# Logisztika

Szoftvertechnológia házi feladat

Szőke Tibor Ádám

GQ5E7S

# 1 A FELADAT LEÍRÁSA

A feladat egy logisztikai rendszer működésének megvalósítása. A futószalag rendszer felépülése után az irányítás átkerül a játékos kezébe, akinek lehetősége van egyesével állítani a váltók kimenetét. Egy váltónak véges számú bemenete és kimenete is lehet. A játékos által kiadott parancs hatására a váltó aktív kimenete (amiből 1 van) mindig a következő kimenetre fog mutatni, s ha a kimenetek listájának végére értünk, előről folytatjuk a választást.

A futószalag rendszer rendelkezik be- és kilépő pontokkal, ahol teherautók adják és kapják a szállítandó csomagokat. Minden csomagot jellemzi a színe, valamint az élettartama. Ha a csomag élettartama lejár (megromlik), akkor a csomag felrobban, és a játékos büntetést kap (-2 pont). Fontos megjegyezni, hogy nem minden csomag romlandó. Ezeket a jellemzőket véletlenszerűen kapják a csomagok, valamint az is véletlenszerű, hogy a belépési pontokhoz rendelt teherautók milyen gyakran helyeznek új csomagot a futószalagra.

A csomagokkal előfordulhat, hogy összetörnek. Ilyen esetben a játékos pontot veszít (-2 pont). Ez például akkor fordulhat elő, ha egy mezőre 2 csomag is érkezik, ekkor a korábban érkezett csomag összetörik, és csak a másik kerül a kérdéses mezőre. A másik opció, amikor a csomag megérkezik a rendszer egy kilépési pontjához, ahol teherautó hiányában a csomag a földre esik és összetörik. Abban az esetben, ha van teherautó, összevetjük a csomag színét a teherautó szállítási tervével.

Minden kilépési ponthoz rendelt teherautó rendelkezik egy szállítási tervvel, amely megadja, hogy az adott színből hány csomagot vár. Egy teherautó akár több színből is várhat csomagokat. Amennyiben a csomagkézbesítés sikeres (a szállítási terv szerint még szükségeltetett ilyen színű csomag), úgy a felhasználó pontjutalomban részesül (+1 pont) és csomag kikerül a rendszerből. Abban az esetben, ha a csomagkézbesítés sikertelen (az ott tartózkodó teherautó nem igényel olyan színű csomagot), a csomag szintén kikerül a rendszerből, azonban a játékost pontlevonással büntetjük (-1 pont). A teherautók szállítási terve véletlenszerű.

A futószalagon közlekedő csomagokkal a futószalag végéhez érve az alábbi esetek fordulhatnak elő:

- A futószalag vége egy kilépési pont, ilyenkor a fent említett vizsgálatokra van szükség.
- A futószalag vége egy olyan elem, mely akár több futószalagnak is része és egy aktív kimenete van. Ez esetben a csomag áthelyeződik erre az egyedüli kimeneti elemre.
- A futószalag vége egy váltó elem, melynek több kimenete van. Ilyenkor egy ütem várakozás után az épp aktív kimeneti elemre kerül a csomag.

# 2 FUNKCIONÁLIS KÖVETELMÉNYEK

## 2.1 ELSŐDLEGES KÖVETELMÉNYEK

| Azonosító | Leírás | Use-case |
|---|---|---|
| 01 | A futószalagrendszer kisebb futószalagokra bontható, melyeket láncolt listával valósítunk meg. A láncolt listát fordított menetirányban építjük fel, azaz első eleme a futószalag utolsó alkotóeleme és utolsó eleme a futószalag első alkotóeleme. | Init |
| 02 | Egy futószalag elem, egyszerre több futószalagnak is része lehet. Ezek az elemek a kimenet számától függően Cross, vagy Switch elemek. | Init |
| 03 | Cross elembe véges számú elem csatlakozhat.<br>(Több láncolt listának is lehet eleme.) | Init |
| 04 | Switch elembe szintén véges számú elem csatlakozhat, valamint véges számú elem következhet belőle. | Init |
| 05 | Switch elemből kimenő elemek közül mindig pontosan 1 aktív. | Init |
| 06 | End elembe mindig 1 elem csatlakozik és 0 elem következik belőle. | Init |
| 07 | Piece elembe csatlakozhat 1 vagy 0 elem (attól függően, hogy a futószalag első eleme-e) és mindig 1 elem következik belőle. | Init |
| 08 | Futószalag végén (láncolt lista elején) mindig Cross/Switch/End elem található. | Init |
| 09 | TruckIn-hoz bármilyen mezőt hozzárendelhetünk. | Init |
| 10 | TruckOut-ot csak End elemhez lehet rendelni. | Init |
| 11 | TruckOut szállítási terve véletlenszerűen generálódik létrehozáskor. | Init |
| 12 | A játékos a rendszer felépülése után kapja meg az irányítást, valamint a rendszer képét, és a konzolon keresztül vezérelheti a váltókat. | Init,<br>View,<br>Control Switch |
| 13 | A játékos képes állítani a váltók állapotát, s az állapotok sorrendje mindig ugyanúgy ismétlődik. Az utolsó állapot után pedig az első állapot következik. | Change Switch |
| 14 | A játékos folyamatosan látja a futószalag-rendszer állapotát. | View |
| 15 | Teherautókról időnként csomag kerülhet a rendszerbe. | Control Packages |
| 16 | Ha a teherautóhoz rendelt mezőn épp van csomag, akkor a teherautó nem generál oda egy új csomagot. | Control Packages |
| 17 | Két egymást követő ütemben nem kerülhet ugyanarról a teherautóról csomag a rendszerbe, mert a váltónál a korábbi csomag mindig összetörne (mert az vár ott egy ütemet). | Control Packages |
| 18 | Nem minden csomag romlandó. Ezek élettartamát negatív számokkal jelöljük. | Control Packages |
| 19 | Az új csomagok tulajdonságai véletlenszerűek. | Control Packages |
| 20 | Hogy mikor kerül be a rendszerbe új csomag, az is véletlenszerű. | Control Packages |
| 21 | Egy csomag minden ütemben 1 mezőt halad előre, kivéve a speciális eseteket. | Control Packages |
| 22 | Egy csomag minden ütemben veszít az élettartamát jelző értékből 1-et. | Control Packages |

