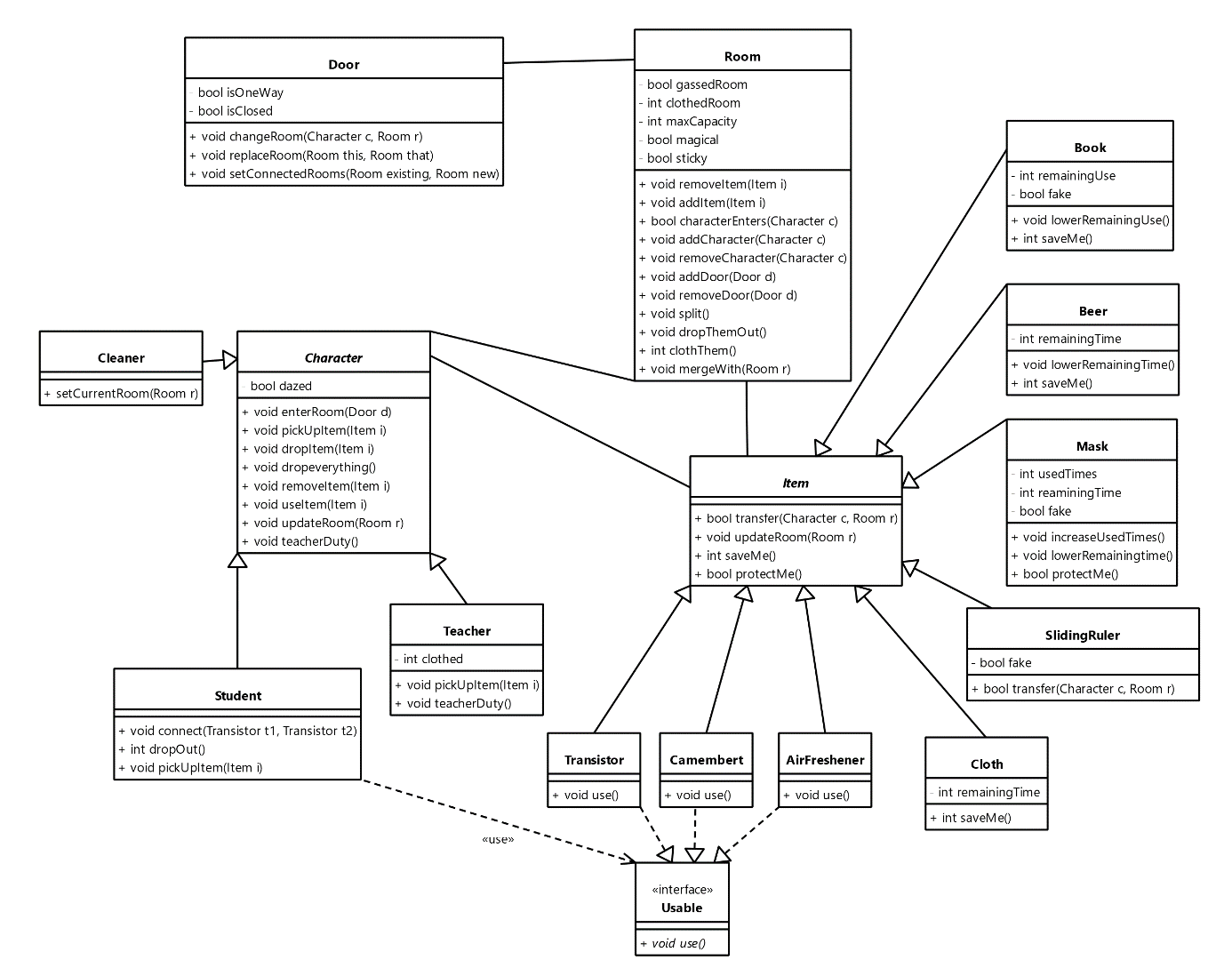
# Prototípus koncepciója

## Változás hatása a modellre

### Módosult osztálydiagram



### Új vagy megváltozó metódusok

#### AirFreshener

* **Felelősség**
  + A légfrissítőt megvalósító osztály.
* **Ősosztályok**
  + Item
* **Interfészek**
  + Usable
* **Asszociációk**
  + -
* **Attribútumok**
  + -
* **Metódusok**
  + **void use()**: Megszünteti a gázos szoba gázosságát.

