Logistic játék

33 – Silent Hill

Konzulens:
Dr. László Zoltán

Csapattagok

Simon Balázs	VGWJLN	simonbalazs@vipmail.hu
Wirth Benjámin Róbert	MW9RHI	ben@pro.hu
Fülöp István Marcell	LK9JPN	pisti.posta@freemail.hu

2005. szeptember 6.

Tartalomjegyzék

1. fejezet

Követelmények leírása

1.1. A követelmények leírása

A csomag-elosztó állomáson a teherautókon beérkező csomagokat egy futószalag-rendszer közvetítésével más teherautókra rakják át. A játékos feladata a futószalag-rendszer irányítása.

A teherautók az elosztó állomás be- és kilépő pontjaihoz kapcsolódva csomagokat hoznak és visznek. A csomagra jellemző a színe és a romlandósága. Az elosztó állomásra belépő csomagot arra a teherautóra kell továbbítani, amely az adott színű csomagra vár. A várt csomag színét a teherautó mellett elhelyezett indikátor lámpa fénye jelzi. Ha a csomag romlandó (nem mindegyik az), akkor annak még a romlási idő lejárta előtt a teherautóra kell érkeznie, a megromlásig hátralévő időt a csomag felett megjelenő számláló jelzi. Ha a romlási idő lejártakor a csomag még nincs a teherautón, akkor a csomag felrobban.

Kezdetben a futószalagrendszer teljesen üres, az összes csomag a beszállító teherautókon található. A belépő ponthoz álló teherautóról a csomagok véletlenszerű időközönként a pontot érintő (vagy onnan induló) futószalagra esnek. A futószalagról – annak végén – a csomag leesik. A leeső csomag eshet egy másik futószalag valamely pontjára, egy a futószalag végén álló teherautóra, egy váltóra vagy a földre (ha nincs ott teherautó, vagy a futószalag a semmibe vezet).

A váltó célja a csomag irányítása a futószalagok között, hasonlóképp, mint a vasútnál. A futószalagok egy kétdimenziós hálózatot alkotnak. Egy váltóhoz egy vagy két futószalag hozza a csomagot, és mindig két vagy három futószalag egyike viszi el. Egy váltóban egyszerre legfeljebb négy futószalag találkozhat. A váltóra eső csomag némi várakozás után a beállítástól függő elmenő futószalagra esik rá. A játékos az egérrel képes a váltók állítására. Egy egérkattintás a váltót a következő elmenő szalag irányába állítja. Ha a váltó az utolsó elmenő szalagra mutat, akkor a következő egérkattintásra ismét az első szalagra áll át. A szalagok közötti váltás sorrendje előre nem ismert, de kötött, tehát a váltó minden esetben ugyanolyan sorrendben megy végig az egyes szalagokon.

Pontot ér, ha a csomag olyan teherautóba esik, amelyik az adott színű csomagra vár. A csomagokat csak a szállítási tervben meghatározott sorrendben lehet felrakni, az éppen felpakolandó csomag színét az indikátor jelzi. Ha nem a színének megfelelő teherautóhoz lett a csomag irányítva, akkor az már hiba, pontlevonás jár érte. Ha a csomag a földre esik, akkor összetörik. Ha a csomag futószalagra vagy váltóra esik és ott ütközik egy éppen ott tartózkodó csomaggal, akkor az ott tartózkodó csomag összetörik. A csomag összetörése illetve felrobbanása nagy hiba, súlyos pontlevonás jár érte. A felrobbant vagy összetört csomag eltűnik és nem akadályozza a rendszer további működését.

A beszállító teherautókon található csomagok száma és minősége (színe, romlandósága) a játékos számára előre nem ismert – véletlenszerű. Ugyancsak véletlenszerű és a játékos által ismeretlen a kiszállító teherautók kapacitása és szállítási terve. A terv előírja, hogy a teherautóra milyen színű csomagból hányat és milyen sorrendben kell felpakolni, a kapacitás pedig a teherautóra rakodható

csomagok maximális számát jelzi. A beszállítási és kiszállítási tervek nem biztos, hogy egyeznek. A rosszul továbbított csomagok elfoglalhatják a szállítási tervben meghatározott csomagok helyét.

Pontosan annyi beszállító van, mint ahány belépési pont és pontosan annyi kiszállító, ahány kilépési pont, helyüket még a játék elején elfoglalják. Ha egy beszállító teherautó kiürül, vagy ha egy kiszállító teherautó megtelik, akkor elhagyja a helyét. Helyükbe új jármű nem érkezik. Üres kilépési pontnál az indikátor színe színtelenre (szürkére) vált.

A játék akkor ér véget, ha minden beszállító által hozott csomag elfogyott (összetört, felrobbant vagy elszállították), vagy ha minden kiszállító teherautó megtelt. A játékos 10 pontot kap minden helyesen továbbított csomagért. Rosszul továbbított csomagért 3 pont levonás jár. Összetört vagy felrobbant csomag mínusz 5 pontot jelent. Negatív összpontszám is elérhető. Ha a játékos a pályát teljesítette, akkor továbbléphet a következő szintre, amely bonyolultabb hálózatot és több be- illetve kilépési pontot tartalmaz.

1.2. Szótár

belépő pont	itt érkeznek be a csomagok a teherautókról.
beszállító teherautó	erről a teherautóról érkeznek a csomagok.
csomag	ezt kell eljuttatni a megfelelő teherautóhoz.
csomag-elosztó állomás	a beérkező csomagokat a kilépő pontokkal összekötő futószalag-rendszer, maga a játéktér.
eltűnik	a csomag megszünik a pálya részének lenni.
felpakol	a kiszállító teherautóra érkezik a csomag.
felrobban	a csomag felrobban ha romlandó, és a romlandósági idő lejár mielőtt célba érkezne.
futószalag-rendszer	több, csomagok szállítására alkalmas futószalag és az azokat összekötő váltók együttes rendszere.
indikátor lámpa	az éppen felpakolandó csomag színét jelzi.
kapacitás	a teherautóra rakodható csomagok maximális száma.
kilépő pont	ide érkeznek a csomagok a futószalagról.
kiszállító teherautó	erre a teherautóra érkeznek a csomagok.
leesik	a csomag a földre esik, ha nincs a szalag végén teherautó.
pálya	itt a csomag-elosztó állomás szinonímája.
pontszám	a játék célja minél több pontot összegyűjteni. Minden helyesen továbbított csomagért 10 pont jár. Rosszul továbbított csomagért 3 pont levonás jár. Felrobbant illetve összetört csomagért 5 pont levonás jár.
összetörik	csomag összetörhet, ha földre leesik vagy ha ráesik egy másik csomag.
ráesik	a csomag áthelyeződik futószalagról váltóra vagy váltóról futószalagra.
romlandóság	az időtartam, ami alatt a csomagnak el kell jutnia a belépési ponttól a megfelelő teherautóra. Csak néhány csomag tulajdonsága.
romlási idő	itt a romlandóság szinonímája.
szalag	itt a futószalag szinonímája.
szállítási terv	előírja, hogy a teherautóra milyen színű csomagból hányat és milyen sorrendben kell felpakolni.
számláló	a csomag felett jelzi, mennyi idő múlva romlik meg a csomag.
szint	a szinthez tartozik egy csomag-elosztó állomás, és egy – a szint számával együtt növekvő – nehézségi fok. Ahogy a szint száma növekszik, a játék egyre nehezebbé válik (nő a be- illetve kilépési pontok száma, a futószalag-hálózat bonyolultabb lesz).
teherautó	a csomagokat beszállító ill. elszállító jármű.
váltó	futószalagok kapcsolatánál helyezkedik el. A csomag továbbhaladásának irányát határozza meg.

2. fejezet

Követelmény, Project, Funkcionalitás

2.1. Követelmény definíció

2.1.1. A program célja, alapvető feladata

A program nem más, mint egy logisztikai játék, ahol a feladat a csomagok szétosztása. A játék célja a csomagok megfelelő irányítása a futószalagokon a váltók állításával. Részletesebb leírás a játék ismertetésénél található.

A fejlesztés célja egy olyan játékprogram előállítása, mely működőképes, élvezhetően játszható, és amely minden olyan gépen futtatható, melyen a megfelelő Java futtatókörnyezet található. A fejlesztés során különleges hangsúlyt kap az UML modellező rendszer minél tökéletesebb használata.

2.1.2. A fejlesztőkörnyezet

A modellezéshez az Aonix Ameos 9.1.5-ös verziójú szoftvert használjuk, amely képes az objektumdiagramból Java forráskódot is generálni. A forráskód további szerkesztéséhez és a fordításhoz a Borland JBuilderX Foundation fejlesztőkörnyezetet választottuk. Közben ügyelünk arra, hogy a program kompatibilis legyen a Sun Java 1.4-es szabványával. Természetesen a cél az, hogy a játékprogram a Hallgatói Számítógép Központban rendszeresített JDK alatt fordítható és futtatható legyen. A dokumentumokat LaTeX formátumban készítjük el a MiKTeX fordító és a TeXnicCenter szövegszerkesztő segítségével. A unit-tesztekre a JUnit csomagot fogjuk használni (http://www.junit.org/). A napló létrehozására egy külön erre a célra készített Delphi programot írtunk, amely képes a napló bejegyzéseit LaTeX formátumba elmenteni.

2.1.3. A futtatáshoz szükséges környezet

Java Runtime Environment, illetve az a számítógép, mely ezt futtatni képes. (A Sun ajánlásai PC-re: Pentium 166Mhz vagy gyorsabb processzor és 32Mb memória.) A játék használatához grafikus képernyő és egér szükséges.

Magának a programnak nem lesz nagy memóriaigénye, de elképzelhető, hogy a grafikus részek miatt meghaladja az 1Mb-ot. (Ez csak becslés.)

2.1.4. A felhasználói felület

A játékprogram végső változata grafikus felhasználói felülettel rendelkezik. A programot a felhasználó az egér segítségével vezérelheti.

2.1.5. Minőségi tényezők

Teljesítmény: A cél az, hogy a játék élvezhetően játszható legyen a fentebb meghatározott minimális rendszeren. A grafikus felületnél törekedni fogunk a folyamatos animációk alkalmazására, és ügyelni fogunk a folyamatos játékmenet biztosítására.
Újrafelhasználhatóság: A cél az, hogy a grafikus felhasználói felületet a program többi részétől teljesen különválasszuk, így lehetővé téve azt, hogy később a grafikus felület egyszerűen és gyorsan változtatható legyen.
Rugalmasság: A rugalmasságot a fejlesztőkörnyezet biztosítja, a játéknak ugyanis minden olyan környezetben futtathatónak kell lennie, melyben létezik megfelelő Java futtatókörnyezet.
Felhasználhatóság: A használat különösebb tanítást nem igényel, alapfokú számítástechnikai tudással akár a felhasználói kézikönyv elolvasása nélkül is könnyen játszható.

2.1.6. A software minősítése

A kifejlesztett software akkor megfelelő, ha minél pontosabban megegyezik a fentebb leírtakkal. Ezt ellenőrizni lehet a játék futtatásával és kipróbálásával, illetve a forráskód és a modell összevetésével.

2.1.7. A kibocsátás

A program kibocsátása először a forráskóddal együtt a konzulens felé fog történni. Az ellenőrzés és az értékelés után a program hozzáférhető lesz az Interneten is.

2.2. Project terv

2.2.1. A fejlesztői csapat

Csapattag neve	feladatköre
Simon Balázs	csapatvezető, kód, egyéb
Wirth Benjámin	dokumentáció, teszt
Fülöp István	dokumentáció

2.2.2. Életciklus modell

A feladat először a program megtervezése, mely a dinamikus- és objektummodelleket foglalja magába. Ha ez készen van, elkezdhető a skeleton implementálása. Ez a lépés már teljesen meghatározott, nem merülhet fel semmilyen komplikáció, ha a modellek megfelelőek voltak.

A következő feladat a prototípus elkészítése. A programnak ebben az állapotban könnyen tesztelhetőnek kell lennie, hogy a programozási és funkcionalitási logikai hibák könnyen felismerhetők legyenek. A könnyű tesztelhetőség azt jelenti, hogy a bemenetet és a kimenetet is lehet állományból illetve állományba generálni, hogy ennek kiértékelése is egyszerű legyen.

Ha már a prototípus is megfelelő, akkor kezdődhet a grafikus felület megvalósítása. Itt is fontos a tesztelés és a kiértékelés, mert a jó megjelenés sokat számít a játék élvezhetőségében. Ha ennek kifejlesztése is sikeres, készen van a program első teljes változata.

A kötelező feladat csak eddig tart. Ezt a változatot kell leadni a dokumentációval és a forráskóddal együtt.

2.2.3. Szervezési struktúra

A csapat három emberből áll. A feladat szempontjából a tudásunk nem azonos, mindenki más-más területet érez a magáénak, illetve a feladat eltérő részinek megoldásához van nagyobb kedvünk. Azt a felépítést választottuk, hogy mindenki az érdeklődésének és tudásának legmegfelelőbb részt kapja az egész feladatból. A feladatok szétosztását találkozókon beszéljük meg, ahol az egyéni kívánságok mellett ügyelünk arra, hogy minden feladat kiosztásra kerüljön, valamint a csapattagok az egész feladat megoldásából nagyjából egyenlő mértékben vegyék ki a részüket. A találkozók keretében, mivel a szétosztott feladatok nagy mértékben függnek egymástól, javaslatokat teszünk egymásnak a feladat megoldásának körülményeit és a határidőt illetően. Ezt követően a csapattagok elvégzik a rájuk kiosztott feladatot, melyek a következő találkozón való egyeztetés után tehetők közzé.

Hogy a fejlesztés minél hatékonyabb és zökkenőmentesebb legyen, a következő korszerű eszközöket alkalmazzuk:

e-mail: Az egymás számára fontos anyagokat, melyeket a találkozókon előzetesen megbeszéltünk, levélben küldjük el, vagy a Hallgatói Számítógép Központban cseréljük ki.

Msn: Felvettük egymást a Microsoft Messenger-be, hogy szükség esetén egymástól is segítséget tudjunk kérni kisebb technikai problémák megoldásában. (Például hogyan működik a szerkesztőprogram.) Természetesen ezek a feladat lényegét, a project-ről hozott döntéseket nem érinthetik, de kivételes helyzetben akár az Interneten is tarthatunk találkozót.

Ftp: A feladatok megoldása közben keletkezett anyagokat egy, kizárólag a csapat tagjai által hozzáférhető helyen tároljuk. Így mindig elérhető a fejlesztések legfrisseb változata.

2.2.4. Fejlesztési ütemterv

A program fejlesztésének három fő lépcsőfoka van. Ezek a következők:

Skeleton: A cél az, hogy mind a dinamikus, mind az objektum modell jól legyen kitalálva. Ha ezek elkészültek, akkor a fejlesztés szempontjából sikeresen leraktuk az alapokat.

Prototípus: Ez már szinte a teljes változat, csak a grafikus felület elemei hiányoznak. Ez a változat tökéletesen megfelelő arra, hogy az objektumok, rutinok, függvények szemantikai helyességét vizsgáljuk.

Grafikus változat: A program teljes változata. Tulajdonképpen a prototípus a grafikus felülettel kiegészítve, esetleg kismértékben továbbfejlesztve.

2.2.5. Határidők

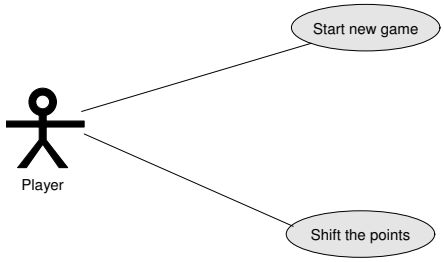
febr. 21.	A team bejelentkezése
febr. 28.	A követelmények leírása
márc. 7.	Követelmények, project, funkcionalitás
márc. 16.	Az analízis modell kidolgozása
márc. 21.	A skeleton tervezése
márc. 29.	A skeleton beadása
ápr. 4.	A prototípus koncepciója
ápr. 11.	Részletes tervek
ápr. 25.	A prototípus beadása
máj. 2.	A grafikus felület specifikációja
máj. 17.	A grafikus változat beadása

2.2.6. Szükséges dokumentációk

Az a dokumentáció, mely a fejlesztés során keletkezik, a program belső működési elvét jeleníti meg, egy esetleges továbbfejlesztésnél nagy előnyt jelent. Ezentúl a felhasználó számára is szükséges egy dokumentáció készítése, a játszhatóság és a könnyű használhatóság érdekében. (Telepítési útmutató, felhasználói leírás.) Ezek a további dokumentumok a fejlesztési dokumentációt ismerő szakembernek semmi újat nem mondanak, az átlagfelhasználótól azonban nem várhatjuk el, hogy kiigazodjon a Use Case-ek és a State Chart-ok között. A program a felhasználónak készül, így azt úgy kell közzétenni, ahogy a felhasználó elvárja, hogy a használata bosszúságok helyett minél több örömöt okozzon neki.

2.3. Essential Use Case-ek

2.3.1. Diagramok



2.1. ábra. Essential Use Case-ek

2.3.2. Use Case leírások

Use Case	Start new game
Actor	Player
Leírás	A játékos a új játékot kezd.

Use Case	Shift the points
Actor	Player
Leírás	A játékos átállítja a váltót.

3. fejezet

Analízis modell

3.1. Objektumok leírása

3.1.1. IPackageCarrier

Ez az általános csatlakozási pont interface. Olyan objektumok valósítják meg, melyekre a játékban más objektumokról csomag eshet, illetve amelyek csomagokat szállíthatnak.

3.1.2. PointOfAssemblyLine

Ez az objektum a futószalag egy pontját reprezentálja, ezen keresztül esik csomag futószalagra. Így lehet biztosítani azt, hogy a futószalag bármely pontját el lehessen érni.

3.1.3. AssemblyLine

Ez az objektum a futószalagot testesíti meg, a szállított csomagokról tartalmaz információkat. Van hossza is. Ha egy csomag pozíciója eléri a végpontot, akkor továbbesik a futószalag végén található tárgyra.

3.1.4. TransporterEntryPoint

Ez az objektum a játékban egy belépő pontot reprezentál, ide csatlakozhatnak azok a teherautók, amelyek még nem teljesített beszállítási tervvel rendelkeznek. A belépési pont tud csak csomagot lepakolni teherautóról, tehát ez egy aktív objektum.

3.1.5. Package

Ez az objektum a csomagot reprezentálja a játékban. Van mérete és típusa. Lehet romlandósági ideje is, ebben az esetben aktív objektumként viselkedik. Ha a csomag összetörik, akkor egy külső objektum fogja összetörni. Felrobbanni a csomag azonban magától fog, de erről értesíti szállítóját is.

3.1.6. AssemblyLinePackageInfo

Ez az objektum a futószalagon szállított csomagról tartalmazza annak pozícióját.

3.1.7. Points

Ez az objektum a váltót testesíti meg a játékban. Aktív objektum, mert bizonyos ideig magánál tartja a csomagot és csak rövid idő múlva engedi tovább. Bármennyi kimenete lehet, a játékban azonban ez az érték 2 és 3 között lesz. A játékos egyedül a váltót tudja vezérelni.

3.1.8. TransporterExitPoint

Ez az objektum egy kilépő pontot testesít meg a játékban, ide csatlakozhatnak azok a teherautók, amelyek még nem teljesített kiszállítási tervvel rendelkeznek. Kezeli az indikátor lámpát is.

3.1.9. PackageInfo

Azért van rá szükség, hogy a Package osztályt ne terheljük túl információkkal, így a felelősségek egyenletesebben lesznek elosztva. Akkor lehet fontos szerepe, ha később továbbfejlesztjük a programot, és nemcsak futószalagokon és váltókon utazhat a csomag.

3.1.10. TransporterJoinPoint

Ez az objektum egy általános belépő vagy kilépő pontot reprezentál. A játék továbbfejlesztésénél fontos szerepe lehet ennek az osztálynak, ugyanis képes olyan teherautókat is fogadni, amelyek mind beszállítási, mind kiszállítási tervvel rendelkeznek.

3.1.11. IndicatorLamp

Ez az objektum azt a kilépő pontnál lévő indikátor lámpát reprezentálja, mely megmutatja, hogy milyen fajtájú csomagra vár a kilépő ponthoz csatlakozó teherautó. Állását a kilépési pont határozza meg.

3.1.12. DeliveryPlan

Az általános szállítási tervet testesíti meg ez az objektum. A szállítási tervet két osztály örökli: a bemenő és a kimenő szállítási terv. A játék elején a beszállítási terv elemeihez tartoznak csomagok, a játék végén pedig a kiszállítási terv elemeihez fognak tartozni. Bár a csomagok véletlenszerűen esnek le a teherautóról és véletlenszerűen várják a teherautók a csomagokat, a szállítási terv létezése azt jelenti, hogy ezek a sorrendek előre meghatározottak. Azonban a szállítási tervek a program indításakor illetve új szintre lépéskor véletlenszerűen generálódnak. Ez a megoldás nagyban elősegíti a tesztelhetőséget és a játék továbbfejleszthetőségét.

3.1.13. Transporter

Ez az objektum a szállító teherautót testesíti meg a játékban. Lehet be- és kiszállítási terve is, de a program jelenlegi követelményei alapján egyszerre csak egyikkel fog rendelkezni. A játék továbbfejlesztésénél azonban fontos szerepe lehet ennek a lehetőségnek is.

A teherautóról csomagot csak a belépési pont tud lepakolni, és csak a kilépési pont tud felpakolni. Lepakolásnál a csomag törlődik a beszállítási tervéből, felpakolásnál megjelenik a kiszállítási tervben.

3.1.14. DeliveryPlanItem

Ez az objektum egy, a szállítási tervben szereplő elemet reprezentál. Tárolja a hozzátartozó csomag típusát és amennyiben a csomag a teherautón található, akkor egy bejegyzést is erről.

3.1.15. InDeliveryPlan

Ez az objektum a teherautó bemenő szállítási tervét reprezentálja.

3.1.16. OutDeliveryPlan

Ez az objektum a teherautó kimenő szállítási tervét testesíti meg.

3.1.17. Ground

Ez az objektum a játékban a földet testesíti meg. Ha a futószalagnak nincs semmi a végpontján, akkor a csomagok a földre fognak esni.

3.1.18. DistributingStation

Ez az objektum magát az egész elosztóállomást testesíti meg. Tartalmazza az állomás gráfját beleértve a futószalagokat, váltókat, be- és kilépési pontokat, és a talajt is.

3.1.19. Game

Ez az osztály a játék fő osztálya, kezeli az új játék kezdését és a szintlépéseket.

3.2. Objektum katalógus

3.2.1. AssemblyLinePackageInfo

A futószalagon szállított Package-ek csomagoló (burkoló) osztálya, amely tárolja a Package pozícióját.

Alaposztályok:

`PackageInfo`

Példányok száma:

n

Konkurencia:

passzív

Perzisztencia:

dinamikus

Komponensek:

nincs

Relációk:

nincs

Változók:

`long`	`position`	a csomag pozíciója a futószalagon

Szolgáltatások:

`long`	`getPosition()`	visszaadja a csomag pozícióját a futószalagon
`void`	`setPosition()`	beállítja a csomag pozícióját a futószalagon
`boolean`	`collideWith()`	igazat ad, ha a csomag ütközni tud az aPosition helyre eső aSize méretű csomaggal

Felelősségek:

3.2.2. PointOfAssemblyLine

A futószalag egy pontja. Ezen az osztályon keresztül esik egy csomag a futószalag meghatározott pozíciójára.

Alaposztályok:

`IPackageCarrier`

Példányok száma:

n

Konkurencia:

passzív

Perzisztencia:

dinamikus

Komponensek:

nincs

Relációk:

nincs

Változók:

`long`	`position`	a továbbítási pont helye a futószalagon

Szolgáltatások:

`void`	`setTarget()`	beállítja a cél futószalagot és pozíciót

Felelősségek:

3.2.3. TransporterExitPoint

A csomagok kilépési pontja, ide csatlakoznak a kiszállító teherautók.

Alaposztályok:

`TransporterJoinPoint`

Példányok száma:

n

Konkurencia:

passzív

Perzisztencia:

dinamikus

Komponensek:

nincs

Relációk:

nincs

Változók:

nincs

Szolgáltatások:

`boolean`	`canConnectTransporter()`	igazat ad vissza, ha a teherautó tud csatlakozni a kilépési ponthoz, vagyis van nem teljesített kiszállítási terve

Felelősségek:

3.2.4. DistributingStation

Az elosztó-állomás "gráfja".

Alaposztályok:

`Object`

Példányok száma:

1

Konkurencia:

passzív

Perzisztencia:

dinamikus

Komponensek:

`TransporterEntryPoint`[1..*]	`entryPoints`	a belépési pontok összessége
`Ground`	`ground`	a talaj
`Points`[0..*]	`points`	a váltók összessége
`AssemblyLine`[0..*]	`assemblyLines`	a futószalagok összessége
`TransporterExitPoint`[1..*]	`exitPoints`	a kilépési pontok összessége

Relációk:

`LogisticGame`	`game`	a játék főosztálya

Változók:

nincs

Szolgáltatások:

`void`	`notifyEmptyJoinPoint()`	a csatlakozási pont értesíti az állomást, hogy a teherautó elhagyta a pontot
`Ground`	`getGround()`	visszaadja a földet

Felelősségek:

3.2.5. IPackageCarrier

Azon osztályok közös interfésze, amikre eshet csomag.

Alaposztályok:

nincs

Példányok száma:

0

Konkurencia:

passzív

Perzisztencia:

dinamikus

Komponensek:

nincs

Relációk:

nincs

Változók:

nincs

Szolgáltatások:

`void`	`packageFall()`	a csomagot ezen az operáción keresztül kapják meg a szállító elemek
`void`	`notifyPackageDestroy()`	a csomag ezen az operáción kereszül értesíti a szállítóját, hogy megsemmisült

Felelősségek:

3.2.6. Ground

A talaj osztálya.

Alaposztályok:

`IPackageCarrier`

Példányok száma:

1

Konkurencia:

passzív
Perzisztencia:
dinamikus
Komponensek:
nincs
Relációk:
nincs
Változók:
nincs
Szolgáltatások:
nincs
Felelősségek:

3.2.7. IndicatorLamp

A kilépési ponthoz tartozó indikátorlámpa.
Alaposztályok:
`Object`
Példányok száma:
n
Konkurencia:
passzív
Perzisztencia:
dinamikus
Komponensek:
nincs
Relációk:
nincs
Változók:

`long`	`state`	az indikátorlámpa állapota

Szolgáltatások:

`void`	`setState()`	beállítja az indikátorlámpa állapotát

Felelősségek:

3.2.8. AssemblyLine

A futószalag osztálya.
Alaposztályok:
`IPackageCarrier`
Példányok száma:
n
Konkurencia:
aktív
Perzisztencia:
dinamikus

Komponensek:
nincs
Relációk:
nincs
Változók:

`long`	`speed`	a futószalag sebessége
`long`	`length`	a futószalag hossza

Szolgáltatások:

`void`	`packageFall()`	a futószalagra az aPackage csomag esik az aPosition helyre
`void`	`setEndPoint()`	a futószalag végpontjának beállítása

Felelősségek:

3.2.9. Package

A csomagot reprezentáló osztály.
Alaposztályok:
`Object`
Példányok száma:
n
Konkurencia:
aktív
Perzisztencia:
dinamikus
Komponensek:
nincs
Relációk:
nincs
Változók:

`long`	`size`	a csomag mérete
`long`	`kind`	a csomag típusa
`long`	`decayTime`	a csomag romlandósági ideje
`IPackageCarrier`	`carrier`	az aktuális szállító objektum

Szolgáltatások:

`void`	`setCarrier()`	beállítja az aktuális szállító objektumot
`long`	`getSize()`	visszaadja a csomag méretét
`long`	`getKind()`	visszaadja a csomag típusát
`void`	`smash()`	a csomag összetörik
`void`	`explode()`	a csomag felrobban
`void`	`startDecay()`	a csomag elkezd megromlani
`void`	`stopDecay()`	a csomag nem romlik tovább

Felelősségek:

3.2.10. LogisticGame

A játék főosztálya.

