**4. Analízis modell II.**

26 – five\_guys

Konzulens:

Huszerl Gábor

Csapattagok

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

2025. március 10.

# **Analízis modell kidolgozása**

## ***Objektum katalógus***

* + 1. ***Rovar***

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

* + 1. ***Gombafonál***

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

* + 1. ***Gombatest***

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

* + 1. ***Tektonok***

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

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

* + 1. ***SzívóTekton***

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

SzívóTektonra nem lehet tenni gombatestet.

* + 1. ***EgyfonalasTekton***

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

* + 1. ***TöbbfonalasTekton***

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

* + 1. ***Spóra***

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

* + - 1. ***Gyorsítóspóra***

Olyan spóra, amely elfogyasztásával a rovar mégegyszer jöhet.

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

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

* + - 1. ***Lassítóspóra***

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

* + 1. ***BénítóSpóra***

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

## ***Statikus struktúra diagramok***

## ***Osztályok leírása***

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

### Insect(rovar)

#### Felelősség

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

#### Ősosztályok

Nincs ősosztálya.

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

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

#### Attribútumok

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

#### Metódusok

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

### FungalThread(gombafonál)

#### Felelősség

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

#### Ősosztályok

Nincs ősosztálya

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

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

#### Attribútumok

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

#### Metódusok

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

### Mushroom(gombatest)

#### Felelősség

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

* + - * **Ősosztályok**

Nincs ősosztálya

#### Interfészek

*Nem valósít meg interface-t*

#### Asszociációk

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

#### Attribútumok

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

#### Metódusok

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

### Tecton(tekton)

#### Felelősség

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

#### Ősosztályok

Nincs ősosztálya

#### Interfészek

Nem valósít meg interface-t.

#### Asszociációk

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

#### Attribútumok

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

#### Metódusok

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

### AbsorbingTecton(*SzívóTekton*)

#### Felelősség

Ő felel a Tecton példányosítható, szívó változatának a reprezentálásáért.

#### Ősosztályok

Tecton -> AbsorbingTecton

#### Interfészek

Nem valósít meg interface-t.

#### Asszociációk

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

#### Attribútumok

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

#### Metódusok

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

### SingleThreadTecton(*EgyfonalasTekton*)

#### Felelősség

Ő felel a Tecton példányosítható, egyfonalas változatának a reprezentálásáért.

#### Ősosztályok

Tecton -> SingleThreadTecton

#### Interfészek

Nem valósít meg interface-t.

#### Asszociációk

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

#### Attribútumok

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

#### Metódusok

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

### MultiThreadTecton(*TöbbfonalasTekton*)

#### Felelősség

Ő felel a Tecton példányosítható, többfonalas változatának a reprezentálásáért.

#### Ősosztályok

Tecton -> MultiThreadTecton

#### Interfészek

Nem valósít meg interface-t.

#### Asszociációk

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

#### Attribútumok

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

#### Metódusok

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

### Spore (Spóra)

#### Felelősség

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

#### Ősosztályok

Nincs ősosztálya.

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

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

#### Attribútumok

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

#### Metódusok

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

### SpeedSpore (Gyorsítóspóra)

#### Felelősség

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

#### Ősosztályok

Spore (Spóra) -> SpeedSpore (Gyorsítóspóra)

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

Csak olyanok, amelyet az ősosztályától örököl.

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl.

#### Metódusok

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

### NoCutSpore

#### Felelősség

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

#### Ősosztályok

Spore (Spóra) -> NoCutSpore

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

Csak olyanok, amelyet az ősosztályától örököl.

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl.

#### Metódusok

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

### SlowingSpore (LassítóSpóra)

#### Felelősség

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

#### Ősosztályok

Spore (Spóra) -> SlowingSpore

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

Csak olyanok, amelyet az ősosztályától örököl.

#### Attribútumok

Csak olyanok, amelyet az ősosztályától örököl.

#### Metódusok

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

### ParalyserSpore (BénítóSpóra)

#### Felelősség

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

#### Ősosztályok

Spore (Spóra) -> ParalyserSpore

#### Interfészek

Nem valósít meg interfészt.

#### Asszociációk

Csak olyanok, amelyet az ősosztályától örököl.

#### Attribútumok

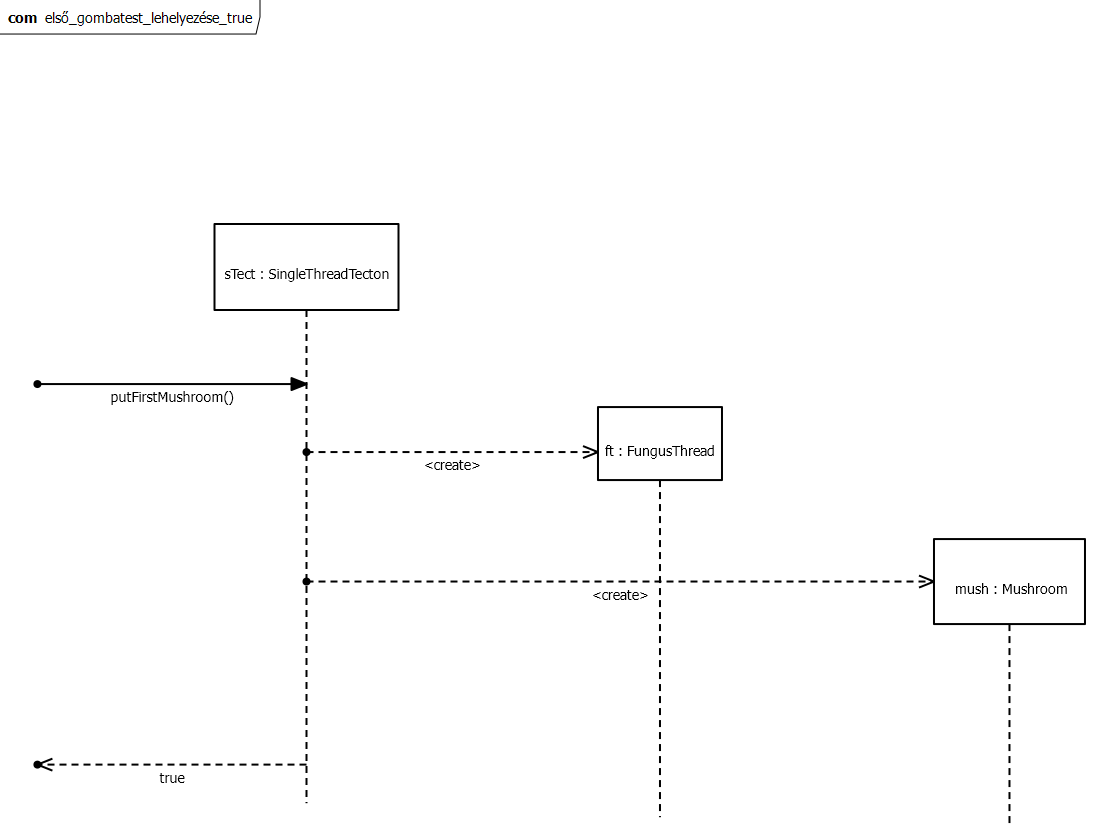
Csak olyanok, amelyet az ősosztályától örököl.

