**Analízis modell (I. változat)**

55 – ubr

Konzulens:

Potyók Csaba

Csapattagok

| Bogár Dániel | I07XOI | bogardanischool@gmailcom |
| --- | --- | --- |
| Koleszár Gábor | GSQ9GD | kolesz.kole@gmail.com |
| Jánovszki András | GUFP6A | janovszki84@gmail.com |
| Zombori Péter  Varjú Zalán Péter | DSYU1P  P2938R | [zombori50@gmail.com](mailto:zombori50@gmail.com)  varjzalan@gmail.com |

2024.03.04.

# **Analízis modell kidolgozása**

## ***Objektum katalógus***

### Dice

A Dice egy tárgyat valósít meg (dobókocka), amelyet felhasználva dobhat még egyszer a játékos.

### FFP2

Az FFP2 objektum egy tárgyat valósít meg (FFP2 Maszk), amelyet felhasználva a játékos bizonyos ideig védettséget élvez a gázos szobák hatásai ellen.

**3.1.3 TVSZ**

A TVSZ objektum egy tárgyat valósít meg (TVSZ), amelyet, ha felvesz egy játékos, akkor meghatározott ideig védett lesz a tanárok ellen.

**3.1.4 Transistor**

A Transistor objektum egy speciális tárgyat valósít meg a programban, amelyből, ha felvesz két felet egy játékos és leteszi őket két különböző szobába, akkor aktiválás után át tud menni a két szoba között.

**3.1.5 Camambert**

A Camambert objektum egy tárgyat valósít meg a programban, amelyet, ha egy játékos letesz egy szobában, akkor az a szoba kör végeztével gázos szobábvá alakul.

**3.1.6 GlassOfBeer**

A GlassOfBeer objektum egy tárgyat valósít meg a játékban (Söröspohár), amelyet a játékos felhasználva egy körig védett lesz a tanárok ellen.

**3.1.7 WetRag**

A WetRag objektum egy tárgyat valósít meg a játékban (Vizes táblatörlő rongy), amelyet, ha felhasznál egy játékos, akkor az abban a szobában lévő összes karakter kimarad egy körből.

A WetRag-et 3 körig lehet felhasználni, ugyanis elveszíti a nedvességét vagyis

kiszárad 3 kör után és, akkor nem lehet felhasználni.

**3.1.8 SlideRule**

A SlideRule objektum egy tárgyat valósít meg a játékban (Logarléc), amelyet, ha felvesz egy diák, akkor a diákok győznek.

Tanárok ezt a tárgyat nem tudják felvenni.

**3.1.9 RegularRoom**

A RegularRoom objektum az egyszerű szobákat (Szoba) valósítja meg, vagyis vannak neki szomszédai, vagyis azok a szobák amelyekbe el lehet innen jutni, illetve figyeli, hogy rakott-e le ide valaki tranzisztort.

Minden szobának van egy maximális tárgy és karakter befogadóképessége, amelyeket ez az objektum kezel, ha ezek közül valamelyik megtelik, akkor, vagy nem lehet több tárgyat abban a szobában hagyni, vagy nem léphet be több karakter ide.

**3.1.10 GasRoom**

A GasRoom a gázos szobákat valósítja meg, amelyek a legtöbb paraméterben megegyeznek a sima szobákkal, azonban, ha ide valaki védelem nélkül (FFP2 Maszk) lép be, akkor a belépett diák elejti az összes nála lévő tárgyat.

**3.1.11 Student**

A Student objektum a karakterek egyik változatát, a diákokat valósítja meg. A diákok úgynevezett action-öket hajtanak végre a játék folyamán, amelyek számát a kockával való dobás határozza meg.

Egy diák kiesik a játékból, ha egy mezőre lép egy tanárral védelem nélkül.

**3.1.12 Instructor**

Az Instructor objektum a karakterek egyik változatát, a tanárokat valósítja meg. A tanárok nem tudnak tárgyakat felhasználni, csak felvenni, illetve, csak egy random számot lépnek minden körben.

