor a tárolt tektonról leveszi a fonalat, majd az elemet kitörli a HashMap-ből |
| **bemenet** | ***<Rovarneve> cut <Tectonneve>*** |

| **Use-case neve** | Spóragenerálás páros körökben |
| --- | --- |
| **Rövid leírás** | A játék során minden páros kör elején minden játékban lévő gombatest automatikusan kap egy új spórát. A teszt célja annak ellenőrzése, hogy egy páratlan körből párosba váltva valóban minden gombatest generál-e egy új spórát. |
| **Aktorok** | - |
| **Forgatókönyv** | * A körszámláló páros számra vált * Minden gombatesten meghívódik a generateSpore metódus * Sikeres generálás esetén true értékkel térnek vissza |
| **bemenet** | *A körök száma párosra vált* |

* 1. ***Tesztelési terv***

| **Teszt-eset neve** | Teszt\_1 |
| --- | --- |
| **Rövid leírás** | Ez a teszteset az első kör lefutását mutatja be, amely során 1 rovarász és 1 gombász kötelezően az első rovarjukat illetve gombatestjüket rakják le helyesen.   * Rovarász : választ t1 Tektont * Gombász választ t2 Tektont   A before és az aftert, a kimentett txt-k alapján komparáljuk |
| **Teszt célja** | Tesztelje a rovar lehelyezése illetve az első gombatest lehelyezése funkciót. |

| **Teszt-eset neve** | Teszt\_2 |
| --- | --- |
| **Rövid leírás** | Ezt a teszt eset a játék egy közbenső körét mutatja be, ahol 5 rovarász és 3 gombász lépéssorozatát mutatja be.  főleg a rovar mozgására koncentrál   * Rovarász1 : rovar mozgatása t1 tekton-ról SpeedSpore-t tartalmazó Tecton-ra, ahol megeszik egy gyorsítóspórát. * Rovarász2 : rovar mozgatása t2 tekton-ról SlowSpore-t tartalmazó Tecton-ra ahol megeszik egy lassítóspórát * Rovarász3 : rovar mozgatása t3 tekton-ról ParalyzingSpore-t tartalmazó Tecton-ra ahol megeszik egy bénítóspórát * Rovarász4 : rovar mozgatása t4 tekton-ról NoCutSpore-t tartalmazó Tecton-ra ahol megeszik egy vágástalanító spórát * Rovarász 5 : rovar mozgatása t5 tekton-ról DividingSpore-t tartalmazó Tectonra, ahol a rovar megeszi ezt a spórát és osztódik * Gombász2 : Gombafonala növesztése t1 tektonról t5 tektonra * Gombász3 : Spóraszórás t2 tektonról t3 tektonra   A before és az aftert, a kimentett txt-k alapján komparáljuk, minden rovarnak megnézzük, hogy megkapta -e a neki való effektet, illetve a pozíciójukat, illetve, hogy a fonal meg a spóra le került -e jó helyre. |
| **Teszt célja** | Leteszteli a rovaroknak a mozgását illetve mind az 5 spórafajta megevését, emellett, leteszteli még egy-egy fejletlen gombatest fonálnövesztését illetve spóralövését. |

| **Teszt-eset neve** | Teszt\_3 |
| --- | --- |
| **Rövid leírás** | Ez a teszteset a játék egy közbenső körét mutatja be, ahol van 1 gombász és 1 rovarász. A teszt a gombász spóraszórására és fonál elágaztatására koncentrál.   * Gombász: spóraszórás szomszédos tektonra (t1) fejletlen gombatestből * Gombász: fonálelágaztatás szomszédos tektonra (t1) * Gombász: fonálelágaztatás az előző tekton szomszédjára (t2). Ez azért lehetséges, mert t1-en volt a gombásznak spórája. Egyébként nem lehetne ugyanabban a körben kétszer elágaztatni fonalat egy gombásznak * Rovarász: rovar mozgatása t1 tektonra, ahol megeszi a gombász egy kilőtt spóráját, ez most ebben a tesztesetben egy lassítóspóra * Rovarász: fonal vágása t1 szomszédján: t2-őn. A fonal fajtája ShortLifeThread, így a következő körben fog törlődni a fonál t2 tektonról |
| **Teszt célja** | A gombász bónusz fonálelágaztatás, spóraszórás tesztelése. A rovar mozgásának, spóra evésének, fonál vágásának tesztelése. |

| **Teszt-eset neve** | Teszt\_4 |
| --- | --- |
| **Rövid leírás** | Körváltás, spóragenerálás, tekton törése, gombatest fejlődése:  Ha minden játékos befejezi a lépését, a körszámláló eggyel növekszik. Páros körben minden gombatest generál egy új spórát. Körváltáskor egy tekton kettétörik, megszakítva a rajta levő fonalat és megsemmisülnek a rajta levő spórák. Azok a gombatestek, amik még nem fejlődtek ki és 5 kör óta lehelyezésre kerültek, fejletté válnak. |
| **Teszt célja** | Ellenőrizni a körszámláló léptetését, minden gombatest spóralistája bővül, tekton törésekor a rajta levő gombafonál rész és a spórák megszűnnek. A gombatest állapota fejletlenről fejletté válik. |

| **Teszt-eset neve** | Teszt\_5 |
| --- | --- |
| **Rövid leírás** | Ez a teszteset a játék egy közbenső körét mutatja be, ahol van 1 gombász és 1 rovarász. A tesztben a 4. körben vagyunk:   * Az Absorbing típusú tektonokról ekkor felszívódik az összes fonál * Ebben a körben is ketté törik egy tekton, amely a tektonon listájában az első * A rovar spóravágásából adódóan maradni fog egy olyan fonál, ami nincs bekötve semmihez, de mivel ez KeepThread típusú tektonon van, ezért ez ekkor nem törlődik |
| **Teszt célja** | Az Absobing típusú tektonokról történő fonalaknak felszívódás, és a KeepThread típusú tektonon való bekötettetlen fonál életben tartásának tesztelése. |

* 1. ***Tesztelést támogató segéd- és fordítóprogramok specifikálása***

-

* 1. ***Napló***

| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| --- | --- | --- | --- |
| 2025.03.29 18:00 | 3 óra | Zelch  Szolár  Fejes  Lekli  Toronyi | Feladatok értelmezése |
| 2025.03.30 10:00 | 10 óra | Zelch | Use-case-k kidolgozása, bemeneti nyelv elkészítése, az új változtatások szerint az osztálydiagram átalakítása, szekvenciadiagramok módosítása, új szekvencia diagramok létrehozása, az utolsó négy use case megírása |
| 2025.03.30 20:30 | 3,5 óra | Fejes | Kiosztott use casek megírása, teszt\_3 teszteset megírása. |
| 2025.03.30. | 4 óra | Lekli | Bemeneti nyelv megírása, kiosztott teszt-eset use casek megírása. |
| 2025.03.30. 14:00 | 5,5 óra | Szolár | Kimeneti nyelv megalkotása. shootSpore,  growMushroom és eatInsect parancsokhoz tartozó use case-ek megírása. Dokumentum formázása. |
| 2025.03.31. 08:00 | 5 óra | Toronyi | Kiosztott use-casek megírása, tesztesetek készítése |
| 2025.03.31. 09:30 | 2,5 óra | Fejes | További tesztesetek megírása. |

**8. Részletes tervek**

26 – five\_guys

Konzulens:

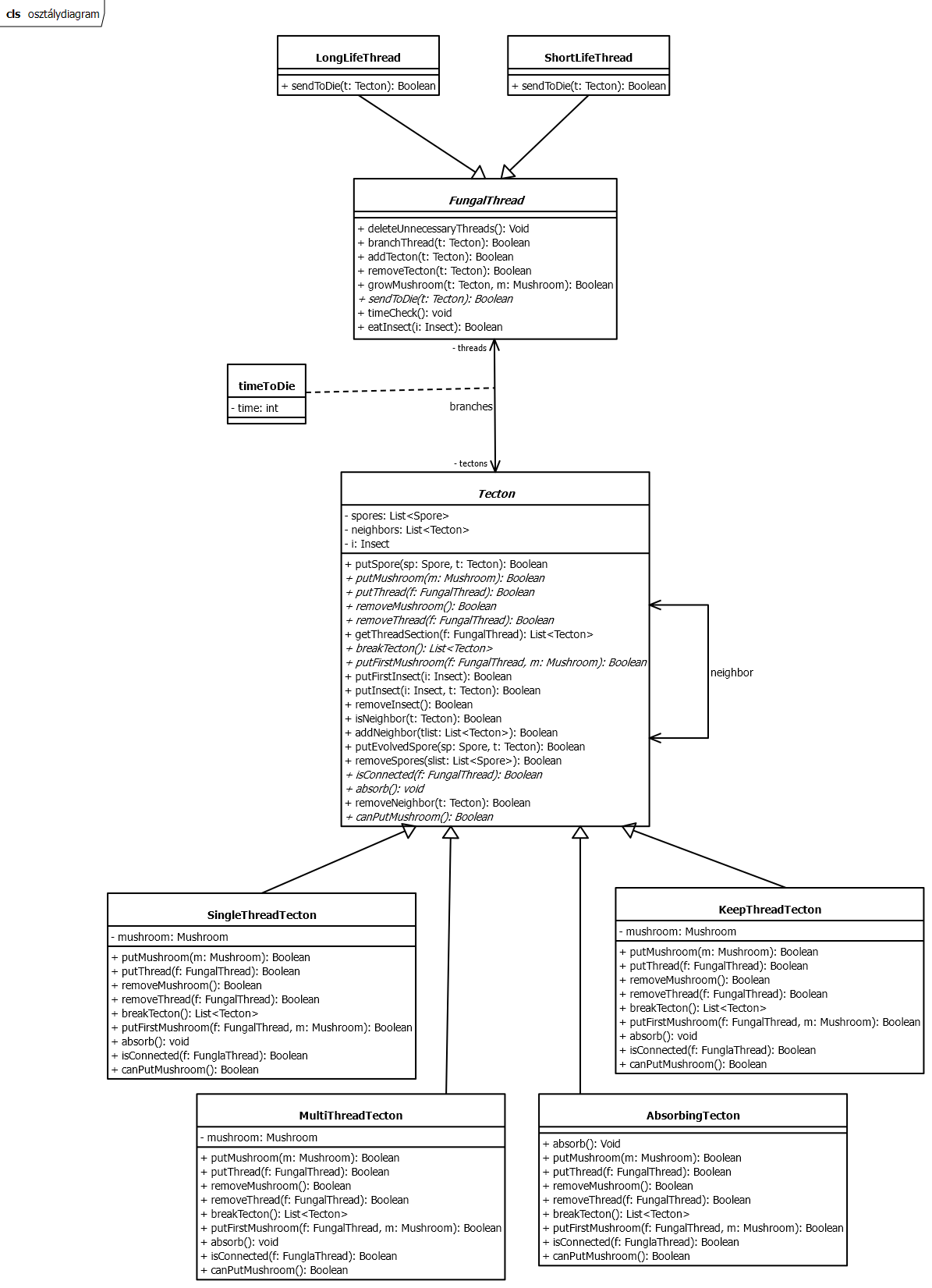
Huszerl Gábor

Csapattagok

| Lekli Zsolt | JENKHQ | zsoltlekli@gmail.com |
| --- | --- | --- |
| Fejes Ábel | AF1Y1C | fejesabel@gmail.com |
| Zelch Csaba | LK0617 | zch03po@gmail.com |
| Szolár Soma | E05SC6 | somaszolar1@gmail.com |
| Toronyi Zsombor | S8F7DV | toronyizsombi@gmail.com |

2025-04-14

# *Változtatások:*

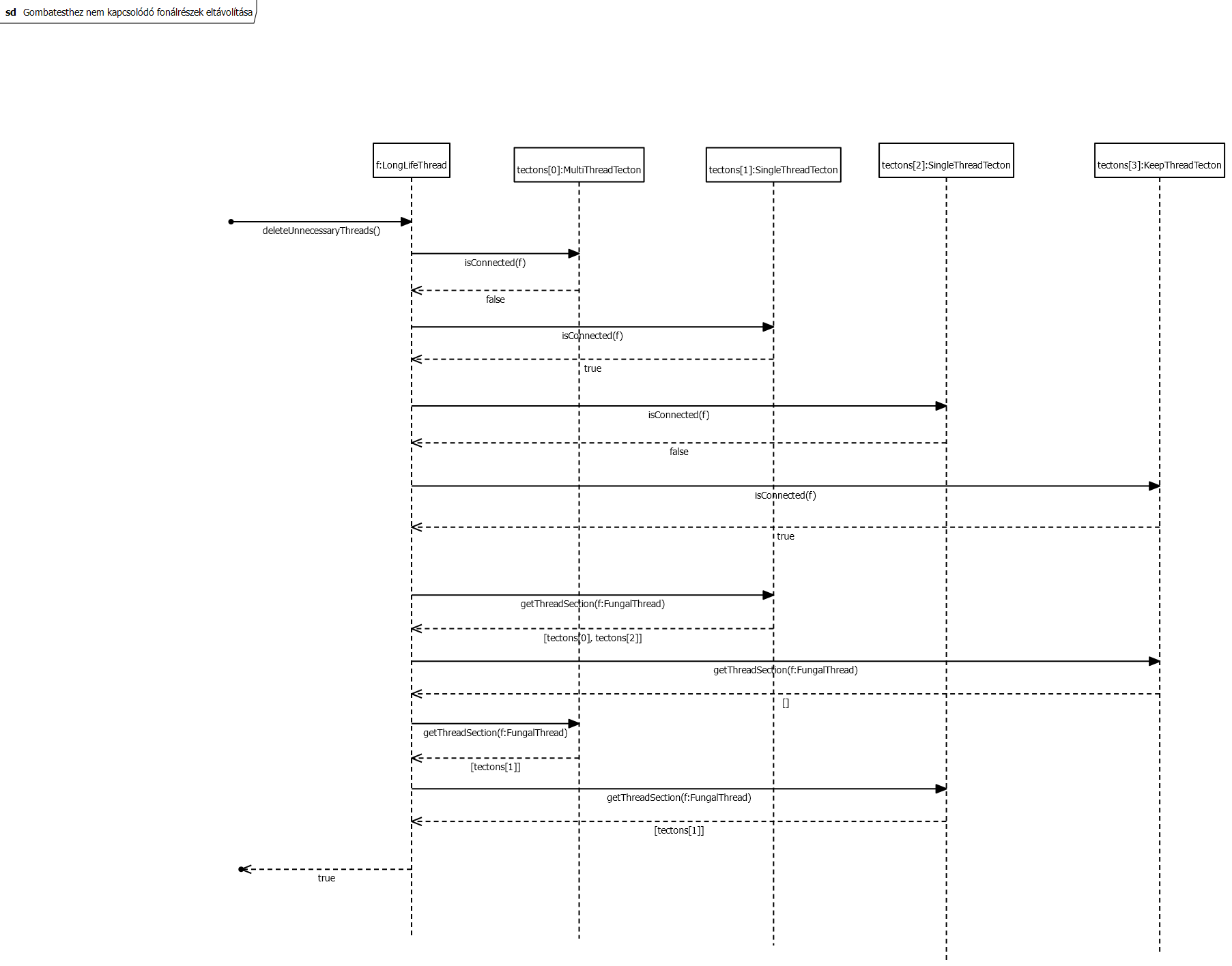
******

A prototípus tervezésénél elkészített osztálydiagramban a következők változtak:

* a **FungalThread** és **Tecton** közötti branch asszociáció egy **asszociációs osztállyal** bővült, ebben tároljuk el, hogy egy adott tektonon a fonál hány kör múlva kell elpusztuljon
* a **branches asszociáció** multiplicitása a különböző tekton leszármazottaknál más:
  + a **MultiThreadTecton, KeepThreadTecton, AbsorbingTecton** osztályok példányai több **FungalThread**-el is kapcsolatban állhatnak, míg a **SingleThreadTecton** osztály példányai nulla vagy egy **FungalThread**-el állnak kapcsolatban
  + a **LongLifeThread** és **ShortLifeThread** osztályok példányai több **Tecton**-al is kapcsolatban állhatnak
* a **Tecton** osztály egy új metódussal bővült az **isConnected(f: FungalThread): Boolean** absztrakt metódussal
  + a metódus leellenőrzi, hogy van-e a tektonon olyan gombatest, amely abból a gombafajból származik, mint az f gombafonál, ha igen true értékkel tér vissza
  + KeepThreadTecton-nál mindig igaz értékkel tér vissza
* Változott a **Tecton putFirstInsect** metódusa, paraméterként egy Insect objektumot kap, így a metóduson belül már nem kell létrehozni a rovart
* Változott a **Tecton putFirstMushroom** metódusa, paraméterként egy gombafonalat és egy gombatestet, így a metóduson belül ezeket nem kell létrehozni
* Változott a **FungalThread growMushroom** metódusa, paraméterként egy tektont és egy gombatestet is kap, így nem kell létrehoznia gombatestet
* Változott a **breakTecton()** függvény, amely visszatérési értékként az újonnan létrejött két tektont adja vissza egy listába belehelyezve
* A Tecton osztály bővült a **canPutMushroom()** metódussal, amely akkor ad vissza igaz értéket, ha a tektonra tehetünk gombatestet, azaz nincs rajta gombatest, illetve nem AbsorbingTecton

# *Megváltozott szekvencia diagramok:*

**Gombatesthez nem kapcsolódó fonálrész eltávolítása**

A deleteUnnecessaryThread függvényhívás érkezhet egy tektontól, egy gombatesttől, vagy akár egy külső kontrollertől

Ezen a diagramon a következő objektumok találhatók:

* f : gombafonál
* tectons[0] : tekton, amelyen az f gombafonál elágazik
* tectons[1] : tekton, amelyen az f gombafonál elágazik
* tectons[2] : tekton, amelyen az f gombafonál elágazik
* tectons[3] : tekton, amelyen az f gombafonál elágazik

Az f gombafonál attribútumként egy listát tárol azokkal a tektonokkal, amelyeken elágazik, ennek neve tectons, a diagramon található tektonok ebből a listából származnak.

A metódus sorban végigmegy a tectons lista tektonjain, és meghívja az isConnected metódusukat. Ez akkor ad vissza true értéket, ha van a tektonon olyan gombatest, amely abból a gombafajból származik, mint az f gombafonál, illetve a KeepThreadTecton mindig true értéket ad vissza. Azokat a tektonokat, amelyek true értékkel tértek vissza eltároljuk.

Miután ezt megcsinálta az eltárolt tektonokon végigmegy, és meghívja rájuk a getThreadSection(f: FungalThread) metódusukat, amely visszaadja azokat a szomszédos tektonokat, amelyeken elágazik az f gombafonál, és ezeket a tektonokat is eltárolja. A kapott tektonokon újra meghívja ezt a metódust, és addig folytatja, amig talál új tektont.

Mikor már nem talál új tektont, azt jelenti, hogy megtalálta az összes olyan fonálrészt, amely közvetve vagy közvetlenül gombatesthez vagy KeepThreadTecton-hoz van kötve, tehát ezeken a gombafonalakat meg kell tartani, és azokon a tektonokon kell törölni, amelyek nincsenek a megtaláltak között, ezek azok a tektonok, amelyek benne vannak a tectons listában, de nincsenek benne a megtalált tektonok között.

# *Bemeneti nyelv változások és kibővítések:*

**saveResult**

Leírás: A result.txt fájlba menti az aktuális állapotot.

Opciók: -

**loadResult**

Leírás:Betölti a result.txt fájlból az állapotot.

Opciók: -

**loadInit <InitFájlneve>**

Leírás: Ahhoz, hogy elkezdhessük a játékot, először fel kell építenünk a pályát, azaz a tektonokat és a szomszédságaikat, amelyet egy adott fájlból fogunk betölteni

Opciók: -

**assert<Elvártkimenetfájlnév>**

Leírás:A result.txt fájlban elmentett állapotát a rendszernek összehasonlítja a tesztesethez tartozó már megadott .txt fájllal, és dönt a helyes működésről

Opciók: -

**runTest<Tesztesetfájlneve>**

Leírás: A parancs során a program megadott .txt fájlból sorra olvassa be és futtatja le a parancsokat, amig az összes parancsot végre nem hajtja a fájlból, azaz a tesztesetet le nem futtatja

Opciók: -

**Arrange parancsok:**

**createTecton <Tektonfajta>**

Leírás: A parancs hatására az adott tektonfajta alapján létrejön egy új tekton, a nevét a kontroller generálja, az alapján, hogy eddig hány tekton volt a játékban, a tekton neveknek a következő a szintaxisa: t<játékban előfordult tektonok száma+1>

Opciók: -

**createShortLifeThread <Tektonnév> <Játékosnév>**

Leírás: Létrehoz egy ShortLifethread fonalat a <Tektonnév> tectonon, illetve odaadja ezt a gombafonalat az adott nevű játékosnak, a fonal nevét a kontroller generálja, az alapján, hogy eddig hány tekton volt a játékban, a gombafonál neveknek a következő a szintaxisa: f<játékban előfordult fonalak száma+1>

Opciók: -

**createLongLifeThread <Tektonnév> <Játékosnév>**

Leírás: Létrehoz egy LongLifethread fonalat a Tektonnév tectonon, illetve odaadja ezt a gombafonalat az adott nevű játékosnak, a fonal nevét a kontroller generálja, az alapján, hogy eddig hány tekton volt a játékban

Opciók: -

**createMushroom <Tektonnév> <Fonálnév> <Játékosnév>**

Leírás: Létrehoz egy gombatestet az adott tektonra, és beállítja, hogy a Fonálnév nevű fonálhoz tartozik, illetve odaadja az adott játékosnak, és beállítja az éltekorát nullára, a gombatest nevét a kontroller generálja, az alapján, hogy eddig hány gombatest volt a játékban, a gombatest neveknek a következő a szintaxisa: m<játékban előfordult gombatestek száma+1>. Ezenkívül növelődik eggyel a <Játékosnév> játékos points attribútuma

Opciók: -

**createEvolvedMushroom <Tektonnév> <Fonálnév> <Játékosnév>**

Leírás: Létrehoz egy fejlett gombatestet az adott tektonra, és beállítja, hogy a <Fonálnév> nevű fonálhoz tartozik, illetve odaadja az adott játékosnak, és beállítja az életkorát 6-ra, a gombatest nevét a kontroller generálja, az alapján, hogy eddig hány gombatest volt a játékban. Ezenkívül növelődik eggyel a <Játékosnév> játékos points attribútuma

Opciók: -

**createInsect <Tektonnév> <Játékosnév>**

Leírás: Létrehoz egy rovart az adott nevű tektonon, majd odaadja az adott nevű játékosnak, a rovar nevét a kontroller generálja, az alapján, hogy eddig hány rovar volt a játékban

Opciók: -

**createFungusPlayers <Játékosnév> <Játékosnév> <Játékosnév> <Játékosnév>**

Leírás: A parancs során létrejönnek a gombászok a megadott nevek szerint, maximum négy nevet adhatunk meg, mivel egyszerre maximum csak négy gombász játszhat a játékkal

Opciók: -

**createInsectPlayers <Játékosnév> <Játékosnév> <Játékosnév> <Játékosnév>**

Leírás: A parancs során létrejönnek a rovarászok a megadott nevek szerint, maximum négy nevet adhatunk meg, mivel egyszerre maximum csak négy rovarász játszhat a játékkal

Opciók: -

**generateSpore <Gombatest név> <Spórafajta>**

**Leírás:** az adott gombatest spores listájába egy <Spórafajta> fajtájú spórát helyez el, a spóra nevét a kontroller generálja, az alapján, hogy eddig hány spóra volt a játékban

**Opciók: -**

**createSpore <Spórafajta> <Tektonnév> <Fonálnév>**

**Leírás:** Ennek során létrejön egy <Spórafajta> fajtájú spóra, amely az adott fonálhoz tartozik, és az adott tektonra kerül, a spóra nevét a kontroller generálja, az alapján, hogy eddig hány spóra volt a játékban

**Opciók: -**

**stepGameRound <Nem negatív egész szám>**

Leírás: A játék köreinek számát lépteti a parancs által megadott egész számmal

Opciók: -

**break <Tektonnév>**

Leírás: A parancs hatására a <Tektonnév>-ben megadott tekton kettétörik, ennek során két új tekton keletkezik, ezeknek neveit a kontroller generálja, az alapján, hogy eddig hány tekton volt a játékban

Opciók: -

**absorb <Tektonnév>**

Leírás: A parancs hatására a <Tektonnév>-ben megadott megadott tektonról felszívódnak a gombafonalak

Opciók: -

**evolve <Gombatestnév>**

Leírás: A parancs hatására a <Gombatestnév>-ben megadott megadott gombatest ha még nem fejlett, akkor fejletté válik

Opciók: -

**deleteUnnecessaryThreads <Fonálnév>**

Leírás: A parancs hatására a megadott fonálnak törlődnek azok a részei, melyek nem kapcsolódnak ugyanolyan fajból származó gombatesthez, vagy KeepThreadTecton-hoz

Opciók: -

**timeCheck**

Leírás: A parancs hatására az összes gombafonalon meghívódik a timeCheck() metódus

Opciók: -

**getState <rovarnév>**

Leírás: A parancs hatására a kapott rovarnak lekérjük az állapotát

Opciók: -

**divide <rovarnév>**

Leírás: A parancs hatására a kapott rovar kettéválik, az új rovar nevét a kontroller generálja, az alapján, hogy eddig hány rovar volt a játékban

Opciók: -

**setMushroomAge <Gombatestnév> <Nem negatív egész szám>**

Leírás: Beállítja a gombatest életkorát, azaz beállítja, hogy hány köre léteziik a gombatest a játékban

Opciók: -

**View felőli interface parancsai:**

**getInsect <Tektonnév>**

Leírás: A parancs hatására a kapott tektonról lekérjük a bogarat

Opciók: -

**getMushroom <Tektonnév>**

Leírás: A parancs hatására a kapott tektonról lekérjük a gombatestet

Opciók: -

**getFungalThread <Tektonnév>**

Leírás: A parancs hatására a kapott tektonról lekérjük a rajta levő fonalakat

Opciók: -

**getSporeTecton <Tektonnév>**

Leírás: A parancs hatására a kapott tektonról lekérjük a spórákat

Opciók: -

**Act parancsok:**

**closestep**

Leírás: A parancsot a játékosok adják ki, amikor le akarják zárni a lépésüket, ezt az első gombatest, illetve az első rovar lehelyezése kivételével minden lépésben ki kell adni

Opciók: -

**putFirstMushroom <Fonáltípus> <Tektonnév>**

Leírás: Gombász első lépése, lehelyez gombatestet adott tektonra, ezzel együtt egy gombafonál is létrejön a tektonon, melynek típusa <Fonáltípus>, a gombatest és a gombafonál nevét a kontroller generálja, az alapján, hogy eddig hány gombatest, illetve gombafonál volt a játékban. Ezenkívül az adott gombásznak növeljük eggyel a points attribútumát

Opciók: -

**putFirstInsect <Tektonnév>**

**Leírás:** A rovarász első lépése, lehelyez egy rovart az adott tektonra, a rovar nevét a kontroller generálja, az alapján, hogy eddig hány rovar, volt a játékban.

Opciók: -

**growMushroom <Fonalnév> <Tektonnév>**

**Leírás:** Adott nevű gombafonál gombatestet növeszt az adott tektonra, a gombatest nevét a kontroller generálja, az alapján, hogy eddig hány gombatest, volt a játékban. Ezenkívül az adott gombásznak növelődik eggyel a points attribútuma

**Opciók: -**

**eatInsect <Fonalnév> <Rovarnév>**

**Leírás:** Adott nevű fonál megeszi az adott rovart. Ha lehetséges, akkor gombatest nő ezen a helyen, ekkor az adott gombásznak növelődik eggyel a points attribútuma.

**Opciók: -**

**move <Rovarnév> <Tektonnév>**

Leírás: Az adott nevű rovar átmozog az adott tektonra. Ha megeszik egy spórát növeljük az adott rovarász points attribútumát eggyel.

Opciók: -

Az act parancsok esetén a parancs leírásának sorrendje felcserélődött, az első helyen mindig a parancs neve szerepel:

**branchThread <Fonalnév> <Tektonnév>**

**shootSpore <Gombatestnév> <Tektonnév>**

**cut <Rovarnév> <Tektonnév>**

# Kimeneti nyelv változtatások:

A kimeneti fájlformátum annyival bővült, hogy nem csak a modellben levő objektumok, hanem a játékos osztályok objektumai és attribútumai is elmentődnek, illetve a fájl végén elmentődik a program állapota, azaz az aktuális körnek a száma, a maximálisan lejátszható körök száma, illetve az aktuális játékos, aki léphet.

A fájlba az objektumokat abc sorrenedben soroljuk fel, illetve ha egy objektumnak olyan attribútuma van, amely egy listát tárol, akkor a listában levő elemek is abc sorrendben fognak kiíródni a fájlba.

Mivel a program a tektonok, gombatestek, gombafonalak, rovarok és spórák neveit beállítja, így mindig, miután egy új neve beállított, kiíratja az adott objektum nevét és fajtáját a konzolra, hogy a játékosok tudják, hogy mi a létrejött objektum neve.

Például új gombatest létrehozása esetén, ha már 3 gombatest volt a játékban, akkor a következőt írja ki a konzolra:

**m4 Mushroom**

Ha a lépés sikertelen, mivel a játékos a játék szabályai miatt nem hajthatja végre, akkor a konzolra kiíródik a következő üzenet:

**Sikertelen**

Mikor a körök száma eléri a maximális körök számát a konzolra kiírjuk a két győztest, és az összes játékost a pontjaival

Például ha van két gombászunk az egyik gombász neve g1 és 3 pontja, a másik gombász neve g2 és 2 pontja, és két rovarászunk, az egyik rovarász neve r1 és 5 pontja, a másik rovarász neve r2 és 6 pontja van, akkor a következő íródik ki a konzolra:

**Nyertesek g1 r2**

**g1 - 3 pont**

**g2 - 2 pont**

**r1 - 5 pont**

**r2 - 6 pont**

# Interfészek:

* **IMushroomController**
  + **Metódusok**

bool shootSpore(Tecton t)

bool evolve()

bool generateSpore(Spore sp)

void setPosition(Tecton t)

void setSpores(List<Spore> slist)

List<Spore> getSpores()

void setThread(FungalThread f)

void setState(MushroomState mState)

void setShootedSporesCount(int count)

int getShootedSporesCount()

* **IFungalThreadController**
  + Metódusok

void deleteUnnecessaryThreads()

bool branchThread(Tecton t)

bool growMushroom(Tecton t, Mushroom m)

void timeCheck()

bool eatInsect(Insect i)

void setTectons(List<Tecton> tlist)

* **ITectonController**
  + Metódusok

List<Tecton> breakTecton()

bool putFirstMushroom(FungalThread f, Mushroom m)

bool putFirstInsect(Insect i)

void absorb()

void setNeighbors(List<Tecton> tlist)

void setMushroom(Mushroom m)

void setInsect(Insect i)

void putThread(FungalThread f)

void setSpores(List<Spore> slist)

* **IInsectController**
  + Metódusok

bool move(Tecton t)

bool cut(Tecton t)

bool divide()

void setPosition(Tecton t)

void setState(InsectState iState)

InsectState getState()

* **ISporeController**
  + Metódusok

FungalThread getThread()

* **ITectonView**
  + Metódusok

Insect getInsect()

Mushroom getMushroom()

List<FungalThread> getThreads()

List<Spore> getSpore()

# *Osztályok és metódusok tervei.*

### Tecton(tekton)

#### Felelősség

Az absztrakt Tecton osztályból származnak le a példányosítható tekton osztályok. A **putSpore, -EvoledSpore, -FirstInsect, -Insect,** a **getThreadSection,** az **add-, is- , removeNeighbor,** és **removeSpores** függvényeket implementálja, ezek közös funkcionalitást képviselnek.

A többi metódus megvalósítását a leszármazottakra hagyja.