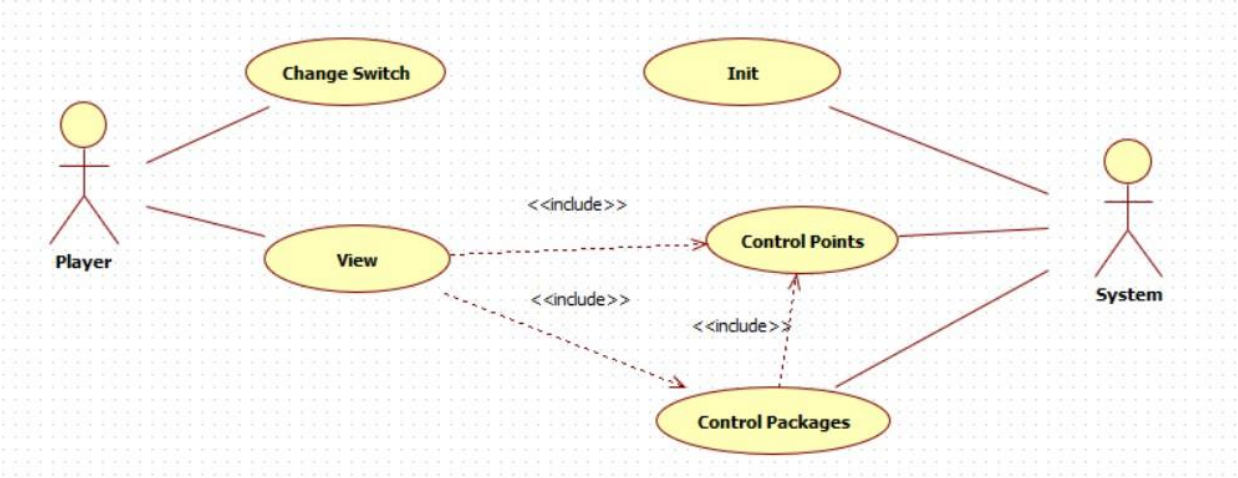
| | | |
|---|---|---|
| 23 | Ha a csomag élettartamának étéke 0, a csomag felrobban és a játékos 2 pontot veszít. | Control Packages, Control Points |
| 24 | Speciális eset, ha a csomag váltóra kerül. Mielőtt továbbhaladna, vár ott 1 ütemet. | Control Packages |
| 25 | A csomag a váltóról a váltó aktív kimenetén szereplő mezőre kerül. | Control Packages |
| 26 | Speciális eset, ha a csomag End mezőre kerül. Onnan vagy a földre esik, vagy ha van hozzárendelve teherautó, akkor arra. | Control Packages |
| 27 | Ha a földre esik, a csomag megsemmisül és a játékos 2 pontot veszít. | Control Packages, Control Points |
| 28 | Sikeres egy csomagkézbesítés, ha egy csomag a teherautóra kerül és a teherautó szállítási terve szerint szükség volt olyan színű csomagra. Minden sikeres csomagkézbesítés után a játékos 1 pontot kap. | Control Packages, Control Points |
| 29 | Hibás egy csomagkézbesítés, ha egy csomag a teherautóra kerül és a teherautó szállítási terve szerint nem volt szükség olyan színű csomagra. Minden hibás csomagkézbesítés után a játékos 1 pontot veszít. | Control Packages, Control Points |
| 30 | A csomag akkor is összetörik, hogyha 1 mezőre 2 csomag is kerülne. Ilyen esetben a korábban odakerült csomag összetörik és a későbbi csomag kerül a kérdéses mezőre. A játékos a csomag összetörése miatt veszít 2 pontot. | Control Packages, Control Points |

## 2.2 TOVÁBBI KÖVETELMÉNYEK

| **Azonosító** | **Leírás** | **Megjegyzés** |
|---|---|---|
| 31 | A futószalagrendszer létrejötte után változatlan marad (nem épül/törlődik eleme, csak mozog). | A rendszer létrehozását nem specifikáljuk, csak a rendszer üzemelését. |
| 32 | A játékos konzolon keresztül tudja változtatni a váltók aktuális kimenetét. | |

# 3 USE-CASE-EK

## 3.1 USE-CASE DIAGRAM




## 3.2 USE-CASE LEÍRÁSOK

| **Cím** | **Change Switch** |
|---|---|
| **Leírás** | A játékos képes állítani a rendszerben található váltókat. |
| **Aktorok** | Player |
| **Főforgatókönyv** | **1.** A játékos parancsot ad a váltásra, majd a parancsban szereplő azonosítójú váltó aktív kimenete a következő elem lesz. |

| **Cím** | **View** |
|---|---|
| **Leírás** | A játékos folyamatosan látja a logisztikai telep aktuális állapotát. |
| **Aktorok** | Player |

| **Cím** | **Init** |
|---|---|
| **Leírás** | A System létrehozza az egész rendszert alkotóelemenként, valamint beállítja a kapcsolatokat a megfelelő rendszeralkotók között. Miután elindította a rendszer működéséhez szükséges időzítőt, átadja a vezérlést a Playernek. |
| **Aktorok** | System |
| **Főforgatókönyv** | *A rendszer kiépítésének megvalósításával nem foglalkozunk, csak a működés megvalósításával. Az osztályok leírásában ezeket a függvényeket dőlt betűtípussal jeleztem.* |

| **Cím** | **Control Points** |
|---|---|
| **Leírás** | A csomagok sorsának függvényében a Játékos pontokat kap/veszít. |
| **Aktorok** | System |
| **Főforgatókönyv** | **1.** Helyes csomagkézbesítés történik (1 pont). |
| **Alternatív forgatókönyv** | **2.** Hibás csomagkézbesítés történik (-1 pont). |
| **Alternatív forgatókönyv** | **3.** A csomag összetörik/felrobban (-2 pont). |

| Cím | **Control Packages** | |
|---|---|---|
| **Leírás** | A csomagok létrehozása, majd továbbítása egészen a belépési ponttól a kilépési pontig, valamint a csomag tulajdonságainak vizsgálata, azok következményében való eljárások. | |
| **Aktorok** | System | |
| **Főforgatókönyv** | **1.** | Létrejön egy csomag és felkerül egy futószalag elemre. |
| **Alternatív forgatókönyvek** | **1.A.1.** | A csomag élettartama lejár és felrobban a csomag. |
| | **1.B.** | A csomag élettartama nem jár le. |
| | **1.B.A.1.** | A következő mező üres, odakerül a csomag. |
| | **1.B.B.1.** | A következő mező nem üres, úgy kerül oda a csomag, hogy a már ott levőt összetöri. |
| | **1.B.A/B.1.A.1.** | Ez a mező Cross/Piece, a csomag továbbhalad a megfelelő mezőre. |
| | **1.B.A/B.1.B.** | Ez a mező Switch mező. |
| | **1.B.A/B.1.B.1.** | A csomag vár itt 1 ütemet. |
| | **1.B.A/B.1.B.2.** | A csomag továbbhalad a következő mezőre. |
| | **1.B.A/B.1.C.** | Ez a mező End mező. |
| | **1.B.A/B.1.C.A.1.** | Nincs hozzárendelve teherautó, a csomag leesik és széttörik. |
| | **1.B.A/B.1.C.B.** | Van hozzárendelve teherautó |
| | **1.B.A/B.1.C.B.A.1.** | A csomag színe megfelel, sikeres kézbesítés megy végbe, a játékos pontot kap. |
| | **1.B.A/B.1.C.B.B.1.** | A csomag színe nem megfelelő, sikertelen a kézbesítés, a játékos pontot veszít. |

*Megjegyzés: Az „A/B” jelölést azért vezettem be, mert lényegtelen a csomag célba jutásának szempontjából, hogy másik csomag összetörésével / másik csomag összetörése nélkül jutott el a teherautóig. A kérdéses eset után mindkét forgatókönyv megegyezik.*

# 4 STRUKTURÁLIS LEÍRÁS

## 4.1 AZ OSZTÁLYOK LEÍRÁSA

### 4.1.1 SystemInit

**Felelősségek**

A rendszer létrehozása alkotóelemenként, kapcsolatok beállítása a megfelelő rendszeralkotók között. Az időzítő működtetése, valamint az irányítás átadása a felhasználónak.

**Metódusok**

| | |
|---|---|
| +main(args: String): void | A felelősségeknek megfelelő működést hajtja végre. Létrehoz egy Timer, Controller és egy BeltCollection példányt. Miután elindította az időzítőt átadja a vezérlést a játékosnak. |

### 4.1.2 Controller

**Felelősségek**

Kapcsolatteremtés a rendszer és a játékos között. Játékos által kiadott parancsok beolvasása, pontműveletek végrehajtása, azok közlése a játékossal.