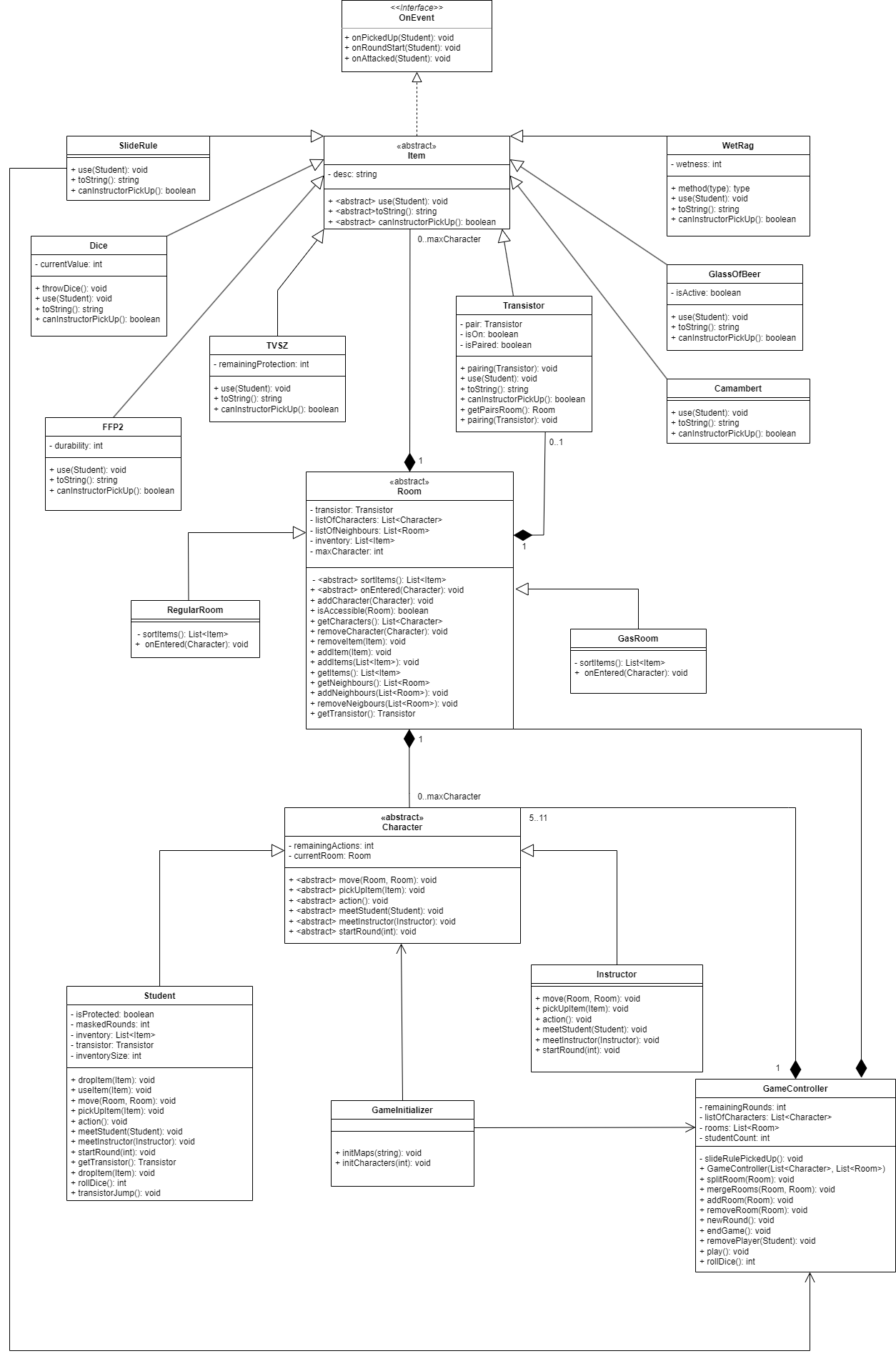
**3.1.13 GameController**

A GameController objektum felelőssége, a játék levezénylése, illetve a szobák és a játékosok számon tartása.

**3.1.14 GameInitializer**

A GameInitializer objektum feladata a játékmenet létrehozása. Beolvassa a pályákat és inicializálja a karaktereket.

## ***Statikus struktúra diagram***



## ***Osztályok leírása***

### 3.3.1 Camambert

#### **Felelősség**

Ez az osztály valósítja meg a Camambert nevű tárgyat.

#### **Ősosztályok**

* Item

#### **Interfészek**

* OnEvent

#### **Attribútumok**

* desc: string

#### **Metódusok**

* **void use(Student s):** A tárgy használatának megvalósítása.
* **string toString():** Visszaadja a tárgy leírását, felülírja az alap toString metódust.
* **boolean canInstructorPickUp():** Visszaadja, hogy a tárgy felvehető-e oktatók által.

### 3.3.2 Character

#### Felelősség

Ez az absztrakt ősosztály felelős, a játékban található összes karakter alap metódusainak a deklarálásáért és a bővíthetőség biztosításáért.

#### Asszociációk

* **asszociáció1**:

A GameInitializer a másik fél.

A kapcsolat célja a karakterek létrehozása a játék kezdetekor.

* **asszociáció2**:

A GameController a másik fél.

A kapcsolat célja a játékmenetben lévő karakterek tárolása.

#### Attribútumok

* remainingActions: int
* currentRoom: Room

#### Metódusok

* **void move(Room r1, Room r2):** A metódus az adott karaktert áthelyezi az egyik bemenetként kapott szobából a másikba.
* **void pickUpItem(Item i ):** A paraméterként kapott tárgyat elhelyezi a karakter inventoryjában.
* **void action():** Végrehajtja egy action-jét a karakternek.
* **void meetStudent(Student s):** Azt az esetet kezeli a függvény, amikor egy karakter egy hallgatóval kerül egy mezőre.
* **void meetInstructor(Instructor i):** Azt az esetet kezeli, amikor egy karakter egy oktatóval kerül egy mezőre.
* **void startRound(int i):** Elindítja a játékos körét, és meghívja a paraméterként kapott számmal az ‘action’ függvényt

### 3.3.3 Dice

#### Felelősség

Ez az osztály felelős a játékban található dobókocka tárgy megvalósításáért.

#### Ősosztályok

* Item

#### Interfészek

* OnEvent

#### Attribútumok

* desc: string
* currentValue: int

#### Metódusok

* **void use(Student):** A tárgy felhasználásáért felelős.
* **string toString():**  Visszaadja a tárgy leírását.
* **boolean canInstructorPickUp():** Visszaadja, hogy a tárgy felvehető-e oktatók által.
* **void throwDice():** Eldob egy virtuális dobókockát, és a dobott értéket hozzáadja az adott játékos maradék action-jéhez.

### 3.3.4 FFP2

#### Felelősség

Ez az osztály felelős a játékban megtalálható FFP2 maszk megvalósításáért.

#### Ősosztályok

* Item

#### Interfészek

* OnEvent

#### Attribútumok

* desc: string
* durability: int

#### Metódusok

* **void use(Student):** A tárgy felhasználásáért felelős.
* **string toString():**  Visszaadja a tárgy leírását.
* **boolean canInstructorPickUp():** Visszaadja, hogy a tárgy felvehető-e oktatók által.