#### Metódusok

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

## ***Szekvencia diagramok***

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

****

A folyamatot a Controller külső komponens indítja.

Ezen a diagramon három objektum található:

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

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

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

****

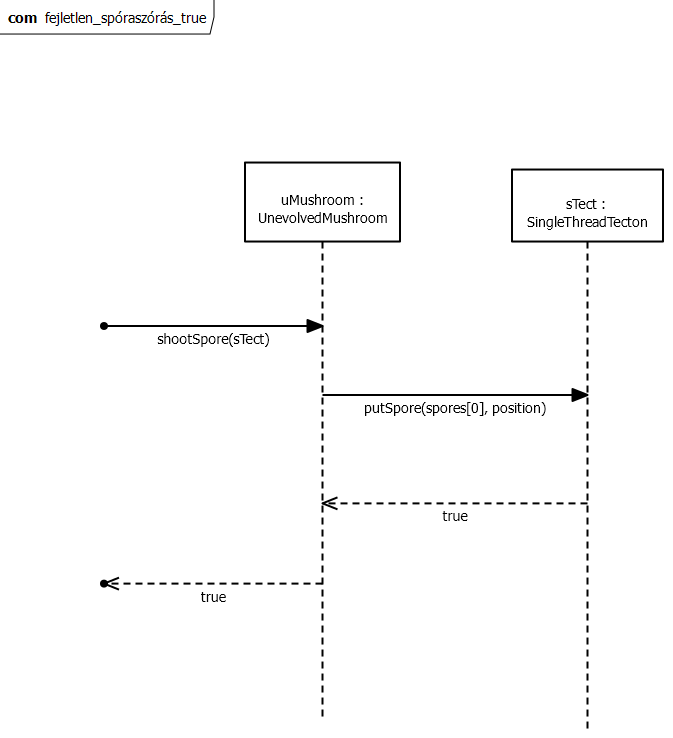
A folyamatot a Controller külső komponens indítja.

Ezen a diagramon egy objektum található:

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

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

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

****

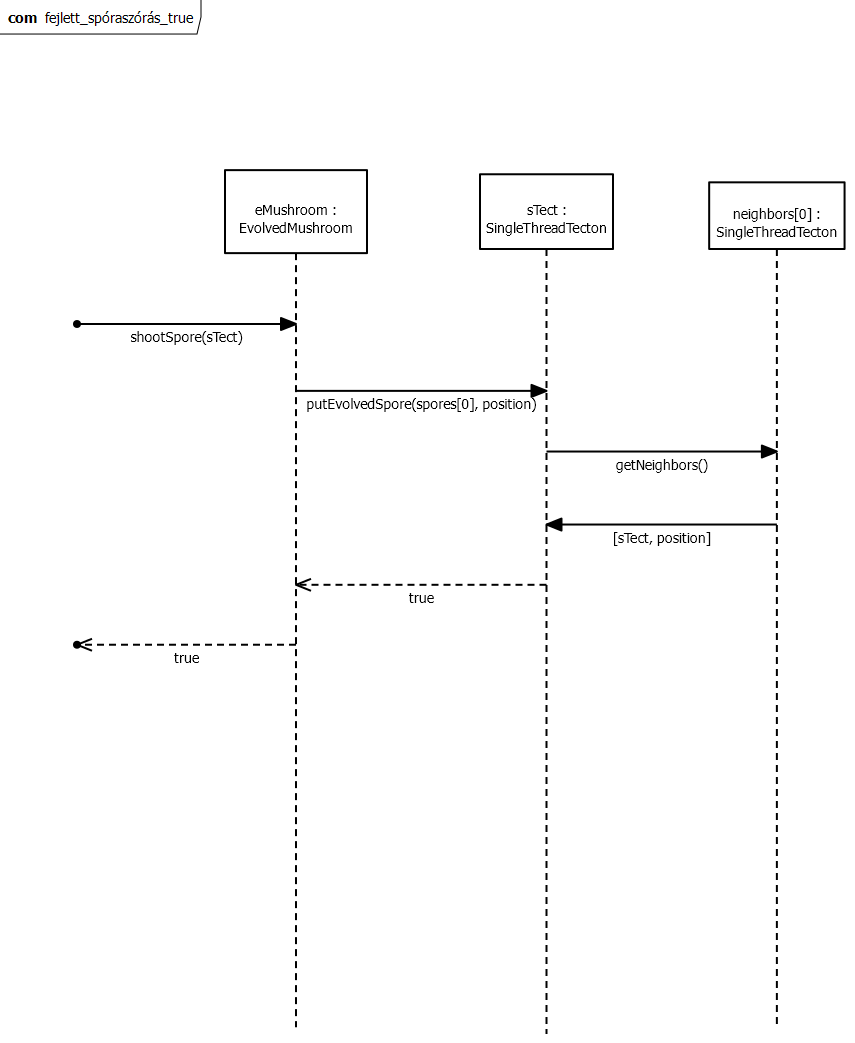
A folyamatot a Controller külső komponens indítja.