**Attribútumok**

| | |
|---|---|
| -points: int | Játékos pontjainak nyilvántartása. |
| -switches: Switch[0..*] | A rendszerben lévő Switchek listája. A Switch azonosítója megegyezik azzal, hogy a tömbben hányadik helyen szerepel. |

**Metódusok**

| | |
|---|---|
| +start(): void | A rendszer inicializálása után a felhasználó kezébe kerül az irányítás. |
| +readCmd(): int | A játék futása során parancsokat olvas be a felhasználótól, azaz hogy mely váltó állapotát szeretné állítani. Visszatér a változtatandó Switch azonosítójával. |
| +addPoint(): void | Sikeres csomagkézbesítés esetén pontszámnövelés esedékes. |
| +reducePoint(penalty: int): void | Csomag összetörése, valamint hibás csomagkézbesítés esetén pontlevonás következik be. A pontlevonás (paraméter) értéke 1, amennyiben hibás csomagkézbesítés történt és az értéke 2, amennyiben a csomag összetört/felrobbant. |
| *+addSwitch(sw: Switch): void* | *Bővíti a switches listát egy új elemmel.* |

### 4.1.3 Timer

**Felelősségek**

A rendszer folyamatos működéséért felel (a szalag folytonos mozgása, valamint a csomagok időnkénti generálása).

**Metódusok**

| | |
|---|---|
| +doAllSteps(): void | Rendszeres időközönként meghívja az összes Steppable interfészt megvalósító osztály példányainak step() metódusát. |

### 4.1.4 Package

**Felelősségek**

Minden csomag létrejöttekor beállítódik (véletlenszerűen) a csomag romlandósági ideje, valamint a csomag színe.

**Attribútumok**

| | |
|---|---|
| -life: int | A csomag élettartamára vonatkozó számadat. Amennyiben a csomag nem romlandó, létrejöttekor -1-et kap értékül. |
| -color: int | Véletlenszerűen generált szín (számként reprezentálva), mely a csomagot jellemzi. |

**Metódusok**

| | |
|---|---|
| +expire(): bool | Minden híváskor csökkenti a life attribútum értékét 1-gyel. Visszatérési értéke true, ha life értéke 0, minden más esetben pedig false. Nem romlandó csomagok esetén soha nem lehet 0 az élettartam (mindig negatív szám lesz az érték). |
| +getColor(): int | Megadja a csomag színét (melyet számokkal jelölünk, pl.: 1-es szín, 4-es szín, stb..). |

### 4.1.5 BeltCollection

**Felelősségek**

Egy tároló osztály, mely tartalmazza az összes láncolt lista kezdő elemét (a futószalagok végét). A Timer osztály doAllSteps() metódusa ennek segítségével halad végig az összes Steppable interfészt megvalósító osztály példányain.

**Attribútumok**

| | |
|---|---|
| -belts: Piece[0..*] | A futószalagok utolsó elemeinek listája.<br>(Azért hátulról épül fel a láncolt lista, mert hatékonyabb, ha hátulról végighaladva vesszük át a csomagokat, ahelyett, hogy elölről indulva adogatánk tovább. Így 2 szomszédos csomag tud együtt haladni anélkül, hogy összetörne az első.) |

**Metódusok**

| | |
|---|---|
| *+addNewBelt(startingPiece: Piece): void* | *Új (paraméterként megadott) futószalagot ad hozzá a tárolóhoz. (Pontosabban a láncolt lista első elemét).* |

### 4.1.6 Steppable

**Felelősségek**

Interface, melyet azon osztályok valósítanak meg, melyeknek step() függvényét az időzítő bizonyos időközönként meghív. Megvalósító osztályok: Piece, Cross, Switch, TruckIn

**Metódusok**

| | |
|---|---|
| +step(): void | Osztályonként változó az implementáció. |

### 4.1.7 Piece

**Felelősségek**

A futószalagot alkotó elem. A futószalagot egy láncolt lista valósítja meg, mely a futószalag menetirányával ellentétes irányban tárolja az elemeket.

**Attribútumok**

| | |
|---|---|
| -next: Piece | Az elemet követő elem a futószalagon. A futószalag működése során a csomag erre a mezőre fog kerülni (feltéve, ha az értéke nem NULL). |
| -pack: Package | Minden mező tartalmazhat csomagot, amennyiben nem, az értéke NULL. |

**Metódusok**

| | |
|---|---|
| +step(): void | Ha a mező tartalmaz csomagot, meghívjuk az expire() függvényt.<br>• Amennyiben ez a metódus true értékkel tért vissza, úgy meghívjuk a reducePoint(2) függvényt majd eltávolítjuk erről a mezőről a csomagot.<br>• Amennyiben ez a metódus false értékkel tért vissza, úgy meghívjuk a következő elem recievePack(p: Package) függvényét és eltávolítjuk erről a mezőről csomagot. |
| +hasPack(): boolean | True-val tér vissza amennyiben az adott mezőn található csomag. |
| +setPack(p: Package): void | Beállítja a paraméterként kapott csomagot a mező pack attribútumának. |
| +getPack(): Package | Visszatérési értéke az adott mezőn lévő csomag. |
| +removePack(): void | Eltávolítja a mezőn található csomagot (NULL értéket állít be). |
| +recievePack(p: Package): void | Ha az adott mezőn épp van csomag és úgy hívódik meg a függvény, úgy a játékostól levonunk 2 pontot és beállítjuk az új, paraméterként kapott csomagot a mezőre.<br>Ha az adott mezőn nincs csomag a metódushívás pillanatában, úgy szimplán beállítjuk a paraméterként kapott csomagot a mezőre. |

### 4.1.8 Cross

**Felelősségek**

Azon mezők osztálya, melyek egyszerre több (legalább 2) futószalagnak részét alkotják és csak 1 kimenettel rendelkeznek. Fontos, hogy futószalag végén helyezkedhet csak el.

**Attribútumok**

| | |
|---|---|
| -next: Piece | Az az elem az értéke, ahová a Cross mező továbbadja a csomagot. |
| -stepdone: boolean | Ez az attribútum arra szolgál, hogy egy ütemben 1 elem ne adhasson tovább kétszer elemet. (Több láncolt listának is eleme, de csak egyszer hajthatja végre a step() metódusban leírtakat.) |

**Metódusok**

| | |
|---|---|
| +step(): void | Ha a mező tartalmaz csomagot és a stepdone értéke false, meghívjuk az expire() függvényt.<br>• Amennyiben ez a metódus true értékkel tért vissza, úgy meghívjuk a reducePoint(2) függvényt.<br>• Amennyiben ez a metódus false értékkel tért vissza, úgy meghívjuk a következő elem recievePack(p: Package) függvényét.<br>Ezek után eltávolítjuk erről a mezőről a csomagot. |

### 4.1.9 Switch

**Felelősségek**

Ez az osztály hasonlít a Cross osztályra, azonban ez az osztály „okosabb”. Annyiban különbözik, hogy több kimenete is van és az aktív kimenetet a játékos változtatni tudja. Továbbá abban is eltér, hogy minden Switch mezőre érkezett csomag vár 1 időegységet ezen a mezőn, mielőtt tovább haladna.