### 3.3.5 GameController

#### Felelősség

Ez az osztály felelős a játékmenet vezérléséért. Ebben az osztályban tartjuk számon a karaktereket és a szobákat.

#### Asszociációk

* **Asszociáció 1:**

A másik fél a Room.

A kapcsolat célja, hogy a GameController számon tudja tartani a szobákat.

* **Asszociáció 2:**

A másik fél a Character.

A kapcsolat célja, hogy a GameController számon tudja tartani a karaktereket.

* **Asszociáció 3:**

A másik fél a GameInitializer.

A kapcsolat célja, hogy a GameInitializer átadja a beolvasott és inicializált adatokat a GameControllernek.

* **Asszociáció 4:**

A másik fél a SlideRule.

A kapcsolat célja, hogy a GameController meg tudja állapítani, ha valaki felvette a logarlécet és ezáltal kezelhesse ezt az esetet.

#### Attribútumok

* remainingRounds: int
* listOfCharacters: List<Character>
* rooms: List<Room>
* studentCount: int

#### Metódusok

* **void slideRulePickedUp()**: Kezeli azt az eseményt, amikor a logarlécet felvették.
* **GameController(List<Character> l1, List<Room> l2)**: Létrehoz egy játékot a megadott karakter lista és szoba lista segítségével.
* **void splitRoom(Room r1)**: Felosztja a paraméterként megadott szobát 2 új szobára.
* **mergeRoom(Room r1, Room r2)**: A paraméterként megadott két szobából egy új szobát hoz létre.
* **void addRoom(Room r1)**: A paraméterként megadott szobát hozzáadja a játék szobáihoz.
* **void removeRoom(Room r1)**: A paraméterként megadott szobát kitörli a játék szobáiból.
* **void newRound()**: elindít egy új kört, meghívja a játékosok ‘startRound’ függvényét
* **void endGame()**: Végetvet a játékmenetnek.
* **void removePlayer(Student)**: Kitörli a paraméterként kapott játékost a játékmenetből.
* **void play()**: Elindítja a játékmenetet.
* **int rollDice()**: Eldob egy virtuális dobókockát, magyarán visszaad egy random számot 1 és 6 között.

### 3.3.6 GameInitializer

#### Felelősség

Ez az osztály felelős a játék létrehozásához szükséges adatok beolvasásáért.

#### Asszociációk

* **Asszociáció 1:**

A másik fél a Character.

A beolvasott Charactereket átadja a karakter osztálynak.

* **Asszociáció 2:**

A másik fél a GameController.

A kapcsolat célja, hogy a GameInitializer átadja a beolvasott és inicializált adatokat a GameControllernek.

#### Metódusok

* **void initMaps(string s1)**: A pálya beolvasása és felépítése fájlból.
* **void initCharacters(int i)**: Paraméterként megadott számú karaktert inicializál a pályára.

### 3.3.7 GasRoom

#### Felelősség

Ez az osztály valósítja meg a játékban található gázos szobákat.

#### Ősosztályok

* Room

#### Metódusok

* **void onEntered(Character c1)**: Azt az eseményt kezeli, amikor egy karakter a mezőre lép, ilyenkor, ha a karakter nem védett, a szoba kifejti rá hatását

### 3.3.7 GlassOfBeer

#### Felelősség

Ez az osztály valósítja meg a játékban található söröspohár tárgyat.

#### Ősosztályok

* Item

#### Interfészek

* OnEvent

#### Attribútumok

- isActive: boolean

#### Metódusok

* **void use(Student s1)**: A tárgy használatának megvalósításáért felelős metódus.
* **string toString()**: string: Visszaadja a tárgy leírását.
* **boolean canInstructorPickUp()**: Visszaadja, hogy oktatók felvehetik-e ezt a tárgyat.

### 3.3.8 Instructor

#### Felelősség

Ez az osztály valósítja meg a játékban található egyik fajta karaktert az oktatót/tanárt.

#### Ősosztályok

* Character

#### Metódusok

* **void move(Room r1, Room r2):** A metódus az adott karaktert áthelyezi az egyik bemenetként kapott szobából a másikba.
* **void pickUpItem(Item i ):** A paraméterként kapott tárgyat elhelyezi a karakter inventoryjában.
* **void action():** Végrehajtja az instruktor egyik lépését, action-jét.
* **void meetStudent(Student s):** Azt az esetet kezeli a függvény, amikor egy karakter egy hallgatóval kerül egy mezőre.
* **void meetInstructor(Instructor i):** Azt az esetet kezeli, amikor egy karakter egy oktatóval kerül egy mezőre.
* **void startRound(int i): -**Elindítja az instructor körét, és meghívja a paraméterként kapott számmal az ‘action’ függvényt

### 3.3.9 Item

#### Felelősség

Ez az osztály felelős a játékban található összes tárgy alap metódusainak megvalósításáért.

#### Interfészek

* OnEvent

#### Attribútumok

- desc: string

#### Metódusok

* **void use(Student s)**: A tárgyak felhasználását valósítja meg.
* **string toString()**: Visszaadja az adott tárgy leírását(desc).
* **boolean canInstructerPickUp()**: Visszaadja, hogy az adott tárgyat felveheti-e oktató.

### 3.3.10 OnEvent

#### Felelősség

Ez az interfész felelős a játékban található tárgyakkal kapcsolatos bizonyos események kezeléséért..

#### Metódusok

* **void onPickedUp(Student)**: Azt az eseményt kezeli, amikor egy tanuló felvesz egy tárgyat.
* **void onRoundStart(Student)**: Elvégzi a kör elején szükséges módosításokat a pályán.
* **void onAttacked(Student)**: Azt az eseményt kezeli, amikor egy hallgató találkozik egy oktatóval, tehát egy mezőre kerülnek.

