# 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.

### PlayerImpl

#### Felelősség

Egy játékos állapotával kapcsolatos adatok kezelése.

#### Interfészek ScoreEvaluable, Player, TurnObserver

#### 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

* **+PlayerImpl(name: String)**Konstruktor, a létrehozáshoz a játékos neve szükséges.
* **+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.

### MycologistImpl

#### Felelősség

Egy gombász játékos állapotát tárolja.

#### Interfészek Mycologist

#### Ősosztályok

PlayerImpl

#### 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.
* **+ownsInscect(i: Insect)**Megadja, hogy egy rovar a játékoshoz tartozik-e.
* **+calculateScore() : int**A saját rovarjai által megevett spórák összegével tér vissza.

**(**Implementálja az ős osztályokból fakadó egyéb metódusokat**)**

### EntomologistImpl

#### Felelősség

Egy rovarász játékos állapotát tárolja.

#### Interfészek Entomolgist

#### Ősosztályok

PlayerImpl

#### Attribútumok

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

#### Metódusok

* **+addMycelium(mycelium: Mycelium)**  
  Hozzáad egy gombafonalat a gombász saját gombafonalaihoz.
* **+removeMycelim(mycelium: Mycelium)**Elvesz egy gombafonalat a gombásztól.
* **+ownsMycelium(mycelium: Mycelium)**  
  Megmondja, hogy a játékoshoz tartozik-e egy gombafonál.
* +**addMushroomBody(mushroomBody: MushroomBody)**Hozzáad egy gombatest a gombász saját gombatesteihez.
* **+removeMushroomBody(mushroomBody: MushroomBody)**

Elvesz egy gombatestet a gombásztól.

* **+ownsMushroomBody(mushroomBody: MushroomBody)**Megmondja, hogy a játékoshoz tartozik-e egy gombatest.

**(**Implementálja az ős osztályokból fakadó egyéb metódusokat**)**

### InputCommand

#### Felelősség

Tárolja a megadott parancsot és argumentumaikat.

#### Attribútumok

* **+commandName: String**A parancs neve, ami alapján egyértelműen azonosítható.
* **+commandParams: List<String>**A parancs paraméterei.

#### Metódusok

* ***+*InputCommand(name: String, params: List<String>)**A változóit inicializáló konstruktor.

### CommandImpl

#### Felelősség

A parancsot olyan formában tartalmazza, hogy egy megfelelő kezelővel futtatható legyen.

#### Interfészek Command

#### Attribútumok

* **~input: InputCommand**A megadott parancs.
* **~actingPlayer: Player**A játékos, akinek éppen köre van, ha ez irreleváns a parancshoz, akkor null. Szükséges a parancs helyességének ellenőrzéséhez.

#### Metódusok

* **+CommandImpl(actingPlayer: Player, inputCommand: InputCommand)**Konstruktor, beállítja a mostani játékost és a parancs adatait.
* ***+execute(commandHandler: CommandHandler)***A parancsot teljesíti a megadott handler által.
* **+getName() : String**Visszaadja a parancs nevét.

(Minden parancsnak van megfelelő CommandImpl osztályból származó saját implantációja, mivel ezekben lényeges változtatás nincs, amit itt fel lehetne tüntetni ezért ebből a dokumentumból olvashatósága megőrzése érdekéből ezeket itt kihagyom.)

### CommandFactoryImpl

#### Felelősség

A parancsok példányosítása.

#### Interfészek

CommandFactory

#### Metódusok

* **+createCommand(type: String): Command**Példányosít egy parancsot a megadott típussal.

### PlayerContainerImpl

#### Felelősség

A játékosok tárolásáért felelős.

#### Interfészek PlayerProvider, PlayerMutator

#### Attribútumok

* **-players: List<Player>**A játékosoknak listája.
* **-mycologists: List<Player>**

A gombászok listája.

* **-entomologists: List<Player>**A rovarászok listája.
* **-currentIndex: int**A mostani játékosnak az indexe.

#### Metódusok

* **+addPlayer(player: Player, type: String)**Hozzáad egy játékost a játékosok listájához, és a megadott típus alapján a megfelelő tárolóban. (A létrehozó parancsban ez a típus meg van adva, ezért itt nem szükséges dinamikus típus lekérdezés.)
* **+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<Player>**Visszaadja a listáját a játékosoknak.
* **+getMycologists(): Iterable<Player>**Visszaadja a gombászok listáját.
* **+getEntomologists(): Iterable<Player>**Visszaadja a rovarászok listját.