#### Room

* **Attribútumok**
  + **bool sticky**: Azt nézi, hogy a szoba ragacsos-e.
* **Metódusok**
  + **void setSticky(bool b)**: Beállítja a szoba ragacsosságát.
  + **bool getSticky()**: Visszaadja, hogy a szoba ragacsos vagy sem.

#### Book

* **Attribútumok**
  + **bool fake**: Azt nézi, hogy a TVSZ az hamis vagy igazi.

#### Mask

* **Attribútumok**
  + **bool fake**: Azt nézi, hogy a TVSZ az hamis vagy igazi.

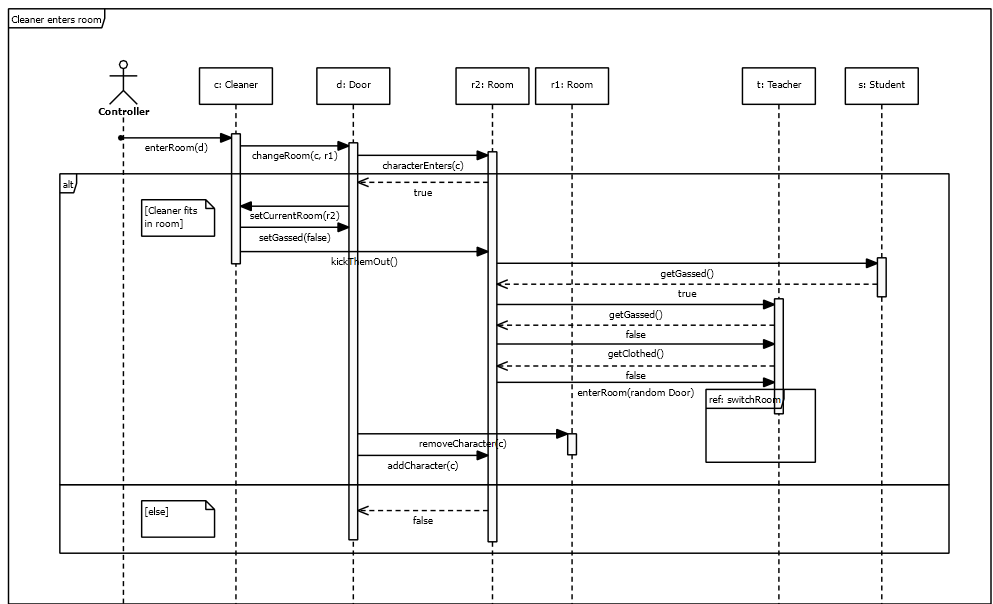
#### SlidingRuler

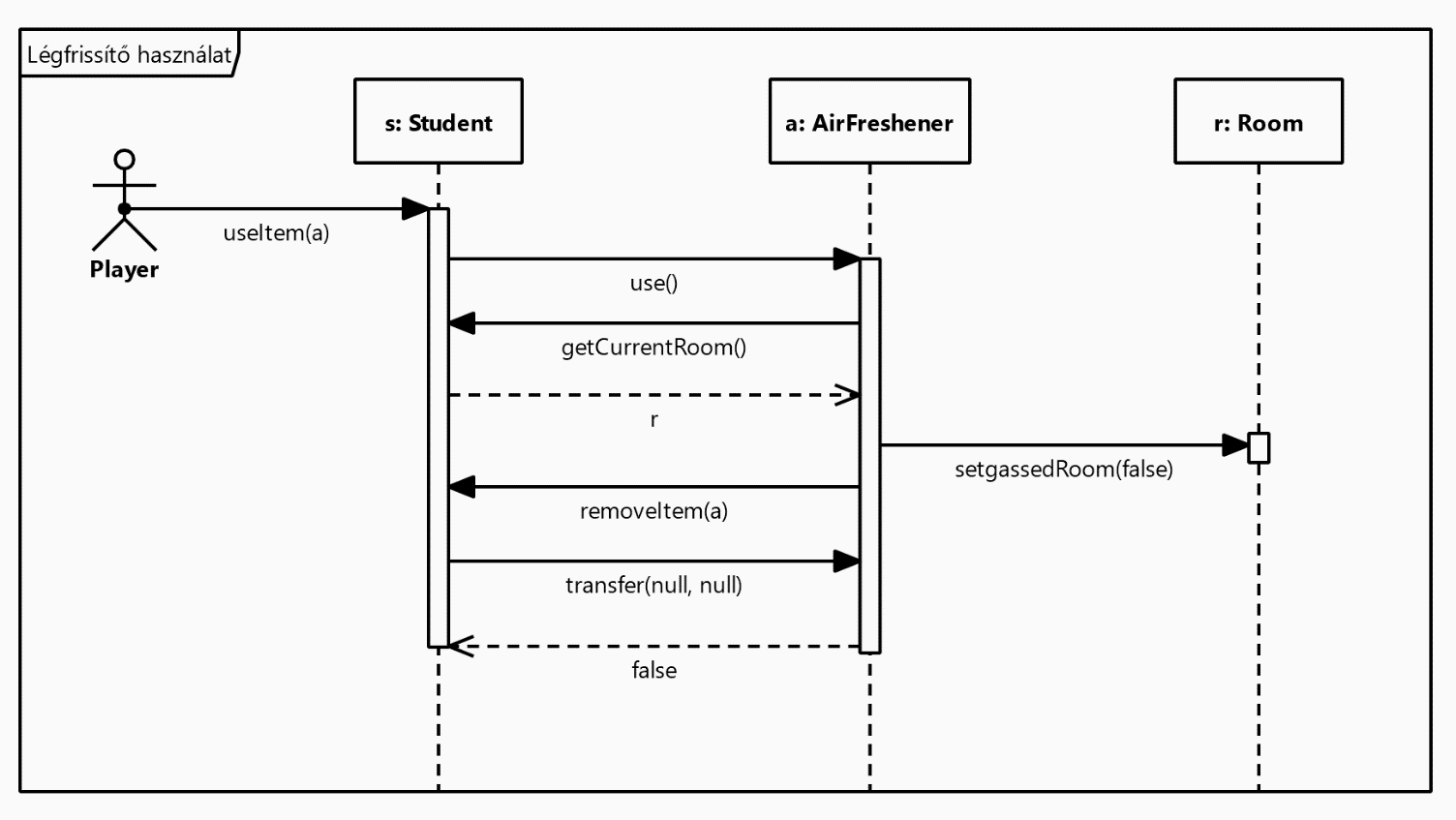
* **Attribútumok**
  + **bool fake**: Azt nézi, hogy a TVSZ az hamis vagy igazi.