### 3.3.11 RegularRoom

#### Felelősség

Ez az osztály felelős a játékban megtalálható általános szobák megvalósításáért.

#### Ősosztályok

* Room

#### Metódusok

* **void onEntered(Character c1)**: Azt az eseményt kezeli, amikor egy karakter a mezőre lép.

### 3.3.12 Room

#### Felelősség

Ez az osztály egy absztrakt ősosztály, amely a játékban megtalálható mezők, vagyis szobák megvalósítására szolgál.

#### Asszociációk

* **Asszociáció 1:**

A másik fél az Item.

A szobák tárgyakat tartalmazhatnak.

* **Asszociáció 2:**

A másik fél a Transistor.

A szobákra elhelyezhető maximum egy tranzisztor.

* **Asszociáció 3:**

A másik fél a Character.

A szobákba beléphet előre meghatározott számú karakter.

* **Asszociáció 4:**

A másik fél a GameController.

A GameController tárolja a játékmenetben lévő szobákat.

#### Attribútumok

* transistor: Transistor
* listOfCharacters: List<Character>
* listOfNeighbours: List<Room>
* inventory: List<Item>
* maxCharacter: int

#### Metódusok

* **void onEntered(Character c1)**: Azt az eseményt kezeli, amikor egy karakter a mezőre lép.
* **void addCharacter(Character c1)**: Hozzáadja az paraméterként kapott karaktert a szobához.
* **boolean isAccessible(Room r1)**: -
* **List<Character> getCharacters()**: Visszaadja a szobában található karaktereket.
* **void removeCharacter(Character c3)**: Kitörli a paraméterként kapott karaktert a szobából.
* **void removeItem(Item)**: Kitörli a paraméterként kapott Itemet a szobából.
* **void addItem(Item)**: Hozzáadja a paraméterként kapott Item-et a szobához.
* **void addItems(List<Item> l1)**: Hozzáadja a szobához tárgyak paraméterként kapott listáját.
* **List<Item> getItems()**: Visszadja a szobában található tárgyakat.
* **List<Room> getNeighbours()**: Visszaadja a szomszédos szobákat.
* **void addNeighbours(List <Room> l2)**: A paraméterként kapott lista tagjait hozzáadja a szoba szomszédaihoz.
* **void removeNeighbours(List<Room> l1)**: Kitörli a szoba szomszédai közül a paraméterként kapottakat.
* **Transistor getTransistor()**: Visszaadja a szobában elhelyezett tranzisztort.

### 3.3.13 SlideRule

#### Felelősség

Ez az osztály felelős a játékban megtalálható logarléc tárgy megvalósításáért.

#### Ősosztályok

* Item

#### Interfészek

* OnEvent

#### Asszociációk

* **Asszociáció 1:**

A másik fél a GameController.

A kapcsolat célja az, hogy a GameController kezeli azt az eseményt, ha egy diák felveszi a logarlécet.

#### Metódusok

* **void use(Student s1)**: A tárgy használatának megvalósításáért felelős metódus.
* **string toString():** Visszaadja a tárgy leírását.
* **boolean canInstructorPickUp()**: Visszaadja, hogy oktatók felvehetik-e ezt a tárgyat.

### 3.3.14 Student

#### Felelősség

Ez az osztály felelős a diákok megvalósításáért.

#### Ősosztályok

* Character

#### Attribútumok

* isProtected: boolean
* maskedRounds: int
* inventory: List<Item>
* transistor: Transistor
* inventorySize: int

#### Metódusok

* **void dropItem(Item i1)**: Egy tárgy eldobását megvalósító metódus.
* **void useItem(Item i2)**: Egy tárgy alkalmazását megvalósító metódus.
* **void move(Room r1, Room r2)**: Egyik mezőről egy másikra való lépést megvalósító metódus.
* **void pickUpItem(Item i3)**: Egy tárgy felvételét megvalósító metódus.
* **void action()**:Végrehajtja a Student egyik action-jét.
* **void meetStudent(Student s1)**: Ez a függvény azt az esetet kezeli, amikor egy hallgató egy másik hallgatóval kerül egy mezőre.
* **void meetInstructor(Instructor in)**: Ez a függvény azt az esetet kezeli, amikor egyhallgató egy oktatóval kerül egy mezőre.
* **void newRound(int i2)**: Elindítja az instructor körét, és meghívja a paraméterként kapott számmal az ‘action’ függvényt.
* **Transistor getTransistor()**: Visszaadja a játékosnál lévő, még nem párosított tranzisztort
* **int rollDice()**: Egy véletlenszerű egész számot generál az 1-től 6-ig terjedő zárt intervallumon.
* **void transistorJump()**: A tranzisztorral való utazást megvalósító függvény.

### 3.3.15 Transistor

#### Felelősség

Ez az osztály felelős a játékban megtalálható tranzisztor tárgy megvalósításáért.

#### Ősosztályok

* Item

#### Asszociációk

* **Asszociáció 1:**

A másik fél a Room.

A kapcsolat célja az, hogy a szobák számon tudják tartani, hogy van-e rajtuk tranzisztor.

#### Attribútumok

- pair: Transistor

- isOn: boolean

- isPaired: boolean

#### Metódusok

* **void pairing(Transistor t1)**: A paraméterként megadott tranzisztor összepárosítja az adott tranzisztorral.
* **void use(Student s1)**: A tárgy használatának megvalósításáért felelős metódus.
* **string toString()**: string: Visszaadja a tárgy leírását.
* **void canInstructorPickUp()**: Visszaadja, hogy oktató felveheti-e a tárgyat.
* **Room getPairsRoom()**: Visszaadja a vele összekapcsolt tranzisztor szobáját.