Ezen a diagramon két objektum található:

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

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

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



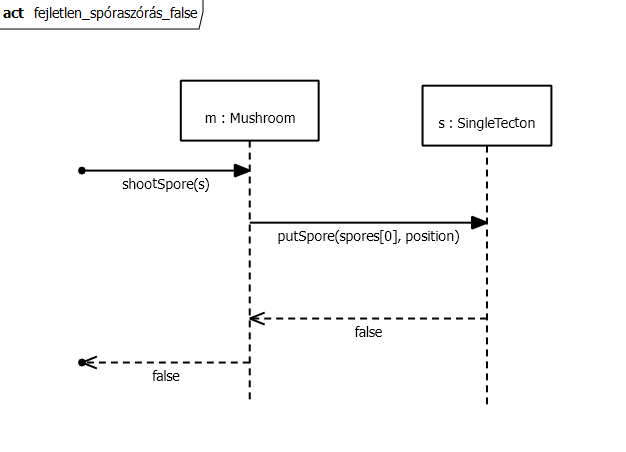
A folyamatot a Controller külső komponens indítja.

Ezen a diagramon három objektum látható:

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

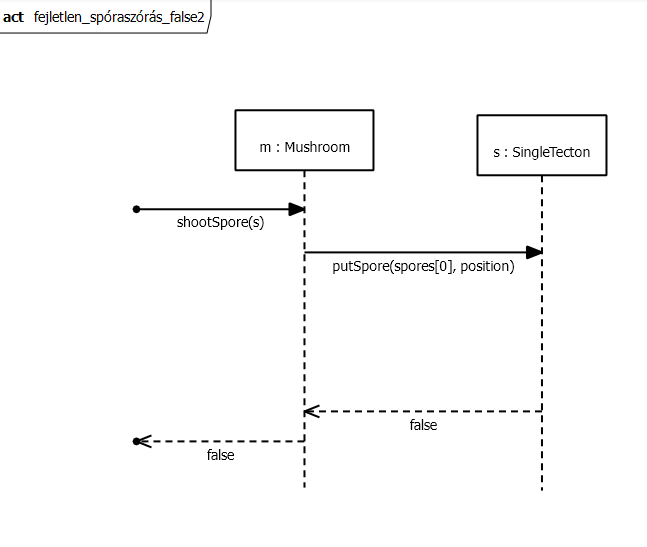
Egy külső komponens meghívja az eMushroom **shootSpore** metódusát, aminek paraméterül adja az sTect tektont. Ezután ez a függvény meghívja az sTect **putEvolvedSpore** függvényét, amely paraméterül kapja az eMushroom által termelt spórák közül a legkorábban termeltet (spores[0]) és azt a tektont, amelyen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

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



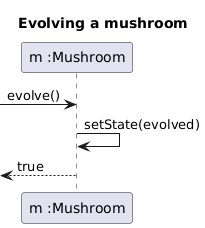
* A folyamatot a controller indítja meg.
* A diagrammon szereplő objektumok:
  + **m**:**Mushroom:** Ő lövi ki a spórát
    - **position:SingleTecton**: A tekton, amin a gomba van.
    - state:MushroomState=UnevolvedMushroom (Csak a szomszédos tektonokra tud spórát tenni.)
    - spores:List<Spore>: A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

**Fejletlen gombatest spóra szórása olyan tektonra, amelyre nem tud, mert már túl sok spóra van rajta. <-Elavult. Már nincs felsőkorlát spóraszámra egy tektonon.**



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

**Gombatest fejlődése**

****

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

***Gombatest elpusztulása***

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

Ezen a diagramon a következő objektumok találhatók:

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

**shotSpore(t1)**

egy külső kontroller hívja meg az e objektum ezen metódusát

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

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

**setshootedSporesCount(shootedSoresCount+1)**

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

leellenőrizzük, hogy kilőttük-e a 10. spórát

**removeMushroom()**

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

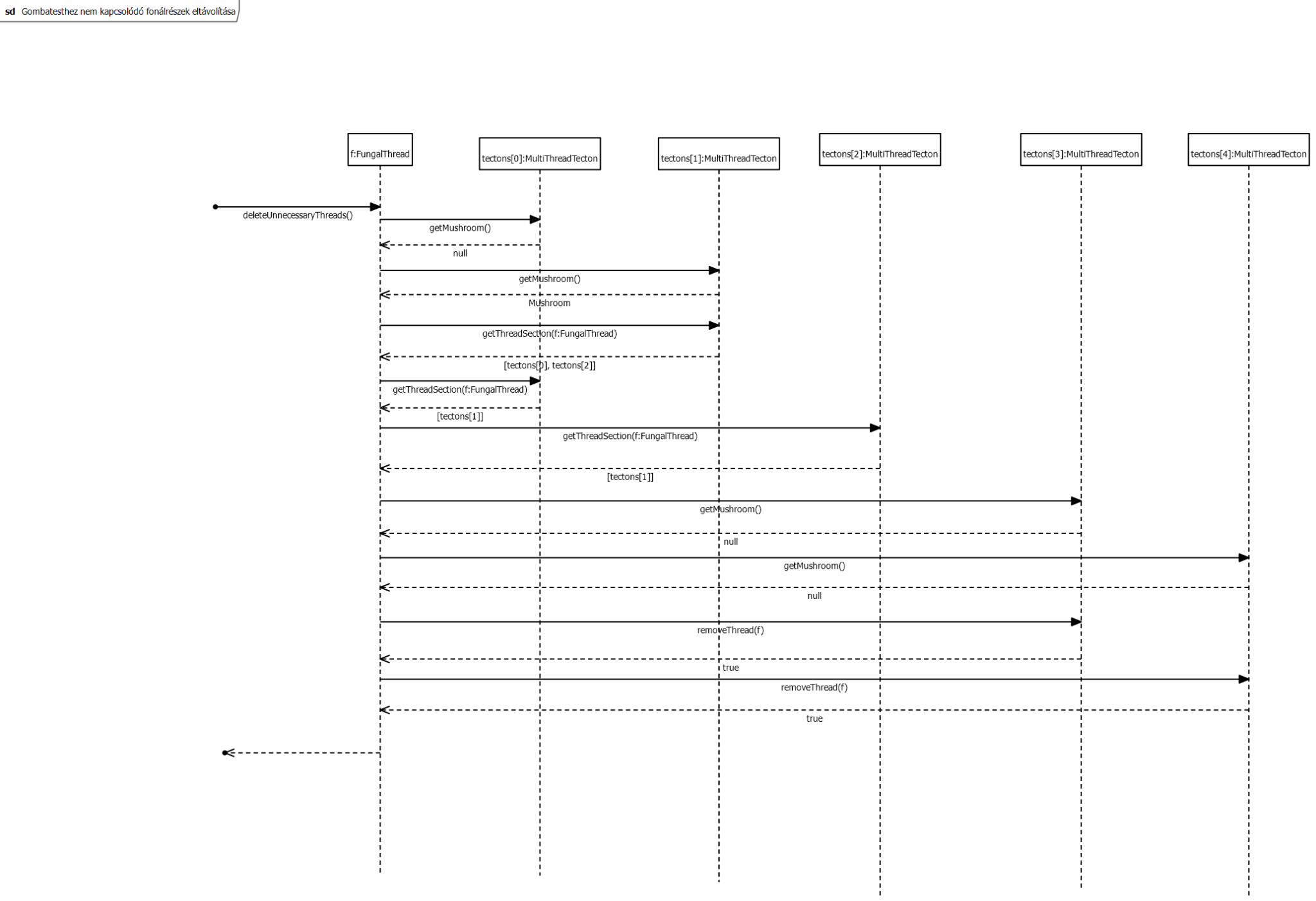
**deleteUnnecessaryThreads()**

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

**getshootedSporesCount()**

a kontroller lekérdi, hogy hány spórát lőttünk ki eddig

ha ez megegyezik 10-el, akkor töröljük a gombatestet



***Gombatesthez nem kapcsolódó fonálrészek eltávolítása* konzisztencia**

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

Ezen a diagramon a következő objektumok találhatók:

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

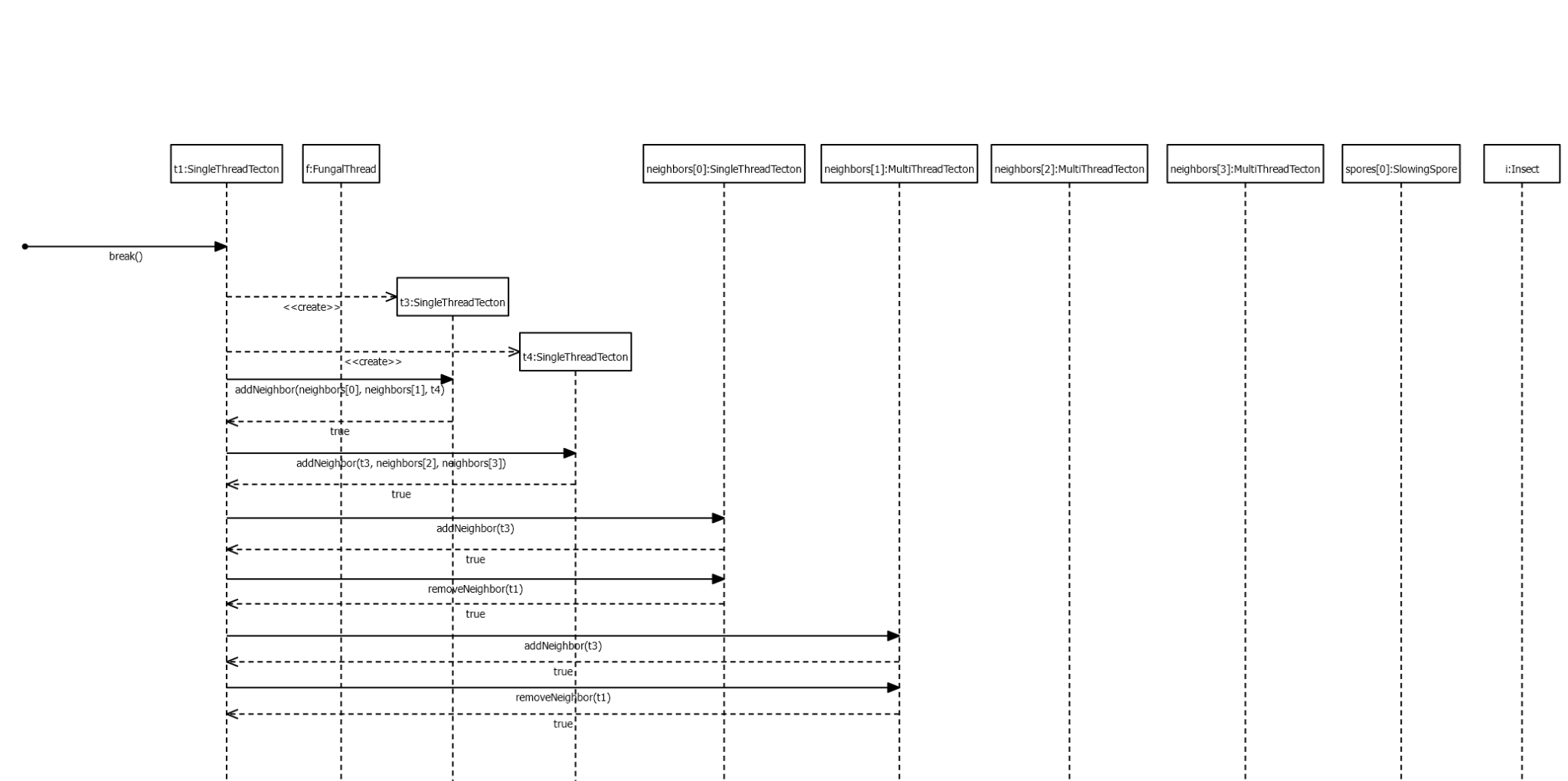
A tectons az f gombafonál egy tömbje, amelyben azokat a tektonokat tárolja, amelyen elágazik.