#### Ősosztályok

Nincs ősosztálya

#### Interfészek

* + - * + **ITectonController**
        + **ITectonView**

#### Attribútumok

* **- spores : List<Spore>** A tektonon lévő (tektonra kilőtt) spórák.
* **- neighbors : List<Tecton>** A szomszédos Tecton-ok.
* **i- : Insect** Éppen a tektonon lévő rovar.

#### Metódusok

* **+ Tecton()**. spores-t, neighbors-t inicializája, i-t null-al teszi egyenlővé
* **+ putSpore(sp: Spore, t: Tecton) : Boolean** Az átvett sporát hozzávesszi spores listájába a tektonnak. A paraméterül kapott tekton az a tekton, ahonnan lőtték a spórát.
  + pszeudó kód:

FÜGGVÉNY putSpore(spóra, tecton) -> logikai érték

HA tecton NINCS a szomszédok között AKKOR

VISSZATÉR HAMIS

KÜLÖNBEN

spórákhoz\_hozzáad(spóra)

VISSZATÉR IGAZ

VÉGE

VÉGE

* **+ putMushroom(m : Mushroom) : Boolean**. Absztrakt metódus.
* **+ putThread(f : FungalThread) : Boolean**. Absztrakt metódus.
* **+ removeMushroom() : Boolean**. Absztrakt metódus.
* **+ removeThread(f : FungalThread) : Boolean**. Absztrakt metódus.
* **+ getThreadSection(f : FungalThread) : ArrayList<Tecton>**. Visszatér egy ArrayList-el, amiben olyan szomszédos tektonok vannak, amiken rajta van a paraméterként kapott fonál egy része.
  + pszeudó kód:

FÜGGVÉNY getThreadSection(fonal) -> lista<Tecton>

létrehoz üres lista: eredmény\_lista

MINDEN szomszéd Tecton ESETÉN

szál\_lista := szomszéd.getThreads()

HA szál\_lista NEM ÜRES ÉS tartalmazza a 'fonal' elemet AKKOR

hozzáad szomszédot az eredmény\_lista-hoz

VÉGE

VÉGE

VISSZATÉR eredmény\_lista

VÉGE

* **+ breakTecton() : Boolean**. Tekton kettétörését valósítja meg (absztrakt metódus).
* **+ putFirstMushroom(f : FungalThread, m : Mushroom) : Boolean**: Játékos első gombatestjének lehelyezése(absztrakt metódus).
* + **putFirstInsect(i : Insect) : Boolean**. Jatékos első rovarának lehelyezése. Ellenőrzi, hogy van -e már rajta rovar, ha nincs, akkor lehet rá tenni.
  + pszeudó kód:

FÜGGVÉNY putFirstInsect(rovar) -> logikai érték

HA nincs hozzárendelve rovar (i == null) AKKOR

beállítja az 'i' értékét a megadott rovarra

VISSZATÉR IGAZ

KÜLÖNBEN

már van hozzárendelt rovar, nem változtat semmit

VISSZATÉR HAMIS

VÉGE

VÉGE

* **+ putInsect(i: Insect, t: Tecton) : Boolean**. Rovar mozgatása t tektonról. A mozgatás akkor sikeres, ha a t szomszédos, és van vele fonál kapcsolat. Illetve, ha még nincs rajta másik rovar még.
  + pszeudó kód:

// rovarTecton: az a tekton, amin a rovar van.

// A rovar átmozog erre a tektonra.

FÜGGVÉNY putInsect(rovar, rovarTecton) -> logikai érték

// 1. Ha a forrás Tecton nem szomszédos ezzel a Tecton-nal, nem megy

HA rovarTecton NEM szomszédos EZZEL a Tecton-nal AKKOR

VISSZATÉR HAMIS

VÉGE

// 2. Ha EZEN a Tecton-on már van rovar, nem lehet ráhelyezni újat, nem sikeres a mozgatás

HA EZEN a Tecton-on VAN rovar AKKOR

VISSZATÉR HAMIS

VÉGE

// 3. Lekérjük a fonalakat

szálakItt := ez.getThreads()

szálakRovaron := rovarTecton.getThreads()

kapcsolatVan := HAMIS

// 4. Keresünk közös fonalat

MINDEN szál ESETÉN szálakItt-ben:

HA szál szerepel szálakRovaron listában AKKOR

kapcsolatVan := IGAZ

BREAK

VÉGE

VÉGE

// 5. Ha van kapcsolat, mozgatjuk a rovart

HA kapcsolatVan IGAZ AKKOR

rovarTecton-ról eltávolítja a rovart

ide (this) beállítja a rovart

VISSZATÉR IGAZ

KÜLÖNBEN

VISSZATÉR HAMIS

VÉGE

VÉGE

* **+ removeInsect() Boolean**. Rovar eltávolítása tektonról, ha van rajta.
  + pszeudó kód:

FÜGGVÉNY removeInsect() -> logikai érték

HA van hozzárendelt rovar (i NEM null) AKKOR

eltávolítja a rovart (i := null)

VISSZATÉR IGAZ

KÜLÖNBEN

nincs eltávolítandó rovar

VISSZATÉR HAMIS

VÉGE

VÉGE

* **+ isNeighbor(t : Tecton) : Boolean**. Átvesz egy tektont és megnézi, hogy benne van-e neighbor listájában.
  + pszeudó kód:

FÜGGVÉNY isNeighbor(tecton) -> logikai érték

HA tecton benne van a szomszédok listában AKKOR

VISSZATÉR IGAZ

KÜLÖNBEN

VISSZATÉR HAMIS

VÉGE

VÉGE

* **+ addNeighbor(tlist : List<Tecton>) : Boolean**. Szomszédsági lista bővítése a kapott tektonokkal.
* **+ removeNeighbor(tlist : List<Tecton>) :** **Boolean**. A kapott tektonok kitörlése a szomszédok közül.
* **+ putEvolvedSpore(sp : Spore, t : Tecton): Boolean**. Az átvett spórát hozzáveszi a spores listájához. Különbség a **putSpore**-hoz képest: ha a cél Tecton nem közvetlen szomszéd, de egy szomszédunk szomszédja, akkor is engedjük a spórát hozzáadni.
  + pszeudó kód:

FÜGGVÉNY putEvolvedSpore(spóra, célTecton) -> logikai érték

// 1. Ha célTecton közvetlen szomszéd, egyszerűen hozzáadjuk a spórát

HA célTecton benne van a szomszédok listában AKKOR

spórák listához hozzáad(spóra)

VISSZATÉR IGAZ

VÉGE

// 2. Ha nem közvetlen szomszéd, megnézzük a szomszédaink szomszédait

MINDEN szomszéd IN szomszédok:

HA szomszéd.getNeighbors() tartalmazza a célTectont AKKOR

spórák listához hozzáad(spóra)

VISSZATÉR IGAZ

VÉGE

VÉGE

// 3. Ha sehol nincs kapcsolat, nem adunk hozzá spórát

VISSZATÉR HAMIS

VÉGE

* **+ removeSpores(slist : List<Spore>): Boolean**. A kapott spórákat kitörli a spores listájából.
* **+ isConnected(f : FungalThread): Boolean.** Absztrakt metódus.
* + **absorb()**. Absztrakt metódus.

### AbsorbingTecton(*SzívóTekton*)

#### Felelősség

Ő felel a Tecton példányosítható, szívó változatának a reprezentálásáért.

#### Ősosztályok

Tecton -> AbsorbingTecton

#### Interfészek

Csak azokat valósítja meg, amiket az őse.

#### Attribútumok

* **- threads : List<FungalThread>**. Az asszociációnak megfelelően, lehet rajta akár több fonál is.

#### Metódusok

* **+ AbsorbingTecton().** meghívja az ős ctor-át és inicializálja a threads listáját.
* **+ putMushroom(m : Mushroom) : Boolean.** Hamissal tér vissza, mert nem lehet Mushroomot elhelyezni rá.
* **+ putFirstMushroom() : Boolean.** Mindig hamissal tér vissza
* **+ removeMushroom() : Boolean.** Mindig hamissal tér vissza
* **+ getMushroom() : Mushroom.** null-al tér vissza.
* **+ isConnected(f : FungalThread) : Boolean.** Mindig hamissal tér vissza.
* **+ setThreads(list : List<FungalThreads>).** threads = list
* **+ getThreads() : List<FungalThreads>.**
* **+ absorb().** 
  + pszeudo kód**:**

FÜGGVÉNY absorb() -> nincsen return érték

// fonalak kapcsolatainak megszakítása a tektonnal

FOR végig iterál a tekton fonalain

aktuális fonál tekton listájából eltávolít(saját maga)

VÉGE

// tekton kapcsolatának megszakítása a fonalakkal

átmenetiFonálLista = új lista

átmenetiFonálLista-ába az összes hozzáadása(fonál lista) //fonál lista = this.threads

fonál lista kiűrítése

// Fonalak megtisztítása: csak az azonos fajhoz tartozó gombatestekhez kötődő részek maradnak meg

FOR végig iterál az átmenetiFonálLista-án

meghívja: deleteUnnecessaryThreads() az aktuális fonálon

VÉGE

VÉGE

* **+ putThread(f : FungalThread) : Boolean.** 
  + pszeudo kód**:**

FÜGGVÉNY putThread(f : FungalThread) -> logikai érték

FOR végig iterál a szomszéd listáján

HA az aktuális tekton fonal listája tartalmazza f-et AKKOR

fonál listájába felvesz(f)

visszatér igaz

VÉGE

VÉGE

VISSZATÉR HAMIS

VÉGE

* **+ removeThread(f : FungalThread) : Boolean.** Visszatér a művelet sikerességével
* **+ breakTecton() : Boolean** 
  + pszeudo kód**:**

FÜGGVÉNY breakTecton()

// Két új tekton létrehozása

t6 = új AbsorbingTecton

t7 = új AbsorbingTecton

// Megállapítja a töréspontot

centre = (a jelenlegi tekton szomszédainak száma) / 2

// T6 szomszédainak beállítása:

neighborList = új lista

FOR index = 0-tól (centre - 1)-ig:

neighborList-ba bevesz(szomszéd lista [index])

VÉGE

neighborList-ba bevesz(t7)

t6 szomszéd listája = neighborList

neighborList üritése

// T7 szomszédainak beállítása:

neighborList-ba bevesz(t6)

FOR index = centre-tól (a jelenlegi tekton szomszédainak száma - 1)-ig:

neighborList-ba bevesz(szomszéd lista [index])

VÉGE

t7 szomszéd listája = neighborList

neighborList üritése

// A jelenlegi tekton szomszédainak frissítése, az első csoport (0-tól centre-1):

neighborList-ba bevesz(t6)

FOR index = 0-tól (centre - 1)-ig:

aktuálisSzomszéd = szomszéd lista [index]

aktuálisSzomszéd szomszéd listájába beletesz(neighborList)

aktuálisSzomszéd szomszéd listájából kivesz(a jelenlegi tekton) // this

VÉGE

neighborList üritése

// A második csoport (centre-tól az utolsó indexig):

neighborList-ba bevesz(t7)

FOR index = centre-tól (a jelenlegi tekton szomszédainak száma - 1)-ig:

aktuálisSzomszéd = szomszéd lista [index]

aktuálisSzomszéd szomszéd listájába beletesz(neighborList)

aktuálisSzomszéd szomszéd listájából kivesz(a jelenlegi tekton)

VÉGE

neighborList üritése

// A tektonon levő bogár áthelyezése t6-ra

t6 bogara = bogár // this.Insect

bogár pozíciója = t6

// A jelenlegi tekton összes fonáljából eltávolítja saját magát

FOR fonál listáján végig iterál:

az adott fonál tektonjai-ból kivesz(jelenlegi tekton) // this

VÉGE

// ha keletkezett olyan fonálrész, amely a kettétörés során már nem kapcsolódnak

// semmihez, eltávolításra kerülnek

meghívódik a deleteUnnecessaryThreads

VISSZATÉR IGAZ

VÉGE

### SingleThreadTecton(EgyfonalasTekton)

#### Felelősség

Ő felel a Tecton példányosítható, egyfonalas változatának a reprezentálásáért.

#### Ősosztályok

Tecton -> SingleThreadTecton

#### Interfészek - ITectonController - ITectonView

#### Attribútumok

* **- thread : FungalThread**. Maximum 1 fonala lehet.
* **- mushroom : Mushroom**. Maximum egy gomba lehet rajta.

#### Metódusok

* **+ SingleThreadTecton():** meghívja az ős konstruktorát
* **+ putThread(f:FungalThread) : boolean**. Csak akkor vesz fel fonalat és tér vissza igazzal, ha még nincsen rajta és a szomszédján is található megfelelő fonal, különben hamissal tér vissza.

FÜGGVÉNY putThread(gombafonal) -> logikai érték

HA nincs még hozzárendelve gombafonal

ÉS egy szomszédja tartalmaz ugyanilyen fonalat AKKOR

a paraméterként kapott lesz beállítva

VISSZATÉR IGAZ

KÜLÖNBEN

VISSZATÉR HAMIS

VÉGE

VÉGE

* **+ putMushroom(m : Mushroom) : boolean**. Megpróbál felvenni egy gombát, igazzal tér vissza, ha van legalább 5 spóra a tektonon.

FÜGGVÉNY putMushroom(gombatest) -> logikai érték

HA van legalább 5 darab spóra a tektonon

ÉS nincs még gomba hozzárendelve AKKOR

beállítja az 'mushroom' értékét a megadott  
 gombatestre

VISSZATÉR IGAZ

KÜLÖNBEN

VISSZATÉR HAMIS

VÉGE

VÉGE

* **+ removeThread(f : FungalThread) : boolean.** Megpróbálja kitörölni a megadott fonalat magáról. A művelet sikerességével tér vissza.

FÜGGVÉNY removeThread(törlendőfonal) -> logikai érték

HA a törlendőfonal megegyezik a tekton

gombafonalával AKKOR

gombafonal értékét null-ra állítjuk

VISSZATÉR IGAZ

KÜLÖNBEN

VISSZATÉR HAMIS

VÉGE

VÉGE

* **+ removeMushroom() : boolean**. Megpróbálja kitörölni a gombát magáról. A művelet sikerességével tér vissza.

FÜGGVÉNY removeMushroom() -> logikai érték

HA nincs hozzárendelve gombatest AKKOR

VISSZATÉR HAMIS

KÜLÖNBEN

beállítja a ‘mushroom’ értékét null-ra

VISSZATÉR IGAZ

VÉGE

VÉGE

* **+ breakTecton(): boolean**: boolean: Tekton kettétörését valósítja meg. Mint az Absorbing tektonnál, csak SingleThreadTecton-ok jönnek létre és le kell ellenőrizni, hogy nincs-e rajta gomba. Ha van rajta, akkor nem törhet ketté.
* **+ putFirstMushroom(f : FungalThread, m : Mushroom) : Boolean**: Játékos első gombatestjének lehelyezése

FÜGGVÉNY putFirstMushroom(fonal, gombatest) -> logikai érték

HA nincs hozzárendelve gombatest és gombafonal

AKKOR

beállítja a ‘thread’ és ‘mushroom’ értékét

a megadott fonal és gombatest értékekre

VISSZATÉR IGAZ

KÜLÖNBEN

VISSZATÉR HAMIS

VÉGE

VÉGE

* **+ absorb():**
* **+ boolean canPutMushroom() :** Absztrakt metódus.

### MultiThreadTecton (TöbbfonalasTekton)

#### Felelősség

Ez az osztály a Tecton osztály leszármazottja, ő felelős olyan tektonok metódusaiért, amelyeken egyszerre több gombafonál is elágazhat. Eltárolja a rajta lévő gombatestet amiből 1 vagy 0 lehet, illetve, megvalósítja a következő metódusokat:  
**putMushroom, putThread, removeMushroom, removeThread, breakTecton, putFirstMushroom, absorb.**

#### Ősosztályok

Tecton -> MultiThreadTecton

#### Interfészek

* csak az ős alapján

#### Attribútumok

* **mushroom**: A tektonon lévő gombatestet tárolja, amennyiben az létezik, ha nem létezik akkor értéke null.
* Típusa : Mushroom
* Láthatósága : -
* **threads**: A tektonon elágazó gombafonalakat tárolja, mivel ezen a tektonon több gombafonál is lehet.
* Típusa : List<FungalThread>
* Láthatósága : -

#### Metódusok

* **+ MultiThreadTecton().** meghívja az ős ctor-át és inicializálja a threads listáját.
* **+ boolean putMushroom(m : Mushroom) :** A paraméterként kapott gombatestet felveszi a tekton mushroom attribútumába, amennyiben az üres.

FÜGGVÉNY putMushroom(gombatest) -> logikai érték

HA nincs hozzárendelve gombatest (mushroom==null)AKKOR

beállítja az 'mushroom' értékét a megadott  
 gombatestre

VISSZATÉR IGAZ

KÜLÖNBEN

VISSZATÉR HAMIS

VÉGE

VÉGE

* **+ boolean putThread(f : FungalThread) :** A paraméterként kapott gombafonalat felveszi a tekton threads listájába és igazzal tér vissza.
* **+ boolean removeMushroom() :** A rajta lévő gombatestet törli a tektonrol, amennyiben az létezik, ha nem akkor nem csinál semmit.

FÜGGVÉNY removeMushroom() -> logikai érték

HA nincs hozzárendelve gombatest AKKOR

VISSZATÉR HAMIS

KÜLÖNBEN

beállítja a ‘mushroom’ értékét null-ra

VISSZATÉR IGAZ

VÉGE

VÉGE

* **+ boolean removeThread(f: FungalThread) :** A paraméterül kapott gombafonalat megkeresi a threads listában, majd törli azt belőle.

FÜGGVÉNY removeThread(törlendőfonál) -> logikai érték

MINDEN gombafonál in Threads:

HA gombafonál == törlendőfonál AKKOR

törlendőfonál törlése listából

VISSZATÉR IGAZ

CIKLUS VÉGE

VISSZATÉR HAMIS

VÉGE

* **+ List<Tecton> breakTecton() :** A tekton széttörik kettő kisebb többfonalas tektonra, ha az lehetséges, azaz nincs rajta gombatest. Ezeken kívül olyan, mint az AbsorbingTecton-nál.
* **+ boolean putFirstMushroom(f : FungalThread, m : Mushroom) :** A játék elején egy játékos első gombatestjének lehelyezését hajtja végre, paraméterül kapja a beállítandó gombatestet és a hozzá tartozó gombafonalat.

FÜGGVÉNY putFirstMushroom(gombatest) -> logikai érték

HA nincs hozzárendelve gombatest (mushroom==null)AKKOR

beállítja az 'mushroom' értékét a megadott  
 gombatestre

VISSZATÉR IGAZ

KÜLÖNBEN

VISSZATÉR HAMIS

VÉGE

VÉGE

* **+ void absorb() : -**
* **+ boolean isConnected(f : FungalThread)** : a metódus leellenőrzi, hogy van-e a tektonon olyan gombatest, amely abból a gombafajból származik, mint az f gombafonál, ha igen true értékkel tér vissza.

FÜGGVÉNY isConnected(gombafonál) -> logikai érték

HA van hozzárendelve gombatest ÉS

gombatest fonálja egyezik a paraméter fonállal AKKOR

VSSZATÉR IGAZ

KÜLÖNBEN

VISSZATÉR HAMIS

VÉGE

* **+ boolean canPutMushroom() :** akkor ad vissza igaz értéket, ha a tektonra tehetünk gombatestet, azaz nincs rajta gombatest.

### KeepThreadTecton (FonálMegtartóTekton)

#### Felelősség

Ez az osztály a Tecton osztály leszármazottja, ő felelős olyan tektonok metódusaiért, amelyeken egyszerre több gombafonál is elágazhat és megtartja azokat a fonalakat, amelyek nincsenek közvetlenül vagy közvetve gombatesthez kötve.

Eltárolja a rajta lévő gombatestet amiből 1 vagy 0 lehet, illetve, megvalósítja a következő metódusokat:  
**putMushroom, putThread, removeMushroom, removeThread, breakTecton, putFirstMushroom, absorb.**

#### Ősosztályok

Tecton -> KeepThreadTecton

#### Interfészek

* csak az ős alapján

#### Attribútumok

* **mushroom**: A tektonon lévő gombatestet tárolja, amennyiben az létezik, ha nem létezik akkor értéke null.
* Típusa : Mushroom
* Láthatósága : -
* **threads**: A tektonon elágazó gombafonalakat tárolja, mivel ezen a tektonon több gombafonál is lehet.
* Típusa : List<FungalThread>
* Láthatósága : -

#### Metódusok

* **+ KeepThreadTecton().** meghívja az ős ctor-át és inicializálja a threads listáját.
* **+ boolean putMushroom(m : Mushroom) :** A paraméterként kapott gombatestet felveszi a tekton mushroom attribútumába, amennyiben az üres.

FÜGGVÉNY putMushroom(gombatest) -> logikai érték

HA nincs hozzárendelve gombatest (mushroom==null)AKKOR

beállítja az 'mushroom' értékét a megadott  
 gombatestre

VISSZATÉR IGAZ

KÜLÖNBEN

VISSZATÉR HAMIS

VÉGE

VÉGE

* **+ boolean putThread(f : FungalThread) :** A paraméterként kapott

gombafonalat felveszi a tekton threads listájába és igazzal tér vissza.

* **+ boolean removeMushroom() :** A rajta lévő gombatestet törli a tektonrol, amennyiben az létezik, ha nem akkor nem csinál semmit.

FÜGGVÉNY removeMushroom() -> logikai érték

HA nincs hozzárendelve gombatest AKKOR

VISSZATÉR HAMIS

KÜLÖNBEN

beállítja a ‘mushroom’ értékét null-ra

VISSZATÉR IGAZ

VÉGE

VÉGE

* **+ boolean removeThread(f: FungalThread) :** A paraméterül kapott gombafonalat megkeresi a threads listában, majd törli azt belőle.

FÜGGVÉNY removeThread(törlendőfonál) -> logikai érték

MINDEN gombafonál in Threads:

HA gombafonál == törlendőfonál AKKOR

törlendőfonál törlése listából

VISSZATÉR IGAZ

CIKLUS VÉGE

VISSZATÉR HAMIS

VÉGE

* **+ List<Tecton> breakTecton() :** A tekton széttörik kettő kisebb KeepThreadTecton, ha az lehetséges, azaz nincs rajta gombatest. Ezeken kívül olyan, mint az AbsorbingTecton-nál.
* **+ boolean putFirstMushroom(f : FungalThread, m : Mushroom) :** A játék elején egy játékos első gombatestjének lehelyezését hajtja végre, paraméterül kapja a beállítandó gombatestet és a hozzá tartozó gombafonalat.

FÜGGVÉNY putFirstMushroom(gombatest) -> logikai érték

HA nincs hozzárendelve gombatest (mushroom==null)AKKOR

beállítja az 'mushroom' értékét a megadott  
 gombatestre

VISSZATÉR IGAZ

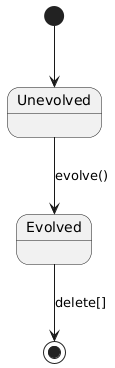
KÜLÖNBEN

VISSZATÉR HAMIS

VÉGE

VÉGE

* **+ void absorb() : -**
* **+ boolean isConnected(f : FungalThread)** : Mindig igazzal tér vissza
* **+ boolean canPutMushroom() :** akkor ad vissza igaz értéket, ha a tektonra tehetünk gombatestet, azaz nincs rajta gombatest.



MushroomState (gombatest állapot)

Enum osztály, amely tartalmazza a gombatestek lehetséges állapotait: Unevolved (fejletlen) és Evolved (fejlett).

### Mushroom (gombatest)

#### Felelősség

A gombatest osztály, felelős a gombatest pozíciójának, illetve a spóráinak listájának eltárolásáért is. Ezen kívül eltárolja a hozzá tartozó gombafonalat, a gombatest állapotát, és azt, hogy eddig hány spórát lőtt ki.

* + - * **Ősosztályok**

Nincs ősosztálya

#### Interfészek

* IMushroomController

#### Attribútumok

* **- position -** Az a tekton, amin a gombatest található
* **- spores-** A testhez tartozó spórák listája
* **- thread -** A testhez tartozó gombafonál
* **- state -** A gombatest állapota
* **- shootedSporesCount -** eddig hány spórát lőtt ki

#### Metódusok

* **+ Mushroom():** inicializálja a **spores** listát és a **state**-et Unevolved-ra állítja
* **+ boolean shootSpore(Tecton t) -** A metódus a paraméterben megadott tektonra kilő egy spórát

FÜGGVÉNY shootSpore(tekton) -> logikai érték

HA a ‘state’ állapot Unevolved értékű AKKOR

meghívjuk a tekton putSpore()

metódusát, ahol a spóra a gombatest legkorábban termelt spórája (0. indexű)

KÜLÖNBEN (HA a ‘state’ Evolved)

meghívjuk a tekton putEvolvedSpore()

metódusát, ahol a spóra a gombatest legkorábban termelt spórája (0. indexű)

VÉGE

* **+ boolean evolve() -** a gombatest a metódus során fejletté válik

(a **state** Evolved lesz)

* **+ boolean generateSpore(Spore sp) -** a paraméterként kapott spóra fajta alapján egy vele megegyező fajtájú spórát hoz létre, amit ki tud majd lőni

FÜGGVÉNY generateSpore(spóra) -> logikai érték

Spóra objektum konstruálása.

A létrejött objektum felvétele a ‘spores’ listába.

VÉGE

### InsectState

#### Felelősség

Enum osztály, amely a rovarnak a lehetséges állapotait tartalmazza, ezek a következők lehetnek: SPEEDBOOST, SLOWED, PARALYZED, NOCUT, NORMAL, DIVIDE

### Insect(rovar)

#### Felelősség

Az Insect osztály felel a rovar lépésének megvalósításáért, vagyis azért, hogy egy másik tektonra lépjen a tektonok között található résen elhelyezkedő gombafonalakon át. Ezenkívül a fonalak elvágásáért is felelős, elvághat olyan gombafonalakat, amelyek a saját tektonja, azaz amelyiken elhelyezkedik és a vele szomszédos tektonok közötti réseken haladnak át. Az rovarnak képesnek kell lennie osztódni, ekkor egy új rovar keletkezik, amely a saját tektonjának az első szomszédjára kerül. Az Insect el kell tárolja azt a tektont, amin elhelyezkedik, illetve az állapotát.

#### Ősosztályok

Nincs ősosztálya.

#### Interfészek

IInsectController

#### Attribútumok

* **position** : Az a tekton, amelyen a rovar elhelyezkedik
  + Láthatóság: -
  + Típus: Tecton
* **state** : Ez egy enum típus, amely tartalmazza a rovar állapotát, ez átlagos esetben normal, de változik, miután a rovar megeszik egy spórát
  + Láthatóság: -
  + Típus: InsectState

#### Metódusok

* **Insect**(): konstruktor, a rovar state attríbutumának értékét beállítja NORMAL-ra
* **void setPosition(Tecton t)** : beállítja a position attríbutumot a kapott tekton szerint
  + Láthatóság: +
* **Tecton getPosition()** : visszatérési értékként visszaadja a position-t
  + Láthatóság: +
* **void setState(InsectState iState)** : beállítja a state attribútumot a kapott állapot alapján
* **InsectState getState()** : visszatérési értékként visszaadja a state attribútum értékét
* **boolean move(Tecton t)** : Ez a metódus felel a rovar lépéséért egy másik tektonra. Paraméterként egy tektont kap, ez lesz a cél, ahova a rovar lépni szeretne. A metódus meghívja a tekton putInsect(i: Insect, t: Tecton) metódusát, amely leellenőrzi, hogyha a rovar odaléphet-e arra a tektonra, majd ha igen, akkor végrehajtja a lépést, és true értékkel tér vissza, ezáltal frissül a rovar pozíciója, és sikeres a lépés
  + Láthatóság: +
  + Pszeudokód:

FÜGGVÉNY move(t)

HA t.putInsect(this, position) AKKOR

position = t

lista = t.getSpores();

HA lista nem üres

lista első eleme.applyEffect(this)

t.removeSpores(lista első eleme)

ELÁGAZÁS VÉGE

VISSZAAD true

KÜLÖNBEN

VISSZAAD false

ELÁGAZÁS VÉGE

FÜGGVÉNY VÉGE

* **boolean cut(Tecton t)** : Ez a metódus felel a fonalak elvágásáért. Paraméterként egy tektont kap, és először leellenőrzi, hogy végre tudja-e hajtani a vágást, azaz a kapott tekton és az a tekton amin elhelyezkedik szomszédosak, illetve a közöttük levő résen van-e gombafonál. Ha a fonalvágás végrehajtható, akkor elvágja a fonalakat, és true értékkel tér vissza, ellenkező esetben pedig false értékkel tér vissza.
  + Láthatóság: +
  + Pszeudokód:

FÜGGVÉNY cut(t)

HA t.isNeighbor(position) AKKOR

lista1 = t.getThreads()

lista2 = position.getThreads()

CIKLUS i=0-tól lista1 elemeinek száma-1-ig

HA list1.get(i) BENNE VAN list2-BEN AKKOR

list1.get(i).sendToDie(t)

ELÁGAZÁS VÉGE

CIKLUS VÉGE

VISSZAAD true

KÜLÖNBEN

VISSZAAD false

ELÁGAZÁS VÉGE

FÜGGVÉNY VÉGE

* **boolean divide()**: A metódus egy új rovart hoz létre, melyet az eredeti rovar tektonjának az első szomszédjára helyez le, ha ez nem lehetséges, mert azon a tektonon már van rovar, tovább lépeget a szomszédain, ha egyetlen szomszédjára se tudta lehelyezni, akkor false-val tér vissza
  + Láthatóság: +
  + Pszeudokód:

FÜGGVÉNY divide()

i2 = ÚJ Insect

lista = position.getNeighbors()

attached = false

i = 0

AMIG i < lista elemeinek száma-1 és attached = false

HA lista[i].getInsect()=null AKKOR

i2.setPosition(lista[i])

lista[i].setInsect(i2)

attached = true

ELÁGAZÁS VÉGE

i = i+1

CIKLUS VÉGE

HA attached = true AKKOR

VISSZAAD true

KÜLÖNBEN

VISSZAAD false

ELÁGAZÁS VÉGE

FÜGGVÉNY VÉGE

### FungalThread(gombafonál)

#### Felelősség

A FungalThread absztrakt osztály, felelős azoknak a tektonoknak a tárolására, amelyeken a gombafonál elágazik, azaz meg kell mutassa, hogy mely tektonokon van elágazva. Ezenkívűl el kell távolítsa a fonál azon részeit, melyek már nincsenek közvetve vagy közvetlenül ugyanolyan fajból származó gombatesthez, vagy KeepThreadTectonon kötve, például, ha egy gombafonál valahol elszakad, meghal egy gombatest vagy a fonál felszívódik a tektonról, gondoskodnia kell ezen fonálrészek eltávolításáról. A GombaFonál felel a fonál növesztéséért, azaz biztosítania kell, hogy a kijelölt tektonon elágazzon a fonál, ha ez lehetséges. Ezenkívül a fonál felel a gombatest növesztéséért is. A gombafonalak el kell tárolják, hogyha elvágták fonalat valahol, akkor hány kör múlva pusztul el arról a tektonról a fonal, és mikor eltelt annyi kör, akkor le kell szedjék a tektonról a fonalat. Képesek ezenkívül tektonjukon található bénult rovarokat megenni.

#### Ősosztályok

Nincs ősosztálya.