### 3.3.16 TVSZ

#### Felelősség

Ez az osztály felelős a játékban megtalálható tvsz tárgy megvalósításáért.

#### Ősosztályok

* Item

#### Interfészek

* OnEvent

#### Attribútumok

- remainingProtection: int

#### Metódusok

* **void use(Student s1)**: A tárgy használatának megvalósításáért felelős metódus.
* **string toString()**: Visszaadja a tárgy leírását.
* **boolean canInstructorPickUp()**: Visszaadja, hogy oktatók felvehetik-e ezt a tárgyat.

### 3.3.17 WetRag

#### Felelősség

Ez az osztály felelős a játékban megtalálható nedves rongy tárgy megvalósításáért.

#### Ősosztályok

* Item

#### Interfészek

* OnEvent

#### Attribútumok

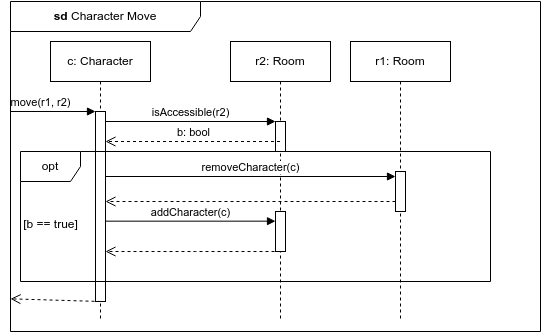
- wetness: int

#### Metódusok

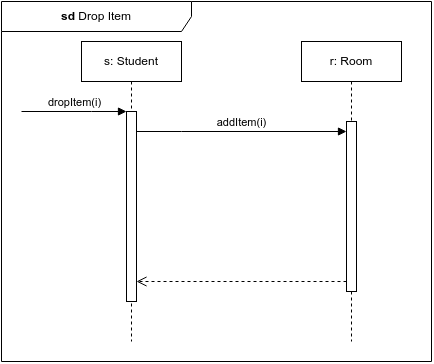
* **void use(Student s1)**: A tárgy használatának megvalósításáért felelős metódus.
* **string toString()**: Visszaadja a tárgy leírását.
* **boolean canInstructorPickUp()**: Visszaadja, hogy oktatók felvehetik-e ezt a tárgyat.

