

Grand Theft Bank

19 - DreamCoders

Konzulens:
Goldschmidt Balázs

Csapattagok

Angeli Róbert
Koczka Tamás
Kovács Balázs
Mikó Norbert

H55DMW	r.angeli.11@gmail.com
DAQ28J	kockatamas@gmail.com
HZFWTN	kockafely@gmail.com
PKWHLM	miko.norbert@gmail.com

2010-05-12

2. Követelmény, projekt, funkcionalitás

2.1 Követelmény definíció

2.1.1 A program célja, alapvető feladatai

Az elkészítendő program egy játék, amelyben egy bankrabló autóját irányítva kell menekülni a rendőrautó(k) elől egy előre megtervezett pályán (városban), melyben természetesen más autók is közlekednek.

A fejlesztőcsapat célja egy olyan kész program előállítása, mely teljesíti a specifikációban megkövetelt elvárásokat, és ami minden olyan gépen lefordítható és futtatható, mely megfelel az alább megfogalmazott követelményeknek.

2.1.2 A felhasználói felület

A program végső változata billentyűzet és egér felhasználásával lesz irányítható, grafikus felhasználói felülettel fog rendelkezni. A felhasználói felület legnagyobb részét a térkép fogja elfoglalni játék közben.

2.1.3 A program futtatásához szükséges követelmények

A futáshoz szükséges, hogy a felhasználó számítógépére telepítve legyen a Java Runtime Environment (JRE) 1.6-os verziója. Habár a program futtatásához szükséges követelmények megegyeznek az előírt JRE verzió mindenkori követelményeivel, így a most használt PC-k nagy részén zökkenőmentesen fut, optimális futtatásához legalább P4 1GHz-es processzorral, 512 MB RAMmal és 200 MB lemezterülettel, billentyűzettel, egérrel és a JRE futtatásához szükséges megjelenítő VGA kártyával rendelkező géppel kell rendelkezni.

2.1.4 A szoftver fejlesztésével kapcsolatos alapkövetelmények, elvek, célok

2.1.4.1 Modellhűség

Az egyik legfontosabb szempont, hogy az elkészült szoftver mennyire szigorúan elégíti ki a specifikációban meghatározott követelményeket. A fejlesztőcsapat fontosnak tartja ennek szem előtt tartását, már a tervezés kezdetétől fogva.

2.1.4.2 Továbbfejleszthetőség

A kész termék működése, felépítése teljes egészében és részleteibe menően megérhető a hozzáadott dokumentáció felhasználásával. A fejlesztőcsapat a modellt úgy tervezi meg, hogy az esetleges későbbi továbbfejlesztést megkönnyítse.

2.1.4.3 Modularitás

A készítőik törekednek arra, hogy a program moduláris legyen, azaz a részegységei minél jobban elkülönüljenek egymástól, így garantálva a későbbi újrafelhasználhatóságot és könnyebb tesztelést. Az elkészült alkalmazás többek között az MVC szerkezeti mintát is fel fogja használni.

2.1.4.4 Teljesítmény, optimalizálhatóság

A fejlesztési folyamat során fontos a használt algoritmusok optimalizálása annak érdekében, hogy a specifikációban meghatározott minimum konfiguráción a program élvezhetően fusson.

2.1.4.5 Felhasználhatóság

A készítők számára fontos, hogy egy könnyen kezelhető programot hozzanak létre, melynek használata akár a felhasználói dokumentáció nélkül is megérthető, elsajátítható.

2.1.4.6 Perzisztencia

A tesztelés megkönnyítése érdekében a program futása során tapasztalt legfontosabb információkat fájlokba kell írni.

2.1.4.7 Modern technológiák használata

Az UML és a RUP használata minimalizálja a tervezés és az implementáció közötti szakadékot, forgatókönyvet ad a munkafolyamatok sorrendjéről. Éppen ezért fejlesztőcsapat beépíti ezek használatát a fejlesztés folyamatába.

2.2 Projekt terv

2.2.1 A felhasznált fejlesztőeszközök

A csapat választása az NetBeans integrált fejlesztői rendszer használatára esett, mivel a csapattagjai korábbi tapasztalatokkal rendelkeznek a fejlesztő eszköz használatával kapcsolatban, így jelentős időt takaríthatnak meg azzal, hogy nem kell egy új eszköz használatát megtanulniuk. Ennél fontosabb szempontnak tekinthető pedig, hogy a Netbeans rendelkezik minden olyan szolgáltatással, ami a fejlesztést megalapozza, mint például az UML modell készítés, illetve a verziókövetés támogatása.

Az UML diagramok és modellek elkészítéséhez felhasználásra került a Visual Paradigm cég Smart Development Environment for NetBeans Community Edition modellező programja, ami beépül a Netbeans fejlesztő környezetbe és így képesek egységként működni. A modellező program választásának szempontja volt, hogy képes legyen UMLben megtervezett modellből Java nyelvű kódot készíteni, ezzel garantálva azt, hogy a Java forráskód a modellel összhangban lesz, és követi azt a szemlélet, miszerint a tervezés az elsődleges a projekt készítése során és egy jól elkészített modell alapján kell felépülnie a programkódnak.

A dokumentáció készítése aszinkron és szinkron módon is történik. Míg az aszinkron módosításokat és a végleges dokumentációt Microsoft® Wordben és .doc kiterjesztésnek megfelelő Word 97-2003-as fájlformátumban készül, addig az szinkron módosításokat egy külön a projekt számára fenntartott és megosztott Google Docs tárterületen, közösen és egyszerre is elvégezhetjük, ugyancsak elősegítve a közös munkán alapuló minél jobb eredmény elkészültét.

A projekt verziókövetését és menedzsmentjét a Google Code szolgáltatás keretében valósítjuk meg, amire a kitűnő fejlesztést segítő szolgáltatásai miatt esett a választás. Legfontosabb a revisionökön és Subversionön alapuló verziókövetése, továbbá hibakövető és -jelentő szolgáltatása, a Wikin alapuló információs oldalai, illetve a változásjelentő rendszere, ami emailben egyből értesíti a csapat tagjait, ha változás történt, amiről érdemes tudniuk.

A csapattagjai közötti aszinkron kommunikációt levelező listán valósítjuk meg, amihez a szolgáltatást a Google Groups nyújtja, míg a szinkron kommunikációhoz a Microsoft® Messenger, illetve a hangkommunikációhoz pedig Skype nevű programot használjuk.

2.2.2 A fejlesztőcsapat tagjai, azok feladatkörei

Név	Feladatok
Angeli Róbert	dokumentáció, tesztelés
Koczka Tamás	vezető, dokumentáció és modell revízió, implementáció
Kovács Balázs	dokumentáció, diagramkészítés, algoritmizálás
Mikó Norbert	dokumentáció, implementáció, tesztelés

A csapattagok közötti munkamegosztás dinamikusan változhat a csapattagok projektre fordítható idejétől függően, de alapvetően az egyenletes munkamegosztásra és a feladatkörök minél jobb elhatárolódását tartjuk szem előtt.

2.2.3 Kommunikációs modell

A csapat kommunikációja gondos tervezés eredménye, több kommunikációs formát, mind aszinkron, mind pedig szinkron módon megteremtettünk. Mivel a fejlesztés kulcsfontosságú kérdésének tekintjük a tagok közötti kommunikációt, ezért minden kommunikációs szoftvert gondosan választottunk ki és a tagok egyöntetű beleegyezése alapján – figyelembe véve azt, hogy ne okozon problémát ezek eszközök használatának megszokása – döntöttünk használatuk mellett.

Mivel nem garantálhatjuk, hogy a projekt lefolyása alatt minden tag folyamatosan elérhető (online), ezért aszinkron kommunikációt használunk minden tagot érintő információ cseréjére. Alapvető bázisköve az aszinkron kommunikációnak a levelező lista, ahol lehetőség van találkozók megtervezésére, a dokumentációban, modellben, kódban való változások jelzésére és ezek felülvizsgálatának kérelmére. Ezenkívül a Google Code felületén lévő módosítások vagy a verziókövető rendbe történő feltöltések automatikusan kikerülnek a levelezési listára, így garantálva, hogy mindenki a lehető leghamarabb értesüljön a változásokról.

Habár az aszinkron kommunikációs formák nagyon hatékonyak bizonyulnak, előfordulnak esetek, amikor nem elég gyors az információáramlás. Ezért megteremtettük a szinkron szöveg- és hangkommunikáció lehetőségét, ami nagyban támogatja az azonnal döntéshozás lehetőségét, így egy-egy problémára nagyságrendekkel gyorsabban születhet válasz. Szinkron kommunikációs formában elsődleges szerepet a fizikai kommunikáció, azaz beszéd tölt be, mivel a csapat tagjai naponta többször is találkoznak, ez lehetőséget teremt arra, hogy folyamatosan tudjanak kommunikálni egymással. Ha erre mégse lenne lehetőség, akkor interneten keresztül tartják előre egyeztetett időben a kapcsolatot szinkron kommunikációs (szöveg és/vagy hang) eszközökkel.

A használt kommunikációs formák részletes leírásait lentebb olvashatja.

Verziókezelés (Subversion):

Garantálja, hogy a forráskód minden csapattag számára szinkronban legyen, azaz ugyanazon a kódon tudjanak dolgozni akár egyszerre többen is (amíg a kód különböző részeit szerkesztik). Ezenkívül nyilván tartja a kód változását, hogy mikor ki és milyen változást eszközölt a projektben.