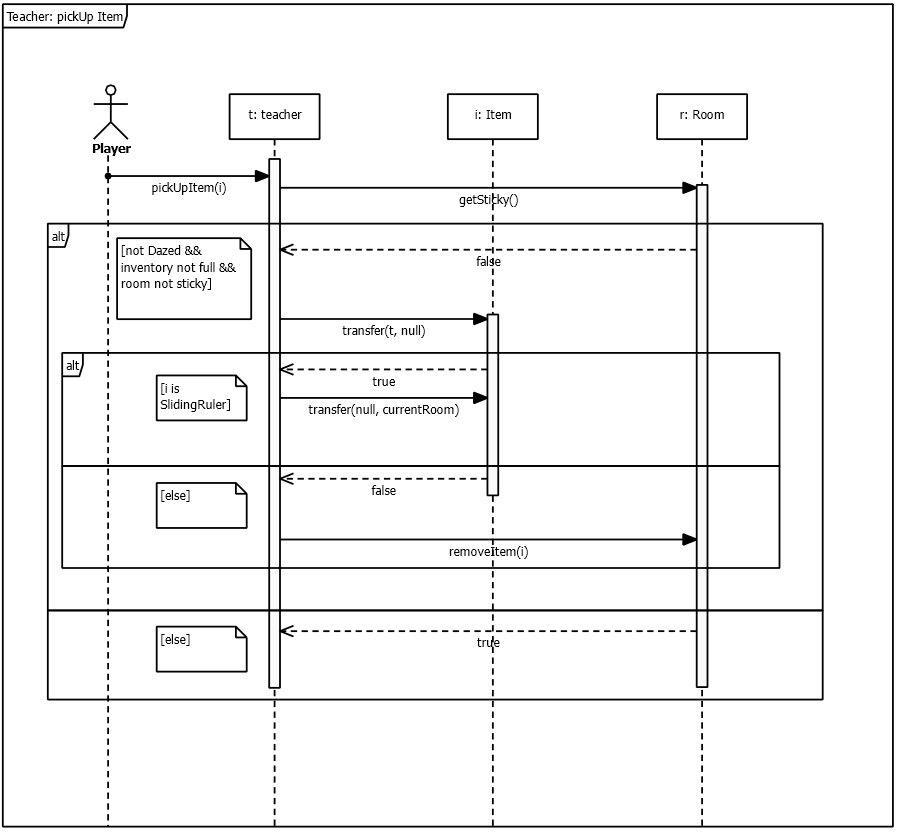
#### Cleaner

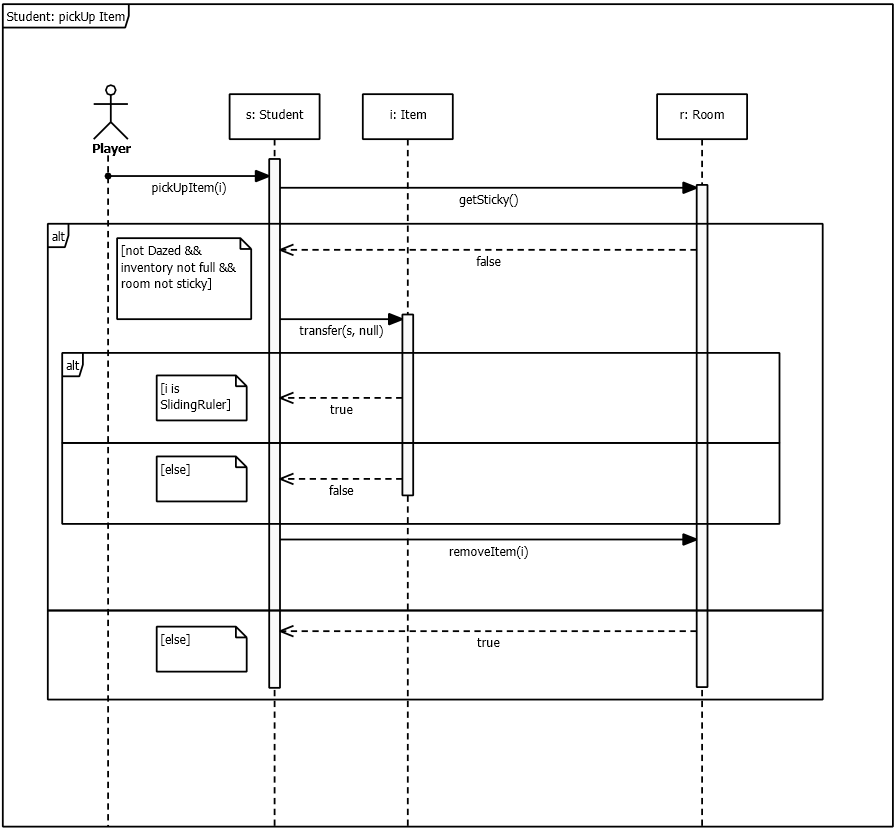
* **Felelősség**
  + A takarítót megvalósító osztály.
* **Ősosztályok**
  + Character
* **Interfészek**
  + -
* **Asszociációk**
  + -
* **Attribútumok**
  + -
* **Metódusok**
  + **void setCurrentRoom(Room r)**: Beállítja a takarító szobáját, kiszellőzteti a szobát és szól a szobának, hogy mindenkit tegyen ki.