## ***Szekvencia diagramok***

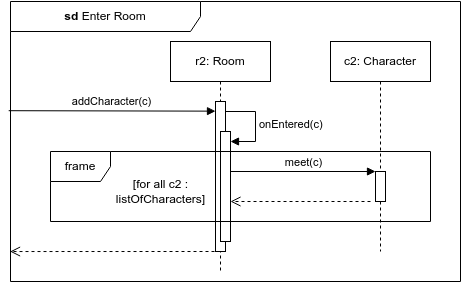
* + 1. Character Move



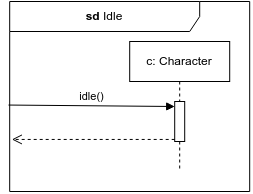
* + 1. Drop Item



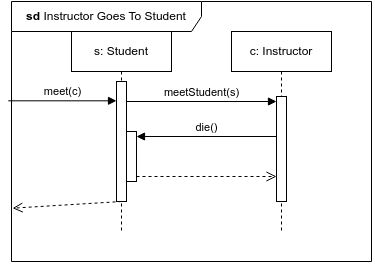
* + 1. Enter Room



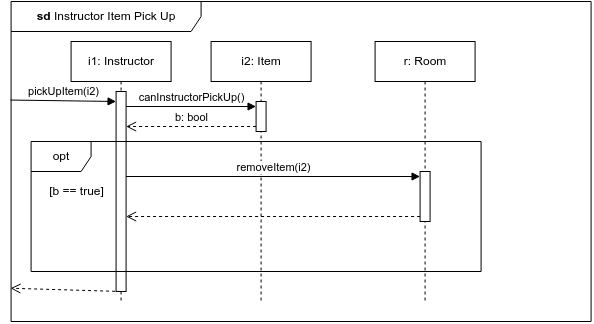
* + 1. Idle



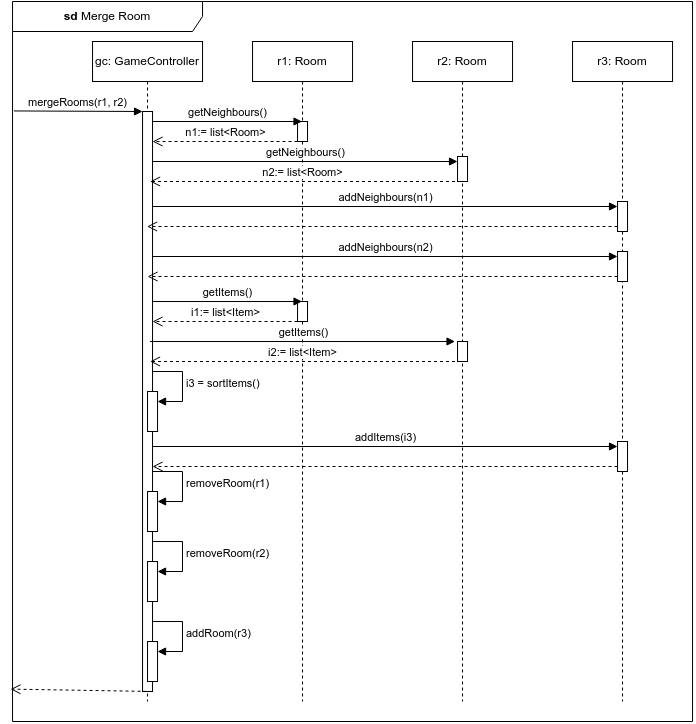
* + 1. Instructor Goes To Student



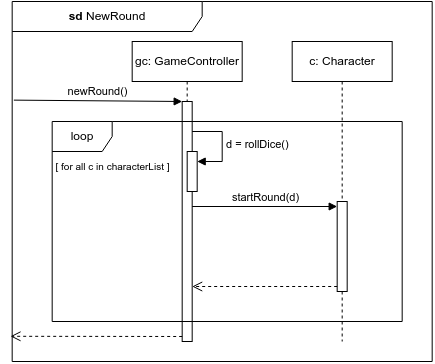
* + 1. Instructor Item Pick Up



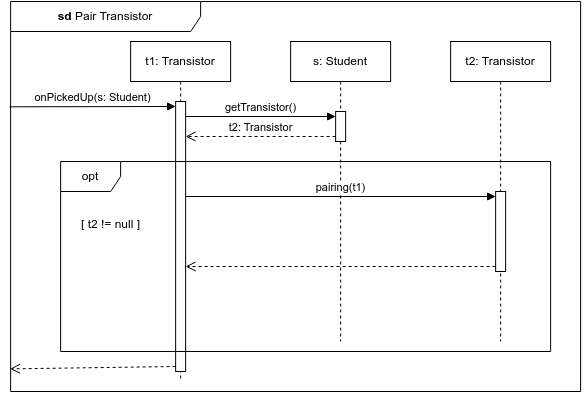
* + 1. Merge Room



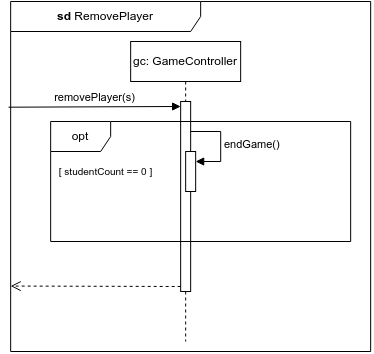
* + 1. New Round



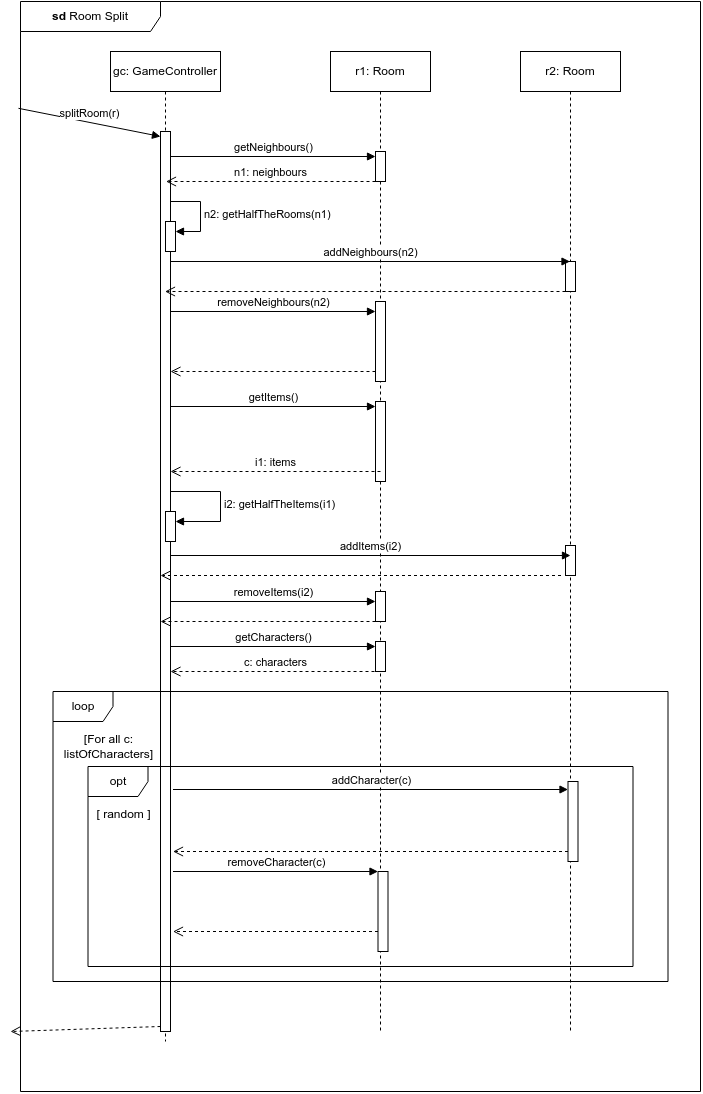
* + 1. Pair Transistor



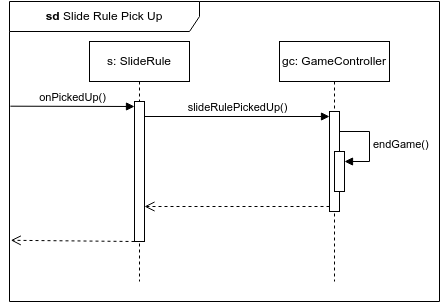
* + 1. Remove Player



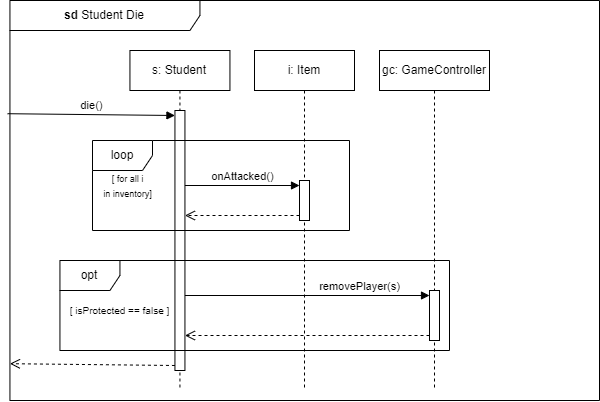
* + 1. Room Split



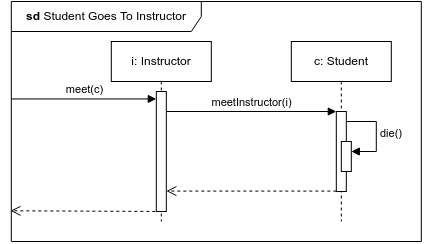
* + 1. Slide Rule Pick Up



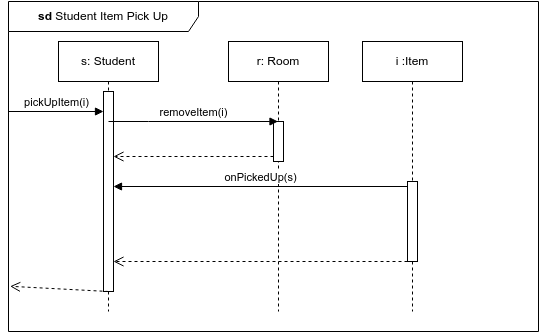
* + 1. Student Die



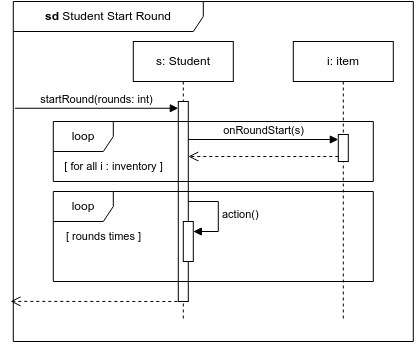
* + 1. Student Goes To Instructor



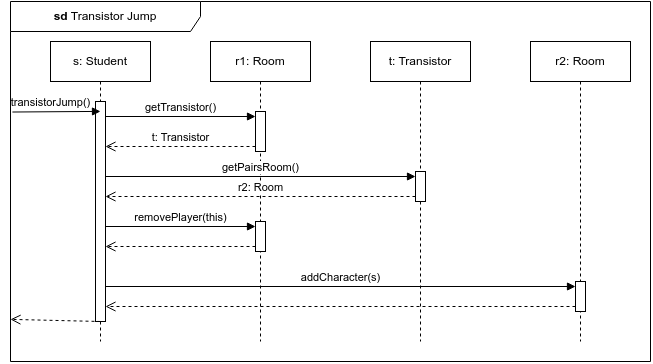
* + 1. Student Item Pick Up