**Attribútumok**

| | |
|---|---|
| -id: int | Ez az egyedi azonosító segít abban, hogy tudjuk melyik Switchre vonatkozik a játékos által kiadott parancs. |
| -wait: boolean | A felelősségként leírt viselkedés megvalósítására szolgáló változó. Alapértelmezetten true értéket képvisel. |
| -next: Piece | A váltó aktív kimenete. Értéke a nextPiecesből az egyik elem. |
| -nextPieces: Piece[2..*] | Annyi (tetszőleges, de legalább 2) elemből álló lista, ahány kimenete van a Switchnek. |

**Metódusok**

| | |
|---|---|
| +step(): void | Ha a mező tartalmaz csomagot és a stepdone értéke false, meghívjuk az expire() függvényt. Amennyiben ez a metódus true értékkel tért vissza, úgy meghívjuk a reducePoint(2) függvényt majd eltávolítjuk erről a mezőről a csomagot.<br>Amennyiben ez a metódus false értékkel tért vissza, úgy két esetet szükséges vizsgálnunk:<br>• Ha a wait értéke false, meghívjuk az aktív kimeneten (next attribútumban) szereplő elem recievePack(p: Package) metódusát. Eltávolítjuk a váltóról a csomagot, majd átváltjuk a wait értékét true-ra.<br>• Ha pedig a wait értéke true, semmi mást nem csinálunk, mint átállítjuk a wait értékét false-ra.<br>*Megjegyzés: Ha 1 csomag a várakozási ütemében megsemmisül, mert ráesett egy másik, akkor az új csomagnak nem kell még egy plusz ütemet várnia.* |
| +changeNext(id: int): void | Meghívása esetén a paraméterként kapott azonosítójú váltó next attribútum értéke a nextPieces listában szereplő következő elem lesz. A listaelemek egy kört alkotnak, így az utolsó elem után az aktív kimenet a lista első eleme lesz. |
| +changeWait(): void | Megváltoztatja a wait attribútum értékét. |
| +getId(): int | Lekérdezi a switch azonosítóját. |

#### 4.1.10 End

**Felelősségek**

A futószalag vége, next attribútuma mindig NULL értékű. A step() metódus hívásának következtében az elemen lévő csomag minden esetben kikerül a rendszerből, azonban ennek több lehetősége is fennáll.

**Attribútumok**

| | |
|---|---|
| -trout: TruckOut | Az End elemhez rendelt teherautó. Ha nincs, értéke NULL. |

**Metódusok**

| | |
|---|---|
| +step(): void | Ha a mezőn található csomag, meghívjuk a csomag expire() metódusát.<br>• Ha a mezőhöz van hozzárendelve teherautó, és a csomag nem járt le, úgy meghívjuk a teherautó recievePack(p: Package) függvényét.<br>• Ha ebből a két feltétel közül legalább az egyik nem teljesül, akkor a csomag vagy összetörik, vagy felrobban, így meg kell hívnunk a reducePoint(2) metódust.<br>Ezek után eltávolítjuk a mezőről a csomagot. |
| +hasTruck(): boolean | Megtudhatjuk, hogy van-e a mezőhöz teherautó rendelve. |
| *+setTruck(t: TruckOut): void* | *E függvény segítségével rendelhetünk teherautót a mezőhöz.* |
| *+removeTruck(): void* | *Vagy akár el is távolíthatjuk, ekkor a trout értéke NULL lesz.* |

### 4.1.11 TruckIn

**Felelősségek**

Véletlenszerű csomaggenerálást valósít meg. Bármilyen futószalag elemhez rendelhető.

**Attribútumok**

| | |
|---|---|
| -entry: Piece | Az a mező ahová a csomagokat generálja (lerakja). |
| -ready: boolean | Arra szolgál, hogy két egymást követő ütemben ne rakjon a teherautó ugyanarra a mezőre csomagot, mert a korábban létrejött csomag mindenképp összetörne az első adandó váltónál. |

**Metódusok**

| | |
|---|---|
| +step(): void | Amennyiben a ready értéke true, valamint a teherautóhoz rendelt mezőn épp nincs csomag, lefut egy véletlenszerű számgenerálás. Ha a kapott szám eleget tesz egy bizonyos feltételnek, meghívódik a createPack() függvény. Ha a ready értéke false, csomag létrehozása nélkül beállítja a ready értékét true-ra. |
| +createPack(): Package | Hívásakor, véletlenszerű adatokkal (romlandó-e és ha igen mennyi ideig jó + milyen színű) létrehoz egy csomagot, és hozzárendeli az entry elemhez. Ezután beállítja a ready értékét false-ra. |
| +changeReady(): void | Megváltoztatja a ready értékét. |
| *+setEntry(p: Piece): void* | *Megadhatjuk a teherautóhoz tartozó belépési elemet.* |
| *+removeEntry(): void* | *Törölhetjük is a belépési elemet. (Értéke NULL lesz.)* |

### 4.1.12 TruckOut

**Felelősségek**

Teljesen más feladattal rendelkezik, mint a TruckIn. Minden TruckOut (kimenetnél álló teherautó) rendelkezik egy szállítási tervvel, ami megadja, hogy milyen színű csomagokból mennyit vár.

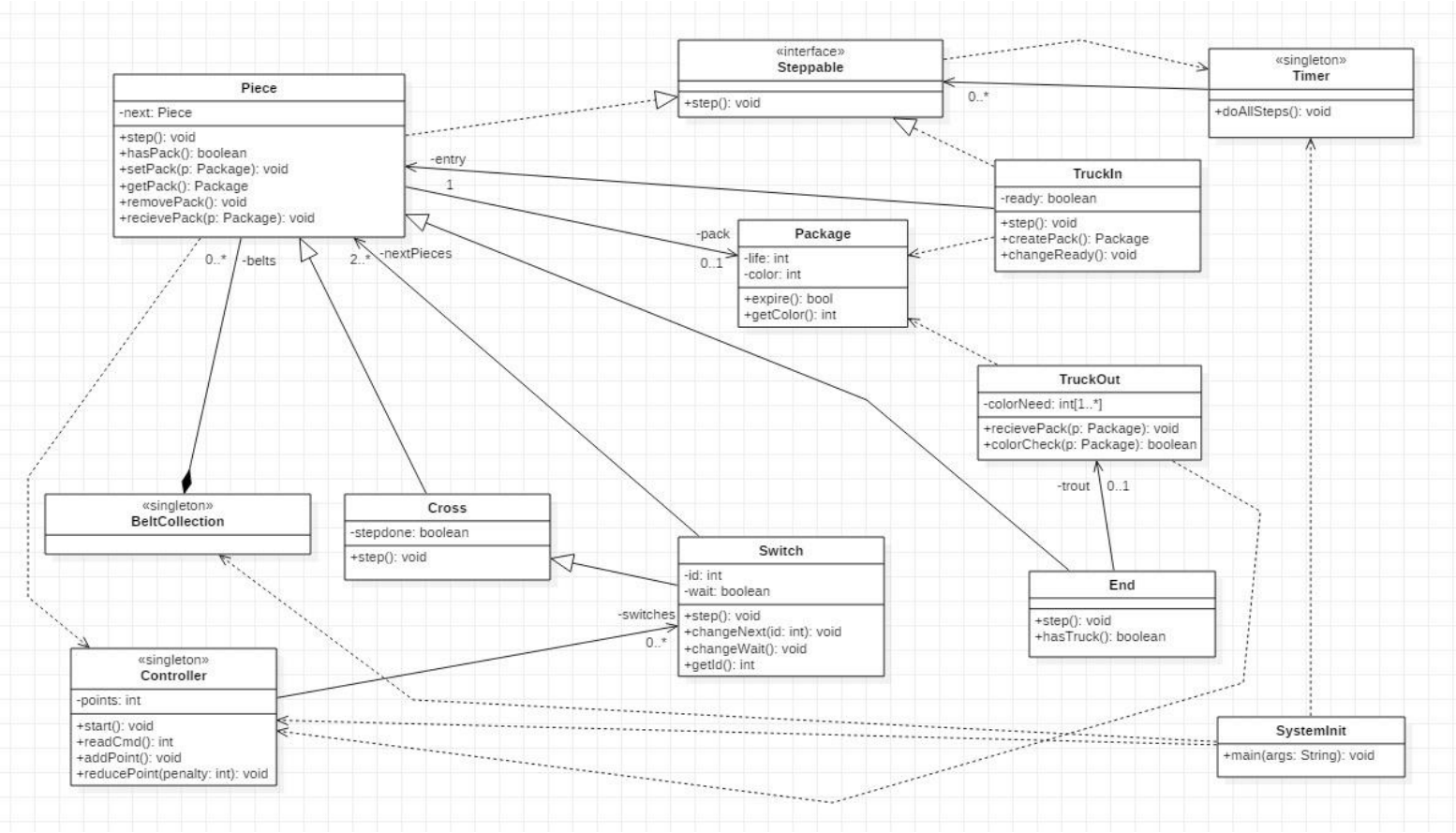
**Attribútumok**

| | |
|---|---|
| -colorNeed: int[1..*] | A szállítási terv megvalósítása. A tömb indexe jelenti a számot, a tömb értéke pedig azt, hogy abból a színből még mennyit vár a teherautó. (Például colodNeed[3] = 5 azt jelenti, hogy még 5 darab 3-mas színű csomagra vár a sofőr.) |