Alaposztályok:

`Object`

Példányok száma:

1

Konkurencia:

passzív

Perzisztencia:

dinamikus

Komponensek:

nincs

Relációk:

`DistributingStation`	`station`	az elosztó-állomás gráfja

Változók:

`long`	`level`	az aktuális szint
`long`	`credits`	a gyűjtött pontok száma, statikus változó, hogy egyszerűbb legyen a számítás

Szolgáltatások:

`void`	`switchToNextLevel()`	váltás a következő szintre
`void`	`startNewGame()`	új játék kezdése
`void`	`addCredit()`	pontok hozzáadása az eddig összegyűjtött pontokhoz (levonás, ha negatív), azért statikus, hogy minden objektum el tudja érni

Felelősségek:

3.2.11. TransporterJoinPoint

A teherautók csatlakozási pontja.

Alaposztályok:

`IPackageCarrier`

Példányok száma:

n

Konkurencia:

passzív

Perzisztencia:

dinamikus

Komponensek:

nincs

Relációk:

nincs

Változók:

`DistributingStation`	`station`	az elosztóállomás

Szolgáltatások:

`boolean`	`canConnectTransporter()`	igazat ad vissza, ha a teherautó tud csatlakozni a ponthoz
`boolean`	`connectTransporter()`	teherautót csatlakoztatja a ponthoz és igazat ad vissza, ha sikerült
`void`	`disconnectTransporter()`	a teherautó lecsatlakoztatása, ha a szállítási terve teljesítve van
`boolean`	`isFree()`	igazat ad vissza, ha szabad a csatlakozási pont

Felelősségek:

3.2.12. Transporter

A teherautók osztálya.

Alaposztályok:

`Object`

Példányok száma:

n

Konkurencia:

passzív

Perzisztencia:

dinamikus

Komponensek:

nincs

Relációk:

nincs

Változók:

nincs

Szolgáltatások:

`void`	`createInDeliveryPlan()`	létrehoz egy beszállítási tervet itemCount elemszámmal
`void`	`createOutDeliveryPlan()`	létrehoz egy kiszállítási tervet itemCount elemszámmal
`Package`	`packDownPackage()`	leveszi a csomagot a teherautóról és visszaadja azt
`void`	`packUpPackage()`	felpakolja a csomagot a teherautóra
`boolean`	`hasInDeliveryPlan()`	igazat ad vissza, ha a teherautónak van beszállítási terve
`boolean`	`hasOutDeliveryPlan()`	igazat ad vissza, ha a teherautónak van kiszállítási terve
`boolean`	`isInDeliveryPlanCompleted()`	igazat ad vissza, ha a teherautó beszállítási terve teljesítve van
`boolean`	`isOutDeliveryPlanCompleted()`	igazat ad vissza, ha a teherautó kiszállítási terve teljesítve van
`long`	`getWaitedPackageKind()`	visszaadja a várt csomag típusát

Felelősségek:

3.2.13. TransporterEntryPoint

A csomagok belépési pontja, ide csatlakoznak a beszállító teherautók.

Alaposztályok:

`TransporterJoinPoint`

Példányok száma:

n

Konkurencia:

aktív

Perzisztencia:

dinamikus

Komponensek:

nincs

Relációk:

nincs

Változók:

`long`	`timeToPackDown`	a következő lepakolásig hátralévő idő

Szolgáltatások:

`boolean`	`canConnectTransporter()`	igazat ad vissza, ha a teherautó tud csatlakozni a belépési ponthoz, vagyis van nem teljesített beszállítási terve
`void`	`packDownPackage()`	lerak egy csomagot a csatlakozott teherautóról
`void`	`setJoinPoint()`	beállítja azt csatlakozási pontot, ahova a teherautóról lerakodott csomagok esnek

Felelősségek:

3.2.14. OutDeliveryPlan

A kiszállítási terv.

Alaposztályok:

`DeliveryPlan`

Példányok száma:

n

Konkurencia:

passzív

Perzisztencia:

dinamikus

Komponensek:

nincs

Relációk:

nincs

Változók:

nincs

Szolgáltatások:

`void`	`packUpPackage()`	felveszi a csomagot
`long`	`getNextPackageKind()`	visszaadja a várt csomag típusát
`boolean`	`isCompleted()`	igazat ad vissza, ha a kiszállítási terv teljesítve van

Felelősségek:

3.2.15. InDeliveryPlan

A beszállítási terv.

Alaposztályok:

`DeliveryPlan`

Példányok száma:

n

Konkurencia:

passzív

Perzisztencia:

dinamikus

Komponensek:

nincs

Relációk:

nincs

Változók:

nincs

Szolgáltatások:

`Package`	`packDownPackage()`	visszaadja a lerakandó csomagot
`boolean`	`isCompleted()`	igazat ad vissza, ha a beszállítási terv teljesítve van

Felelősségek:

3.2.16. DeliveryPlanItem

a szállítási terv eleme

Alaposztályok:

`Object`

Példányok száma:

n

Konkurencia:

passzív

Perzisztencia:

dinamikus

Komponensek:

nincs

Relációk:

nincs

Változók:

`long`	`kind`	a szállítási terv elemének típusa

Szolgáltatások:

`long`	`getKind()`	visszaadja a szállítási terv elemének típusát
`Package`	`getPackage()`	visszaadja a szállítási terv eleméhez rendelt csomagot
`void`	`setPackage()`	hozzárendeli a csomagot a szállítási terv eleméhez

Felelősségek:

3.2.17. DeliveryPlan

A szállítási tervek ősosztálya.

Alaposztályok:

`Object`

Példányok száma:

n

Konkurencia:

passzív

Perzisztencia:

dinamikus

Komponensek:

`DeliveryPlanItem[0..*]`	`items`	a szállítási terv elemei

Relációk:

nincs

Változók:

`long`	`completedCount`	a szállítási tervben teljesített tételek száma

Szolgáltatások:

`long`	`count()`	visszaadja a szállítási terv elemeinek számát
`DeliveryPlanItem`	`getItem()`	visszaadja a szállítási terv adott sorszámú elemét
`boolean`	`isCompleted()`	igazat ad vissza, ha a szállítási terv teljesítve van

Felelősségek:

3.2.18. Points

A váltót reprezentáló osztály.

Alaposztályok:

`IPackageCarrier`

Példányok száma:

n

Konkurencia:

aktív

Perzisztencia:

dinamikus

Komponensek:

nincs

Relációk:

nincs

Változók:

`long`	`holdTime`	a váltó ennyi ideig tartja magánál a csomagot
`long`	`actLine`	a váltó aktuális iránya

Szolgáltatások:

`void`	`shiftToNextLine()`	a váltó a következő irányba vált
`void`	`addLine()`	új ágat ad a váltóhoz

Felelősségek:

3.2.19. PackageInfo

A Package-ek szállításához szükséges információk tárolásáért felelős csomagolóosztályok őse.

Alaposztályok:

`Object`

Példányok száma:

n

Konkurencia:

passzív

Perzisztencia:

dinamikus

Komponensek:

nincs

Relációk:

nincs

Változók:

nincs

Szolgáltatások:

`Package`	`getPackage()`	visszaadja a csomagot
`boolean`	`hasPackage()`	igazat ad vissza, ha erről a csomagról tároljuk az információt

Felelősségek:

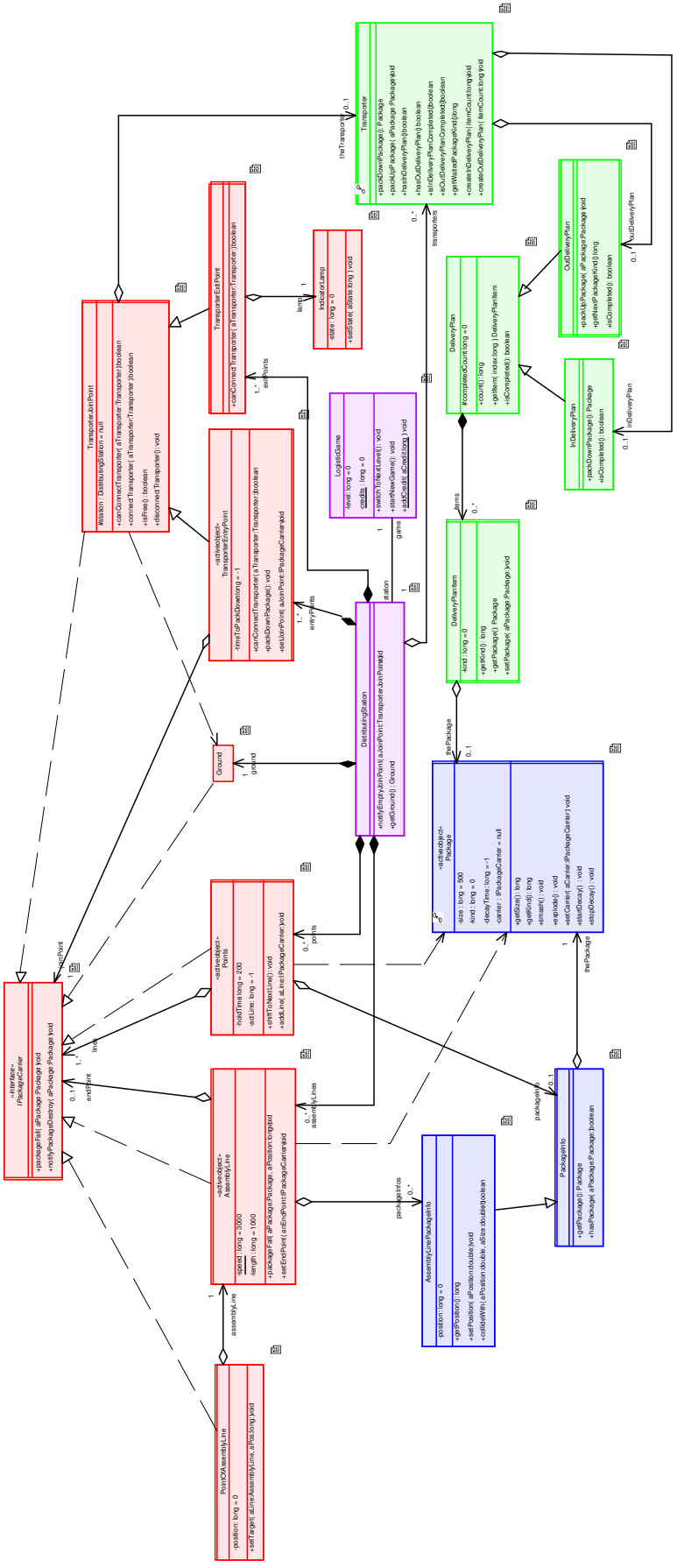
3.3. Megjegyzések a diagramokhoz

3.3.1. Osztálydiagram

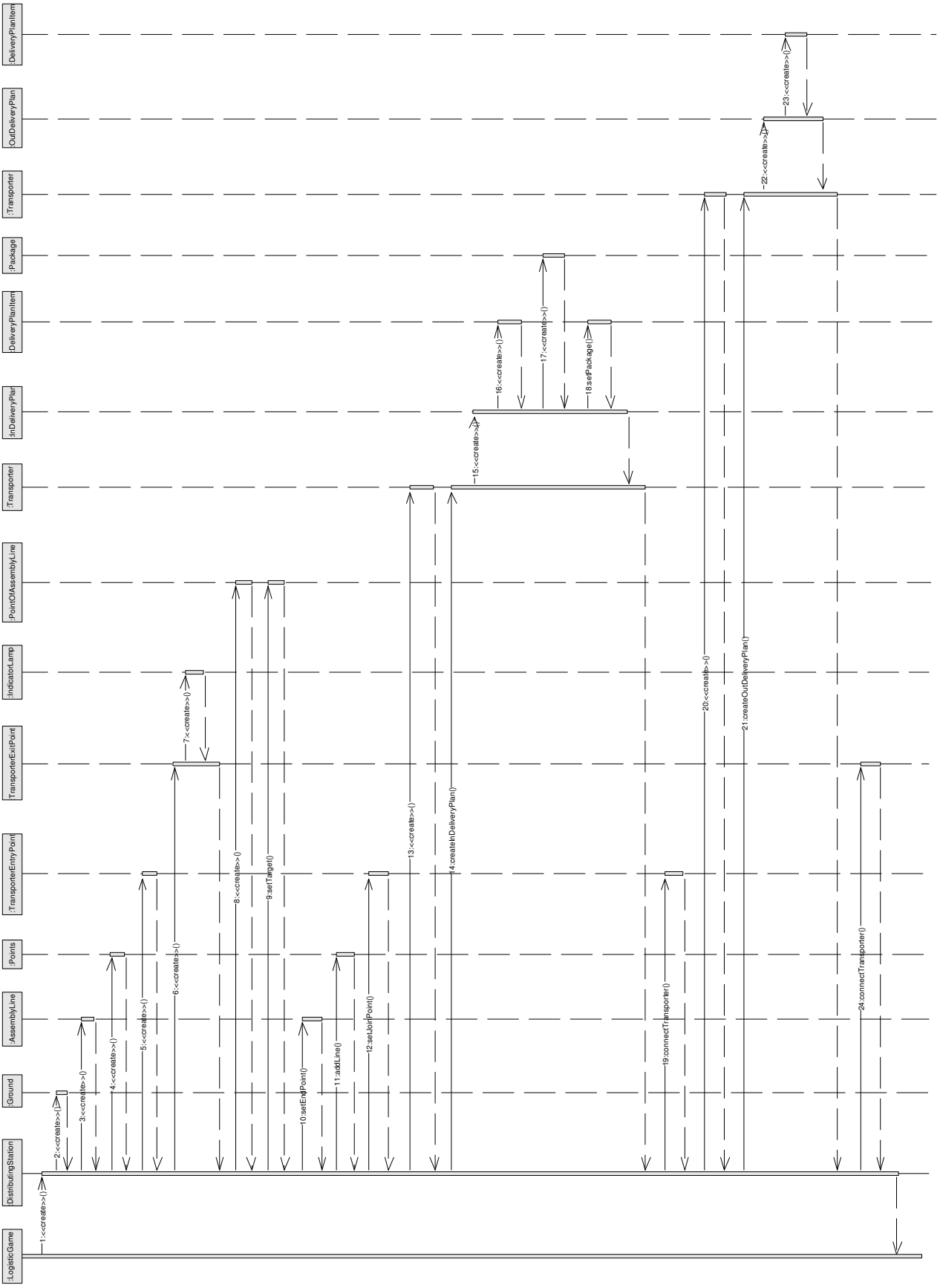
Az osztálydiagramon néhány attribútumot lekérdező függvényt az áttekinthetőség kedvéért nem ábrázoltunk, de ezek a függvények az objektumkatalógusban szerepelnek. Lehetséges, hogy a program fejlesztése során további attribútum lekérdező és beállító függvényekre is szükség lesz majd.

3.3.2. Szekvencia diagramok

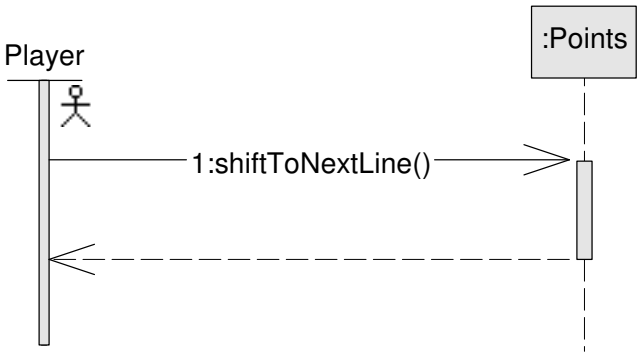
A szekvencia diagramok az UML specifikáció szerint objektumpéldányokat tartalmaznak, így a dobozokban a szövegeket alá kellene húzni, de sajnos a modellező tool ezt nem teszi meg, és nem tud kollekciókat sem ábrázolni. A kollekciókat a későbbiekben a diagram mellé írt megjegyzéssel fogjuk jelezni.



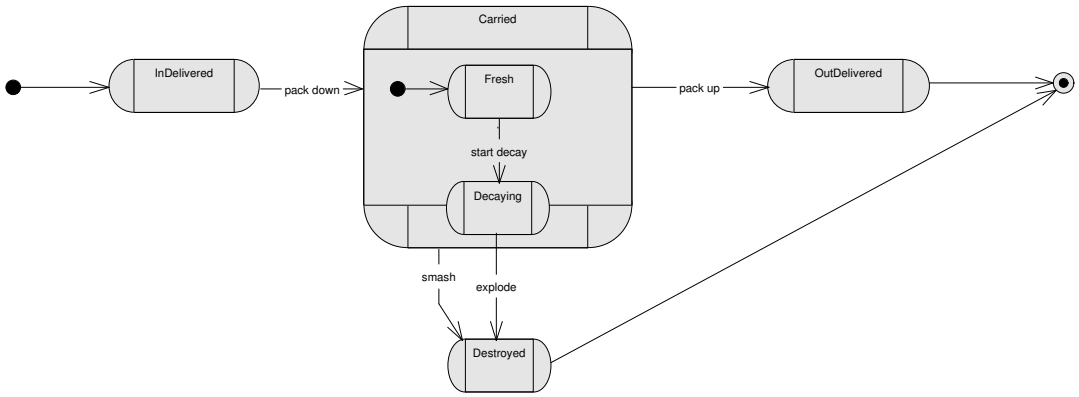
3.1. ábra. Static structure diagram



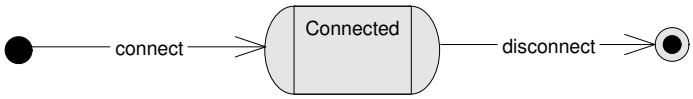
3.2. ábra. Szekvenciadiagram inicializálásra, új játék kezdésére(Start new game use-case) és szintlépésre



3.3. ábra. Szekvenciadiagram a Shift the points use-case-re



3.4. ábra. A `Package` state-chartja



3.5. ábra. A `Transporter` state-chartja

4. fejezet

Analízis modell

4.1. A skeleton valóságos use case-ei

Use Case	Start new game
Actor	Player
Leírás	Ez a use-case azt írja le, hogy mi történik akkor, amikor a user új játékot kezd.

Use Case	Next level
Actor	Player
Leírás	Ez a use-case azt írja le, hogy mi történik akkor, amikor a user szintet lép.

Use Case	Shift the points
Actor	Player
Leírás	Ez a use-case azt írja le, hogy mi történik akkor, amikor a user átállítja valamelyik váltót.

Use Case	Package to assembly line
Actor	Player
Leírás	Ez a use-case azt írja le, hogy mi történik akkor, amikor a futószalagra csomag esik, de nem ütközik másik csomaggal.

Use Case	Package to points
Actor	Player
Leírás	Ez a use-case azt írja le, hogy mi történik akkor, amikor a váltóra csomag esik, de nem ütközik másik csomaggal.

Use Case	Package to assembly line with smash
Actor	Player
Leírás	Ez a use-case azt írja le, hogy mi történik akkor, amikor a futószalagra csomag esik és másik csomaggal ütközik.

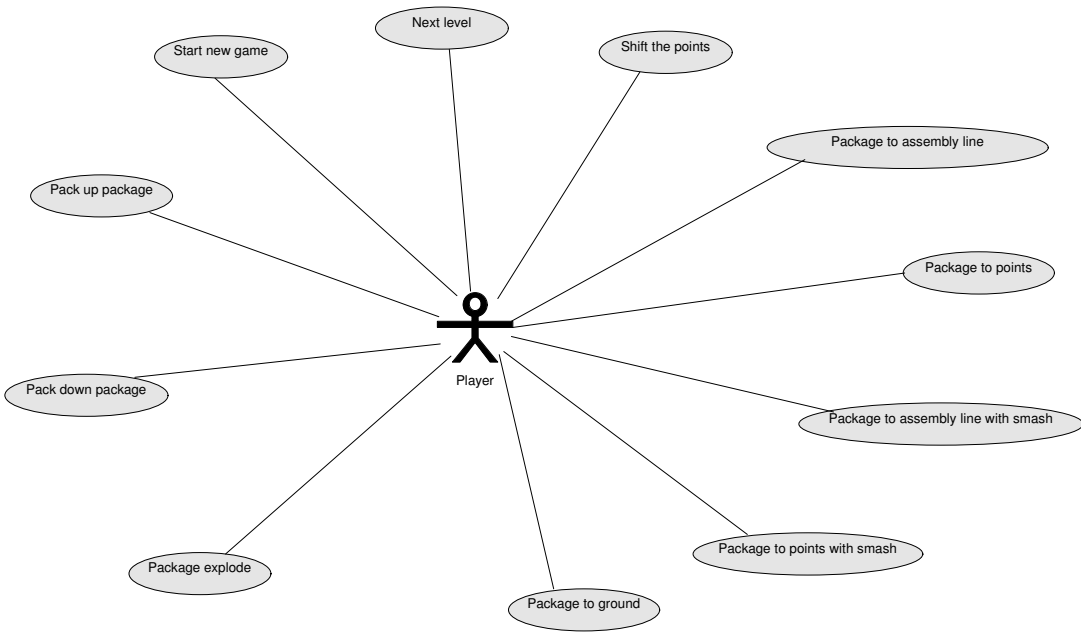
Use Case	Package to points with smash
Actor	Player
Leírás	Ez a use-case azt írja le, hogy mi történik akkor, amikor a váltóra csomag esik és másik csomaggal ütközik.

Use Case	Package to ground
Actor	Player
Leírás	Ez a use-case azt írja le, hogy mi történik akkor, amikor a földre csomag esik.

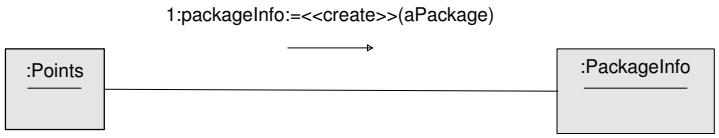
Use Case	Package explode
Actor	Player
Leírás	Ez a use-case azt írja le, hogy mi történik akkor, amikor a csomag felrobban.

Use Case	Pack down package
Actor	Player
Leírás	Ez a use-case azt írja le, hogy mi történik akkor, amikor a belépő pontnál lepakolnak egy csomagot.

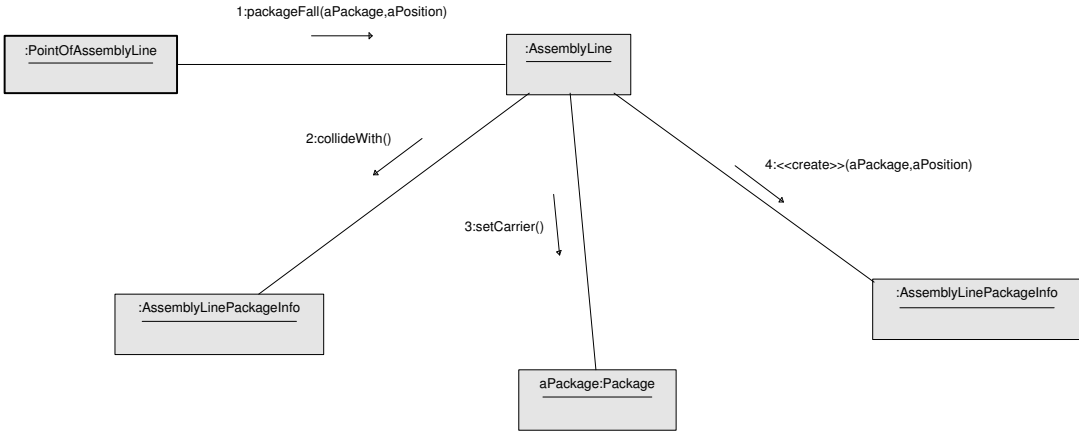
Use Case	Pack up package
Actor	Player
Leírás	Ez a use-case azt írja le, hogy mi történik akkor, amikor a kilépési pontnál felpakolnak egy csomagot.



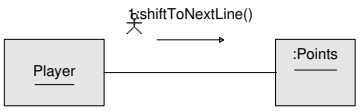
4.2. Kollaborációs diagramok



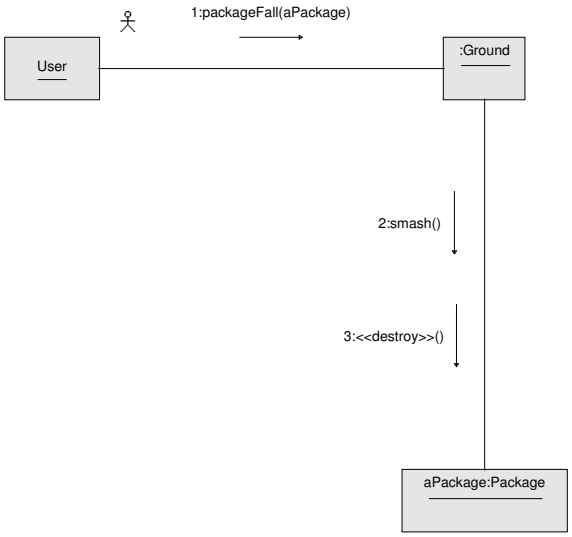
4.1. ábra. Kollaborációs diagram a "Package to points" use-case-hez



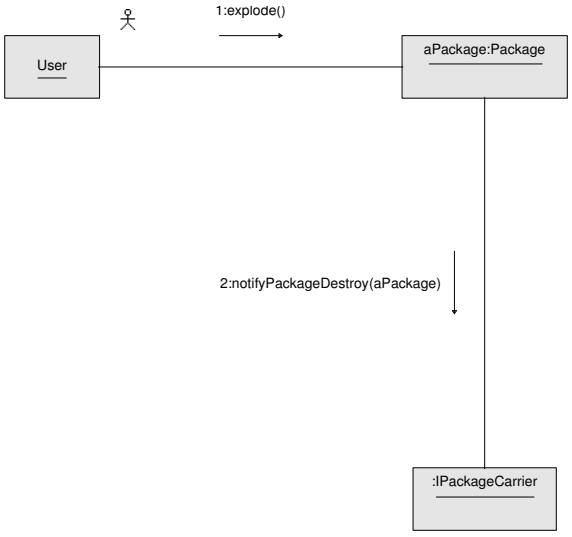
4.2. ábra. Kollaborációs diagram a "Package to assembly line" use-case-hez



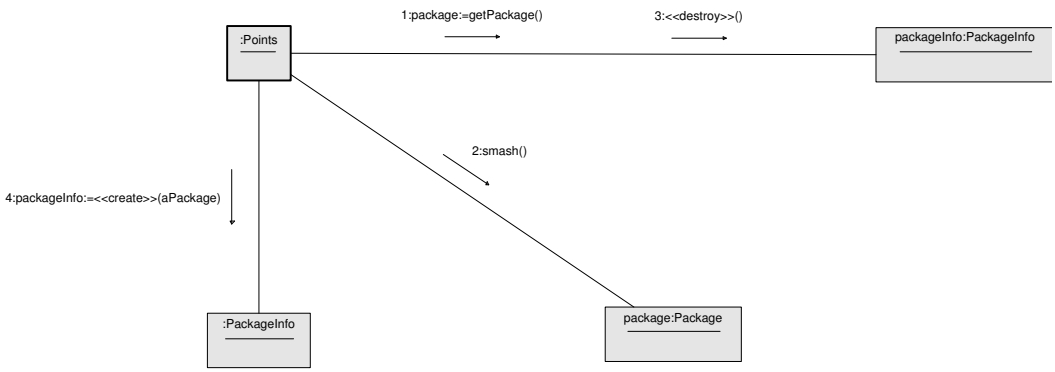
4.3. ábra. Kollaborációs diagram a "Shift the points" use-case-hez