#### Interfészek

IFungalThreadController

#### Attribútumok

* **tectons :** Ez egy tektonokból álló lista lesz, segítségével eltároljuk azokat a tektonokat, amelyeken a gombafonál elágazik
  + Láthatóság: -
  + Típus: List<Tecton>
* **life**: Ez egy timeToDie asszociációs osztályokból alló lista, segítségével eltároljuk, hogy hány kör múlva kell az eltárolt tektonokon elpusztuljanak a fonalak
  + Láthatóság: -
  + Típus: List<timeToDie>

#### Metódusok

* **FungalThread():** konstruktor, a life listát létrehozza üres listaként
* **void setTectons(List<Tecton> tlist)** : a tectons lista szettere
* **List<Tecton> getTectons()** : visszaadja a tectons listát
* **void deleteUnnecessaryThreads()** : A metódus sorban végigmegy a tectons listán, meghívja a tektonokra az isConnected metódust, amely true értékkel tér vissza, ha tektonokon van ugyanolyan fajból származó gombatest, illetve KeeppThreadTecton esetén mindig true-val tér vissza. Azokat a tektonokat, melyre true értékkel tért vissza a metódus a program eltárolja, végül megkeresi az összes olyan fonalat, amelyek elérhetők az elmentett tektonokról haladva. Ezek a fonálrészek lesznek azok a részek, amelyeket meg kell tartani, a többit pedig töröljük.
  + Láthatóság: +
  + Pszeudokód:

FÜGGVÉNY deleteUnnecessaryThreads()

CIKLUS i=0-tól tectons elemeinek száma-1-ig

HA tectons.get(i).isConnected(this) AKKOR

fungalList.HOZZÁAD(tectons.get(i))

ELÁGAZÁS VÉGE

CIKLUS VÉGE

AMIG fungalList nem üres

connectedTectons.HOZZÁAD(fungalList.get(0))

lista = fungalList.get(0).getThreadSection(this)

CIKLUS i=0-tól lista elemeinek száma-1-ig

HA list.get(i) nincs benne connectedTectons-ban AKKOR

fungalList.add(list.get(i))

ELÁGAZÁS VÁGE

CIKLUS VÉGE

fungalList.TÖRÖL első elem

CIKLUS VÉGE

CIKLUS i=0-tól tectons elemeinek száma-1-ig

HA tectons.get(i) nincs benne a connectedTectons-ban AKKOR

tectons.get(i).removeThread(this)

ELÁGAZÁS VÉGE

CIKLUS VÉGE

FÜGGVÉNY VÉGE

* **boolean branchThread(Tecton t)** : A metódus a fonál elágazásáért felel. Paraméterként egy tektont kap, amely a cél tekton lesz, vagyis amelyre a fonalat el akarjuk ágaztatni. Ekkor leellenőrzi, hogy a fonál tovább tud-e ágazni arra a tektonra. Ha igen, akkor kapott tektonon is elágaztatja a fonalat, és true értékkel tér vissza, ellenkező esetben false értékkel fog visszatérni.
  + Láthatóság: +
  + Pszeudokód:

FÜGGVÉNY branchThread(t)

HA t.putThread(this) AKKOR

tectons.HOZZÁAD(t)

VISSZATÉR true

KÜLÖNBEN

VISSZATÉR false

ELÁGAZÁS VÉGE

FÜGGVÉNY VÉGE

* **boolean removeTecton(Tecton t) :** A metódus arról felel, hogy a paraméterként kapott tektonnal megegyező tektont kivegye a tectons listából
  + Láthatóság: +
* **boolean addTecton(Tecton t) :** A paraméterként kapott Tectont hozzáveszi a tectons listához
  + Láthatóság: +
* **boolean growMushroom(Tecton t, m: Mushroom)**: A metódus azért felel, hogy a paraméterként kapott tektonra a paraméterként kapott gombatestet elhelyezze, ehhez meg kell néznie, hogy a kapott tektonon elágazik-e fonál, illetve van-e elég spóra
  + Láthatóság: +
  + Pszeudokód:

FÜGGVÉNY growMushroom(t,m)

lista = t.getSpores()

CIKLUS i=0-tól lista elemeinek száma-1-ig

HA lista.get(i).getThread() = this

lista2.HOZZÁAD(lista.get(i))

ELÁGAZÁS VÉGE

CIKLUS VÉGE

HA lista2 elemeinek száma >= 3 AKKOR

m = ÚJ Mushroom

m.setPosition(t)

m.setThread(m)

t.putMushroom(m)

t.removeSpores(lista2)

VISSZATÉR TRUE

KÜLÖNBEN

VISSZATÉR FALSE

ELÁGAZÁS VÉGE

FÜGGVÉNY VÉGE

* **boolean sendToDie(t: Tecton)**: Absztrakt metódus, létrehoz egy timeToDie asszociációs osztályt, melybe a t tektont, és egy számot tárol el, amely megmondja, hogy a tektonon hány kör múlva kell elpusztulnia a fonálnak, majd ezt az osztályt hozzáadja a life listához
  + Láthatóság: +
* **void timeCheck()**: a metódust minden kör elején a kontroller meghívja az összes gombafonálra, és ha a life lista nem üres, akkor csökkenti 1-el a benne levő timeToDie osztályok time attribútumát. Ha az érték elérte a 0-át, akkor a timeToDie által tárolt tektonról leveszi a fonalat meghívva a removeThread(f: FungalThread) metódusát, illetve a gombafonál tectons listájából törli a tectont, és lefuttatja a deleteUnnecessaryThreads() metódust, végül a life listából törli az adott timeToDie objektumot
  + Láthatóság: +
  + Pszeudokód:

FÜGGVÉNY timecheck()

CIKLUS i=0-tól life elemeinek száma-1-ig

lista.get(i).decrease()

CIKLUS VÉGE

i=0

AMIG i < lista elemeinek száma

HA lista.get(i).getTime()=0 AKKOR

lista.get(i).getTecton().removeThread(this)

tectons.remove(lista.get(i).getTecton())

KÜLÖNBEN

i=i+1

ELÁGAZÁS VÉGE

CIKLUS VÉGE

deleteUnnecessaryThreads()

FÜGGVÉNY VÉGE

* **eatInsect(i: Insect):boolean**: leveszi a rovart a tektonról, majd törli a rovart. Abban az esetben, ha a rovar nem AbsorbingTectonon, vagy nem egy olyan tektonon helyezkedett el, ahol már volt gombatest, akkor növeszt egy új gombatestet
  + Láthatóság: +
  + Pszeudokód:

FÜGGVÉNY eatInsect(i)

HA i.getPosition()..removeInsect() AKKOR

VISSZAAD true

KÜLÖNBEN

VISSZAAD false

ELÁGAZÁS VÉGE

FÜGGVÉNY VÉGE

### timeToDie asszociációs osztály

#### Attribútumok

* **t** : tekton, amelyen a gombafonálnak el kell pusztulnia
  + Láthatóság: -
  + Típus: Tecton
* **time**: eltárolja, hogy hány kör múlva kell a gombafonálnak elpusztulnia az eltárolt tektonon
  + Láthatóság: -
  + Típus: int

#### Metódusók

* **void setT(Tecton tect)**: beállítja a t attribútumot a kapott paraméter alapján
  + Láthatóság: +
* **Tecton getT()**: visszaadja a t attribútumot
  + Láthatóság: +
* **void setTime(int t)** : beállítja a time attribútumot a kapott paraméter alapján
  + Láthatóság: +
* **int getTime()** : visszaadja a time attribútumot
  + Láthatóság: +

### LongLifeThread

#### Felelősség

Az osztály a FungalThread osztály leszármazottja, így felelősségei hasonlóak, ezen felül vágás esetén biztosítja, hogy 2 kör után az elvágott fonal elpusztul a tektonról

#### Ősosztályok

FungalThread → LongLifeThread

#### Interfészek

Csak olyanok, amelyet az ősosztályától örököl

#### Attribútumok

Csak amelyeket az ősosztályától örököl

#### Metódusok

* **boolean sendToDie(t: Tecton) :** A metódus létrehoz egy timeToDie asszociációs osztályt, melybe a t tektont, és egy számot tárol el, ennek értéke 2 lesz, amely megmondja, hogy a tektonon két kör múlva kell elpusztulnia a fonálnak, majd ezt az osztályt hozzáadja a life listához
  + Láthatóság: +

### ShortLifeThread

#### Felelősség

Az osztály a FungalThread osztály leszármazottja, így felelősségei hasonlóak, ezen felül vágás esetén biztosítja, hogy 1 kör után az elvágott fonal elpusztul a tektonról

#### Ősosztályok

FungalThread → ShortLifeThread

#### Interfészek

Csak olyanok, amelyet az ősosztályától örököl

#### Attribútumok

Csak amelyeket az ősosztályától örököl

#### Metódusok

* **boolean sendToDie(t: Tecton) :** A metódus létrehoz egy timeToDie asszociációs osztályt, melybe a t tektont, és egy számot tárol el, ennek értéke 1 lesz, amely megmondja, hogy a tektonon egy kör múlva kell elpusztulnia a fonálnak, majd ezt az osztályt hozzáadja a life listához
  + Láthatóság: +

### Spore

#### Felelősség

Ez az absztrakt osztály reprezentálja a spórákat és a hozzájuk tartozó tulajdonságokat. A spórákról tároljuk azt, hogy melyik gombafonálhoz tartoznak. (Egy gombafonál kizárólag egy játékoshoz tartozik.) Ez az osztály azért absztrakt, mert több fajta spóra létezik a játék során, amelyeket leszármazással valósít meg a program. Továbbá az osztály azért is felel, hogy az őt elfogyasztó rovar objektum state attribútumát be tudja állítani az alapján, hogy milyen fajta a példánya. Ez egy absztrakt függvényként jelenik meg ebben az osztályban, amit a leszármazottak fognak megvalósítani.

#### Ősosztályok

Nincs ősosztálya.

#### Interfészek

ISporeController

#### Attribútumok

* + - * **thread**:
        + Az a gombafonál, amihez a spóra tartozik. Fontos, hogy egy gombász játékosnak egy gombafonala van. Így egyértelműen meghatározható, hogy egy játékos fonalához mely spórák tartoznak.
        + Láthatóság: -
        + Típus: FungalThread

#### Metódusok

* **Spore():**
  + Alapértelmezett konstruktor, ami csak létrehoz egy példányt, az attribútumot a setter függvény állítja be.
* **FungalThread getThread():** Visszaadja a thread attribútumot.
* **void setThread(FungalThread t):** Beállítja a thread attribútumot t paraméterre.
* **void applyEffect(i: Insect)**:
  + Absztrakt függvény. Tehát csak a leszármazott osztályok fogják megvalósítani.
  + Láthatóság: +
* **FungalThread getThread():** Visszaadja a thread attribútumot.
  + Láthatóság: +
* **void setThread(thread: FungalThread):** Beállítja a thread attribútumot a kapott paraméter alapján.

Láthatóság: +

### SlowingSpore

#### Felelősség

Azokért a spórákért felelős, amelyek elfogyasztása után a rovar lassult állapotba kerül. Ekkor a rovar a következő körben nem léphet, de vághat fonalat.

#### Ősosztályok

Spore -> SlowingSpore

#### Interfészek

Csak olyanok, amelyet az ősosztályától örököl (ISporeController)

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl. (thread)

#### Metódusok

* **SlowingSpore():** Meghívja az ős konstruktorát.
* **void applyEffect(i: Insect)**:

A paraméterül kapott i Insect objektumon meghívja az Insect setState függvényét, aminek paraméterül Slowed SporeEffectet ad át. Így az i Insect objektumnak a state attribútumát Slowed értékre állítja a függvény.

Láthatóság: +

### SpeedSpore

#### Felelősség

Azokért a spórákért felelős, amelyek elfogyasztása után a rovar gyorsított állapotba kerül. Gyorsított állapotban a rovar még egyszer léphet.

#### Ősosztályok

Spore -> SpeedSpore

#### Interfészek

Csak olyanok, amelyet az ősosztályától örököl (ISporeController)

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl.(thread)

#### Metódusok

* **SpeedSpore():** Meghívja az ős konstruktorát.
* **void applyEffect(i: Insect)**:

A paraméterül kapott i Insect objektumon meghívja az Insect setState függvényét, aminek paraméterül SpeedBoosted SporeEffectet ad át. Így az i Insect objektumnak a state attribútumát SpeedBoosted értékre állítja a függvény.

Láthatóság: +

### ParalysingSpore

#### Felelősség

Azokért a spórákért felelős, amelyek elfogyasztása során a rovar bénult állapotba kerül. Ekkor a rovar a körében már nem csinálhat semmit, és a következő körben sem léphet és nem is vághat fonalat, tehát kimarad egy körből.

#### Ősosztályok

Spore -> ParalysingSpore

#### Interfészek

Csak olyanok, amelyet az ősosztályától örököl (ISporeController)

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl. (thread)

#### Metódusok

* **ParalysingSpore():** Meghívja az ős konstruktorát.
* **void applyEffect(i: Insect)**:

A paraméterül kapott i Insect objektumon meghívja az Insect setState függvényét, aminek paraméterül Paralysed SporeEffectet ad át. Így az i Insect objektumnak a state attribútumát Paralysed értékre állítja a függvény.

Láthatóság: +

### NoCutSpore

#### Felelősség

Azokért a spórákért felelős, amelyek elfogyasztása után a rovar nem tud fonalat vágni. Ekkor ha a körében még nem vágott fonalat a rovar, akkor már a körében sem tud. A következő körben pedig minden esetben nem tud fonalat vágni a rovar.

#### Ősosztályok

Spore -> NoCutSpore

#### Interfészek

Csak olyanok, amelyet az ősosztályától örököl. (ISporeController)

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl.(thread)

#### Metódusok

* **NoCutSpore():** Meghívja az ős konstruktorát.
* **void applyEffect(i: Insect)**:

A paraméterül kapott i Insect objektumon meghívja az Insect setState függvényét, aminek paraméterül NoCut SporeEffectet ad át. Így az i Insect objektumnak a state attribútumát NoCut értékre állítja a függvény.

Láthatóság: +

### DividingSpore

#### Felelősség

Azokért a spórákért felelős, amelyek elfogyasztása után a rovar osztódik, tehát a játékosnak lesz még egy új rovara. Az új rovar az eredeti rovar tektonjának az első szomszédján (ami a tekton neighbors listájában a legelső tekton) jön létre.

#### Ősosztályok

Spore -> DividingSpore

#### Interfészek

Csak olyanok, amelyet az ősosztályától örököl (ISporeController)

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl.(thread)

#### Metódusok

* **DividingSpore():** Meghívja az ős konstruktorát.
* **void applyEffect(i: Insect)**:

A paraméterül kapott i Insect objektumon meghívja az Insect setState függvényét, aminek paraméterül Divided SporeEffectet ad át. Így az i Insect objektumnak a state attribútumát Divided értékre állítja a függvény.

Láthatóság: +

### Controller

### 

#### Felelősség

A játék kezeléséért és a játék lebonyolításáért felelős osztály. Tárolja: az aktuális kört, az utolsó kört (ami után véget ér a játék), a játékosokat, az aktuálisan jövő játékost és minden a játék során szereplő objektumhoz tárolja az objektum nevét (Map<string, object>, ez igazából a teszteléskor történő txt kiírásokhoz szükséges). Az osztály felelőssége továbbá a konzolon írt parancsok feldolgozása és végrehajtása.

#### Ősosztályok

Nincs ősosztály.

#### Interfészek

Nem valósít meg interfészt.

#### Attribútumok

* + - * **round**:

Az aktuális körnek a számát a jelöli. A körök 0-tól indulnak, a 0. körben a játékosok csak lehelyezhetik az első objektumokat (a gombász az első gombatestjét, a rovarász az első rovarát), mást nem csinálhatnak.

Láthatóság: -

Típus: int

* **objects:**

Minden, a játék során szereplő objektumot és a hozzájuk tartozó nevet tárolja. Ez igazából a txt-be történő kiíráshoz szükséges, amit a teszteléshez használunk.

Láthatóság: -

Típus: Map<String, Object>

* **fungusPlayers:**

A játékban résztvevő gombász játékosokat tárolja.

Láthatóság: -

Típus: List<FungusPlayer>

* **insectPlayers:**

A játékban résztvevő rovarász játékosokat tárolja.

Láthatóság: -

Típus: List<InsectPlayer>

* **maxRound:**

Annak a körnek a sorszáma, amelyik után véget ér a játék.

Láthatóság: -

Típus: int

* **currentPlayer**:

Az aktuálisan jövő játékos (aki épp most van “lépésben”).

Láthatóság: -

Típus: Player

* **randomize**:

**Ha ez be van kapcsolva, a program minden egyes alkalomkor, amikor valami random értéket szeretne, a felhasználóhoz fordul, és kér egy értéket**

Ha false, akkor a program nem randomizál, hanem minden egyes alkalomkor, mikor random értékhez nyúlna kikérdez a felhasználóhoz, hogy adjon neki egy értéket. Egyébként ha true, akkor randomizál a program.

Láthatóság: -

Típus: boolean

* **mushroomCount:**

Az objects Map karbantartását segítő attribútum. Segítségével rendelünk nevet a gombatest objektumokhoz.

Láthatóság: -

Típus: int

* **fungalThreadCount:**
  + Az objects Map karbantartását segítő attribútum. Segítségével rendelünk nevet a fonál objektumokhoz.
  + Láthatóság: -
  + Típus: int
* **sporeCount:**
  + Az objects Map karbantartását segítő attribútum. Segítségével rendelünk nevet a spóra objektumokhoz.
  + Láthatóság: -
  + Típus: int
* **insectCount:**
  + Az objects Map karbantartását segítő attribútum. Segítségével rendelünk nevet a rovar objektumokhoz.
  + Láthatóság: -
  + Típus: int
* **tectonCount:**
  + Az objects Map karbantartását segítő attribútum. Segítségével rendelünk nevet a tekton objektumokhoz.
  + Láthatóság: -
  + Típus: int
* **fungalPlayerCount:**

Az objects Map karbantartását segítő attribútum. Segítségével rendelünk nevet a fungalPlayer objektumokhoz.

Láthatóság: -

Típus: int

* **insectPlayerCount:**

Az objects Map karbantartását segítő attribútum. Segítségével rendelünk nevet az insectPlayer objektumokhoz.

Láthatóság: -

Típus: int

* **tList**:
  + A játék pályáját felépítő tektonokat tároló lista.
  + Láthatóság: -
  + Típus: List<Tecton>

#### Metódusok

* + - * **boolean processCmd(String cmd):**
  + Feldolgozza a paraméterül kapott parancsot és végrehajtja azt.
  + pszeudó kód:

FÜGGVÉNY processCmd(cmd: String)

tömb = cmd felszeletelve szóközönként

SWITCH (tömb[0])

//Arrange/create parancsok

CASE “createTecton”: //<Tektonfajta>

// Paraméterek kinyerése

típus = tömb[1]

// Megfelelő objektum inicializálása

tekton = új Tecton

Switch(típus)

CASE “MultiThreadTecton“:

tekton = új MultiThreadTecton

BREAK

CASE “SingleThreadTecton“:

tekton = új SingleThreadTecton

BREAK

CASE “AbsorbingThreadTecton“:

tekton = új AbsorbingThreadTecton

BREAK

DEFAULT:

kiír konzolra: “helytelen parancs”

visszatér

BREAK

//Konténerekbe bele

nev = generált név // ha n tektonunk van, akkor tn+1 lesz

objects-be beletesz(tekton , nev )

tList-be beletesz(tekton)

BREAK

CASE “createSpore”: //<Spórafajta> <Tektonnév> <Fonálnév>

// Paraméterek kinyerése

típus = tömb[1]

tektonNev = tömb[2]

fonalNev = tömb[2]

// Megfelelő objektum inicializálása

spora = új Spore

Switch(típus)

CASE “SlowingSpore“:

spora = új SlowingSpore

BREAK

CASE “SpeedSpore“:

spora = új SpeedSpore

BREAK

CASE “ParalysingSpore“:

spora = új ParalysingSpore

BREAK

CASE “NoCutSpore“:

spora = új NoCutSpore

BREAK

DEFAULT:

kiír konzolra: “helytelen parancs”

visszatér

BREAK

// Asszociációk beállítása

tekton = objects-ből a tektonNev nevű tekton

tekton spóra listájába bele(spora)

fonal = objects-ből a fonalNev nevű fonál

spora fonala = fonal

// Konténerbe bele

nev = generált név

objects-be beletesz(spora , nev )

BREAK

CASE “generateSpore”://<Gombatest név> <Spórafajta>

// Paraméterek kinyerése

gombaNev = tömb[1]

tipus = tömb[2]

// Megfelelő objektum inicializálása

spora = új Spore

Switch(típus)

CASE “SlowingSpore“:

spora = új SlowingSpore

BREAK

CASE “SpeedSpore“:

spora = új SpeedSpore

BREAK

CASE “ParalysingSpore“:

spora = új ParalysingSpore

BREAK

CASE “NoCutSpore“:

spora = új NoCutSpore

BREAK

DEFAULT:

kiír konzolra: “helytelen parancs”

visszatér

BREAK

// Asszociációk beállítása

gomba = objects-ből a gombaNev nevű gomba

gomba spóra listájába bele(spora)

fonal = gomba fonala

spora fonala = fonal

// Konténerbe bele

nev = generált név

objects-be beletesz(spora , nev )

BREAK

CASE “createShortLifeThread”: // <Tektonnév> <Játékosnév>

// Paraméterek kinyerése

tektonNev = tömb[1]

jatekosNev = tömb[2]

// Megfelelő objektum inicializálása

fonal = új ShortLifeThread

// Asszociációk beállítása

tekton = objects-ből a tektonNev nevű tekton

fonal tekton listájába bele(tekton)

tekton fonala(i)ba bele(fonal)

jatekos = objects-ből a jatekosNev nevű játékos

jatekos fonala = fonal

//Konténerekbe bele

nev = generált név

objects-be beletesz(fonal , nev )

BREAK

CASE “createLongLifeThread”: // <Tektonnév> <Játékosnév>

ugyanaz mint az előző, csak LongLifeThread-et hozunk létre

BREAK

CASE “createMushroom”: // <Tektonnév> <Fonálnév> <Játékosnév>

// Paraméterek kinyerése

tektonNev = tömb[1]

fonalNev = tömb[2]

jatekosNev = tömb[3]

// Megfelelő objektum inicializálása

gomba = új Mushroom

gomba kora = 0

// Asszociációk beállítása

tekton = objects-ből a tektonNev nevű tekton

gomba pozíciója = tekton

tekton gombája = gomba

fonal = objects-ből a fonalNev nevű fonál

gomba fonala = fonal

jatekos = objects-ből a jatekosNev nevű játékos

játékos gombáiba bele(gomba)

//Konténerbe bele

nev = generált név

objects-be beletesz(gomba , nev )

BREAK

CASE “createEvolvedMushroom”: // <Tektonnév> <Fonálnév> <Játékosnév>

// Paraméterek kinyerése

tektonNev = tömb[1]

fonalNev = tömb[2]

jatekosNev = tömb[3]

// Megfelelő objektum inicializálása

gomba = új Mushroom

gomba kora = 6

gomba állapota = fejlett

// Asszociációk beállítása

tekton = objects-ből a tektonNev nevű tekton

gomba pozíciója = tekton

tekton gombája = gomba

fonal = objects-ből a fonalNev nevű fonál

gomba fonala = fonal

jatekos = objects-ből a jatekosNev nevű gombász

jatekos gombáiba bele(gomba)

//Konténerbe bele

nev = generált név

objects-be beletesz(gomba , nev )

BREAK

CASE “createInsect”: // <Tektonnév> <Játékosnév>

// Paraméterek kinyerése

tektonNev = tömb[1]

jatekosNev = tömb[2]

// Megfelelő objektum inicializálása

rovar = új Insect

// Asszociációk beállítása

tekton = objects-ből a tektonNev nevű tekton

rovar pozíciója = tekton

tekton rovara = rovar

jatekos = objects-ből a jatekosNev nevű rovarász

jatekos rovaraiba bele(rovar)

//Konténerbe bele

nev = generált név

objects-be beletesz(rovar , nev )

BREAK

CASE “createFungusPlayers”://<Játékosnév><Játékosnév><Játékosnév><Játékosnév>

// Ellenőrizzük a parancs helyességét

HA tömb hossza>5 vagy tömb hossza!=fungusPlayerCount+1 AKKOR

kiír konzolra: “túl sok paraméter”

visszatér

VÉGE

// Objektumok incializálása

FOR (index fut 1-től (tömb hossza - 1)-ig)

jatekos = új FungusPlayer

jatekosNev = tömb[index]

//Konténerekbe bele

objects-be beletesz(jatekos , jatekosNev)

fungusPlayers-be bele(jatekos)

VÉGE

BREAK

CASE “createInsectPlayers”://<Játékosnév><Játékosnév><Játékosnév> <Játékosnév>

// Ellenőrizzük a parancs helyességét

HA tömb hossza>5 vagy tömb hossza!=insectPlayerCount+1 AKKOR

kiír konzolra: “túl sok paraméter”

visszatér

VÉGE

// Objektumok incializálása

FOR (index fut 1-től (tömb hossza - 1)-ig)

jatekos = új InsectPlayer

jatekosNev = tömb[index]

//Konténerekbe bele

objects-be beletesz(jatekos , jatekosNev)

insectPlayers-be bele(jatekos)

VÉGE

BREAK

//Arrange/set parancsok

//tekton setter

CASE “setNeighbors”://<Tektonnév> [Tektonnév] [Tektonnév] [Tektonnév]

// cél objektum előállítása

tektonNev = tömb[1]

tekton = objects-ből a tektonNev nevű Tecton

//attribútum beállítás

FOR (index fut 2-től (tömb hosszának - 1)-ig)

szomszed = objects-ből a tömb[index] nevű Tecton

tekton szomszéd tárolójába beletesz(szomszed)

szomszed szomszéd tárolójába beletesz(tekton)

VÉGE

BREAK

//fonál setter

CASE “setTectons”://<Fonálnév> [Tektonnév] [Tektonnév] [Tektonnév]

// cél objektum előállítása

fonalNev = tömb[1]

fonal = objects-ből a fonalNev nevű FungalThread

// attribútum beállítás

tektonok = új Tecton tároló

FOR (index fut 2-től (tömb hosszának - 1)-ig)

tekton = objects-ből a tömb[index] nevű Tecton

tekton fonál tárolójába beletesz(fonal)

tektonok-ba beletesz(tekton)

VÉGE

fonál tekton tárolója = tektonok

BREAK

//gomba setterek

CASE “setShotSpores”://<Gombatest név> <Nem negatív egész szám>

// cél objektum előállítása

gombaNev = tömb[1]

gomba = objects-ből a gombaNev nevű Mushroom

// attribútum beállítás

gomba kilött spóráinak a száma = tömb[2]

BREAK

CASE “setMushroomAge”://<Gombatestnév> <Nem negatív egész szám>

// cél objektum előállítása

gobaNev = tömb[1]

gomba = objects-ből a gombaNev nevű Mushroom

// attribútum beállítás

gomba kora = tömb[2]

BREAK

//rovar setterek

CASE “setState”://<Rovarnév> <InsectState>

// cél objektum előállítása

rovarNev = tömb[1]

rovar = objects-ből a rovarNev nevű Insect

// attribútum beállítás

tipus = tömb[2]

Switch(tipus)

CASE “SPEEDBOOST“:

rovar állapota = SPEEDBOOST

BREAK

CASE “SLOWED“:

rovar állapota = SLOWED

BREAK

CASE “PARALYZED“:

rovar állapota = PARALYZED

BREAK

CASE “NOCUT“:

rovar állapota = NOCUT

BREAK

CASE “NORMAL“:

rovar állapota = NORMAL

BREAK

DEFAULT:

kiír konzolra: “helytelen állapot”

visszatér

BREAK

BREAK

//controller setterek

CASE “setMaxRound”: //<Pozitív egész szám>

maxRound = tömb[1]

BREAK

CASE “stepGameRound”: //<Pozitív egész szám>

round += tömb[1]

BREAK

CASE “setFungalPlayerCount”: //<Pozitív egész szám>

fungalPlayerCount = tömb[1]

BREAK

CASE “setInsectPlayerCount”: //<Pozitív egész szám>

insectPlayerCount = tömb[1]

BREAK

//Arrange/funkció parancsok

CASE “break”: //<Tektonnév>

// objektum előállítása

tektonNev = tömb[1]

tekton = objects-ből a tektonNev nevű Tecton

// parancs végrehajtása az objektumon

toredekek = tekton kettétörését követően előálló kettő Tecton

HA toredekek = null AKKOR

kiír konzolra: Sikertelen volt a parancs

visszatérés

VÉGE

// új objektumok a konténerekbe

tektonNev1 = generált név

objects-be beletesz(toredekek[0], tektonNev1)

tList-be beletesz(toredekek[0])

tektonNev2 = generált név

objects-be beletesz(toredekek[1], tektonNev2)

tList-be beletesz(toredekek[1])

// régit kivenni

objects-ből kivesz(tekton)

tList-ből kivesz(tekton)

BREAK

CASE “absorb”: //<Tektonnév>

// objektum előállítása

tektonNev = tömb[1]

tekton = objects-ből a tektonNev nevű Tecton

// parancs végrehajtása az objektumon

tekton-on lévő fonalak felszívatása

BREAK

CASE “evolve”: //<Gombatestnév>

// objektum előállítása

gombaNev = tömb[1]

gomba = objects-ből a gombaNev nevű Mushroom

// parancs végrehajtása az objektumon

gomba állapota = EVOLVED

gomba kora = 6

BREAK

CASE “deleteUnnecessaryThreads”: //<Fonálnév>

// objektum előállítása

fonalNev = tömb[1]

fonal = objects-ből a fonalNev nevű FungalThreads

// parancs végrehajtása az objektumon

fonal eltörlődik a már nem bekötött tektonokon//deleteUnnecessaryThread()

BREAK

CASE “divide”: // <rovarnév>

// objektum előállítása

rovarNev = tömb[1]

rovar = objects-ből a rovarNev nevű Insect

// parancs végrehajtása az objektumon

utod = rovar kettéválását követően előálló új Insect

HA utod = null AKKOR

kiír konzolra: Sikertelen volt a parancs

visszatérés

VÉGE

// új objektumok a konténerekbe

utodNev = generált név

objects-be beletesz(utod, utodNev)

// régit most nem kell kivenni

BREAK

CASE “timeCheck”:

FOR végig íterál a fungusPlayers-en

fonal = aktuális FungusPlayer fonala

fonal-on meghívódik a timeCheck metódus

VÉGE

BREAK

CASE "saveResult" AKKOR

objektumok rendezése név szerint növekvő sorrendbe

// Az objektumokat és attribútumait fájlba írja

// objektumok->attribútum nevek->attribútum érték alapján rendezve

fájlba kiírás("result.txt", objektumok)

BREAK

// egyéb parancsok

CASE "loadResult"

allapotok = fájlt beolvas("result.txt")

allapotok kiírása a konzolra

BREAK

CASE “loadInit”: //<InitFájlnév>

// Beolvassa az <InitFájlnév> fájlt

utasitasok = fájlt beolvas(<InitFájlnév>)

utasitasTomb = utasitasok felszeletelve szóközönként

FOR utasitasTomb-ön végig iterálva

utasitas = utasitasTomb aktuális eleme

feldolgozó parancs meghívás utasitas-t átadva paraméterül

VÉGE

//végül előáll a megfelelő alapállapot

BREAK

CASE “runTest”: //<Tesztesetfájlneve>

AMIG el nem érted a fájl végét

utasitás = kiolvas file sora

feldolgozó parancs meghívás utasítást átadva paraméterül

VÉGE

BREAK

CASE “turnOffRandom”:

randomize = false

BREAK

CASE “turnOnRandom”:

randomize = true

BREAK

CASE “assert”: //<Tesztesetnév>

tesztesetSzama = tömb[1]

result.txt összehasonlÍtása tesztesetSzama-nak megfelelő .txt fájlal

BREAK

//act parancsok

CASE “closeStep”:

setCurrentPlayer()

BREAK

CASE “putFirstMushroom”//<Gombatest név> <Fonálnév> <Fonáltípus> <Tektonnév>:

gombaNev = tömb[1]

fonalNev = tömb[2]

fonalTipus = tömb[3]

tektonNev = tömb[4]

fonal = objects-ből a fonalNev nevű, fonalTipus típusú gombafonal

gomba = objects-ből a gombaNev nevű gombatest

tekton = objects-ből a tektonNev nevű tekton

tekton.putFirstMushroom(fonal, gomba)

BREAK

CASE “branchThread”: // <Fonal név> <Tektonnév>

HA currentPlayer.branchThread igaz, AKKOR

//Objektum előállítása

fonalNev = tömb[1]

tektonNev = tömb[2]

fonal = objects-ből a fonalNev nevű FungalThread

tekton = objects-ből a tektonNev nevű Tecton leszármazottputfirstinsect

//Parancs végrehajtása

fonal.branchThread(tekton)

currentPlayer.branchThread = false

VÉGE

BREAK

CASE “shootSpore”: //<Gombatest név> <Tektonnév>

//Objektum előállítása

gombaNev = tömb[1]

tektonNev = tömb[2]

gombal = objects-ből a gombaNev nevű Mushroom

tekton = objects-ből a tektonNev nevű Tecton leszármazott

//Parancs végrehajtása

gomba.shootSpore(tekton)

BREAK

CASE “growMushroom <Fonal név> <Gombatest név> <Tektonnév>”:

HA nincs gomba a tektonnév tektonon AKKOR

//Objektum előállítása

fonalNev = tömb[1]

gombaNev = tömb[2]

tektonNev = tömb[3]

gomba = objects-ből a gombaNev nevű Mushroom

fonal = objects-ből a fonalNev nevű FungalThread

tekton = objects-ből a tektonNev nevű Tecton leszármazott

//Parancs végrehajtása

fonal.growMushroom(tekton,gomba)

BREAK

CASE “eatInsect”: //<Fonal név> <Rovarnév>

//Objektum előállítása

fonalNev = tömb[1]

rovarNev = tömb[2]

fonal = objects-ből a fonalNev nevű FungalThread

rovar = objects-ből a rovarNev nevű Insect

//Parancs végrehajtása az objektumon

HA fonal.eatInsect(rovar) igaz AKKOR

HA rovar.getPosition().canPutMushroom() igaz AKKOR

gombatest létrehozása a tektonon

objects tömbbe való elhelyezés a getNewMushroomName segítségével

currentPlayer.points++

BREAK

CASE “putFirstInsect <Rovarnév> <Tektonnév>”:

rovarNev = tömb[1]

tektonNev = tömb[2]

rovar = objects-ből a rovarNev nevű rovar

tekton = objects-ből a tektonNev nevű tekton

tekton.putFirstInsect(rovar, tekton)

BREAK

CASE “move”: // <Rovarnév> <Tektonnév>

// Paraméterek kinyerése

rovarNev = tömb[1]

tektonNev = tömb[2]

// Megfelelő objektumok előszedése

tekton = objects-ből a tektonNev nevű tekton

rovar = objects-ből a rovarNev nevű rovar

// Segéd objektumok

sporaLista = tekton spóra listája

rovarJatekos = mostani játékos

rovarJatekosAssoc = rovarJatekos asszociációjai közül az, amelyikben

szerepel a rovar

//Az ő rovarával akar lépni?