### PlayerFactoryImpl

#### Felelősség

A játékosok példányosítása

#### Interfészek

PlayerFactory

#### Metódusok

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

### PlayerControllerImpl

#### Felelősség

A játékosok létrehozásának irányítása.

#### Interfészek

PlayerFactory, CommandHandler

#### Attribútumok

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

#### Metódusok

* **+PlayerControllerImpl(factory: PlayerFactory, container: PlayerMutator)**Beállítja a példányosító és tároló objektumot.
* **+handleCommand(command: Command)**Kezeli a játékosok létrehozásával kapcsolatos parnacsokat.
* **+createPlayer(String type, String name)**A létrehoz egy játékos példány

### RoundObserverImpl

#### Felelősség

Értesíteni a feliratkozóit arról, hogy a játékkörök körbeértek.

#### Interfészek

RoundObserver

#### 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.

### TurnControllerImpl

#### Felelősség

A körök elkezdését meghatározza.

#### Interfészek TurnController, TurnInitializer, CommandHandler

#### Attribútumok

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

#### Metódusok

* **+TurnControllerImpl(conatiner: PlayerProvider, observer: RoundObserver)**Konstruktor, szükséges megadni egy tárolót és egy figyelő objektumot, ki értesíti a feliratkozóit, ha a körök körbeértek.
* **+handleCommand(command: Command)**A körök kezelésével kapcsolatos parancsokat kezeli.
* **+endTurn()**A mostani játékos körét befejezi.
* **+beginFirstTurn()**Elkezd egy kört, anélkül, hogy a jelenlegit befejezné.
* **+getCurrentPlayer()**Visszaadja azt a játékost, akinek éppen aktív köre van.

### ScoreCalculatorImpl

#### Felelősség

Eldönteni, hogy melyik játékos nyerte meg a játékot.

#### Interfészek

ScoreCalculator

#### Metódusok

* **+determineWinner(canidates: Iterable<ScoreEvaluable>) : Iterable<ScoreEvaluable>**Visszaadja a játékosokat, akiknek a pontjai a legmagasabbak.  
    
  Pszeudókód:

determineWinner(canidates: Iterable<ScoreEvaluable>):

winners: List<ScoreEvaluable>

scores: List<int>

for canidate in canidates do begin

scores.add(candiate.calculateScore())

end

for canidate in canidates do begin

if score = scores.max() then begin

winners.add(canidate)

end

end

return winners

### GameEndManagerImpl

#### Felelősség

A játék végét vezérlő osztály.

#### Interfészek OnRoundBeginSubscriber, GameEndManager, GameLengthSetter

#### Attribútumok

* **-scoreCalculator: ScoreCalculator**Az osztály amelyik kiszámolja, hogy ki a győztes.
* **-gameLength: int**A játéknak a hossza, azaz a játék végéhez hányszor érje

#### Metódusok

* **+GameEndManagerImpl(scoreCalculator: ScoreCalculator)**Konstruktor, szükséges egy objektum ami eldönti, hogy ki a győztes.
* **+onRoundBegin()**Mindig amikor a játékosok köre körbér, a játék maradék ideje eggyel csökken.
* **+setGameLength(newLength: int)**A játéknak hosszának setterje.
* **+getGameLength(): int**

Játék hosszának getterje.

* **+showWinners()**Kiírja a győzteseket

### GameControllerImpl

#### Felelősség A játék menetével elindításért

#### Interfészek GameController, CommandHandler

#### Attribútumok

* **-turnController: TurnInitializer**A köröket irányító kontroller, amit a játékkezdetekor elindít.
* -**gameEndManager: GameLengthSetter**A játék végét számontartó objektum, a játék kezdetekor a visszaszámlálása elindul.

#### Metódusok

* **+GameControllerImpl(turnIntializer: TurnInitializer,   
  gameLengthSetter: GameLengthSetter)**Konstruktor, meg kell adni az osztályt ami elkezdi számolni a köröket és ami beállítja a játék hosszát.
* **+handleCommand(command: Command)**Kezeli a játék kezdésével vagy befejezésével kapcsolatos parancsokat.
* +**beginGame(length: int)**A játék elkezdődik a megadott megadott kör limittel.
* **+endGame()**Befejezi a játékot.

