**8. Részletes tervek**

26 – five\_guys

Konzulens:

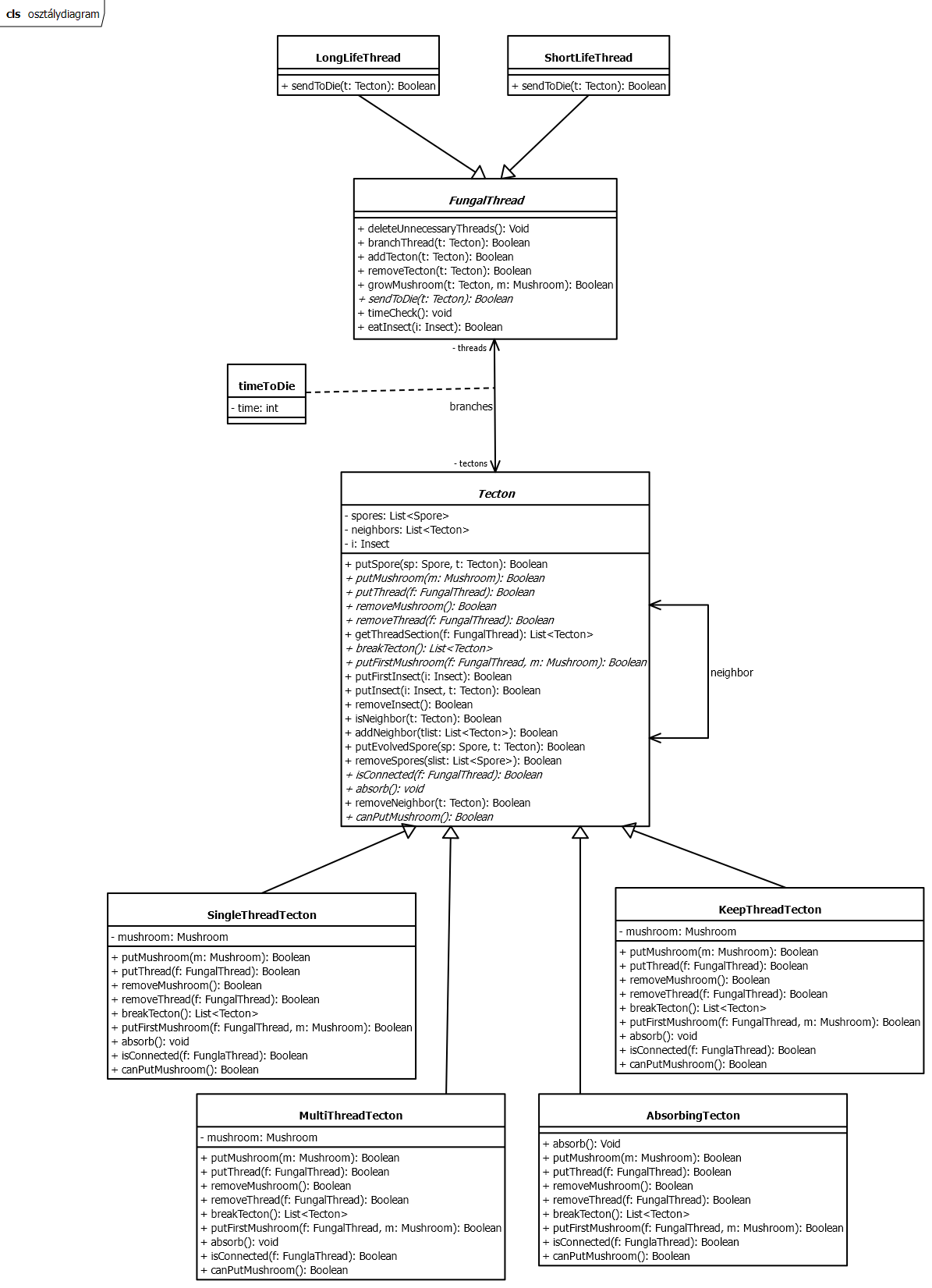
Huszerl Gábor

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:***

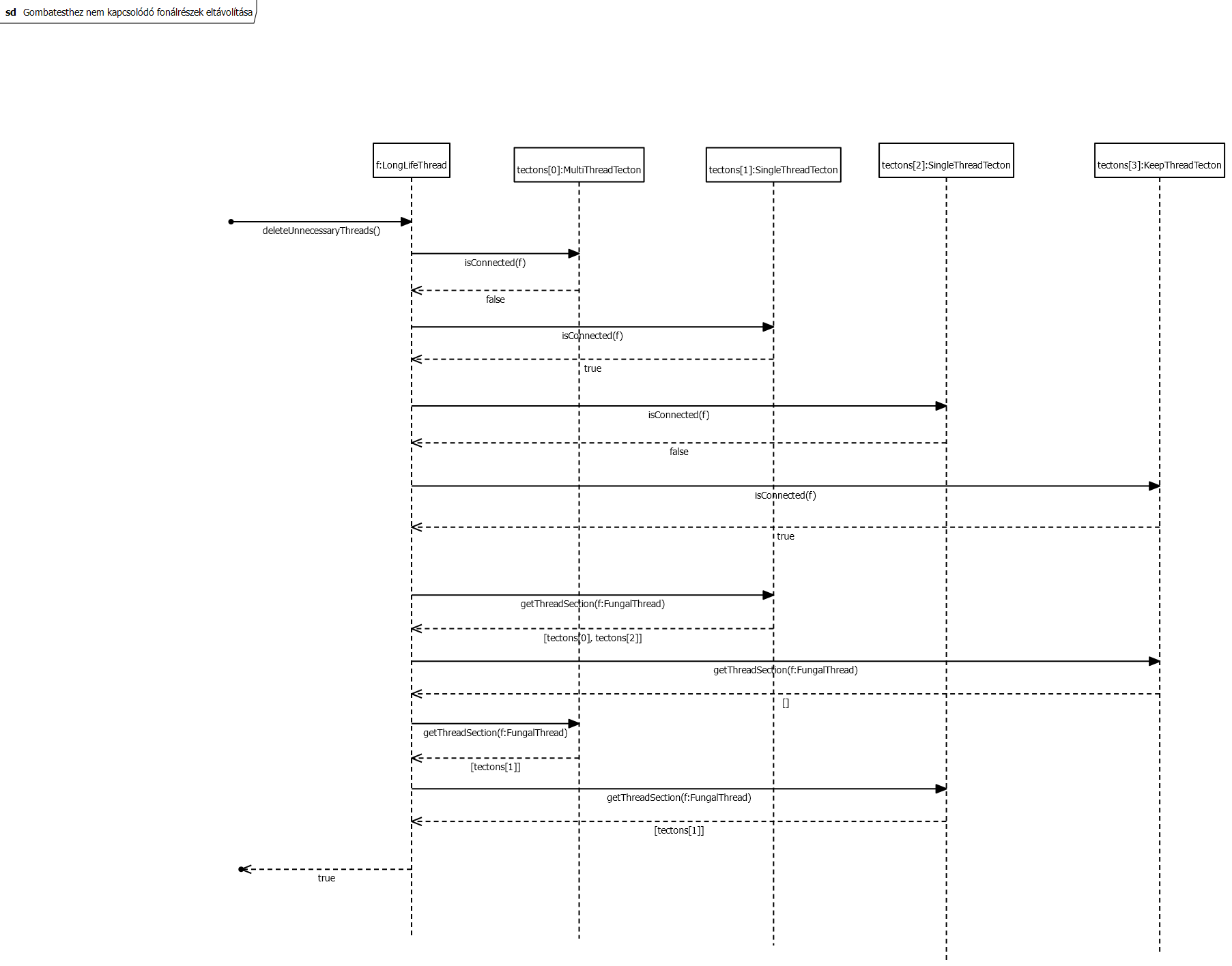
******

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

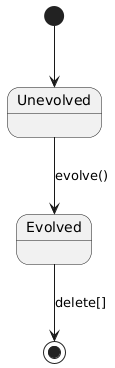
KÜLÖNBEN

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)

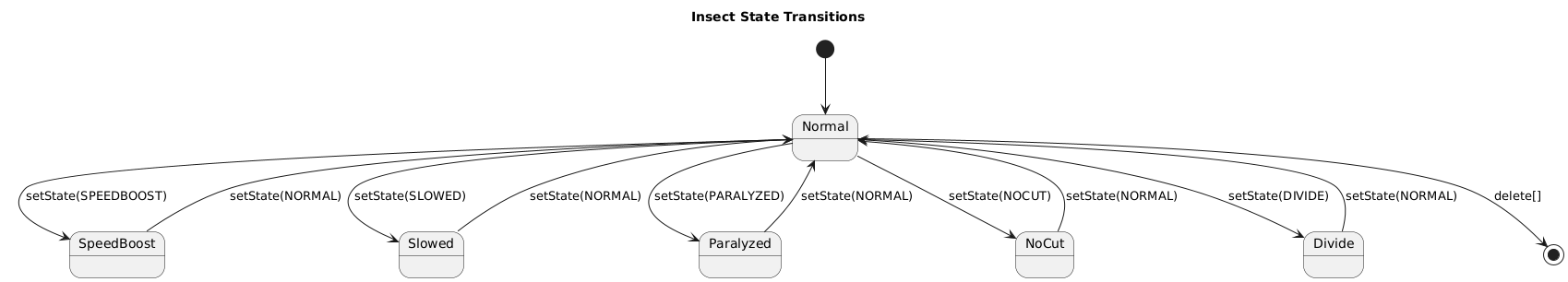
* **+ boolean generateSpore(Spore sp) -** a paraméterként kapott spóra fajta alapján egy vele megegyező fajtájú spórát hoz létre, amit ki tud majd lőni

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

* ***Egy kibaszott nagy megjegyzés, mikor a játékos lép, tehát act parancsok esetén mindig de mindig le kell ellenőrizni, hogy a játékos tényleg a saját dolgaival lépett, ez fontos, mivel másképp előfordul, hogy egy másik játékos dolgaival lépünk, és a játék helyesnek ítéli, az meg kurva fos***

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. |