HA rovarJatekosAssoc null AKKOR visszatér VÉGE

// Tud lépni?

HA rovar lépett igaz AKKOR visszatér VÉGE

//Lesz evés?

eszik = hamis

HA sporaLista nem üres AKKOR eszik = igaz VÉGE

// Léptetés

rovar átlép(tekton)

rovarJatekosAssoc lépett értéke = igaz

HA sikeres volt AKKOR

//evett?

HA eszik = igaz AKKOR

spora = sporaLista első eleme

objects-ből kivesz(spora)

rovarJatekos pontszám növelése

VÉGE

VÉGE

HA rovar állapota = kettévált AKKOR

rovar kettévál

rovarMasik = rovar-ból kiváló másik rovar

tektonSzomszed = az első olyan szomszédja a tekton-nak amire

le lehet helyezni egy új rovart

rovarMasik pozíciója = tektonSzomszed

rovarMasikNev = generált név

objects-be bevesz(rovarMasik ,rovarMasikNev)

rovarJatekos asszociációjaiba felvesz(rovarMasik, nem lépett, nem vágott)

VÉGE

HA rovar állapota = gyorsított AKKOR

rovarJatekosAssoc lépett értéke = hamis

VÉGE

HA rovar állapota = nincs vágás VAGY paralizált AKKOR

rovarJatekosAssoc vágott értéke = igaz

VÉGE

BREAK

CASE “cut”: // <Rovarnév> <Tektonnév>

// Paraméterek kinyerése

rovarNev = tömb[1]

tektonNev = tömb[2]

// Megfelelő objektumok előszedése

tekton = objects-ből a tektonNev nevű tekton

rovar = objects-ből a rovarNev nevű rovar

// Segéd objektumok

rovarJatekos = mostani játékos

rovarJatekosAssoc = rovarJatekos asszociációjai közül az, amelyikben

szerepel a rovar

//Az ő rovarával akar vágni?

HA rovarJatekosAssoc null AKKOR visszatér VÉGE

// Tud vágni?

HA rovarJatekosAssoc vágott igaz AKKOR visszatér VÉGE

//Vágás

rovar vág(tekton)

rovarJatekosAssoc vágott = igaz

BREAK

CASE “initRound”:

Meghívódik az initRound metódus.

BREAK

DEFAULT:

kiír konzolra: “helytelen parancs”

visszatér

BREAK

VÉGE

VÉGE

* **void initRound() :**

[informális leírás]: A kontroller növeli eggyel a round attribútumot, ellenőrzi, ha még nem értük-e el a maximális körszámot, meghívja az összes fonál timeCheck() metódusát, beállítja a rovarászok által tárolt cut és moved attribútumokat a rovarok állapotai alapján, illetve ha valamely rovar állapota nem NORMAL, akkor erre állítja, ezután beállítja a gombászok branchThread attribútumát false-ra, és a tárolt age attribútumokat növeli eggyel. Ha valamelyik érték elérte az 5-öt, akkor az ennek megfelelő gombatestnek meghívódik evolve() metódusa és fejetté válik. Ha a kör páros minden gombatestnek a program generál egy új spórát, ha a random generálás ki van kapcsolva, akkor ennek típusa mindig SpeedSpore, ellenkező esetben a program random dönt a típusáról, ha a körnek a száma osztható 4-el, minden AbsorbingTectonon felszívódnak a gombafonalak, illetve ha a random generálás ki van kapcsolva az első tekton kettétörik, ellenkező esetben a program random választ egyet.

Láthatóság: +

* pszeudo-kód:

FÜGGVÉNY initRound():

round++;

HA round < maxRound

FOR i = 0-tól i= fungusPlayers.size -1-ig:

fungusPlayers.getFungalThread().timeCheck()

VÉGE

FOR i = 0-tól i = insectPlayers.size - 1-ig:

FOR j = 0-tól j = insectPlayers[i].insects.size - 1-ig:

InsectState state =insectPlayers[i].insects[j].insect.getState()

HA state == Slowed

moved = false

cut = true

insectPlayers[i].insects[j].insect.setState(Normal)

HA state == Paralysed

moved = false

cut = false

insectPlayers[i].insects[j].insect.setState(Normal)

HA state == NoCut

moved = true

cut = false

insectPlayers[i].insects[j].insect.setState(Normal)

HA state == Divided

insectPlayers[i].insects[j].insect.setState(Normal)

HA state == SpeedBoosted

insectPlayers[i].insects[j].insect.setState(Normal)

VÉGE

VÉGE

FOR i = 0-tól i= fungusPlayers.size - 1-ig:

fungusPlayer[i].branchThread = false

FOR j = 0-tól j = fungusPlayers [i].mushrooms.size()-1-ig

int mAge = fungusPlayers[i].mushrooms[j].age++;

HA mAge == 5

fungusPlayers[i].mushrooms[j].mushroom.evolve()

VÉGE

HA round mod 2 == 0

Spore spore

HA randomize

int randomNum = randomize(5)

SWITCH(randomNum)

CASE 0:

spore = új SlowingSpore

BREAK

CASE 1:

spore = új SpeedSpore

BREAK

CASE 2:

spore = új ParalysingSpore

BREAK

CASE 3:

spore = új NoCutSpore

BREAK

CASE 4:

spore = új DividingSpore

BREAK

VÉGE

VÉGE

EGYÉBKÉNT

spore = új SpeedSpore

VÉGE

Spore.setThread(fungusPlayers[i].FungalThread.getThread())

fungusPlayers[i].mushrooms[j].mushroom.generateSpore(Spore)

VÉGE

VÉGE

VÉGE

HA round mod 4 == 0

FOR i = 0-tól i = tList.size -1 -ig

tList[i].absorb()

VÉGE

VÉGE

HA randomize

int randNum = randomize(tList.size)

tList[randNum].breakTecton()

VÉGE

EGYÉBKÉNT

tList[0].breakTecton()

VÉGE

VÉGE

EGYÉBKÉNT

FungusPlayer fWinner

InsectPlayer iWinner

int fMaxPoint = 0

int iMaxPoint = 0

FOR i = 0-tól i = fungusPlayers.size -1

HA fungusPlayers[i].point > fMaxPoint

fWinner = fungusPlayers[i]

fMaxPoint = fungusPlayers[i].point

VÉGE

KIÍR KONZOL

fungusPlayers[i].name - fungusPlayers[i].point

VÉGE

VÉGE

FOR i = 0-tól i = insectPlayer.size -1

HA insectPlayers[i].point > fMaxPoint

iWinner = insectPlayers[i]

iMaxPoint = insectPlayers[i].point

VÉGE

KIÍR KONZOL

insectPlayers[i].name - insectPlayers[i].point

VÉGE

VÉGE

KIÍR KONZOL

Nyertesek: fWinner iWinner

VÉGE

VÉGE

FÜGGVÉNY VÉGE

* **void setCurrentPlayer():**
  + Láthatóság: +
  + Beállítja a következő játékost, aki következik.
  + Pszeudo-kód:

FÜGGVÉNY setCurrentPlayer():

int indexCurrentPlayer = -1

HA currentPlayer benne van a fungusPlayers listában

FOR int i = 0-tól i = fungusPlayers.size -1-ig

HA currentPlayer == fungusPlayers[i]

indexCurrentPlayer = i

VÉGE

VÉGE

HA indexCurrentPlayer == fungusPlayers.size-1

currentPlayer = insectPlayer[0]

VÉGE

EGYÉBKÉNT

currentPlayer = fungusPlayers[indexCurrentPlayer + 1]

VÉGE

VÉGE

EGYÉBKÉNT

FOR int i = 0-tól i = insectPlayers.size -1-ig

HA insectPlayer == insectPlayers[i]

indexNextPlayer = i

VÉGE

VÉGE

HA indexCurrentPlayer == insectPlayers.size-1

initRound()

currentPlayer = fungusPlayers[0]

VÉGE

EGYÉBKÉNT

currentPlayer = insectPlayers[indexCurrentPlayer + 1]

VÉGE

VÉGE

FÜGGVÉNY VÉGE

* **void String getNewMushroomName():**
  + Láthatóság: +
  + Visszaad egy új gombatest nevet, ami még nem volt használva. Az objects Map karbantartására használjuk.
  + pszeudo-kód:

FÜGGVÉNY getNewMushroomName()

mushroomCount++

newName = “m” + mushroomCount

RETURN newName

FÜGGVÉNY VÉGE

* **void String getNewThreadName():**
  + Láthatóság: +
  + Visszaad egy új gombafonál nevet, ami még nem volt használva. Az objects Map karbantartására használjuk.
  + pszeudo-kód:

FÜGGVÉNY getNewThreadName()

fungalThreadCount++

newName = “m” + fungalThreadCount

RETURN newName

FÜGGVÉNY VÉGE

* **void String getNewSporeName():**
  + Láthatóság: +
  + Visszaad egy új spóra nevet, ami még nem volt használva. Az objects Map karbantartására használjuk.
  + pszeudo-kód:

FÜGGVÉNY getNewSporeName()

sporeCount++

newName = “m” + sporeCount

RETURN newName

FÜGGVÉNY VÉGE

* **void String getNewInsectName():**
  + Láthatóság: +
  + Visszaad egy új rovar nevet, ami még nem volt használva. Az objects Map karbantartására használjuk.
  + pszeudo-kód:

FÜGGVÉNY getNewInsectName()

insectCount++

newName = “m” + insectCount

RETURN newName

FÜGGVÉNY VÉGE

* **void String getNewTectonName():**
  + Láthatóság: +
  + Visszaad egy új tekton nevet, ami még nem volt használva. Az objects Map karbantartására használjuk.
  + pszeudo-kód:

FÜGGVÉNY getNewtectonName()

tectonCount++

newName = “m” + tectonCount

RETURN newName

FÜGGVÉNY VÉGE

* **int randomize(int domain):**
  + Láthatóság: +
  + Visszaad a paraméternek megfelelő tartományban egy random generált egész számot. Fontos, hogy a random generálás nem ebben a függvényben történik, a függvény a view felől egy interfészen keresztül kéri a random generált egész számot.
  + Pszeudo-kód:

FÜGGVÉNY randomize(int domain)

int randNum = View felől generált random egész szám

RETURN randNum mod domain

FÜGGVÉNY VÉGE

### Player

#### Felelősség

A gombász és a rovarász játékosok közös tulajdonságainak reprezentálása.

#### Ősosztályok

Nincsen ősosztálya.

#### Interfészek

Nem valósít meg interfészeket.

#### Attribútumok

* **- points : int.** Ajátékos teljesítményét reprezentálja. Értelmezése játékos típustól függ.
* **- name : String.** A játék elején meg lehet adni, tesztelés közben lehet rá hívatkozni, a játék végén kiíratásra kerülnek a hozzájuk tartozó pontokkal együtt.

#### Metódusok

Nincsenek metódusai.

### InsectPlayer

#### Felelősség

A rovarász játékosok adatainak tárolása.

#### Ősosztályok

InsectPlayer -> Player

#### Interfészek

Nem valósít meg interfészeket.

#### Attribútumok (öröklötteken kívül)

* **- i : List <Insect>.**  Rovarász által irányított rovarok.
* **- moved : Boolean.** Lépett-e már a körben.
* **- cut : Boolean.** Vágott-e már fonalat a körben.

#### Metódusok

Nincsenek metódusai.

### FungusPlayer

#### Felelősség

A gombász játékosok adatainak tárolása.

#### Ősosztályok

FungusPlayer -> Player

#### Interfészek

Nem valósít meg interfészeket.

#### Attribútumok (öröklötteken kívül)

* **- thread : FungalThread.**  Gombász által birtokolt fonál.
* **- branchThread : Boolean.** Ágaztatott-e már a körben.
* **- mushrooms : List<Mush>.** Birtokolt gombák + a koruk.

#### Metódusok

Nincsenek metódusai.

### Mush (asszociációs osztály)

#### Felelősség

Kapcsolat megteremtése a gombásszal és a gombatest között. Gombatest korának eltárolása.

#### Ősosztályok

Nincsen ősosztálya.

#### Interfészek

Nem valósít meg interfészeket.

#### Attribútumok (öröklötteken kívül)

* **- mushroom : Mushroom.** Asszociációban szereplő gombatest.
* **- age : int.** Test kora.

#### Metódusok

Nincsenek metódusai.

## *A tesztek részletes tervei, leírásuk a teszt nyelvén*

### 0. kör tesztelése

* **Leírás**

Ez a teszteset a nulladik kör lefutását mutatja be. Két rovarász és két gombász van, a gombász kezd, aki a **putFirstMushroom ShortLifeThread t1** paranccsal lehelyezi az első gombatestjét a t1 tektonra. Ezután a következő játékos jön, az fplayer2 aki a **putFirstMushroom LongLifeThread t4** paranccsal le szeretné helyezni az első gombatestjét a t4 tektonra, azonban ez AbsorbingTecton, így sikertelen a próbálkozása, ezután a **putFirstMushroom LongLifeThread t1** paranccsal a t1 tektonra szeretné lehelyezni gombatestjét, azonban itt már van egy gombatest, így újra sikertelen a lépés. Ezután a **putFirsMushroom LongLifeThread t2** paranccsal sikeresen lehelyezi az első gombatestjét. Ezután az iplayer1 nevű rovarász jön, ő a **putFirstInsect t3** paranccsal lehelyezi az első rovarát a t3 tektonra. Ezután a következő játékos jön, az iplayer2, aki kezdetben a t3 tektonra próbálja lehelyezni a rovarát a **putFirstInsect t3** paranccsal, azonban sikertelenül, ezután a **putFirstInsect t1** paranccsal sikeresen lehelyezi a rovarát a t1 tektonra. Mivel a körben minden játékos elvégezte a lépését, az első játékos kerül újra sorra, illetve új kör jön, így megtörténik a kör inicializálása az initRound() metódus segítségével.

* **Ellenőrzött funkcionalitás, várható hibahelyek**
* Gombász első gombatestjének lehelyezése
* első gombatest lehelyezése olyan tektonra, amelyen már van gombatest
* első gombatest lehelyezése absorbing tektonra
* Rovarász első gombatestjének lehelyezése
* első bogár lehelyezése olyan tektonra, ahol már van bogár
* A 0. kör utáni körinicializálás tesztelése
* **Bemenet**

**arrange**

**turnOffRandom**

**createTecton MultiThreadTecton**

**createTecton MultiThreadTecton**

**createTecton SingleThreadTecton**

**createTecton AbsorbingTecton**

**setNeighbors t1 t2 t3**

**setNeighbors t2 t4 t1**

**setNeighbors t3 t1 t4**

**setNeighbors t4 t3 t2**

**setFungusPlayerCount 2**

**setInsectPlayerCount 2**

**createFungusPlayers fplayer1 fplayer2**

**createInsectPlayers iplayer1 iplayer2**

**setMaxRound 5**

**end**

**act**

**putFirstMushroom ShortLifeThread t1**

**putFirstMushroom LongLifeThread t4**

**putFirstMushroom LongLifeThread t1**

**putFirstMushroom LongLifeThread t2**

**putFirstInsect t3**

**putFirstInsect t3**

**putFirstInsect t1**

**end**

**saveResult**

* **Elvárt kimenet**

-begin-

ShortLifeThread f1

LongLifeThread f2

FungusPlayer fplayer1

FungusPlayer fplayer2

Insect i1

Insect i2

InsectPlayer iplayer1

InsectPlayer iplayer2

Mushroom m1

Mushroom m2

MultiThreadTecton t1

MultiThreadTecton t2

SingleThreadTecton t3

AbsorbingTecton t4

—

f1

tectons t1

life -

f1

tectons t2

life -

fplayer1

points 1

branchThread false

fungalThread f1

mushrooms m1 1

fplayer2

points 1

branchThread false

fungalThread f2

mushrooms m2 1

i1

position t3

state NORMAL

i2

position t1

state NORMAL

iplayer1

points 0

insects i1 false false

iplayer1

points 0

insects i2 false false

m1

position t1

spores -

thread f1

state UNEVOLVED

shootedSporesCount 0

m2

position t2

spores -

thread f2

state UNEVOLVED

shootedSporesCount 0

t1

spores -

neighbors t2 t3

insect i2

mushroom m1

threads f1

t2

spores -

neighbors t1 t4

insect -

mushroom m2

threads f2

t3

spores -

neighbors t1 t4

insect i1

mushroom -

threads -

t4

spores -

neighbors t2 t3

insect -

mushroom -

threads -

GameState

round 1

maxRound 5

currentPlayer fplayer1

-end-

* **A konzol kimenete:**

t1 Tecton

t2 Tecton

t3 Tecton

t4 Tecton

f1 FungalThread

m1 Mushroom

Sikertelen

Sikertelen

f2 FungalThread

m2 Mushroom

i1 Insect

Sikertelen

i2 Insect

### Gombász és rovarász alap lépései

* **Leírás**

Ebben a tesztesetben egy gombász és egy rovarász van. A kört a rovarász kezdi, aki a gombafonalat a **branchThread f t2 paranccsal a** t2 tektonra ágaztatja el, ezután kilő egy spórát a **shootSpore m t3 paranccsal a** t3 tektonra és lezárja lépését. Ezután a rovarász következik, aki a **move i t1** paranccsal az i rovarával a t1 tektonra lép, ezután lezárja a lépését. Ezután a következő játékos jön, mivel a körben minden játékos elvégezte a lépését, az első játékos kerül újra sorra, illetve új kör jön, így megtörténik a kör inicializálása az initRound() metódus segítségével. Mivel a második kör jön minden gombatest kap egy új spórát, valamint egyes részeinek el kell pusztulnia a vágás szerint

* **Ellenőrzött funkcionalitás, várható hibahelyek**
* A gombafonál elágaztatása tektonra
* Gombatest spóra kilövése tektonra
* Bogár mozgása
* Bogár fonál vágása
* Az első kör utáni körinicializáló tesztelése, ekkor megnöveli a körök számát 1-el, mivel 2, azaz páros lesz ennek során a körök száma, így minden gombatestnek egy új spórája kell generálódjon
* Mivel egy ShortLifeThread-et vágott el a rovar, ezért ahogy elvágta, nem pusztul el a tektonról a fonál, de a következő körben az init lefutása után már el kell pusztulnia
* **Bemenet**

**arrange**

**turnOffRandom**

**createTecton MultiThreadTecton**

**createTecton MultiThreadTecton**

**createTecton SingleThreadTecton**

**createTecton SingleThreadTecton**

**setNeighbors t1 t2 t3**

**setNeighbors t2 t4 t1**

**setNeighbors t3 t1 t4**

**setNeighbors t4 t3 t2**

**setFungusPlayerCount 1**

**setInsectPlayerCount 1**

**createFungusPlayers fplayer**

**createInsectPlayers iplayer**

**createShortLifeThread t1 fplayer**

**createMushroom t1 f1 fplayer**

**createInsect t2 iplayer**

**generateSpore m1 SlowingSpore**

**stepGameRound 1**

**setMaxRound 5**

**end**

**act**

**branchThread f1 t2**

**shootSpore m1 t3**

**closestep**

**move i1 t1**

**cut i1 t2**

**closestep**

**end**

**saveResult**

* **Elvárt kimenet**

-begin-

ShortLifeThread f1

FungusPlayer fplayer

Insect i1

InsectPlayer iplayer

Mushroom m1

SlowingSpore s1

SlowingSpore s2

MultiThreadTecton t1

MultiThreadTecton t2

SingleThreadTecton t3

SingleThreadTecton t4

—

f1

tectons t1

life -

fplayer

branchThread false

fungalThread f1

mushrooms m1 1

i1

position t1

state NORMAL

iplayer

insects i false false

m1

position t1

spores s2

thread f1

state UNEVOLVED

shootedSporesCount 1

s1

thread f1

s2

thread f1

t1

spores -

neighbors t2 t3

insect i1

mushroom m1

threads f1

t2

spores -

neighbors t1 t4

insect -

mushroom -

threads -

t3

spores s1

neighbors t1 t4

insect -

mushroom -

threads -

t4

spores -

neighbors t2 t3

insect -

mushroom -

threads -

GameState

round 2

maxRound 5

currentPlayer fplayer

-end-

* **A konzol kimenete:**

t1 Tecton

t2 Tecton

t3 Tecton

t4 Tecton

f1 FungalThread

m1 Mushroom

i1 Insect

s1 Spore

s2 Spore

### Gombász és rovarász alap lépései olyan esetekben, mikor nem tudja végrehajtani

* **Leírás**

Arrange: Létrehozunk 9 tektont, ezeknek a program ad nevet t1, t2, … , t9. Ezután beállítjuk a tektonok szomszédsági listáját. Beállítjuk a gombászok számát 2-re, majd a rovarászok számát is 2-re, majd létrehozzuk a játékosokat. Létrehozunk egy gombatestet a t1 tektonra az fplayer1 gombász számára, majd létrehozunk a t5 tektonon egy gombatestet az fplayer2 számára, és gombafonalát elágaztatjuk a t4 és t8 tektonokon is. A t8 tektonon létrehozunk egy rovart az iplayer1 rovarásznak, a t3 tektonon pedig az iplayer2-nek. Az m1 gombatestnek generálunk egy spórát, ezután léptetjük a játék körét eggyel, és beállítjuk a maximális körök számát 5-re.

Act: Ebben a tesztesetben két gombász és két rovarász van. Az fplayer1 kezdi a kört, aki a gombafonalát a t3 tektonra próbálja ágaztatnia **branchThread f1 t3 paranccsal**, azonban sikertelenül, hasonlóan a t4-re is sikertelen az ágaztatás a **branchThread f1 t4**

paranccsal, mivel a SingleThreadTectonon van már gombafonál, ezután spórát próbál lőni a t3 tektonra a **hootSpore m1 t3**paranccsal sikertelenül, és az f1 gombafonál növeszteni akar gombatestet a t2 tektonra a **growMushroom f1 t2 paranccsal,** azonban ez is sikertelen lesz. Ezután a gombász lezárja lépését az fplayer 2 gombász jön, aki az fplayer1 fonaljával próbál lépni a **branchThread f1 t2**paranccsal, azonban a lépés nem lesz végrehajtva, ezután lezárja lépését. Az iplayer1 következik, aki rovarjával a t2 tektonra szeretne lépni a **move i1 t2** paranccsal, azonban sikertelen lesz, illetve vágni szeretne a **cut i1 t5** paranccsal, azonban ez is sikertelen, mivel gombatest van ezen a tektonon. Ezután lezárja a lépését, és az iplayer2 rovarász jön, aki az iplayer1 rovarával akar lépni a **move i1 t5** paranccsal, azonban a lépés sikertelen lesz.

* **Ellenőrzött funkcionalitás, várható hibahelyek**
* gombafonál növesztése tektonra, ahova nem tud növeszteni, mert a tekton messze van
* gombafonál növesztése olyan SingleThreadTecton-ra, amelyen már van gombafonál
* sikertelen fejletlen spóraszórás, mert nem szomszédos a céltekton
* gombatest növesztése olyan tektonra, amin nincs semmi
* játékos megpróbál egy másik játékos objektumával végrehajtani lépést
* rovar mozgatása olyan tektonra, amelyre ez nem lehetséges
* rovar fonál vágása olyan tekton esetében, amelyen gombatest van(ez nem lehetséges)
* a closestep hatására akkor is a következő játékos jön, ha a játékos még nem végzett el sikeres lépést
* ha a játékos lépései sikertelenek, a játék úgy veszi, hogy a játékos nem hajtott végre lépést
* a kontroller folyton észben tartja, hogy ki az a játékos, akinek jelenleg lépnie kell
* **Bemenet**

**arrange**

**turnOffRandom**

**createTecton MultiThreadTecton**

**createTecton MultiThreadTecton**

**createTecton SingleThreadTecton**

**createTecton SingleThreadTecton**

**createTecton MultiThreadTecton**

**createTecton MultiThreadTecton**

**createTecton AbsorbingTecton**

**createTecton KeepThreadTecton**

**createTecton MultiThreadTecton**

**setNeighbors t1 t2 t4**

**setNeighbors t2 t1 t3 t5**

**setNeighbors t3 t2 t6**

**setNeighbors t4 t1 t5**

**setNeighbors t5 t2 t6 t8 t4**

**setNeighbors t6 t5 t3 t9**

**setNeighbors t7 t4 t8**

**setNeighbors t8 t7 t5 t9**

**setNeighbors t9 t8 t6**

**setFungusPlayerCount 2**

**setInsectPlayerCount 2**

**createFungusPlayers fplayer1 fplayer2**

**createInsectPlayers iplayer1 iplayer2**

**createShortLifeThread t1 fplayer1**

**createMushroom t1 f1 fplayer1**

**createLongLifeThread t5 fplayer2**

**setTectons f2 t4 t8**

**createMushroom t5 f2 fplayer2**

**createInsect t8 iplayer1**

**createInsect t3 iplayer2**

**generateSpore m1 SlowingSpore**

**stepGameRound 1**

**setMaxRound 5**

**end**

**act**

**branchThread f1 t3**

**branchThread f1 t4**

**shootSpore m1 t3**

**growMushroom f1 t2**

**closestep**

**branchThread f1 t2**

**closestep**

**move i1 t2**

**cut i1 t5**

**closestep**

**move i1 t5**

**end**

**saveResult**

* **Elvárt kimenet**

-begin-

ShortLifeThread f1

LongLifeThread f2

FungusPlayer fplayer1

FungusPlayer fplayer2

Insect i1

Insect i2

InsectPlayer iplayer1

InsectPlayer iplayer2

Mushroom m1

Mushroom m2

SlowingSpore s1

MultiThreadTecton t1

MultiThreadTecton t2

SingleThreadTecton t3

SingleThreadTecton t4

MultiThreadTecton t5

MultiThreadTecton t6

AbsorbingTecton t7

KeepThreadTecton t8

MultiThreadTecton t9

—

f1

tectons t1

life -

f2

tectons t4 t5 t8

life -

fplayer1

points 1

branchThread false

fungalThread f1

mushrooms m1 0

fplayer2

points 1

branchThread false

fungalThread f2

mushrooms m2 0

i1

position t8

state NORMAL

i2

position t3

state NORMAL

iplayer1

points 0

insects i1 false false

iplayer2

points 0

insects i2 false false

m1

position t1

spores s1

thread f1

state UNEVOLVED

shootedSporesCount 0

m2

position t5

spores -

thread f2

state UNEVOLVED

shootedSporesCount 0

s1

thread f1

t1

spores -

neighbors t2 t4

insect -

mushroom m1

threads f1

t2

spores -

neighbors t1 t3 t5

insect -

mushroom -

threads -

t3

spores -

neighbors t2 t6

insect i2

mushroom -

threads -

t4

spores -

neighbors t1 t5 t7

insect -

mushroom -

threads f2

t5

spores -

neighbors t2 t4 t6 t8

insect -

mushroom m2

threads f2

t6

spores -

neighbors t3 t5 t9

insect -

mushroom -

threads -

t7

spores -

neighbors t4 t8

insect -

mushroom -

threads -

t8

spores -

neighbors t5 t7 t9

insect i1

mushroom -

threads f2

t9

spores -

neighbors t6 t8

insect -

mushroom -

threads -

GameState

round 1

maxRound 5

currentPlayer iplayer2

-end-

* **A konzol kimenete:**

t1 Tecton

t2 Tecton

t3 Tecton

t4 Tecton

t5 Tecton

t6 Tecton

t7 Tecton

t8 Tecton

t9 Tecton

f1 FungalThread

m1 Mushroom

f2 FungalThread

m2 Mushroom

i1 Insect

i2 Insect

s1 Spore

Sikertelen

Sikertelen

Sikertelen

Sikertelen

Sikertelen

Sikertelen

Sikertelen

Sikertelen

### Fonálgyorsítás, 10 spóra kilövés, bogárevés

* **Leírás**

Arrange: Létrehozunk 4 tektont, ezeknek a program ad nevet t1, t2, t3, t4. Ezután beállítjuk a tektonok szomszédsági listáját. Beállítjuk a gombászok számát 1-re, majd a rovarászok számát is 1-re, majd létrehozzuk a játékosokat. Létrehozunk egy fejlett gombatestet a t1 tektonra az fplayer gombász számára, a gombafonalát elágaztatjuk a t3 tektonra is. A gombatestnek beállítjuk, a shootedSporesCount attribútumát 9-re. Ezután létrehozunk egy gombatestet a t3 tektonon is. A t1 tektonon létrehozunk egy rovart az iplayer rovarásznak. Az m1 gombatestnek generálunk egy spórát. A t2 tektonra is lehelyezünk egy ParalysingSpore típusú spórát. Ezután léptetjük a játék körét eggyel, és beállítjuk a maximális körök számát 5-re.