### InsectControllerImpl

#### Felelősség

A rovarokat irányítása.

#### Interfészek

InsectController, CommandHandler

#### Metódusok

* **+handleCommand(command: Command)**Kezeli a rovar irányításával kapcsolatos parancsokat.
* **+cut(insect: Insect)**A kiválasztott rovarral elvágja a tektont.
* **+eat(insect: Insect)**A kiválasztott rovarral eszik.
* **+move(insect: Insect, destination: Tecton)**

A kiválasztott rovart mozgatja.

### MyceliumFactoryImpl

#### Felelősség

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

#### Interfészek

MyceliumFactory

#### Metódusok

* +**createMycelium(type: String, name: String, location: Tecton): Mycelium**A növesztési feltételek követésével növeszt egy gomafonalat, a megadott tektonra, a megadott névvel.

### CheatMyceliumFactory

#### Felelősség

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

#### Interfészek

MyceliumFactory

#### Metódusok

* +**createMycelium (type: String, name: String, location: Tecton): Mycelium**A növesztési feltételek nélkül növeszt egy gombafonalat, a megadott tektonra, a megadott névvel.

### DefaultMushroomBodyFactory

#### Felelősség

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

#### Interfészek

MushroomBodyFactory

#### Metódusok

* +**createMushroomBody(name: String, location: Tecton) : MushroomBody**A növesztési feltételek követésével növeszt egy gombatestet, a megadott tektonra, a megadott névvel.

### CheatMushroomBodyFactory

#### Felelősség

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

#### Interfészek

MushroomBodyFactory

#### Metódusok

* +**createMushroomBody(name: String, location: Tecton) : MushroomBody**A növesztési feltételek nélkül növeszt egy gombatestet, a megadott tektonra, a megadott névvel.

### MushroomBodyControllerImpl

#### Felelősség

Gombatestek irányítása.

#### Interfészek

MushroomBodyController, CommandHandler

#### Attribútumok

* **-sporeFactory: SporeFactory**A spórát példányosító objektum.

#### Metódusok

* **+MushroomBodyControllerImpl(factory: SporeFactory)**Konstruktor. Szükséges egy spórákat példányosító objektum megadása.
* **+eject(source: MushroomBody, target: Tecton)**A megadott gombatest spóráit kilöveti a megadott tektonra.
* **+deactivate(mushroomBody: MushroomBody)**A megadott gombatestet deaktiválja, azaz már nem lőhet ki több spórát.
* **+addSpores(sporeType: String, sporeName: String, target: MushroomBody)**Egy új spórát hozzáad a gombatesthez, a megadott paraméterek alapján.

### TectonFactoryImpl

#### Felelősség

Tektonok példányosítása

#### Interfészek

TectonFactory

#### Metódusok

* **+create(type: String, name: String): Tecton**Példányosít egy tektont a megadott típussal és névvel.

### TectonControllerImpl

#### Felelősség

Tektonok vezérlése

#### Interfészek

TectonController, CommandHandler

#### . Attribútumok

* **-sporeFactory: SporeFactory**A spórát példányosító objektum.

#### Metódusok

* **+TectonControllerImpl(factory: SporeFactory)**Konstruktor. Szükséges egy spórákat példányosító objektum megadása.
* **+handleCommand(command: Command)**Kezeli a tektonok kezelésével kapcsolatos parancsokat.
* **+break(tecton: Tecton)**Eltöri a megadott tektont.
* **+setBreakTimer(tecton: Tecton, time: int)**A törési időzítőt a megadott értékre.
* **+addNeighbour(tecton1: Tecton, tecton2: Tecton)**A megadott két megadott tektont egymással szomszédossá teszi.
* **+putSpore(sporeType: String, sporeName: String, target: Tecton)**Egy új spórát hozzáad a tektonhoz, a megadott paraméterek alapján.

### InsectFactoryImpl

#### Felelősség

Rovarok példányosítása

#### Interfészek

InsectFactory

#### Metódusok

* **+create(name: String) : Insect**Példányosít egy rovart a megadott névvel.

### SporeFactoryImpl

#### Felelősség

Rovarok példányosítása

#### Interfészek

SporeFactory

#### Metódusok

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

Példányosít egy spórát a megadott névvel és típussal.

### MapCreationControllerImpl

#### Felelősség

A játéktér létrehozásának irányítása.

