# Analízis modell kidolgozása

## Objektum katalógus

[Minden, a feladatban szereplő objektum rövid, egy-két bekezdés hosszú ismertetése. Meg kell jelenjen minden objektumhoz, hogy mi a felelőssége. Informális leírás, ezért nem szabad még foglalkozni az örökléssel, az interfészekkel, az absztrakt osztályokkal, a segédosztályokkal.]

### Objektum1

[Felelősség informális leírása]

### Objektum2

[Felelősség informális leírása]

[Az objektumkatalógus alapján kiindulva kell megalkotni az objektumorientált analízis modellt. A 3.2, 3.3, 3.4, 3.5 alfejezetek ugyanannak a modellnek a különböző nézetei, ezért egyidőben, egymással összefüggésben készülnek. Megtörténik az objektumkatalógusban tárgyalt objektumok felelősségének formalizálása osztályokká, asszociációkká, attribútumokká, metódusokká. Csak publikus metódusok szerepelhetnek. Megjelennek az interfészek, az öröklés, az absztrakt osztályok. Segédosztályokra még nincs szükség.]

## Statikus struktúra diagramok

[Az objektum katalógusban szereplő objektumokat megvalósító osztályok kapcsolatait és publikus metódusait bemutató osztálydiagram(ok). Tipikus hibalehetőségek: csillag-topológia, szigetek.]

## Osztályok leírása

[A diagramon szereplő osztályok bemutatása. Az osztályok ABC sorrendben kövessék egymást. Interfészek esetén az Interfészek, Attribútumok pontok kimaradnak.]

\*\*\*

**EZ EGY PLUSZ OLDAL LESZ A LEADANDÓ ANYAG ELEJÉRE A VÁLTOZÁSOKRÓL**

**Végrehajtott koncepcionális változások a *2. Követelmény, projekt, funkcionalitás* elnevezésű dokumentumban meghatározottakhoz képest**

**1.** A *2.2.1 Általános áttekintés A. Architektúra és alrendszerek* pontjához:

* a felhasználói alrendszer és az adattárolási alrendszer törlésre került a koncepcióból.

**2.** A *2.2.1 Általános áttekintés B. Az alrendszerek közötti interfészek* pontja a következőképpen került módosításra:

„Az alrendszerek között a következő interfészek kerülnek kialakításra:

1. a játékosok a döntéseket a megjelenítési alrendszeren keresztül hozzák meg;
2. a megjelenítési alrendszer a játékosok döntéseit továbbítja a logikai alrendszer felé, amely ez alapján frissíti a játékállapotokat.”

**3.**A *2.4.2 Use-case diagram* pontnál a diagramból az *Operációs rendszer* és a *Játék logika* mint aktorok törlésre kerültek*.*

**4.**A játék koncepciója kapcsán megfogalmazásra került, hogy minden gombász egy véletlenszerűen kiválasztott tektonra elhelyezett gombatesttel kezdi a játékot, és – amennyiben a tektonokra vonatkozó szabályok ezt lehetővé teszik – a gombatestekből minden szomszédos tekton felé 1 hosszúságú gombafonál vezet.

\*\*\*

### *AridTecton* (osztály)

#### Felelősség

Az *AridTecton* olyan *FertileTecton*, amelyen legfeljebb 1 gombafonál és legfeljebb 1 gombatest növekedhet. Rajta a gombafonalak 5 kör után automatikusan felszívódnak.

#### Ősosztályok

*Tecton* → *FertileTecton* → *AridTecton*

#### Interfészek

*n/a*

#### Aggregációk

*n/a*

#### Asszociációk

*n/a*

#### Kompozíciók

*n/a*

#### Függőségek

*n/a*

#### Attribútumok

*n/a*

#### Metódusok

* ***canGrowMycelium()*:** ellenőrzi, hogy a tektonon nőhet-e gombafonál.
* ***canGrowMushroomBody()*:** ellenőrzi, hogy a tektonon nőhet-e gombatest.
* ***onTurnBegin(e: Player)*:** a *TurnBeginSubscriber* interfésztől értesítést kap arra vonatkozóan, hogy új játékos következik.
* ***growMycelium()*:** gombafonalat növeszt.

