# Részletes tervek

[A dokumentum célja, hogy pontosan specifikálja az implementálandó osztályokat, beleértve a privát attribútumokat és metódusokat, ezek definícióját is.

A dokumentum második fele részletesen be kell mutassa a korábban definiált be- és kimeneti nyelv szintakszisát felhasználva, hogy mely tesztekkel lesz a prototípus ellenőrizve.]

## Osztályok és metódusok tervei.

### DefaultPlayer

#### Felelősség

Egy játékos állapotával kapcsolatos adatok kezelése.

#### Interfészek Publisher, ScoreEvaluable, Player

#### Attribútumok

* **-onTurnBeginSubscirbers: List<OnTurnBeginSubscriber>**Azoknak az objektumoknak az összegsége, amelyek értesítést szeretnének arról, hogy jelen játékos köre elkezdődött.
* -**name**: String  
  Jelen játékos neve, amellyel azonosítható.

#### Metódusok

* **+subscibe(subscriber: OnTurnBeginSubscriber)**A megadott feliratkozó értesítést kér, arról, hogy jelen játékos köre elkezdődött.
* **+setName(name: String)**A név setterje
* **+getName() : String**A név getterje
* **+notifySubscribers()**A feliratkozókat értesíti.
* ***+calculateScore() : int***A játékos pontszámát kiszámolja, majd visszatér vele.
* ***+getControllables (): List<Controllable>***A játékos irányítható játékelemeit adja vissza.

### DefaultMycologistImpl

#### Felelősség

Egy gombász játékos állapotát tárolja.

#### Interfészek Mycologist

#### Ősosztályok

Player

#### Attribútumok

* **-insects**: List<Insect>  
  A játékos által irányított rovarok.

#### Metódusok

* **+addInsect(i: Insect)**Egy rovart ad hozzá a játékoshoz, amit irányítani tud.
* **+removeInsect()**Egy játékos által irányított rovart elvesz a játékostól.

### DefaultEntomologist

#### Felelősség

Egy rovarász játékos állapotát tárolja.

#### Interfészek Entomolgist

#### Ősosztályok

Player

#### Attribútumok

* **-mycelia: List<Mycelium>**A gombász gombafonalai.
* **-mushroomBodies: List<MushroomBody>**A gombász gombatestei

#### Metódusok

* **+addMycelium(mycelium: Mycelium)**  
  Hozzáad egy gombafonalat a gombász saját gombáihoz.
* **+removeMycelim(mycelium: Mycelium)**Elvesz egy gombafonalat a gombásztól.

### DefaultPlayerContainer

#### Felelősség

A játékosok tárolásáért felelős.

#### Interfészek PlayerContainer

#### Attribútumok

* **-players: List<Player>**A játékosoknak egy listája.
* **-currentIndex: int**A mostani játékosnak az indexe.

#### Metódusok

* **+addPlayer(player: Player)**Hozzáad egy játékost a játékosok listájához.
* **+removePlayer(player: Player)**Kivesz egy játékost a játékosok listájából.
* **+getNextPlayer() : Player**A következő játékost visszaadja, és a mostani játékos a listában a következő lesz.
* **+getCurrentPlayer(): Player**Visszaadja a jelenlegi játékost.
* **+getPlayers(): Iterable<Players>**Visszaadja a listáját a játékosoknak.

### DefaultPlayerFactory

#### Felelősség

A játékosok példányosítása

#### Interfészek

PlayerFactory

#### Metódusok

* **+createPlayer(type: String, name: String): Player**Példányosít egy játékost a megadott típussal és névvel.

### PlayerController

#### Felelősség

A játékosok létrehozásának irányítása.

#### Attribútumok

* **-playerContainer: PlayerContainer**A játékosokat tároló objektum
* **-playerFactory: PlayerFactory**A játékosokat példányosító objektum

#### Metódusok

* **+createPlayer(String type, String name)**A létrehoz egy játékos példány

### RoundObserver

#### Felelősség

Értesíteni a feliratkozóit arról, hogy a játékkörök körbeértek.

#### Interfészek

Publisher

#### Attribútumok

* **-onRoundBeginSubscirbers: List<OnRoundBeginSubscriber>**Az értesítendő objektumok listája.

#### Metódusok

* **+subscribe(subscriber: OnRoundBeginSubscriber)**Az adott objektum jelentkezik, hogy szeretne értesítést arról, hogy a játékosok köre körbeért.
* **+notifySubscribers()**Értesíti a feliratkozókat.