★ r56	2.2 is kész kbra	Yesterday (38 hours ago)	kockafely
★ r54	2.1 Követelmény definíció kész, nézzétek meg hogy tetszik	Yesterday (39 hours ago)	kockafely
★ r53	Atírtam Angeli e-mail címet	Yesterday (39 hours ago)	kockafely
★ r52	[No log message]	Feb 14 (2 days ago)	kockafely
★ r51	[No log message]	Feb 14 (2 days ago)	kockafely
★ r50	Edited wiki page through web user interface.	Feb 14 (2 days ago)	koczkatamas
★ r49	Fedlap kész, 02-es doksi kiegészítve vele	Feb 14 (2 days ago)	koczkatamas
★ r48	Edited wiki page through web user interface.	Feb 14 (2 days ago)	koczkatamas

Levelezési lista:

Alapvető aszinkron kommunikációs forma. A GoogleCode rendszer bejegyzést ír a projekt bármely részhalmazának változása esetén és beállítható hogy levelet is küldjön a levelezési listára, mert a projektben résztvevők számára kulcsfontosságú, hogy átlássák és megfelelő időben értesüljenek a projektben bekövetkezett változásokról. Ezenkívül a találkozók és döntések megbeszélésére is használható.

★ dreamcoders	Inbox	DC	[dreamcoders] r59 committed - Doksi első része kész, tekintsetek meg :D - Revision: 59 Author: koi	12:05 pm
★ dreamcoders	Inbox	DC	[dreamcoders] r58 committed - Essential Use Case javítás és doksi írogatás - Revision: 58 Author: l	11:14 am
★ dreamcoders	Inbox	DC	[dreamcoders] r57 committed - Essential Use Case - Revision: 57 Author: kockafely Date: Tue Feb 16 15	12:00 am
★ dreamcoders	Inbox	DC	[dreamcoders] r56 committed - Essential Use Case próba, nem tudom, jó-e - Revision: 56 Author: kockaf	Feb 15
★ dreamcoders	Inbox	DC	[dreamcoders] r55 committed - 2.2 is kész kbra - Revision: 55 Author: kockafely Date: Mon Feb 15 12:48	Feb 15
★ me		DC	Projekt előkészületek - Az ütemterv kibővült, jelenleg a következő adatok érhetők el: - beadások időpontjai ...	Feb 14
★ Balázs, me, Ukrán (3)	Inbox	DC	UML editor keresése - Kene egy megfelelő programot találni erre a feladatra. -- Balázs	Feb 14
★ dreamcoders		DC	[dreamcoders] r27 committed - UML model teszt - Revision: 27 Author: koczkatamas Date: Sat Feb 13 12:03:58	Feb 13
★ dreamcoders		DC	[dreamcoders] r26 committed - UML model teszt - Revision: 26 Author: koczkatamas Date: Sat Feb 13 11:55:18	Feb 13
★ Kövi, me (2)	Inbox	DC	Próba - Mukodik	Feb 13
★ me		DC	Ütemterv - Felraktam a wikibe az ütemtervünket: http://code.google.com/p/dreamcoders/wiki/Utenterv	Feb 13

Wiki (Google Code részként):

Tudásbázisként funkcionáló aszinkron kommunikációs forma. Ide kerülnek fel azok az anyagok, amiket a projekt menete során lerögzítettünk és a tagok innen mindig tudnak informálódni, ha szükségük van a már eldöntött információkra.

Teljesített

#	Konzi	Beadás	p	Név
1	2010.02.08	2010.02.10		14 h - csapatok regisztrációja

Aktív

[Szerkeszthető doksik](#), [fedlap](#)

#	Konzi	Beadás	p	Név	Eredeti		Minták		Szerkeszt	
2	2010.02.15	2010.02.18	10p	Követelmény, projekt, funkcionalitás	DOC	PDF	2005	2009	DOC	google docs
3	2010.02.22	2010.02.25	20p	Analízis modell kidolgozása 1.	DOC	PDF	2005	2009	DOC	google docs
4	2010.03.01	2010.03.04	30p	Analízis modell kidolgozása 2.	DOC	PDF	2005	2009	DOC	google docs
5	2010.03.08	2010.03.11	20p	Szkeleton tervezése	DOC	PDF	2005	2009	DOC	google docs

További kommunikációs formák:

Microsoft® Messenger: szinkron kommunikációs forma, itt a tagok általában online elérhetőek, gyors döntések és megbeszélések hozására alkalmas

Skype: szinkron kommunikációs forma, rendeltetése az előzőhöz hasonló, de sokkal gyorsabb információcserét garantál, internettel rendelkező környezetben használjuk.

Mobil telefon: ugyanaz, mint az előző, csak internettel nem rendelkező környezetben használjuk

Fizikai kommunikáció (találkozók): élőbeszéd, fontosabb döntések meghozatalakor használjuk, amikor fontos a tagok teljes figyelme

2.2.4 Fejlesztési mérföldkövek, ütemterv

Skeleton:

Az első fontosabb mérföldkő a program vázának megalkotása. Elsődleges fontosságú a program modelljének megtervezése, annak helyességének és részletességének ellenőrzése, maga a tervezési folyamat során a gondos és odafigyelő döntéshozás, mert ezek a már elkészült program minőségét nagyban befolyásolják. A projekt ezen része sok ráfordítandó időt kíván a tagoktól, de csak ezzel lehet garantálni, hogy a projekt előrehaladásában a későbbiekben esetlegesen felmerülő problémákra könnyedén születik megoldás, ezért a projekt ezen részének minél előretekinthetőbb elkészítése kulcsfontosságú a projekt szempontjából.

Prototípus:

A második mérföldkő a program működőképes változata, ami még grafikát nem tartalmaz. Ebben az állapotban már lehetőség nyílik a program egészének tesztelésére. Ebben az állapotban már a modell rész működés szempontjából lezárult, az algoritmusok véglegesedtek és így a grafikus felülettől különálló egészet jelentenek.

Grafikus felület:

A harmadik és egyben utolsó mérföldkő a program fejlesztésében a grafikus felület elkészítése. Ebben a szakaszban ügyelni kell arra, hogy a jól működő modell köré egy olyan felhasználói környezetet teremtsünk, ami a program használatának hatékonyságát minél jobban maximalizálja. A grafikus felület (továbbiak esetenként GUI) megtervezésekor ezért a könnyű használhatóságra, ergonómiára és könnyen tanulhatóságra nagy hangsúlyt kell fektetni.

2.2.5 Határidők

febr.10.	Csapatok regisztrációja
febr. 18.	Követelmény, projekt, funkcionalitás – beadás
febr. 25.	Analízis modell kidolgozása 1. – beadás
márc. 4.	Analízis modell kidolgozása 2. – beadás
márc. 11.	Szkeleton tervezése – beadás
márc. 18.	Skeleton – beadás
márc. 25.	Prototípus koncepciója – beadás
ápr. 1.	Részletes tervek – beadás
ápr. 15.	Prototípus – beadás
ápr. 22.	Grafikus felület specifikációja – beadás
máj. 6.	Grafikus változat – beadás
máj. 13.	Összefoglalás - beadás

2.2.6 Átadás

Az projekt előrehaladtáról készült dokumentációkat hetente (lásd 2.2.5 Határidők pont) konzultáció formájában adjuk át. Ezenkívül a három mérföldkő alkalmával a működő programkód is bemutatásra kerül az erre kijelölt laboratórium számítógépein. Végül a projekt elkészültekor az összes aktualizált dokumentáció és forráskód feltöltésre kerül a tárgy honlapjának szerverére.

2.2.7 Egyéb fontos megjegyzések

A programnak a specifikációban meghatározott környezetben futnia kell.

2.2.8 Szükséges dokumentációk

2. Követelmény, projekt, funkcionalitás
3. Analízis modell kidolgozása 1
4. Analízis modell kidolgozása 2
5. Skeleton tervezése
6. Skeleton
7. Prototípus koncepciója
8. Részletes tervek
10. Prototípus
11. Grafikus felület specifikációja
13. Grafikus változat
14. Összefoglalás

2.3 Feladateleírás

Egy kis német városban, Verkehrsmeldungenben egy rabló bankot robbant!

A bankrablás után a banktól el szeretne jutni a búvóhelyére. A bankrabló mindent megtesz azért, hogy ezt véghezvigye. Eközben a város rendőrei elkezdik üldözni a bankrablót, és mindent megtesznek azért, hogy a bankrablót elkapják. Mind a rendőrök, mind a rabló felturbózott személyautóval próbálja célját elérni.

A városban (próbálva a valóságot szimulálni) nem csak ez a két jármű található. A város utcáin az autók valamilyen véletlenszerű sebességgel közlekednek.

Az autók a városhatárnál ki tudnak lépni a városból, illetve be tudnak lépni a városba. Ezeknek a ki/belépéseknek a helyei véletlenszerűen választódnak. Feltételezzük, hogy lényegében ugyanakkora forgalom van végig a városban, így amikor egy autó kilép a városból, akkor egy másik autó (egy másik helyen) belép a városba. A rabló azonban a település elhagyása esetén védettséget élvez, hiszen ekkor a város tohonya rendőrei már nem hajlandóak őt követni a megye határain túlra. Bankrablónk, ha készen áll folytatni kalandjait, akkor ugyanott visszatérhet a városba ahol az előbb azt elhagyta azonban ezzel újra magára vonva a város összes rendőrének lebilincselő figyelmét.

Verkehrsmeldungenben híres városában az arisztokrácia színe JAVA lakik, így a lakosok, hogy minél jobban megfeleljenek a társadalom elvárásainak, mindenképpen betartják a

közlekedési szabályokat, nehogy a következő XXI. századi bálon valamely lakó rossz színben tüntesse fel őket.

Továbbá a városban lényegében csak átmenő autósforgalom található, mivel autója csak az arisztokratáknak van, és az arisztokraták már csak elvből sem költöznek a város koszos gyárkéményektől füstölő, szürke ködfelhőbe, illetve nem ereszkednek le az őket szolgáló pórnéphez. Az arisztokraták csak a város határain túl laknak a meleg, üde, déli domboldalakon elhelyezett bortermesztésre kikényszerített szőlőrengeteg fölé épített kúriákban / kastélyokban. A várost viszont gyakran használják átutazásként, amikor meglátogatják valamelyik arisztokrata barátjukat egy másik borvidéken. Az őket szolgáló pórnép a városban csak földalatti metróon közlekedik, amelynek a tárgyalása nem esik jelen project alá. Így az autók a városban sehol nem állnak meg várakozni.