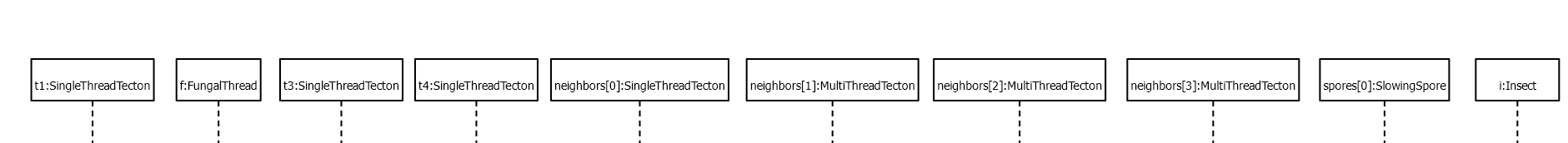
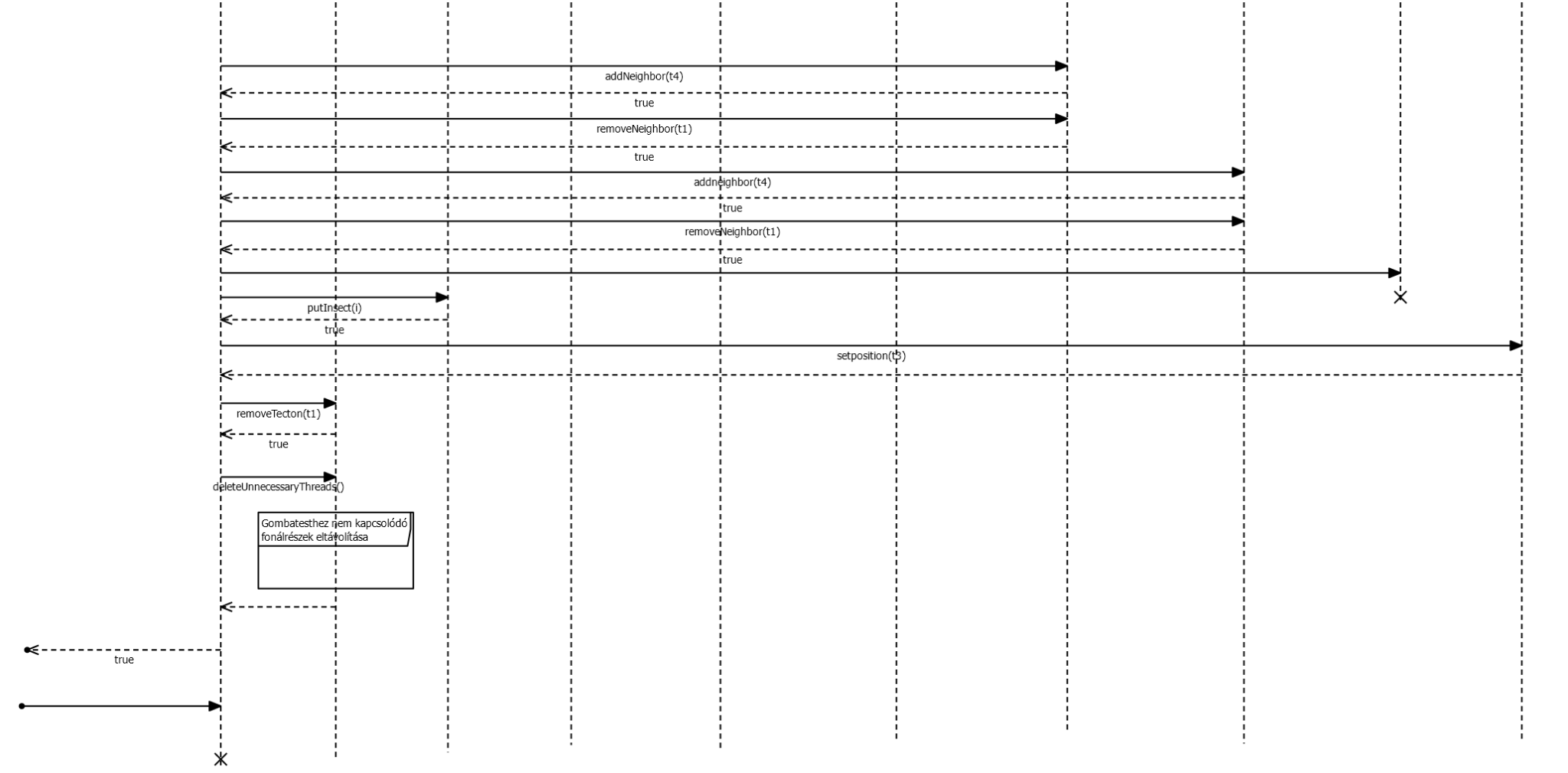
Ezeken a tektonokon sorban végigiterál, és megkérdezi, ha van-e rajtuk gombatest. Ha nincs megnézi a sorban következő tektonra is, ha van, és ez a gombatest ugyanahhoz a gombafajhoz származik, akkor megkeresi azt a fonálrészt, amely ezzel a gombatesttel összeköttetésben van.