### *CoarseTecton* (osztály)

#### Felelősség

A *CoarseTecton* olyan *Tecton*, amelyen nem növekedhet se gombatest, se gombafonál.

#### Ősosztályok

*Tecton* → *CoarseTecton*

#### Interfészek

*n/a*

#### Aggregációk

*n/a*

#### Asszociációk

*n/a*

#### Kompozíciók

*n/a*

#### Attribútumok

*n/a*

#### Függőségek

*n/a*

#### Metódusok

* ***canGrowMycelium()*:** ellenőrzi, hogy a tektonon nőhet-e gombafonál.
* ***canGrowMushroomBody()*:** ellenőrzi, hogy a tektonon nőhet-e gombatest.

### *Entomologist* (osztály)

#### Felelősség

A rovarok (*Insect*) irányításáért felelős játékosok osztálya.

#### Ősosztályok

*Player* → *Entomologist*

#### Interfészek

*n/a*

#### Aggregációk

*n/a*

#### Asszociációk

* Az *Entomologist* asszociatív kapcsolatban áll az *Insect*tel, mivel a rovarász irányítja a saját rovarát, és minden rovarásznak 1 rovarja lehet. Ennek megfelelően a kapcsolat 1:1-es számosságú. A kapcsolat mindkét oldalon navigálható.

#### Kompozíciók

*n/a*

#### Függőségek

*n/a*

#### Attribútumok

*n/a*

#### Metódusok

* ***onTurnBegin(e: Player)*:** a *TurnBeginSubscriber* interfésztől értesítést kap arra vonatkozóan, hogy új játékos következik.
* ***calculateScore*():** számolja, hogy hány pontja van.

### *FertileTecton* (osztály)

#### Felelősség

A *FertileTecton* olyan *Tecton*, amelyen a feltételek fennállása esetén lehet gombatest vagy gombafonál:

* gombatest akkor növeszthető a tektonon, ha gombafonál és legalább 3 spóra található rajta;
* gombafonál gombatestből vagy gombafonálból nőhet ki és szomszédos tektonra növeszthető. Ha egy tektonon található gombatest, rá nem lehet gombafonalat tenni.

#### Ősosztályok

*Tecton* → *FertileTecton*

(A *FertileTecton* leszármazottja az *AridTecton* és a *MultiLayeredTecton*.)

#### Interfészek

*n/a*

#### Aggregációk

*n/a*

#### Asszociációk

*n/a*

#### Kompozíciók

*n/a*

#### Függőségek

*n/a*

#### Attribútumok

*n/a*

#### Metódusok

* ***canGrowMycelium()*:** ellenőrzi, hogy a tektonon nőhet-e gombafonál.
* ***canGrowMushroomBody()*:** ellenőrzi, hogy a tektonon nőhet-e gombatest.

### *GameBeginSubscriber* (interfész)

#### Felelősség

A *GameManager subscriber*eként tájékoztatást kap az új játék kezdetéről, és ezt az információt az interfészt megvalósító osztályok részére továbbítja.

(Az interfészt a következő osztályok valósítják meg: *GameBoard*, *TurnManager*.)

#### Ősosztályok

*n/a*

#### Interfészek

*n/a*

#### Aggregációk

* A *GameBeginSubscriber* aggregációs kapcsolatban áll a *GameManager*rel, amelytől *subscriber*ként tájékoztatást kap az új játék kezdetéről. A *GameManager* az aggregátor.

#### Asszociációk

*n/a*

#### Kompozíciók

*n/a*

#### Függőségek

*n/a*

#### Attribútumok

*n/a*

#### Metódusok

* ***onGameBegin()*:**tájékoztatást nyújt a játék kezdetéről.