A városban így a következő alapvető szabályokat vetették pergamenre:

- minden jármű köteles a következő kereszteződés piros lámpájának észlelése esetén lelassítani és megállni, majd a zöld fény felgyulladására esetén véletlenszerűen megválasztva továbbhaladási irányát, ha azt a szabályok nem tiltják, továbbhaladni.
- Stop tábla észlelése esetén a táblával egy vonalban meg kell állni, majd környező utcákból érkező forgalom megvizsgálása után, ha továbbhaladás szabad, akkor a kiválasztott irányban tovább kell haladni
- Ha egy jármű egy adott utcán belül utolér egy másik, nála lassabban haladó járművet, akkor a helyi szokásoknak megfelelően nem előzi meg: egyrészt az út fizikai méretei miatt, másrészt az arisztokrácia elv miatt. Ami kimondja, hogy a gyorsabban haladó "megteheti", hogy nem kerüli ki, így felveszi az előtte haladó jármű sebességét, majd a következő kereszteződésben, ha a két jármű különböző úton halad tovább, akkor visszatérhet a gyorsabb jármű a saját sebességére.
- Továbbá a városban minden vezető a további szabályokat ismeri, és ezek szerint közlekedik, ill. egy adott kereszteződésre jellemző szabályok az adott kereszteződésben megtalálhatók táblákon.

Verkehrsmeldungen városának alapjait még a középkor hajnalán rakták le így még az akkoriban jellemző szűk utcák váltak jellemzővé a város úthálózatára. A polgármester ezt figyelembe véve a balesetek minimalizálása érdekében bevezette, hogy az utcákon csak egy irányban lehessen közlekedni. Ez a szabály olyannyira belopta magát a lakosság szívébe, hogy még a legelvetemültebb bűnözők sem képesek őket megszegni.

Mint különböző ásatásokból kiderült, hogy a város egy ősi rabló temetőre épült, így a városban élő csirkefogóknak ennek köszönhetően három életük van. Azaz másik járművel való ütközés esetén sebességük a 0-ra csökken és addigi életük száma csorbát szenved. Így az üldözött összesen 3-szor ütközhet, ám mindennek ára van, hiszen a kocsisérüléseket szenved, mely a játékban úgy jelenik meg, hogy a rabló már nem lesz képes elérni az addigi maximális sebességét. Ez a sebességből való levonás minden ütközésnél megismétlődik. Ha a rabló 3-nál többször karambolozik, akkor megszakad a téridő kontinuum és vége lesz a játéknak.

A pénztárolóintézmény kiürítője (csak a szóismétlés elkerülése végett) rendkívül hegyes füllel jött a világra (többek között az ebből fakadó társadalmi kirekesztettsége vezette őt téves utakra). Ezek a szinte az Avatar c. filmre jellemző módon kifejlett érzékszervei hősünket viszont képessé tették a rendőrség rádiófrekvenciájának a vételére melynek hála a játékos mindig pontos képpel rendelkezik a rendőrautók tartózkodási helyéről. A rabló a börtön hideg rácsaitól félve, próbálja menteni a bőrét, így agya tulajdonképpen kiszűri a táblákat, irányjelző

világítóberendezések kisugárzott fotonjait, és ezért lelkében egy csorba sem esik és nyugodt szívvel és kellően biztos gázpedáltaposással áthágja a szabályokat, ott ahol ez előnyére válik, de előzni ő sem tud, mert ez a szabály a szíve mélyén van.

A bankrablást követően a kirabolt bank, mivel nem a legújabb technikai biztonsági rendszerekbe investálta a felhalmozott kamatrés általi nyereséget, így nem rendelkezik a legújabb vezeték nélküli technológiákkal, és a különböző interferenciás hatások miatt, mint például a környező házak, illetve a lázadók által épített járdablokádok eredményeképpen a bank csak a 600-800 méter sugarú körgyűrű területén lévő rendőröket tudja értesíteni. De az ezen a területen lévő rendőrök az értesítést követően egyből elindulnak a gumicsikorgással távozó rabló kézre kerítésére. Az ilyen rablási esetek kezelése érdekében a levegőben folyamatosan cirkál egy helikopter (fizikailag a játékban nem vesz részt, csak a szemléltetés kedvéért), ami on-line szinte 0 sec overHead-del tudja tájékoztatni az izzadtsággal átitatott kormány mögött emelkedett adrenalin szintű rendőröket. A rendőrök a rabló által nem ismert, titkos agy furmányos módon próbálják megközelíteni és elkapni a rablót, így adrenalin szintjük miatt, és a bércsökkentés fenyegetettsége miatt nekik nem kell betartani a fent leírt közlekedési szabályokat. Ha egy olyan utcába tévednek, ahol már előttük lesz egy autó, akkor azt ők sem tudják megelőzni, így kénytelenek lelassítani és egy képviselő fánkot rágva, bosszankodva meg kell, hogy várják a következő kereszteződést az előttük haladó jármű tempójában, majd onnan újra felizzítva a vasat nekieshetnek a rabló további üldözéséhez.

2.4 Szótár

Térkép: a város úthálózatát, keresztezéseit, bankot, búvóhelyet jelző grafikai felület egészen a városhatárral bezárólag.

Bank: az a hely a játékban, aminek elérése a Játékos (vagy Bankrabló) célja és ezen hely elérése után fogják a Rendőrök üldözni őt.

Rejtekhely: az a hely a játékban a térképen, amely elérése a játék sikeres végét jelenti. Ez a cél. Itt már a rendőrök nem találják meg a játékost.

Bankrabló: a felhasználó által irányított virtuális személy (jármű), akinek célja a Bank elérése, majd a Rejtekhely elérése anélkül, hogy más autókkal vagy Rendőrrel érintkezzen.

Rendőr: a játék által irányított jármű, ill. járművek, amelyek elsődleges célja a Bankrabló pillanatnyi helyének elérése.

Egy hely elérése: a térképen a hely pozíciójának megfelelő koordinátára menni az autóval. (A hely és egy jármű koordinátáinak távolsága kisebb, mint 2 méter)

Városhatár: a térképről kivezető illetve a térképre bevezető utak területe.

Városba belépés: a városhatár valamely pontján egy új autó megjelenik és a város irányába halad.

Városból kilépés: a városhatár elérése esetén (kivezető úton), az elérő jármű eltűnik a térképről.

Városból való kilépés rablóval: a városhatár elhagyása esetén a rabló mentességet élvez, addig, míg újra vissza nem tér a játéktérre.

Mentesség: az a jelenség mikor a rendőrök nem követik a rablót mivel nem tudják beazonosítani tartózkodási helyének koordinátáit.

Játék vége: az a pillanat, amikor a rendőr pozíciója megegyezik a rabló pozíciójával, vagy a rabló már háromszor ütközött, vagy ha a rabló eléri a rejtékhelyet.

Menni az autóval: az autóval az utakon a leírásban meghatározott szabályok szerint közlekedni.

Az autó sebessége: a jármű sebessége km/h mértékegységben megadva, amellyel pillanatnyilag halad.

- Rabló maximális sebessége: 140 km/h (ütközés esetén csökkenhet)
- Rendőr maximális sebessége: 120 km/h
- Egyéb autó maximális sebessége: 50 km/h

Járművezetők látótere: Egyenes úton haladva 50 méter, kereszteződésben a környező utak forgalmát is belátja 50 méterre

Járművek viselkedése piros lámpához érkezéskor, illetve stop tábla esetén: 30 méterre a táblától/lámpától, egyenletes lassítással 0 km/h-ra csökkenti a sebességét.

Járművek viselkedése egy úton egy másik jármű elérése esetén: a jármű 20 méterre az előtte haladó autótól egyenletes lassítással felveszi az előtte haladó jármű sebességét majd konstans 10 méteres távolsággal követi.

Ütközés: két jármű ütközik, amikor a térképen a járművek által lefedett területeknek van közös koordinátájuk. A városban csak a rabló tud ütközni mivel a többi jármű a karambol elkerülése végett felveszi az előtte haladó kocs sebességét.

Rablóval való ütközés: ha az általunk irányított rablóval ütközünk, akkor két dolog történik: életet veszünk, végsebességünk csökken.

Végsebesség csökkenése: az ütközés következtében a rablóra kirótt büntetés melynek során a végsebesség a következő stádiumok szerint csökken karambolonként:

- 140 km/h: a rabló sérülésmentes sebessége.
- 120 km/h: 1. ütközés utáni maximális végsebesség.
- 100 km/h: 2. ütközés utáni maximális végsebesség.

Élet: egy adott szám (jelen esetben 3) mely azt reprezentálja, hogy a rabló hányszor ütközhet addig, míg véget nem ér számára a játék. Karambol esetén az adott értékből 1 levonásra kerül. A 0 elérése esetén ér véget a játék.

Út: két kereszteződést összekötő 1 sáv, 1 irányú szakasz, mely az autók a megadott irányba való haladását biztosítja.

Egyirányú: az adott úton kizárólag csak a meghatározott irányban lehet közlekedni minden járműnek. E szabály alól senki sem képez kivételt.

Egysáv: az adott útszakasz szélességét tekintve csak egyetlen jármű közlekedését biztosítja, azaz több jármű nem fér el egymás mellett, így nem lehetséges előzés.

Előzés: az a manőver mellyel az egyik jármű egy másik előtte haladó járművet kikerül. A játékban szereplő utcahálózat nem biztosít lehetőséget előzésre semmilyen jármű részére.

Kereszteződés: egy kereszteződés biztosítja, hogy az ide érkező utakról átkanyarodhassunk az innen kivezető utakra. Az irányváltásokat a kereszteződésre jellemző táblák, illetve lámpák szabályozzák.