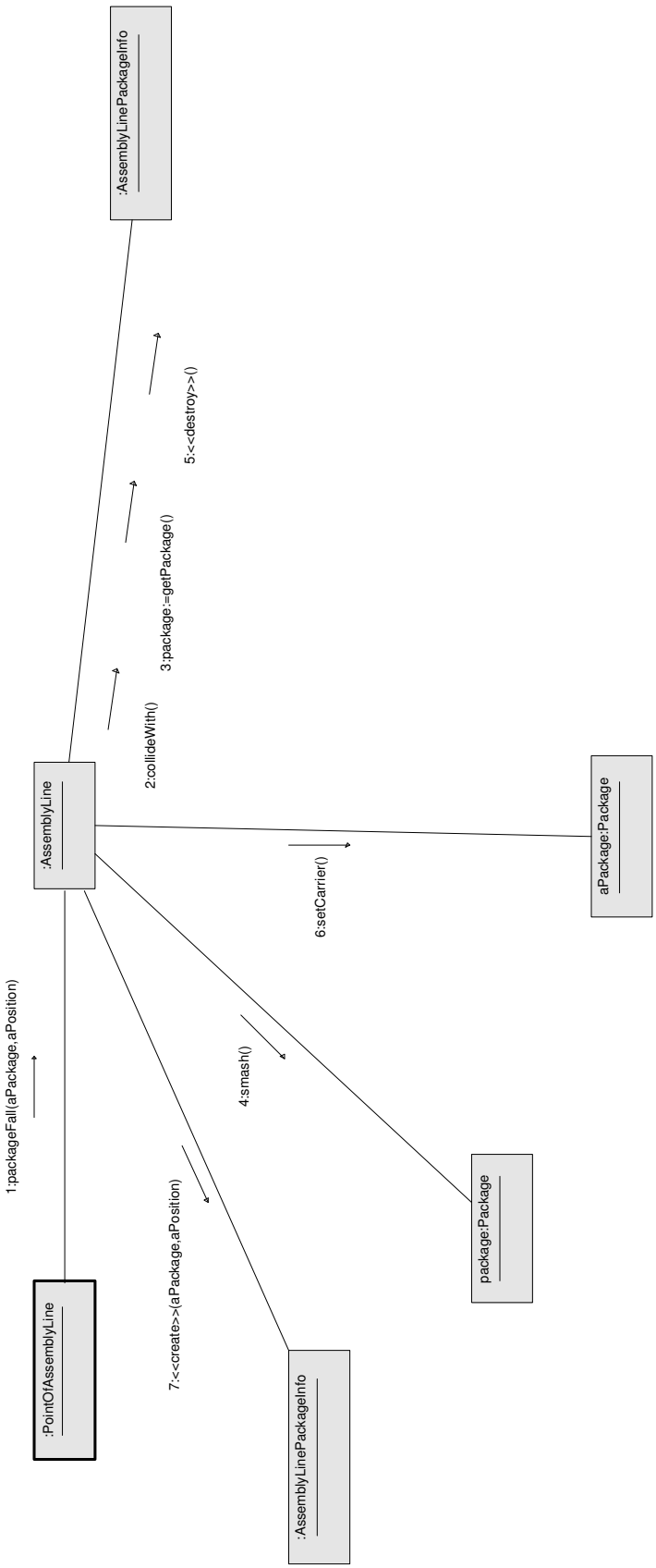
4.4. ábra. Kollaborációs diagram a "Package to ground" use-case-hez



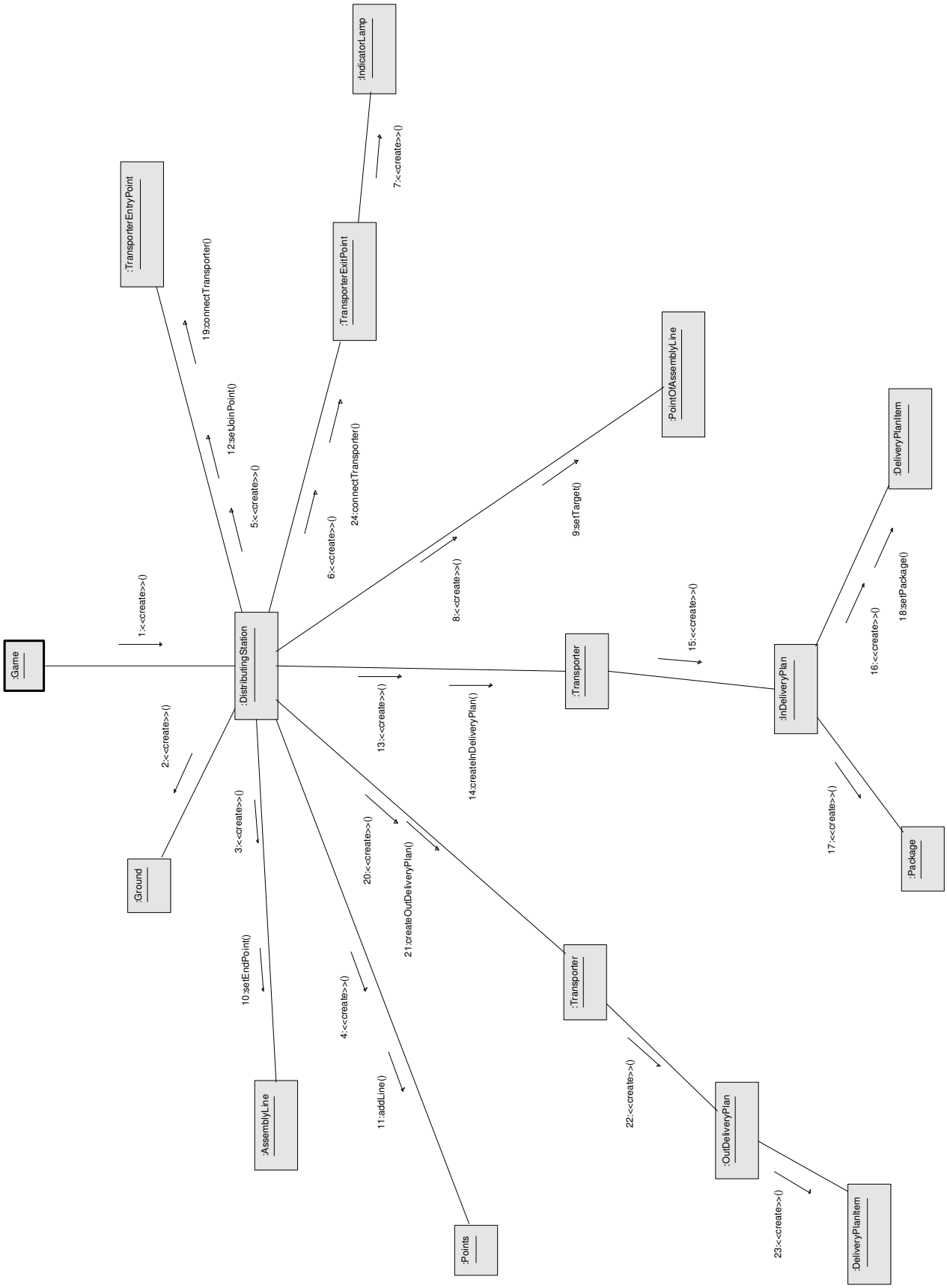
4.5. ábra. Kollaborációs diagram a "Package explode" use-case-hez



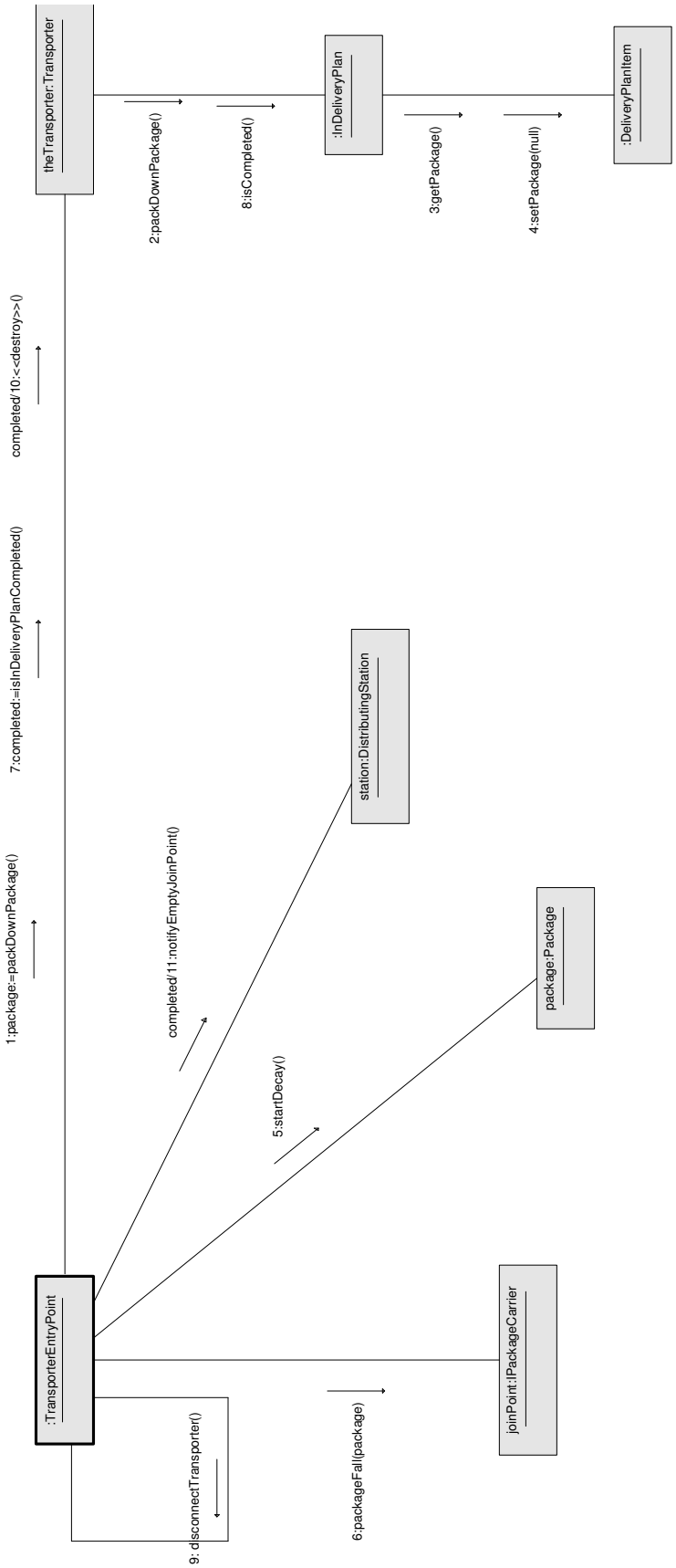
4.6. ábra. Kollaborációs diagram a "Package to points with smash" use-case-hez



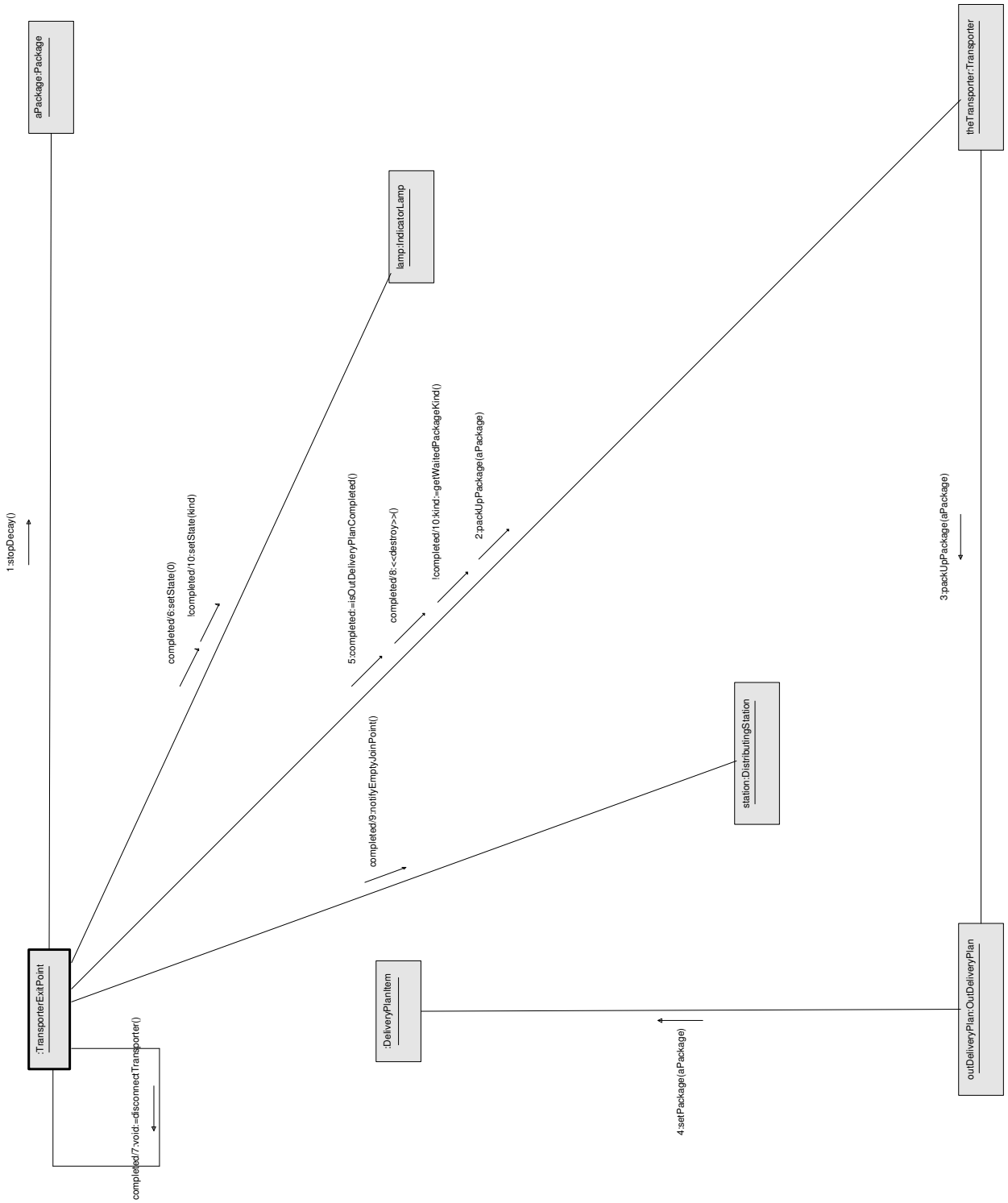
4.7. ábra. Kollaborációs diagram a "Package to assembly line with smash" use-case-hez



4.8. ábra. Kollaborációs diagram az inicializáláshoz, "Start new game" és "Next level" use-case-hez



4.9. ábra. Kollaborációs diagram a "Pack down package" use-case-hez



4.10. ábra. Kollaborációs diagram a "Pack up package" use-case-hez

4.3. A skeleton kezelői felületének terve, dialógusok

A program skeleton változatának egy lényeges jellemzője, hogy a bemeneteit és a kimeneteit a parancssorból vegye. (System.in, System.out.) Ez biztosítja azt, hogy a program tesztelése külön erre készült célprogramokkal megvalósítható legyen. Hogy a működés követhető legyen a felhasználó számára is, célszerű valamilyen megjelenítési formát is kidolgozni. A skeleton-nál az egyik szempont, hogy a megjelenítés magában a parancssorban történjen.

A programnak, ha bemenetre vár, körül kell írnia a felhasználó számára, hogy pontosan milyen adatra van szüksége. Ennek a megvalósítása szintén a parancssorban kell, hogy történjen.

A program a vizsgálatok során felmerülő kérdéseknél közvetlenül a felhasználót kérdezi, ezért a programban szereplő objektumok attribútumai tulajdonképpen feleslegesek, megjelenítésük nem szükséges.

A tesztelésnél azt kell megvalósítani, hogy minden függvény kiírja, hogyha meghívták, majd kiírja azt is, amikor visszatér. A függvénynek ez a két jelzése körülöleli az adott függvényen belül található többi függvényhívást, ebből következik, hogy függvényenkét egyetlen kiírás nem elegendő. Többnyire elhanyagolhatóak a függvények argumentumai is.

Azonos osztályba tartozó objektumok között is felmerülhet függvényhívás, ez csökkentheti az áttekinthetoséget, esetleg félreértésekhez vezethet, így célszerűnek látszik egy olyan opció biztosítása, hogy az objektumok egyedi azonosítóval (sorszámmal) rendelkezzenek. A probléma például akkor kerülhet elő, mikor nem tiszta, hogy egy belső, objektumon belüli függvényhívás történt-e, vagy egy másik de azonos osztályú objektum egy függvényét hívtuk.

Így a kiírandó adatok a függvény meghívásakor és a visszatéréskor: az objektum sorszáma, az adott osztály neve, a függvény neve, valamint paramétereinek nevei. Az egyes függvényhívásokon belüli újabb függvényhívások leírását a szkeleton a jobb áttekinthetőség érdekében néhány karakterrel beljebb kezdi, így a kimenet hasonlít a szekvenciadiagramra.

A kérdések feltétele a következő formában történik: a kérdés és az elfogadható válaszlehetőségek.

A képernyő képe:

```
     -> Package[3].setCarrier(IPackageCarrier aCarrier)
     <- Package[3].setCarrier(IPackageCarrier aCarrier)
     -> PointOfAssemblyLine[5].packageFall(Package aPackage)
         -> AssemblyLine[2].packageFall(Package aPackage, long aPosition)
             -> AssemblyLinePackageInfo[10].collideWith(long aPosition, long aSize)
Collide with package? Yes/No?
             <- AssemblyLinePackageInfo[10].collideWith(long aPosition, long aSize)
             -> Package[4].smash()
                 -> Package[4].stopDecay()
                 <- Package[4].stopDecay()
             <- Package[4].smash()
             -> Package[3].setCarrier(IPackageCarrier aCarrier)
             <- Package[3].setCarrier(IPackageCarrier aCarrier)
             -> PackageInfo.PackageInfo(Package aPackage)  [11]
             <- PackageInfo.PackageInfo(Package aPackage)
             -> AssemblyLinePackageInfo.AssemblyLinePackageInfo(Package aPackage, long aPosition)  [12]
             <- AssemblyLinePackageInfo.AssemblyLinePackageInfo(Package aPackage, long aPosition)
         <- AssemblyLine[2].packageFall(Package aPackage, long aPosition)
     <- PointOfAssemblyLine[5].packageFall(Package aPackage)
 <- AssemblyLine[1].forwardPackage(long aIndex)
```

4.4. Architektúra

A program helyes működésének ellenőrzéséhez a skeleton változatban a következő pályákat valósítojuk majd meg:

4.4.1. Első pálya

Az első pálya egyetlen belépő pontot tartalmaz, melyhez egy beszállító teherautó csatlakozik. A beszállítási tervben egyetlen csomag szerepel. A belépő pont egy kilépő ponthoz kapcsolódik, melyhez egy kiszállító teherautó csatlakozik. A kiszállítási tervben is egyetlen csomag szerepel.

Ezen a pályán lehet tesztelni a "Pack down package" és a "Pack up package" use case-eket, ahogy a beszállító teherautó szállítási tervéből a csomag a belépő pontra kerül, és ahogy a kilépő pontról a kiszállító teherautó szállítási tervébe kerül.

4.4.2. Második pálya

Az második pálya tartalmaz egy futószalagot, rajta egy csomag tartózkodik. A futószalag egy futószalag csatlakozási ponton keresztül egy másik futószalaghoz kapcsolódik, amely a földhöz csatlakozik, és ezen is egy csomag helyezkedik el.

Ezen a pályán lehet tesztelni a "Package to assemly line with smash" use case-t, ahogy összetörik a futószalagon tartózkodó csomag, amikor egy másik csomag a futószalag egy pontján keresztül ráesik.

4.4.3. Harmadik pálya

A harmadik pálya egy futószalagot tartalmaz. A futószalagon egy csomag tartózkodik és egy váltóhoz kapcsolódik. A váltó a földhöz csatlakozik, rajta egy csomag van.

Ezen a pályán lehet tesztelni a "Package to points with smash" use case-t, ahogy összetörik-a váltóban tartózkodó csomag, amikor egy másik csomag a futószalagról ráesik.

4.4.4. Negyedik pálya

A negyedik pálya egy futószalagot tartalmaz. A futószalagon egy csomag tartózkodik és a földhöz kapcsolódik.

Ezen a pályán lehet tesztelni a "Package to ground" use case-t, hogy mi történik a csomaggal, amikor a belépő pontról a földre esik. Lehet tesztelni a "Package explode" use case-t, ahogy a csomag felrobban, amikor lejár a romlandósági ideje.

4.4.5. Ötödik pálya

Az ötödik pálya tartalmaz egy futószalagot. A futószalagon egy csomag tartózkodik és egy váltóhoz kapcsolódik. A váltónak két állása van, melyek egy-egy futószalag csatlakozási ponton keresztül egy-egy futószalaghoz kapcsolódnak. A futószalagok a földhöz csatlakoznak.

Ezen a pályán lehet tesztelni a "Shift the points" use case-t, hogy a váltó állításának hatására mely futószalag csatlakozási pontjára esik a váltóról a csomag. Lehet tesztelni a "Package to points" use case-t, ahogy a csomag a futószalagról a váltóra esik. Lehet tesztelni a "Package to assembly line" use case-t, ahogy a csomag a váltóról a futószalagra esik.

4.4.6. Hatodik pálya

A hatodik pálya egy belépési pontot tartalmaz, amelynél egy teherautó található, egyelemű szállítási tervvel. A belépési pont futószalaghoz kapcsolódik, amely váltóban végződik. A váltó közvetlenül egy kilépési ponthoz csatlakozik, amelynél szintén egy teherautó található egyelemű szállítási tervvel.

Ezen a pályán az inicializálást, valamint a "Start new game" és a "Next level" use-case-eket lehet tesztelni.

4.5. Ütemezés

A program végső változatában az ütemezést szálak futása biztosítja majd. A megfelelő szálak az objektumok belső osztályai lesznek.

A skeleton-ban az időnek nincs jelentősége, az ütemezés teljesen kézzel, a parancssorból történik. Így nincs szükség a bonyolult, több szálas ütemezési rendszer felépítésére sem.

5. fejezet

Skeleton beadása

5.1. A skeleton fordítása és futtatása

5.1.1. DOS parancssorból

1. A mellékelt zip file-t ki kell csomagolni egy könyvtárba.
2. A `compile.bat` paranccsal lefordítjuk a forrásálományokat.
3. A program indítása a `run.bat` paranccsal történik.
4. A JavaDoc dokumentációt a `doc.bat` (internetkapcsolat hiányában a `docl.bat`) segítségével állíthatjuk elő. Ekkor létrejön egy Doc nevű könyvtár, amelyből az index.html oldalt webböngészőben megnyitva rendelkezésünkre áll egy jól használható objektumkatalógus. Az itt szereplő leírások a forráskódban is megtalálhatóak kommentek formájában.

5.1.2. DOS parancssor hiányában

1. A mellékelt zip file-t ki kell csomagolni egy könyvtárba.
2. A következő parancsokkal fordíthatjuk le a rendszert:
```
javac logistic/*.java logistic/skeleton/*.java
    logisticskeleton/*.java
jar cmf logistic.mf logistic.jar logistic/*.class
    logistic/skeleton/*.class logisticskeleton/*.class
```
3. A program indítása a `java -jar logistic.jar` parancs kiadásával történik.
4. A JavaDoc dokumentációt a
```
javadoc -locale hu_HU -encoding iso-8859-2 -charset iso-8859-2
    -docencoding iso-8859-2
    -link http://java.sun.com/products/jdk/1.3/docs/api
    -d Doc -sourcepath logistic -private -author -windowtitle
    "Logistic game - Silent Hill" logistic/*.java
    logistic/skeleton/*.java logisticskeleton/*.java
```
internetkapcsolat hiányában pedig a
```
javadoc -locale hu_HU -encoding iso-8859-2 -charset iso-8859-2
    -docencoding iso-8859-2 -linkoffline -d Doc -sourcepath
    logistic -private -author -windowtitle
    "Logistic game - Silent Hill" logistic/*.java
```

`logistic/skeleton/*.java logisticskeleton/*.java`
segítségével állíthatjuk elő.

5.2. A mellékelt álomány tartalma

Az elküldött zip file tartalma az alábbi listában szereplő file-ok.

`./` tartalma:

A `compile.bat` és a `logistic.mf` a program fordításához, a `logistic.jar` file elkészítéséhez szükségesek. A `doc.bat` és a `docl.bat` file-ok a dokumentációt generálják. A `run.bat` pedig elindítja a lefordított `logistic.jar` file-t.

```
2005.03.14.  12:59               174 compile.bat
2005.03.25.  17:12               291 doc.bat
2005.03.25.  17:12               252 docl.bat
2005.03.14.  12:52                43 logistic.mf
2005.03.14.  12:59                22 run.bat
```

`./logistic` tartalma:

Az egyes file-ok a file nevének megfelelő osztályok implementációját tartalmazzák.

```
2005.03.28.  11:13            10 822 AssemblyLine.java
2005.03.28.  11:13             7 582 AssemblyLinePackageInfo.java
2005.03.28.  11:13             7 594 DeliveryPlan.java
2005.03.28.  11:13             7 496 DeliveryPlanItem.java
2005.03.28.  11:13            13 019 DistributingStation.java
2005.03.28.  11:13             7 250 Game.java
2005.03.28.  11:13             6 088 Ground.java
2005.03.28.  11:13             6 529 InDeliveryPlan.java
2005.03.28.  11:13             5 730 IndicatorLamp.java
2005.03.28.  11:13             1 369 IPackageCarrier.java
2005.03.28.  11:13             6 875 OutDeliveryPlan.java
2005.03.28.  11:13             9 073 Package.java
2005.03.28.  11:13             6 919 PackageInfo.java
2005.03.28.  11:13             7 768 PointOfAssemblyLine.java
2005.03.28.  11:13             9 250 Points.java
2005.03.28.  11:13            10 909 Transporter.java
2005.03.28.  11:13             8 274 TransporterEntryPoint.java
2005.03.28.  11:13             7 277 TransporterExitPoint.java
2005.03.28.  11:13            10 135 TransporterJoinPoint.java
```

`./logistic/skeleton` tartalma:

A `Skeleton.java` a skeleton főosztálya, ez vezérli a tesztelést. A `UseCaseTest.java` az egyes tesztesetek ősosztálya. A `UCTestCase*.java` file-ok pedig a "Skeleton terv" c. dokumentációban szereplő pályák megvalósítói.

```
2005.03.28.  11:13             11 133 Skeleton.java
2005.03.28.  11:13              3 999 UCTestCase1.java
2005.03.28.  11:13              2 364 UCTestCase2.java
2005.03.28.  11:13              1 948 UCTestCase3.java
2005.03.28.  11:13              1 715 UCTestCase4.java
2005.03.28.  11:13              3 277 UCTestCase5.java
2005.03.28.  11:13              1 239 UCTestCase6.java
2005.03.28.  11:13                222 UseCaseTest.java
```

`./logisticskeleton` tartalma:

A `LogisticGame.java` a program főosztálya, ebben az állapotában csak a Skeleton-t indítja el.

```
2005.03.28.  11:13                488 LogisticGame.java
```

5.3. A skeleton használata

A program először felkínálja a "Skeleton terv" c. dokumentációban leírt 6 pályát, mint 6 tesztesetet. Az egyes tesztesetekben pontosan az említett dokumentációban felsorolt use-case-eket vizsgálhatjuk meg, összehasonlítva a kollaborációs diagramokkal.

Először választani kell egy számot 1-6-ig, hogy melyik pályán akarunk tesztelni. Ha a programból ki akarunk lépni, akkor itt 0-át kell megadni.

Az egyes pályákon általában elég Enter-eket nyomni, a program mindig ki fogja írni, hogy éppen melyik use-case tesztesete forog fenn. A program bizonyos esetekben feltehet eldöntendő kérdéseket is, ezekre igennel(`y`) vagy nemmel(`n`) válaszolhatunk.

A program mindig jelzi, amikor be- illetve kilép egy függvényből. A jobb áttekinthetőség kedvéért, ha egy függvényen belül újabb függvényhívás történik, akkor ezen függvények be- és kilépésének jelzését beljebb kezdi a progam.

Az objektumok a könnyebb követhetőség érdekében egyedi sorszámot kapnak, amely funkció kizárólag a skeletonra (esetleg teszteléshez a prototípusra) terjed ki. A program későbbi állapotában már nem fognak szerepelni. Amikor objektum jön létre, akkor a konstruktorba belépés jelzésénél a sor végén szögletes zárójelben megjelenik a sorszám, amit a létrejövő objektum megkap. Ez a sorszám ezután minden olyan operáció meghívásában szerepelni fog (ugyancsak szögletes zárójelben), amely hívás az adott objektumra vonatkozik. Az 1-5 teszteseteknél a program nem írja ki a konstruktorok meghívását azért, hogy a felhasználó a tényleges use-case-ekre tudjon koncentrálni, de segítségképpen a teszt elején a legfontosabb objektumok sorszámait kilistázza.