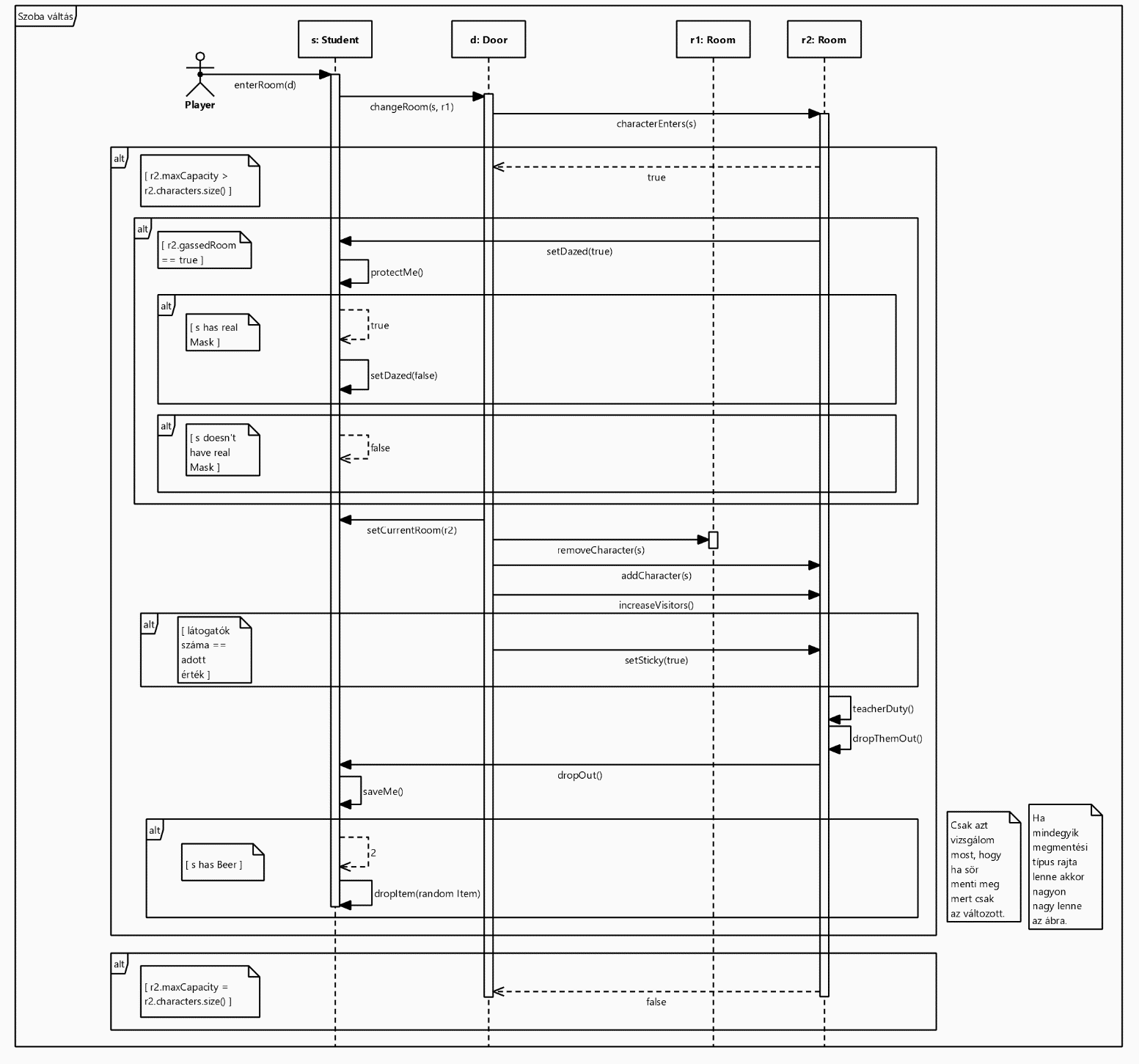
### Szekvencia-diagramok











A három tárgy, aminél lehet hamisítvány azoknál azért nem módosítottunk a szekvencia diagrammokon mert a meglévő függvények belsejében lesz csak egy if ág ami vizsgálja, hogy a tárgy hamis-e vagy sem. Emiatt nem látszódna a módosítás mert nem lenne felvéve új függvény vagy új válasz.

## Prototípus interface-definíciója

[Definiálni kell a teszteket leíró nyelvet. Külön figyelmet kell fordítani arra, hogy ha a rendszer véletlen elemeket is tartalmaz, akkor a véletlenszerűség ki-bekapcsolható legyen, és a program determinisztikusan is tesztelhető legyen.]

### Az interfész általános leírása

[A protó (karakteres) input és output felületeit úgy kell kialakítani, hogy az input fájlból is vehető legyen illetőleg az output fájlba menthető legyen, vagyis kommunikációra csak a szabványos be- és kimenet használható.]

### Bemeneti nyelv

[Definiálni kell a teszteket leíró nyelvet (szintakszis és szemantika). Külön figyelmet kell fordítani arra, hogy ha a rendszer véletlen elemeket is tartalmaz, akkor a véletlenszerűség ki-bekapcsolható legyen, és a program determinisztikusan is futtatható legyen. A szálkezelést is tesztelhető, irányítható módon kell megoldani. A programot egy adott konfigurációból is el kell tudni indítani, vagyis kell olyan parancs, amivel konkrét előre megadott állapotból indul a rendszer (pl. load).]

**Parancs1**

**Leírás:**

**Opciók:**

**Parancs2**

**Leírás:**

**Opciók:**

[Ha szükséges, meg kell adni a konfigurációs (pl. pályaképet megadó) fájlok nyelvtanát is.]