**Metódusok**

| | |
|---|---|
| +recievePack(p: Package): void | Meghívja a colorCheck(p: Package) függvényt.<br>• Ha a visszatérési érték true, a játékos kap 1 pontot, valamint a colorNeed megfelelő tömbelemét csökkentjük egyel.<br>• Ha a visszatérési érték false, a játékos veszít egy pontot. |
| +colorCheck(p: Package): boolean | A paraméterként kapott csomag színét veti össze a teherautó igényeivel. Ha van ilyen színű csomagra igény true, ha nincs, akkor pedig false értékkel tér vissza. |

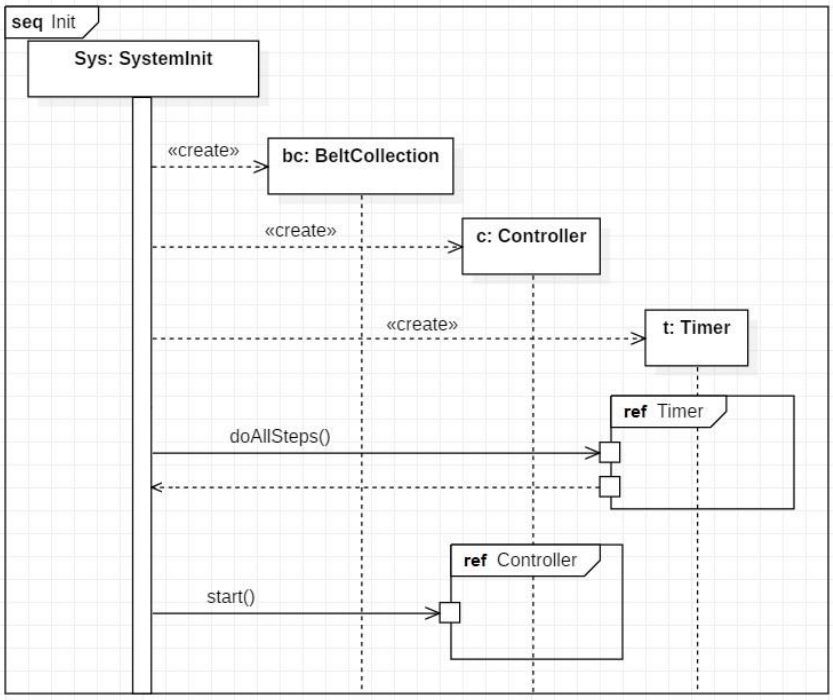
## 4.2 OSZTÁLYDIAGRAM



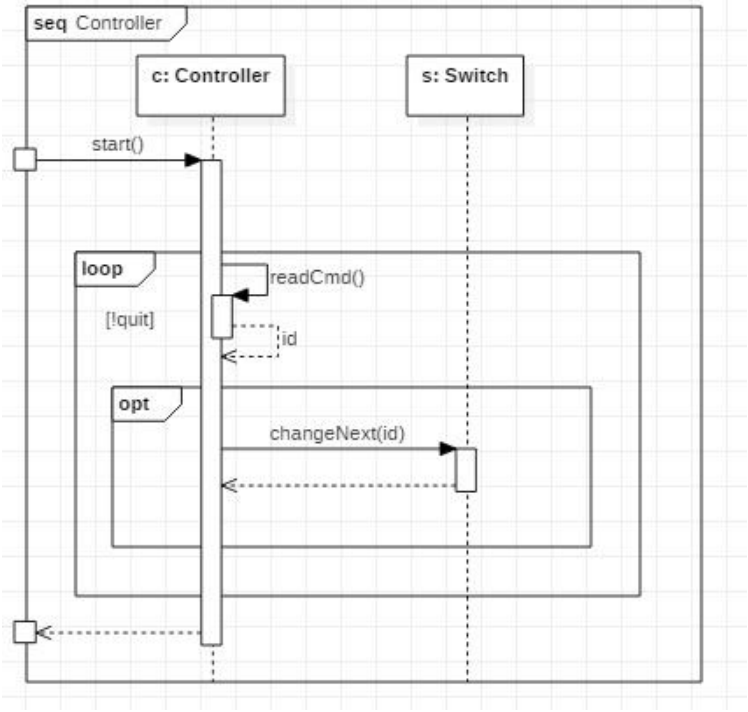
# 5 VISELKEDÉS LEÍRÁSA

## 5.1 SZEKVENCIA DIAGRAMOK

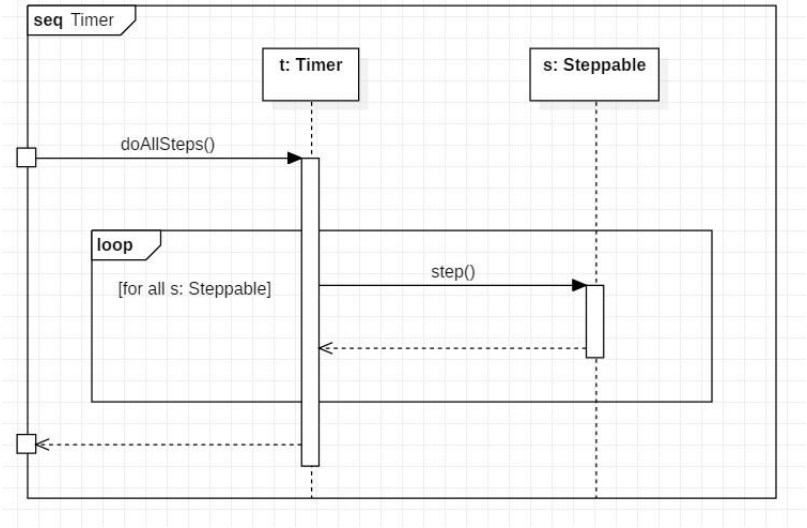
### 5.1.1 Init szekvenciája



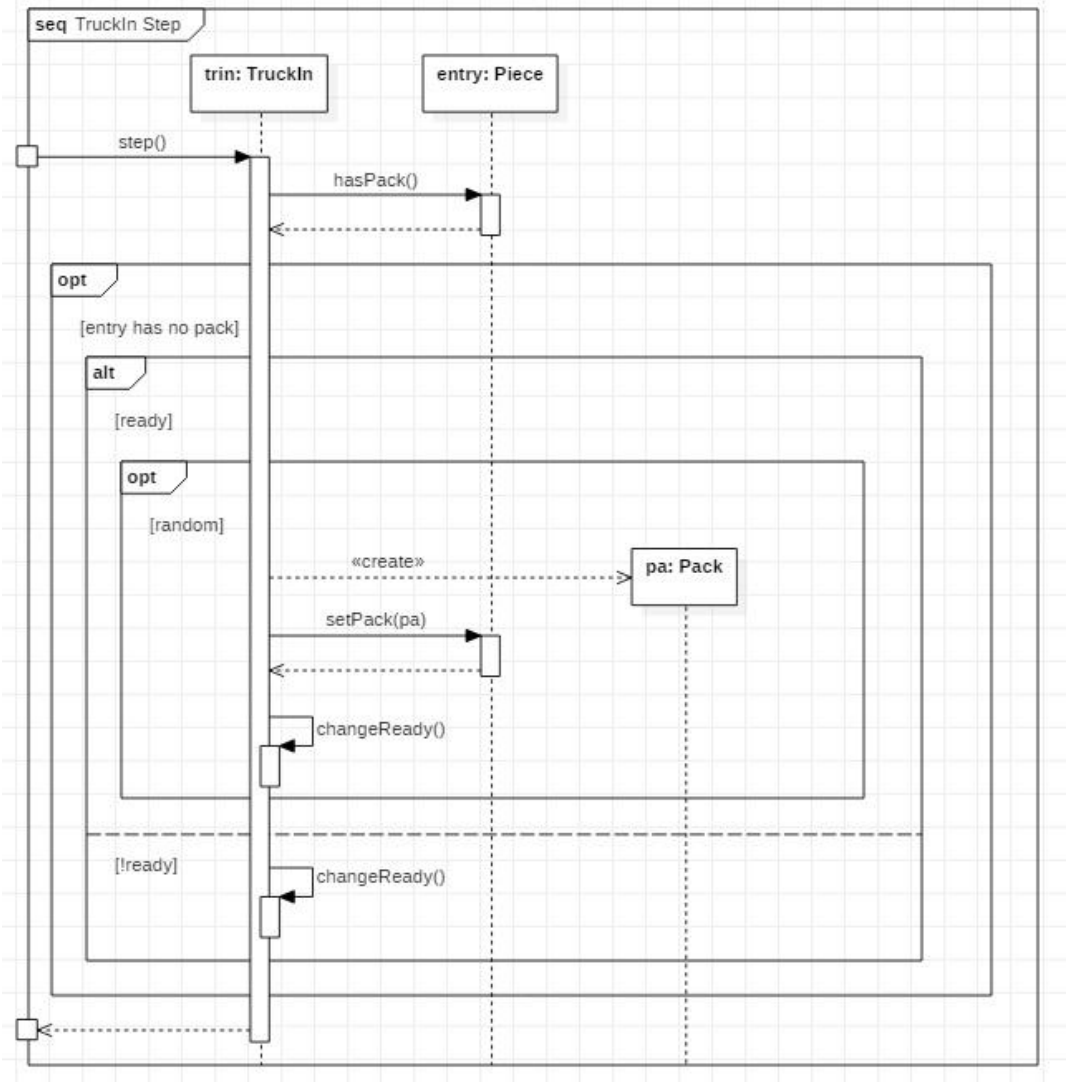

### 5.1.2 Controller szekvenciája



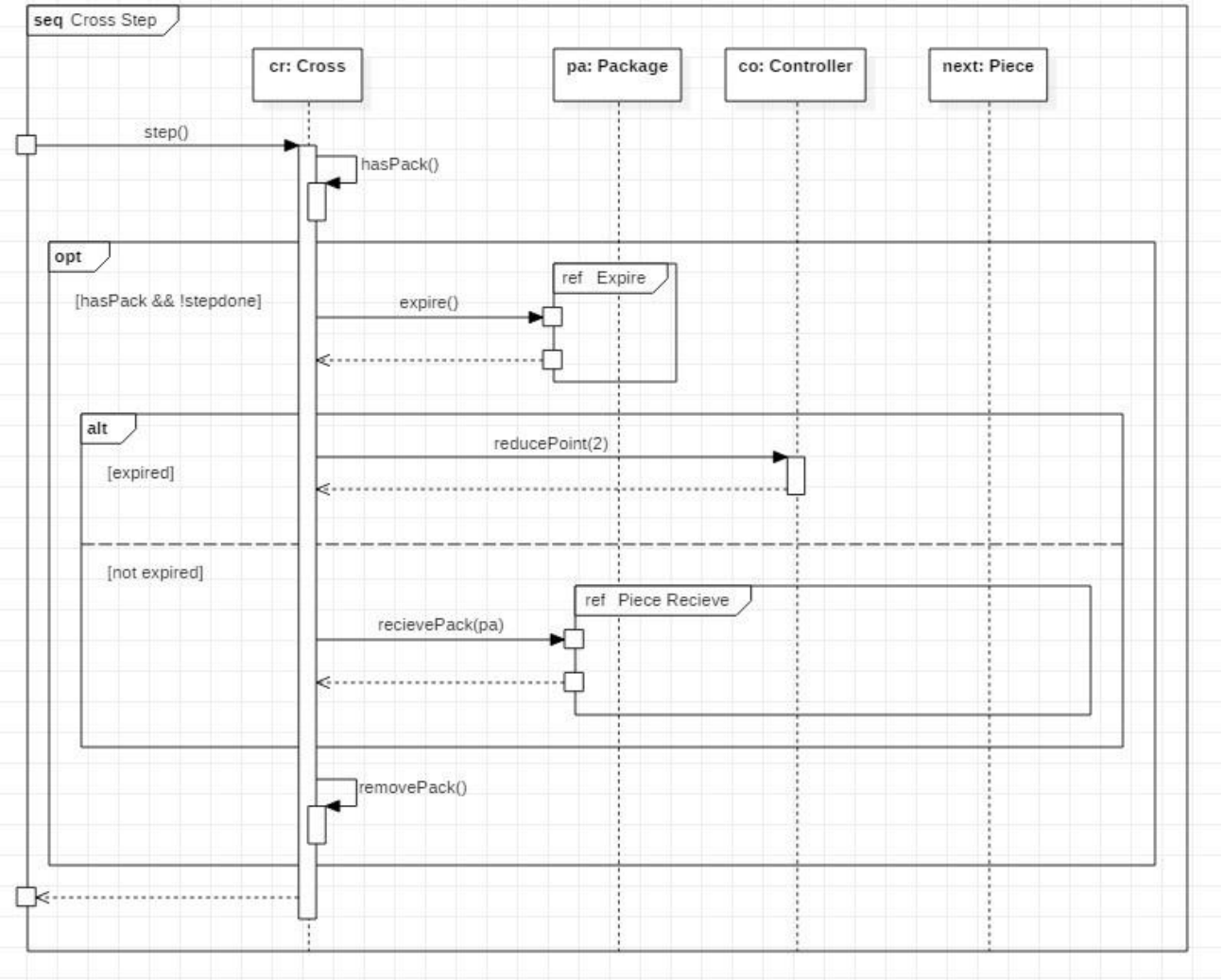
### 5.1.3 Timer szekvenciája



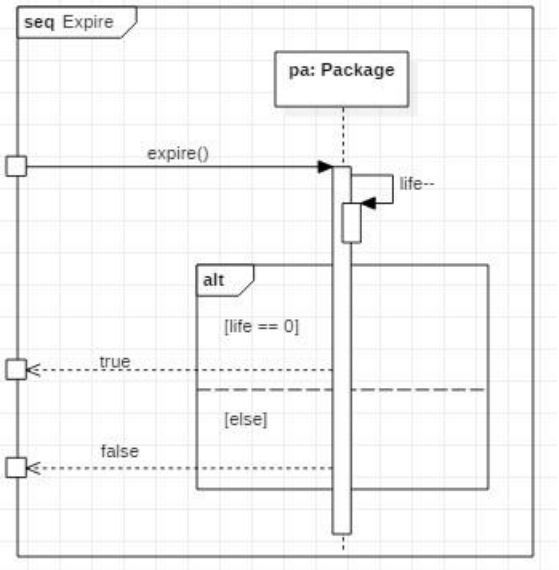