A Java nyelv szigorú szabályai miatt nem teljesen konzisztens a konstruktorba be- illetve kilépés jelzése, ugyanis a `super()` hívás előtt nem szerepelhet semmi más kód. Ez azt jelenti, hogy ha a program a például következőket írja ki a képernyőre:

```
 -> DeliveryPlan.DeliveryPlan(long aCount)  [1]
 <- DeliveryPlan.DeliveryPlan(long aCount)
 -> InDeliveryPlan.InDeliveryPlan(long itemCount)  [2]
 <- InDeliveryPlan.InDeliveryPlan(long itemCount)
```

akkor ez valójában a következőképpen hajtódik végre:

```
 -> InDeliveryPlan.InDeliveryPlan(long itemCount)  [2]
     -> DeliveryPlan.DeliveryPlan(long aCount)  [1]
     <- DeliveryPlan.DeliveryPlan(long aCount)
```

```
<- InDeliveryPlan.InDeliveryPlan(long itemCount)
```

A skeleton elkészítése után a kiírásokat összehasonlítva a kollaborációs diagramokkal azt tapasztaltuk, hogy majdnem minden esetben visszakaptuk azt, amit vártunk. Egy-két olyan eset fordult csupán elő, hogy néhány hívás nem szerepel a kollaborációs diagramokon, holott a skeletonban meghívásra kerülnek. Ez is a skeleton szükségességét bizonyítja, hiszen így tudjuk ellenőrizni azt, hogy a tervezés sikeres volt-e.

A kollaborációs diagramokon szereplő «`destroy`» hívások viszont nem szerepelnek a skeletonban, mert a Java nem támogatja az explicit felszabadítást a szemétgyűjtő mechanizmus miatt. Ezek a hívások általában null pointeres értékadást jelentenek.

A skeleton egyszerűbb működtetéséhez néhány attribútum-lekérdező függvényt is implementálni kellett, hogy például ne kelljen mindig explicit típuskasztolást végezni. Ezek a függvények azonban az áttekinthetőség kedvéért nem írják ki azt, ha be- illetve kiléptünk belőlük.

5.4. Értékelés

A csapattagok az eddig elvégzett munkából, ha nem is minden részfeladat esetén, egyenlő mértékben vették ki a részüket, ezért egyhangúlag a pontok egyenlő elosztása mellett döntöttünk.

A csapat hamar összerázódott és kiderült, hogy a csapattagok képesek hatékonyan együttműködni. A csapaton belül hamar kialakult egy rend a feladatok megoldását illetően, így minden csapattag a neki legkellemesebb munkát kapta meg és végezte el. Az első pár hét alatt az is kiderült, hogy a csapattagok komolyan veszik feladatukat, tudásuk legjavát adják. A belső határidőket mindenki igyekezett megtartani, így minden feladatot sikerült határidőre elvégezni.

Kiderült az is, hogy érdemes volt a dokumentáció készítéséhez LaTeX formátumot választani, mert így a mechanikus feladatokat (Napló, Objektumkatalógus) gyakorlatilag egy karakter gépelése nélkül generáltatni tudtuk. Illetve ha változás történik a modellen, ugyanilyen könnyen újrageneráltathatjuk azt. Ebben nagy segítségünkre van az Ameos program, amely scriptnyelvének segítségével az UML modellből egyszerűen készíthetünk olyan kódot, amire szükségünk van (ez lehet Java, C++, LaTeX és még sok más).

6. fejezet

Prototípus koncepciója

6.1. Prototípus interface definíció

6.1.1. Pálya leíró fileformátum

```
/*
 multi line comment
*/
// one line comment

// az elosztóállomás létrehozása:
DistibutingStation

// a talaj létrehozása:
Ground

// <length> hosszú, <speed> sebességű futószalag létrehozása:
AssemblyLine <length> <speed>

/* futószalag csatlakozási pont létrehozása, amely az <assemblyline> sorszámú
 futószalag <pos> pozíciójára mutat:*/
PointOfAssemblyLine <assemblyline> <pos>

// váltó létrehozása:
Points

// belépési pont létrehozása:
EntryPoint

// kilépési pont létrehozása:
ExitPoint

/* beszállító teherautó létrehozása <itemcount> darab csomaggal,
 és csatlakoztatása az <entrypoint> indexű belépési ponthoz:*/
TransporterIn <itemcount> <entrypoint>
   /* a szállítási terv egy elemének létrehozása, amelyhez <kind> típusú
    <id> azonosítójú csomag tartozik, opcionális <decaytime>
    miliszekundumokban megadott romlási idővel:*/
   DeliveryPlanItem <id> <kind> [<decaytime>]
```

```
/* kiszállító teherautó létrehozása <itemcount> darab várt csomaggal,
és csatlakoztatása az <exitpoint> indexű kilépési ponthoz,*/
ha az <itemcount> értéke *, akkor véletlenszerű a darabszám*/
TransporterOut <itemcount> <exitpoint>
   /* a szállítási terv egy elemének létrehozása, amelyhez <kind> típusú
    csomag tartozik:  (ha a <kind> értéke *, akkor véletlenszerű a típus)*/
   DeliveryPlanItem <kind>

// két objektum csatlakoztatása:
Connect [EntryPoint <index>|AssemblyLine <index>|Points <index>]
        [ExitPoint <index>|AssemblyLine <index>|Points <index>|
        PointOfAssemblyLine <index>|Ground]
```

6.1.2. Proto bemenet fileformátum

```
/*
 multi line comment
*/
// one line comment

// 1 lépés időtartama milliszekundumokban:
StepTime <miliseconds>

// szint betöltése:
LoadLevel "<filename>"

// a szint elindítása:
Start

// kilépés a programból:
Exit

// a <points> sorszámú váltó átállítása:
ShiftThePoints <points>

// az <assemblyline> sorszámú futószalag sebességének növelése:
IncAssemblyLineSpeed <assemblyline>
// az <assemblyline> sorszámú futószalag sebességének csökkentése:
DecAssemblyLineSpeed <assemblyline>

// léptetés, a <stepcount> lépésszám opcionális:
Step [<stepcount>]

// megjegyzés kiírása //<annotation> formátumban:
Remark <annotation>

// az aktuális állapot kiírása:
ShowState

// új játék kezdése:
Hack Game StartNewGame
// következő szintre lépés:
Hack Game SwitchToNextLevel

// a szerzett pontok beállítása:
Hack Credits <credits>

// a lepakolásig hátralévő idő beállítása az <index> belépési pontnál:
Hack EntryPoint TimeToPackDown <index> <miliseconds>

// az <index> sorszámú futószalag sebességének <speed>-re állítása:
Hack AssemblyLine Speed <index> <speed>

// a csomag megtartási idejének beállítása az <index> váltónál:
Hack Points HoldTime <index> <miliseconds>
// az <index> sorszámú váltó <direction> irányba állítása:
Hack Points Direction <index> <direction>

// csomag manuális létrehozása <id> azonosítóval:
Hack Package Create <id> [AssemblyLine <index> <pos>]|Points <index>] <kind>
                    [<decaytime>]
// csomag összetörése:
Hack Package Smash <id>
// csomag felrobbantása:
Hack Package Explode <id>
```

6.1.3. Proto kimenet fileformátum

```
/*
 multi line comment
*/
// one line comment

/* A program futása során folyamatosan keletkeznek eventek,
 ezek a következők lehetnek:  */

// a bemenet feldolgozásának eseményei:
Error Invalid [EntryPoint|Points|Package] <index> referenced.
Error Syntax error:  <command>

// a játék eseményei:
Event StartGame // a játék elkezdése
Event ExitGame // kilépés a játékból
Event NextLevel <level> // szintlépés
Event Load <filename> // file betöltése
Event Credit <delta> <credits> // pontok módosítása
Event Step // lépés
    /*
    <state>
    */
Event ShowState // az aktuális állapot kiírása
    /*
    <state>
    */

// a belépési pont eseményei:
Event EntryPoint PackDownPackage <entrypoint> <kind> // lepakolás
Event EntryPoint DisconnectTransporter <entrypoint> // teherautó lecsatlakoztatása

// a kilépési pont eseményei:
Event ExitPoint PackUpPackage <exitpoint> <waitedkind> <arrivedkind>// felpakolás
Event ExitPoint DisconnectTransporter <exitpoint> // teherautó lecsatlakoztatása

// a csomag eseményei:
Event Package Explode <package> // felrobbant
Event Package Smash <package> // összetört
Event Package StartDecay <package> // elkezd megromlani
Event Package StopDecay <package> // nem romlik tovább

// a futószalag eseményei:
Event AssemblyLine PackageFall <assemblyline> <package> // csomag érkezett
Event AssemblyLine SetSpeed <assemblyline> <speed> // a sebesség állítása
Event AssemblyLine NotifyDestroyed <assemblyline> <package> // csomag felrobban

// a váltó eseményei:
Event Points PackageFall <points> <package> // csomag érkezett
Event Points ShiftThePoints <points> // váltó állítása
Event Points NotifyDestroyed <points> <package> // csomag felrobbant

// az indikátorlámpa eseményei:
Event Lamp ChangeState <exitpoint> <kind> // új állapot

// a föld eseményei:
Event Ground PackageFall <package> // csomag földre esett
```

```
//<state> = <game> + <graph>
//    <game> = game <level> <credits>
//    <graph> = graph + {<entrypoint>} + {<exitpoint>} +
//                      {<assemblyline>} + {<points>}
//      <entrypoint> = entrypoint <id> + [<transporter>]
//      <exitpoint> = exitpoint <id> + [<transporter>]
//      <transporter> = transporter <completedcount> + {<item>}
//      <item> = item <kind>
//      <assemblyline> = assemblyline <id> <speed> + {<package>}
//      <points> = points <id> <direction> + <package>
//      <package> = package <id> <kind> <decaytime> [<position>]
```

6.2. Tesztelési terv

A program a standard inputon várja a parancsokat, a kimenet pedig a standard output lesz. Teszteléshez ezeket átirányítjuk file-okra.

A be- és kimeneti parancsformátumok úgy lettek meghatározva, hogy azok teljes mértékben elősegítsék a tesztelést. Ez a teljes program tesztelését, és a funkciók külön-külön való vizsgálatát is magában foglalja. A bemeneten egyszerűen, szövegesen lehet megadni a tesztelési forgatókönyvet. A program helyes működését a kimeneten lehet majd nyomonkövetni és ellenőrizni.

6.2.1. A bemeneten előforduló hibák kezelése

Egyszerű szintaktikai hiba: ezt a hibát nem lehet figyelmen kívül hagyni. A program futása egy hibaüzenettel leáll.
Hibás állománynév: ez is végzetes hiba. A program hibaüzenettel leáll.

6.2.2. A tesztelés menete

A program különböző funkcióinak tesztelésére létrehozunk tesztforgatókönyveket. Ezeket lefuttatjuk, majd elemezzük, hogy a kimeneten kapott eredmények megfelelnek-e a bemenet alapján vártaknak. A teszteket többször kell elvégezni, hiszen a programnak ugyanolyan bemenetekre ugyanúgy kell reagálnia.

Fontos szempont az is, hogy több környezetben és különböző gépeken is tesztelésre kerüljön a program. Java alatti fejlesztésről lévén szó, a platformfüggetlenség megtartása nem nehéz feladat, de mivel sokfajta modul található a nyelv alatt, mégis ésszerű lehet a sokfajta környezetben folyó futtatási teszt - bár átfogó tesztelést elég csupán egy gépen folytatni -, hiszen így ráakadhatunk olyan apróbb környezeti hibákra, melyeket specifikálva a felhasználót már nem érheti meglepetés.

A kimenet helyességének ellenőrzéséhez egy egyszerű Java programot fogunk használni, amely soronként összehasonlít két állományt.

6.2.3. A szükséges tesztforgatókönyvek

Inicializálás
Egy játék elindítása. Pályabetöltés és az objektumok helyes létrehozása.
A felhasználói beavatkozásra történő kilépés tesztelése.
A következő pályára lépés tesztelése.

Csomag a váltóban
A váltók állítása.
Csomag ráesése a váltóra.
Csomag ráesése a váltóra, amikor abban már van csomag, a váltóban lévő csomag összetörése.
Csomag váltóban tartózkodása egy ideig, majd továbbítása.

Csomag a futószalagon
A futószalag sebességének állítása.
Csomag ráesése a futószalagra annak egyik pontján keresztül.
Csomag ráesése a futószalagra, amikor a szalagon már van csomag, a futószalagon lévő csomag megmarad.

Csomag ráesése a futószalagra, amikor a szalagon már van csomag, a futószalagon lévő csomag összetörése.
Csomag ráesése a futószalagra, amikor két csomag van a szalagon, az egyik csomag összetörése.
Csomag ráesése a futószalagra, amikor két csomag van a szalagon, mindkét csomag összetörése.
Csomag végigfutása a futószalagon, két különböző sebességnél.

Csomag összetörése
Nem romlandó csomag törik össze futószalagon
Romlandó csomag törik össze futószalagon.
Nem romlandó csomag törik össze váltóban.
Romlandó csomag törik össze váltóban.
Nem romlandó csomag törik össze kilépési pontnál.
Romlandó csomag törik össze kilépési pontnál.
Nem romlandó csomag földre esése, a csomag összetörése.
Romlandó csomag földre esése, a csomag összetörése.

Csomag lepakolása
Nem romlandó csomag lepakolása a beszállító teherautóról a belépési pontra.
Romlandó csomag lepakolása a beszállító teherautóról a belépési pontra.
Nem utolsó csomag lepakolása a beszállító teherautóról a belépési pontra.
Utolsó csomag lepakolása a beszállító teherautóról a belépési pontra.
Minden beszállító teherautó szállítási terve kiürül.

Csomag felpakolása
Csomag érkezése a kilépési ponthoz, ha a ponthoz csatlakozó teherautó a csomag típusával azonos csomagot vár.
Csomag érkezése a kilépési ponthoz, ha a ponthoz csatlakozó teherautó a csomag típusával nem azonos csomagot vár.
Nem romlandó csomag felpakolása a kilépési pontról a kiszállító teherautóra.
Romlandó csomag felpakolása a kilépési pontról a kiszállító teherautóra.
Nem utolsó csomag felpakolása a kilépési pontról a kiszállító teherautóra.
Utolsó csomag felpakolása a kilépési pontról a kiszállító teherautóra.
Minden kiszállító teherautó szállítási terve megtelik.

Csomag megromlása
Nem utolsó csomag romlandósági idejének lejárta futószalagon.
Utolsó csomag romlandósági idejének lejárta futószalagon.
Csomag romlandósági idejének lejárta váltón.

6.3. Változtatások a követemények módosulása miatt

Ezentúl a futószalagok sebességének állíthatónak kell lennie. Mivel a modellben már felvettük a futószalag sebességet statikus konstans attribútumként, csak annyit kell tennünk, hogy ezentúl az attribútum nem statikus és nem konstans lesz. Továbbá lesz két függvény, `long getSpeed()`, amely lekérdezi a futószalag sebességét és `void setSpeed(long aSpeed)`, amely beállítja azt. A `MAX` értéket egy privát statikus konstans attribútumként vesszük fel, értéke még függ a teszteléstől. A sikeres tervezésnek tudható be, hogy csak ilyen apró változtatást kell elvégezni a modellen.

7. fejezet

Részletes tervek

7.1. Változtatások a modellen

A tesztelhetőség érdekében az aktív objektumok új metódust kaptak: `void step(long dt)`. Ez a metódus fogja tartalmazni azokat a parancsokat, amelyeket végre kell hajtani egy lépés alatt. Az aktív objektumok a kész software-ben külön szállal rendelkeznek, amelyek ezt a `step()` függvényt fogják meghívni, így elkerülhető a kód duplázódása és egyszerűbb lett a tesztelhetőség. A `DistributingStation` osztály egy `void load(String fileName)` metódust is kapott, amely majd egy pályát tölt be a megadott file-ból. A `Package` osztályból hiányzott az az attribútum, amely azt jelezte, hogy a csomag éppen romlik-e. Ezt pótoltuk, neve `decaying` lett, típusa `boolean`. A teherautó `createInDeliveryPlan()` és `createOutDeliveryPlan()` metódusai ezentúl nem `void`-dal, hanem `InDeliveryPlan` illetve `OutDeliveryPlan` értékkel térnek vissza, hogy a `DistributingStation` osztály `load()` metódusa módosítani tudjon a szállítási terv elemein. Ehhez a `DeliveryPlanItem` kapott egy `void setKind(long aKind)` metódust is, amely beállítja az adott szállítási-tervelem típusát.

7.2. Objektumok és metódusok tervei

7.2.1. PointOfAssemblyLine

PointOfAssemblyLine.setTarget()
Beállítja a cél futószalagot és a pozíciót, ahova a csomagokat továbbítani kell.

PointOfAssemblyLine.packageFall()
Az érkező csomagot továbbítja a beállított futószalagra. Ez annak `packageFall()` metódusának meghívásával teszi meg.

7.2.2. AssemblyLine

AssemblyLine.AssemblyLine()
Létrehozza a csomaginformációkat tároló tömböt.

AssemblyLine.packageFall()
A futószalag csomagot ezen a metóduson keresztül kap meg. Végigmegy az összes csomaginformáción, és ha az érkező csomag ütközik valamelyikkel, akkor annak meghívja a `smash()` metódusát, majd törli a csomaginformációkból ezt az elemet. Miután minden futószalagon lévő ele-

met megvizsgált, az érkező csomagnak beállítja a hordozóját, annak `setCarrier()` függvényének meghívásával. Végül létrehoz egy csomaginformációt ehhez a csomaghoz és beteszi ezt a csomaginformációkat tároló tömbbe.

AssemblyLine.setEndPoint()
Beállítja a futószalag végpontját, ahova a csomagok leesnek, amikor a futószalag végére érnek.

AssemblyLine.getPackageInfo()
Visszaadja az adott indexű csomaginformációt.

AssemblyLine.forwardPackage()
Törli az adott indexű csomag csomaginformációit a tömbből és törli a csomag szállítóját annak `setCarrier()` függvényének meghívásával, amely `null` paramétert kap. Ezután továbbítja az adott indexű csomagot a végpontra, annak `packageFall()` metódusán keresztül.

AssemblyLine.getSpeed()
Visszaadja a futószalag sebességét.

AssemblyLine.setSpeed()
Beállítja a futószalag sebességét.

AssemblyLine.step()
Minden csomaginformációnál beállítja a `getPosition` lekérdezéssel kapott értékből kiszámolt új pozíciót a `setPosition()` meghívásával. Ha egy csomag túlmegy a futószalag hosszán, akkor azt továbbítja a `forwardPackage` metódussal.

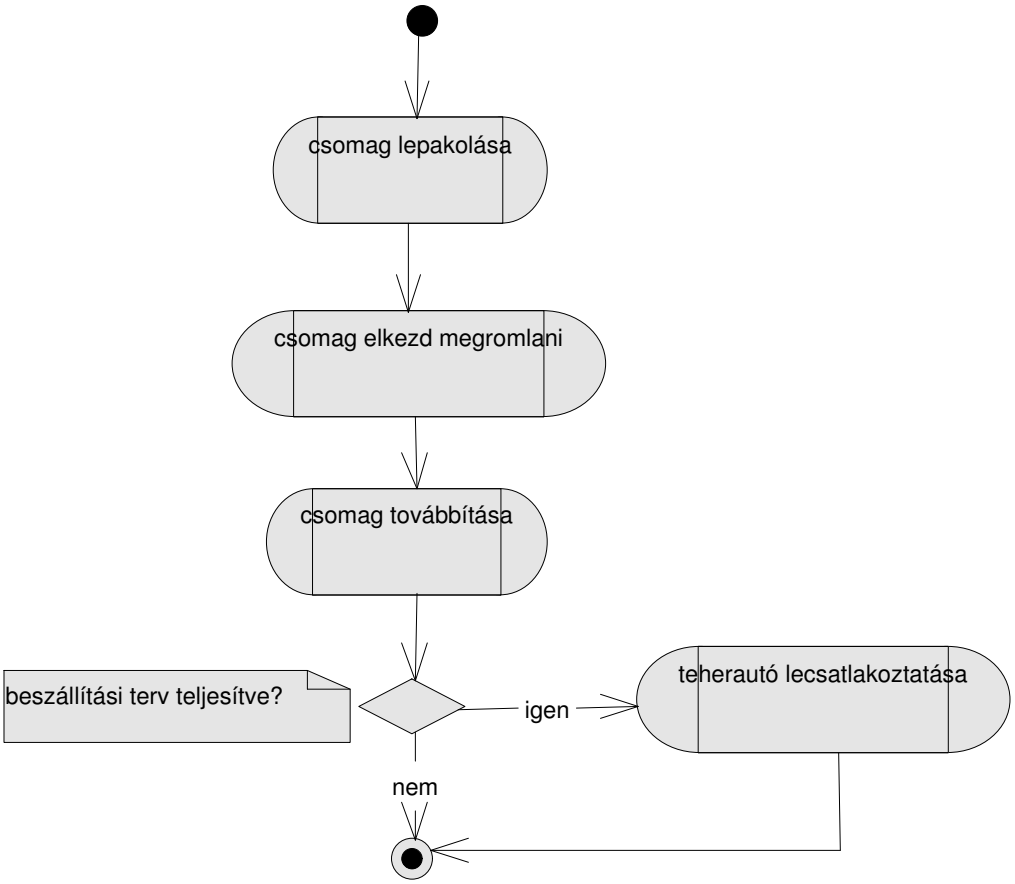
7.2.3. TransporterEntryPoint

TransporterEntryPoint.canConnectTransporter()
Visszaadja, hogy egy teherautó csatlakozhat-e a belépési ponthoz. Ez akkor tehető meg, ha a teherautó rendelkezik nem teljesített beszállítási tervvel. Ezt az információt annak `hasInDeliveryPlan()` és `isInDeliveryPlanCompleted()` függvényeivel lehet lekérdezni.

TransporterEntryPoint.setJoinPoint()
Beállítja a csatlakozási pontot, ahova a csomagokat le kell pakolni.

TransporterEntryPoint.step()
Csökkenti a `timeToPackDown` számlálót, és ha ez elérte a nullát, akkor lepakol egy csomagot a `packDownPackage()` meghívásával. Ezután új véletlenszámot generál, amelyet a `timeToPackDown` számláló kap meg értékül.

TransporterEntryPoint.packDownPackage()
Ha van csatlakozott teherautó, akkor annak `packDownPackage()` függvényével lepakol egy csomagot. Ennek a csomagnak el kell kezdenie megromlani, ez a `startDecay()` meghívásával történik. A csomagot ezután a csatlakozási ponthoz továbbítja a `packageFall()` metóduson keresztül, végül ha a teherautó kiszállítási terve teljesült (`isInDeliveryPlanCompleted()` igazzal tér vissza), akkor lecsatlakoztatja a teherautót a `disconnectTransporter()` meghívásával.



7.1. ábra. Activity diagram a TransporterEntryPoint.packDownPackage() metódushoz

7.2.4. Package

Package.Package()
A konstruktor. Beállítja a típust és a megromlási időt.

Package.setCarrier()
Beállítja a csomag vivőjét.

Package.getSize()
Visszaadja a csomag méretét.

Package.getKind()
Visszaadja a csomag típusát.

Package.smash()
Összetöri a csomagot, megállítja a romlást a `stopDecay()` meghívásával. A `LogisticGame.addCredit()`-tel levonja az összetörésért járó pontot.

Package.explode()
Felrobbantja a csomagot, megállítja a romlást a `stopDecay()` meghívásával, majd értesíti a vivőt arról, hogy megsemmisült. Ez annak `notifyPackageDestroy` metódusával teszi meg. A `LogisticGame.addCredit()`-tel levonja a felrobbanásért járó pontot.

Package.startDecay()
A csomag elkezd megromlani.

Package.stopDecay()
A csomag nem romlik tovább.

Package.step()
Ha a csomag romlik, akkor csökkenti a romlási időt. Ha ez eléri a nullát, akkor az `explode()` metódussal felrobbantja.

7.2.5. AssemblyLinePackageInfo

AssemblyLinePackageInfo.AssemblyLinePackageInfo()
Konstuktor, beállítja a csomagot és annak pozícióját.

AssemblyLinePackageInfo.getPosition()
Visszaadja a csomag pozícióját.

AssemblyLinePackageInfo.setPosition()
Beállítja a csomag pozícióját.

AssemblyLinePackageInfo.collideWith()
Visszaadja, hogy a tárolt csomag ütközik-e a paraméterként megadott csomaggal, ez akkor történik meg, ha a két csomag távolsága kisebb, mint a két csomag méretei felének összege.

7.2.6. Points

Points.shiftToNextLine()
Ha a váltónak vannak kimenetei, akkor a váltót a következő állásba állítja. Ha ez az állás túlmegy az utolsó kimenet indexén is, akkor értéke ismét nulla lesz. Mindezt egy egyszerű maradékképzéssel lehet megvalósítani.

Points.addLine()
Új kimenetet ad a váltóhoz. Ha ez az első kimenet, amit hozzáadunk, akkor az aktuális váltóállás erre fog mutatni.

Points.forwardPackage()
Ha a váltón csomag tartózkodik, akkor törli a vivőjét a `setCarrier()` metódussal, `null` paraméterrel. Ezután továbbítja a váltón lévő csomagot az aktuális kimenetre annak `packageFall()` metódusával. Végül törli a csomaginformációt is.

Points.step()

Csökkenti a csomag továbbítási időt. Ha ez eléri a nullát, továbbítja a csomagot a `forwardPackage()` meghívásával.

Points.packageFall()
Ha a váltóban még van csomag, akkor azt összetöri annak `smash()` metódusával, majd törli a csomaginformációt. Az új csomagnak beállítja a vivőjét a `setCarrier()`-rel, és létrehoz neki egy csomaginformációt is.

7.2.7. TransporterExitPoint

TransporterExitPoint.canConnectTransporter()
Visszaadja, hogy egy teherautó csatlakozhat-e a kilépési ponthoz. Ez akkor tehető meg, ha a teherautó rendelkezik nem teljesített kiszállítási tervvel. Ezt az információt annak `hasOutDeliveryPlan()` és `isOutDeliveryPlanCompleted()` függvényeivel lehet lekérdezni.
TransporterExitPoint.packageFall()
A csomag megérkezik a kilépési ponthoz. Megállítja annak romlását a `stopDecay()` meghívásával. Ha nincs csatlakozott teherautó, akkor a `smash()`-sel összetöri azt. Ha van teherautó, akkor felpakolja arra a csomagot a `packUpPackage()` metódussal. Ha ez a teherautó megtelt (`isOutDeliveryPlanCompleted()`), akkor lecsatlakoztatja a `disconnectTransporter()` függvénnyel. Ha lecsatlakozott a teherautó, akkor az indikátor lámpát üres állásba állítja a `setState()` 0-paraméterrel való meghívásával. Amennyiben a teherautó még mindig a kilépési pontnál áll, akkor a `getWaitedPackageKind()` függvénye által visszaadott értéket állítja be a lámpa `setState()` metódusával.

7.2.8. PackageInfo

PackageInfo.PackageInfo()
Konstruktor, beállítja a csomagot, amiről az információt tárolni kell.

PackageInfo.getPackage()
Visszaadja a csomagot.

PackageInfo.hasPackage()
Visszaadja, hogy a paraméterként kapott csomagról tárol-e információt.

7.2.9. TransporterJoinPoint

TransporterJoinPoint.TransporterJoinPoint()
Konstruktor, beállítja az elosztóállomást.

TransporterJoinPoint.canConnectTransporter()
Megadja, hogy a teherautó csatlakozhat-e a ponthoz, hamis értékkel tér vissza. A leszármazott osztályokban felül van definiálva.

TransporterJoinPoint.connectTransporter()
Csatlakoztatja a teherautót.

TransporterJoinPoint.disconnectTransporter()

Lecsatlakoztatja a teherautót és értesíti erről az elosztóállomást a `notifyEmptyJoinPoint()` metóduson keresztül.

TransporterJoinPoint.isFree()
Igazat ad vissza, ha a ponthoz nem csatlakozik teherautó.