### *GameBoard* (osztály)

#### Felelősség

A *GameBoard* biztosítja a játékteret azzal, hogy kezeli a tektonokkal kapcsolatos legfontosabb eseményt, a tektontörést, továbbá megvizsgálja, hogy egy gombafonál elveszítette-e a folytonosságát.

#### Ősosztályok

*n/a*

#### Interfészek

* A *GameBoard* megvalósítja a *GameBeginSubscriber* interfészt, amelynek révén értesítést kap a játék kezdetéről. Erre a tektontörés folyamata miatt van szüksége.

#### Aggregációk

*n/a*

#### Asszociációk

*n/a*

#### Kompozíciók

* A *GameBoard* kompozíciós kapcsolatban áll a *Tecton*nal, mivel a tekton a játékmező alapeleme. A *GameBoard* kezeli tektontörés esetén a tektonokban bekövetkezett változásokat. Ennek megfelelően egy rész-egész viszony alapulvételével a *Gameboard* képviseli az egészt.

#### Függőségek

*n/a*

#### Attribútumok

*n/a*

#### Metódusok

* ***addTecton()*:**tektontörés esetén új tektont ad a játékmezőhöz.
* ***onGameBegin()*:**tájékoztatást nyújt a játék kezdetéről.
* ***checkMyceliumConnectivity()*:**megvizsgálja, hogy egy gombafonál elveszítette-e a folytonosságát.

### *GameManager* (osztály)

#### Felelősség

A *GameManager* indítja el a játékot és határozza meg a végén a győztes személyét.

#### Ősosztályok

*n/a*

#### Interfészek

* Az *GameManager* megvalósítja a *TurnBeginSubscriber* interfészt. Az interfész révén kap tájékoztatást arról, ha új játékos következik.

#### Aggregációk

* A *GameManager* aggregációs kapcsolatban áll a *GameBeginSubscriber*rel, amelyet *publisher*ként tájékoztat az új játék kezdetéről. A *GameManager* az aggregátor.

#### Asszociációk

*n/a*

#### Kompozíciók

* A *GameManager* kompozíciós kapcsolatban áll a *Player*rel, mivel nyilvántartja a játékosok listáját. Ennek megfelelően egy rész-egész viszony alapulvételével a *GameManager* képviseli az egészt.

#### Függőségek

*n/a*

#### Attribútumok

*n/a*

#### Metódusok

* ***startNewGame()*:**új játékot indít.
* ***determineWinner()*:** meghatározza a győztes személyét.
* ***addPlayer(p: Player)*:** új versenyzőt ad a játékhoz.
* ***subcribe(s: GameBeginSubscriber)***: a segítségével a *publisher* szolgáltatásaira lehet feliratkozni.
* ***unsubscribe(s: GameBeginSubscriber):*** a segítségével a *publisher* szolgáltatásairól lehet leiratkozni.

### *Insect* (osztály)

#### Felelősség

A rovarász (*Entomologist*) irányítása alá tartozó objektum. Képes a gombafonalak mentén mozogni, gombafonalat elvágni és spórát fogyasztani.

#### Ősosztályok

*n/a*

#### Interfészek

* Az *Insect* megvalósítja a *TurnBeginSubscriber* interfészt. Az interfész révén kap tájékoztatást arról, ha új játékos következik.

#### Aggregációk

*n/a*

#### Asszociációk

* Az *Insect* asszociatív kapcsolatban áll az *Entomologist*tal, mivel a rovarász irányítja a saját rovarát, és minden rovarásznak 1 rovarja lehet. Ennek megfelelően a kapcsolat 1:1-es számosságú. A kapcsolat mindkét oldalon navigálható.

#### Kompozíciók

*n/a*

#### Függőségek

* Az *Insect* asszociatív kapcsolatban áll az *Entomologist*tel, mivel a rovart az azért felelős rovarász irányítja.

#### Attribútumok

* ***location:Tecton*** → nyilvántartja, melyik tektonon tartózkodik a rovar.
* ***remainingMoves:int*** → számolja, hogy hány lépése van még az adott körben.
* ***sporeCount: int*** → számolja a megevett spórákat.
* ***owner: Entomologist*** → nyilvántartja a rovarászát.