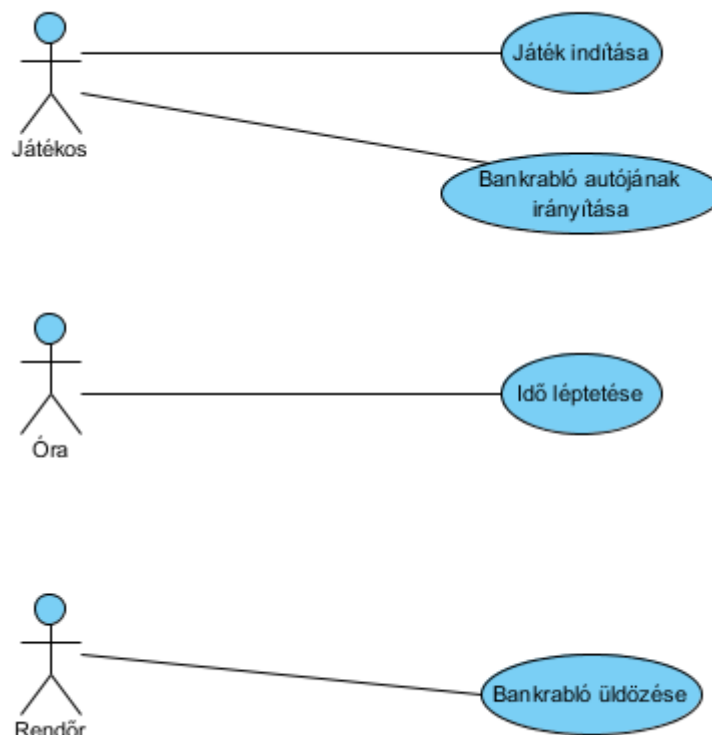
Kereszteződésben való kanyarodás: miután egy jármű eléri az útja végén található kereszteződést, akkor a kereszteződés, ha engedélyezi a továbbhaladását, akkor lekerül az előző útról és rákerül a kiválasztott útra. Ha a kereszteződés nem engedi őt tovább, akkor addig vár, amíg a kereszteződés engedélyt nem ad a továbbhaladásra.

Lámpa: a kereszteződéseken való áthaladás szabályozását befolyásoló és irányító eszköz. Zöld esetén a kereszteződésen való áthajtás engedélyezett, piros esetén tiltott. Ezeket a szabályokat a városban közlekedő autók betartják, kivételt képeznek ez alól a rabló és a rendőrség járművei.

Stop-tábla: a kereszteződéseken való áthaladás szabályozását befolyásoló és irányító eszköz. A közlekedő autóknak kötelességük megállni a táblánál egy adott időre. Továbbhaladásuk csak akkor engedélyezett, ha nem észlelnek olyan járművet a közelben mellyel szemben elsőbbségadási kötelezettségük lenne. Az imént említett szabályok alól felmentettként jelennek meg a rendőr(ök) és a rabló.

2.5 Essential use-case-ek

2.5.1 Use-case diagram



2.5.2 Use-case leírások

Use-case neve	Játék indítása
Rövid leírás	A játékos új játékot kezd. A pálya kezdőállapota töltődik be.
Aktorok	Játékos
Forgatókönyv	1. A játékos kiválasztja a menüben a "Új játék indítása" címkét és rákattint. 2. A pálya kezdőállapota töltődik be, a játékos (bankrabló) autója egy véletlen helyen van a térképen.

Use-case neve	Bankrabló autójának irányítása
Rövid leírás	A játékosnak a bankrabló autóját irányítva a rejtkehelyre kell érnie úgy, hogy ne kapják el a rendőrök.
Aktorok	Játékos
Forgatókönyv	1. A játékos a billentyűzet segítségével (fel, le, balra, jobbra nyilak) irányíthatja az autót 2. A autó ennek megfelelően mozog

Use-case neve	Idő léptetése
Rövid leírás	Bizonyos időközönként lépteti az időt, ami alapján mozognak a városi autók, rendőrök, illetve váltanak a lámpák.
Aktorok	Óra
Forgatókönyv	-

Use-case neve	Bankrabló üldözése
Rövid leírás	Autójával a bankrablót (aki a játékos) üldözi.
Aktorok	Rendőr
Forgatókönyv	1. A játékos kirabolja a Bankot. 2. A rendőrautók üldözőbe veszik

2.6 Napló

Kezdet	#idő	Részt- vevők	Leírás
2010.02.08. 20:00	3 óra	Koczka	Kiválasztja a fejlesztő eszközöket, létrehozza a fejlesztési környezetet, a többi csapattag felé elkészíti a használati útmutatókat.
2010.02.13 13:00.	2 óra	Koczka	Koczka összesíti a projekt ütemtervét, előkészíti a csapat dokumentációnak keretét, feltölti őket a közös tárolóba
2010.02.13. 17:30	1 óra	Kovács	A Visual Paradigm UML rajzoló rendszer tanulmányozása
2010.02.13. 18:00	3 óra	Angeli Koczka Kovács Mikó	MSN értekezlet. Döntés: • Kovács megírja a követelmény definíciót és az essential use case-eket • Koczka megírja a projekt tervet • Angeli és Mikó megírja a szótárat és a feladatkírást • Ezen kívül mindig elolvassák a másik munkáját és véleményezik azt
2010.02.16. 20:00	3 óra	Angeli Mikó	a GoogleDocs-on közösen megírják a részletes 2.3. feladatléírást
2010.02.16. 20:00	2 óra	Kovács	2.1 Követelmény definíció megírása; az eddig elkészült dokumentumok átolvasása, javítása
2010.02.17. 14:00	2 óra	Koczka	2.2 Projekt terv megírása; az eddig elkészült dokumentumok átolvasása, javítása
2010.02.17. 19:00	3 óra	Angeli Mikó	a GoogleDocs-on közösen összegyűjtik és megszerkesztik a 2.4. szótárat
2010.02.17. 20:00	3 óra	Kovács	2.5. essential use case-ek megírása; az eddig elkészült dokumentumok átolvasása, javítása, formázása

3. Analízis modell kidolgozása

3.1 Objektum katalógus

3.1.1 Game

Ez az az objektum, ami az összes többit létrehozza, tartalmazza, megsemmisíti és az időzítő jeleit eljuttatja a többi objektumhoz. Ezen kívül a Game kezeli a játék fő eseményeit: például vesztes, nyeres, a Robber kiment a városból stb.

3.1.2 Map

Az úthálózatot tartalmazza (TrafficJunction, Road, TrafficLight, StopSign, Bank, Shelter, CityBorder).

3.1.3 TrafficJunction

Egy útkereszteződést testesít meg, ehhez kapcsolódnak Road-ok és más objektumok (Bank, Shelter).

3.1.4 Road

Egy utat testesít meg, amely egyirányú és egysávos, ezen mennek az autók. Több táblát tartalmazhat. Mozgatja a rajta levő kocsikat, megmondja egy kocsinak, hogy előtte milyen messze vannak különféle akadályok.

3.1.5 StopSign

Stop tábla, akkor enged át, ha az előtte lévő autó megállt.

3.1.6 TrafficLight

Közlekedési lámpa, periodikusan átenged/nem enged át.

3.1.7 Vehicle

Egy autót testesít meg, felelőssége az ütközések kezelése, tárolja, hogy éppen melyik Road-okon halad.

3.1.8 Robber

A bankrabló autója, a játékos irányítja.

3.1.9 Police

A rendőr autója, MI irányítja (vagy egy másik játékos), üldözi a Robber-t. Rá nem vonatkoznak a közlekedési táblák szabályai, de nem ütközik a többi autóval, csak a Robberrel. Feladata, hogy elkapja a Robbert.

3.1.10 Citizen

Városi autó, ami nem rendőr és nem bankrabló autója. Nem megy neki az előtte lévőnek, lelassít, ha az előtte lévő lassít.

3.1.11 Bank

A bank, ezt kell kirabolnia a játékosnak.

3.1.12 Shelter

A rejtékhely. Miután kirabolta a játékos a bankot, ide kell tud elérni anélkül, hogy elkapják a rendőrök. Ha ez sikerül, akkor megnyeri a játékot.

3.1.13 CityBorder

A város határa, ha ezen túl megy a játékos, nem üldözik tovább a rendőrök. Persze, ha visszatér a városba, akkor újra üldözőbe veszik.

3.1.14 Driver

Ő vezeti a járműveket, dönt a gyorsulásról, illetve arról, hogy a jármű merre forduljon a kereszteződésekben.

3.2 Osztályok leírása**3.2.1 Bank**

- **Felelősség**
 - A bank, ezt kell kirabolnia a játékosnak.
- **Össztályok**
 - Building
- **Interfészek**
 - nincs
- **Attribútumok**
- **Metódusok**

3.2.2 Building

- **Felelősség**
 - A Mapen elhelyezett kereszteződésekénél lévő épületek. (Shelter, Bank, CityBorder, stb.)
- **Össztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
- **Metódusok**
 - abstract void accept(IVehicleVisitor vehicleVisitor) - Ha egy jármű érkezik az épülethez ez hívódik meg.

3.2.3 Citizen

- **Felelősség**
 - Városi autó, ami nem rendőr és nem bankrabló autója.
- **Össztályok**
 - Vehicle
- **Interfészek**
 - nincs
- **Attribútumok**
 - boolean goingOutOfCity - Akkor igaz, ha a kocsí eleje már belement egy cityborderes kereszteződésbe, de a kocsí vége még nem ért ki a városból. A jármű még rajta van valamelyik Road-on.
- **Metódusok**
 - nincs

3.2.4 CitizenAI

- **Felelősség**
 - Ez az osztály azért felelős, hogy a városi autókat vezesse.
- **Ősosztályok**
 - nincs
- **Interfészek**
 - IDriver
- **Attribútumok**
 - Vehicle vehicle - Az irányított városi autó.
- **Metódusok**
 - nincs

3.2.5 CityBorder

- **Felelősség**
 - A városhatár. Itt tudnak a Citizen-ek kilépni/belépni a városba, illetve itt tud a bankrabló eltűnni ideiglenesen a rendőrök elől.
- **Ősosztályok**
 - Building
- **Interfészek**
 - nincs
- **Attribútumok**
 - nincs
- **Metódusok**
 - nincs

3.2.6 Game

- **Felelősség**
 - Ez az osztály hozza létre és semmisíti meg a térképet és a kocsikat, meghatározza a kocsik viselkedését.
 - Ez az osztály "irányítja" a játék mérföldköveit
- **Ősosztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - Vehicle[0..*] vehicles - A járműveket tartalmazó tömb.
 - Map map - A térkép.
 - Timer timer - A különböző objektumok időzítéséért felelős.
 - Robber[0..*] robbers - A játékban résztvevő rabló(k).
 - Police[0..*] polices - A játékban résztvevő rendőrök.
- **Metódusok**
 - Robber[] getRobbers() - Visszaadja a rablókat tároló tömböt.
 - void bankRobbed() - Akkor hívódik meg, amikor egy rabló kirabolja az első bankot.
 - void citizenLeftCity(Citizen citizen) - A paraméterül kapott citizen jármű városelhagyásakor hívódik meg, megsemmisíti a citizent és újat hoz létre.
 - void robberCaught(Robber robber) - Ha egy rendőr elér egy rablót, ez a függvény hívódik meg. A játék sikertelen végét jelenti.