### DefaultTurnController

#### Felelősség

A körök elkezdését meghatározza.

#### Interfészek TurnController

#### Attribútumok

* **-playerContainer**: **PlayerContainer**A játékosok listája.
* **-roundObserver: Publisher**Az objektum, ami arról értesít más objektumokat, hogy a játékosok körei véget értek.

#### Metódusok

* **+endTurn()**A mostani játékos körét befejezi.
* **+beginFirstTurn()**Elkezd egy kört, anélkül, hogy a jelenlegit befejezné.

### DefaultScoreCalculator

#### Felelősség

Eldönteni, hogy melyik játékos nyerte meg a játékot.

#### Interfészek

ScoreCalculator

#### Metódusok

* **+determineWinner(canidates: Iterable<ScoreEvaluable>) : ScoreEvaluable**Visszaadja a játékost, akinek a pontjai a legmagasabbak.

### DefualtGameEndManager

#### Felelősség

A játék végét vezérlő osztály.

#### Interfészek OnRoundBeginSubscriber, GameEndManager

#### Attribútumok

* **-scoreCalculator: ScoreCalculator**
* **-gameLength: int**

#### Metódusok

* **+onRoundBegin()**
* **+setGameLength(newLength: int)**
* **+getGameLength(): int**
* **-showWinner()**

### DefaultGameController

#### Felelősség A játék menetével elindításért

#### Interfészek GameController

#### Attribútumok

* **-turnController: TurnController**
* -**gameEndManager: GameEndManager**

#### Metódusok

* +**beginGame(length: int)**
* **+endGame()**

### InsectController

#### Felelősség

A rovarokat irányítása.

#### Attribútumok

* **-turnController: TurnController**
* **-controlledInsect: Insect**
* **-selectedTecton: Tecton**

#### Metódusok

* **+cut()**A kiválasztott rovarral elvágja a tektont.
* **+eat()**A kiválasztott rovarral eszik.
* **+move()**

A kiválasztott rovart mozgatja.

* **+selectInsect()**

A rovart

* **+selectTecton()**

### DefaultMyceliumFactory

#### Felelősség

Gombafonalak példányosítása, úgy, hogy a példányosítás a növesztés feltételei szerint növekedjen.

#### Interfészek

MushroomFactory

#### Metódusok

* +**createMycelium(type: String, name: String, location: Tecton): Mycelium**

### CheatMyceliumFactory

#### Felelősség

Gombafonalak példányosítása, úgy, hogy a növesztés feltételeit nem veszik figyelembe.

#### Interfészek

MushroomFactory

#### Metódusok

* +**createMycelium (type: String, name: String, location: Tecton): Mycelium**

### DefaultMushroomBodyFactory

#### Felelősség

Gombatestek példányosítása, úgy, hogy a példányosítás a növesztés feltételei szerint növekedjen.

#### Interfészek

MushroomBodyFactory

#### Metódusok

* +**createMushroomBody(name: String, location: Tecton) : MushroomBody**

### CheatMushroomBodyFactory

#### Felelősség

Gombatestek példányosítása, úgy, hogy a növesztés feltételeit nem veszik figyelembe.

#### Interfészek

MushroomBodyFactory

#### Metódusok

* +**createMushroomBody(name: String, location: Tecton) : MushroomBody**

### DefaultCommandReader

#### Felelősség

#### Interfészek

CommandReader

#### Metódusok

* **+getNextCommand()**

### MushroombodyController

#### Felelősség

Gombatestek irányítása.

#### Attribútumok

* **-turnController: TurnController**
* **-selectedMushroomBody: MushroomBody**
* **-targetTecton: Tecton**

#### Metódusok

* **+eject()**
* **+selectTecton()**
* **+selectMushroomBody()**

### DefaultTectonFactory

#### Felelősség

Tektonok példányosítása

#### Interfészek

TectonFactory

#### Metódusok

* **+create(type: String, name: String, neighbours: Iterable<Tecton>): Tecton**

### DefaultInsectFactory

#### Felelősség

Rovarok példányosítása

#### Interfészek

InsectFactory

#### Metódusok

* **+create(name: String) : Insect**

### DefaultInsectSpore

#### Felelősség

Rovarok példányosítása

#### Interfészek

SporeFactory

#### Metódusok

* **+create(name: String, type: String) : Spore**

### MapCreationController

#### Felelősség

A játéktér létrehozásának irányítása.