7.2.10. IndicatorLamp

IndicatorLamp.setState()
Beállítja a lámpa állapotát.

7.2.11. DeliveryPlan

DeliveryPlan.DeliveryPlan()
Konstruktor, létrehozza az adott elemszámú szállítási tervet.

DeliveryPlan.count()
Visszaadja a szállítási terv elemszámát.

DeliveryPlan.getItem()
Visszaadja a szállítási terv adott elemét.

DeliveryPlan.isCompleted()
Igazat ad vissza, ha a szállítási terv teljesítve van. Ez mindig hamissal tér vissza, a leszármazott osztályokban felüldefiniált függvény adhat majd igazat eredményül.

7.2.12. Transporter

Transporter.createInDeliveryPlan()
Létrehozza az adott elemszámú a beszállítási tervet.

Transporter.createOutDeliveryPlan()
Létrehozza az adott elemszámú a kiszállítási tervet.

Transporter.packDownPackage()
Ha van beszállítási terv, akkor lepakol egy csomagot a szállítási terv `packDownPackage()` metódusának meghívásával.

Transporter.packUpPackage()
Csomag érkezett a teherautóra. Ha van kiszállítási terv, akkor felpakolja a csomagot a szállítási terv `packUpPackage()` metódusának meghívásával.

Transporter.hasInDeliveryPlan()
Igazat ad vissza, ha van beszállítási terv.

Transporter.hasOutDeliveryPlan()
Igazat ad vissza, ha van kiszállítási terv.

Transporter.isInDeliveryPlanCompleted()

Igazat ad vissza, ha a beszállítási terv teljesítve van, ez akkor teljesül, ha van beszállítási terv és annak `isCompleted()` metódusa igazzal tér vissza.

Transporter.isOutDeliveryPlanCompleted()
Igazat ad vissza, ha a kiszállítási terv teljesítve van, ez akkor teljesül, ha van kiszállítási terv és annak `isCompleted()` metódusa igazzal tér vissza.

Transporter.getWaitedPackageKind()
Visszaadja a várt csomag típusát, ha van kiszállítási terv. Ezt a kiszállítási terv `getNextPackageKind()` függvényével kérdezi le.

7.2.13. DeliveryPlanItem

DeliveryPlanItem.DeliveryPlanItem()
Konstruktor.

DeliveryPlanItem.getKind()
Visszaadja a szállítási terv elemének típusát.

DeliveryPlanItem.setKind()

DeliveryPlanItem.getPackage()
Visszaadja a szállítási terv eleméhez rendelt csomagot.

DeliveryPlanItem.setPackage()
Hozzárendeli a csomagot a szállítási terv eleméhez.

7.2.14. InDeliveryPlan

InDeliveryPlan.InDeliveryPlan()
Konstruktor, létrehozza a beszállítási tervet. Először egy `super()` hívással szállítási terv készül adott elemszámmal, majd mindegyik elemhez egy csomagot rendel az egyes elemek `setPackage()` függvényét meghívva.

InDeliveryPlan.packDownPackage()
Lepakolja a következő csomagot. Ezt úgy teszi meg, hogy az aktuális beszállítási tervelemtől lekérdezi a csomagot a `getPackage()` hívással, majd törli a hozzárendelt csomagot a tervelemből a `setPackage()` hívással, amelyet `null` paraméterrel hív meg. Végül megnöveli a teljesített tételek számát, és visszaadja a lekérdezett csomagot.

InDeliveryPlan.isCompleted()
Megadja, hogy a beszállítási terv készen van-e. Ez akkor igaz, ha a teljesített tételek száma eléri a szállítási terv elemeinek számát.

7.2.15. OutDeliveryPlan

OutDeliveryPlan.OutDeliveryPlan()
Konstruktor, létrehozza a beszállítási tervet. Egy `super()` hívással készül el az adott elemszámú szállítási terv.

OutDeliveryPlan.packUpPackage()
A teherautóra csomag érkezett. Az aktuális kiszállítási tervelemhez rendeli a csomagot a `setPackage()` metódus segítségével, majd megnöveli a teljesített tételek számát. Ha az érkezett csomag típusa megegyezik a szállítási tervelem típusával, akkor plusz pont, ellenkező esetben pontlevonás jár ezért. Ezt a `Game.addCredit()` meghívásával érvényesíti.

OutDeliveryPlan.getNextPackageKind()
Visszaadja az éppen várt csomag típusát, ezt az aktuális tervelem `getKind()` hívásával kérdezi le.

OutDeliveryPlan.isCompleted()
Visszaadja, hogy a kiszállítási terv teljesítve van-e. Ez akkor igaz, ha a teljesített tételek száma eléri a szállítási terv elemeinek számát.

7.2.16. Ground

Ground.packageFall()
Összetöri a csomagit annak `smash()` metódusával.

7.2.17. DistributingStation

DistributingStation.DistributingStation()
Konstruktor, létrehozza az elosztóállomást, valamint a futószalagokat, váltókat, be- és kilépési pontokat, futószalag csatlakozási pontokat és a teherautókat tároló tömböket.

DistributingStation.notifyEmptyJoinPoint()

Az elosztóállomást értesítik arról, hogy a teherautóját egy be- vagy kilépési pont lecsatlakoztatta. Ellenőrzi azt, hogy van-e még foglalt be- illetve kilépési pont. Ezt az `isFree()` függvénnyel teszi meg. Ha minden belépési pont vagy minden kilépési pont üres, akkor a következő szintre kell lépni, amelyet a játék objektum `switchToNextLevel()` metódusának meghívásával ér el.

DistributingStation.load()
Felszabadítja az éppen aktuális pályát, majd betölti a paraméterül kapott file-ban definiált pályát.

DistributingStation.getGround()
Visszaadja a talajt.

DistributingStation.getAssemblyLine()
Visszaadja az adott indexű futószalagot.

DistributingStation.getPoints()
Visszaadja az adott indexű váltót.

DistributingStation.getEntryPoint()
Visszaadja az adott indexű belépési pontot.

DistributingStation.getExitPoint()
Visszaadja az adott indexű kilépési pontot.

DistributingStation.getTransporter()
Visszaadja az adott indexű teherautót.

DistributingStation.getPointOfAssemblyLine()
Visszaadja az adott indexű futószalag-pontot.

7.2.18. Game

Game.startNewGame()
Új játékot kezd. Az aktuális szintet 0-ra állítja, majd meghívja a `switchToNextLevel()` függvényt.

Game.exitGame()
A pálya felszabadítása, kilépés a játékból.

Game.addCredit()
A gyűjtött pontokhoz hozzáadja a paraméterül kapott pontszámot. Mivel statikus metódus, ezért bármely objektum el tudja érni, és így a pontokat is statikus attribútumként tároljuk.

Game.switchToNextLevel()
A következő szintre lép, megnövelve az aktuális szint sorszámát. Megszünteti az aktuális elosztóállomást, majd újat hoz létre. Végül betölti az új pályát – amennyiben még van – az elosztóállomás `load()` függvényén keresztül.

7.3. A tesztek részletes tervei

7.3.1. Inicializálás

Egy játék elindítása. Pályabetöltés és az objektumok helyes létrehozása.

Teszt célja:

Azt vizsgáljuk, hogy a program helyesen építi-e fel a pályát.

Megvalósítás:

A pályán minden lehetséges objektumból legalább egyet létrehozunk és a játékot elindítjuk.

Várt eredmény:

A pálya felépül és a játék elindul.

Lehetséges hibaforrások:

Az esetleges hiba a Game konstruktorában keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel StartNewGame.lvl
Start
Step
Hack Game StartNewGame
Step
Exit
```

A felhasználói beavatkozásra történő kilépés tesztelése.

Teszt célja:

Azt vizsgáljuk, mikor a játékból kilép a felhasználó.

Megvalósítás:

Elindítunk egy játékot, majd kilépünk belőle.

Várt eredmény:

A játék terminálódik.

Lehetséges hibaforrások:

A hiba a Game.exit() metódusában keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel ExitGame.lvl
Start
Step
Exit
```

A következő pályára lépés tesztelése.

Teszt célja:

Azt vizsgáljuk, mikor a következő pályára lépünk.

Megvalósítás:

A következő pályára lépés feltételeit teljesítjük.

Várt eredmény:

A játék a következő pályára lép.

Lehetséges hibaforrások:

Az esetleges hiba a DistributingStation.notifyEmptyJoinPoint() vagy a Game.switchToNextLevel() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel SwitchToNextLevel.lvl
Start
Hack EntryPoint TimeToPackDown 0 1500
Step 3
Exit
```

7.3.2. Csomag a váltóban

A váltók állítása.

Teszt célja:

Azt vizsgáljuk, amikor a váltókat állítjuk.

Megvalósítás:

Egy váltót fogunk állítani, és minden állásnál továbbítunk egy csomagot. Ezt többször megismételjük, hogy a váltó mutatója körbe érjen.

Várt eredmény:

A váltó mindig átáll a következő irányba, és arra továbbítja a csomagot.

Lehetséges hibaforrások:

Az esetleges hiba a Points.shiftToNextLine() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel ShiftThePoints.lvl
Start
Hack Package Create 1 Points[0]
Step
ShiftThePoints[0]
Hack Package Create 2 Points[0]
Step
ShiftThePoints[0]
Hack Package Create 3 Points[0]
Step
ShiftThePoints[0]
Hack Package Create 4 Points[0]
Step
Exit
```

Csomag ráesése a váltóra.

Teszt célja:

Azt vizsgáljuk, mikor a csomag ráesik a váltóra.

Megvalósítás:

A csomag a futószalagon halad a váltó felé, majd ráesik.

Várt eredmény: A csomag ráes

Lehetséges hibaforrások:

Az esetleges hiba a Points.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackageToPoints.lvl
Start
Hack Package Create 1 AssemblyLine[0] 0
Step 3
Exit
```

Csomag ráesése a váltóra, amikor abban már van csomag, a váltóban lévő csomag összetörése.

Teszt célja:

Azt vizsgáljuk, amikor a csomag ráesik egy váltóra, amin már van egy csomag.

Megvalósítás:

A váltón áll egy csomag, miközben a futószalgon halad egy másik csomag a váltó felé.

Várt eredmény:

A futószalagról érkező csomag összetöri a váltón tartozkodó csomagot, majd megjelenik a váltón.

Lehetséges hibaforrások:

Az esetleges hiba a Points.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackageToPointsSmash.lvl
Start
Hack Points HoldTime 0 5000
Hack Package Create 1 AssemblyLine[0] 90
Hack Package Create 2 Points[0]
Step 3
Exit
```

Csomag váltóban tartózkodása egy ideig, majd továbbítása.

Teszt célja:

Azt vizsgáljuk, mikor a csomag a váltón várakozik egy keveset aztán továbbhalad.

Megvalósítás:

A csomag a váltón tartózkodik.

Várt eredmény:

A csomag továbbhalad a váltó irányába miután várakozott egy keveset.

Lehetséges hibaforrások:

Az esetleges hiba a Points.step()-ben illetve a Points.forwardPackage() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PointsHoldTime.lvl
Start
Hack Points HoldTime 0 1000
Hack Package Create 1 Points[0]
Step 3
Exit
```

7.3.3. Csomag a futószalagon

A futószalag sebességének állítása.

Teszt célja:

Azt vizsgáljuk, mikor a futószalag sebességét állítjuk.

Megvalósítás:

A futószalg egy adott sebességgel halad, majd állítjuk a sebességét.

Várt eredmény:

A futószalag sebessége megváltozik, a csomag más sebességgel halad.

Lehetséges hibaforrások:

Az esetleges hiba az AssemblyLine.setSpeed() illetve az AssemblyLine.step() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel AssemblyLineSpeed.lvl
Start
Hack Package Create 1 AssemblyLine[0] 0
Step
Hack AssemblyLine Speed 0 25
Step 3
Exit
```

Csomag ráesése a futószalagra annak egyik pontján keresztül

Teszt célja:

Azt vizsgáljuk, mikor a csomag ráesik a futószalag egy pontjára.

Megvalósítás:

A csomag megérkezik a futószalag egy pontjához.

Várt eredmény:

A csomag ráesik a futószalagra.

Lehetséges hibaforrások:

Az esetleges hiba a PointOfAssemblyLine.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackageToPointOfAssemblyLine.lvl
Start
Hack Package Create 1 AssemblyLine[0] 25
Step 3
Exit
```

Csomag ráesése a futószalagra, amikor a szalagon már van csomag, a futószalagon lévő csomag megmarad.

Teszt célja:

Azt vizsgáljuk, mikor a futószalagra, amin már van egy csomag, ráesik még egy csomag.

Megvalósítás:

A futószalgon halad egy csomag, és ráesik a szalagra még egy csomag olyan helyre, ahol nem tud ütközni az ott tartózkodó csomaggal.

Várt eredmény:

A futószalagon mindkét csomag továbbhalad.

Lehetséges hibaforrások:

Az esetleges hiba az AssemblyLine.packageFall() ill. AssemblyLine.forwardPackage() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackageToAssemblyLineWithPackage.lvl
Start
Hack Package Create 1 AssemblyLine[0] 0
Step
Hack Package Create 2 AssemblyLine[0] 0
Step 3
Exit
```

Csomag ráesése a futószalagra, amikor a szalagon már van csomag, a futószalagon lévő csomag összetörése.

Teszt célja:

Azt vizsgáljuk, mikor a futószalagra ráesik egy csomag, úgy hogy az éppen alatta lévő csomag összetörik.

Megvalósítás:

A futószalagon van egy csomag, és ugyanoda ráesik egy másik csomag.

Várt eredmény:

A futószalagon tartozkodó csomag összetörik, az érkező csomag rákerül a futószalagra.

Lehetséges hibaforrások:

Az esetleges hiba a AssemblyLine.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackageToAssemblyLineWithPackageSmash.lvl
Start
Hack Package Create 1 AssemblyLine[0] 0
Hack Package Create 2 AssemblyLine[0] 0
Step 3
Exit
```

Csomag ráesése a futószalagra, amikor két csomag van a szalagon, az egyik csomag összetörése.

Teszt célja:

Azt vizsgáljuk, mikor a futószalagon 2 csomag van, és még egy csomag ráesik az egyikre.

Megvalósítás:

A futószalagon 2 csomag tartózkodik. Ráesik még egy csomag a futószalagra arra a pozícióra, ahol az egyik csomag található.

Várt eredmény:

A futószalagon tartozkodó 2 csomag közül az egyik összetörik, az érkező csomag rákerül a futószalagra.

Lehetséges hibaforrások:

Az esetleges hiba a AssemblyLine.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackageToAssemblyLineWith2Package1Smash.lvl
Start
Hack Package Create 1 AssemblyLine[0] 0
Hack Package Create 2 AssemblyLine[0] 25
Hack Package Create 3 AssemblyLine[0] 25
ShowState
Exit
```

Csomag ráesése a futószalagra, amikor két csomag van a szalagon, mindkét csomag összetörése.

Teszt célja:

Azt vizsgáljuk, mikor a futószalagra ráeső csomag összetöri a futószalagon tartózkodó 2 csomagot.

Megvalósítás:

A futószalagon 2 csomag tartózkodik. A futószalagra ráesik még egy csomag úgy, hogy mindkét csomaggal ütközzön.

Várt eredmény:

A futószalagon tartozkodó 2 csomag összetörik, az új csomag megjelenik a futószalagon.

Lehetséges hibaforrások:

Az esetleges hiba a AssemblyLine.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackageToAssemblyLineWith2Package2Smash.lvl
Start
Hack Package Create 1 AssemblyLine[0] 0
Hack Package Create 2 AssemblyLine[0] 30
Hack Package Create 3 AssemblyLine[0] 15
ShowState
Exit
```

Csomag végigfutása a futószalagon, két különböző sebességnél.

Teszt célja:

Azt vizsgáljuk, mikor a csomag két különböző sebeséggel fut végig a futószalagon.

Megvalósítás:

Két különböző sebességű futószalagunk van, mindegyiken egy-egy csomag. A két csomag különböző időpontban ér az egyes szalagok végére.

Várt eredmény:

A csomagok különböző időpontban érkeznek a szalag végéhez.

Lehetséges hibaforrások:

Az esetleges hiba az AssemblyLine.step() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackagesWithDifferentSpeed.lvl
Start
Hack Package Create 1 AssemblyLine[0] 0
Hack Package Create 2 AssemblyLine[1] 0
Step 5
Exit
```

7.3.4. Csomag összetörése

Nem romlandó csomag törik össze futószalagon

Teszt célja:

Azt vizsgáljuk, amikor nem romlandó csomag törik össze a futószalagon.

Megvalósítás:

A futószalagon egy nem romlandó csomag, annak kézi összetörése.

Várt eredmény:

A futószalagon lévő csomag összetörik.

Lehetséges hibaforrások:

Az esetleges hiba a Package.smash() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel SmashDecAssembly.lvl
Start
Hack Package Create 0 AssemblyLine[0] 0 0
Hack Package Smash 0
Step 1
Exit
```

Romlandó csomag törik össze futószalagon

Teszt célja:

Azt vizsgáljuk, amikor romlandó csomag törik össze a futószalagon.

Megvalósítás:

A futószalagon egy romlandó csomag, annak kézi összetörése.

Várt eredmény:

A futószalagon lévő csomag összetörik.

Lehetséges hibaforrások:

Az esetleges hiba a Package.smash() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel SmashNotDecAssembly.lvl
Start
Hack Package Create 0 AssemblyLine[0] 0 0 5000
Hack Package Smash 0
Step 1
Exit
```

Nem romlandó csomag törik össze váltóban

Teszt célja:

Azt vizsgáljuk, amikor nem romlandó csomag törik össze a váltóban.

Megvalósítás:

A váltóban egy nem romlandó csomag, annak kézi összetörése.

Várt eredmény:

A váltóban lévő csomag összetörik.

Lehetséges hibaforrások:

Az esetleges hiba a Package.smash() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel SmashNotDecPoints.lvl
Start
Hack Package Create 0 Points[0] 0
Hack Package Smash 0
Step 1
Exit
```

Romlandó csomag törik össze váltóban

Teszt célja:

Azt vizsgáljuk, amikor romlandó csomag törik össze a váltóban.

Megvalósítás:

A váltóban egy romlandó csomag, annak kézi összetörése.

Várt eredmény:

A váltóban lévő csomag összetörik.

Lehetséges hibaforrások:

Az esetleges hiba a Package.smash() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel SmashDecPoints.lvl
Start
Hack Package Create 0 Points[0] 0 5000
Hack Package Smash 0
Step 1
Exit
```

Nem romlandó csomag törik össze kilépési pontnál

Teszt célja:

Azt vizsgáljuk, amikor nem romlandó csomag törik össze a kilépési pontnál.

Megvalósítás:

A kilépési pont a futószalag végén, a futószalagon egy nem romlandó csomag, a kilépési ponthoz nem csatlakozik teherautó.

Várt eredmény:

A kilépési pontnál lévő csomag összetörik.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterExitPoint.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
Load SmashNotDecExit.lvl
Start
Hack Package Create 0 AssemblyLine[0] 99 0
Step 2
Exit
```

Romlandó csomag törik össze kilépési pontnál

Teszt célja:

Azt vizsgáljuk, amikor romlandó csomag törik össze a kilépési pontnál.

Megvalósítás:

A kilépési pont a futószalag végén, a futószalagon egy romlandó csomag, a kilépési ponthoz nem csatlakozik teherautó.

Várt eredmény:

A kilépési pontnál lévő csomag összetörik.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterExitPoint.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
Load SmashDecExit.lvl
Start
Hack Package Create 0 AssemblyLine[0] 99 0 5000
Step 2
Exit
```

Nem romlandó csomag földre esése, a csomag összetörése

Teszt célja:

Azt vizsgáljuk, amikor nem romlandó csomag esik a földre és törik össze.

Megvalósítás:

A futószalag a földhöz csatlakozik, a futószalagon egy nem romlandó csomag.

Várt eredmény:

A földre eső csomag összetörik.

Lehetséges hibaforrások:

Az esetleges hiba a Ground.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel SmashNotDecGround.lvl
Start
Hack Package Create 0 AssemblyLine[0] 99 0
Step 2
Exit
```

Romlandó csomag földre esése, a csomag összetörése

Teszt célja:

Azt vizsgáljuk, amikor romlandó csomag esik a földre és törik össze.

Megvalósítás:

A futószalag a földhöz csatlakozik, a futószalagon egy romlandó csomag.

Várt eredmény:

A földre eső csomag összetörik.

Lehetséges hibaforrások:

Az esetleges hiba a Ground.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel SmashDecGround.lvl
Start
Hack Package Create 0 AssemblyLine[0] 99 0 5000
Step 2
Exit
```

7.3.5. Csomag lepakolása

Nem romlandó csomag lepakolása a beszállító teherautóról a belépési pontra

Teszt célja:

Azt vizsgáljuk, amikor nem romlandó csomagot pakolunk le a belépési pontra.

Megvalósítás:

A belépési ponthoz egy beszállító teherautó csatlakozik, a szállítási tervben egy nem romlandó csomag.

Várt eredmény:

A szállítási tervben lévő csomag a belépési pontra kerül.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterEntryPoint.packDownPackage() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackDownNotDec.lvl
Start
Hack EntryPoint TimeToPackDown 0 0
Step 2
Exit
```

Romlandó csomag lepakolása a beszállító teherautóról a belépési pontra

Teszt célja:

Azt vizsgáljuk, amikor romlandó csomagot pakolunk le a belépési pontra.

Megvalósítás:

A belépési ponthoz egy beszállító teherautó csatlakozik, a szállítási tervben egy romlandó csomag.

Várt eredmény:

A szállítási tervben lévő csomag a belépési pontra kerül.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterEntryPoint.packDownPackage() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackDownDec.lvl
Start
Hack EntryPoint TimeToPackDown 0 0
Step 2
Exit
```

Nem utolsó csomag lepakolása a beszállító teherautóról a belépési pontra

Teszt célja:

Azt vizsgáljuk, amikor nem utolsó csomagot pakolunk le a teherautóról a belépési pontra.

Megvalósítás:

A belépési ponthoz egy beszállító teherautó csatlakozik, a szállítási tervben két csomag, az elsőt pakoljuk le.

Várt eredmény:

A szállítási tervben lévő csomag a belépési pontra kerül.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterEntryPoint.packDownPackage() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackDownNotLast.lvl
Start
Hack EntryPoint TimeToPackDown 0 600
Step 1
Exit
```

Utolsó csomag lepakolása a beszállító teherautóról a belépési pontra

Teszt célja:

Azt vizsgáljuk, amikor utolsó csomagot pakolunk le a belépési pontra.

Megvalósítás:

A belépési ponthoz egy beszállító teherautó csatlakozik, a szállítási tervben két csomag, mindegyiket lepakoljuk. A belépési ponthoz egy futószalag csatlakozik.

Várt eredmény:

A szállítási tervben lévő két csomag a belépési pontra kerül, a teherautó lecsatlakozik.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterEntryPoint.packDownPackage() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackDownLast.lvl
Start
Hack EntryPoint TimeToPackDown 0 600
Step 1
Step 1
Exit
```

Minden beszállító teherautó szállítási terve kiürül

Teszt célja:

Azt vizsgáljuk, amikor az összes tehearutóról az utolsó csomag is a belépési pontra kerül.

Megvalósítás:

Két belépési ponthoz egy-egy beszállító teherautó csatlakozik, a szállítási tervekben egy-egy csomag, ezeket egymás után lepakoljuk. Mindkét belépési ponthoz egy-egy futószalag csatlakozik.

Várt eredmény:

A szállítási tervekben lévő csomagok a belépési pontra kerülnek, a teherautók lecsatlakoznak és a program új szintet tölt be.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterEntryPoint.packDownPackage() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackDownFinish.lvl
Start
Hack EntryPoint TimeToPackDown 0 600
Hack EntryPoint TimeToPackDown 1 800
Step 1
Step 1
Exit
```

7.3.6. Csomag felpakolása

Csomag érkezése a kilépési ponthoz, ha a ponthoz csatlakozó teherautó a csomag típusával azonos csomagot vár

Teszt célja:

Azt vizsgáljuk, amikor a kilépési ponthoz egy megfelelő típusú csomag érkezik, a csomagot felpakoljuk a teherautóra.

Megvalósítás:

A kilépési pont a futószalag végén, a futószalagon a megfelelő típusú csomag, a kilépési ponthoz egy kiszállító teherautó csatlakozik.

Várt eredmény:

A csomag a kiszállító teherautó szállítási tervébe kerül, a játékos plusz pontot kap.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterExitPoint.packageFall() illetve a Transporter.packUpPackage() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackUpRight.lvl
Start
Hack Package Create 0 AssemblyLine[0] 99 0
Step 2
Exit
```

Csomag érkezése a kilépési ponthoz, ha a ponthoz csatlakozó teherautó a csomag típusával nem azonos csomagot vár

Teszt célja:

Azt vizsgáljuk, amikor a kilépési ponthoz egy nem megfelelő típusú csomag érkezik, a csomagot felpakoljuk a teherautóra.

Megvalósítás:

A kilépési pont a futószalag végén, a futószalagon a nem megfelelő típusú csomag, a kilépési ponthoz egy kiszállító teherautó csatlakozik.

Várt eredmény:

A csomag a kiszállító teherautó szállítási tervébe kerül, pont levonására kerül sor.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterExitPoint.packageFall() illetve a Transporter.packUpPackage() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackUpNotRight.lvl
Start
Hack Package Create 0 AssemblyLine[0] 99 1
Step 2
Exit
```

Nem romlandó csomag felpakolása a kilépési pontról a kiszállító teherautóra

Teszt célja:

Azt vizsgáljuk, amikor a kilépési ponthoz nem romlandó csomag érkezik, a csomagot felpakoljuk a teherautóra.

Megvalósítás:

A kilépési pont a futószalag végén, a futószalagon egy nem romlandó csomag, a kilépési ponthoz egy kiszállító teherautó csatlakozik.

Várt eredmény:

A csomag a kiszállító teherautó szállítási tervébe kerül.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterExitPoint.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackUpNotDec.lvl
Start
Hack Package Create 0 AssemblyLine[0] 99 0
Step 2
Exit
```

Romlandó csomag felpakolása a kilépési pontról a kiszállító teherautóra

Teszt célja:

Azt vizsgáljuk, amikor a kilépési ponthoz romlandó csomag érkezik, a csomagot felpakoljuk a teherautóra.

Megvalósítás:

A kilépési pont a futószalag végén, a futószalagon egy romlandó csomag, a kilépési ponthoz egy kiszállító teherautó csatlakozik.

Várt eredmény:

A csomag a kiszállító teherautó szállítási tervébe kerül.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterExitPoint.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackUpDec.lvl
Start
Hack Package Create 0 AssemblyLine[0] 99 0 5000
Step 2
Exit
```

Nem utolsó csomag felpakolása a kilépési pontról a kiszállító teherautóra

Teszt célja:

Azt vizsgáljuk, amikor a kilépési ponthoz nem utolsó csomag érkezik, a csomagot felpakoljuk a teherautóra.

Megvalósítás:

A kilépési pont a futószalag végén, a futószalagon egy csomag, amely a kilépési ponthoz érkezik. A csomag felkerül a teherautóra.

Várt eredmény:

A csomag a kiszállító teherautó szállítási tervébe kerül, de a teherautó nem csatlakozik le.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterExitPoint.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackUpNotLast.lvl
Start
Hack Package Create 0 AssemblyLine[0] 99 0
Step 1
Exit
```

Utolsó csomag felpakolása a kilépési pontról a kiszállító teherautóra

Teszt célja:

Azt vizsgáljuk, amikor a kilépési ponthoz utolsó csomag érkezik, a csomagot felpakoljuk a teherautóra.

Megvalósítás:

A kilépési pont a futószalag végén, a futószalagon két csomag, sorban a kilépési ponthoz érkeznek, a kilépési ponthoz egy kiszállító teherautó csatlakozik.