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

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

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

***Tekton kettétörése***

***Folytatás***

Egy külső kontroller meghívja a t1 tekton break() metódusát

Ezen a diagramon a következő objektumok találhatók:

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

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

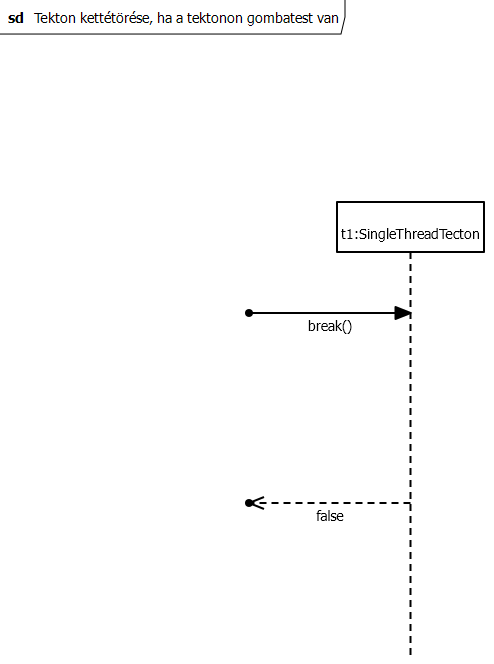
Ha van a t1 tektonon spóra, akkor ezt töröljük.

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

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

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

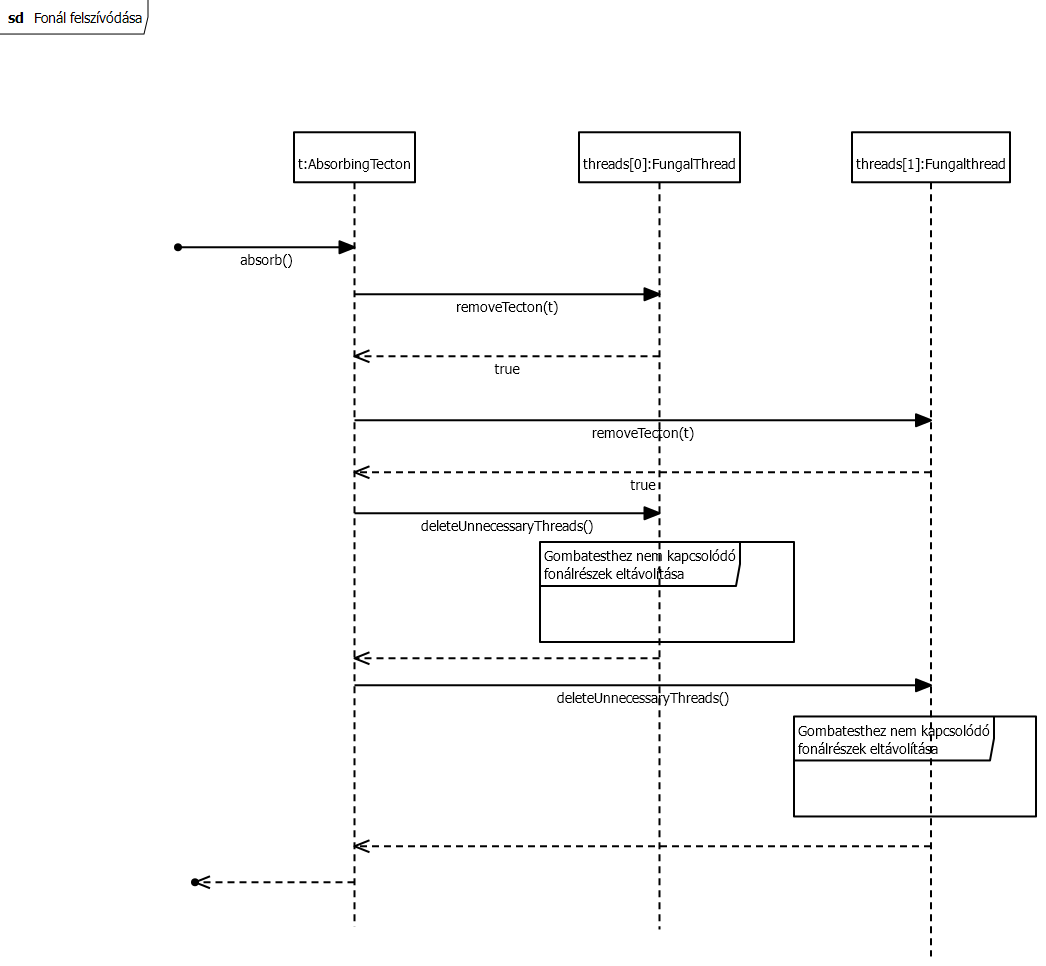
***Tekton kettétörése, ha a tektonon gombatest van***