#### Attribútumok

* **-selectedMushroomBodyType: String**
* **-selectedMyceliumType: String**
* -**selectedSporeType: String**
* -**selectedName: String**
* **-mushroomBodyFactory: MushroomBodyFactory**
* **-myceliumFactory: MushroomBodyFactory**
* **-insectFactory: MushroomBodyFactory**
* **-sporeFactory: MushroomBodyFactory**

#### Metódusok

* **+selectMushroomBodyType()**
* **+selectMyceliumType()**
* **+selectSporeType()**
* **+selectName()**
* **+createMycelium()**
* **+createMushroomBody()**
* +**createInscet()**
* **+createSpore()**

### GrowthController

#### Felelősség

[Mi az osztály felelőssége. Kb 1 bekezdés. Ha szükséges, akkor state-chart is.]

#### Attribútumok

* -**selectedName: String**
* **-mushroomBodyFactory: MushroomBodyFactory**
* **-myceliumFactory: MushroomBodyFactory**

#### Metódusok

* **+selectName()**
* **+growMycelium()**
* **+createMushroomBody()**

### Osztály2

#### Felelősség

[Mi az osztály felelőssége. Kb 1 bekezdés. Ha szükséges, akkor state-chart is.]

#### Ősosztályok

[Mely osztályokból származik (öröklési hierarchia)]

Legősebb osztály → Ősosztály2 → Ősosztály3...

#### Interfészek

[Mely interfészeket valósítja meg.]

#### Attribútumok

[Milyen attribútumai vannak, beleértve a nem publikusakat is.]

* **attribútum1**: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa
* **attribútum2**: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa

#### Metódusok

[Milyen publikus, protected és privát metódusokkal rendelkezik. Metódusonként precíz leírás, ha szükséges, activity diagram is a metódusban megvalósítandó algoritmusról. Minden olyan metódusnak szerepelnie kell, amelyiket az osztály megvalósít vagy felüldefiniál.]

* **int foo(Osztály3 o1, Osztály4 o2)**: metódus leírása, láthatósága (UML jelöléssel)
* **int bar(Osztály5 o1)**: metódus leírása, láthatósága (UML jelöléssel)

## A tesztek részletes tervei, leírásuk a teszt nyelvén

[A tesztek részletes tervei alatt meg kell adni azokat a bemeneti adatsorozatokat, amelyekkel a program működése ellenőrizhető. Minden bemenő adatsorozathoz definiálni kell, hogy az adatsorozat végrehajtásától a program mely részeinek, funkcióinak ellenőrzését várjuk és konkrétan milyen eredményekre számítunk, ezek az eredmények hogyan vethetők össze a bemenetekkel. A tesztek leírásakor az előző dokumentumban (proto koncepciója) megadott szintakszist kell használni.]

## A tesztek részletes tervei, leírásuk a teszt nyelvén

[A tesztek részletes tervei alatt meg kell adni azokat a bemeneti adatsorozatokat, amelyekkel a program működése ellenőrizhető. Minden bemenő adatsorozathoz definiálni kell, hogy az adatsorozat végrehajtásától a program mely részeinek, funkcióinak ellenőrzését várjuk és konkrétan milyen eredményekre számítunk, ezek az eredmények hogyan vethetők össze a bemenetekkel. A tesztek leírásakor az előző dokumentumban (proto koncepciója) megadott szintakszist kell használni.]

### Teszteset1

* **Leírás**

[szöveges leírás, kb. 1-5 mondat.]

* **Ellenőrzött funkcionalitás, várható hibahelyek**
* **Bemenet**

[a proto bemeneti nyelvén megadva (lásd előző anyag)]

* **Elvárt kimenet**

[a proto kimeneti nyelvén megadva (lásd előző anyag)]

### Teszteset2

* **Leírás**

[szöveges leírás, kb. 1-5 mondat.]

* **Ellenőrzött funkcionalitás, várható hibahelyek**
* **Bemenet**

[a proto bemeneti nyelvén megadva (lásd előző anyag)]

* **Elvárt kimenet**

[a proto kimeneti nyelvén megadva (lásd előző anyag)]

## A tesztelést támogató programok tervei

[A tesztadatok előállítására, a tesztek eredményeinek kiértékelésére szolgáló segédprogramok részletes terveit kell elkészíteni.]

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2010.03.21. 18:00 | 2,5 óra | Horváth  Németh  Tóth  Oláh | Értekezlet.  Döntés: Horváth elkészíti az osztálydiagramot, Oláh a use-case leírásokat. |
| 2010.03.23. 23:00 | 5 óra | Németh | Tevékenység: Németh implementálja a tesztelő programokat. |
| … | … | … | … |