Várt eredmény:

A csomagok a kiszállító teherautó szállítási tervébe kerülnek, majd a teherautó lecsatlakozik.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterExitPoint.packageFall() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackUpLast.lvl
Start
Hack Package Create 0 AssemblyLine[0] 99 0
Hack Package Create 1 AssemblyLine[0] 49 0
Step 1
Step 1
Exit
```

Minden kiszállító teherautó szállítási terve megtelik

Teszt célja:

Azt vizsgáljuk, amikor az utolsó kiszállító teherautó szállítási terve is megtelik.

Megvalósítás:

Két kilépési pont egy-egy futószalag végén, a futószalagokon egy-egy csomag, ezek egymás után a kilépési pontokhoz érkeznek, a kilépési ponthokhoz egy-egy kiszállító teherautó csatlakozik.

Várt eredmény:

A csomagok a kiszállító teherautók szállítási terveibe kerülnek, a teherautók lecsatlakoznak és a program új szintet tölt be.

Lehetséges hibaforrások:

Az esetleges hiba a TransporterExitPoint.packageFall() illetve a DistributingStation.notifyEmptyJoinPoint() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel PackUpFinish.lvl
Start
Hack Package Create 0 AssemblyLine[0] 99 0
Hack Package Create 1 AssemblyLine[1] 49 0
Step 1
Step 1
Exit
```

7.3.7. Csomag megromlása

Nem utolsó csomag romlandósági idejének lejárta futószalagon

Teszt célja:

Azt vizsgáljuk, amikor nem utolsó csomag romlandósági ideje jár le a futószalagon.

Megvalósítás:

A futószalagon két csomag, az elsőnek lejár a romlandósági ideje.

Várt eredmény:

A futószalagon lévő csomag felrobban.

Lehetséges hibaforrások:

Az esetleges hiba a Package.explode() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel DecayNotLastAssembly.lvl
Start
Hack Package Create 0 AssemblyLine[0] 0 0
Hack Package Create 1 AssemblyLine[0] 50 0 600
Step 1
Exit
```

Utolsó csomag romlandósági idejének lejárta futószalagon

Teszt célja:

Azt vizsgáljuk, amikor utolsó csomag romlandósági ideje jár le a futószalagon.

Megvalósítás:

A futószalagon egy csomag, lejár a romlandósági ideje.

Várt eredmény:

A futószalagon lévő csomag felrobban.

Lehetséges hibaforrások:

Az esetleges hiba a Package.explode() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel DecayLastAssembly.lvl
Start
Hack Package Create 0 AssemblyLine[0] 0 0 600
Step 1
Exit
```

Csomag romlandósági idejének lejárta váltón

Teszt célja:

Azt vizsgáljuk, amikor a csomag romlandósági ideje lejár a váltóban.

Megvalósítás:

A váltóban egy csomag, lejár a romlandósági ideje.

Várt eredmény:

A váltóban lévő csomag felrobban.

Lehetséges hibaforrások:

Az esetleges hiba a Package.explode() metódusban keresendő.

A bemeti tesztfile:

```
StepTime 1000
LoadLevel DecayPoints.lvl
Start
Hack Points HoldTime 0 1200
Hack Package Create 0 Points[0] 0 600
Step 1
Exit
```

7.4. A tesztelést támogató programok tervei

Két file összehasonlítására egy egyszerű Java programot használunk, neve `Compare` lesz, és a `compare` package-ben található. Soronként összehasonlítja a parancssori paraméterként kapott két file-t, és amely soroknál különbséget észlel, azokat sorszámukkal együtt egymás alá kiírja.

Az összes teszteset lefuttatásához olyan batch file-t írunk, amely minden teszteseten végigmegy, majd összehasonlíttatja a fenti programmal a proto eredményét a várt kimenettel.

Neve `testall.bat`, szerkezete a következő lesz:

```
echo off
echo -----------------------------------------------------
echo testcase
echo -----------------------------------------------------
run <testcase.in >testcase.res
java compare.Compare testcase.res testcase.out
...
```

Ahol a `run` a skeleton beadásánál definiált batch-file, ez indítja el a programot. A `compare` pedig a fent definiált Java program.

A `testall.bat`-ot parancssorból a következő paranccsal fogjuk elindítani:

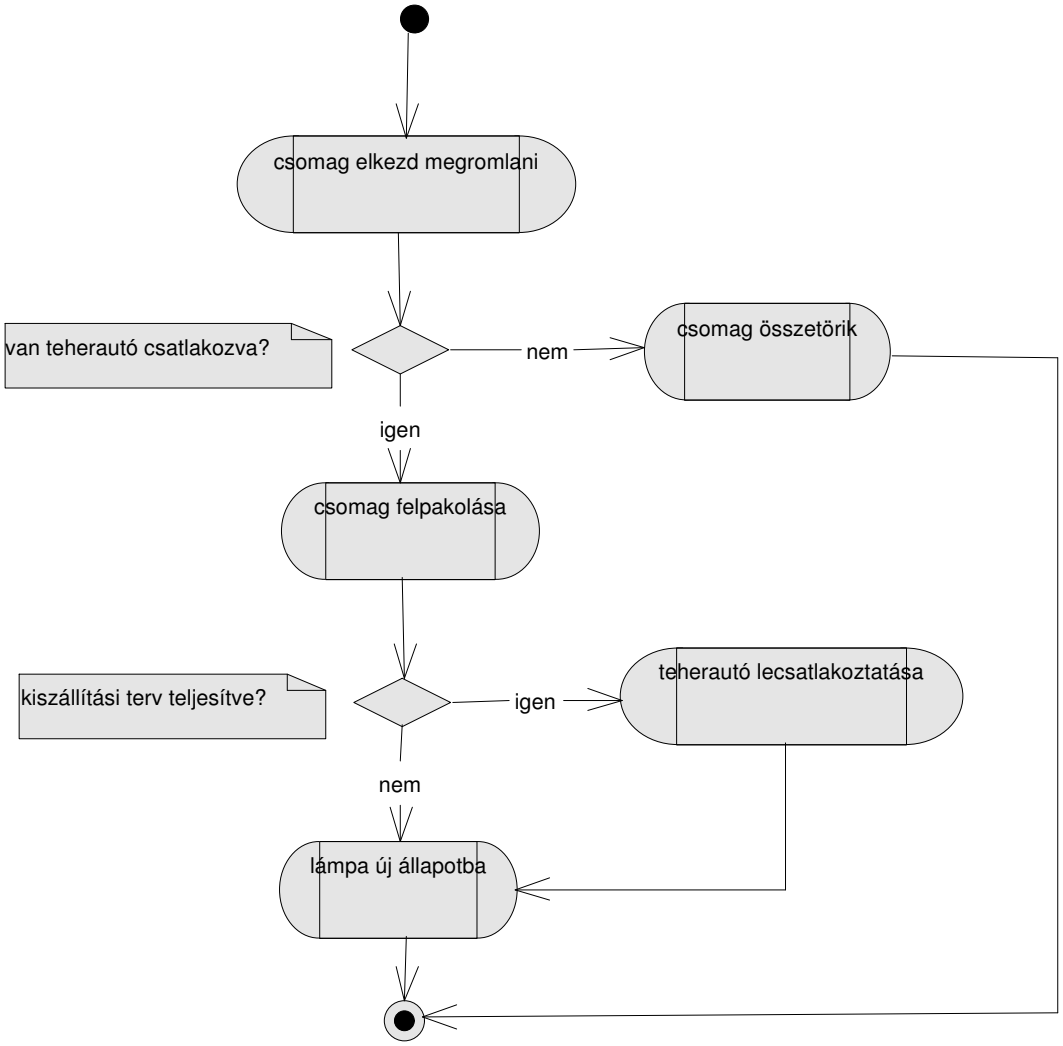
```
testall >testresults.txt
```

A keletkező `testresults.txt` file-t kell majd megvizsgálni, hogy a tesztesetek rendben lefutottak-e.

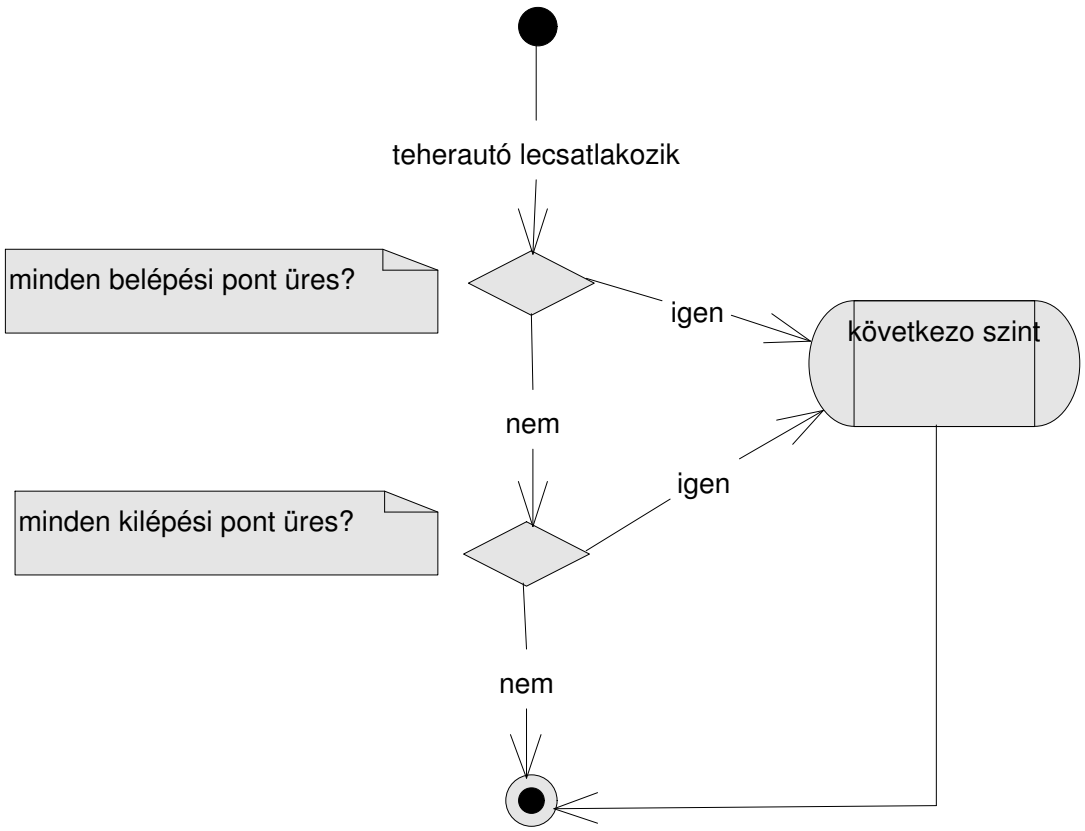
Hogy ne kelljen mindig minden tesztesetet lefuttatni, definiálunk egy `test.bat` file-t is, tartalma a következő:

```
echo off
echo -----------------------------------------------------
echo %1
echo -----------------------------------------------------
run <%1.in >%1.res
java compare.Compare %1.out %1.res
```

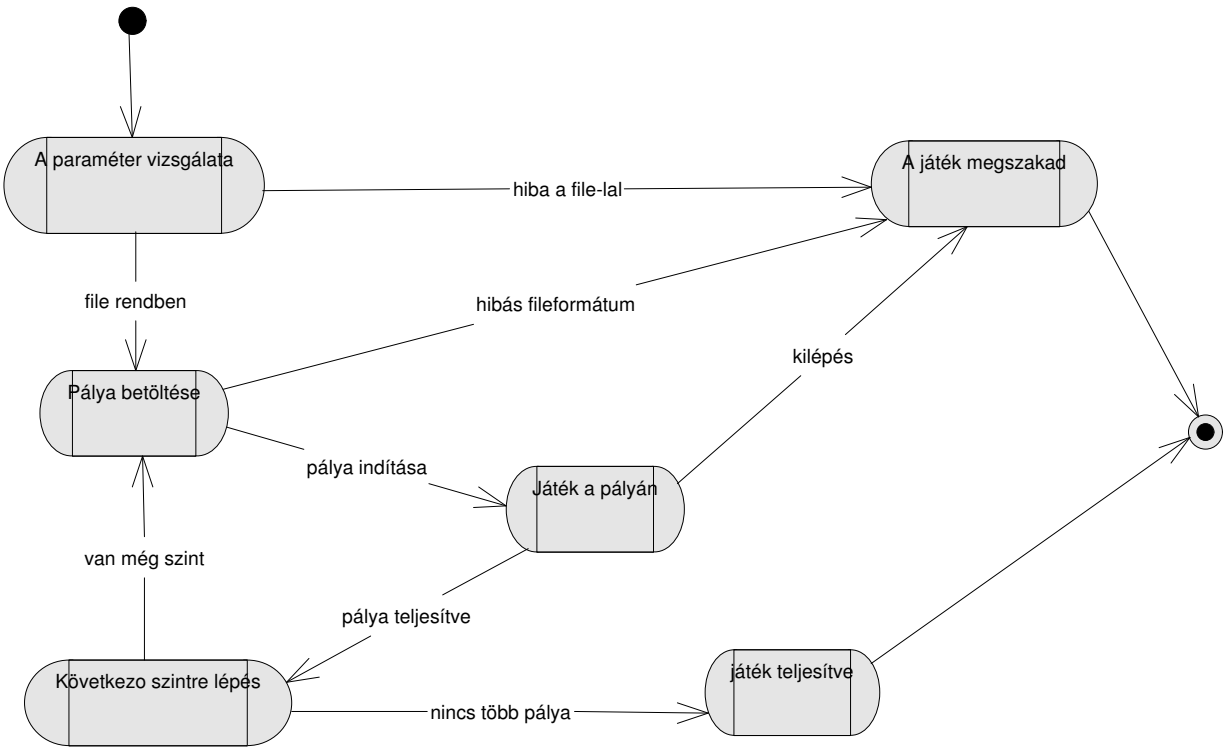
így csak a paraméterként kapott tesztesetet fogja lefuttatni.



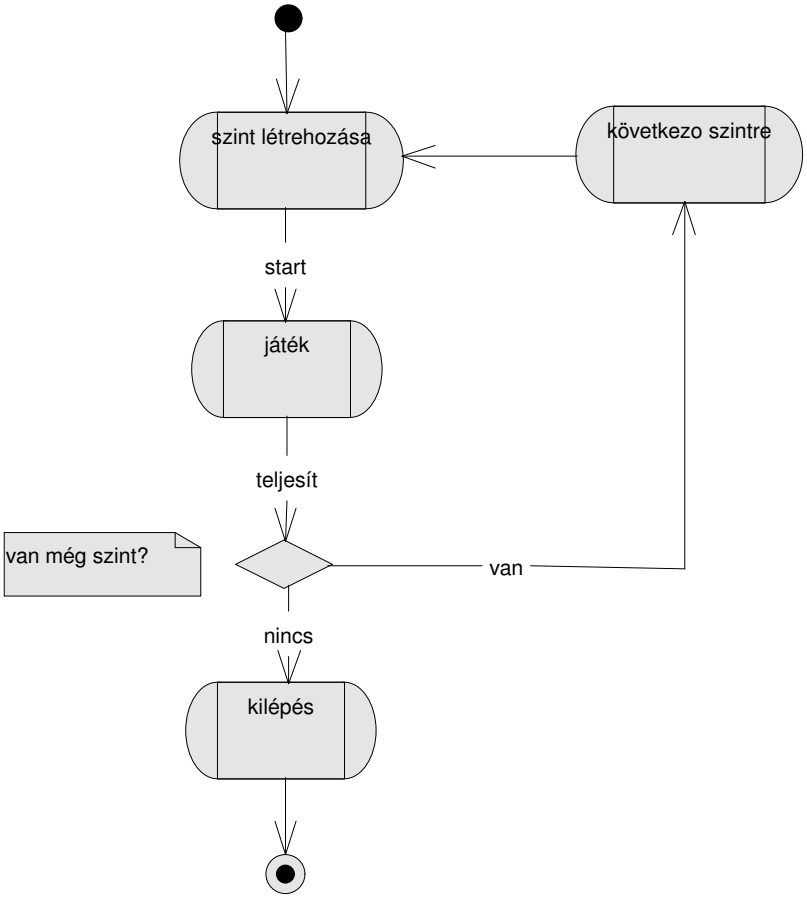
7.2. ábra. Activity diagram a TransporterExitPoint.packageFall() metódushoz



7.3. ábra. Activity diagram a DistributingStation.notifyEmptyJoinPoint() metódushoz



7.4. ábra. Statechart a játékhoz



7.5. ábra. Activity diagram a Game.switchToNextLevel() metódushoz

8. fejezet

A prototípus beadása

8.1. A proto fordítása és futtatása

8.1.1. A HSZK-ban DOS parancssorból

Mivel a skeleton fordításával a HSZK-ban problémák merültek fel, ezért a tesztelők munkáját megkönnyítjük azzal, hogy a HSZK-beli futtatásra elkészíttettük a speciális fordító és futtató file-okat.

1. A mellékelt zip file-t ki kell csomagolni egy könyvtárba.
2. A `setjavapath.bat` paranccsal beállítjuk a PATH környezeti változó értékét.
3. A `compile.bat` paranccsal lefordítjuk a forrásálományokat.
4. A program indítása a `run.bat hszk` paranccsal történik.
5. A egy teszteset lefuttatását a `test.bat hszk <PathToTestCase>` parancs végzi el, ahol a `<PathToTestCase>` értéke pl. `test\Decay\DecayLastAssembly\DecayLastAssembly`.
6. A teszt eredményét a `compare.bat <PathToTestCase>` paranccsal ellenőrizhetjük, ahol a `<PathToTestCase>` értéke pl. `test\Decay\DecayLastAssembly\DecayLastAssembly`.
7. Az összes teszteset lefuttatása a `testall.bat hszk` paranccsal történik.
8. A JavaDoc dokumentációt a `doc.bat` (internetkapcsolat hiányában a `docl.bat`) segítségével állíthatjuk elő. Ekkor létrejön egy Doc nevű könyvtár, amelyből az index.html oldalt webböngészőben megnyitva rendelkezésünkre áll egy jól használható objektumkatalógus. Az itt szereplő leírások a forráskódban is megtalálhatóak kommentek formájában.

8.1.2. DOS parancssorból

1. A mellékelt zip file-t ki kell csomagolni egy könyvtárba.
2. A `compile.bat` paranccsal lefordítjuk a forrásálományokat.
3. A program indítása a `run.bat <JAVAPATH>` paranccsal történik, ahol a `<JAVAPATH>` a Java könyvtára.
4. A egy teszteset lefuttatását a `test.bat <JAVAPATH> <PathToTestCase>` parancs végzi el, ahol a `<PathToTestCase>` értéke pl. `test\Decay\DecayLastAssembly\DecayLastAssembly`.
5. Az összes teszteset lefuttatása a `testall.bat <JAVAPATH>` paranccsal történik.

6. A teszt eredményét a `compare.bat <PathToTestCase>` paranccsal ellenőrizhetjük, ahol a `<PathToTestCase>` értéke pl. `test\Decay\DecayLastAssembly\DecayLastAssembly`.

7. A JavaDoc dokumentációt a `doc.bat` (internetkapcsolat hiányában a `docl.bat`) segítségével állíthatjuk elő. Ekkor létrejön egy Doc nevű könyvtár, amelyből az index.html oldalt webböngészőben megnyitva rendelkezésünkre áll egy jól használható objektumkatalógus. Az itt szereplő leírások a forráskódban is megtalálhatóak kommentek formájában.

8.1.3. DOS parancssor hiányában

1. A mellékelt zip file-t ki kell csomagolni egy könyvtárba.

2. A következő parancsokkal fordíthatjuk le a rendszert:
```
javac logistic/*.java logistic/proto/*.java
    logisticproto/*.java
jar cmf logistic.mf logistic.jar logistic/*.class
    logistic/proto/*.class logisticproto/*.class
```

3. A program indítása a `<JAVAPATH>\bin\java -jar logistic.jar -classpath <JAVAPATH>\lib\*.jar;<JAVAPATH>\jre\lib\*.jar; <JAVAPATH>\jre\lib\ext\*.jar;<JAVAPATH>\jre\javaws\*.jar` parancs kiadásával történik, ahol a `<JAVAPATH>` a Java könyvtára.

4. A JavaDoc dokumentációt a
```
javadoc -locale hu_HU -encoding iso-8859-2 -charset iso-8859-2
    -docencoding iso-8859-2
    -link http://java.sun.com/products/jdk/1.3/docs/api
    -d Doc -sourcepath logistic -private -author -windowtitle
    "Logistic game - Silent Hill" logistic/*.java
    logistic/proto/*.java logisticproto/*.java
```
internetkapcsolat hiányában pedig a
```
javadoc -locale hu_HU -encoding iso-8859-2 -charset iso-8859-2
    -docencoding iso-8859-2 -linkoffline -d Doc -sourcepath
    logistic -private -author -windowtitle
    "Logistic game - Silent Hill" logistic/*.java
    logistic/proto/*.java logisticproto/*.java
```
segítségével állíthatjuk elő.

8.2. A mellékelt álomány tartalma

Az elküldött zip file tartalma az alábbi listában szereplő file-ok. Két nagy könyvtár van: `LogisticTest`, a szálak nélküli, determinisztikusan tesztelhető program és a `LogisticThreadsTest`, a szálas, nem determinisztikus lefutású program. A két könyvtár majdnem ugyanazokat a file-okat tartalmazza, a `LogisticThreadsTest`-ből hiányoznak a teszteléshez szükséges tesztadatok és tesztfile-ok.

8.3. A determinisztikusan tesztelhető változat (LogisticTest könyvtár)

`./LogisticTest/` tartalma:

A `setjavapath.bat`,`compile.bat` és a `logistic.mf` a program fordításához, a `logistic.jar` file elkészítéséhez szükségesek. A `doc.bat` és a `docl.bat` file-ok a dokumentációt generálják. A `run.bat` pedig elindítja a lefordított `logistic.jar` file-t. A `test.bat` és a `testall.bat` a tesztelést végzik, a `compare.bat` pedig a teszt eredményének ellenőrzését.

```
2005.04.22.  19:03               220 compare.bat
2005.04.16.  14:36               188 compile.bat
2005.04.16.  13:44               285 doc.bat
2005.04.16.  13:44               246 docl.bat
2005.04.16.  13:44                40 logistic.mf
2005.04.21.  15:33               393 run.bat
2005.04.20.  16:36                52 setjavapath.bat
2005.04.22.  19:03               213 test.bat
2005.04.22.  19:04             4 929 testall.bat
```

`./LogisticTest/compare` tartalma:

A teszt kapott és várt eredményét összehasonlító Java program.

```
2005.04.16.  14:16             1˙717 Compare.java
```

`./LogisticTest/logistic` tartalma:

Az egyes file-ok a file nevének megfelelő osztályok implementációját tartalmazzák, szálak nélkül.

```
2005.04.16.  14:06             7˙435 AssemblyLine.java
2005.04.16.  11:29             3˙578 AssemblyLinePackageInfo.java
2005.04.16.  13:14             3˙686 DeliveryPlan.java
2005.04.16.  13:15             3˙779 DeliveryPlanItem.java
2005.04.20.  17:33            15˙802 DistributingStation.java
2005.04.16.  15:04             4˙683 Game.java
2005.04.16.  13:34             2˙929 Ground.java
2005.04.16.  11:53             2˙950 InDeliveryPlan.java
2005.04.16.  13:33             2˙837 IndicatorLamp.java
2005.04.16.  11:53             1˙342 IPackageCarrier.java
2005.04.16.  14:35             3˙277 OutDeliveryPlan.java
2005.04.16.  14:08             5˙609 Package.java
2005.04.16.  11:58             3˙050 PackageInfo.java
2005.04.16.  11:59             3˙512 PointOfAssemblyLine.java
2005.04.17.  16:24             5˙845 Points.java
2005.04.16.  13:21             5˙877 Transporter.java
2005.04.21.  16:23             5˙089 TransporterEntryPoint.java
2005.04.21.  16:08             3˙671 TransporterExitPoint.java
2005.04.20.  17:29             4˙647 TransporterJoinPoint.java
```

`./LogisticTest/logistic/levels` tartalma:

Három egyszerű pálya, azért kellenek, hogy a program el tudjon indulni és a tesztesetek rendben le tudjanak futni.

```
2005.04.21.  16:10               361 level1.lvl
2005.04.16.  18:08                62 level2.lvl
2005.04.16.  15:43                46 level3.lvl
```

`./LogisticTest/logistic/proto` tartalma:

A `Proto.java` a proto főosztálya, ez vezérli a tesztelést.

```
2005.04.21.  16:29            13˙761 Proto.java
```

`./LogisticTest/logisticproto` tartalma:

A `LogisticGame.java` a program főosztálya, ebben az állapotában csak a Proto-t indítja el.

```
2005.04.15.  18:08               535 LogisticGame.java
```

`./LogisticTest/test` tartalma:

A tesztesetek, a 8.7 pontban leírt könyvtárak, mindegyikben a megfelelő pálya (`.lvl`), a bemenet (`.in`), a várt kimenet (`.out`) és a prototípus által szolgáltatott eredmény (`.res`) találhatók.

8.4. A szálas változat (LogisticThreadsTest könyvtár)

`./LogisticThreadsTest/` tartalma:

A `setjavapath.bat`,`compile.bat` és a `logistic.mf` a program fordításához, a `logistic.jar` file elkészítéséhez szükségesek. A `doc.bat` és a `docl.bat` file-ok a dokumentációt generálják. A `run.bat` pedig elindítja a lefordított `logistic.jar` file-t.

```
2005.04.21.  20:36               162 compile.bat
2005.04.16.  13:44               285 doc.bat
2005.04.16.  13:44               246 docl.bat
2005.04.16.  13:44                40 logistic.mf
2005.04.21.  15:33               393 run.bat
2005.04.20.  16:36                52 setjavapath.bat
```

`./LogisticThreadsTest/logistic` tartalma:

Az egyes file-ok a file nevének megfelelő osztályok implementációját tartalmazzák, szálakkal megvalósítva.

```
2005.04.21.  18:09             7˙969 AssemblyLine.java
2005.04.16.  11:29             3˙578 AssemblyLinePackageInfo.java
2005.04.21.  19:15             3˙713 DeliveryPlan.java
2005.04.21.  19:14             3˙820 DeliveryPlanItem.java
2005.04.21.  19:44            15˙732 DistributingStation.java
2005.04.21.  17:49             4˙822 Game.java
2005.04.16.  13:34             2˙929 Ground.java
2005.04.21.  19:16             3˙084 InDeliveryPlan.java
2005.04.16.  13:33             2˙837 IndicatorLamp.java
2005.04.16.  11:53             1˙342 IPackageCarrier.java
2005.04.16.  14:35             3˙277 OutDeliveryPlan.java
2005.04.21.  19:46             6˙266 Package.java
2005.04.16.  11:58             3˙050 PackageInfo.java
2005.04.16.  11:59             3˙512 PointOfAssemblyLine.java
2005.04.21.  18:09             6˙377 Points.java
2005.04.16.  13:21             5˙877 Transporter.java
2005.04.21.  19:03             5˙571 TransporterEntryPoint.java
2005.04.21.  16:08             3˙671 TransporterExitPoint.java
2005.04.20.  17:29             4˙647 TransporterJoinPoint.java
```

`./LogisticThreadsTest/logistic/levels` tartalma:

Három pálya hosszú futószalagokkal a játszhatóság érdekében.

```
2005.04.21.  18:59              655 level1.lvl
2005.04.21.  18:51              526 level2.lvl
2005.04.21.  18:59              526 level3.lvl
```

`./LogisticThreadsTest/logistic/proto` tartalma:

A `Proto.java` a proto főosztálya, ez kezeli a parancsokat.

```
2005.04.21.  19:45           13˙488 Proto.java
```

`./LogisticThreadsTest/logisticproto` tartalma:

A `LogisticGame.java` a program főosztálya, ebben az állapotában csak a Proto-t indítja el.

```
2005.04.21.  17:32              535 LogisticGame.java
```

8.5. A szálak nélküli proto használata (LogisticTest)

A proto elindítás után várja a parancsokat. Ezek a parancsok azok lehetnek, amelyek a Prototípus koncepciója c. dokumentum 6.1.2. Proto bemenet fileformátum pontjában szerepelnek. A csomag létrehozó parancs következő változással használható:

Hack Package Create <id> [AssemblyLine <index> <pos>]|Points <index>] <kind> [<decaytime>]

továbbá az `AssemblyLine` és a `Points` után az indexet közvetlenül utánuk egy szögletes zárójelben kell megadni. Ez vonatkozik a pálya file-ra is: ahol egy pályaelem indexe közvetlenül az elemnév után szerepel, akkor azt szóköz nélkül, közvetlenül a név után szögletes zárójelben kell megadni. Ezt a változtatást az átláthatóság érdekében tettük. Például:

Hack Package Create 1 AssemblyLine[0] 500 2 200

amely egy 1-es azonosítójú csomagot hoz létre a 0-ás futószalag 500-as pozícióján. A csomag típusa 2-es, megromlási ideje 200.