#### Interfészek

MapCreationController, CommandHandler

#### Attribútumok

* **-mushroomBodyFactory: MushroomBodyFactory**A gombatesteket a térkép létrehozásakor példányosító objektum.
* **-myceliumFactory: MyceliumFactory**A gombafonalakat a térkép létrehozásakor példányosító objektum.
* **-insectFactory: InsectFactory**A rovarokat a térkép létrehozásakor példányosító objektum.

#### Metódusok

* **+MapCreationControllerImpl(  
  mushroomBodyFactory: MushroomBodyFactory,  
  myceliumFactory: MyceliumFactory, insectFactory: InsectFactory)**Konstruktor. Meg kell adni az objektumokat, amelyek példányosítják a gombatesteket, gombafonalakat és a rovarokat.
* **+handleCommand(command: Command)**A térkép létrehozásával kapcsolatos parancsokat kezeli.
* **+createMycelium(name: String, type: String, location: Tecton)**Létrehoz egy gombafonalat a megadott paraméterek alapján.
* **+createMushroomBody(name: String, location: Tecton)**Létrehoz egy gombatestet a megadott paraméterek alapján.
* +**createInscet(name: String)**Létrehoz egy rovart a megadott paraméterek alapján.

### GrowthControllerImpl

#### Felelősség

A gombarészek növekedésének irányítása.

#### Interfészek

GrowthController, CommandHandler

#### Attribútumok

* + - * **-mushroomBodyFactory: MushroomBodyFactory**A gombatestet példányosító objektum.
      * **-myceliumFactory: MushroomBodyFactory**A gombafonalat példányosító objektum.

#### Metódusok

* **+GrowthControllerImpl(mushroomBodyFactory: MushroomBodyFactory  
  myceliumFactory: MushroomBodyFactory)**Konstruktor. Meg kell adni az objektumokat, amelyek példányosítják a gombatesteket és a gombafonalakat.
* **+handleCommand(command: Command)**Kezeli a növesztéssel kapcsolatos parancsokat.
* **+growMycelium(name: String, location: Tecton)**Növeszt egy gombafonalat a megadott tektonra, a megadott névvel.
* **+growMushroomBody(name: String, location: Tecton)**Növeszt egy gombatestet a megadott tektonra, a megadott névvel.

### CommandRouterImpl

#### Felelősség A parancsokat a megfelelő kezelőnek továbbadja.

#### Interfészek

CommandRouter

#### Attribútumok

* **-commandRepository: Map<String, CommandHandler>**Egy tábla, ami leírja, hogy egy adott parancsot melyik kezelő fogadja be.
* **-commandFactory: CommandFactory**A parancsokat példányosító objektum.

#### Metódusok

* **+CommandRouterImpl(factory: CommandFactory)**Konstruktor, a létrehozáshoz szükséges megadni a parancsokat példányosító objektumot.
* **+routeCommand(command: InputCommand)**A kapott parancsot továbbítja a megfelelő kezelőnek.  
  Pszeudókód:

routeCommand(command: InputCommand)

routedCommand: Command := commandFactory.create(command.name)

handler: CommandHandler := commandRepository[command.name]

handler.handleCommand(routedCommand)

end

* **+addCommand(commandName: String, commandHandler: CommandHandler)**A táblában egy új bejegyzést ír be, ami azt írja le, hogy a parancsot melyik

### CommandReaderImpl

#### Felelősség

A felhasználó által megadott parancsokat olvassa be, majd továbbítja az értelmezőnek.

#### Interfészek

CommandReader, CommandHandler

#### Attribútumok

* **- commandRouter: CommandRouter**A parancsokat továbbküldő osztály.
* **- inputBuffer: Queue<String>**

Egy input puffer, amiből az olvasó elsődlegesen kiolvas.

#### Metódusok

* **+CommandReaderImpl(commandRouter: CommandRouter)**Konstruktor, meg kell adni az objektumot aminek továbbküldi a parancsot.
* **+handleCommand(command: Command)**Fogadhat parancsokat is vissza, pl: a run parancs, ami egy fájlal feltölti a puffert.
* **+ getNextCommand()**Értelmezi a következő parancsot a pufferből, ha a puffer üres, akkor a játékostól kér új parancsot.