- void robberReachedShelter(Robber robber) - Ha a rabló eléri a búvóhelyt, ez a függvény hívódik meg. A játék sikeres végét jelenti.
- void robberDied(Robber robber) - Ez a függvény kezeli le, ha a rablónak elfogynak az élete és úgy ér véget sikertelenül a játék.
- void robberDisappeared(Robber robber) - Ez a függvény kezeli le, ha a rabló "kimenekül" a városhatáron kívülre.

3.2.7 IDriver

- **Felelősség**
 - A járművek vezetése ezen az interfacen keresztül történik.
- **Össztályok**
 - nincs
- **Interfészek**
- **Attribútumok**
- **Metódusok**
 - Road chooseDirection(Road[] roads) - Visszaadja, hogy merre halad tovább a Vehicle a kereszteződésekénél.
 - double getAcceleration() - Visszaadja a Vehicle gyorsulását az akadályokat figyelembe véve.

3.2.8 ITick

- **Felelősség**
 - Az egész játék időzítése ezen az interfacen keresztül történik. Lámpakezelés, autó mozgatása, játék vezérlése.
- **Össztályok**
 - nincs
- **Interfészek**
- **Attribútumok**
 - nincs
- **Metódusok**
 - void doTick(int time, int deltaTime) - Meghívódik, ha éppen egy időegységnyivel később van. Folyamatosan hívódik a Timer osztályon keresztül. Ez "lépteti" a játékot.

3.2.9 IVehicleVisitor

- **Felelősség**
 - Az autó különböző Buildingek elérésekor felvetődő különböző szituációk (pl Bank kifosztása) lekezelése. A különböző épületek ezen keresztül kommunikálnak a járművekkel.
- **Össztályok**
 - nincs
- **Interfészek**
- **Attribútumok**
 - nincs
- **Metódusok (Visitor Pattern)**
 - void visit(bank : Bank) - Meghívódik, ha egy jármű olyan kereszteződésbe ér, ahol Bank van (a TrafficJunction-ba ért Vehicle esetén a TrafficJunction meghívja az összes tartalmazott Buildingre a void accept(IVehicleVisitor vehicleVisitor)-t az adott Vehicle-el.)

- void visit(shelter : Shelter) - Meghívódik, ha egy jármű olyan kereszteződésbe ér, ahol Shelter van.
- void visit(cityBorder : CityBorder) - Meghívódik, ha egy jármű olyan kereszteződésbe ér, ahol városhatár van.

3.2.10 KeyboardPlayer

- **Felelősség**
 - Ez az osztály azért felelős, hogy billentyűzettel irányíthassuk a járművünket.
- **Össztályok**
 - nincs
- **Interfészek**
 - IDriver
- **Attribútumok**
 - nincs
- **Metódusok**
 - nincs

3.2.11 Map

- **Felelősség**
 - Az úthálózatot tartalmazza. (TrafficJunction, Road, TrafficSign, Building)
- **Össztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - TrafficJunction[0..*] trafficJunctions - A kereszteződések tartalmazó tömb.
 - Road[0..*] roads - Az utakat tartalmazó tömb.
 - TrafficSign[0..*] trafficsigns - A táblákat tartalmazó tömb.
 - Building[0..*] buildings - Az épületeket tartalmazó tömb (amikkel interakcióba léphet valamelyik jármű).
- **Metódusok**
 - Building[] getBuildings() - Lekéri az épületeket tartalmazó tömböt.
 - TrafficJunction[] getTrafficJunctions - Lekéri a kereszteződések tartalmazó tömböt.
 - Road[] getRoads() - Lekéri az utakat tartalmazó tömböt.
 - TrafficSign[] getTrafficSigns() - Lekéri a táblákat tartalmazó tömböt.

3.2.12 Police

- **Felelősség**
 - A rendőr autója, MI irányítja, üldözi a Robber-t.
- **Össztályok**
 - Vehicle
- **Interfészek**
 - nincs
- **Attribútumok**
 - IDriver defaultdriver - A rendőrautó alapértelmezett vezetője (irányítója). Legtöbbször a PoliceAI de ugyanakkor lehetőséget biztosít arra hogy KeyboardPlayer-ként más játékos is irányítsa a rendőrt.

- boolean active - A rendőr állapotát jelző attributum. (true - a rendőr üldözi a rablót; false - a rendőr nem üldözi a rablót).
- **Metódusok**
 - void activate () - Az adott rendőr aktiválásáért felelős függvény. (aktív -> defaultdriver veszi át az irányítást).
 - void deactivate() - Az adott rendőr deaktiválásáért felelős függvény. (deaktív -> citizen-ként viselkedik).
 - boolean getActivate() - A függvény segítségével lekérdezhető az adott rendőr állapota (aktivitása).
 - void setDefaultDriver (IDriver : defaultDriver) - A rendőrautó alapértelmezett vezetőjének (irányítójának) beállításáért felelős függvény.

3.2.13 PoliceAI

- **Felelősség**
 - Ez az osztály azért felelős, hogy a rendőrt vezérelje, és az elkapja a rablót.
- **Össztályok**
 - nincs
- **Interfészek**
 - IDriver
- **Attribútumok**
 - Vehicle vehicle - Az irányított rendőr járműve.
 - Map map - A térkép amit a rendőr ismer.
 - Game game - A játékot irányító Game osztályra egy referencia.
- **Metódusok**
 - nincs

3.2.14 Road

- **Felelősség**
 - Egy utat testesít meg, amely egyirányú és egysávos, melyen a járművek közlekednek. Tartalmazhat különböző közlekedést szabályozó eszközöket (táblák, lámpa).
- **Össztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - VehicleOnRoad[0..*] vehicles - Az úton tartózkodó (közlekedő) járműveket tartalmazó tömb.
 - TrafficJunction[2] trafficJunctions - Az út két végén található kereszteződéseket tartalmazó tömb. Az elsőből a másodikba megy az út.
 - TrafficSign[0..*] trafficSigns - Az úton található különböző közlekedést szabályozó eszközök tömbje (lámpák, táblák). Mindegyik az út végén található (egyirányúság).
- **Metódusok**
 - double getLength() - Kiszámolja és visszaadja az út hosszát.
 - boolean isOccupied(TrafficJunction trafficJunction) - Az út azon részének foglaltságát jelzi ami még belelóg a kereszteződésbe.
 - void addVehicle(Vehicle vehicle) - A függvény az útra helyez egy járművet.
 - void removeVehicle(Vehicle vehicle) - A függvény levesz az útról egy járművet.

- `double getNearestPhysicalBarrierDistance(Vehicle vehicle)` - A függvény a legközelebbi fizikai akadálytól való távolságot (az út irányában) adja vissza (fizikai: olyan dolog amivel ütközni lehet pl.: jármű)
- `double getNearestLogicalBarrierDistance(Vehicle vehicle)` - A függvény a legközelebbi logikai akadálytól való távolságot (az út irányában) adja vissza (logikai: közlekedési szabály(ok)ból eredő akadály)
- `Vehicle getNearestCar(Vehicle vehicle)` - Egy autótól az adott úton lévő legközelebbi jármű lekérdezésére szolgáló metódus (az út irányában).
- `TrafficJunction[] getTrafficJunctions()` - A függvény visszatérési értékül adja az adott úthoz tartozó közlekedési csomópontokat.
- `void moveVehicle(Vehicle vehicle, double distance)` - A függvény az adott járművet a megadott távolságban mozgatja az úton.
- `Vehicle getVehicleAtSign (TrafficSign trafficSign)` - A függvény visszatér azzal a járművel, akire az adott közlekedési jelzés hat.

3.2.15 Robber

- **Felelősség**
 - A bankrabló autója, a játékos irányítja.
- **Ősosztályok**
 - Vehicle
- **Interfészek**
 - nincs
- **Attribútumok**
 - `int lives` - A játékos életeinek száma.
 - `boolean wanted` - A játékost körözik-e a rendőrök. A bankrablásnál vált true-ra.
 - `boolean goingOutOfCity` - Akkor igaz, ha a kocsí eleje már belement egy CityBorderes kereszteződésbe, de a kocsí vége még nem ért ki a városból. Még rajta van valamelyik Road-on.
 - `TrafficJunction cityBorderTrafficJunction` - Az a kereszteződés, amelyiknél kilépett a rabló a városból.
- **Metódusok**
 - `void goBackToCity()` - Ez a függvény hívódik meg akkor, ha a rabló visszatér a városba.