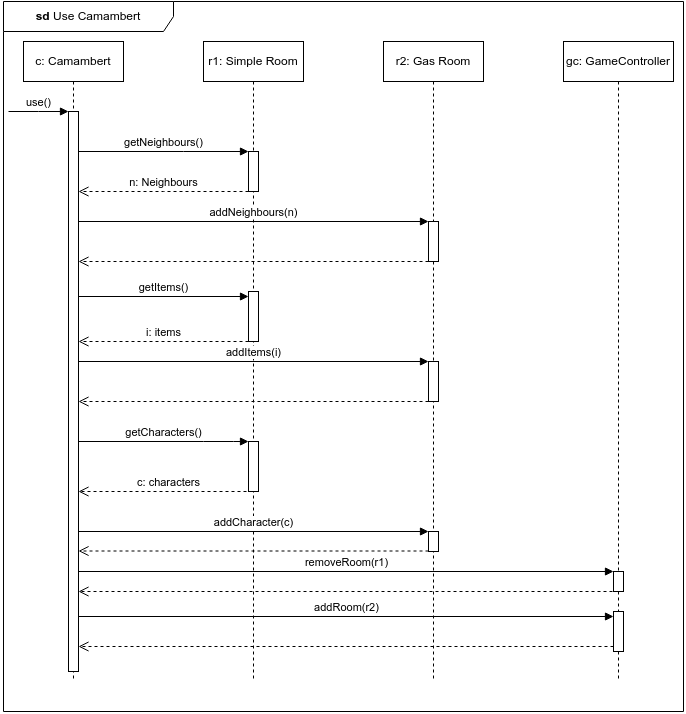
* + 1. Student Start Round



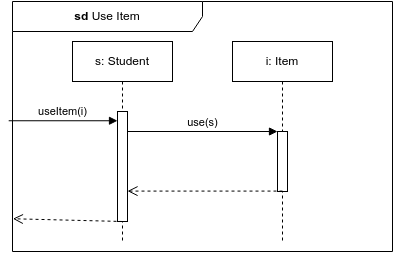
* + 1. Transistor Jump



* + 1. Use Camembert

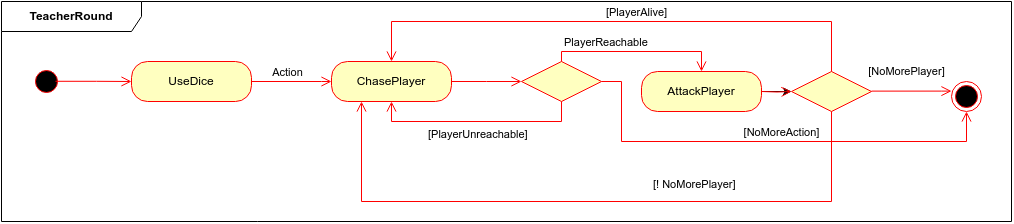


* + 1. Use Item

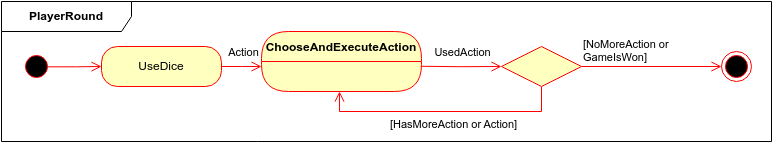


## ***State-chartok***

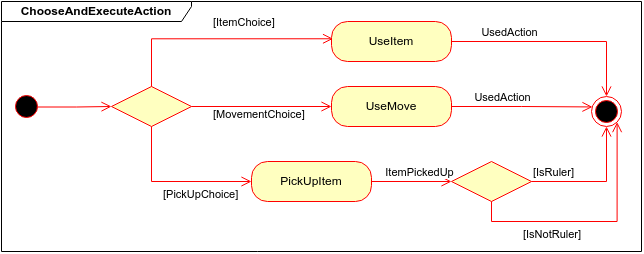
* + 1. TeacherRound



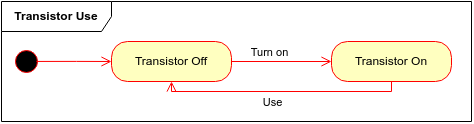
* + 1. PlayerRound



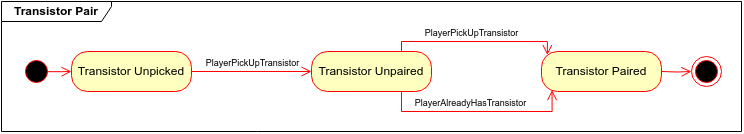
* + 1. ChooseAction



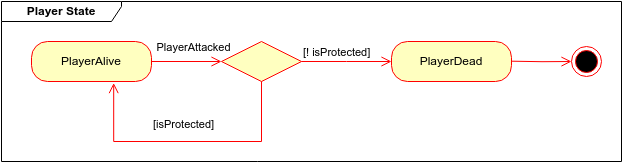
* + 1. TransistorUse



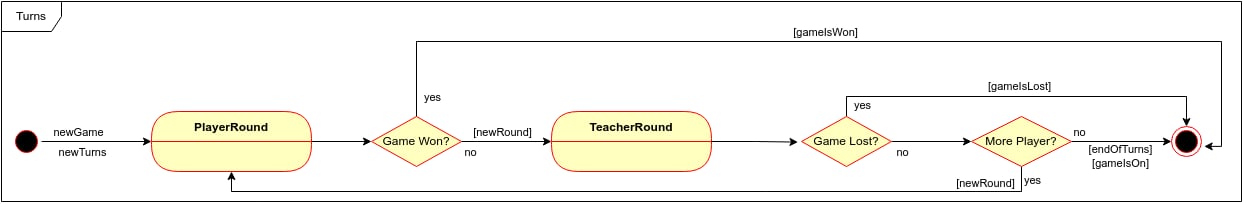
* + 1. TransistorPair



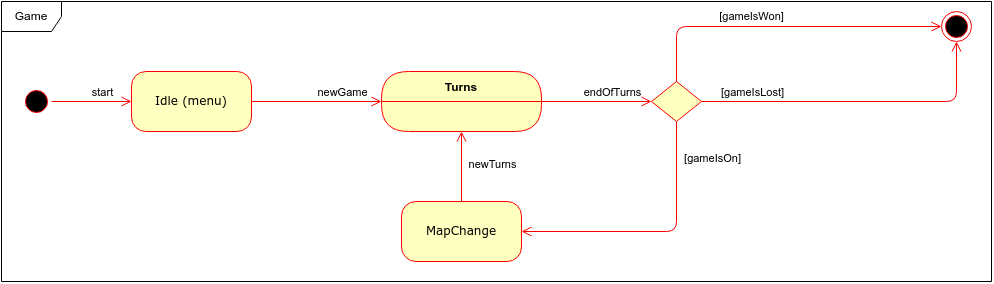
* + 1. PlayerState



* + 1. Turns



* + 1. Game



## ***Napló***

| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| --- | --- | --- | --- |
| 2024.02.26. 21:00 | 2,5 óra | Bogár  Jánovszki  Koleszár  Varjú  Zombori | Értekezlet.  Döntés: A program osztályoknak megállapítása, kapcsolataik mellett döntés |