Act: Ebben a tesztesetben egy gombász és egy rovarász van. A gombász kezd, aki a t2 tektonra ágaztatja a fonalát a **branchThread f1 t2** paranccsal, itt egy spóra van, így ez gyorsítja a fonál növekedését, így újra növeszthet fonalat, ezt meg is teszi a **branchThread f1 t4** paranccsal. Ezután kilő egy spórát a t4 tektonra a **shootSpore m1 t4** paranccsal, mivel a 10. spóráját lőtte ki, így ez után meghal, végül lezárja a lépését. Ezután a rovarász következik, aki rovarával a t2 tektonra lép a **move i1 t2** paranccsal, ahol megeszi a spórát, így PARALYSED állapotba kerül, ezután lezárja a lépését. Újra a gombász jön, illetve megtörténik a körinicializálás, mivel második kör, minden gombatest kap egy spórát, és a gombász fonalával megeszi a rovart az **eatInsect f1 i1** paranccsal, helyére egy új gombatest nő.

* **Ellenőrzött funkcionalitás, várható hibahelyek**
  + gombafonál növesztése tektonra, ahol spóra van, ekkor újból növeszteni tudjuk eggyel, egy olyan tektonra, amely szomszédos azzal a tektonnal, amelyen a spóra rajta van
  + Fejlett gombatest spóra szórása tektonra, a céltekton nem közvetlen szomszéd
  + gombatest 10. spórájának elszórása, itt a gombatest meghal(ehhez létre kell hozni majd egy olyan gombatestet, amely fejlett és már elszórt 9 spórát)
  + rovar mozgatása olyan tektonra, amelyen ParalysingSpore van(megváltozik az állapota)
  + gombafonál megeszi a bénult rovart(ha nem absorbing tekton, akkor új gombatest nő)
* **Bemenet**

**arrange**

**turnOffRandom**

**createTecton MultiThreadTecton**

**createTecton MultiThreadTecton**

**createTecton SingleThreadTecton**

**createTecton SingleThreadTecton**

**setNeighbors t1 t2 t3**

**setNeighbors t2 t4 t1**

**setNeighbors t3 t1 t4**

**setNeighbors t4 t3 t2**

**setFungusPlayerCount 1**

**setInsectPlayerCount 1**

**createFungusPlayers fplayer**

**createInsectPlayers iplayer**

**createShortLifeThread t1 fplayer**

**setTectons f1 t3**

**createEvolvedMushroom t1 f1 fplayer**

**setShotSpores m1 9**

**createMushroom t3 f1 fplayer**

**createInsect t1 iplayer**

**generateSpore m1 SlowingSpore**

**createSpore ParalysingSpore t2 f1**

**stepGameRound 1**

**setMaxRound 5**

**end**

**act**

**branchThread f1 t2**

**branchThread f1 t4**

**shootSpore m1 t4**

**closestep**

**move i1 t2**

**closestep**

**eatInsect f1 i1**

**end**

**saveResult**

* **Elvárt kimenet**

**valami ver**

-begin-

ShortLifeThread f1

FungusPlayer fplayer

InsectPlayer iplayer

Mushroom m2

Mushroom m3

SlowingSpore s1

SlowingSpore s3

MultiThreadTecton t1

MultiThreadTecton t2

SingleThreadTecton t3

SingleThreadTecton t4

—

f1

tectons t1 t2 t3 t4

life -

fplayer

points 3

branchThread false

fungalThread f1

mushrooms m2 1 m3 0

iplayer

points 1

insects -

m2

position t3

spores s3

thread f1

state UNEVOLVED

shootedSporesCount 0

m2

position t2

spores -

thread f1

state UNEVOLVED

shootedSporesCount 0

s1

thread f1

s3

thread f1

t1

spores -

neighbors t2 t3

insect -

mushroom -

threads f1

t2

spores -

neighbors t1 t4

insect -

mushroom m3

threads f1

t3

spores -

neighbors t1 t4

insect -

mushroom m2

threads f1

t4

spores s1

neighbors t1 t3

insect -

mushroom -

threads f1

GameState

round 2

maxRound 5

currentPlayer fplayer

-end-

* **A konzol kimenete:**

t1 Tecton

t2 Tecton

t3 Tecton

t4 Tecton

f1 FungalThread

m1 Mushroom

m2 Mushroom

i1 Insect

s1 Spore

s2 Spore

s3 Spore

m3 Mushroom

### Gombatest növesztése, bogár gyorsítása, LongLifeThread vágása

* **Leírás**

Ebben a tesztesetben egy gombász és egy rovarász van. A rovarász kezd, aki a t4 tektonra gombatestet növeszt a **growMushroom f1 t4** paranccsal, ezután lezárja lépését és a rovarász jön. A rovarász rovarával a t2 tektonra lép a **move i1 t2** paranccsal, ahol megesz egy gyorsítóspórát, így újra léphet, amit meg is tesz, rálép a t4 tektonra a **move i1 t4** paranccsal, ezután pedig elvágja a fonalat a t2 tektonról a **cut i1 t2** paranccsal, amely csak a következő kör utáni kör init részében fog elpusztulni.

* **Ellenőrzött funkcionalitás, várható hibahelyek**
  + Ellenőrzött funkcionalitás, várható hibahelyek
  + gombatest növesztése olyan tektonra, amelyen rajta van a gombafonala, meg 3 spórája
  + rovar mozgatása olyan tektonra, amelyen SpeedSpore van(ekkor még léphet egyet, ezután visszatér normal állapotba)
  + a játékos attribútumaiban ténylegesen eltároljuk, hogy a játékos egy olyan lépést hajtott végre, amit körönként csak egyszer tus
  + LongLifeThread esetén a rovar elvágja a fonalat
* **Bemenet**

**arrange**

**turnOffRandom**

**createTecton MultiThreadTecton**

**createTecton MultiThreadTecton**

**createTecton SingleThreadTecton**

**createTecton SingleThreadTecton**

**setNeighbors t1 t2 t3**

**setNeighbors t2 t4 t1**

**setNeighbors t3 t1 t4**

**setNeighbors t4 t3 t2**

**setFungusPlayerCount 1**

**setInsectPlayerCount 1**

**createFungusPlayers fplayer**

**createInsectPlayers iplayer**

**createLongLifeThread t3 fplayer**

**setTectons f1 t4 t2**

**createMushroom t3 f fplayer**

**createInsect t1 iplayer**

**createSpore SlowingSpore t4 f1**

**createSpore SlowingSpore t4 f1**

**createSpore SlowingSpore t4 f1**

**createSpore SpeedSpore t2 f1**

**stepGameRound 1**

**setMaxRound 5**

**end**

**act**

**growMushroom f1 t4**

**closestep**

**move i1 t2**

**move i1 t4**

**cut i1 t2**

**end**

**saveResult**

* **Elvárt kimenet**

valami ver

-begin-

LongLifeThread f1

FungusPlayer fplayer

Insect i1

InsectPlayer iplayer

Mushroom m1

Mushroom m2

MultiThreadTecton t1

MultiThreadTecton t2

SingleThreadTecton t3

SingleThreadTecton t4

—

f1

tectons t2 t3 t4

life t2 2

fplayer

points 2

branchThread false

fungalThread f1

mushrooms m1 0 m2 0

i1

position t4

state NORMAL

iplayer

points 1

insects i true true

m1

position t3

spores -

thread f1

state UNEVOLVED

shootedSporesCount 0

m2

position t4

spores -

thread f1

state UNEVOLVED

shootedSporesCount 0

t1

spores -

neighbors t2 t3

insect -

mushroom -

threads -

t2

spores -

neighbors t1 t4

insect -

mushroom -

threads f1

t3

spores -

neighbors t1 t4

insect -

mushroom m1

threads f1

t4

spores s1

neighbors t2 t3

insect i1

mushroom m2

threads f1

GameState

round 1

maxRound 5

currentPlayer iplayer

-end-

* **A konzol kimenete:**

t1 Tecton

t2 Tecton

t3 Tecton

t4 Tecton

f1 FungalThread

m1 Mushroom

i1 Insect

s1 Spore

s2 Spore

s3 Spore

s4 Spore

### Rovar SlowingSpore és NoCutSpore evése, és ennek hatása a következő körben

* **Leírás**

Ebben a tesztesetben egy gombász és két rovarász van. A gombász kezd, aki egyből lezárja a lépését anélkül, hogy valamit tett volna. Ezután az iplayer1 jön, aki a t1 tektonra lép a rovarjával a **move i1 t1 paranccsal**, és megeszi az ott található spórát. Ezután lezárja lépését. Az iplayer2 következik, aki a t2 tektonra lép a **move i2 t2 paranccsal**, ő is megeszi a tektonon található spórát, és lezárja a lépését. Ezután újra a gombász jön, mivel új körre váltott át a játék, így megtörténik a körinicializálás, a rovarászok rovarainak beállítódnak a cut és moved attribútumai a rovar állapota szerint, illetve a gombatesteknek egy új spórája keletkezik, mivel a második körbe váltottunk. A gombász ugyancsak lezárja a lépését nem csinálva semmit. Ezután az iplayer1 rovarász jön, aki lépni szeretne rovarával a **move i1 t2 paranccsal**, de a lépés sikertelen lesz.

* **Ellenőrzött funkcionalitás, várható hibahelyek**
  + rovar mozgatása olyan tektonra, amelyen SlowingSpore van(megváltozik az állapota)
  + rovar mozgatása olyan tektonra, amelyen NoCutSpore van(megváltozik az állapota)
  + a SlowingSpore és NoCutSpore hatása a rovarakra a következő körben
* **Bemenet**

**arrange**

**turnOffRandom**

**createTecton MultiThreadTecton**

**createTecton MultiThreadTecton**

**createTecton SingleThreadTecton**

**createTecton SingleThreadTecton**

**setNeighbors t1 t2 t3**

**setNeighbors t2 t4 t1**

**setNeighbors t3 t1 t4**

**setNeighbors t4 t3 t2**

**setFungusPlayerCount 1**

**setInsectPlayerCount 2**

**createFungusPlayers fplayer**

**createInsectPlayers iplayer1 iplayer2**

**createShortLifeThread t2 fplayer**

**setTectons f1 t1 t3 t4**

**createMushroom t2 f1 fplayer**

**createInsect t2 iplayer1**

**createInsect t4 iplayer2**

**createSpore SlowingSpore t1 f1**

**createSpore NoCutSpore t3 f1**

**stepGameRound 1**

**setMaxRound 5**

**end**

**act**

**closestep**

**move i1 t1**

**closestep**

**move i2 t2**

**closestep**

**closestep**

**move i1 t2**

**end**

**saveResult**

* **Elvárt kimenet**

**valami ver**

-begin-

ShortLifeThread f1

FungusPlayer fplayer

Insect i1

Insect i2

InsectPlayer iplayer1

InsectPlayer iplayer2

Mushroom m1

SlowingSpore s3

MultiThreadTecton t1

MultiThreadTecton t2

SingleThreadTecton t3

SingleThreadTecton t4

—

f1

tectons t1 t2 t3 t4

life -

fplayer

points 1

branchThread false

fungalThread f1

mushrooms m1 1

i1

position t1

state NORMAL

i2

position t3

state NORMAL

iplayer1

points 1

insects i1 true false

iplayer2

points 1

insects i2 false true

m1

position t2

spores s3

thread f1

state UNEVOLVED

shootedSporesCount 0

t1

spores -

neighbors t2 t3

insect i1

mushroom -

threads f1

t2

spores -

neighbors t1 t4

insect -

mushroom m1

threads f1

t3

spores -

neighbors t1 t4

insect i2

mushroom -

threads f1

t4

spores -

neighbors t2 t3

insect -

mushroom -

threads f1

GameState

round 2

maxRound 5

currentPlayer iplayer1

-end-

* **A konzol kimenete:**

t1 Tecton

t2 Tecton

t3 Tecton

t4 Tecton

f1 FungalThread

m1 Mushroom

i1 Insect

i2 Insect

s1 Spore

s2 Spore

s3 Spore

Sikertelen

### Fonalak felszívódása, tekton kettétörése, gombatest fejlődése

* **Leírás**

Ebben a tesztesetben egy gombász van. Már egyből a 3. körben kezdünk. A gombász elágaztatja gombafonalát a t2 tektonra a **branchThread f1 t2** paranccsal, ezután lezárja lépését. Mivel nincs több játékos, újra ő jön, illetve belépünk a következő körbe, ez a negyedik, így minden gombatest kap egy új spórát, az Absorbing tektonokon a fonalak felszívódnak, illetve mivel a random ki van kapcsolva, az első tekton törik ketté amit létrehoztunk, azaz a t1. Ezután a gombász lezárja lépését anélkül, hogy csinálna valamit, és újra ő jön, megtörténik a körinicializálás. Mivel a gombatest már 5 kör óta a játékban van EVOLVED állapotba kerül.

* **Ellenőrzött funkcionalitás, várható hibahelyek**
  + mikor lejár 4 kör tényleg eltörik egy tekton,mivel a random generálás ki van kapcsolva, emiatt az elsőnek létrehozott tekton törik ketté
  + minden 4. körben felszívódnak a gombafonalak az absorbingtektonról
  + ha a gombatest keletkezése óta eltelt öt kör, akkor ez fejletté vált-e
* **Bemenet**

**arrange**

**turnOffRandom**

**createTecton MultiThreadTecton**

**createTecton AbsorbingTecton**

**createTecton SingleThreadTecton**

**createTecton SingleThreadTecton**

**setNeighbors t1 t2**

**setNeighbors t2 t1 t3**

**setNeighbors t3 t2 t4**

**setNeighbors t4 t3**

**setFungusPlayerCount 1**

**createFungusPlayers fplayer**

**createShortLifeThread t3 fplayer**

**setTectons f1 t4**

**createMushroom t3 f1 fplayer**

**setMushroomAge m1 3**

**stepGameRound 3**

**setMaxRound 8**

**end**

**act**

**branchThread f1 t2**

**closestep**

**closestep**

**end**

**saveResult**

* **Elvárt kimenet**

**valami ver**

-begin-

ShortLifeThread f1

FungusPlayer fplayer

Mushroom m1

SlowingSpore s1

MultiThreadTecton t2

SingleThreadTecton t3

SingleThreadTecton t4

MultiThreadTecton t5

MultiThreadTecton t6

—

f1

tectons t3 t4

life -

fplayer

points 1

branchThread false

fungalThread f1

mushrooms m1 5

m1

position t3

spores s1

thread f1

state EVOLVED

shootedSporesCount 0

s1

thead f1

t2

spores -

neighbors t6 t3

insect -

mushroom -

threads -

t3

spores -

neighbors t2 t4

insect i2

mushroom m1

threads f1

t4

spores -

neighbors t3

insect -

mushroom -

threads f1

t5

spores -

neighbors t6

insect -

mushroom -

threads -

t6

spores -

neighbors t5 t2

insect -

mushroom -

threads -

GameState

round 5

maxRound 8

currentPlayer fplayer

-end-

* **konzol kimenete:**

t1 Tecton

t2 Tecton

t3 Tecton

t4 Tecton

f1 FungalThread

m1 Mushroom

s1 Spore

t5 Tecton

t6 Tecton

### Rovar DividingSpore evése

* **Leírás**

Ebben a tesztesetben egy gombász és egy rovarász van. A gombás lezárja a lépését, anélkül, hogy valamit tett volna, ezután a rovarász jön, aki a t1 tektonra lép a **move i1 t1** paranccsal, amelyen megeszik egy DividingSpore típusú spórát, és kettéválik.

* **Ellenőrzött funkcionalitás, várható hibahelyek**
  + rovar mozgatása olyan tektonra, amelyen Dividing spore van
* **Bemenet**

**arrange**

**turnOffRandom**

**createTecton MultiThreadTecton**

**createTecton MultiThreadTecton**

**createTecton SingleThreadTecton**

**createTecton SingleThreadTecton**

**setNeighbors t1 t2 t3**

**setNeighbors t2 t4 t1**

**setNeighbors t3 t1 t4**

**setNeighbors t4 t3 t2**

**setFungusPlayerCount 1**

**setInsectPlayerCount 1**

**createFungusPlayers fplayer**

**createInsectPlayers iplayer**

**createShortLifeThread t3 fplayer**

**setTectons f1 t1 t2 t4**

**createMushroom t3 f1 fplayer**

**createInsect t2 iplayer**

**createSpore DividingSpore t1 f1**

**stepGameRound 1**

**setMaxRound 5**

**end**

**act**

**closestep**

**move i1 t1**

**end**

**saveResult**

* **Elvárt kimenet**

-begin-

ShortLifeThread f1

FungusPlayer fplayer

Insect i1

Insect i2

InsectPlayer iplayer1

Mushroom m1

MultiThreadTecton t1

MultiThreadTecton t2

SingleThreadTecton t3

SingleThreadTecton t4

—

f1

tectons t1 t2 t3 t4

life -

fplayer

points 1

branchThread false

fungalThread f1

mushrooms m1 0

i1

position t1

state NORMAL

i2

position t2

state NORMAL

iplayer

points 1

insects i1 true false i2 true true

m1

position t3

spores -

thread f1

state UNEVOLVED

shootedSporesCount 0

t1

spores -

neighbors t2 t3

insect i1

mushroom -

threads f1

t2

spores -

neighbors t1 t4

insect i2

mushroom -

threads f1

t3

spores -

neighbors t1 t4

insect -

mushroom m1

threads f1

t4

spores -

neighbors t2 t3

insect -

mushroom -

threads f1

GameState

round 1

maxRound 5

currentPlayer iplayer

-end-

* **A konzol kimenete:**

t1 Tecton

t2 Tecton

t3 Tecton

t4 Tecton

f1 FungalThread

m1 Mushroom

i1 Insect

s1 Spore

i2 Insect

### LongLifeThread vágása, rovar megevése mikor AbsorbingTecton-on van

* **Leírás**

Ebben a tesztesetben egy gombász és egy rovarász játékos van. A gombász kezd, aki lezárja a lépését, ezután a rovarász elvágja a fonalat a t1 tektonról a **cut i1 t1 paranccsal**, és a t3 tektonra lép a **move i1 t3 paranccsal**, ahol megeszik egy ParalysingSpore típusí spórát. Ezután lezárja lépését, a gombász jön, és megtörténik a körinicializálás, mivel a második kör következik, minden gombatest kap egy új spórát. A gombász a gombafonállal megeszi az i1 rovart a **eatInsect f1 i1 paranccsal**, ezután lezárja a lépését. Mivel az iplayernek nem maradt több rovara már nem kell lezárja a lépést, egyből a következő játékos jön, vagyis a rovarász. Megtörténik a körinicializáció, most fog ténylegesen a fonál elpusztulni a t1 tektonról.

* **Ellenőrzött funkcionalitás, várható hibahelyek**
  + gombafonál megeszi a bénult rovarat, mivel AbsorbingTektonon volt a rovar, ezért nem nő új gombatest
  + fonál vágása esetén, LongLifeThread-ről van szó, akkor csak a következő kör utáni kör esetében történik meg a tényleges
* **Bemenet**

**arrange**

**turnOffRandom**

**createTecton MultiThreadTecton**

**createTecton MultiThreadTecton**

**createTecton AbsorbingThreadTecton**

**createTecton SingleThreadTecton**

**setNeighbors t1 t2**

**setNeighbors t2 t1 t3**

**setNeighbors t3 t2 t4**

**setNeighbors t4 t3**

**setFungusPlayerCount 1**

**setInsectPlayerCount 1**

**createFungusPlayers fplayer**

**createInsectPlayers iplayer**

**createShortLifeThread t4 fplayer**

**setTectons f1 t1 t2 t3**

**createMushroom t4 f1 fplayer**

**createInsect t2 iplayer**

**createSpore ParalyzingSpore t3 f1**

**stepGameRound 1**

**setMaxRound 5**

**end**

**act**

**closestep**

**cut i1 t1**

**move i1 t3**

**closestep**

**eatInsect f1 i1**

**closestep**

**end**

**saveResult**

* **Elvárt kimenet**

-begin-

ShortLifeThread f1

FungusPlayer fplayer

InsectPlayer iplayer

Mushroom m1

SlowingSpore s2

MultiThreadTecton t1

MultiThreadTecton t2

SingleThreadTecton t3

SingleThreadTecton t4

—

f1

tectons t2 t3 t4

life -

fplayer

points 1

branchThread false

fungalThread f1

mushrooms m1 2

iplayer

points 1

insects -

m1

position t4

spores s1

thread f1

state UNEVOLVED

shootedSporesCount 0

s2

thread f1

t1

spores -

neighbors t2

insect -

mushroom -

threads -

t2

spores -

neighbors t1 t3

insect -

mushroom -

threads f1

t3

spores -

neighbors t2 t4

insect -

mushroom -

threads f1

t4

spores -

neighbors t3

insect -

mushroom m1

threads f1

GameState

round 3

maxRound 5

currentPlayer fplayer

-end-

* **A konzol kimenete:**

t1 Tecton

t2 Tecton

t3 Tecton

t4 Tecton

f1 FungalThread

m1 Mushroom

i1 Insect

s1 Spore

s2 Spore

### Játék befejezésének leellenőrzése

* **Leírás**

Ebben a tesztesetben két gombász és két rovarász játszik. Először az fplayer1 gombász kezd, aki gombafonalát a t2 tektonra ágaztatja el a **branchThread f1 t2 paranccsal**, ezután befejezi a lépését. Ezután az fplayer2 jön, aki egy spórát lő ki a t8 tektonra a **shootSpore m2 t8 paranccsal** és lezárja a lépését. Ezután az iplayer1 rovarász jön, aki a t8 tektonra lép a **move i1 t8 paranccsal**, és megeszi a spórát a tektonon, ezután lezárja a lépését. Az iplayer2 jön, aki lezárja a lépését. Mivel a körben ő volt az utolsó játékos, új kör következik, mivel ennek a körnek a száma megegyezik a max körök számával, nem fut le az új kör inicializálása és kiíródik a konzolra ki nyert, és az összes játékos neve a pontszámaival együtt.

* **Ellenőrzött funkcionalitás, várható hibahelyek**
  + ha a játékosok lejátszák az előre meghatározott körök számát, akkor a játék befejeződik
* **Bemenet**

**arrange**

**turnOffRandom**

**createTecton MultiThreadTecton**

**createTecton MultiThreadTecton**

**createTecton SingleThreadTecton**

**createTecton SingleThreadTecton**

**createTecton MultiThreadTecton**

**createTecton MultiThreadTecton**

**createTecton AbsorbingTecton**

**createTecton KeepThreadTecton**

**createTecton MultiThreadTecton**

**setNeighbors t1 t2 t4**

**setNeighbors t2 t1 t3 t5**

**setNeighbors t3 t2 t6**

**setNeighbors t4 t1 t5**

**setNeighbors t5 t2 t6 t8 t4**

**setNeighbors t6 t5 t3 t9**

**setNeighbors t7 t4 t8**

**setNeighbors t8 t7 t5 t9**

**setNeighbors t9 t8 t6**

**setFungusPlayerCount 2**

**setInsectPlayerCount 2**

**createFungusPlayers fplayer1 fplayer2**

**createInsectPlayers iplayer1 iplayer2**

**createShortLifeThread t1 fplayer1**

**createMushroom t1 f1 fplayer1**

**createLongLifeThread t5 fplayer2**

**setTectons f2 t4 t8 t9**

**createMushroom t5 f2 fplayer2**

**createMushroom t9 f2 fplayer2**

**createInsect t5 iplayer1**

**createInsect t3 iplayer2**

**generateSpore m2 SlowingSpore**

**stepGameRound 1**

**setMaxRound 2**

**end**

**act**

**branchThread f1 t2**

**closestep**

**shootSpore m2 t8**

**closestep**

**move i1 t8**

**closestep**

**closestep**

**end**

**saveResult**

* **Elvárt kimenet**

-begin-

ShortLifeThread f1

LongLifeThread f2

FungusPlayer fplayer1

FungusPlayer fplayer2

Insect i1

Insect i2

InsectPlayer iplayer1

InsectPlayer iplayer2

Mushroom m1

Mushroom m2

Mushroom m3

MultiThreadTecton t1

MultiThreadTecton t2

SingleThreadTecton t3

SingleThreadTecton t4

MultiThreadTecton t5

MultiThreadTecton t6

AbsorbingTectont7

KeepThreadTecton t8

MultiThreadTecton t9

—

f1

tectons t1 t2

life -

f2

tectons t4 t5 t8 t9

life -

fplayer1

points 1

branchThread true

fungalThread f1

mushrooms m1 0

fplayer2

points 2

branchThread false

fungalThread f2

mushrooms m2 0

i1

position t8

state SLOWED

i2

position t3

state NORMAL

iplayer1

points 1

insects i1 true false

iplayer2

points 0

insects i2 false false

m1

position t1

spores -

thread f1

state UNEVOLVED

shootedSporesCount 0

m2

position t5

spores -

thread f2

state UNEVOLVED

shootedSporesCount 0

m3

position t9

spores -

thread f2

state UNEVOLVED

shootedSporesCount 0

t1

spores -

neighbors t2 t4

insect -

mushroom m1

threads f1

t2

spores -

neighbors t1 t3 t5

insect -

mushroom -

threads f1

t3

spores -

neighbors t2 t6

insect i2

mushroom -

threads -

t4

spores -

neighbors t1 t5 t7

insect -

mushroom -

threads f2

t5

spores -

neighbors t2 t4 t6 t8

insect -

mushroom m2

threads f2

t6

spores -

neighbors t3 t5 t9

insect -

mushroom -

threads -

t7

spores -

neighbors t4 t8

insect -

mushroom -

threads -

t8

spores -

neighbors t5 t7 t9

insect i1

mushroom -

threads f2

t9

spores -

neighbors t6 t8

insect -

mushroom m3

threads f2

GameState

round 1

maxRound 2

currentPlayer iplayer2

-end-

* **A konzol kimenete:**

t1 Tecton

t2 Tecton

t3 Tecton

t4 Tecton

t5 Tecton

t6 Tecton

t7 Tecton

t8 Tecton

t9 Tecton

f1 FungalThread

m1 Mushroom

f2 FungalThread

m2 Mushroom

m3 Mushroom

i1 Insect

i2 Insect

s1 Spore

fplayer1 - 1 pont

fplayer2 - 2 pont

iplayer1 - 1 pont

iplayer2 - 0 pont

Nyertesek: fplayer2 iplayer1

## *A tesztelést támogató programok tervei*

Miután a teszteset fájlában található parancsokat lefuttattuk, a program a result.txt fájlban előállítja a játék állapotát. Itt a megadott kimeneti nyelv szerint az objektumok abc sorrendben lesznek felsorolva, és ha egy objektum attribútumai között lista is szerepel, akkor a lista elemei is abc sorrendben lesznek kiírva az objektumok játékbeli neve alapján.

Ezt a fájlot az adott tesztesethez tartozó ellenőrző fájllal kell összehasonlítani.

Az összehasonlítás a következőképp történik:

Sorban vesszük a result.txt és az ellenőrző fájl egy-egy sorát, és összehasonlítjuk egymással. Mivel egyetlen objektumnak sincs lebegőpontos attribútuma, így nem történhet meg, hogy a sorok nem egyeznek meg, de a teszteset mégis helyes.

A teszteset eredménye akkor lesz helyes, ha mindkét fájl végére értünk, és minden sor megegyezett.

## *Napló*

| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| --- | --- | --- | --- |
| 2025. 04.09. 15:00 | 4 óra | Fejes | Controller osztály dokumentálásának folytatása. |
| 2025.04.06 19:00 | 2 óra | Zelch  Fejes  Szolár  Toronyi  Lekli | Megbeszélés, feladatok felosztása |
| 2025.04.07 14:00 | 6.5 óra | Zelch | Az előző dokumentumból a parancsok javítása, illetve a diagram javítása, valamint az Insect és FungalThread osztályok leírása |
| 2025.04.07 14:00 | 10 óra | Szolár | processCMD függvényben arrange parancsok feldolgozásának pszeudó kód leírása |
| 2025.04.08 10:00 | 7 óra | Zelch | Osztályok befejezése, tesztesetek leírása, kimeneti nyelv kibővítése, parancsok bővítése, kontroller diagram |
| 2025.04.08. 10:00 | 6 óra | Fejes | Modellbeli Spore osztály dokumentálása. Controller tervezése és dokumentálásának elkezdése. |
| 2025.04.08. 10:00 | 6 óra | Szolár | függvényben move, cut és egyéb parancsok feldolgozásának pszeudó kód leírása, tecton és absorbingtecton osztályok leírása, formázás. |
| 2025.04.09 14:00 | 6 óra | Zelch | Tesztesetek leírása |
| 2025.04.11 19:00 | 2 óra | Szolár  Fejes  Zelch  Toronyi  Lekli | Megbeszélés, eddigi munkák átnézése, további feladatok kiosztása |
| 2025.04.12 22:00 | 1 óra | Zelch | Saját osztályok véglegesítése, runTest parancs kidolgozása, tesztelést támogató programok tervei |
| 2025.04.12. 15:00 | 4,5 óra | Lekli | Saját osztályok megírása, kiosztott parancsok pszeudo-kódolása. |
| 2025.04.13. 09:00 | 5 óra | Toronyi | Kiosztott osztályok részletes leírása, parancsok pszeudokódjának megvalósítása |
| 2025.04.13. 10:00 | 2 óra | Fejes | Controller osztály dokumentálásának folytatása. |
| 2025.04.14. 10:30 | 1 óra | Fejes | Dokumentum formázása. |

**9. Prototípus beadása**

26 – five\_guys

Konzulens:

Huszerl Gábor

Csapattagok

| Lekli Zsolt | JENKHQ | zsoltlekli@gmail.com |
| --- | --- | --- |
| Fejes Ábel | AF1Y1C | fejesabel@gmail.com |
| Zelch Csaba | LK0617 | zch03po@gmail.com |
| Szolár Soma | E05SC6 | somaszolar1@gmail.com |
| Toronyi Zsombor | S8F7DV | toronyizsombi@gmail.com |

2025. 04. 28.