3.2.16 Shelter

- **Felelősség**
 - A rejtékhely. Miután kirabolta a játékos a bankot, ide kell tud elérni anélkül, hogy elkapják a rendőrök. Ha ez sikerül, akkor megnyeri a játékot.
- **Ősosztályok**
 - Building
- **Interfészek**
 - nincs
- **Attribútumok**
- **Metódusok**

3.2.17 StopSign

- **Felelősség**
 - Stop tábla, akkor enged át, ha az előtte lévő autó megállt.
- **Ősosztályok**

- TrafficSign
- **Interfészek**
 - nincs
- **Attribútumok**
 - Road road - Az az út, amelyiken az adott tábla elhelyezkedik.
- **Metódusok**
 - nincs

3.2.18 Timer

- **Felelősség**
 - A játék órája. Az időzítésért felelős.
- **Összaltályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - ITick[] tickers - Azon objektumok, amelyek függnnek az idő múlásától. (pl.: autók, lámpák, stb.) Ezeket kezeljük folyamatosan.
 - double deltaTime - Ez az a delta érték amennyi időnként újra Tick-eljük az objektumokat.
- **Metódusok**
 - void start() - Ez indítja el az időt (egy szálat).
 - void stop() - Ez megállítja az időt.
 - void setDeltaTime(double deltaTime) - Itt lehet beállítani a fent leírt időközöket.

3.2.19 TrafficJunction

- **Felelősség**
 - Egy útkereszteződést testesít meg, ebbe vannak az Road-ok és más objektumok (Bank, Shelter).
- **Összaltályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - Road[0..*] inRoads - A kereszteződésbe bemenő utak tömbje.
 - Road[0..*] outRoads - A kereszteződésből kilépő utak tömbje.
 - Building[0..*] buildings - A kereszteződésben található épületek tömbje (amikkel interakcióba léphet valami).
 - double x - A kereszteződés x koordinátája.
 - double y - A kereszteződés y koordinátája.
- **Metódusok**
 - boolean isOccupied() - A kereszteződés foglaltságát jelzi (tartózkodik-e jármű a kereszteződésben).
 - void vehicleArrived(Vehicle vehicle) - Meghívódik amint érkezik egy autó a kereszteződésbe.
 - double getX() - Visszatérési értékül adja a kereszteződés x koordinátáját.
 - double getY() - Visszatérési értékül adja a kereszteződés y koordinátáját.
 - void setX(double x) - A kereszteződés x koordinátájának beállítására szolgáló metódus.

- void setY(double y) - A kereszteződés y koordinátájának beállítására szolgáló metódus.
- Road[] getInRoads() - A függvény a kereszteződésbe belépő utak tömbjét adja vissza.
- Road[] getOutRoads() - A függvény a kereszteződésből kilépő utak tömbjét adja vissza.

3.2.20 TrafficLight

- **Felelősség**
 - Közlekedés irányító lámpa, periodikusan átenged/nem enged át.
- **Össztályok**
 - TrafficSign
- **Interfészek**
 - nincs
- **Attribútumok**
 - nincs
- **Metódusok**
 - nincs

3.2.21 TrafficSign

- **Felelősség**
 - Közlekedési objektum, ami megmondja, hogy mehet-e a kocsi (tábla, lámpa stb).
- **Össztályok**
 - nincs
- **Interfészek**
 - ITick
- **Attribútumok**
- **Metódusok**
 - boolean isFree() - Megmondja, hogy ebben a pillanatban átmehet-e ezen a közlekedési objektumon a jármű. Ha true, akkor mehet át.

3.2.22 Vehicle

- **Felelősség**
 - Egy járművet testesít meg (Városi autó, rendőr, rabló, stb.).
- **Össztályok**
 - nincs
- **Interfészek**
 - ITick
 - IVehicleVisitor
- **Attribútumok**
 - driver: IDriver - Az autó vezetője, ő adja meg, hogy milyen gyorsulással és merre menjen az autó.
 - double velocity - Az autó pillanatnyi sebessége.
 - Road[] roads - Azon utak listája, amelyeken az autó pillanatnyilag rajta van. (Hosszú jármű v. kanyarodó út esetén fontos.)
 - VehicleType type - Az autó "tulajdonságait" adja meg.
- **Metódusok**
 - Road getFirstRoad() - Visszaadja azt a Road-t, amelyiken az autó eleje van.

- Road getDirection(Road[] roads) - Kiválasztjuk a továbbhaladási irányt.
- void addFirstRoad(Road road) - Hozzáadja a Road[] listához azt az utat, amire az autó eleje rátért.
- void removeLastRoad() - Leveszi a Road[] listáról legutolsó utat, amit az autó hátulja éppen elhagy.
- double getVelocity() - Visszaadja a jármű pillanatnyi sebességét.
- void crash(Robber robber) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
- void crash(Citizen citizen) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
- void crash(Police police) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
- void acceptCrash(Vehicle vehicle) - Meghívódik, ha a járműbe belementek.
- void outOfCity() - Meghívódik, ha a jármű letért az összes Roadról (vagyis kiment a városból).