### Kimeneti nyelv

[Egyértelműen definiálni kell, hogy az egyes bemeneti parancsok végrehajtása után előálló állapot milyen formában jelenik meg a szabványos kimeneten. A program képes legyen olyan kimenetet előállítani, amellyel az objektumok állapota ellenőrizhető (pl. save). Ebben az alfejezetben is precízen definiálni kell, hogy a kimenet nyelve milyen elemekből és milyen szintakszissal áll elő.]

## Összes részletes use-case

[A use-case-eknek a részletezettsége feleljen meg a kezelői felületnek, azaz a felület elemeire kell hivatkozniuk a bemeneti nyelv parancsai alapján.

Alábbi táblázat minden use-case-hez külön-külön.]

|  |  |
| --- | --- |
| **Use-case neve** |  |
| **Rövid leírás** |  |
| **Aktorok** |  |
| **Forgatókönyv** |  |

## Tesztelési terv

[A tesztelési tervben definiálni kell, hogy a be- és kimeneti fájlok egybevetésével miként végezhető el a program tesztelése. Meg kell adni magas szintű teszt forgatókönyveket. Az egyes teszteket elég informálisan, szabad szövegként leírni, tesztesetenként egy-öt mondatban. Minden teszthez meg kell adni, hogy mi a célja, a proto mely funkcionalitását, osztályait stb. teszteli. Az alábbi táblázat minden teszt-esethez külön-külön elkészítendő.]

|  |  |
| --- | --- |
| **Teszt-eset neve** |  |
| **Rövid leírás** |  |
| **Teszt célja** |  |

## Tesztelést támogató segéd- és fordítóprogramok specifikálása

[Rövid bemutatással (elvárt funkcionalitás) specifikálni kell a tesztelést támogató segédprogramokat.]

## 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. |
| … | … | … | … |