| 2024.02.27. 12:00 | 1,5 óra | Bogár | Tevékenység: Bogár pontosítja az osztályokat, utánanéz tervezési mintáknak |
| 2024.02.27. 21:30 | 2 óra | Varjú  Koleszár | Tevékenység: UML osztálydiagram létrehozása és átbeszélése |
| 2024.02.28. 19:00 | 3 óra | Varjú  Koleszár  Bogár | Tevékenység: UML osztálydiagram pontosítása |
| 2024.02.29. 21:00 | 3 óra | Bogár  Jánovszki | Tevékenység: Szekvencia diagramok készítése. 1/2 |
| 2024.03.01. 23:00 | 3,5 óra | Jánovszki | Tevékenység: Szekvencia diagramok készítése és fixálása. 1.3/2 |
| 2024.03.02. 11:30 | 3 óra | Bogár  Jánovszki | Tevékenység: Szekvencia diagramok készítése 2/2 |
| 2024.03.02. 21:00 | 3 óra | Bogár | Tevékenység: Szekvencia diagramok formális javítása |
| 2024.03.03. 20:30 | 5 óra | Varjú  Koleszár | Tevékenység: UML osztálydiagram pontosítása és dokumentálása. |
| 2024.03.04. 08:00 | 1 óra | Varjú  Koleszár | Tevékenység: UML osztálydiagram pontosítása és dokumentálása. |
| 2024.03.04. 12:00 | 1 óra | Bogár  Zombori | Tevékenység: Zombori és Bogár pontosítja a State Machine-okat |