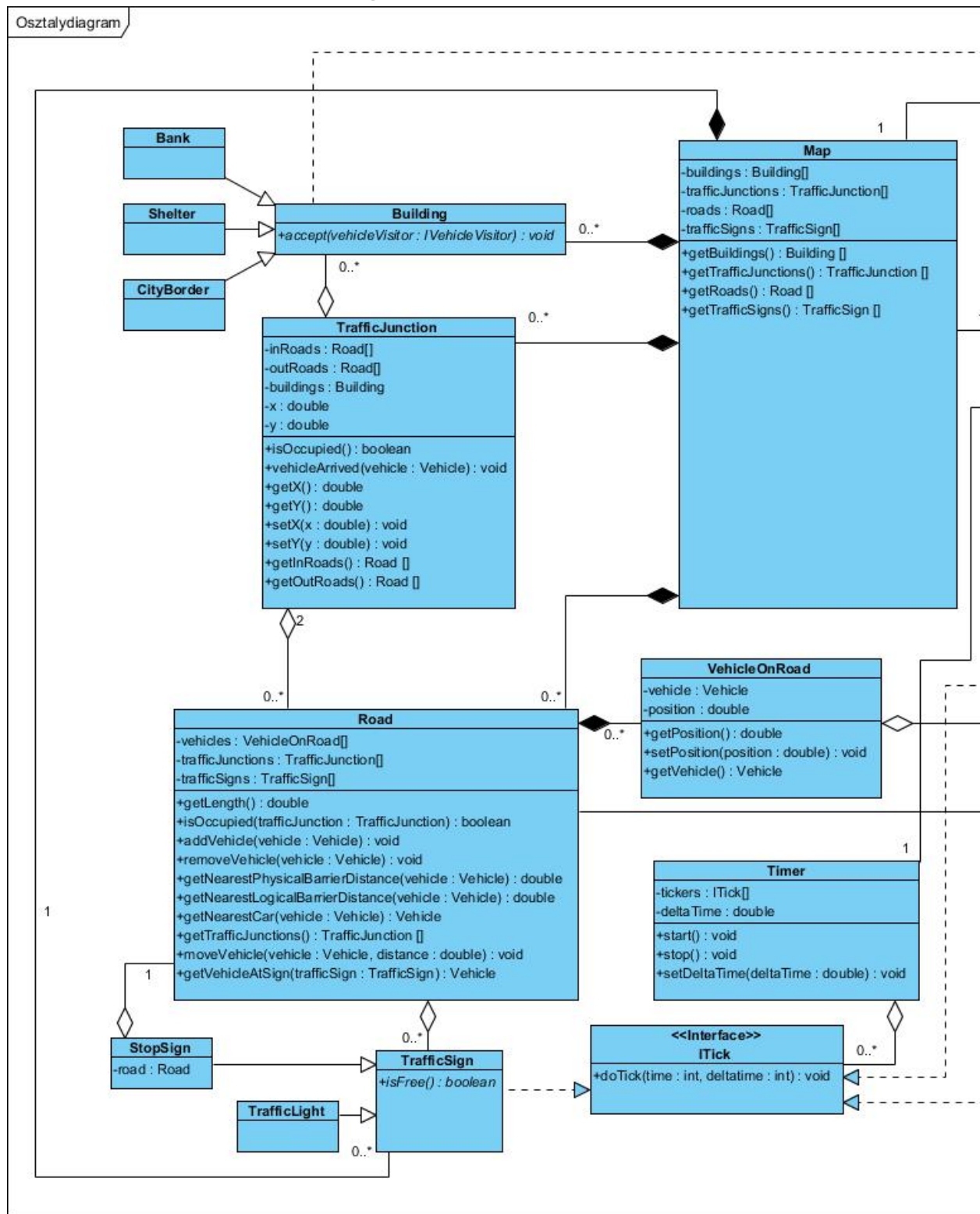
3.2.23 VehicleOnRoad

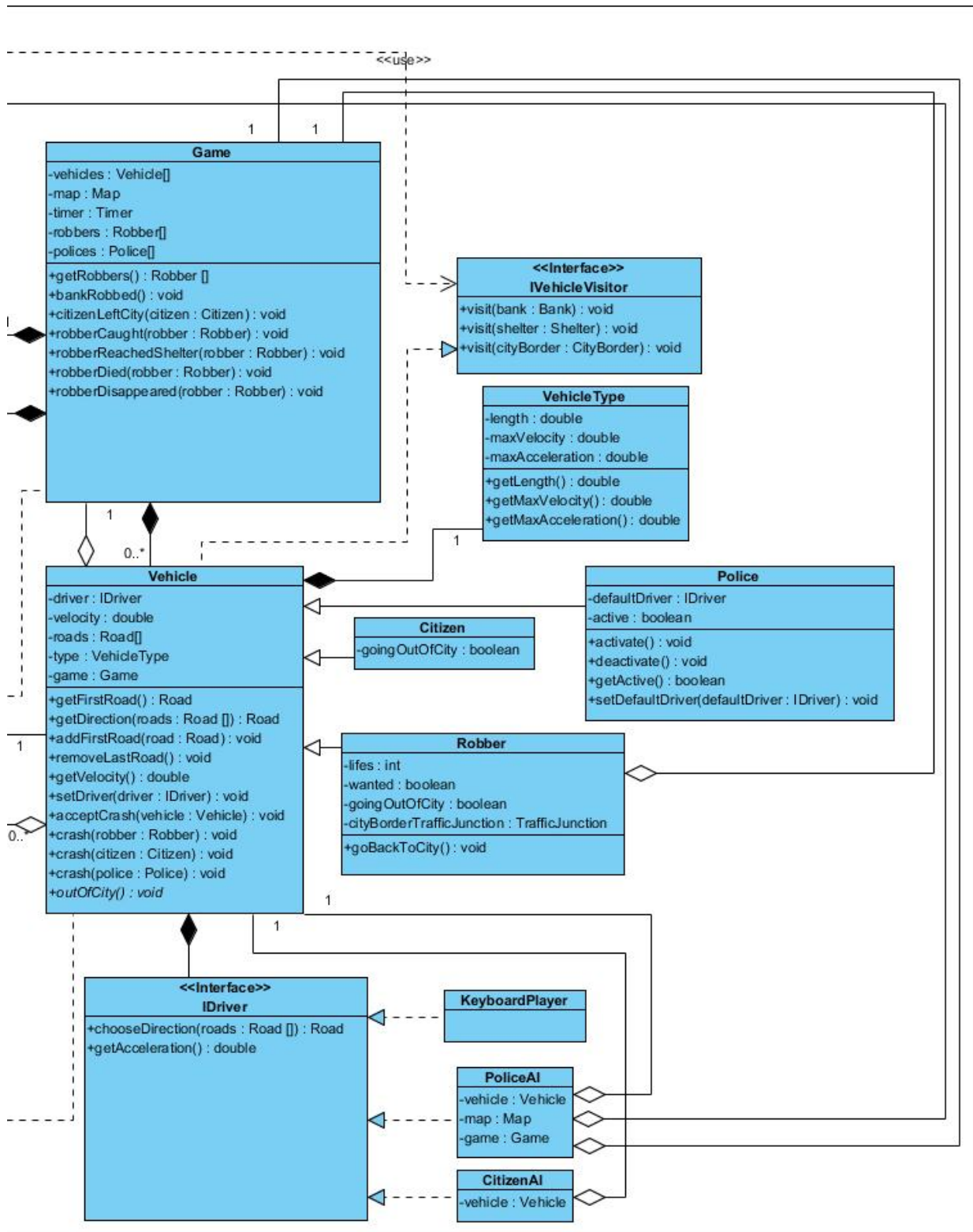
- **Felelősség**
 - Egy járművet bedobozolunk. Ezen keresztül tárolja a Road osztály a járműveket. Itt tároljuk a jármű helyét az adott úton.
- **Ősosztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - vehicle Vehicle - A jármű, amit tartalmaz .
 - double position - A jármű pozíciója az adott úton.
- **Metódusok**
 - double getPosition() - visszaadja a jármű position attribútumát.
 - void setPosition(double position) - Beállíthatjuk a jármű pozícióját, például az autó mozgásánál az úton, ezt állítjuk.
 - Vehicle getVehicle() - Visszaadja a tárolt járművet.

3.2.24 VehicleType

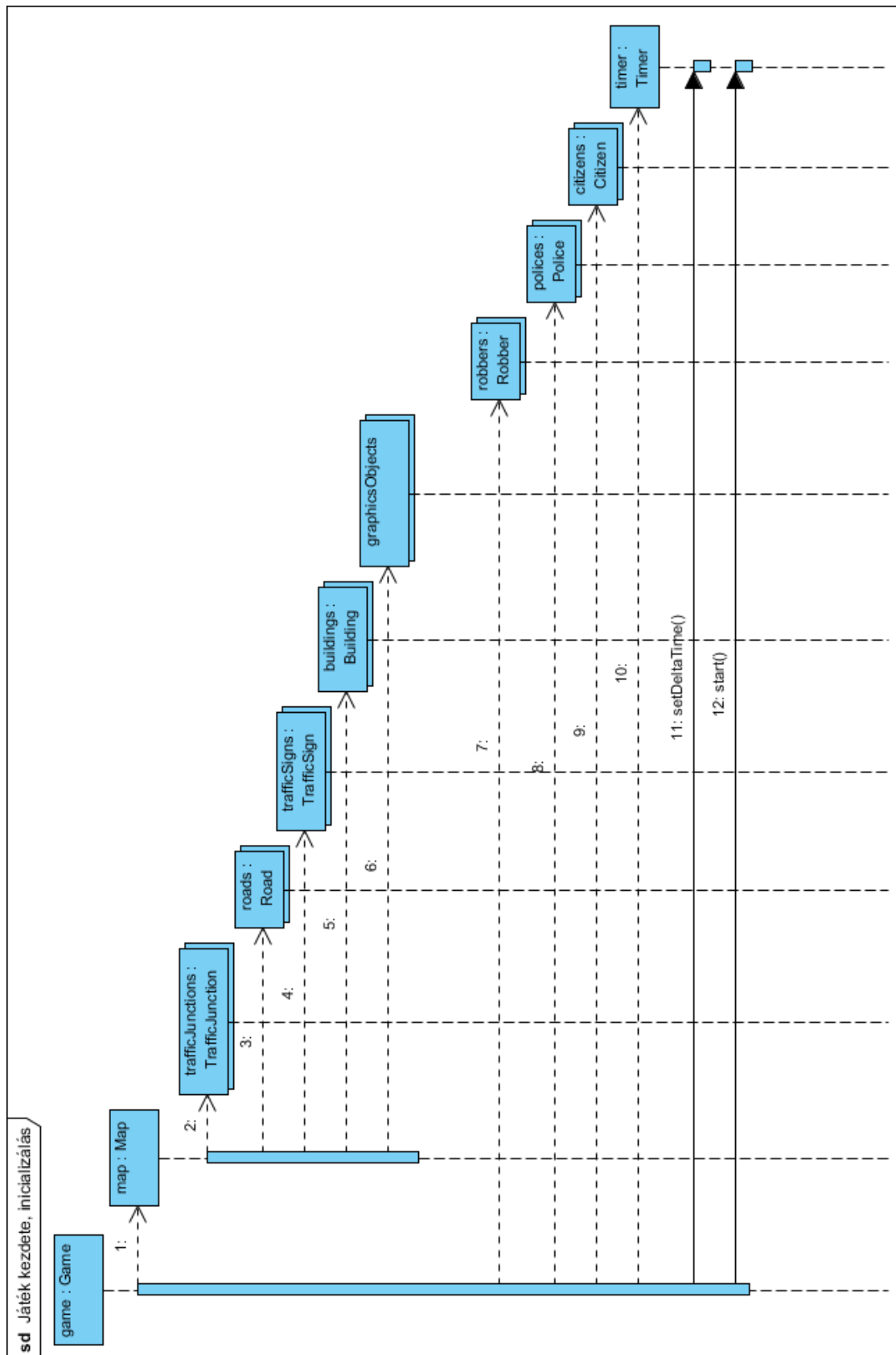
- **Felelősség**
 - Egy járműre jellemző tulajdonságokat tároljuk ebben az osztályban.
- **Ősosztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - double length - A jármű hossza.
 - double maxVelocity - A jármű maximális sebessége.
 - double maxAcceleration - A jármű maximális gyorsulása.
- **Metódusok**
 - double getLength() - Visszaadja a jármű hosszát.
 - double getMaxVelocity() - Visszaadja a jármű maximális sebességét.
 - double getMaxAcceleration() - Visszaadja a jármű maximális gyorsulását.