8.5.1. Teszt pálya példa

```
DistributingStation
Ground
AssemblyLine 10000 1000
Connect AssemblyLine[0] Ground
```

8.5.2. Teszt bemenet példa

```
StepTime 1000
LoadLevel test\Decay\DecayLastAssembly\DecayLastAssembly.lvl
Start
Hack Package Create 0 AssemblyLine[0] 0 0 600
Step 1
Exit
```

8.5.3. Teszt kimenet példa

```
Event NextLevel 1
Event Load levels/level1.lvl
Event Load test\Decay\DecayLastAssembly\DecayLastAssembly.lvl
Event Start
Event AssemblyLine PackageFall 0 0
Event Package StartDecay 0
Event Step
Event Package Explode 0
Event Package StopDecay 0
Event Credit -5 -5
Event AssemblyLine NotifyDestroyed 0 0
/*
game 1 -5
graph
assemblyline 0 1000
*/
Event Exit
```

8.6. A szálas proto használata (LogisticThreadsTest)

A proto elindítás után várja ugyanazokat a parancsokat, amelyeket a szál nélküli, kivéve a `Step` és `StepTime <ms>` parancsokat. A kiírásnál is hiányzik az `Event Step` és a hozzátartozó állapotkiírás. A példapályákban a futószalagokat elég hosszúra vettük és a belépési pontok lepakolási idejét is megnöveltük, hogy a játék konzolról vezérelhető legyen. Ajánlott a `ShowState`, `IncAssemblyLineSpeed <index>`, `DecAssemblyLineSpeed <index>` és a `ShiftThePoints <index>` parancsok használata. Begépelésüknél az események lehet, hogy széttördelik a parancsokat, de ezzel nem kell törődni, a parancsértelmező külön szálon fut, és végre fogja azt hajtani.

8.7. A tesztelés eredményeinek összefoglalása

8.7.1. Inicializálás

./test/Initial/StartNewGame

Teszt: Egy játék elindítása. Pályabetöltés és az objektumok helyes létrehozása.
Hiba: Nem volt.
Javítás: -

./test/Initial/ExitGame

Teszt: A felhasználói beavatkozásra történő kilépés tesztelése.
Hiba: Nem volt.
Javítás: -

./test/Initial/SwitchToNextLevel

Teszt: A következő pályára lépés tesztelése.
Hiba: A belépési pontok nem pakolják le a csomagot.
Javítás: Az TransporterEntryPoint.step() metódusban a lepakolási idő számítását kellett javítani.

8.7.2. Csomag a váltóban

./test/PackageOnPoints/ShiftThePoints
Teszt: A váltók állítása.
Hiba: Nem volt.
Javítás: -

./test/PackageOnPoints/PackageToPoints
Teszt: Csomag ráesése a váltóra.
Hiba: Nem volt.
Javítás: -

./test/PackageOnPoints/PackageToPointsSmash
Teszt: Csomag ráesése a váltóra, amikor abban már van csomag, a váltóban lévő csomag összetörése.
Hiba: Nem volt.
Javítás: -

./test/PackageOnPoints/PointsHoldTime
Teszt: Csomag váltóban tartózkodása egy ideig, majd továbbítása.
Hiba: Nem volt.
Javítás: -

8.7.3. Csomag a futószalagon

./test/PackageOnAssemblyLine/AssemblyLineSpeed
Teszt: A futószalag sebességének állítása.
Hiba: Nem volt.
Javítás: -

./test/PackageOnAssemblyLine/PackageToPointOfAssemblyLine
Teszt: Csomag ráesése a futószalagra annak egyik pontján keresztül.
Hiba: Nem volt.
Javítás: -

./test/PackageOnAssemblyLine/PackageToAssemblyLineWithPackage
Teszt: Csomag ráesése a futószalagra, amikor a szalagon már van csomag, a futószalagon lévő csomag megmarad.
Hiba: Nem volt.
Javítás: -

./test/PackageOnAssemblyLine/PackageToAssemblyLineWithPackageSmash
Teszt: Csomag ráesése a futószalagra, amikor a szalagon már van csomag, a futószalagon lévő csomag összetörése.
Hiba: Nem volt.
Javítás: -

./test/PackageOnAssemblyLine/PackageToAssemblyLineWith2Package1Smash
Teszt: Csomag ráesése a futószalagra, amikor két csomag van a szalagon, az egyik csomag összetörése.
Hiba: Nem volt.
Javítás: -

./test/PackageOnAssemblyLine/PackageToAssemblyLineWith2Package2Smash
Teszt: Csomag ráesése a futószalagra, amikor két csomag van a szalagon, mindkét csomag összetörése.
Hiba: Nem volt.
Javítás: -

./test/PackageOnAssemblyLine/PackagesWithDifferentSpeed

Teszt: Csomag végigfutása a futószalagon, két különböző sebességnél.
Hiba: Nem volt.
Javítás: -

8.7.4. Csomag összetörése

./test/Smash/SmashNotDecAssembly

Teszt: Nem romlandó csomag törik össze futószalagon
Hiba: Nem volt.
Javítás: -

./test/Smash/SmashDecAssembly

Teszt: Romlandó csomag törik össze futószalagon
Hiba: Nem volt.
Javítás: -

./test/Smash/SmashNotDecPoints

Teszt: Nem romlandó csomag törik össze váltóban
Hiba: Nem volt.
Javítás: -

./test/Smash/SmashDecPoints

Teszt: Romlandó csomag törik össze váltóban
Hiba: Nem volt.
Javítás: -

./test/Smash/SmashNotDecExit

Teszt: Nem romlandó csomag törik össze kilépési pontnál
Hiba: Nem volt.
Javítás: -

./test/Smash/SmashDecExit

Teszt: Romlandó csomag törik össze kilépési pontnál
Hiba: A csomag nem tört össze.
Javítás: A TransporterExitPoint.packageFall() metódus nem foglalkozott azzal az esettel, amikor nincs teherautó a kilépési pontnál, így ez javítva lett.

./test/Smash/SmashNotDecGround

Teszt: Nem romlandó csomag földre esése, a csomag összetörése
Hiba: Nem volt.
Javítás: -

./test/Smash/SmashDecGround

Teszt: Romlandó csomag földre esése, a csomag összetörése
Hiba: Nem volt.
Javítás: -

8.7.5. Csomag lepakolása

./test/PackDown/PackDownNotDec

Teszt: Nem romlandó csomag lepakolása a beszállító teherautóról a belépési pontra
Hiba: Nem volt.
Javítás: -

./test/PackDown/PackDownDec

Teszt: Romlandó csomag lepakolása a beszállító teherautóról a belépési pontra
Hiba: Nem volt.
Javítás: -

./test/PackDown/PackDownNotLast

Teszt: Nem utolsó csomag lepakolása a beszállító teherautóról a belépési pontra
Hiba: Nem volt.
Javítás: -

./test/PackDown/PackDownLast

Teszt: Utolsó csomag lepakolása a beszállító teherautóról a belépési pontra
Hiba: Az egyik belépési pontnál lévő teherautó kiürült és a program szintet váltott, pedig nem kellett volna.
Javítás: A TransporterJoinPoint.isFree() függvényének visszatérési értéke mindig true volt, ki lett javítva, hogy ellenőrizze a teherautó meglétét.

./test/PackDown/PackDownFinish

Teszt: Minden beszállító teherautó szállítási terve kiürül
Hiba: Nem volt.
Javítás: -

8.7.6. Csomag felpakolása

./test/PackUp/PackUpRight

Teszt: Csomag érkezése a kilépési ponthoz, ha a ponthoz csatlakozó teherautó a csomag típusával azonos csomagot vár
Hiba: Nem volt.
Javítás: -

./test/PackUp/PackUpNotRight

Teszt: Csomag érkezése a kilépési ponthoz, ha a ponthoz csatlakozó teherautó a csomag típusával nem azonos csomagot vár
Hiba: Nem volt.
Javítás: -

./test/PackUp/PackUpNotDec

Teszt: Nem romlandó csomag felpakolása a kilépési pontról a kiszállító teherautóra
Hiba: Nem volt.
Javítás: -

./test/PackUp/PackUpDec

Teszt: Romlandó csomag felpakolása a kilépési pontról a kiszállító teherautóra
Hiba: Nem volt.
Javítás: -

./test/PackUp/PackUpNotLast

Teszt: Nem utolsó csomag felpakolása a kilépési pontról a kiszállító teherautóra
Hiba: Nem volt.
Javítás: -

./test/PackUp/PackUpLast

Teszt: Utolsó csomag felpakolása a kilépési pontról a kiszállító teherautóra
Hiba: Nem volt.
Javítás: -

./test/PackUp/PackUpFinish

Teszt: Minden kiszállító teherautó szállítási terve megtelik
Hiba: Nem volt.
Javítás: -

8.7.7. Csomag megromlása

./test/Decay/DecayNotLastAssembly

Teszt: Nem utolsó csomag romlandósági idejének lejárta futószalagon
Hiba: Nem volt.
Javítás: -

./test/Decay/DecayLastAssembly

Teszt: Utolsó csomag romlandósági idejének lejárta futószalagon
Hiba: Nem volt.
Javítás: -

./test/Decay/DecayPoints

Teszt: Csomag romlandósági idejének lejárta a váltón
Hiba: A csomag nem robbant fel.
Javítás: A hiba a proto parancsfeldolgozójában volt, a megromlási időket nem vette figyelembe, így ezt kellett kijavítani.

8.8. Értékelés

A belső határidőket mindenki igyekezett megtartani, így minden feladatot sikerült határidőre elvégezni. A csapattagok az eddig elvégzett munkából egyenlő mértékben vették ki a részüket, ezért egyhangúlag a pontok egyenlő elosztása mellett döntöttünk.

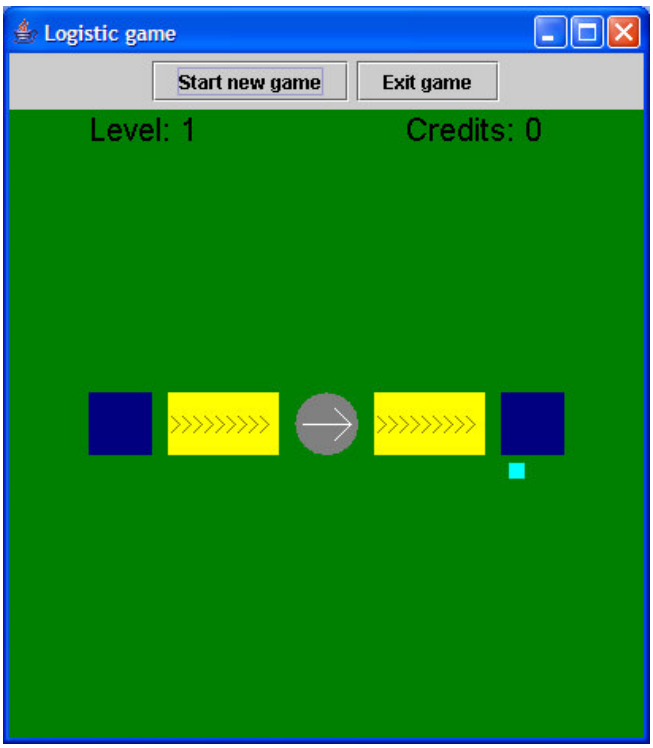
Továbbra is nagy segítségünkre volt az Ameos UML modellező program, hiszen a skeleton nagy részét fel tudtuk használni a prototípus készítésében. Ehhez csak úgy kellett újragenerálni a forráskódot, hogy kihagytuk a loggolási információkat, így a skeletonban implementált részeket újrahasznosíthattuk.

A tesztelésnél főleg olyan hibák jöttek elő, amelyeket apróbb elírások, egy-egy kisebb részeset kihagyása okozott, de mégis olyan hatásaik voltak, amelyet egy grafikus, többszálú programnál nagyon nehezen lehetne megkeresni. A parancssori tesztelés nagyban megkönnyítette a hibák okának és helyének felfedezését. Bebizonyosodott, hogy a tesztelés a szoftverfejlesztés fontos részét kell, hogy képezze.

9. fejezet

Grafikus felület specifikálása

9.1. A menürendszer, a kezelői felület grafikus képe



9.1. ábra. A grafikus felület terve

Az ábrán a tervezett grafikus felület képe látható. Az "Exit game" gombbal lehet kilépni a játékból, a "Start new game" gombbal pedig új játékot lehet kezdeni. A váltókat úgy tudjuk állítani, hogy egérrel rákattintunk. A futószalagok sebességét növelni tudjuk, ha a futószalag végére kattintunk, illetve csökkenteni tudjuk, ha az elejére kattintunk. Ha egy teherautó teljesítette a szállítási tervét, akkor eltűnik a pályáról. Ha egy csomag összetörik vagy felrobban, akkor az is eltűnik.

9.2. A felület működésének elve, a grafikus rendszer architektúrája

A megjeleníthető objektumoknak saját osztályai vannak. A business objektumok és a grafikus objektumok egyaránt tartalmaznak egymásra referenciát, hogy könnyen el tudják egymást érni. A business objektumok azonban csak egy interface-en keresztül láthatják a grafikus objektumokat, így lehet elérni azt, hogy a grafikus felület teljes egészében leválasztható legyen az alkalmazói programról.

A business objektumoknál beállítható és lekérdezhető a hozzátartozó grafikus objektum referenciája, de grafikus objektum metódusát busines objektum nem hívja meg. A business objektumok kizárólag a `ViewController` metódusait használják, azokat is csak egy interface-en keresztül. Az eredeti modellen tehát csak ennyi változtatást kell végezni. Minden business osztály, amelynek van megjeleníthető megfelelője kap egy `graphicObject:ViewInterface` attribútumot. Ezen kívül lesz egy `setGraphicObject(aGraphicObject:ViewInterface)` és egy `getGraphicObject():ViewInterface` metódus is. A `ViewController` példánya egy publikus statikus objektum lesz, hogy minden business objektum el tudja érni.

A megjelenítést és az egéresemények kezelését a `ViewController` irányítja (külön szálon fut), minden grafikus objektumra tartalmaz referenciát, így tudja majd őket kirajzoltatni (`draw()`) illetve átadni nekik az információt, hogy egérkattintás történt (`click()`). Egy szint betöltésekor az `addViewObject()` metódussal hozzáadjuk a megjelenítendő objektumokat, az előzőeket pedik a `clear()`-rel töröljük. Ha egy csomagot lepakolunk, akkor az `addViewObject()`-tel azt is hozzáadjuk a listához, amikor pedig felrobban, összetörik vagy felpakolják, akkor a `removeViewObject()`-tel törölni lehet a sorból. A megjelenítendő objektumlista első eleme mindig a föld lesz, utána következnek a pályaelemek és végül a csomagok. Ez biztosítja a megfelelő takarást.

A grafikus objektumok azért tartalmaznak referenciát a business objektumokra, hogy ne kelljen feleslegesen információt cserélni a két réteg között. Így bármilyen adatra is van szüksége a grafikus felületnek, azt azonnal elérheti. A megjelenítő objektumokat a konstruktorukban inicializáljuk, mezőiket később megváltoztatni nem lehet.

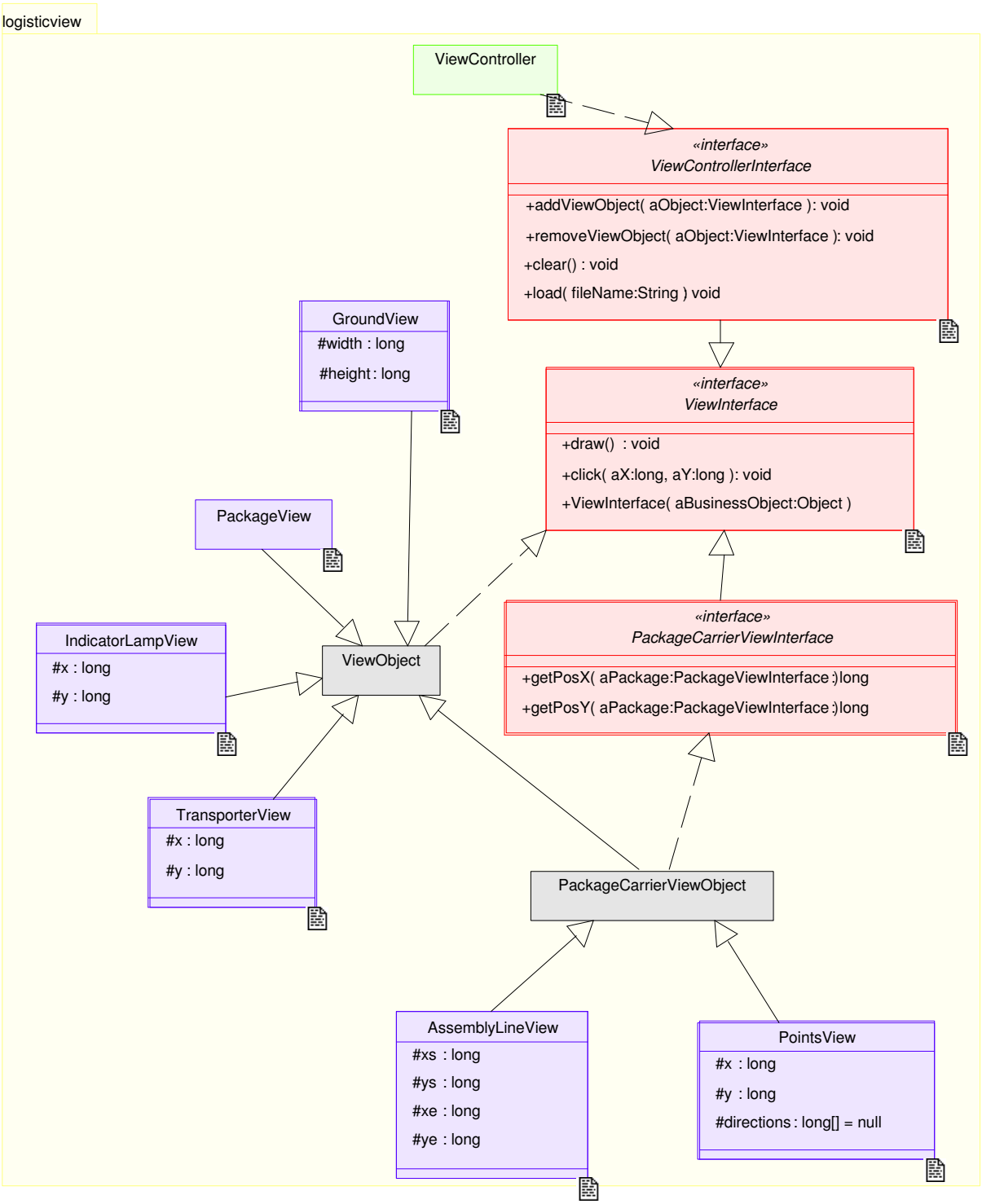
Ha egy csomagot kell kirajzolni, akkor az először lekérdezi a business csomag szállítójának grafikus megfelelőjét, majd a `PackageCarrierViewInterface`-en keresztül a saját grafikai koordinátáját. Ehhez átadja a saját business objektumának referenciáját a megfelelő grafikus objektumnak. Ha ez futószalag, akkor az a saját business objektumától lekérdezi a csomag pozícióját és ennek alapján meghatározza a koordinátát.

A grafikus váltó inicializáláskor kap egy listát a lehetséges irányokról, amerre állhat. Ezek az irányok fokokban vannak megadva, ezek alapján rajzolja ki a váltó a továbbhaladási irányt mutató nyilat.

A többi objektum kirajzolása illetve a váltón lévő csomag kirajzolása egyértelmű.

Egy pálya leíró file mellé egy másik ugyanolyan nevű, de '.vw' kiterjesztésű állomány fog kerülni, ezáltal a most implementált megjelenítés leíró file is le van választva a modellről. A `ViewController` fogja betölteni (`load()`), formátuma a következő:

```
/*
 multi line comment
*/
// one line comment

// a talaj <width> széles és <height> magas:
Ground <width> <height>

// az <index> sorszámú futószalag hossza:  <length>,
// kezdőkoordinátája <xs>,<ys> és végkoordinátája <xe>,<ye>:
AssemblyLine'['<index>']' <length> <xs> <ys> <xe> <ye>

// az <index> sorszámú váltó koordinátája <x>,<y>
// leheséges állásirányai <dir>:
Points'['<index>']' <x> <y> '('<dir>{','<dir>}')'

// az <index> sorszámú kilépési pont lámpájának koordinátája <x> <y>:
IndicatorLamp'['<index>']' <x> <y>

// az <index> sorszámú belépési ponthoz csatlakozó teherautó koordinátája <x> <y>:
EntryPoint'['<index>']' <x> <y>

// az <index> sorszámú kilépési ponthoz csatlakozó teherautó koordinátája <x> <y>:
ExitPoint'['<index>']' <x> <y>
```



9.2. ábra. A megjelenítésért felelős osztályok statikus struktúra diagramja

9.3. A grafikus objektumok felsorolása, kapcsolatuk az alkalmazói rendszerrel

9.3.1. PackageCarrierViewObject

Felelősségek:

Alaposztályok:
`PackageCarrierViewInterface`
`ViewObject`

Példányok száma:
0

Konkurencia:
passzív

Perzisztencia:
dinamikus

Komponensek:
nincs

Relációk:
nincs

Változók:
nincs

Szolgáltatások:
nincs

9.3.2. IndicatorLampView

Felelősségek:
Az indikátorlámpa view osztálya.

Alaposztályok:
`ViewObject`

Példányok száma:
n

Konkurencia:
passzív

Perzisztencia:
dinamikus

Komponensek:
nincs

Relációk:
nincs

Változók:

`IndicatorLamp`	`theIndicatorLamp`	
`long`	`x`	A lámpa pozíciójának x koordinátája.
`long`	`y`	A lámpa pozíciójának y koordinátája.

Szolgáltatások:
nincs

9.3.3. AssemblyLineView

Felelősségek:
A futószalag view osztálya.

Alaposztályok:
`PackageCarrierViewObject`
Példányok száma:
n
Konkurencia:
passzív
Perzisztencia:
dinamikus
Komponensek:
nincs
Relációk:
nincs
Változók:

`AssemblyLine`	`theAssemblyLine`	
`long`	`xs`	A futószalag elejének x koordinátája.
`long`	`ys`	A futószalag elejének y koordinátája.
`long`	`xe`	A futószalag végének x koordinátája.
`long`	`ye`	A futószalag végének y koordinátája.

Szolgáltatások:
nincs

9.3.4. PackageView

Felelősségek:
A csomag view osztálya.

Alaposztályok:
`ViewObject`
Példányok száma:
n
Konkurencia:
passzív
Perzisztencia:
dinamikus
Komponensek:
nincs
Relációk:
nincs
Változók:

`Package`	`thePackage`

Szolgáltatások:
nincs

9.3.5. PointsView

Felelősségek:
A váltó megjelenítéséért felelős osztály.

Alaposztályok:
`PackageCarrierViewObject`
Példányok száma:
n
Konkurencia:
passzív
Perzisztencia:
dinamikus
Komponensek:
nincs
Relációk:
nincs
Változók:

`Points`	`thePoints`	
`long`	`x`	A váltó pozíciójának x koordinátája.
`long`	`y`	A váltó pozíciójának y koordinátája.
`long`	`directions`	A váltó view objektumának lehetséges állásai.

Szolgáltatások:
nincs

9.3.6. ViewInterface

Felelősségek:
Közös megjelenítendő pályaelem interface.

Alaposztályok:
nincs
Példányok száma:
0
Konkurencia:
passzív
Perzisztencia:
dinamikus
Komponensek:
nincs
Relációk:
nincs
Változók:
nincs
Szolgáltatások:

`void`	`draw()`	A megjelenítendő pályaelem kirajzolása.
`void`	`click()`	Egérkattintás érzékelése.

9.3.7. PackageCarrierViewInterface

Felelősségek:
Közös interface a csomag szállítóinak.

Alaposztályok:
`ViewInterface`
Példányok száma:
0
Konkurencia:
passzív
Perzisztencia:
dinamikus
Komponensek:
nincs
Relációk:
nincs
Változók:
nincs
Szolgáltatások:

`long`	`getPosX()`	A szállított csomag x koordinátájának lekérdezése.
`long`	`getPosY()`	A szállított csomag y koordinátájának lekérdezése.

9.3.8. GroundView

Felelősségek:
A föld view osztálya.

Alaposztályok:
`ViewObject`
Példányok száma:
1
Konkurencia:
passzív
Perzisztencia:
dinamikus
Komponensek:
nincs
Relációk:
nincs
Változók:

`Ground`	`theGround`	
`long`	`width`	A föld, a pálya szélessége.
`long`	`height`	A föld, a pálya magassága.

Szolgáltatások:

nincs

9.3.9. TransporterView

Felelősségek:
A szállító teherautó view osztálya.

Alaposztályok:
`ViewObject`
Példányok száma:
n
Konkurencia:
passzív
Perzisztencia:
dinamikus
Komponensek:
nincs
Relációk:
nincs
Változók:

`Transporter`	`theTransporter`	
`long`	`x`	A teherautó pozíciójának x koordinátája.
`long`	`y`	A teherautó pozíciójának y koordinátája.

Szolgáltatások:
nincs

9.3.10. ViewControllerInterface

Felelősségek:
Interface a grafikus megjelenítés vezérlésére.

Alaposztályok:
`ViewInterface`
Példányok száma:
0
Konkurencia:
passzív
Perzisztencia:
dinamikus
Komponensek:
nincs
Relációk:
nincs
Változók:
nincs
Szolgáltatások:

`void`	`clear()`	Az összes megjelenítendő pályaelem törlése.
`void`	`addViewObject()`	Egy megjelenítendő pályaelem hozzáadása.
`void`	`removeViewObject()`	Egy megjelenítendő pályaelem törlése.
`void`	`load()`	A grafikus megjelenítést leíró állomány betöltése.

9.3.11. ViewObject

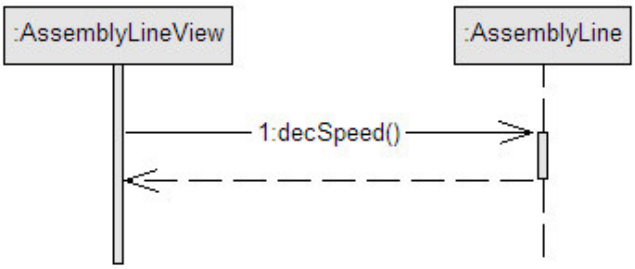
Felelősségek:

Alaposztályok:
`ViewInterface`
Példányok száma:
0
Konkurencia:
passzív
Perzisztencia:
dinamikus
Komponensek:
nincs
Relációk:
nincs
Változók:
nincs
Szolgáltatások:
nincs

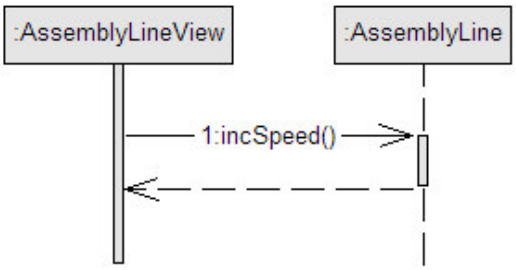
9.3.12. ViewController

Felelősségek:
A grafikus megjelenítés vezérlését végző osztály.