### 5.1.4 TruckIn Step szekvenciája



### 5.1.5 Cross Step szekvenciája

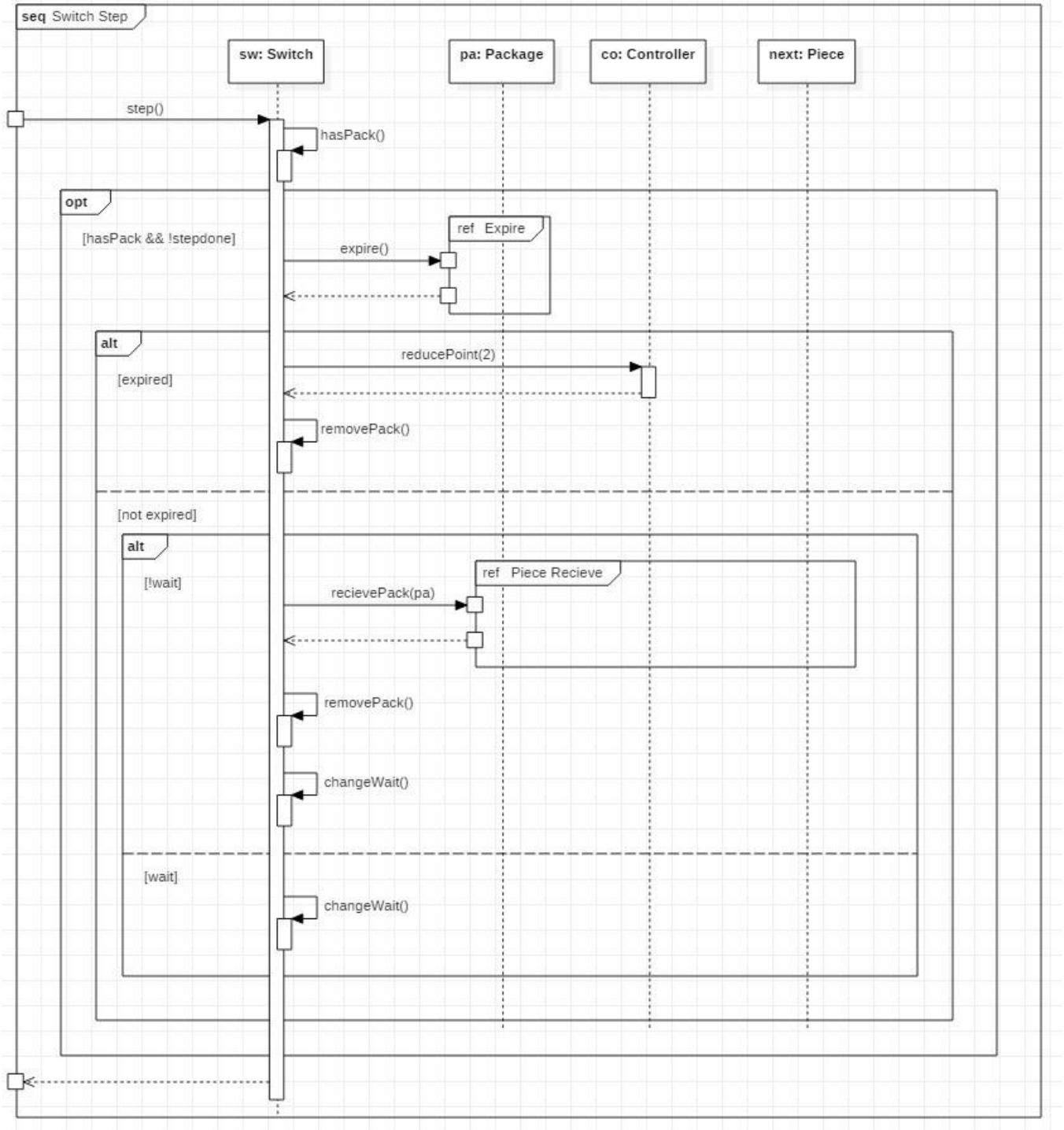



### 5.1.6 Expire szekvenciája

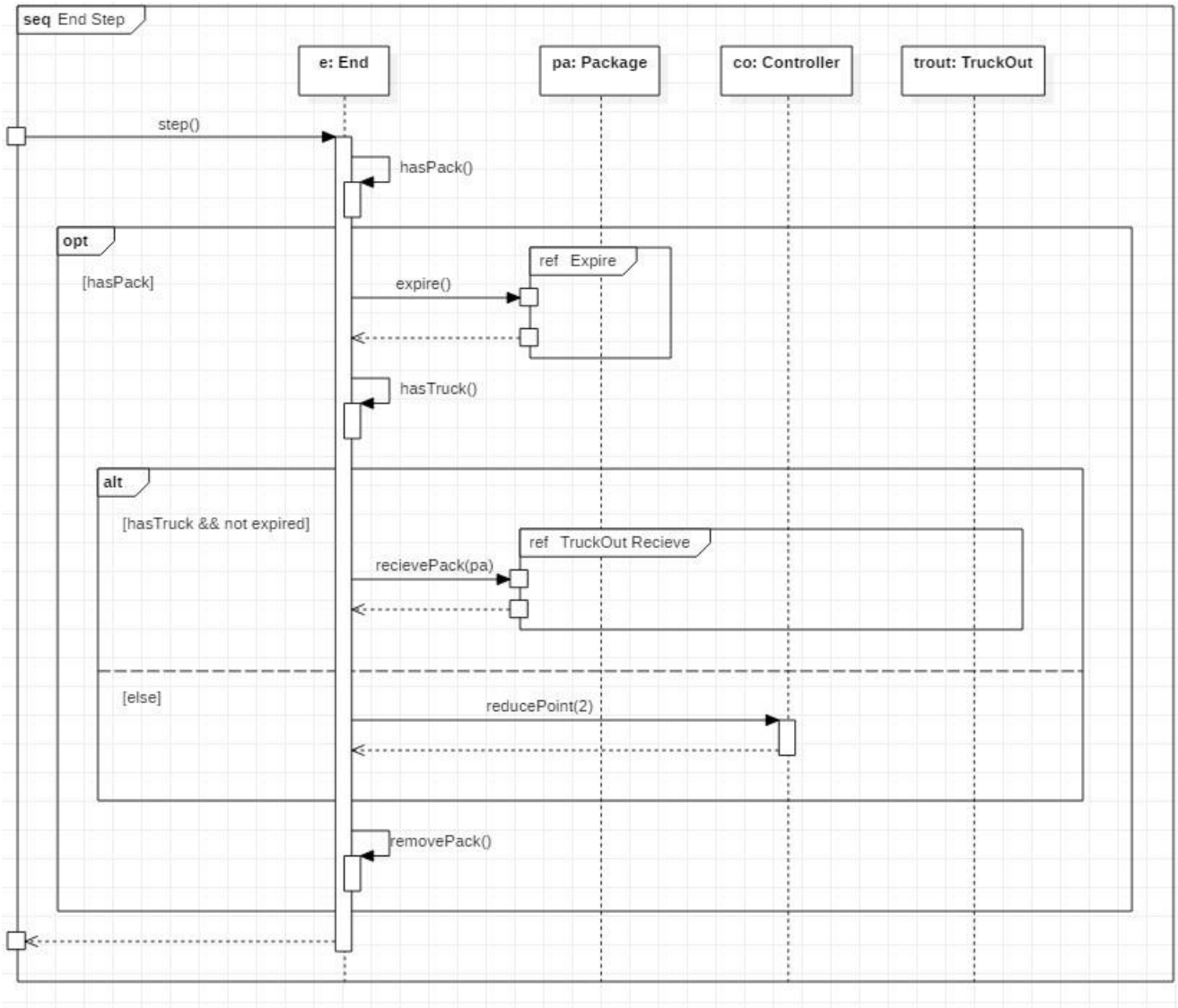



*Megjegyzés: Piece Step szekvencia diagramot azért nem készítettem, mert a Cross Step szekvencia diagramja teljes egészében magában hordozza azt. Csupán annyi különbség van, hogy a Cross Step esetében vizsgálni kell a !stepdone kritériumot is.*