**

Egy külső kontroller meghívja a t1 tekton break() metódusát

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

***Fonál felszívódása tektonról***

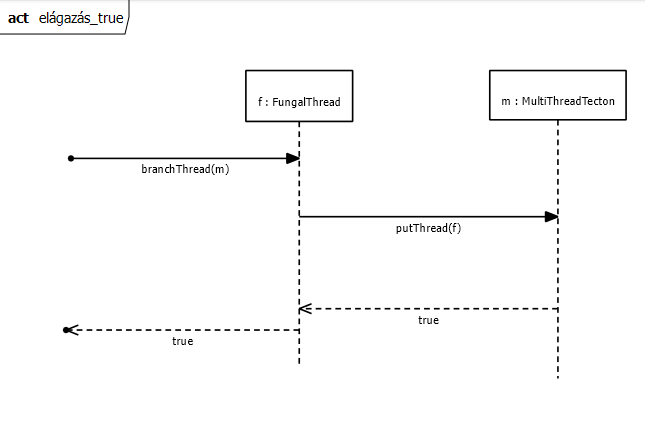
Egy külső kontroller meghívja a t tekton absorb() metódusát

Ezen a diagramon a következő objektumok találhatók:

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

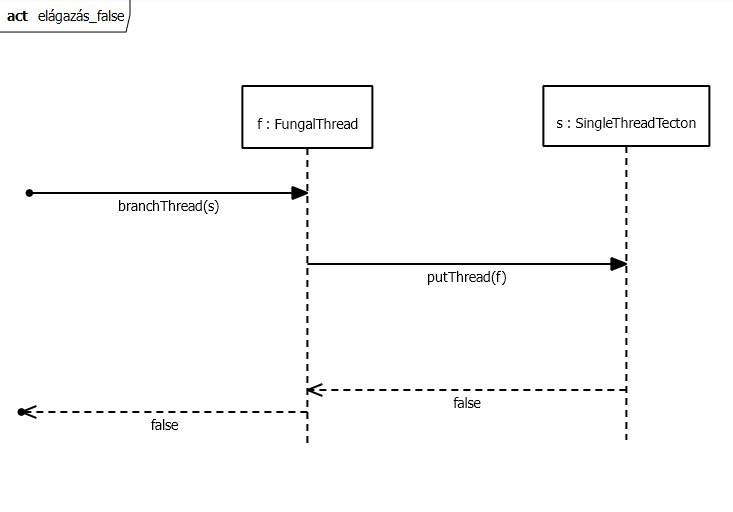
Ezen a szekvencia diagramon szemléltetem a fonál felszívódását egy AbsorbingTecton objektumon, amelyen 5 kör után felszívódnak a gombafonalak. A t tektonon két gombafonál ágazik el, az f1 és az f2. Mivel a t tektonról felszívódtak a fonalak törölni kell, hogy az f1 és az f2 fonál elágazik a t tektonon,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

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



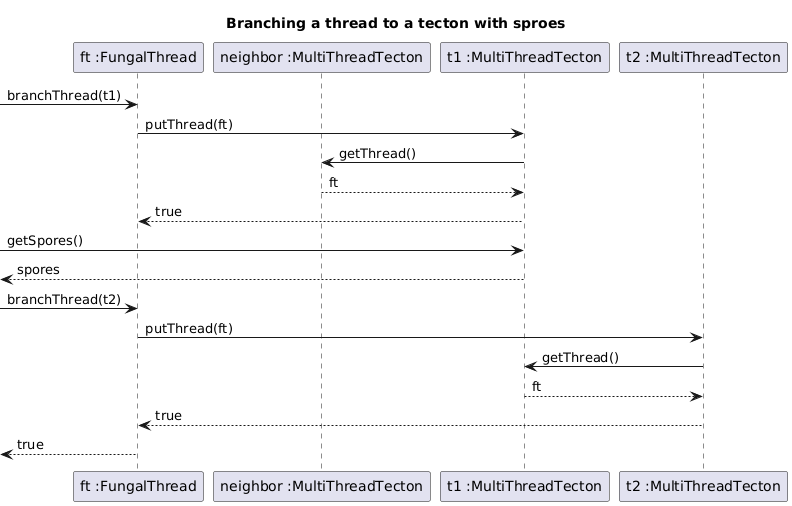
* A folyamatot a controller indítja el.
* Objektumok:
  + **f** : **FungalThread** (fonal)
  + **m** : **MultiThread** (többfonalas tekton)
    - neighbors : List<Tecton>
    - threads : List<FungalThread>
* A külső objektum meghívja az f **branchThread** függvényét m-et átadva paraméterül.
* A **branchThread**-en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

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



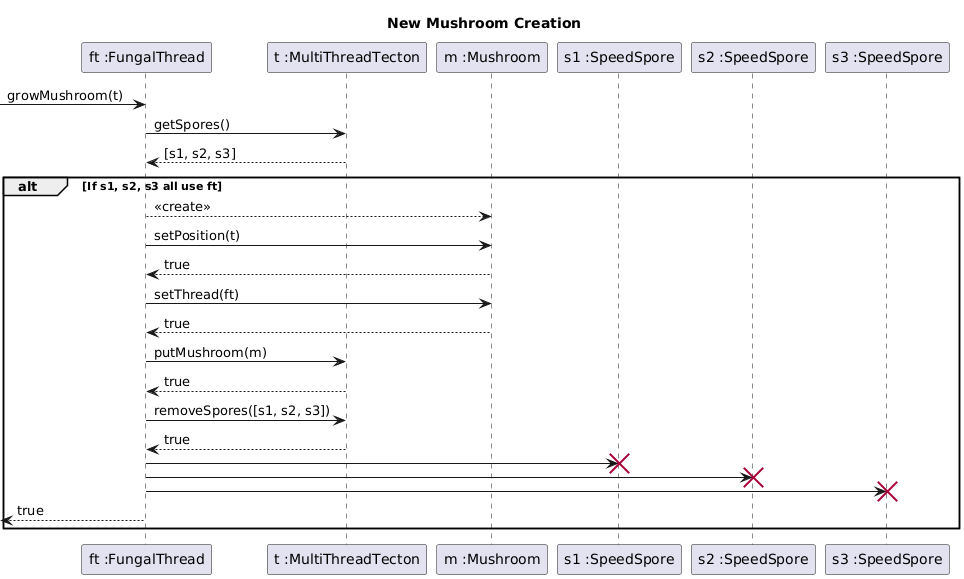
* A folyamatot a controller indítja el.
* Kettő objektum:
  + **f** : **FungalThread** (fonal)
  + **s** : **SingleThreadTecton** (egyfonalas tekton)
* A külső objektum meghívja az **f branchThread** függvényét **s**-et átadva paraméterül.
* A **branchThread**-en belül meghívódik a **putThread** függvény **s**-en, **f** átadja saját magát.
  + A **putThread**-en belül: **s** először megnézi, hogy van-e már rajta fonál. Most tfh. nincs. **-5 pont**
* Mivel van már rajta fonál és egyfonalas tektonokon csak egy fonál ágazhat el, ezért **putThread=**false->s nem tudja elraktározni f-et.
* **branchThread=**false->f nem tehető s-re.