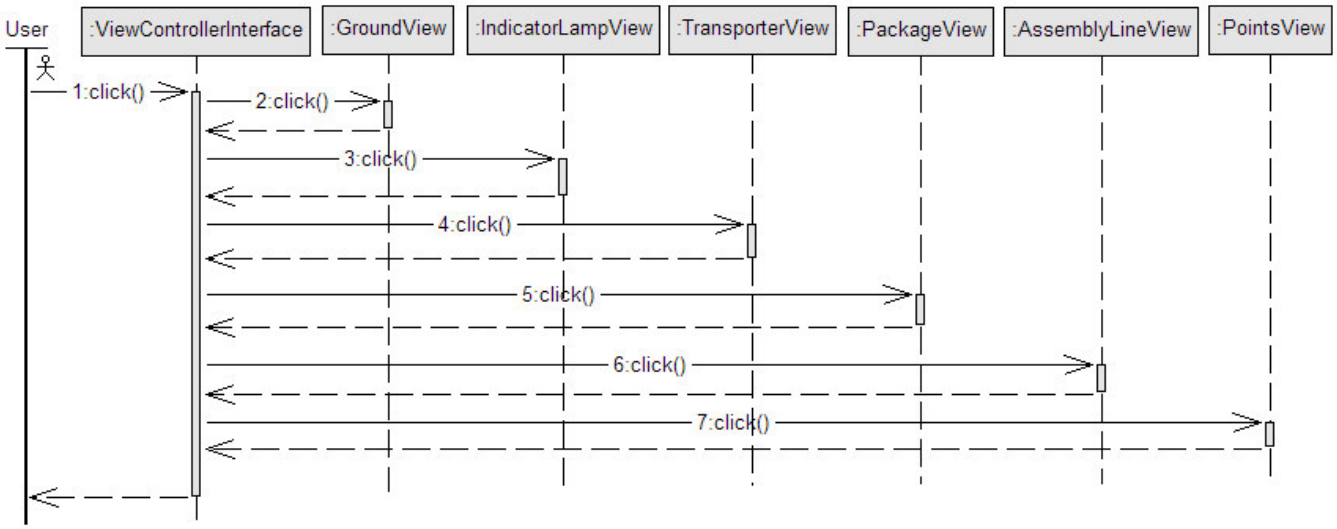
Alaposztályok:
`ViewControllerInterface`
Példányok száma:
1
Konkurencia:
aktív
Perzisztencia:
dinamikus
Komponensek:
nincs
Relációk:
nincs
Változók:
nincs
Szolgáltatások:
nincs



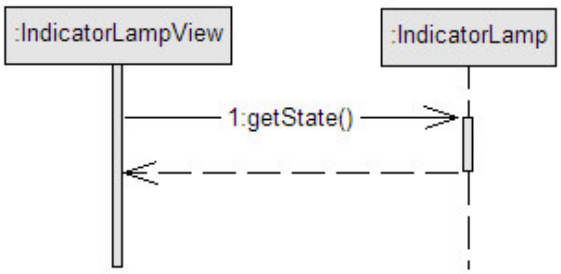
9.3. ábra. Futószalag elejére kattintás szekvenciadiagramja



9.4. ábra. Futószalag végére kattintás szekvenciadiagramja

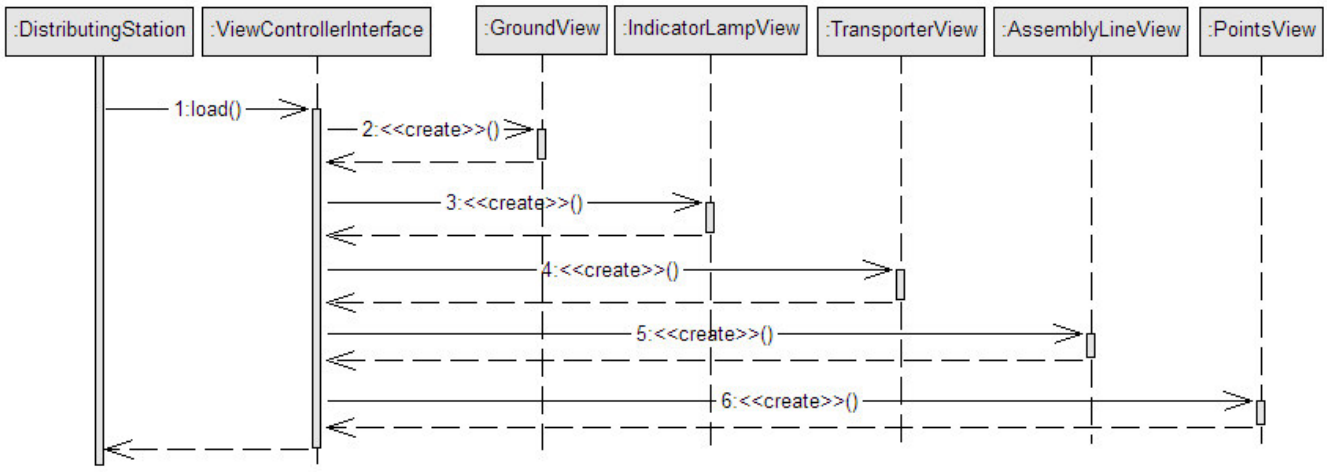


9.5. ábra. User kattintás szekvenciadiagramja

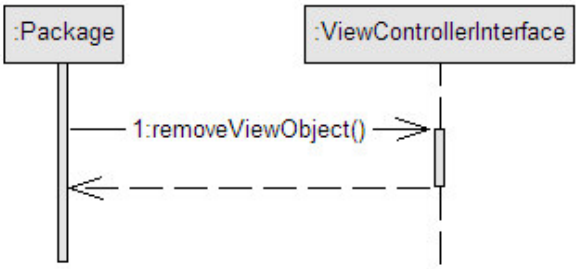


9.6. ábra. Lámpa kirajzolás szekvenciadiagramja

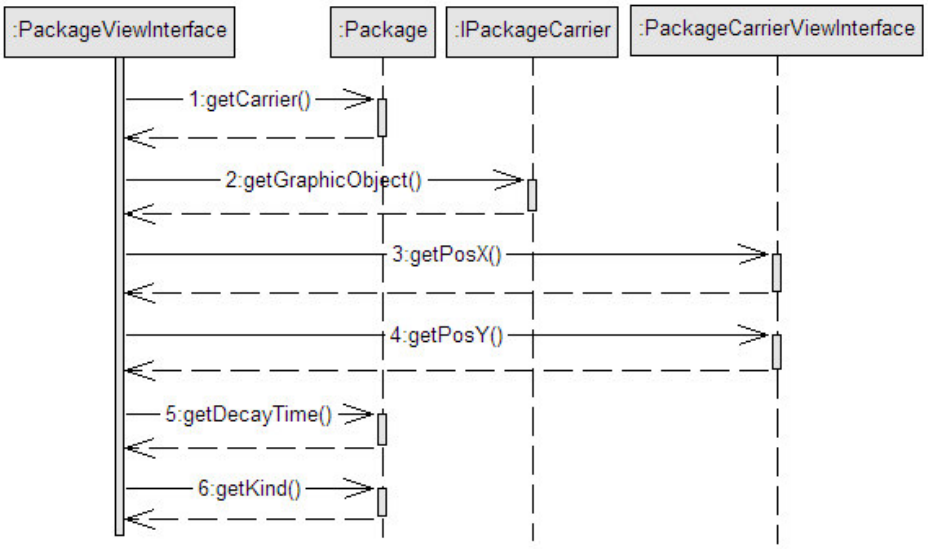
9.7. ábra. Teherautó lecsatlakoztatás szekvenciadiagramja



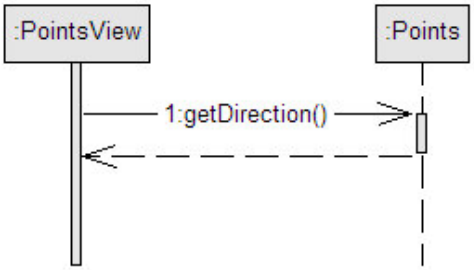
9.8. ábra. Betöltés szekvenciadiagramja



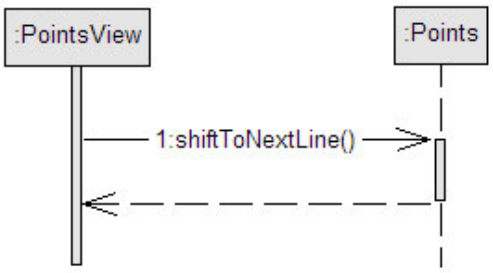
9.9. ábra. Csomag megsemmisülés szekvenciadiagramja



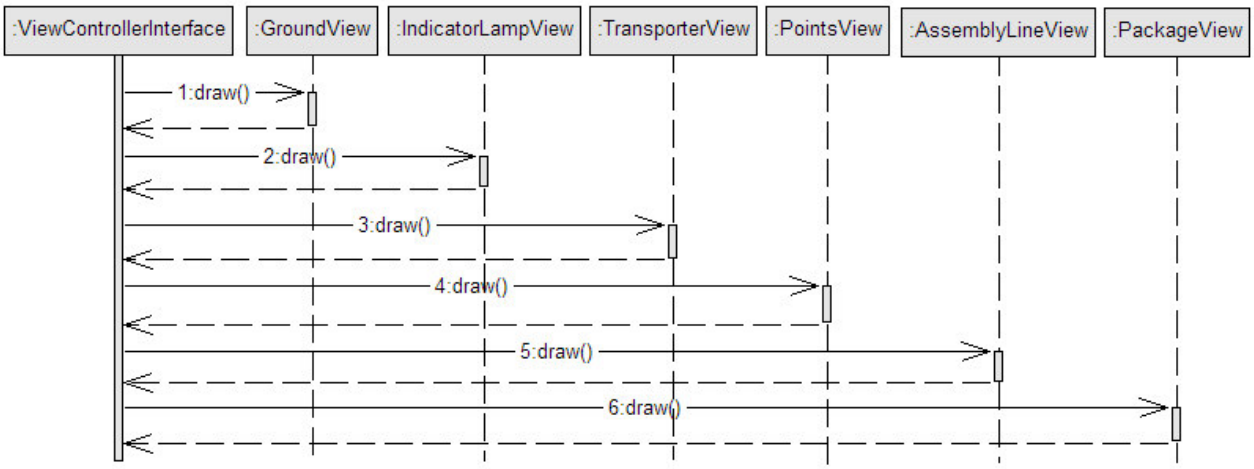
9.10. ábra. Csomag rajzolás szekvenciadiagramja



9.11. ábra. Váltó kirajzolás szekvenciadiagramja



9.12. ábra. Váltóra kattintás szekvenciadiagramja



9.13. ábra. Kirajzolás szekvenciadiagramja

10. fejezet

A grafikus változat beadása

10.1. A grafikus változat telepítése és futtatása

10.1.1. A HSZK-ban DOS parancssorból

Mivel a skeleton fordításával a HSZK-ban problémák merültek fel, ezért a felhasználók munkáját megkönnyítjük azzal, hogy a HSZK-beli futtatásra elkészíttettük a speciális fordító és futtató file-okat.

1. A mellékelt zip file-t ki kell csomagolni egy könyvtárba.
2. A `setjavapath.bat` paranccsal beállítjuk a PATH környezeti változó értékét.
3. A `compile.bat` paranccsal lefordítjuk a forrásálományokat.
4. A program indítása a `run.bat hszk` paranccsal történik.
5. A JavaDoc dokumentációt a `doc.bat` (internetkapcsolat hiányában a `docl.bat`) segítségével állíthatjuk elő. Ekkor létrejön egy Doc nevű könyvtár, amelyből az index.html oldalt webböngészőben megnyitva rendelkezésünkre áll egy jól használható objektumkatalógus. Az itt szereplő leírások a forráskódban is megtalálhatóak kommentek formájában.

10.1.2. DOS parancssorból

1. A mellékelt zip file-t ki kell csomagolni egy könyvtárba.
2. A `compile.bat` paranccsal lefordítjuk a forrásálományokat.
3. A program indítása a `run.bat <JAVAPATH>` paranccsal történik, ahol a `<JAVAPATH>` a Java könyvtára.
4. A JavaDoc dokumentációt a `doc.bat` (internetkapcsolat hiányában a `docl.bat`) segítségével állíthatjuk elő. Ekkor létrejön egy Doc nevű könyvtár, amelyből az index.html oldalt webböngészőben megnyitva rendelkezésünkre áll egy jól használható objektumkatalógus. Az itt szereplő leírások a forráskódban is megtalálhatóak kommentek formájában.

10.1.3. DOS parancssor hiányában

1. A mellékelt zip file-t ki kell csomagolni egy könyvtárba.
2. A következő parancsokkal fordíthatjuk le a rendszert:
```
javac logistic/*.java logisticgame/*.java logisticview/*.java
    logisticproto/*.java
```

```
jar cmf logistic.mf logistic.jar logistic/*.class
    logisticgame/*.class logisticview/*.class
```

3. A program indítása a `<JAVAPATH>\bin\java -jar logistic.jar -classpath <JAVAPATH>\lib\*.jar;<JAVAPATH>\jre\lib\*.jar; <JAVAPATH>\jre\lib\ext\*.jar;<JAVAPATH>\jre\javaws\*.jar` parancs kiadásával történik, ahol a `<JAVAPATH>` a Java könyvtára.
4. A JavaDoc dokumentációt a
```
javadoc -locale hu_HU -encoding iso-8859-2 -charset iso-8859-2
    -docencoding iso-8859-2
    -link http://java.sun.com/products/jdk/1.3/docs/api
    -d Doc -sourcepath logistic -private -author -windowtitle
    "Logistic game - Silent Hill" logistic/*.java
    logisticgame/*.java logisticview/*.java
```
internetkapcsolat hiányában pedig a
```
javadoc -locale hu_HU -encoding iso-8859-2 -charset iso-8859-2
    -docencoding iso-8859-2 -linkoffline -d Doc -sourcepath
    logistic -private -author -windowtitle
    "Logistic game - Silent Hill" logistic/*.java
    logisticgame/*.java logisticview/*.java
```
segítségével állíthatjuk elő.

10.2. A mellékelt álomány tartalma

`./` tartalma:

A `setjavapath.bat`,`compile.bat` és a `logistic.mf` a program fordításához, a `logistic.jar` file elkészítéséhez szükségesek. A `doc.bat` és a `docl.bat` file-ok a dokumentációt generálják. A `run.bat` pedig elindítja a lefordított `logistic.jar` file-t.

```
2005.05.10.  23:26               156 compile.bat
2005.05.10.  23:26               282 doc.bat
2005.05.10.  23:26               243 docl.bat
2005.05.10.  23:25                39 logistic.mf
2005.04.21.  15:33               393 run.bat
2005.04.20.  16:36                52 setjavapath.bat
```

`./logistic` tartalma:

Az egyes file-ok a file nevének megfelelő osztályok implementációját tartalmazzák.

```
2005.05.07.  18:40             7˙861 AssemblyLine.java
2005.05.06.  18:49             3˙252 AssemblyLinePackageInfo.java
2005.05.06.  18:53             3˙471 DeliveryPlan.java
2005.05.06.  18:53             3˙423 DeliveryPlanItem.java
2005.05.07.  14:49            14˙256 DistributingStation.java
2005.05.07.  18:50             4˙785 Game.java
2005.05.06.  19:15             2˙812 Ground.java
2005.05.06.  18:49             2˙776 InDeliveryPlan.java
2005.05.06.  19:16             2˙763 IndicatorLamp.java
2005.05.07.  14:07             1˙469 IPackageCarrier.java
2005.05.06.  19:37             2˙960 OutDeliveryPlan.java
2005.05.07.  15:28             6˙266 Package.java
```

```
2005.05.06.  18:49            2˙742 PackageInfo.java
2005.05.07.  14:22            3˙375 PointOfAssemblyLine.java
2005.05.07.  13:56            5˙915 Points.java
2005.05.06.  19:16            5˙398 Transporter.java
2005.05.06.  19:02            4˙930 TransporterEntryPoint.java
2005.05.07.  13:45            3˙344 TransporterExitPoint.java
2005.05.07.  13:08            4˙575 TransporterJoinPoint.java
```

`./levels` tartalma:

Három pálya a játékhoz.

```
2005.05.07.  18:35              304 level1.lvl
2005.05.07.  17:20              185 level1.vw
2005.05.07.  18:35              655 level2.lvl
2005.05.07.  18:51              360 level2.vw
2005.05.07.  18:48            2˙209 level3.lvl
2005.05.07.  18:28            1˙126 level3.vw
```

`./logisticgame` tartalma:

A `LogisticGame.java` a program főosztálya, a `GameFrame.java` a grafikus ablakot jeleníti meg.

```
2005.05.07.  11:49            1˙978 GameFrame.java
2005.05.07.  12:20              698 LogisticGame.java
```

`./logisticview` tartalma:

A grafikus megjelenítésért felelős osztályok megvalósításai.

```
2005.05.07.  15:25            7˙499 AssemblyLineView.java
2005.05.07.  13:32            2˙715 GroundView.java
2005.05.07.  13:30            3˙192 IndicatorLampView.java
2005.05.07.  15:25            1˙356 PackageCarrierViewInterface.java
2005.05.07.  15:25            3˙526 PackageCarrierViewObject.java
2005.05.07.  15:53            3˙281 PackageView.java
2005.05.07.  18:25            6˙010 PointsView.java
2005.05.07.  13:39            2˙852 TransporterView.java
2005.05.07.  18:49           10˙459 ViewController.java
2005.05.07.  00:17            1˙573 ViewControllerInterface.java
2005.05.07.  00:19            1˙214 ViewInterface.java
2005.05.07.  09:53            2˙439 ViewObject.java
```

10.3. A grafikus változat használata

A játékból az "Exit game" gombbal léphetünk ki. Bármikor kezdhetünk új játékot a "Start new game" gombbal. A váltókat úgy állíthatjuk, ha rájuk kattintunk. A futószalag sebességének csökkentéséhez a futószalag elejére, növeléséhez a futószalag végére kell kattintani.

10.4. A project tapasztalatai

A tárgyat nagyon hasznosnak találtuk, az egyik legérdekesebb tárgynak a félévben. Bár a félév folyamán rendelkezésre álló idő nyilvánvalóan csak egy jelképes példán tette lehetővé a tanult eszközök

és módszerek kipróbálását, mégis érdekes bepillantást nyerhettünk a csoportos, tervezett programfejlesztés részleteibe. A megoldandó feladat a tanult eszközökhöz képest egyértelműen kicsi, így nem is szerezhettünk a programfejlesztésből nagyobb gyakorlatot a félév során, a tárgynak azonban ez a tanulmányi lehetőségek korlátai miatt nem is lehetett célja. Viszont így is sok hasznos tapasztalatot szerezhettünk a tanultak gyakorlati alkalmazásából.

A tárgy egyik legfontosabb tapasztalata az volt, ahogy az információ áramlott a csapattagok között, hogy egymás munkáinak megértéséhez a megfelelő információt biztosítsuk, hiszen az elvégzendő feladatok szorosan egymásra épültek. Az egyik legnehezebb feladat a munkák megfelelő szétosztása volt. A kapott feladat sokszor csak a másik csoporttag munkájának befejezése után volt elvégezhető, és igényelte a más által elvégzett feladat megértését és egyértelmű értelmezését is.

Viszonylag nehezebb volt a feladat modelljének alapötleteit kitalálni, a terveket részletesen kidolgozni, valamint a működő prototípust a tesztelésekkel együtt előállítani. Ezekhez képest viszonylag könnyebb volt a követelmények dokumentálása, a grafikus változat tervezése és beadása.

Az első három héten a rendelkezésre álló idő elegendő volt, a pontszámot az elvégzett feladatnak megfelelőnek éreztük. Az analízis modell és a skeleton tervezése és elkészítése már jóval több munkát jelentett, a feladatért járó pont azonban itt is megfelelt az elvégzendő munkának. A prototípus tervezésénél és a részletes terveknél már nem volt olyan nagy ugrás az elvégzendő munka mennyiségében, sőt, itt úgy éreztük, hogy a feladatért járó pontokat már kevesebb munkával szereztük. A prototípus elkészítése és tesztelése nagyon nagy munkát jelentett, az ezért járó pontokat kevésnek találtuk. Véleményünk szerint ez volt a legnagyobb feladat, mely a pontok számában nem tükröződött eléggé. A grafikus változat tervezése és készítése viszonylag kevés munka volt, melyhez képest aránytalanul sok pont járt a feladatért.

A tárgyból nagy segítség volna, ha az elkészítendő dokumentumok jobban meghatározattak lennének, azaz a csapattagoknak kevésbé kellene a saját fantáziájukra hagyatkozniuk. Felmerült ötletként, hogy a csapatok félév közben kicseréljék egymás között a dokumentált munkákat, de ez elég igazságtalannak tűnik azokkal a csapatokkal szemben, akik rendesen dolgoznak. További probléma lenne az is, hogy aki úgy véli, hogy rossz modellt kap, az ráerőlteti a saját ötleteit a már meglévő elemekre.

Feladatként például egy régi játékot, a TIM-et ajánlanánk vagy esetleg a Worms játék egyszerűsített változatát. Ezek érdekes és gondolkodtató játékok.

10.5. Értékelés

A félév alatt nem változott az az előzetes megállapodás, hogy a kapott feladatokat és így a pontokat is egyenlő arányban osztjuk el egymás között. A feladatok elvégzésében mindenki képességeinek megfelelően dolgozott, a munkát úgy osztottuk szét, hogy lehetőleg mindenki a neki legmegfelelőbb feladatot kapja. Mindent sikerült határidőre beadni.

A. Függelék

Napló

Dátum	Alany	Tárgy
2005.02.21. 14:15 - 2005.02.21. 14:45	Simon, Wirth, Fülöp	Megbeszélés a heti teendőkről. Döntések: – Simon ír egy logoló programot – Wirth elkészíti a szótárt – A követelmények részletes tartalmát megkérdezzük a szerdai konzultáción
2005.02.21. 18:00 - 2005.02.21. 20:00	Simon	A logoló program elkészítése
2005.02.23. 10:15 - 2005.02.23. 11:15	Simon, Wirth, Fülöp	Konzultáció, a követelmények részleteinek megbeszélése. Döntések: – Simon elkészíti a követelmények leírását – Wirth a kész követelményekből elkészíti a szótárt
2005.02.23. 15:00 - 2005.02.23. 17:00	Simon	A részletes követelmények készítése
2005.02.23. 21:00 - 2005.02.23. 22:30	Simon	A részletes követelmények befejezése
2005.02.24. 14:00 - 2005.02.24. 14:30	Simon, Wirth, Fülöp	Értekezlet. Döntések: – Fülöp elkészíti a követelmény definíciót és a projekt tervet.
2005.02.26. 10:00 - 2005.02.26. 16:00	Simon	A static structure diagram tervezése
2005.02.27. 14:00 - 2005.02.27. 15:00	Fülöp	A követelmény definíció megírása
2005.02.27. 20:00 - 2005.02.27. 21:00	Fülöp	A project terv megírása
2005.03.02. 16:00 - 2005.03.02. 20:00	Simon	A static structure diagram megrajzolása
2005.03.03. 17:00 - 2005.03.03. 18:00	Simon	A use case diagramok elkészítése

Dátum	Alany	Tárgy
2005.03.04. 11:45 - 2005.03.04. 12:00	Simon, Wirth, Fülöp	Értekezlet. Döntések: – Simon elkészíti az objektumkatalógust és objektumleírást generáló kódot az Ameos modellezőprogramhoz. – Wirth beírja a kommenteket az osztálydiagramba, ebből közvetlenül generáljuk a kész objektumkatalógust. – Fülöp a generált objektumleírást kiegészíti a részletekkel. – Simon elkészíti a szekvenciadiagramokat. – Simon megrajzolja a state-chartokat.
2005.03.05. 14:00 - 2005.03.05. 16:00	Simon	Az objektumkatalógust és objektumleírást generáló kód megírása az Ameos-hoz.
2005.03.07. 20:00 - 2005.03.07. 23:00	Wirth	Objektum katalógus elkészítése. Döntések: – Az Ameos UML Developer program telepítése, aztán az Object Description-ek elkészítése
2005.03.08. 21:00 - 2005.03.08. 22:30	Wirth	Osztály katalógus befejezése, finomítása
2005.03.10. 16:00 - 2005.03.10. 19:00	Simon	Az osztálydiagram finomítása és a szekvenciadiagramok megrajzolása.
2005.03.10. 21:00 - 2005.03.10. 22:00	Fülöp	A részletes objektumleírás készítése
2005.03.12. 15:00 - 2005.03.12. 15:30	Simon	A state-chartok megrajzolása.
2005.03.14. 09:00 - 2005.03.14. 11:00	Fülöp	Az objektumleírás elkészítése
2005.03.14. 10:00 - 2005.03.14. 12:00	Simon	A skeleton parancsfeldolgozójának elkészítése
2005.03.14. 11:00 - 2005.03.14. 12:00	Fülöp	A részletes objektumleírás készítése
2005.03.14. 14:00 - 2005.03.14. 18:00	Simon	Az Ameos által generált skeleton kiegészítése a use case-ek modellezéséhez
2005.03.14. 15:00 - 2005.03.14. 18:30	Wirth	Kollaborációs diagrammok elkészítése
2005.03.18. 22:00 - 2005.03.18. 23:00	Fülöp	A skeleton kezelői felületének leírása
2005.03.20. 11:30 - 2005.03.20. 12:30	Wirth	Kollaborációs diagramok, use-case-ek befejezése
2005.03.20. 16:00 - 2005.03.20. 17:00	Fülöp	Az Architektúra írása.
2005.03.23. 16:00 - 2005.03.23. 20:00	Simon	A skeleton készítése.
2005.03.24. 09:00 - 2005.03.24. 11:00	Simon	A skeleton készítésének befejezése.
2005.03.25. 16:00 - 2005.03.25. 18:00	Simon	A skeleton beadási dokumentációjának elkészítése.
2005.03.26. 11:00 - 2005.03.26. 13:00	Simon	A prototípus fileformátumainak megtervezése.
2005.04.02. 21:00 - 2005.04.02. 22:30	Fülöp	Tesztelési terv írása

Dátum	Alany	Tárgy
2005.04.03. 10:00 - 2005.04.03. 10:15	Simon	A prototípus fileformátumainak módosítása a követelmények változása miatt.
2005.04.03. 11:30 - 2005.04.03. 12:00	Fülöp	Tesztelési terv írása
2005.04.03. 13:00 - 2005.04.03. 14:00	Wirth	Use Case-ek megrajzolása
2005.04.03. 18:00 - 2005.04.03. 20:00	Wirth	Use Case-ek leírása
2005.04.09. 13:30 - 2005.04.09. 16:30	Simon	Az objektumok és metódusok terveinek készítése
2005.04.09. 16:00 - 2005.04.09. 22:00	Wirth	Teszteset definíciók elkészítése
2005.04.09. 17:30 - 2005.04.09. 19:00	Simon	Az objektumok és metódusok terveinek készítése
2005.04.10. 09:30 - 2005.04.10. 12:30	Fülöp	A tesztfájlok létrehozása
2005.04.10. 10:30 - 2005.04.10. 12:30	Wirth	Tesztesetek elkészítése
2005.04.10. 13:30 - 2005.04.10. 14:30	Fülöp	A tesztek részletes terveinek írása
2005.04.10. 16:15 - 2005.04.10. 16:45	Fülöp	A tesztek részletes terveinek írása
2005.04.15. 15:00 - 2005.04.15. 20:00	Simon	A prototípus készítése
2005.04.16. 09:00 - 2005.04.16. 14:00	Simon	A prototípus készítésének befejezése
2005.04.16. 15:30 - 2005.04.16. 19:00	Wirth	tesztelés
2005.04.17. 11:00 - 2005.04.17. 12:30	Wirth	tesztelés
2005.04.17. 14:30 - 2005.04.17. 16:40	Wirth	tesztelés
2005.04.20. 20:00 - 2005.04.20. 22:00	Fülöp	A prototípus tesztelése
2005.04.30. 17:30 - 2005.04.30. 20:00	Simon	A grafikus kezelői felület terve, a grafikus rendszer architektúrája
2005.05.01. 07:00 - 2005.05.01. 08:00	Fülöp	A view class diagram készítése
2005.05.01. 12:00 - 2005.05.01. 15:00	Wirth	Szekvencia diagramok elkészítése
2005.05.07. 10:00 - 2005.05.07. 16:00	Simon	A grafikus változat elkészítése
2005.05.13. 17:00 - 2005.05.13. 18:00	Simon	A grafikus változat telepítési dokumentációjának elkészítése
2005.05.14. 20:00 - 2005.05.14. 22:00	Fülöp	A tapasztalatok és az értékelés írása