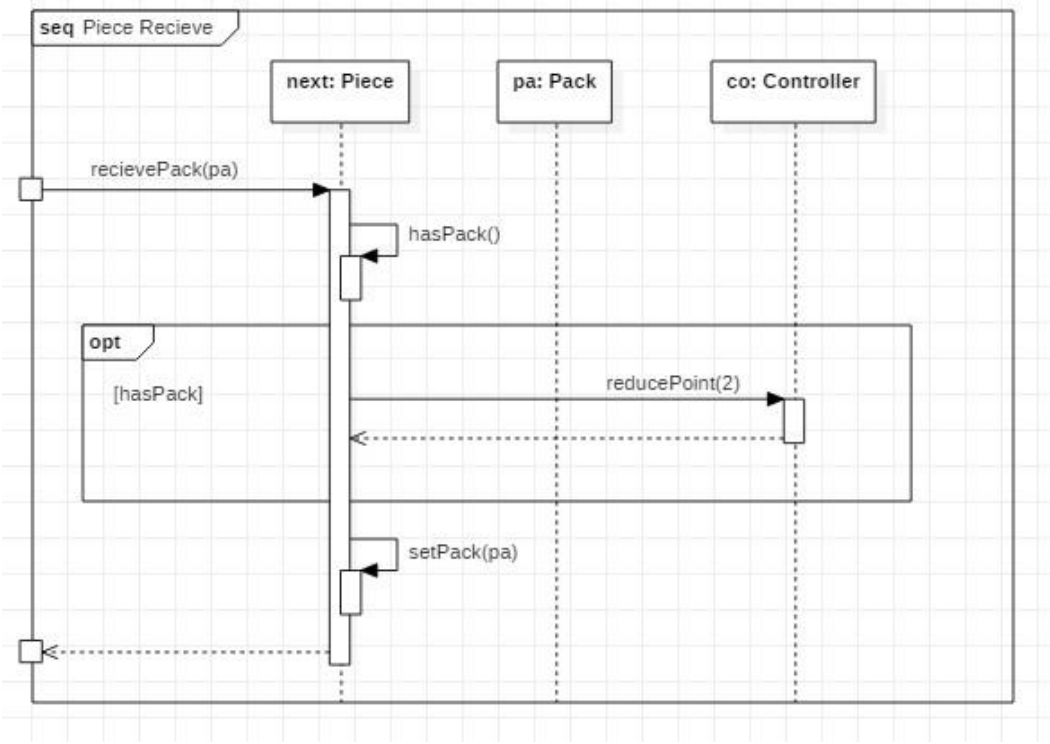
### 5.1.7 Switch Step szekvenciája



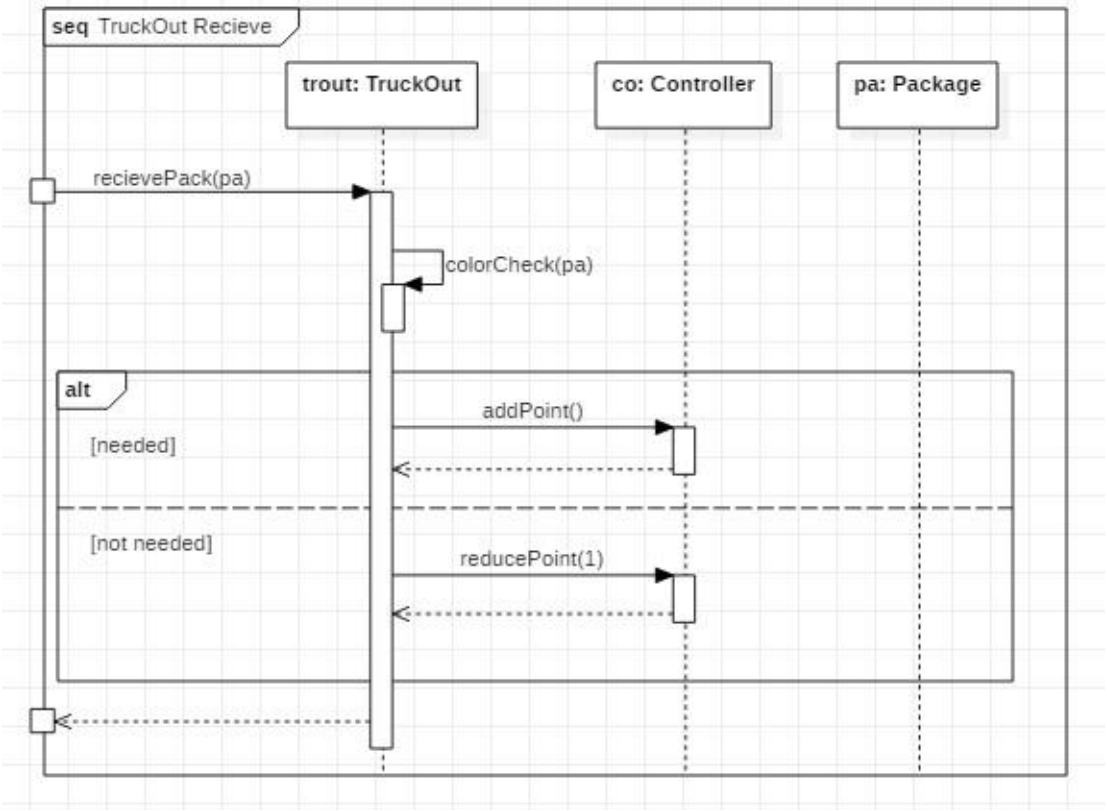
### 5.1.8 End Step szekvenciája



### 5.1.9 Piece Recieve szekvenciája




### 5.1.10 TruckOut Receive



# 6 NAPLÓ

| Kezdet | Időtartam | Elvégzett munka | Hivatkozások |
|---|---|---|---|
| 2018.10.11. 16:00 | 30 perc | Feladat megértése, tudatosulása | 1 |
| 2018.10.11. 16:30 | 15 perc | Feladat leírása | 1 |
| 2018.10.11. 16:45 | 2 óra | Követelmények megalkotása | 2 |
| 2018.10.11. 18:45 | 3 óra | Use-casek megalkotása | 3 |
| 2018.10.12. 16:00 | 5 óra | Osztálydiagram megalkotása | 4.2 |
| 2018.10.13. 18:00 | 5 óra | Osztályok leírása | 4.1 |
| 2018.10.14. 14:00 | 4 óra | Osztályok finomítása, hibák javítása | 4.1 |
| 2018.10.14. 18:00 | 4 óra | Szekvencia diagramok megalkotása | 5.1 |
| 2018.10.14. 22:00 | 15 perc | Átolvasás ellenőrzés | 1,2,3,4,5,6 |
| 2018.10.22. 20:00 | 5 óra | Szekvencia diagramok újraalkotása | 5.1 |
| 2018.10.23. 11:00 | 15 perc | Osztálydiagram átalakítása | 4.2 |
| 2018.10.23. 11:15 | 1 óra | Osztályok leírásának frissítése | 4.1 |
| 2018.10.23. 20:00 | 1 óra | További javítások, átolvasás, ellenőrzés, egyeztetés, dokumentum formázása | 1,2,3,4,5,6 |
| 2018.10.24. 13:00 | 1 óra 15 perc | Szekvencia diagramok redundanciájának csökkentése | 5.1.5, 5.1.7, 5.1.8, 5.1.9 |
| 2018.10.26. 14:00 | 30 perc | Konzultáción hallottak kijavítása | 3.1, 4.1.5, 4.2, 5.1.1 |
| 2018.11.12. 21:30 | 2 óra | Végleges áttekintés, utolsó simítások. | 1,2,3,4,5,6 |

**Összes elvégzett munka:** 24 + 11 óra

**Modellező eszköz:**

- StarUML (Osztálydiagram, Szekvencia diagramok)
- WhiteStarUML (Use-case diagram)

**Egyéb eszközök:**

- Microsoft Word
- Paint