**Gombafonal elágazása**



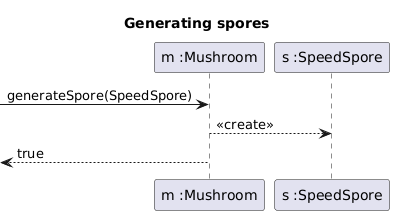
Ez a szekvencia diagram azt mutatja be, hogy egy tektonon található spóra hogyan befolyásolja a gombafonal elágazását. Ha a kiválasztott tektonon van spóra, akkor még egy további tekton kiválasztható a fonal terjesztésére. A folyamat során a rendszer ellenőrzi a szomszédos tektonokat és a spórák jelenlétét,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

**Új gombatest lerakása**

****

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

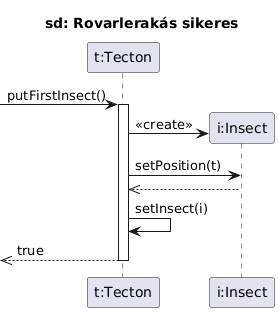
**Új spóra termelődése** le kell írni: speed spore ctor-a a fonalat megkapja

****

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

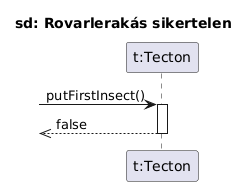
**Rovar elhelyezése játék elején:**

**-Megfelelő tektonra sikerül:**



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

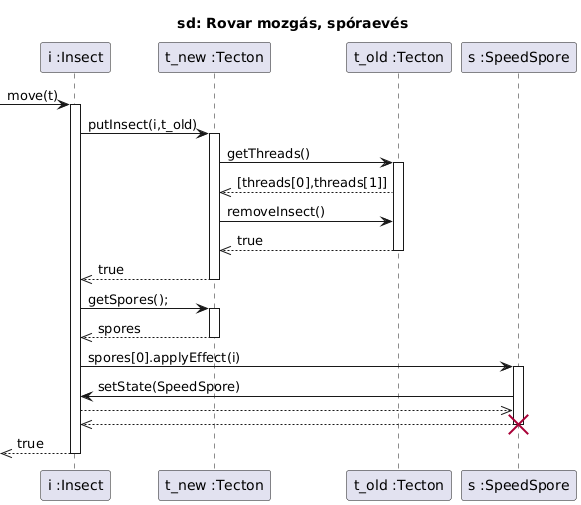
**-Nem megfelelő tektonra nem sikerül**



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

**Rovar mozgása és spóra evés: Javítva**

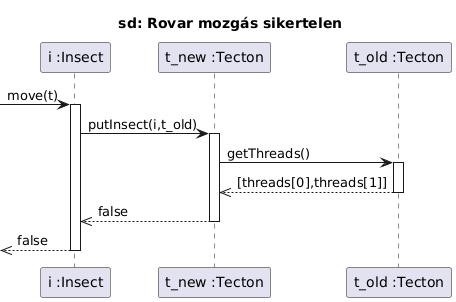
**-Sikeres mozgás és spóraevés**



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

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

**-Sikertelen mozgás**



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

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

**Gombafonál elvágása**



Egy külső kontroller meghívja az i rovar cut metódusát.

Ezen a diagramon a következő objektumok találhatók:

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

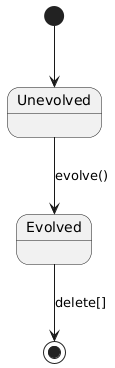
A **cut(t)** metódushívás után leellenőrizzük, hogy a paraméterül kapott t tekton szomszédos-e a position tektonnal, azaz azzal a tektonnal, amelyen a rovar rajta van.

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

.

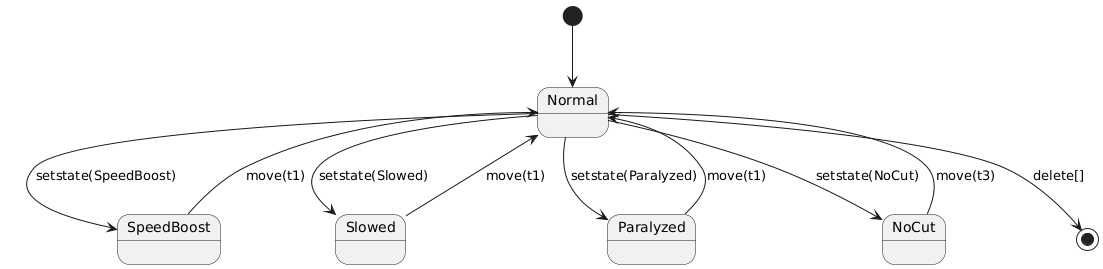
## ***State-chartok***

**Gombatest állapot diagramja**



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

**Rovar állapot diagramja** spórát, nem state-et kap



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

## ***Napló***

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