#### Metódusok

* ***cutMycelium()*:** elvág egy gombafonalat.
* ***eatSpore()*:** elfogyaszt egy spórát.
* ***move()*:** gombafonál mentén halad.
* ***onTurnBegin()*:** a *TurnBeginSubscriber* interfésztől értesítést kap arra vonatkozóan, hogy új játékos következik. Ennek segítségével tud a rovar egy véletlenszerű fonálra „elmenekülni”, ha eltűnik alatta a fonál.

### *InsectEffect* (enum)

#### Felelősség

Az *InsectEffect* révén kerül meghatározásra, hogy a spóra elfogyasztása milyen élettani hatást gyakorol a rovarra:

* *slowness*: lelassítja a rovart, azaz egy körben csak 1 lépést tehet meg. Ez a hatás 3 körön át tart.
* *speed*: felgyorsítja a rovart, azaz egy körben 3 lépést tehet meg. Ez a hatás 3 körön át tart.
* *preventCut*: megakadályozza a rovart abban, hogy fonalat vágjon el. Ez a hatás 3 körön át tart.
* *stun*: a rovar 1 körön át nem tud semmilyen aktivitást kifejteni.
* *nothing*: semmilyen hatással nem jár.

#### Ősosztályok

*n/a*

#### Interfészek

*n/a*

#### Aggregációk

*n/a*

#### Asszociációk

*n/a*

#### Kompozíciók

*n/a*

#### Függőségek

* Az *InsectEffect* függőségi kapcsolatban áll a *Spore*-ral, mivel meghatározza, milyen következménnyel jár a rovar számára a spóra elfogyasztása.
* Az *InsectEffect* függőségi kapcsolatban áll az *Insect*tel, mivel meghatározza, milyen következménnyel jár a rovar számára a spóra elfogyasztása.

#### Attribútumok

*n/a*

#### Metódusok

*n/a*

### *Mushroom* (osztály)

#### Felelősség

A gombász felelősségi körébe tartozó objektum, amelyből gombafonál nőhet ki.

#### Ősosztályok

(A *Mushroom* leszármazottja a *Mycelium* és a *MushroomBody*.)

#### Interfészek

*n/a*

#### Aggregációk

*n/a*

#### Asszociációk

* A *Mushroom* asszociatív kapcsolatban áll a *Mycologist*tel, mivel a gombász felelős a saját gombáiért és gombafonalaiért. Ez meghatározza a kapcsolat számosságát (*Mycologist* – *Mushroom*: 1 – 0..\*). A kapcsolat mindkét oldalon navigálható.

#### Kompozíciók

*n/a*

#### Függőségek

*n/a*

#### Attribútumok

* ***location:Tecton*** → nyilvántartja, melyik tektonon található a gomba.
* **owner: Mycologist** → nyilvántartja a gombászát.

#### Metódusok

* ***growMycelium()*:** gombafonalat növeszt.
* ***sustains()***: megállapítja, hogy gombafonálról vagy gombatestről van-e szó.

### *MushroomBody* (osztály)

#### Felelősség

A spórák termelésért és kilövésért felelős gombatest.

#### Ősosztályok

*Mushroom* → *MushroomBody*

#### Interfészek

* Az *MushroomBody* megvalósítja a *TurnBeginSubscriber* interfészt. Az interfész révén kap tájékoztatást arról, ha új játékos következik. Erre az információra a spóratermelés miatt van szüksége.

#### Aggregációk

* A *MushroomBody* aggregációs kapcsolatban áll a *Spore*-ral, mivel az előbbi termeli és szétszórja a spórákat. Ennek megfelelően a *MushroomBody* az aggregátor.

#### Asszociációk

* A *MushroomBody* asszociatív kapcsolatban áll a *Tecton*nal, mivel egy tektonon legfeljebb egy gombatest nőhet. Ez meghatározza a kapcsolat számosságát (*Tecton* – *MushroomBody*: 0..1 – 1). A kapcsolat mindkét oldalon navigálható.