1. Prototípus beadása

# *Fordítási és futtatási útmutató*

## Fájllista

| **Fájl neve** | **Méret** | **Keletkezés ideje** | **Tartalom** |
| --- | --- | --- | --- |
| Controller.java | 55,244 bytes | ‎Tuesday, ‎April ‎22, ‎2025, ‏‎2:53:42 PM | kontroller osztály |
| FungusPlayer.java | 1,843 bytes | Tuesday, ‎April ‎22, ‎2025, ‏‎2:53:42 PM | Gombász játékos osztály |
| InsectAssociation.java | 695 bytes | ‎Friday, ‎April ‎25, ‎2025, ‏‎5:54:35 PM | Rovar és rovarász lépési amiket megtehet |
| InsectPlayer.java | 1,425 bytes | Tuesday, ‎April ‎22, ‎2025, ‏‎2:53:42 PM | Rovarász játékos osztály |
| MushroomAssociation.java | 530 bytes | Friday, ‎April ‎25, ‎2025, ‏‎5:54:35 PM | Gombatest és gombász lépései amiket megtehet |
| Player.java | 274 bytes | Tuesday, ‎April ‎22, ‎2025, ‏‎2:53:42 PM | Játékos osztály |
| AbsorbingTecton.java | 9,559 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | AbsorbingTecton osztály |
| DividingSpore.java | 753 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | DividingSpore osztály |
| FungalThread.java | 7,709 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Gombafonál osztály |
| IFungalThreadController.java | 412 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Gombafonál interfésze |
| IInsectController.java | 336 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Rovar interfésze |
| IMushroomController.java | 603 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Gombatest interfésze |
| Insect.java | 4,049 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Rovar osztály |
| InsectState.java | 674 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Rovar állapotait tartalmazó enum |
| ISporeController.java | 105 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Spóra interfész |
| ITectonController.java | 573 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Tecton interfész kontroller fele |
| ITectonView.java | 267 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Tecton interfész view fele |
| KeepThreadTecton.java | 7,807 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Fonalat megörző tekton osztály |
| LongLifeThread.java | 317 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Hosszan megörző fonál osztály |
| MultiThreadTecton.java | 8,328 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Tecton, amelyen több gombafonál elágaztatható |
| Mushroom.java | 5,732 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Gombatest osztály |
| MushroomState.java | 486 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Gombatest állapotait tároló enum |
| NoCutSpore.java | 966 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | NoCutSpore osztály |
| ParalysingSpore.java | 982 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | ParalysingSpore osztály |
| ShortLifeThread.java | 319 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Röviden megörző fonál osztály |
| SingleThreadTecton.java | 9,410 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | SingleThreadTecton osztály |
| SlowingSpore.java | 746 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | SlowingSpore osztály |
| SpeedSpore.java | 908 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | SpeedSpore osztály |
| Spore.java | 1,508 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Spóra osztály |
| Tecton.java | 9,752 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Tecton osztály |
| timeToDie.java | 369 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Segédosztály hogy a gombafonalak vágás után ne egybőlpusztuljanak el |
| Main.java | 3,253 bytes | ‎Friday, ‎April ‎25, ‎2025, ‏‎6:48:07 PM | A main tartalmazó osztály |
| IView.java | 75 bytes | Tuesday, ‎April ‎22, ‎2025, ‏‎2:53:42 PM | A View kontroller felőli interfésze |
| View.java | 194 bytes | Tuesday, ‎April ‎22, ‎2025, ‏‎2:53:42 PM | A View osztály |
| result.txt | 1 637 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | A lefuttatott teszteset eredményeként előálló állapot |
| test1.txt | 611 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | 0. kör tesztelése bemenet |
| test1assert.txt | 1 096 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | 0. kör tesztelése elvárt kimenet |
| test2.txt | 613 B | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Gombász és rovarász alap lépései bemenet |
| test2assert.txt | 822 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Gombász és rovarász alap lépései elvárt kimenet |
| test3.txt | 1 122 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Gombász és rovarász alap lépései olyan esetekben, mikor nem tudja végrehajtani bemenet |
| test3assert.txt | 1 568 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Gombász és rovarász alap lépései olyan esetekben, mikor nem tudja végrehajtani  elvárt kimenet |
| test4.txt | 749 báj | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Fonálgyorsítás, 10 spóra kilövés, bogárevés  bemenet |
| test4assert.txt | 874 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Fonálgyorsítás, 10 spóra kilövés, bogárevés  elvárt kimenet |
| test5.txt | 710 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Gombatest növesztése, bogár gyorsítása, LongLifeThread vágása bemenet |
| test5assert.txt | 860 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Gombatest növesztése, bogár gyorsítása, LongLifeThread vágása elvárt kimenet |
| test6.txt | 702 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Rovar SlowingSpore és NoCutSpore evése, és ennek hatása a következő körben bemenet |
| test6assert.txt | 914 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Rovar SlowingSpore és NoCutSpore evése, és ennek hatása a következő körben elvárt kimenet |
| test7.txt | 495 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Fonalak felszívódása, tekton kettétörése, gombatest fejlődése bemenet |
| test7assert.txt | 740 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Fonalak felszívódása, tekton kettétörése, gombatest fejlődése elvárt kimenet |
| test8.txt | 578 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Rovar DividingSpore evése bemenet |
| test8assert.txt | 826 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Rovar DividingSpore evése elvárt kimenet |
| test9.txt | 623 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | LongLifeThread vágása, rovar megevése mikor AbsorbingTecton-on van bemenet |
| test9assert.txt | 717 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | LongLifeThread vágása, rovar megevése mikor AbsorbingTecton-on van elvárt kimenet |
| test10.txt | 1080 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Játék befejezésének leellenőrzése bemenet |
| test10assert.txt | 1637 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Játék befejezésének leellenőrzése elvárt kimenet |
| init1.txt | 531 bájt | 2025. ‎április ‎28., ‎hétfő, ‏‎10:23:31 | Kezdő játékállapot |

## Fordítás

A programot a BME kari felhőjében biztosított, Windows 10 20H2 – JDK-Eclipse-WSU sablonnal ellátott virtuális gépen kell tudni fordítan. A játék fordítására kizárólag a JDK szükséges, amely biztosítja a java és javac parancsok elérhetőségét.

A következő parancs segítségével a **program** package-ben található állomány fordításra kerül (a package-t tartalmazó **src** mappából kiadva):

**javac -d . program/Main.java**

## Futtatás

A programot a BME kari felhőjében biztosított, Windows 10 20H2 – JDK-Eclipse-WSU sablonnal ellátott virtuális gépen kell tudni futtatni. A játék futtatásához kizárólag a JDK szükséges, amely biztosítja a java és javac parancsok elérhetőségét.

A fordítást követően az alábbi parancs segítségével a program futtatható:

**java program.Main**

# *Tesztek jegyzőkönyvei*

## Teszteset1

| **Tesztelő neve** | Fejes Ábel |
| --- | --- |
| **Teszt időpontja** | 2025.04.27. 13:00 |
| **Teszt eredménye** | A teszt sikertelen. |
| **Lehetséges hibaok** | Elírás a test1assert.txt-ben. Rossz thread van az elírás szerint az egyik tektonon. |
| **Változtatások** | test1assert.txt átírása megfelelően. |

| **Tesztelő neve** | Fejes Ábel |
| --- | --- |
| **Teszt időpontja** | 2025.04.27. 13:30 |
| **Teszt eredménye** | Sikeres |

## Teszteset2

| **Tesztelő neve** | Fejes Ábel |
| --- | --- |
| **Teszt időpontja** | 2025.04.27. 14:00 |
| **Teszt eredménye** | A teszt sikertelen. |
| **Lehetséges hibaok** | Elírás a test2assert.txt-ben. Tektonok szomszédossági listájánál elírás. |
| **Változtatások** | test2.assert.txt átírása megfelelően. |

| **Tesztelő neve** | Fejes Ábel |
| --- | --- |
| **Teszt időpontja** | 2025.04.27. 14:15 |
| **Teszt eredménye** | A teszt sikertelen. |
| **Lehetséges hibaok** | A modellbeli move függvény rossz, nem kerül át a rovar a megfelelő tektonra. |
| **Változtatások** | Move függvény javítása. |

| **Tesztelő neve** | Fejes Ábel |
| --- | --- |
| **Teszt időpontja** | 2025.04.27. 15:00 |
| **Teszt eredménye** | Sikeres |

## Teszteset3

| **Tesztelő neve** | Fejes Ábel |
| --- | --- |
| **Teszt időpontja** | 2025.04.28. 10:00 |
| **Teszt eredménye** | A teszt sikertelen. |
| **Lehetséges hibaok** | Elírás a test3assert.txt-ben. f2 fonál life attribútuma elírva. |
| **Változtatások** | test3.assert.txt átírása megfelelően. |

| **Tesztelő neve** | Fejes Ábel |
| --- | --- |
| **Teszt időpontja** | 2025.04.28. 10:15 |
| **Teszt eredménye** | Sikeres |

## Teszteset4

| **Tesztelő neve** | Fejes Ábel |
| --- | --- |
| **Teszt időpontja** | 2025.04.28. 10:30 |
| **Teszt eredménye** | A teszt sikertelen. |
| **Lehetséges hibaok** | A rovar megevése után nem törlődik a rovar a Controller objects Map-éből. |
| **Változtatások** | Processcmd függvény megfelelő case-ének átírása. |

| **Tesztelő neve** | Fejes Ábel |
| --- | --- |
| **Teszt időpontja** | 2025.04.28. 11:00 |
| **Teszt eredménye** | Sikeres |

## Teszteset5

| **Tesztelő neve** | Szolár Soma |
| --- | --- |
| **Teszt időpontja** | 2025. 04. 26. |
| **Teszt eredménye** | Sikertelen |
| **Lehetséges hibaok** | Elírás a test5assert.txt-ben. Szereplők felsorolásánál ABC sorrend nem teljesült. |
| **Változtatások** | test5assert.txt kijavítása |

| **Tesztelő neve** | Szolár Soma |
| --- | --- |
| **Teszt időpontja** | 2025. 04. 26. |
| **Teszt eredménye** | Sikertelen |
| **Lehetséges hibaok** | createFungusplayers-ben egy for ciklus rossz indextől indúlt, így rosszul jöttek létre a player-ek. |
| **Változtatások** | createFungusplayers, createInsectplayers (u.az a hiba) kijavítása. |

| **Tesztelő neve** | Szolár Soma |
| --- | --- |
| **Teszt időpontja** | 2025. 04. 27. |
| **Teszt eredménye** | Sikeres |

## Teszteset6

| **Tesztelő neve** | Szolár Soma |
| --- | --- |
| **Teszt időpontja** | 2025. 04. 26. |
| **Teszt eredménye** | Sikertelen |
| **Lehetséges hibaok** | Elírás a test6assert.txt-ben. Objektumok felsorolásánál ABC sorrenddel hiba. |
| **Változtatások** | test5assert.txt kijavítása |

| **Tesztelő neve** | Szolár Soma |
| --- | --- |
| **Teszt időpontja** | 2025. 04. 26. |
| **Teszt eredménye** | Sikertelen |
| **Lehetséges hibaok** | createInsect nem teszi bele a megfelelő konténerbe a rovart. |
| **Változtatások** | createInsect kijavítása |

| **Tesztelő neve** | Szolár Soma |
| --- | --- |
| **Teszt időpontja** | 2025. 04. 26. |
| **Teszt eredménye** | Sikeres |

## Teszteset7

| **Tesztelő neve** | Toronyi Zsombor |
| --- | --- |
| **Teszt időpontja** | 2025.04.27. |
| **Teszt eredménye** | Sikertelen |
| **Lehetséges hibaok** | Elírás a részletes tervek dokumentumban, a pszeudokódban. |
| **Változtatások** | Pszeudokód kijavítása, helyes implementáció |

| **Tesztelő neve** | Toronyi Zsombor |
| --- | --- |
| **Teszt időpontja** | 2025. 04. 27. |
| **Teszt eredménye** | Sikeres |

## Teszteset8

| **Tesztelő neve** | Toronyi Zsombor |
| --- | --- |
| **Teszt időpontja** | 2025.04.27. |
| **Teszt eredménye** | Sikertelen |
| **Lehetséges hibaok** | Az újonnan létrejött rovar példány nem került bele az objects tárolóba |
| **Változtatások** | Az új objektum felvétele az objectsbe. |

| **Tesztelő neve** | Toronyi Zsombor |
| --- | --- |
| **Teszt időpontja** | 2025. 04. 27. |
| **Teszt eredménye** | Sikeres |

## Teszteset9

| **Tesztelő neve** | Lekli Zsolt |
| --- | --- |
| **Teszt időpontja** | 2025. 04. 26. |
| **Teszt eredménye** | Sikertelen |
| **Lehetséges hibaok** | Elírás a test9.txt-ben. |
| **Változtatások** | test9.txt kijavítása |

| **Tesztelő neve** | Lekli Zsolt |
| --- | --- |
| **Teszt időpontja** | 2025. 04. 26. |
| **Teszt eredménye** | Sikeres |

## Teszteset10

| **Tesztelő neve** | Lekli Zsolt |
| --- | --- |
| **Teszt időpontja** | 2025. 04. 26. |
| **Teszt eredménye** | Sikertelen |
| **Lehetséges hibaok** | Helytelen objektum sorrend és kisebb elírás test10assert.txt-ben |
| **Változtatások** | test10assert.txt javítása |

| **Tesztelő neve** | Lekli Zsolt |
| --- | --- |
| **Teszt időpontja** | 2025. 04. 26. |
| **Teszt eredménye** | Sikeres |

# *Értékelés*

| **Tag neve** | **Tag neptun** | **Munka százalékban** |
| --- | --- | --- |
| Fejes Ábel | AF1Y1C | 20 |
| Lekli Zsolt | JENKHQ | 20 |
| Szolár Soma | E05SC6 | 20 |
| Toronyi Zsombor | S8F7DV | 20 |
| Zelch Csaba | LK0617 | 20 |

# *Napló*

| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| --- | --- | --- | --- |
| 2025.04.18 | 7 óra | Zelch | A model rész helyre pofozása, controller és az osztályok szkeletonjának létrehozása |
| 2025.04.21 | 4 óra | Zelch | Controller osztály és a Player osztályok funkcióinak elkezdése |
| 2025.04.22 | 3 óra | Toronyi  Lekli  Zelch  Szolár  Fejes | Meeting, feladatok kiosztása, részek logikájának ismertetése. |
| 2025.04.23 | 5 óra | Fejes | Parancsfeldolgozó kiadott case-einek megírása. Tesztelést támogató kiíró függvény implementálása. |
| 2025.04.24 | 3 óra | Lekli | Parancsfeldolgozó kiadott eseteinek megírása |
| 2025.04.25 | 2 óra | Fejes | Tesztelést támogató kiíró befejezése. |
| 2025.04.25 | 6 óra | Szolár | turnOnRandom  loadInit  assert  createTecton  createShortLifeThread  createLongLifeThread  createMushroom  createEvolvedMushroom  createInsect  createFungusPlayers  createInsectPlayers  createSpore  move  cut  parancsok implementálása a Controllerben. |
| 2025.04.25. | 4 óra | Toronyi | Kiosztott parancsok implementálása a pszeudokód alapján |
| 2025.04.26 | 2 óra | Toronyi  Lekli  Zelch  Szolár  Fejes | Rövid meeting, parancsfeldolgozóban talált hibák megtárgyalása. |
| 2025.04.26 | 1.5 óra | Lekli | Parancsfeldolgozó esetek hibáinak kijavítása saját részeken. |
| 2025.04.26 | 8 óra | Zelch | Controller parancsfeldolgozó funkcióinak javítása, illetve modellbeli javítások |
| 2025.04.26 | 4 óra | Szolár | 5-ös és 6-os tesztesetek futtatása, hibák kijavítása. |
| 2025.04.27 | 1 óra | Fejes | Kiosztott tesztesetek tesztelése. |
| 2025.04.27. | 4 óra | Toronyi | 7-es és 8-as tesztesetek kiértékelése, javítása. Javadoc dokumentáció elkészítése a codebase-hez, forrásfájlok formázása. |
| 2025.04.28 | 1 óra | Fejes | Tesztesetek dokumentálása. |

**11. Grafikus felület specifikációja**

26 – five\_guys

Konzulens:

Huszerl Gábor

Csapattagok

| Lekli Zsolt | JENKHQ | zsoltlekli@gmail.com |
| --- | --- | --- |
| Fejes Ábel | AF1Y1C | fejesabel@gmail.com |
| Zelch Csaba | LK0617 | zch03po@gmail.com |
| Szolár Soma | E05SC6 | somaszolar1@gmail.com |
| Toronyi Zsombor | S8F7DV | toronyizsombi@gmail.com |

2025. 05. 05.

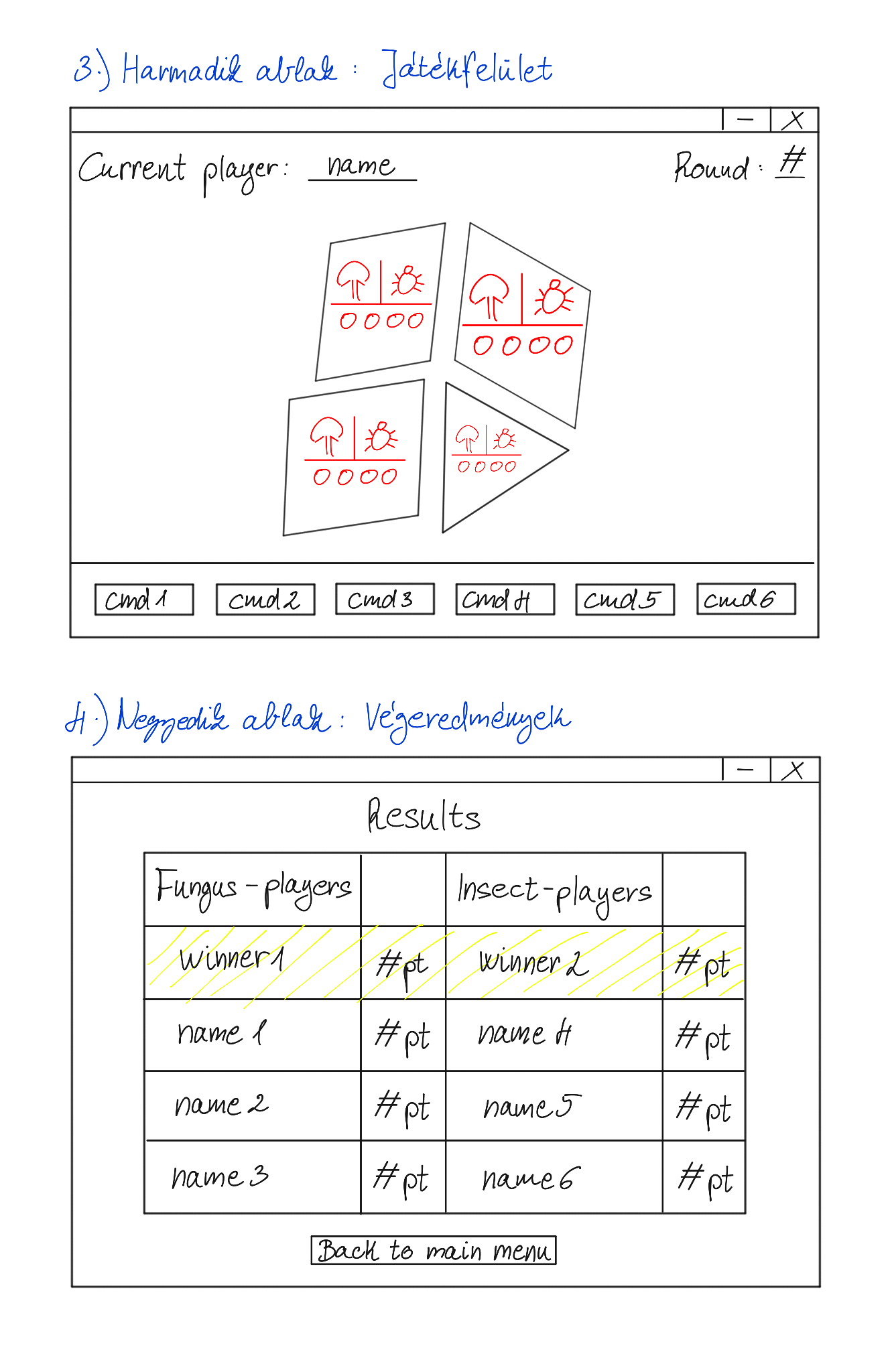
1. Grafikus felület specifikációja

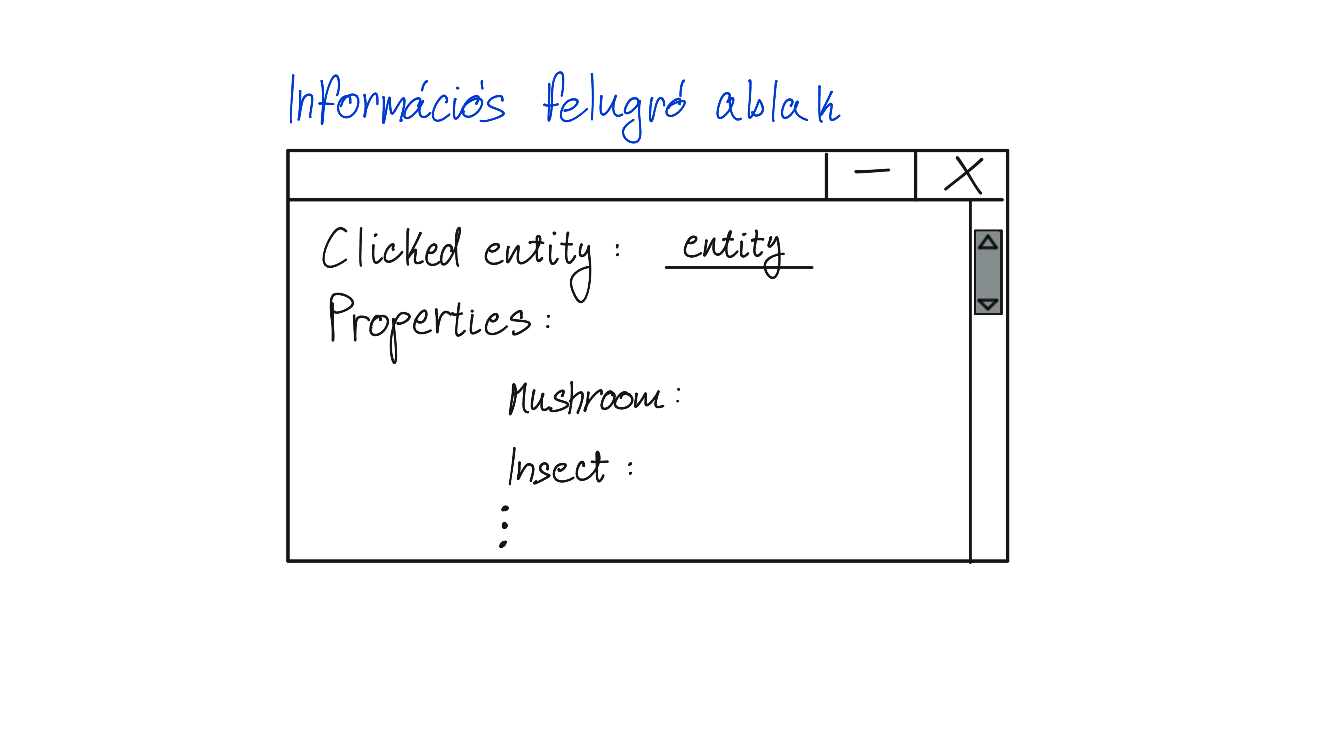
# *Változtatások*

* Mushroom-nak, Spore-nak, Insect-nek G-osztály, View Interface. A GTecton draw fg-e kirajzolja magát és meghívja a GMushroom, GSpore, GInsect draw fg-eit.
* GTecton-ban nem tároljuk el a szomszédos GTecton-okat.
* GamePanel tartalmaz egy DrawingPanel és ezen lesznek rajta a GTectonok, ő végzi a rajzolásukat

# *A grafikus interfész*

**

**

**

A grafikus felhasználói felület 5 fő különböző megjelenést támogat.

1. **A játékosok számának megadására szolgáló nézet**

Ez az első megjelenő ablak tartalma a program futtatását követően. Tartalmazza a játék nevét, mint címet. Ezen kívül címkéket, mellettük pedig beviteli mezőket, amelyek a megfelelő játékos típusok számának (maximum 4-4), illetve a játék körei számának felső határának megadására adnak lehetőséget. Bal alsó sarokban pedig található egy gomb, amivel tovább lehet lépni a következő nézetre.

1. **A játékosok neveinek megadására szolgáló nézet**

Ez a nézet az első nézet ‘Next’ gombját követően mutatkozik meg, ha a megadott paraméterek helyesek. Címként látható egy utasítás, ami elárulja, hogy itt a játékosok neveinek megadása fog következni. A cím alatt egy 2 oszlopos és legfeljebb 5 soros rács mintájában lesz látható a két játékos kategória (Fungus-players, Insect-players) és alattuk pedig az előző nézetben megadott számoknak megfelelő mennyiségű beviteli mező, ahova a játékosok neveit várja a program. Továbbá még található egy ‘Back’ és egy ‘Start’ gomb is az alsó sarkokban, amikkel az előző nézetre vissza lehet lépni, vagy pedig elindítani a játékot.

1. **A játékfelület**

Az előző nézet ‘Start’ gombjára kattintva kerül megjelenítésre maga a játékfelület, ha minden mező helyesen lett kitöltve. A játékfelület bal felső sarkában lesz látható az aktuális játékosnak a neve, tehát annak, aki éppen sorra került, jobb felül pedig az aktuális kör sorszáma. Középen lesz látható a térkép, ahol a játék zajlik. Itt lesznek a tektonok (sokszögek), rajta a gombák, rovarok, spórák (ikonok) táblázatos formában és köztük a gombafonalak. Legalul pedig egy sor gomb foglal helyet, melyeken az aktuális játékos típusának megfelelő parancsok szerepelnek majd.

1. **A végeredmény megtekintésére szolgáló nézet**

A játék befejeztével kerülünk erre a nézetre, ahol a cím is megmutatja, hogy a végeredményt fogjuk tudni megtekinteni. Egy 4 oszlopos táblázat fogad minket, az első sora megmutatja melyik oszlopban milyen típusú játékosok nevei és eredményei láthatók. Az első oszlop a Fungus-player játékosok neveit tartalmazza, a második azoknak az elért pontjait, a harmadik pedig az Insect-player játékosok neveit, a negyedik pedig szintén azok pontjait. A játékosok adatai pontszám szerint csökkenően jelennek meg, illetve az egyes kategóriában legtöbb pontszámot elért játékosok adatainak a háttere sárga színnel lesz kiemelve. Illetve a táblázat alatt található egy gomb a ‘Back to main menu’ felirattal, ami visszavisz minket az első nézetre, hogy esetleg egy újabb játéknak foghassunk neki.

1. **Egy információs felugró ablak**

Ez egy olyan felugró ablak, ami a ‘Show info’ gombra kattintva jelenik meg, ha azt megelőzve bal egérgomb lenyomásával kijelölésre került egy entitás (tekton, gomba, rovar, spóra, fonal). Az első sorában láthatjuk a kijelölt entitás típusát, alatta pedig annak jellemzőit. (például tekton esetében a rajta levő gombát, rovart, stb.)

# *A grafikus rendszer architektúrája*

## A felület működési elve

A felület működési elve kevert. A lépések után, ahol a felület változik, a kontroller értesíti a nézetet a változásról, ezáltal újrarajzolódik a felület. A nézet újrarajzolásakor a GTectonok a modellben levő tektonokról lekérdezik a rajtuk levő objektumokat, és ennek függvényében rajzolják ki a tektonokat.

## A felület osztály-struktúrája

# *A grafikus objektumok felsorolása*

## IController

* + - * **Felelősség**

A Controller által megvalósított és a View osztályai által használt interfész.

## IGamePanel

A GamePanel által megvalósított és a Controller által használt interfész.

## Controller (változtatások)

Megvalósítja az IController interfészt. Tartalmazza az IGamePanelt.

* + - * **Új interfészek**

IController

* + - * **Új attribútumok**
* **- IGamePanel gamePanel**
  + - * **Új megvalósított metódusok**
* **+ String getName(Object o):** Kikeresi a controller objects map-jéből a paraméterül kapott o-hoz a nevét és visszaadja azt.
* **+ Tecton createTecton(String tectonType):** Létrehozz egy tektont a paraméterül kapott tektontípus alapján és visszaadja azt.
* **+ String getCurrentPlayerName():** Visszaadja a currentPlayer nevét (kikeresi az objects mapből).
* **+ int getRoundNumber():** Visszaadja hogy hányadik körben jár a játék.
* **+ Player getMushroomPlayer(Mushroom: m)**: Visszaadja a paraméterül kapott gombához tartozó játékost.
* **+ Player getInsectPlayer(Insect i):** Visszaadja a paraméterül kapott rovarhoz tartozó játékost.
* **+ Player getThreadPlayer(FungalThread f):** Visszaadja a paraméterül kapott fonálhoz tartozó játékost.
* **+ Player getSporePlayer(Spore s):** Visszaadja a paraméterül kapott spórához tartozó játékost.
* **+ List<Player> getPlayers():** Visszaadja az összes játékost (gombászokat, rovarászokat).

## JFrame

Java Swing-beli osztály.

## JPanel

Java Swing-beli osztály.

## MainWindow

* + - * **Felelősség**

A MainWindow osztály felelős a játék megjelenítéséért, CardLayout-ot használ a játék további képernyői közötti váltásra. A JFrame osztályból származik le. Ezzel a megoldással csak egyetlen JFrame kell, nem kell minden képernyőváltáskor újat létrehozni.

* + - * **Ősosztályok**

JFrame -> MainWindow

* + - * **Interfészek**

-

* + - * **Attribútumok**
* **- CardLayout cardLayout**: Java layout típus, segítségével könnyen váltogathatunk a képernyők között.
* **- JPanel cardPanel:** A megfelelő játék képernyők közötti váltogatásra szolgál.
* - **StartScreenPanel startScreenPanel:** A kezdő képernyő megjelenítéséért.
* **- NameEntryPanel nameEntryPanel:** A nevek megadásáért.
* **- WinPanel winPanel:** A győztesek megjelenítéséért.
* **- Controller controller:** A Controllerrel való kommunikálásért.
  + - * **Metódusok**
* **+ void showStartScreen()**: A kezdőképernyő megjelenítéséért felelős metódus
* **+ void showNameEntry( int FPlayerCount, int IPlayerCount ):** Annak a képernyőnek megjelenítéséért felelős, amelyiknél meg kell adni a játékosok nevét, annyi írható szövegdoboz fog megjelenni, amennyit paraméterként bekap rovarászra illetve gombászra.
* **+ void showGame():** A játékképernyő megjelenítése, itt fog történni a játél érdemi része, itt láthatóak majd a tektonok, stb.
* **+ void showWin()**: A játék végén megjelenő képernyő megjelenítése, ranglista látható rajta a játékosokról.

## StartScreenPanel

* + - * **Felelősség**

Ez az osztály felelős a kezdőképernyő megjelenítéséért, a rajta található 3 írható szövegdobozban megadott paraméterek alapján beállítódnak a játék megfelelő paraméterei: a gombászok száma, a rovarászok száma és a maximális körök száma.

* + - * **Ősosztályok**

JPanel -> StartScreenPanel

* + - * **Interfészek**

-

* + - * **Attribútumok**
* **- JTextField FPlayerCount**: írható szövegdoboz, itt kell megadni hogy mennyi gombász lesz a játékban.
* **- JTextField IPlayerCount**: írható szövegdoboz, itt kell megadni, hogy mennyi rovarász lesz a játékban.
* **- JTextField MaxRounds:** írható szövegdoboz, itt kell megadni, hogy maximum mennyi kör legyen a játékban.
* **- JButton: nextWindow:** Gomb, a következő ablakra váltáshoz.
* **- MainWindow: mainWindow:** Lehessen továbblépni a NameEntryPanel-re.
  + - * **Metódusok**
* **+ void actionPerformed(ActionEvent e)**: A panel eseménykezelője.

## NameEntryPanel

* + - * **Felelősség**

Ez az osztály felelős a játékos nevek megadásához szükséges ablakért, rajta található annyi írható szövegdoboz, amennyit a StartScreenPanel-en megadtunk. Az itt megadott nevek lesznek a játékosok nevei a játék során.

* + - * **Ősosztályok**

JPanel -> NameEntryPanel

* + - * **Interfészek**

-

* + - * **Attribútumok**
* **- List<JTextField> FPlayerName:** A Gombászok neveinek megadásához szükséges írható szövegdoboz lista. Annyi szövegdoboz található benne, ahány gombász van.
* **- List<JTextField> FPlayerName:** A Rovarászok neveinek megadásához szükséges írható szövegdoboz lista. Annyi szövegdoboz található benne, ahány rovarász van.
* **- JButton nextWindow:** A következő ablakra váltás gombja.
* **- MainWindow: mainWindow:** Lehessen továbblépni a GamePanel-re.
  + - * **Metódusok**
* **+ void actionPerformed(ActionEvent e):** A panel eseménykezelője.

## WinPanel

* + - * **Felelősség**

Ez az osztály felelős a játék végén kiírandó eredmény tábla grafikus megjelenítéséért. A játékosok nevét és a hozzájuk tartozó pontot írja ki.

* + - * **Ősosztályok**

JPanel -> WinPanel

* + - * **Interfészek**

-

* + - * **Attribútumok**
* **- MainWindow: mainWindow:**  Hogy át lehessen menni a StartScreenPanel-re, kezdődik előlröl a folyamat.
  + - * **Metódusok**
* **+ void actionPerformed(ActionEvent e):** A panel eseménykezelője.