Pszeudókód:

getNextCommand()

readCommand: String

if inputbuffer is not empty then begin

readCommand := inputBuffer.poll()

end

else then begin

readCommand := input()

end

splitCommand: String[] := readcommand split by spaces

inputCommand: InputCommand := new(splitCommand[0],splitCommand[1:n])

commandRouter.routeCommand(inputCommand)

end

* **+ bufferCommand(input: String)**Egy parancsot berak a pufferbe.

### TracablePrinterImpl

#### Felelősség

Az ellenőrizhetőség érdekében, úgy írja ki, hogy visszaolvasható legyen a kódban.

#### Interfészek

CommandHandler, TracablePrinter

#### Attribútumok

* **-printHistory: List<String>**A kiírt szövegeket tároló lista.

#### Metódusok

* **+clearHistory()**Kitölri a visszaolvasható
* **+readHistory(): Iterable<String>**Visszaadja a jelenleg eltárolt régi kiírásokat.
* **+print(output: String)**Kiírja a szöveget egy új sorba, és eltárolja.
* **+printLine(output: String)**Kiírja a szöveget egy új sorba, és eltárolja.
* **+handleCommand(command: Command)**Kezeli a kiírással kapcsolatos parancsokat (pl: STATE)

### ObjectRegistry

#### Felelősség

Számon tartja a névvel ellátott objektumokat

#### Attribútumok

* **-registeredObjects: Map<String, Object>**Az objektumok szövegesen hivatkozható nevei és a velük asszociált objektumok.

#### Metódusok

* **-ObjectRegistry()**

A konstruktor privát, mert nem szabad példányosítani.

* **+registerObject(name: String, registeredObject: Object)**Beírja az objektumot és a nevét a tárba.
* **+clearRegistry()**Kitörli az összes objektumot a tárból.
* **+removeFromRegistry(name: String)**Töröl egy objektumot a tárból.
* **+getObject(name: String) : Object**Név alapján visszatér a keresett objektummal.

## 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 program JUnit 5 1.12.2-es verziójának a „Console Standalone” változatát használja. Minden teszthez tartozik egy külön metódus, amiben a teszt futtatásához szükséges parancsok le vannak írva.

A teszt „arrange” és „act” része kiadható parancsokkal van leírva. Az „assert” része a kiírt kimenetet ellenőrzi. Ezt a „TracablePrinter” osztállyal valósítsuk meg, ami a kiírt kimenetet eltárolja, ezzel vissza lehet olvasni. Az ellenőrzését a JUnit „Assertions” osztályának megfelelő függvényeivel valósítjuk meg.

A tesztek szét vannak választva külön-külön tesztosztályokba tematikájuk alapján. Ha a felhasználó egy bizonyos tesztet szeretne futtatni, vagy egy tesztosztályba tartozó teszteket akkor azt megteheti a futtatás --select-class <Tesztosztály> vagy --select-method ’<Tesztosztály>#<Tesztmetódus>’ paraméterével. Ha az összes tesztet futtatni szeretné, akkor a futtatáshoz a --scan-classpath paramétert kell megadni.

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2025.04.03. 16:30 | 1 óra 30 perc | Kohár | Tevékenység:   * Controller osztályok kezdetleges terveinek leírása |
| 2025.04.04. 17:20 | 2 óra | Kohár | Tevékenység:   * Controller osztályok tagfüggvényeinek kidolgozása |
| 2025.04.05. 09:45 | 1 óra 45 perc | Kohár | Tevékenység:   * Controller osztályok javítása |
| 2025.04.07. 18:20 | 1 óra 45 perc | Kohár | Tevékenység:   * Controller osztályok kiegészítése hiányzó osztályokkal |
| 2025.04.08. 17:25 | 1 óra | Kohár | Tevékenység:   * Controller osztályok bonyolultabb tagfüggvényeinek leírása pszeudókóddal |
| 2025.04.09. 11:15 | 15 perc | Kohár | Tevékenység:   * Pszeudókód javítása |
| 2025.04.10. 16:15 | 45 perc | Kohár | Tevékenység:   * Controller hiányzó függvényeinek leírása |
| 2025.04.11. 19:30 | 1 óra | Kohár | Tevékenység:   * Tesztelést támogató programok terveinek leírása |

|  |  |  |  |
| --- | --- | --- | --- |
| 2025.04.13. 11:10 | 30 perc | Kohár | Tevékenység:   * Tesztelést támogató programok terveiben javítás |