#### Kompozíciók

*n/a*

#### Függőségek

*n/a*

#### Attribútumok

*n/a*

#### Metódusok

* ***ejectSpores(target: Tecton)*:** szétszórja a spórákat.
* ***sustains()***: megállapítja, hogy gombafonálról vagy gombatestről van-e szó.
* ***onTurnBegin(e: Player)*:** a *TurnBeginSubscriber* interfésztől értesítést kap arra vonatkozóan, hogy új játékos következik. Erre az információra a spóratermelés miatt van szüksége.

### *Mycelium* (osztály)

#### Felelősség

Gombatest növesztésére képes gombarész – a gombafonál. Tektontörés esetén elszakad és elsorvad.

#### Ősosztályok

*Mushroom* → *Mycelium*

#### Interfészek

*n/a*

#### Aggregációk

*n/a*

#### Asszociációk

*n/a*

#### Kompozíciók

* A *Mycelium* kompozíciós kapcsolatban áll a *Tecton*nal, mivel egy tekton létezése meghatározza a gombafonál létezését is. Ennek megfelelően egy rész-egész viszony alapulvételével a *Tecton* képviseli az egészt.

#### Függőségek

*n/a*

#### Attribútumok

*n/a*

#### Metódusok

* ***cut()*:** a gombafonál elszakad.
* ***delete()*:** a gombatesttel közvetlen kapcsolatban nem lévő gombafonál elsorvad.
* ***sustains()***: megállapítja, hogy gombafonálról vagy gombatestről van-e szó.

### *Mycologist* (osztály)

#### Felelősség

A gombákért (*Mushroom*) felelős játékosok osztálya. Tudomással bír arról, ha ő következik és számolja a pontszámát. Képes a játékmezőhöz egy új saját gombatestet hozzáadni.

#### Ősosztályok

*Player* → *Mycologist*

#### Interfészek

*n/a*

#### Aggregációk

*n/a*

#### Asszociációk

* Az *Mycologist* asszociatív kapcsolatban áll a *Mushroom*mal, mivel a gombász felelős a gombáiért és a gombafonalaiért. Ez meghatározza a kapcsolat számosságát (*Mycologist* – *Mushroom*: 1 – 0…\*). A kapcsolat mindkét oldalon navigálható.

#### Kompozíciók

*n/a*

#### Függőségek

*n/a*

#### Attribútumok

* ***grows: int*:** tárolja, hogy a gombász mennyi gombafonalat tud növeszteni az adott körben.

#### Metódusok

* ***addMushroomBody*():** hozzáad egy új gombatestet ad a játékmezőhöz.
* ***calculateScore*():** számolja, hogy hány pontja van.
* ***onTurnBegin(e: Player)*:** a *TurnBeginSubscriber* interfésztől értesítést kap arra vonatkozóan, hogy új játékos következik.

### *MultiLayeredTecton* (osztály)

#### Felelősség

A *MultiLayeredTecton* olyan *FertileTecton*, amelyen legfeljebb 1 gombatest és legfeljebb 3 gombafonál növekedhet.

#### Ősosztályok

*Tecton* → *FertileTecton* → *MultiLayeredTecton*

#### Interfészek

*n/a*

#### Aggregációk

*n/a*

#### Asszociációk

*n/a*

#### Kompozíciók

*n/a*

#### Függőségek

*n/a*

#### Attribútumok

*n/a*

#### Metódusok

* ***canGrowMycelium()*:** ellenőrzi, hogy a tektonon nőhet-e gombafonál.
* ***canGrowMushroomBody()*:** ellenőrzi, hogy a tektonon nőhet-e gombatest.

### *Player* (osztály)

#### Felelősség

A játék részvevőinek (*Entomologist*, *Mycologist*) ősosztálya. Tudomással bír arról, hogy melyik játékos következik és számolja a játékosok pontszámát.