## GamePanel

* + - * **Felelősség**

Ez az osztály felelős a játékablakért, ahol a játékpálya és a játékos lépésfajtájának kiválasztásához tartozó gombok találhatóak. A játékpályán helyezkednek el a tektonok és a rajtuk lévő modellbeli objektumok. A játékosokhoz tartozó lépésfajták (amikhez a gombok tartoznak) a következők:

* Első kör esetén:
  + Gombász: Első gombatest lehelyezése
  + Rovarász: Első rovar lehelyezése
* Egyébként:
  + Gombász: Spóraszórás, fonál elágaztatás
  + Rovarász: Rovar léptetése, fonál vágása

Mindegyik játékosnak van továbbá egy olyan gombja, amellyel a körét zárja és ezzel jelzi, hogy jöhet a következő játékos. Ezek a gombok csak akkor jelennek meg ha a játékos a szabályok szerint azt a lépésfajtát a körében elvégezheti. Miután kiválasztja a játékos a lépésfajtáját, azután a lépés során kiválasztandó objektumot a megfelelő tektonra kattintva a játékpályán választhatja ki.

* + - * **Ősosztályok**

Jpanel -> GamePanel

* + - * **Interfészek**

-

* + - * **Attribútumok**
* **- GameState state:** A játék állapotát tárolja: milyen lépésfajta van kiválasztva az adott körben az adott játékos által.
* - **GTecton selectedSource**: Ha olyan lépésfajta van érvényben, ahol ki kell választani egy forrás és egy cél tektont, abban az esetben ez a forrás tekton. (Pl.: Ha az adott gombász játékos kiválasztott lépésfajtája spóraszórás, akkor ez lesz a forrás tekton, amin a kilövést végrehajtó gombatest rajta van.) Egyébként null.
* - **GTecton selectedTarget**: Ha olyan lépésfajta van érvényben, ahol ki kell választani egy forrás és egy cél tektont, abban az esetben ez a cél tekton. (Pl.: Ha az adott gombász játékos kiválasztott lépésfajtája spóraszórás, akkor ez lesz a cél tekton, amire a kilövést végrehajtó gombatest a spórát lövi.) Egyébként null.ű
* **- Map<String, GTecton> mtecton:** Olyan map, ami minden GTecton objektumhoz tárolja a hozzá tartozó egyedi nevet.
* **- List<GTecton>: tlist:** A játékpályát felépítő GTecton objektumokat tárolja.
* **- Map<InterfacePlayer, color> PColors:** A játékban résztvevő játékosokhoz tárolja a színűket. A játékosnak a tektonon lévő objektumai ezzel a színnel fognak megjelenni.
  + - * **Metódusok**
* **+ boolean createGameMap():** Létrehozza a modellben a játékpályát felépítő tektonokat és a hozzájuk tartozó View-beli GTectonokat és ezeket eltárolja a tlist-ben és az mtecton-ban. Az mtecton-ban a kulcs értékeket a controllertől kéri el. Tehát a controller-beli tektonnév lesz az mtectonban a grafikus GTectonhoz hozzárendelve. Visszatérési értéke igaz ha sikerült létrehoznia mindent megfelelően, egyébként hamis.
* + **void endGame():** Befejezi a játékot és hatására megjelenik az az ablak, ami kiírja a végeredményt (játékosok és pontjaik).
* **+ void repaint():** Újra rajzolja a játékpályát a tlist-ben tárolt GTecton-ok alapján.
* **- ActionListener tectonListener(int i):** A tektonra való kattintás esetén hívódik meg és a state attribútumnak megfelelően végrehajtódik a kívánt esemény a state-nek megfelelően.
* - **ActionListener buttonListener(int i):** A lépésfajt ákat ábrázoló gomb kattintása esetén hívódik meg, beállítja a gombnak megfelelően a state attribútumot.
* **+ void setPlayerColors():** Beállítja a játékosok színét és eltárolja a PColors map-be. Két játékosnak nem lehet ugyanaz a színe.
* + **boolean breakTecton(Tecton: Source, Tecton: newTecon1, Tecton: newTecton2):** Miután a modellben megvalósult egy tekton ketté törése azután hívódik meg ez a függvény. Paraméterül a ketté tört tektont (Source) és a két újonnan keletkezett tektont (newTecton1, newTecton2) kapja. A törést a Viewban is elvégzi. Létrehoz a két új tektonnak egy-egy GTectont, aminek a paramétereit a Source tekton GTectonja utáni törésnek megfelelően állítja be (points, sideNum, neighbors, tecton). Működését az *Inicializálás\_lefutása\_4\_el\_osztható\_kör\_esetén\_ekkor\_egy\_tekton\_kettétörik* szekvencia jól bemutatja.
* **+ boolean canBreak(Tecton t):** Megmondja a paraméterül kapott tektonról, hogy a viewbeli GTecton-ja törhető-e, tehát több mint 3 oldala van e (sideNumb > 3).

## GTecton

* + - * **Felelősség**

A GTecton a játék mezőjeinek grafikus megjelenítéséért és kezeléséért felelős. Tárolja a mező poligon‐alakját (oldalak száma és csúcspontok), a mezőhöz tartozó nézet‐interfészt, valamint a szomszédos mezők referenciáit. Feladata továbbá, hogy a játékpanel (GamePanel) hívására a saját alakzatát és a rajta levő objektumokat kirajzolja.

* + - * **Ősosztályok**

Nincs őse.

* + - * **Interfészek**

Nem valósít meg interfészt.

* + - * **Attribútumok**
* **- sideNumb : int -** hány oldala van**.**
* **- points : List<Vector<float>> -** tárolja a pontjainak x, y koordinátáit.
* **- tecton : IViewTecton** - tárolja, hogy a Controllerben milyen tecton felel meg neki, ennek a tectonnak meg kell valósítania a IViewTecton interfészt.
* **- neighbors : List<GTecton>** - tárolja a szomszédait.
  + - * **Metódusok**
* **- draw() : void** - felelős a mező és azon lévő entitások (gombatest, rovar, spórák, fonál) grafikus megjelenítéséért a játéktérben.

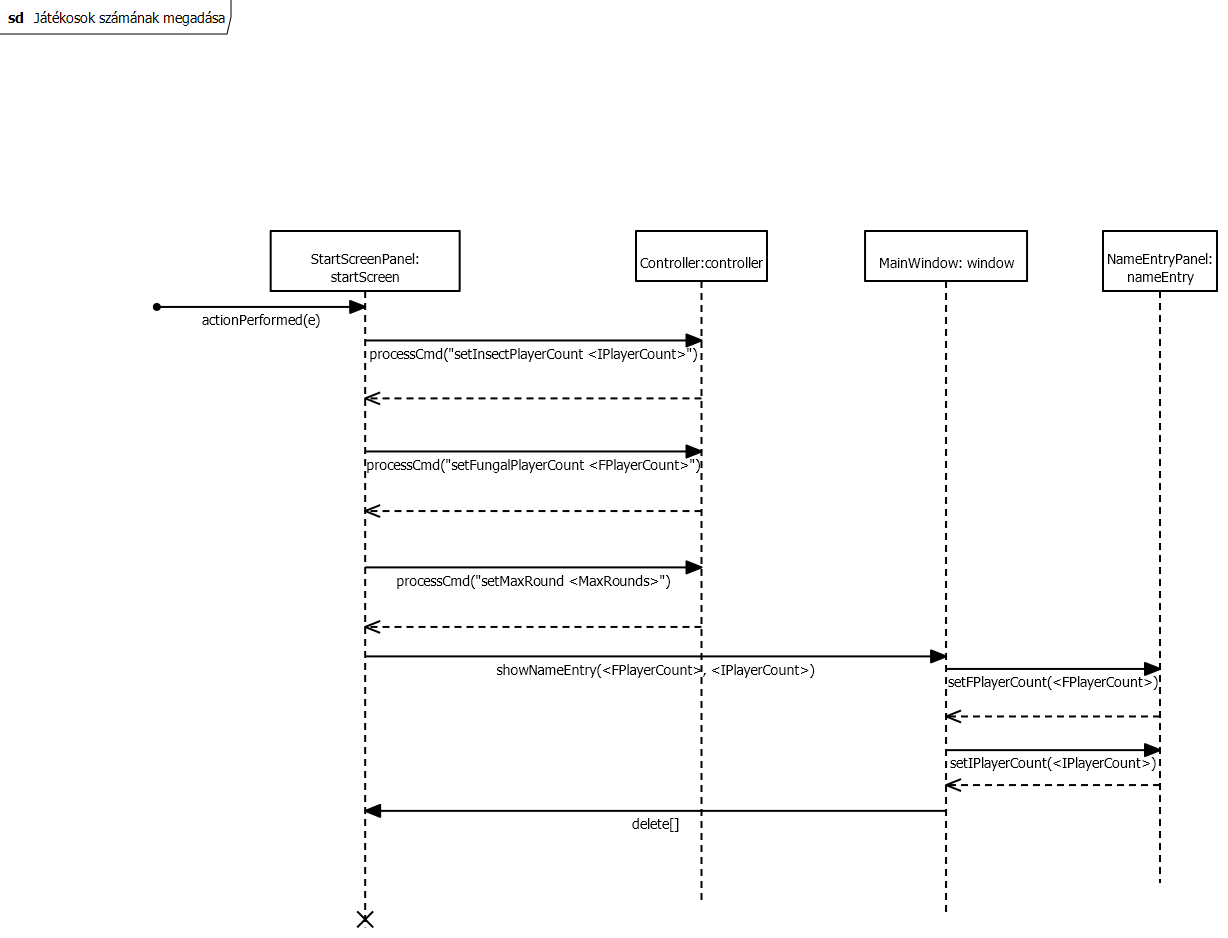
Részletes feladatok sorrendben

1. Poligon megrajzolása
   * A GTecton saját koordinátáiból felépíti a poligon alakú mezőt.
   * Kitöltéssel és körvonallal megjeleníti a mezőt.
2. Gombatest kezelése
   * Lekérdezi a mezőhöz tartozó Mushroom objektumot (tecton.getMushroom()).
   * Ha létezik gombatest:
     1. Lekéri a gombatest aktuális állapotát a modellből.
     2. A vezérlőtől meghívja a mezőhöz rendelt játékost (controller.getMushroomPlayer(m)), hogy megkapja a kirajzoláshoz szükséges színt.
     3. A gombatestet a tecton közepén levő táblázat bal felső sarkába kirajzolja egy olyan ikonként, ami függ a játékos színétől, és a lekért típústól.
3. Rovar kezelése
   * Lekérdezi, hogy van-e az adott mezőn Insect.
   * Ha igen, ugyanúgy lekéri a hozzá tartozó játékost (controller.getInsectPlayer(i)) és egy kis „rovar-ikon” kirajzolásával jeleníti meg. (függ a játékos színétól) (jobb felső sarokba kerül)
4. Spórák kezelése
   * A mező getSpores() metódusával megszerzi a spórákat, ezeken végigmegy.
   * Minden spóra esetén lekéri a spóra játékosát (controller.getSporePlayer(s)) és kis körökként vagy ikonként jeleníti meg az alsó sorban. Ha egy játékosnak van több spórája a tectonon, az akkor is csak egy körként jelenik meg.
5. Gombafonalak (Threads) kirajzolása
   * Lekérdezi a mezőn lévő gombafonálokat (getThreads()).
   * Minden fonál esetén lekéri a megfelelő játékost (controller.getThreadPlayer(t)).
   * Megvizsgálja az összes, e mezőhöz szomszédos GTecton-t (neighbors): ha van közös, ugyanolyan gombafajhoz tartozó fonál, akkor a két mező közé, a játékos színével, egy vékony vonallal ábrázolja az elágazást.

* **- removeNeighbor(g : GTecton) : void**
* **- addNeighbor(g : GTecton) : void**

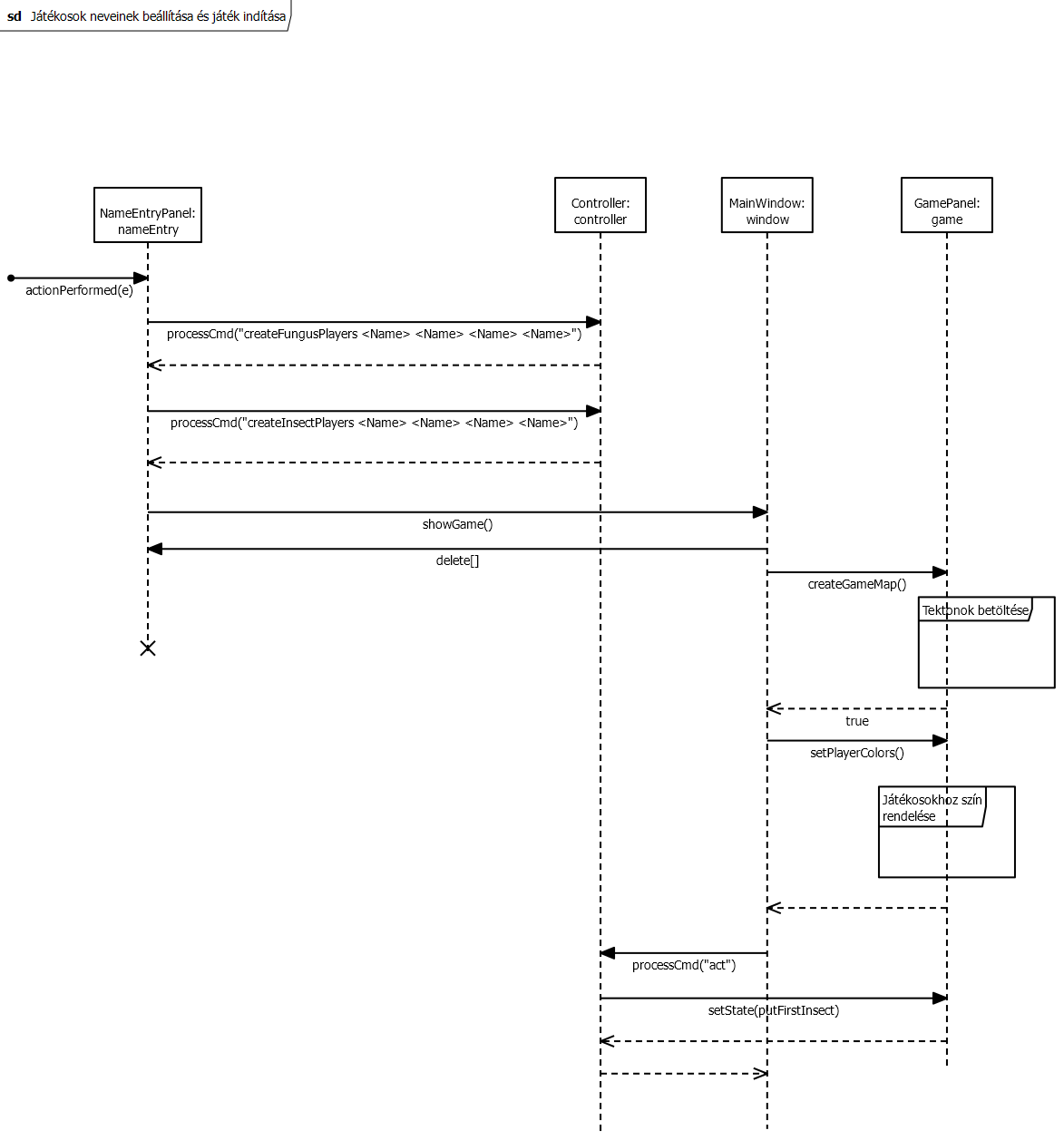
# *Kapcsolat az alkalmazói rendszerrel*

## Játékosok\_számának\_megadása

Hívás: rákattintunk a képernyőn levő nextWindow gombra

A szekvencia diagram azt ábrázolja, mikor az a képernyő van megjelenítve, amellyel beállíthatjuk a gombászok és a rovarászok számát, valamint a maximális körök számát. Miután a JTextFieldekbe beírtuk az adatokat rákattinthatunk a next gombra, ennek hatására meghívódik az actionPerformed függvény, ami beállítja a kontrollerben a gombászok és rovarászok számát, valamint a maximális körszámot, ezután meghívja a window showNameEntry metódusát, egy NameEntryPanel típusú JPanelt fog megjeleníteni az ablakban.

## Játékosok\_neveinek\_beállítása\_és\_játék\_indítása

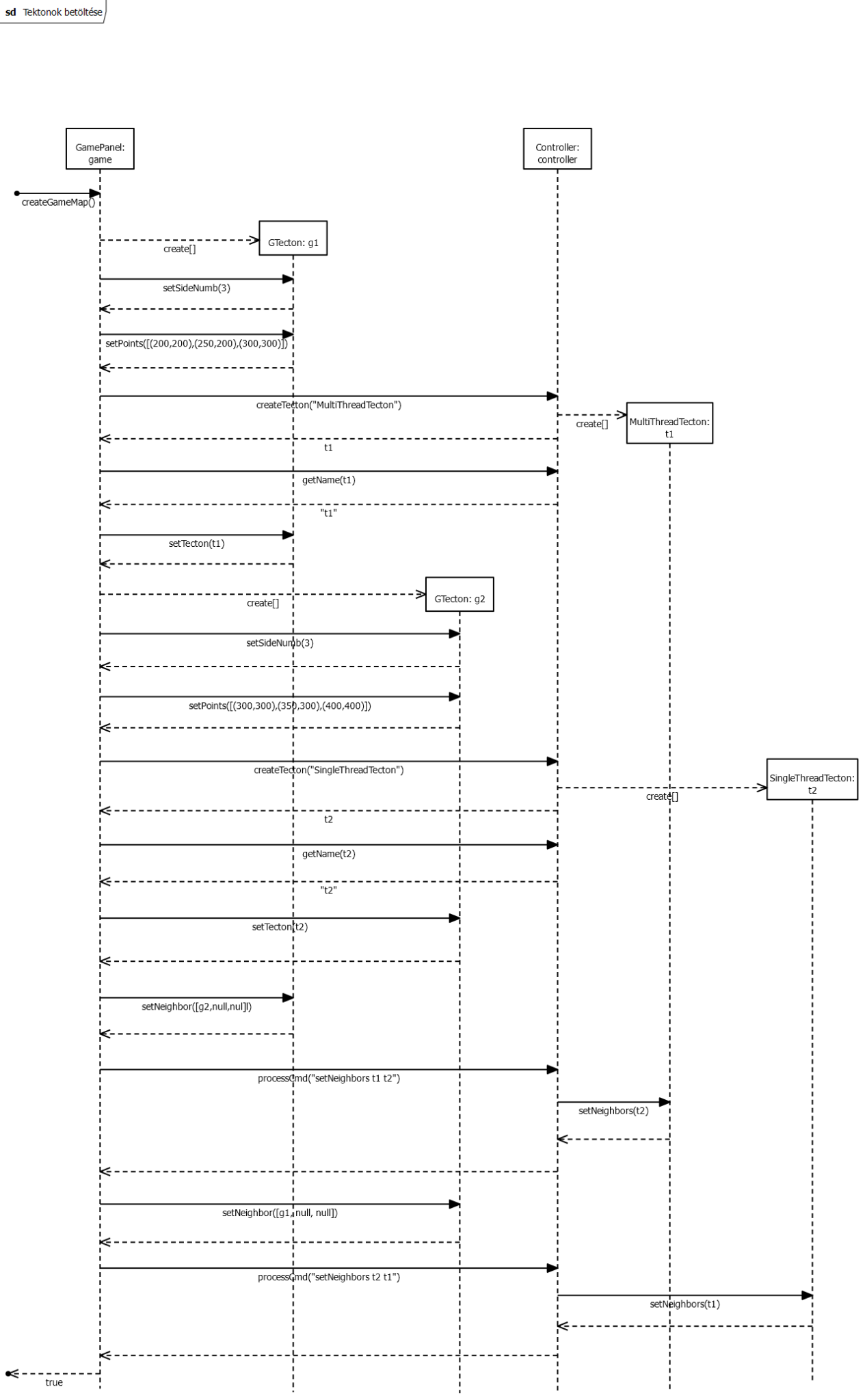
**

Hívás: rákattintunk a képernyőn levő nextWindow gombra

A szekvencia diagram azt ábrázolja, mikor a játékosok neveit adja meg. Ekkor a JTextFieldekbe beírhatjuk a neveket, miután ezt megtettük, rákattinthatunk a start gombra, ennek hatására meghívódik az actionPerformed metódus, ami beállítja a játékosok neveit a kontrollerben, és meghívja a MainWindow showGame metódusát, ezáltal betöltődnek a txt fájlból a tektonok, beállítódnak a játékosok színei, és elkezdődik a játék.

A Tektonokat a txt fájlokban a következőképp tároljuk: eltároljuk, hogy hány oldaluk van, eltároljuk minden pontjának a koordinátáját, majd miután az összes tektont felsoroltuk, eltároljuk a szomszédságokat a kontrollernél megismert módszer segítségével, azzal a különbséggel, hogy minden oldalra eltároljuk a szomszédot, és amely oldal nem szomszédos egyetlen tektonnal sem, ott a szomszéd értéke null.

## Tektonok\_betöltése

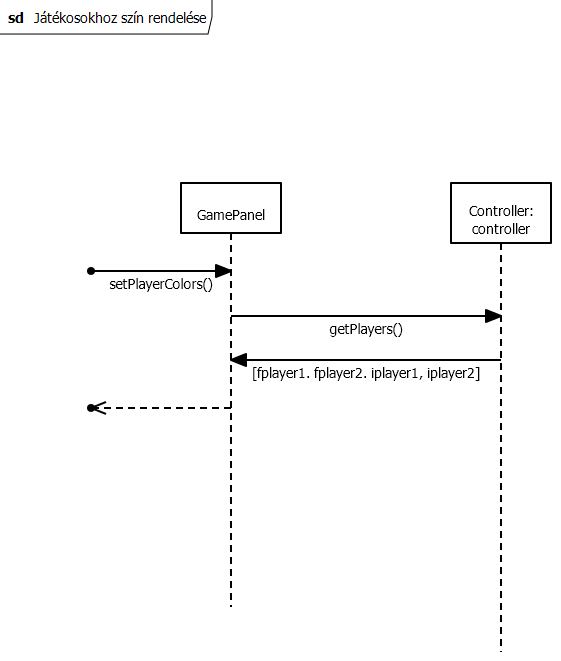


A MainWindow meghívja a GamePanel createGameMap() metódusát.

Ennek során sorra olvassuk be a tektonokat a txt fájlból, és a beolvasott adatoknak megfelelően létrehozunk GTectonokat, melyeknek a paramétereit a beolvasott adatoknak megfelelően beállítjuk. Ezután a kontroller createTecton függvéynét meghívva, létrehozunk a grafikus tektonnak megfelelő tektont, amit visszaadunk a GamePanelnek, ezáltal a GTektonnak beállíthatjuk az attribútumát. Lekérjük az adott tekton nevét is string-ként, és eltároljuk egy Map-ben a név GTekton összerendelést, erre majd szükségünk lesz a szomszédságok beállításánál.

Miután minden tektont beolvastunk, a szomszédságokat is be kell állítanunk, a text fájlban a kontrollernél megísmert módon vannak a szomszédságok eltárolva, azzal a különbséggel, hogy minden oldalra eltároljuk a szomszédot, és amely oldal nem szomszédos egyetlen tektonnal sem, ott a szomszéd értéke null. A Map összerendelés segítségével beállítjuk a GTektonoknál a szomszédságokat, majd a megfelelő parancsokat kiadva a modellben is beállítjuk a szomszédságokat.

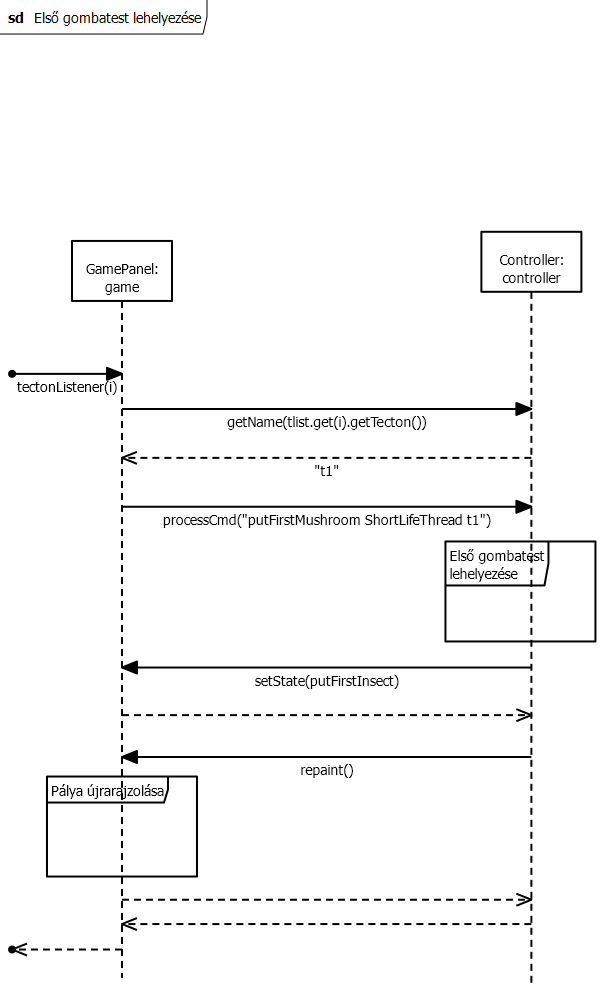
## Játékosokhoz\_szín\_rendelése



A MainWindow meghívja a GamePanel setPlayerColors metódusát

A metódus során a GamePanel lekérdi a kontrollertől az összes játékost, majd belehelyezi egy Map-be, egy színt rendelve hozzájuk úgy, hogy két különböző játékos ugyanazt a színt ne kapja. Ez majd fontos lesz, mivel egy játékos minden objektumát a tektonokon ilyen színnel színezünk ki.

## Első\_gombatest\_lehelyezése

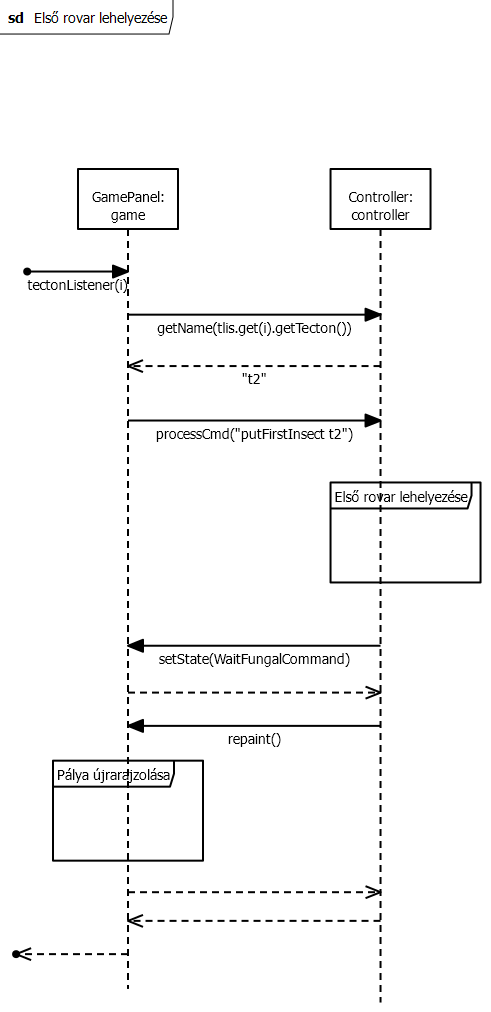


Hívás: Rákattintottunk egy tektonra.

A GamePanel putFirstMushroom állapotban van, azaz vár, hogy az adott gombász helyezze le az első gombatestjét.

Miután a gombész kiválasztotta a tektont és rákattintott, meghívódik a tectonListener metódus, amely paraméterként megkapja, hogy a kattintott tekton hányadik a tlist listában. Lekérjük a kontrollertől a tekton nevét, majd létrehozzuk a parancsot, amellyel lehelyezhetjük az első gombatestet, és odaadjuk ezt a kontrollernek. Ennek során megtörténik az első gombatest lehelyezése, a program tovább lép a következő játékosra, amely egy rovarász, így a GamePanel állapota putFirstInsect-be vált, azaz várja, hogy a rovarász az első rovarát lehelyezze, valamint a repaint() metódus meghívása során újrarajzolódik az ablak.

## Első\_rovar\_lehelyezése

**

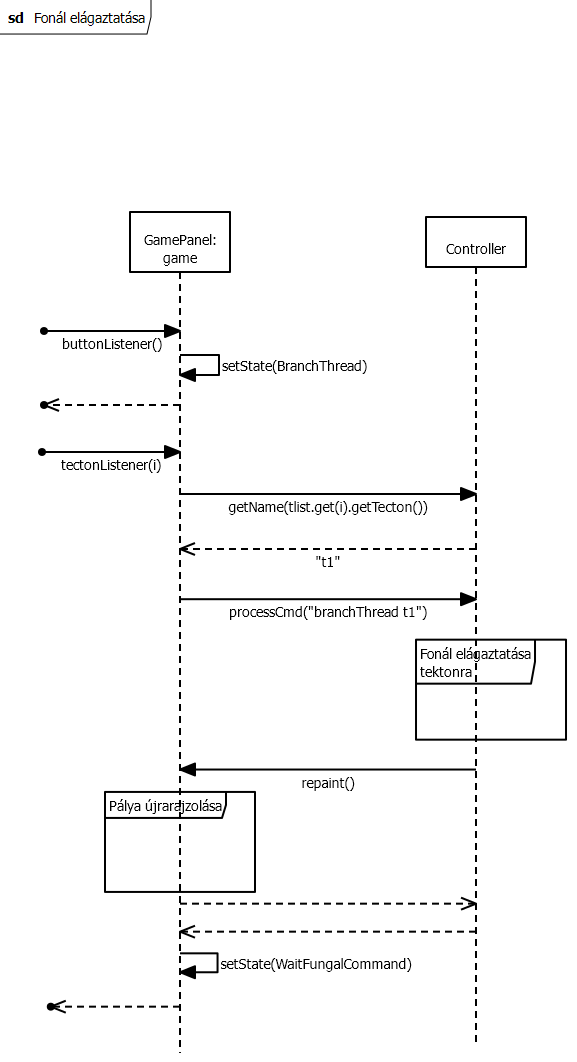
Hívás: Rákattintottunk egy tektonra.

A GamePanel putFirstInsect állapotban van, azaz vár, hogy az adott rovarász helyezze le az első rovarát..

A tektonra kattintás után meghívódik a tectonListener, amely paraméterként megkapja a tekton helyét a tlist tömbben, majd ennek alapján előállítja azt a parancsot, amellyel a rovarász lehelyezheti az első rovarát a modellben.

Miután megtörtént a rovar lehelyezése, mivel több rovarász nincs, ezért új kör következik, lefut az új körnek az inicializálása. Megkezdjük tehát az első kört, a gombászok kezdenek, így a GamePanel a WaitFungalCommand állapotba kerül. Újrarajzoljuk a pályát is.

## Fonál\_elágaztatása



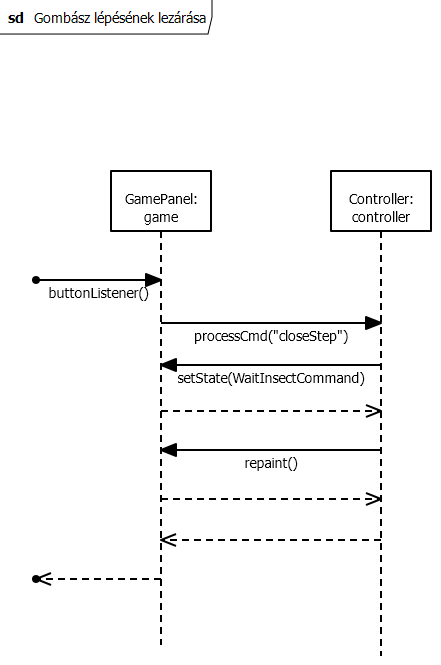
Hívás: rákattintottunk a fonál ágaztatása gombra

A GamePanel a WaitFungalCommand állapotban van, azaz várja, hogy a gombász válassza ki, hogy milyen lépést szeretne végrehajtani.