#### Ősosztályok

*n/a*

#### Interfészek

* A *Player* megvalósítja a *TurnBeginSubscriber* interfészt. Az interfész révén kap tájékoztatást arról, ha új játékos következik.

#### Aggregációk

*n/a*

#### Asszociációk

*n/a*

#### Kompozíciók

* A *Player* kompozíciós kapcsolatban áll a *GameManager*rel, amely nyilvántartja a játékosok listáját. Ennek megfelelően egy rész-egész viszony alapulvételével a *GameManager* képviseli az egészt.

#### Függőségek

*n/a*

#### Attribútumok

* ***name: String*:** nyilvántartja a játékos nevét.
* ***color: String*:** nyilvántartja a játékoshoz tartozó színt.

#### Metódusok

* ***calculateScore():*** számolja, hogy hány pontja van az adott játékosnak.
* ***onTurnBegin(e: Player)*:** a *TurnBeginSubscriber* interfésztől értesítést kap arra vonatkozóan, hogy új játékos következik.

### *Spore* (osztály)

#### Felelősség

A rovarok spórákkal táplálkoznak, és az elfogyasztásuk élettani hatást gyakorolhat a rovarra.

#### Ősosztályok

*n/a*

#### Interfészek

*n/a*

#### Aggregációk

* A *Spore* aggregációs kapcsolatban áll a *MushroomBody*-val, amely termeli és szétszórja a spórákat. Ennek megfelelően a *MushroomBody* az aggregátor.
* A *Spore* aggregációs kapcsolatban áll a *Tecton*nal, mivel a tektonra spóraszórás esetén spórák eshetnek. Az egy gombatestből kilövellt összes spóra ugyanarra a tektonra esik, és a *Tecton* az aggregátor.

#### Asszociációk

*n/a*

#### Kompozíciók

*n/a*

#### Függőségek

* A *Spore* függőségi kapcsolatban áll az *InsectEffect*tel, amely meghatározza, hogy milyen következménnyel jár a rovar számára a spóra elfogyasztása.

#### Attribútumok

*n/a*

#### Metódusok

*n/a*

(A *getEffect()* annak hangsúlyozása céljából szerepel az osztálydiagramon, hogy ennél az osztálynál csak *getter* van, *setter* nincs. Ezért a metódust itt nem tüntettük fel.)

### *Tecton* (osztály)

#### Felelősség

A tektöntörés keretében felelős az új tektonok létrehozásáért. Ennek érdekében nyilvántartja, hogy hány kör múlva következik be tektontörés. Az ehhez szükséges körök száma véletlenszerűen kerül meghatározásra.

#### Ősosztályok

(A *Tecton* leszármazottja a *CoarseTecton* és a *FertileTecton*.)

#### Interfészek

* A *Tecton* megvalósítja a *TurnBeginSubscriber* interfészt. Az interfész révén kap tájékoztatást arról, ha új játékos következik, amelynek segítségével a köröket számolni tudja.

#### Aggregációk

* A *Tecton* aggregációs kapcsolatban áll a *Spore*-ral, mivel a tektonra spóraszórás esetén spórák eshetnek. Az egy gombatestből kilövellt összes spóra ugyanarra a tektonra esik, és a *Tecton* az aggregátor.

#### Asszociációk

* A *Tecton* asszociatív kapcsolatban áll a *MushroomBody*val, mivel egy tektonon legfeljebb egy gombatest nőhet. Ez meghatározza a kapcsolat számosságát (*Tecton* – *MushroomBody*: 0..1 – 1). A kapcsolat mindkét oldalon navigálható.

#### Kompozíciók

* A *Tecton* kompozíciós kapcsolatban áll a *GameBoard*dal, mivel a tekton a játékmező alapeleme. A *GameBoard* kezeli tektontörés esetén a tektonokban bekövetkezett változásokat. Ennek megfelelően egy rész-egész viszony alapulvételével a *Gameboard* képviseli az egészt.
* A *Tecton* kompozíciós kapcsolatban áll a *Mycelium*mal, mivel egy tekton létezése meghatározza a gombafonál létezését is. Ennek megfelelően egy rész-egész viszony alapulvételével a *Tecton* képviseli az egészt.