A gombász a fonál elágaztatása gombra kattintott, ennek hatására a buttonListener fog meghívódni, amely beállítja a GamePanel állepotát BranchThread-re, azaz várni fog hogy kattintsunk egy tektonra, amelyre a gombafonalat elágaztathatjuk.

Miután rákattintottunk a tektonra, meghívódik a tectonListener, átadva paraméterként, hogy a tlist listában hányadik az a tekton, amelyre kattintottunk. Létrehozzuk ez alapján a parancsot, majd kiadjuk a kotnrollernek. Megtörténik a modellben a gombafonál ekágaztatása, majd újrarajzoljuk az ablakot. A GamePanel újra a WaitFunglaCommand állapotba kerül, mivel a gombász még nem zárta le a lépését, így várja, hogy a következő lépések közül válasszon.

## Gombász\_lépésének\_lezárása

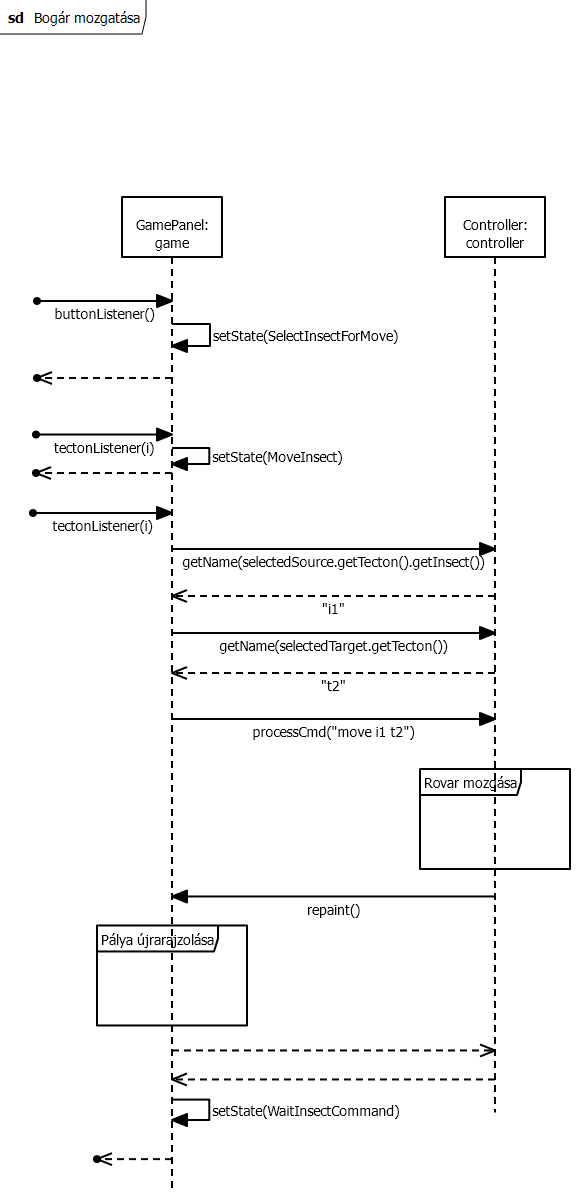
**

Hívás: rákattintunk a lépés befejezése gombra

A GamePanel a WaitFungalCommand állapotban van, azaz várja, hogy a gombász válassza ki, hogy milyen lépést szeretne végrehajtani.

A gombász a lépés lezárása gombra kattintott, ezáltal kiadjuk a kontrollernek a closeStep parancsot. Mivel egy rovarász következik, így nem történik körváltás, beállítjuk a GamePanel állapotát WaitInsectCommand-ra, majd úrjarajzoljuk az ablakot.

## Bogár\_mozgatása

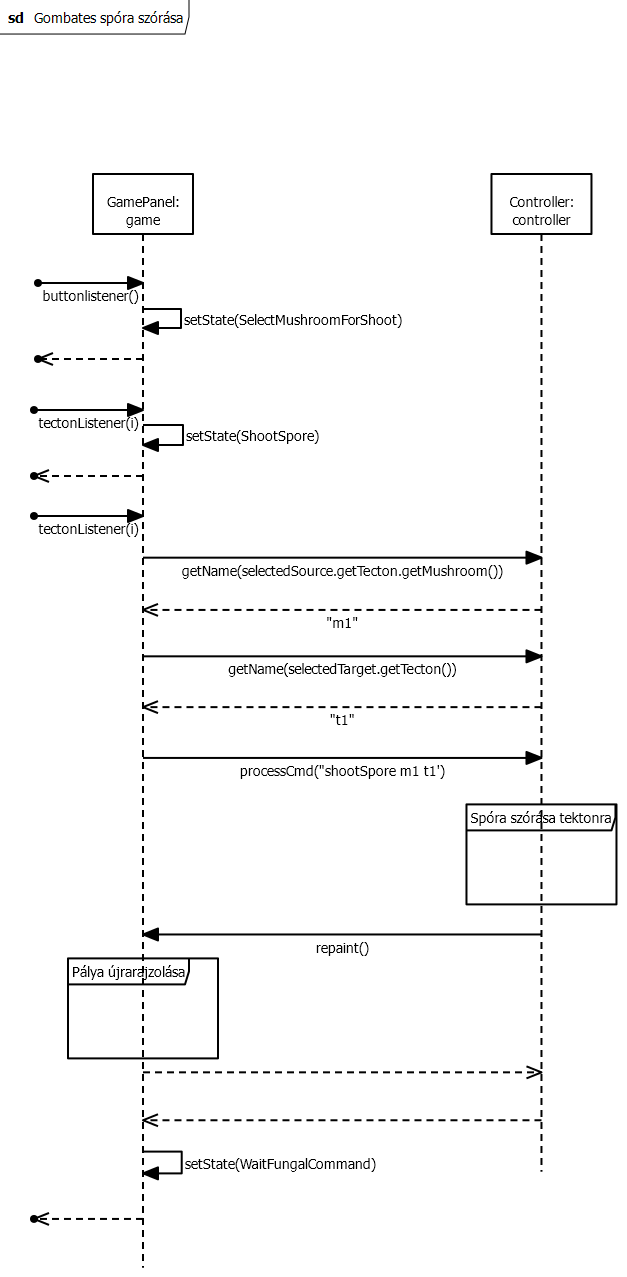


Hívás: A rovarász rákattint a bogár mozgatása gomra

A GamePanel kezdeti állapota WaitInsectCommand, azaz vár, hogy a rovarász válasszon ki egyet a lépései közül.

A rovarász a rovar mozgatása gombra kattint, ennek hatására az állapot átvált SelectInsectFormove-ra, azaz vájuk, hogy a rovarász jelölje ki azt a rovart, amellyel szeretne lépni. Ezután a rovarász egy tektonra kattint. Ekkor A GamePanel átkerül a MoveInsect állapotba, tehát azt a tektont várja, amelyre a rovarat mozgatni akarjuk, illetve a selectedSource változónak átadjuk a kijelölt tektont. A rovarász újra kijelöl egy tektont, a selectedTarget változőnak átadjuk a kijelölt tektont, majd ennek és a selectedSourcenak megfelelően létrehozzuk a paracsot, amit kiadunk a kontrollernek. Megtörténik a rovar mozgása, ezután újrarajzoljuk az ablakot. Mivel a rovarász még nem zárta le a lépését, így a GamePanel a WaitInsectCommand állapotba kerül, azaz vár, hogy a rovaráász kiválasszon egy lépést.

## Gombatest\_spóra szórása

**

Hívás: A gombász rákkatint a spóra szórása gombra.

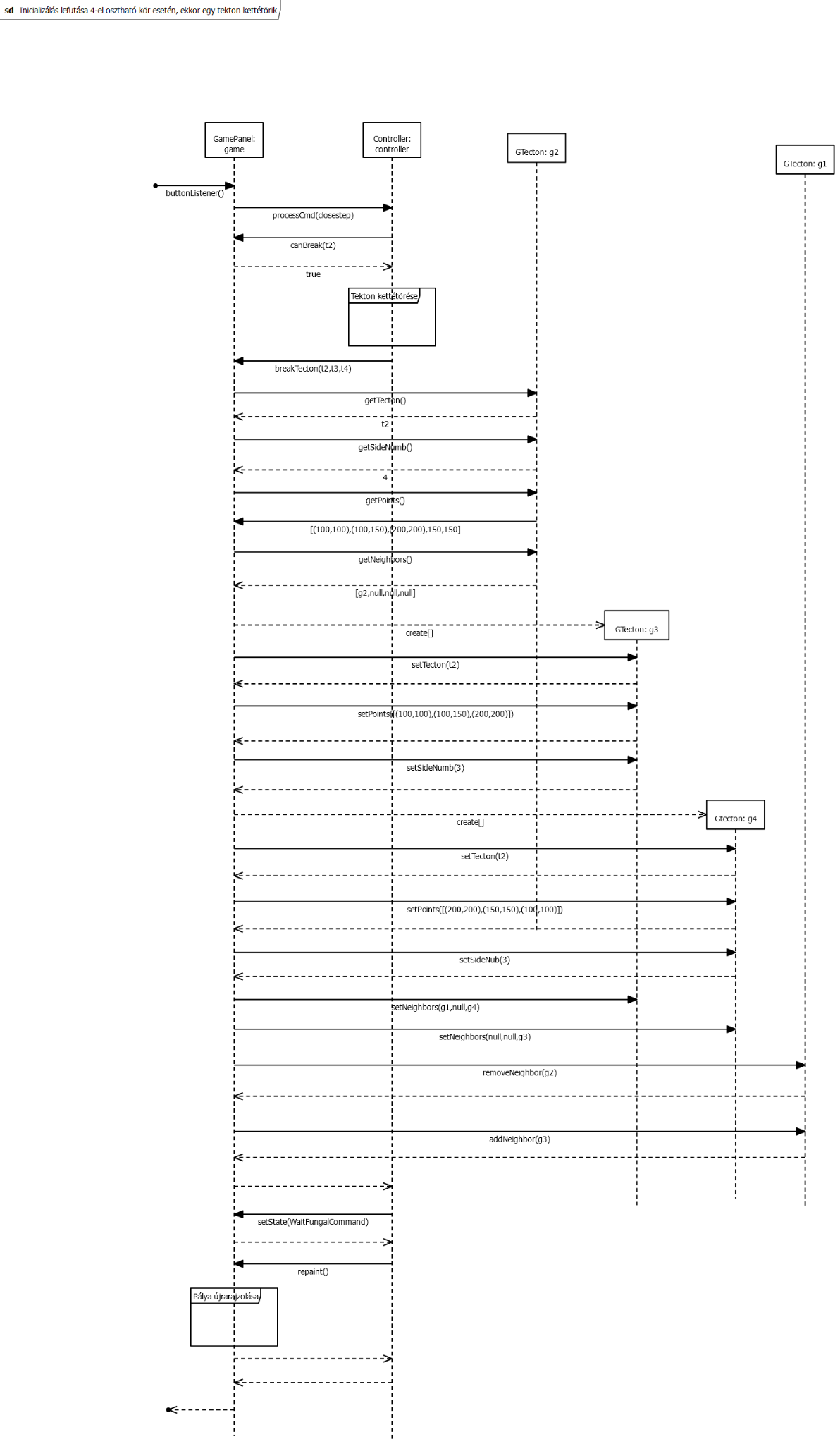
A Gamepanel kezdeti állapota a WaitFungalCommand, azaz vár, hogy a gombász válassza ki milyen lépést szeretne végrehajtani.

A gombász kijelöli a spóra szórása gombot, ennek során a GamePanel a SelectedMushroomForShoot állapotba kerül, azaz várja, hogy jelöljük ki azt a gombatestet, amellyel spórát szeretnénk kilőni.

Ezután kijelölünk egy tektont, amely során a ShootSpore állapotba kerülünk, azaz a program arra vár, hogy kijelöljük, hogy melyik tektonra akarjuk a spórákat kilőni.

Ezután újra kijelölünk egy tektont, létrehozzuk a parancsot, amit átadunk a kontrollernek, megtörténik a spóra kilövése, újrarajzoljuk az ablakot, és újra a WaitFungalCommand állapotba kerülünk.

## Inicializálás\_lefutása\_4\_el\_osztható\_kör\_esetén\_ekkor\_egy\_tekton\_kettétörik



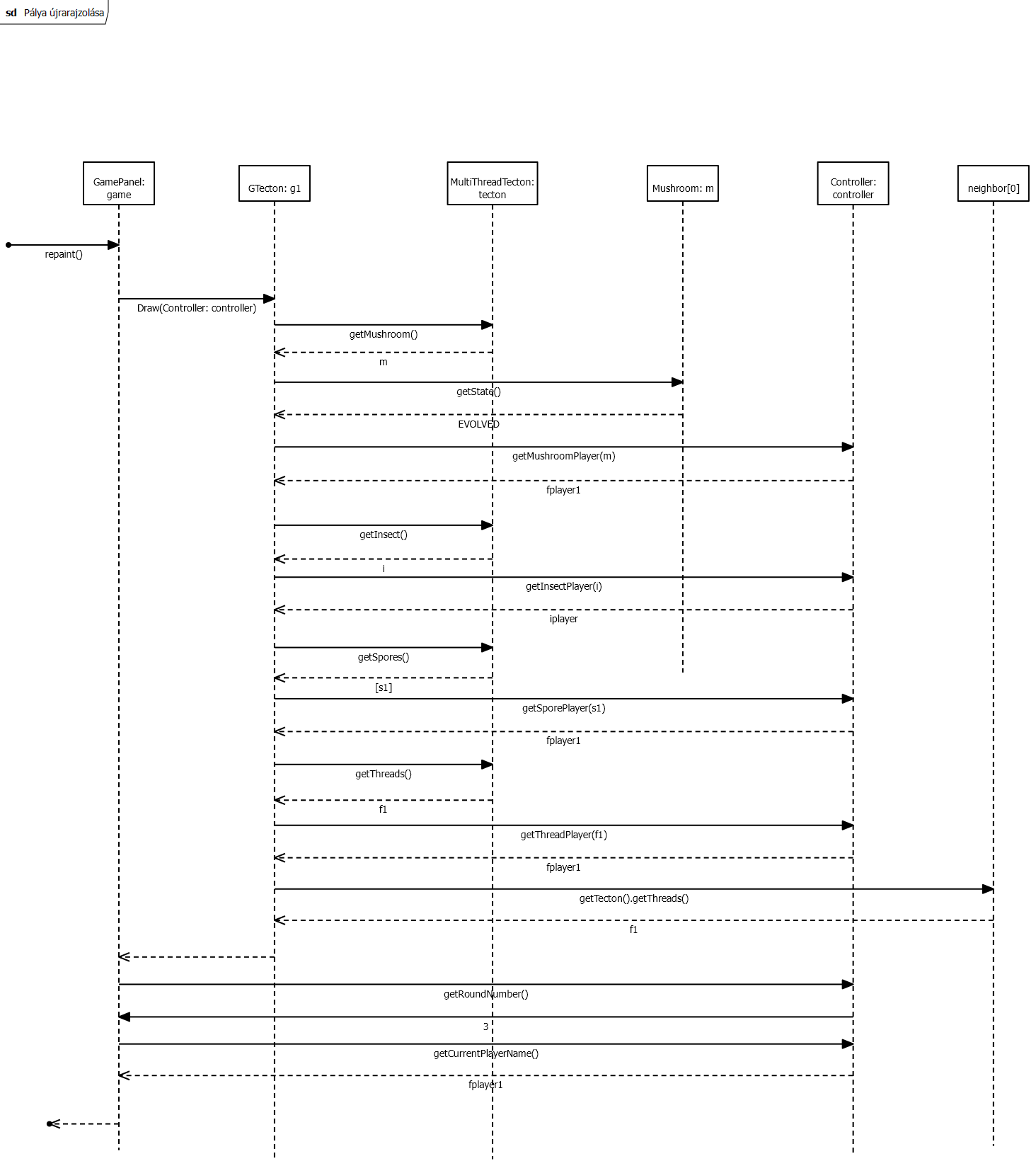
Hívás: A rovarász a lépés lezárása gombra kattint.

A GamePanel a WaitInsectCommand állapotban van, azaz várja, hogy a rovarász kiválassza, hogy milyen lépést szeretne végrehajtani.

A rovarász a lépés lezárása gombra kattint, ezáltal a kontrollernek kiadjuk a closeStep parancsot. Mivel több rovarász játékos nincs, ezáltal egy új kör következik, amely a 4. kör lesz, azaz egy tektonnak ketté kell törnie. Először a kontroller megkérdi, hogy a törésre kijelölt tekton valóban kettétörhető-e, azaz legalább 4 csúcsa van. ezután végrehajtódik a modellben a törés. A GamePanelnek meghívódik a breakTecton metódusa, átadva azt a tektont, amit ketté kell törni, illetve a létrejött két tektont is. A program megkeresi melyik tekton törött ketté végigmenve a tlist listán, amíg meg nem találja. Ezután létrehoz két új tektont, ezeknek a koordinátáját, és oldalszámát az eredeti tekton szerint beállítja, majd beállítja a két új tekton szomszédságait is, valamint azoknak a grafikus tektonoknak a szomszédságait is, amellyel szomszédos volt. Végül a kettétört tektont töröljük a tlist listából.

Miután megtörtént a kör inicializálása, a GamePanel a WaitFungalCommand állapotba kerül, mivel az első gombász jön, és újrarajzolódik az ablak.

## Pálya\_újrarajzolása

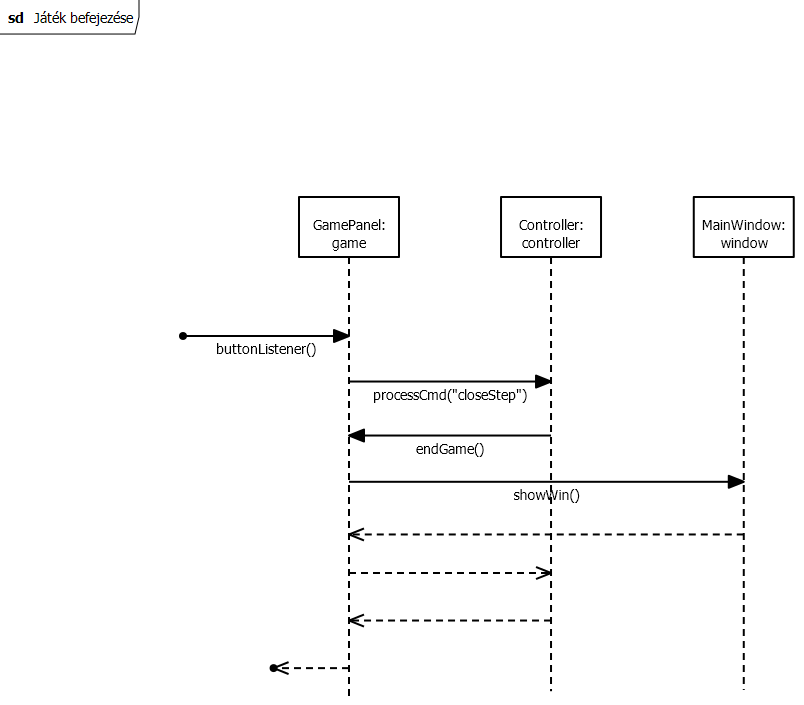
**

Hívás: A kontroller meghívja a GamePanel repaint metódusát.

Ennek során sorban végigmegyünk a tlist-ben eltárolt összes GTectonon, és meghívjuk a Draw metódusukat, paraméterként átadva a kontrollert. Ennek során a GTecton által eltárolt tektontól lekérdezzük előszőr a gombatestet, ha van gombatest, akkor lekérdezzük a gombatest állapotát. illetve lekérjük a kontrollertől a gombatest játékosát, ez kell, hogy tudjuk, hogy milyen színnel rajzoljuk fel a tektonra. Ezután megnézzük, ha van a tektonon rovar, és ha van akkor ennek is lekérjük a játékosát. Ezután megnézzük ha van spóra, és sorban lekérjük ezek játékosát. Végül megnézzük, ha van gombafonál rajta, és ezeknek is lekérjük a játékosát. Ezután kirajzoljuk a tektont, majd feltüntetjük rajta a gombateste, rovart, spórákat, ha vannak. Megnézzük, hogy az adott tektonnal szomszédos tektonokon van-e ugyanolyan gombafajból származó fonal, és ha igen a játékos színével húzunk egy vonalat a két tekton közé, jelezve, hogy elágazik rajtuk a gombafonál.

Miután a tektonokat kirajzoltuk lekérjük a kontrollertől az adott kör számát, és a játékos nevét, aki most van soron, ezeket is megjelenítjük a képernyő felső részén.

## Játék\_befejezése

**

Hívás: A rovarász megnyomja a lépés lezárása gombot.

A GamePanel kezdeti állapota WaitInsectCommand, azaz a program várja, hogy a rovarász kiválassza milyen lépést fog végrehajtani.

A rovarász a lépés lezárása gombra kattint, ennek során a kontrollernek átadjuk a closeStep parancsot. Mivel nincs több rovarász, ezért a következő kör jön azonban a következő kör megegyezik a max körök számával, így vége lesz a játéknak, meghívódik a GamePanel endGame() függvénye, amely meghívja a MainWindow showWin függvényét, amely egy WinPanel segjtségével megjeleníti, hogy ki nyert és a játékosok pontszámait.

# 

## Napló

| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| --- | --- | --- | --- |
| 2025.05.03 17:00 | 3 óra | Fejes  Szolár  Lekli  Toronyi  Zelch | Grafikus felület tervezése |
| 2025.05.04 10:00 | 9 óra | Zelch | Grafikus felület tervezése, osztálydiagram elkészítése, szekvenciadiagramok elkészítése |
| 2025.05.04 10:00 | 1 óra | Fejes | Osztályok leírásának pontosítása, befejezése. |
| 2025.05.04 13:00 | 4 óra | Szolár | GTecton leírása, formázás. |
| 2025.05.04 15:00 | 3 óra | Lekli | Osztályok leírása |
| 2025.05.04 17:00 | 4 óra | Fejes | Kiosztott View-beli osztályok leírása. |
| 2025.05.05 19:00 | 3 óra | Toronyi | Grafikus felület megjelenítésének finomítása, vázlatrajzok elkészítése és részletes dokumentálása |

**13. Grafikus változat elkészítése**

26 – five\_guys

Konzulens:

Huszerl Gábor

Csapattagok

| Lekli Zsolt | JENKHQ | zsoltlekli@gmail.com |
| --- | --- | --- |
| Fejes Ábel | AF1Y1C | fejesabel@gmail.com |
| Zelch Csaba | LK0617 | zch03po@gmail.com |
| Szolár Soma | E05SC6 | somaszolar1@gmail.com |
| Toronyi Zsombor | S8F7DV | toronyizsombi@gmail.com |

2025-05-19

1. **Grafikus változat beadása**
   1. ***Fordítási és futtatási útmutató***
      1. **Fájllista**

| **Fájl neve** | **Méret** | **Keletkezés ideje** | **Tartalom** |
| --- | --- | --- | --- |
| ***Kontroller fájljai*** | | | |
| Controller.java | 35,586 bytes | ‎Tuesday, ‎April ‎22, ‎2025, ‏‎2:53:42 PM | Kontroller osztály. |
| FungusPlayer.java | 5,023 bytes | Tuesday, ‎April ‎22, ‎2025, ‏‎2:53:42 PM | Gombász játékos osztály. |
| InsectAssociation.java | 2,426 bytes | ‎Friday, ‎April ‎25, ‎2025, ‏‎5:54:35 PM | Rovar és rovarász lépési amiket megtehet. |
| InsectPlayer.java | 4,298 bytes | Tuesday, ‎April ‎22, ‎2025, ‏‎2:53:42 PM | Rovarász játékos osztály. |
| MushroomAssociation.java | 1,857 bytes | Friday, ‎April ‎25, ‎2025, ‏‎5:54:35 PM | Gombatest és gombász lépései amiket megtehet. |
| Player.java | 874 bytes | Tuesday, ‎April ‎22, ‎2025, ‏‎2:53:42 PM | Játékos osztály. |
| ***Modell fájljai*** | | | |
| AbsorbingTecton.java | 9,274 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | AbsorbingTecton osztály. |
| DividingSpore.java | 740 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | DividingSpore osztály. |
| FungalThread.java | 10,013 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Gombafonál osztály. |
| IFungalThreadController.java | 744 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Gombafonál interfésze a kontroller felé. |
| IFungalThreadView.java | 124 bytes | Saturday, May 17, 2025, 10:27:20 AM | Gombafonál interfésze a view felé. |
| IInsectController.java | 431 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Rovar interfésze a kontroller felé. |
| IInsectView.java | 125 bytes | Saturday, May 17, 2025, 10:27:20 AM | Rovar interfésze a view felé. |
| IMushroomController.java | 870 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Gombatest interfésze a kontroller felé. |
| IMushroomView.java | 255 bytes | Saturday, May 17, 2025, 10:27:20 AM | Gombatest interfésze a view felé. |
| Insect.java | 5,250 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Rovar osztály. |
| InsectState.java | 678 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Rovar állapotait tartalmazó enum. |
| ISporeController.java | 287 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Spóra interfésze a kontroller felé. |
| ITectonController.java | 814 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Tecton interfésze a kontroller felé. |
| ITectonView.java | 410 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Tecton interfésze a view felé. |
| KeepThreadTecton.java | 8,662 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Fonalat megörző tekton osztály. |
| LongLifeThread.java | 861 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Hosszan megörző fonál osztály. |
| MultiThreadTecton.java | 9,157 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Tecton, amelyen több gombafonál elágaztatható. |
| Mushroom.java | 6,147 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Gombatest osztály. |
| MushroomState.java | 499 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Gombatest állapotait tároló enum. |
| NoCutSpore.java | 786 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | NoCutSpore osztály. |
| ParalysingSpore.java | 832 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | ParalysingSpore osztály. |
| ShortLifeThread.java | 992 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Röviden megörző fonál osztály. |
| SingleThreadTecton.java | 9,557 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | SingleThreadTecton osztály. |
| SlowingSpore.java | 680 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | SlowingSpore osztály. |
| SpeedSpore.java | 768 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | SpeedSpore osztály. |
| Spore.java | 1,374 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Spóra osztály. |
| Tecton.java | 10,506 bytes | Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Tecton osztály. |
| timeToDie.java | 611 bytes | ‎Wednesday, ‎April ‎16, ‎2025, ‏‎3:07:29 AM | Segédosztály hogy a gombafonalak vágás után ne egybőlpusztuljanak el. |
| ***Program*** | | | |
| Main.java | 329 bytes | ‎Friday, ‎April ‎25, ‎2025, ‏‎6:48:07 PM | A main tartalmazó osztály. |
| ***View fájljai*** | | | |
| DrawingPanel.java | 15,296 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:34:31 AM | Grafikus felhasználói felületet megjelenítő osztály, egérkattintási eseményeket kezeli. |
| FieldCreator.java | 2,504 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:34:31 AM | Létrehoz egy egyedi formázású JLabel példányt, amely adott szöveget jelenít meg. |
| GamePanel.java | 17,389 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:34:31 AM | A játék fő panelja, játékosneveket, körszámot, gombokat és DrawingPanelt tartalmazza. |
| GameState.java | 541 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:34:31 AM | A GameState enum a játékban előforduló különböző állapotokat reprezentálja. |
| GInsect.java | 3,980 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:34:31 AM | Insect entitások grafikus megjelenítéséért felelős. |
| GMushroom.java | 3,867 bytes | ‎Saturday, ‎May ‎17, ‎2025, ‏‎10:34:31 AM | Mushroom entitások grafikus megjelenítéséért felelős. |
| GTecton.java | 16,180 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:34:31 AM | GTecton osztály a tektonokat reprezentálja a játékban. |
| MainWindow.java | 3,465 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:34:31 AM | A játék főablakáért felelős, amely örökli a JFrame funkcióit. |
| NameEntryPanel.java | 6,260 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:34:31 AM | A játékosok neveinek beviteléért felelős grafikus komponens. |
| ObjectColor.java | 254 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:34:31 AM | Enum, az objektumok színezésére szolgáló színek halmazát definiálja. |
| StartScreenPanel.java | 5,043 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:34:31 AM | A StartScreenPanel osztály a játék kezdőképernyője. |
| WinPanel.java | 6,994 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:34:31 AM | A WinPanel osztály a játék eredménylistájának (leaderboard) megjelenítésére szolgál. |
| ***Egyéb fájlok*** | | | |
| bug1.png | 17,989 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:27:20 AM | Piros színű rovar képe. |
| bug2.png | 18,158 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:27:20 AM | Kék színű rovar képe. |
| bug3.png | 18,158 bytes | ‎Saturday, ‎May ‎17, ‎2025, ‏‎10:27:20 AM | Zöld színű rovar képe. |
| bug4.png | 17,990 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:27:20 AM | Sárga színű rovar képe. |
| mush1.png | 11,555 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:27:20 AM | Piros színű gombatest képe. |
| mush2.png | 12,509 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:27:20 AM | Kék színű gombatest képe. |
| mush3.png | 11,851 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:27:20 AM | Zöld színű gombatest képe. |
| mush4.png | 11,185 bytes | ‎Saturday, ‎May ‎17, ‎2025, ‏‎10:27:20 AM | Sárga színű gombatest képe. |
| spore1.png | 10,202 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:27:20 AM | Piros színű spóra képe. |
| spore2.png | 10,133 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:27:20 AM | Kék színű spóra képe. |
| spore3.png | 10,105 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:27:20 AM | Zöld színű spóra képe. |
| spore4.png | 10,178 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:27:20 AM | Sárga színű spóra képe. |
| ok.txt | 1,961 bytes | Saturday, ‎May ‎17, ‎2025, ‏‎10:27:20 AM | A tektonok leírását tartalmazza, ez alapján építendő fel a pálya. |

## 

* + 1. **Fordítás és telepítés**

A programot a BME kari felhőjében biztosított, Windows 10 20H2 – JDK-Eclipse-WSU sablonnal ellátott virtuális gépen kell tudni fordítan. A játék fordítására kizárólag a JDK szükséges, amely biztosítja a java és javac parancsok elérhetőségét.

A következő parancs segítségével a **program** package-ben található állomány fordításra kerül (a package-t tartalmazó **src** mappából kiadva):

**javac -d . program/Main.java**

* + 1. **Futtatás**

A programot a BME kari felhőjében biztosított, Windows 10 20H2 – JDK-Eclipse-WSU sablonnal ellátott virtuális gépen kell tudni futtatni. A játék futtatásához kizárólag a JDK szükséges, amely biztosítja a java és javac parancsok elérhetőségét.

A fordítást követően az alábbi parancs segítségével a program futtatható:

**java program.Main**

* 1. ***Értékelés***

| **Tag neve** | **Tag neptun** | **Munka százalékban** |
| --- | --- | --- |
| Fejes Ábel | AF1Y1C | 20 |
| Lekli Zsolt | JENKHQ | 20 |
| Szolár Soma | E05SC6 | 20 |
| Toronyi Zsombor | S8F7DV | 20 |
| Zelch Csaba | LK0617 | 20 |

* 1. ***Napló***

| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| --- | --- | --- | --- |
| 2025.05.10. | 8 óra | Zelch | Grafikus felület készítése |
| 2025.05.11. | 9 óra | Zelch | Grafikus felület készítése |
| 2025.05.11. | 4 óra | Fejes | NamePanel, StartPanel elkészítése. |
| 2025.05.11. | 7 óra | Szolár | WinPanel GamePanel, spórák kirajzolásának elkészítése. |
| 2025.05.15. | 2,5 óra | Fejes | Gombák, spórák, rovarok kirajzolása tektonokra. Tesztelés. |
| 2025.05.17. | 3 óra | Lekli | Dokumentum elkészítése |
| 2025.05.17. | 4 óra | Toronyi | Információs ablak implementálása, megjelenítési problémák javítása, JavaDoc dokumentáció elkészítése |
| 2025.05.17. | 5 óra | Szolár | Grafikus felület szépítése |