#### Attribútumok

* ***mushroomBody***: nyilvántartja a tektonon található gombatesteket.
* ***mycelia*:** nyilvántartja a tektonon található gombafonalakat.

#### Metódusok

* ***canGrownMycelium()*:** megállapítja, hogy a tektonon növeszthető-e gombafonál.
* ***canGrowMushroomBody()*:** megállapítja, hogy a tektonon növeszthető-e gombatest.
* ***growMushroomBody()*:** a tektonon gombafonalat növeszt.
* ***growMycelium()*:** a tektonon gombatestet növeszt.
* ***onTurnBegin(e: Player)*:** a *TurnBeginSubscriber* interfésztől értesítést kap arra vonatkozóan, hogy új játékos következik.

### *TurnBeginSubscriber* (interfész)

#### Felelősség

A *TurnManager subscriber*eként tájékoztatást kap, ha egy új játékos következik, és ezt az információt az interfészt megvalósító osztályok részére továbbítja.

(Az interfészt a következő osztályok valósítják meg: *GameManager*, *Tecton*, *Player*, *Insect*, *Mycelium*.)

#### Ősosztályok

*n/a*

#### Interfészek

*n/a*

#### Aggregációk

* A *TurnBeginSubscriber* aggregációs kapcsolatban áll a *TurnManager*rel, amelytől *subscriber*ként tájékoztatást kap arról, ha egy új játékos következik. A *TurnManager* az aggregátor.

#### Asszociációk

*n/a*

#### Kompozíciók

*n/a*

#### Függőségek

*n/a*

#### Attribútumok

*n/a*

#### Metódusok

* ***onTurnBegin(e: Player)*:** tájékoztatást nyújt arra vonatkozóan, ha új játékos következik.

### *TurnManager* (osztály)

#### Felelősség

Ez az osztály irányítja, hogy mikor következik egy új játékos. Mint *publisher* tájékoztatja erről a feliratkozó *TurnBeginSubscriber* interfészt.

#### Ősosztályok

*n/a*

#### Interfészek

* A *TurnManager* megvalósítja a *GameBeginSubscriber* interfészt. Az interfész révén tájékoztatást kap a játék kezdetéről.

#### Aggregációk

* A *TurnManager* aggregációs kapcsolatban áll a *TurnBeginSubscriber*rel, amelyet *publisher*ként tájékoztat a kör végéről. A *TurnManager* az aggregátor.

#### Asszociációk

*n/a*

#### Kompozíciók

*n/a*

#### Függőségek

*n/a*

#### Attribútumok

*n/a*

#### Metódusok

* ***subcribe(s: TurnBeginSubscriber)***: a segítségével a *publisher* szolgáltatásaira lehet feliratkozni.
* ***unsubscribe(s: TurnBeginSubscriber):*** a segítségével a *publisher* szolgáltatásairól lehet leiratkozni.
* ***endTurn():*** egy másik játékos körének kezdetekor meghívja a *notifySubscribers()*-t.
* ***notifySubscribers()*:** tájékoztatja a feliratkozó(ka)t *(subscribers)* a kör végéről.
* ***onGameBegin()*:**tájékoztatást nyújt a játék kezdetéről.

## Szekvencia diagramok

[Inicializálásra, use-case-ekre, belső működésre. Konzisztens kell legyen az előző alfejezettel. Minden metódus, ami ott szerepel, fel kell tűnjön valamelyik szekvenciában. Minden metódusnak, ami szekvenciában szerepel, szereplnie kell a valamelyik osztálydiagramon. Fontos, hogy az aktor use-case-eitől bármelyik diagramhoz el lehessen jutni a metódushívások követésével.]

## State-chartok

[Csak azokhoz az osztályokhoz, ahol van értelme. Egyetlen állapotból álló state-chartok ne szerepeljenek. A játék működését bemutató state-chart-ot készíteni tilos.]

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
|  |  |  |  |