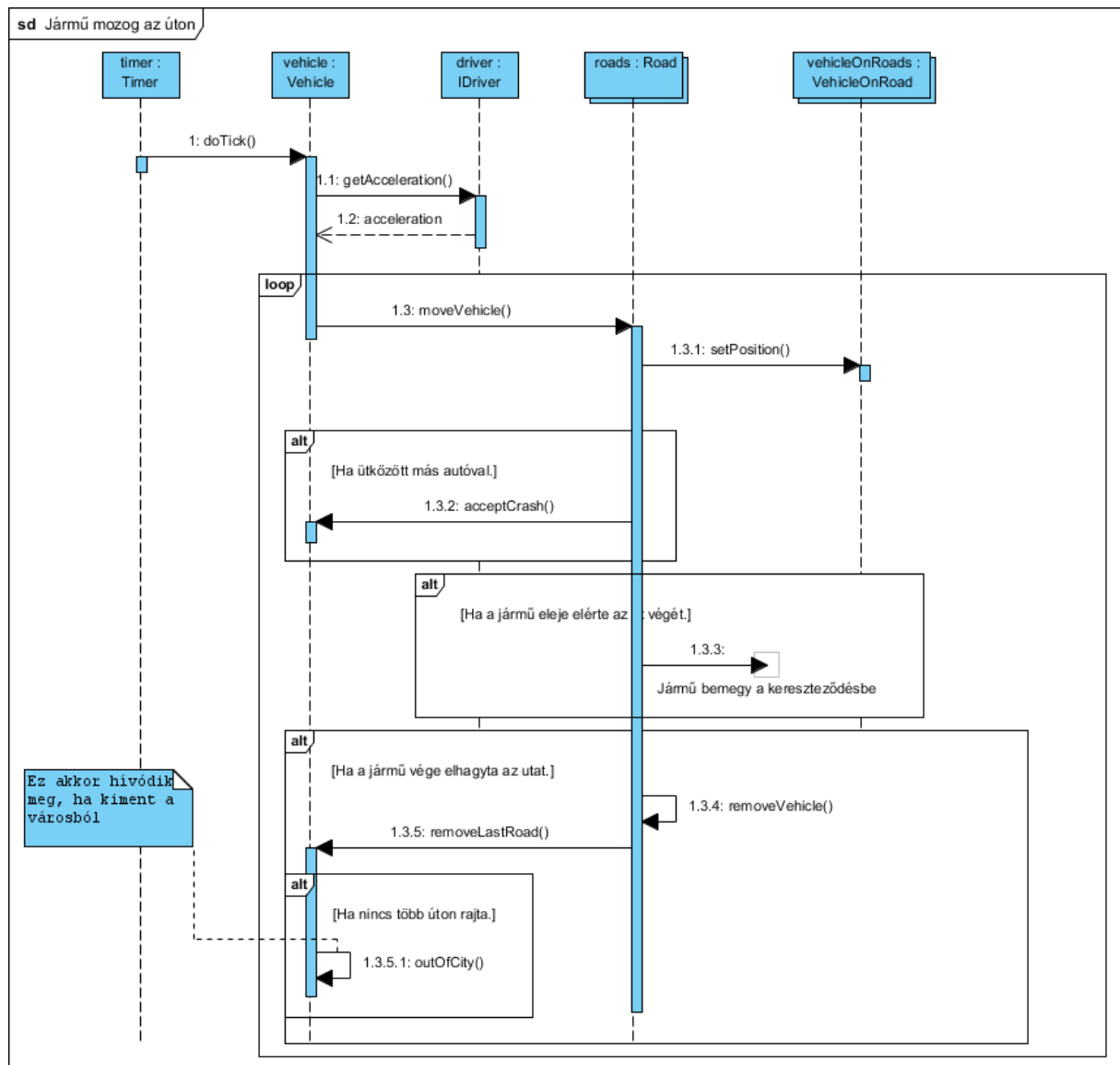
3.3 Statikus struktúra diagramok

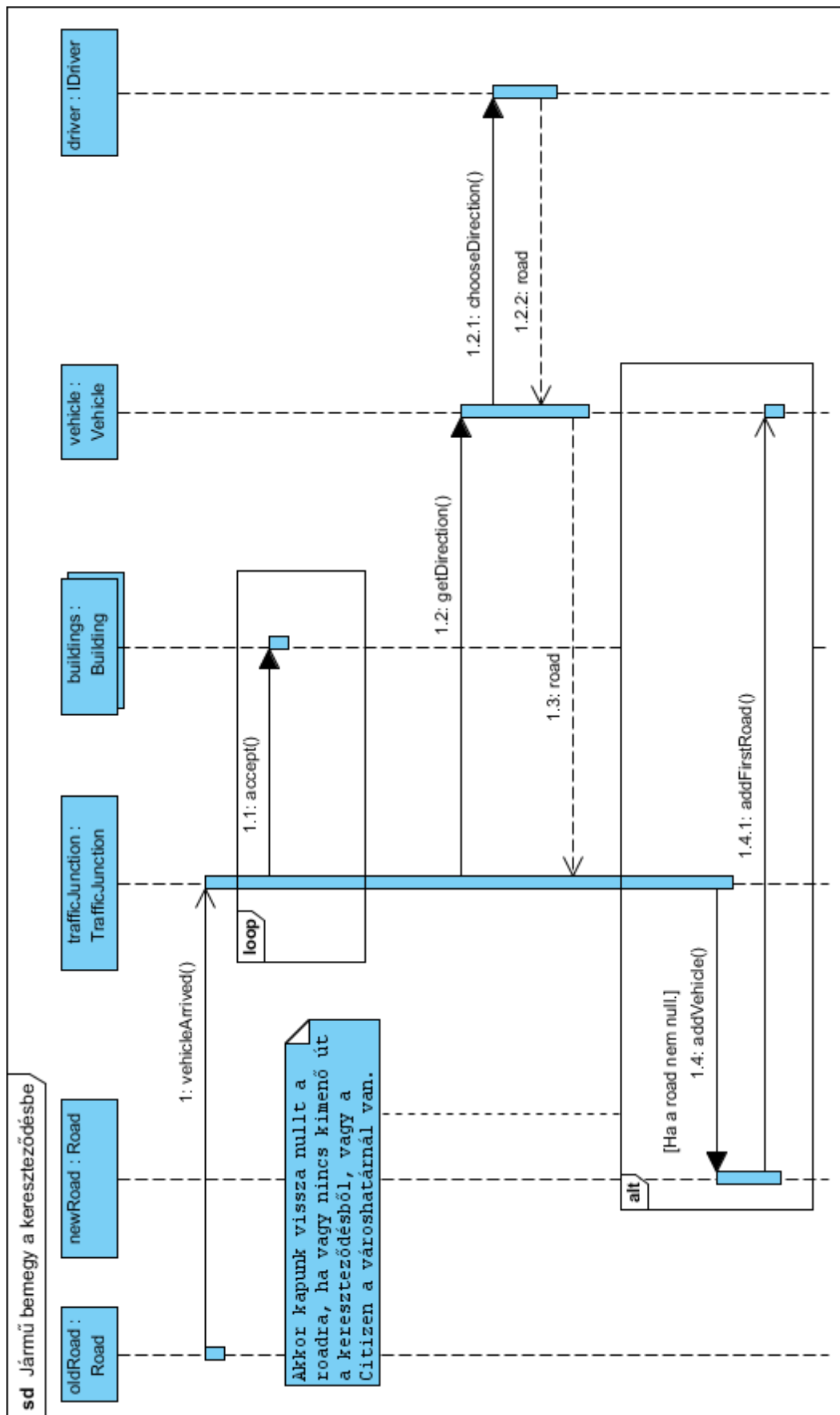


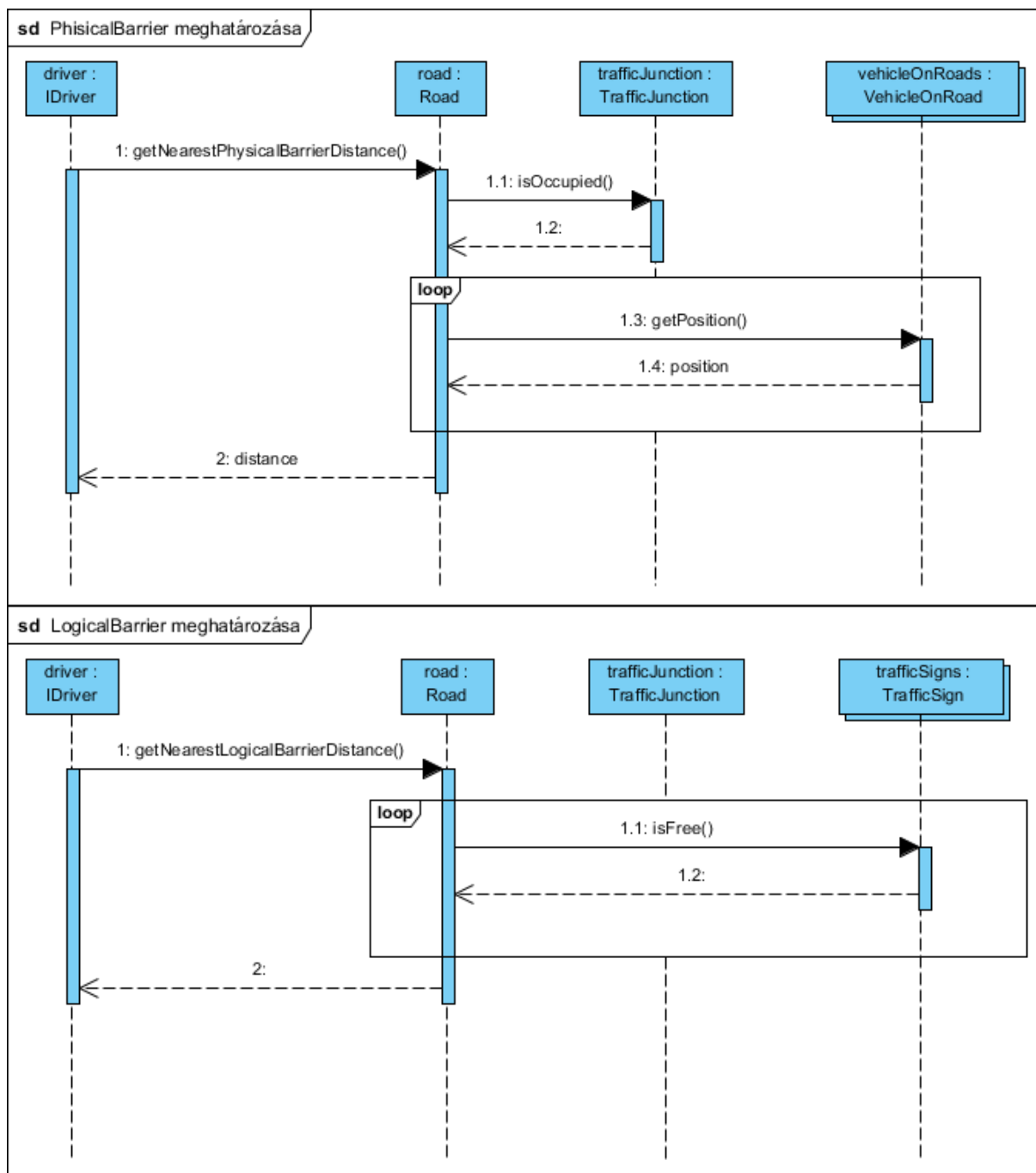


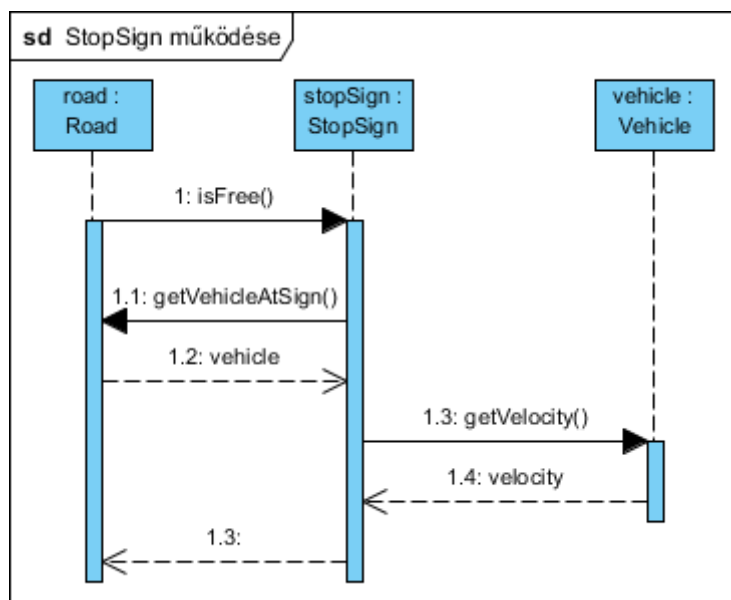
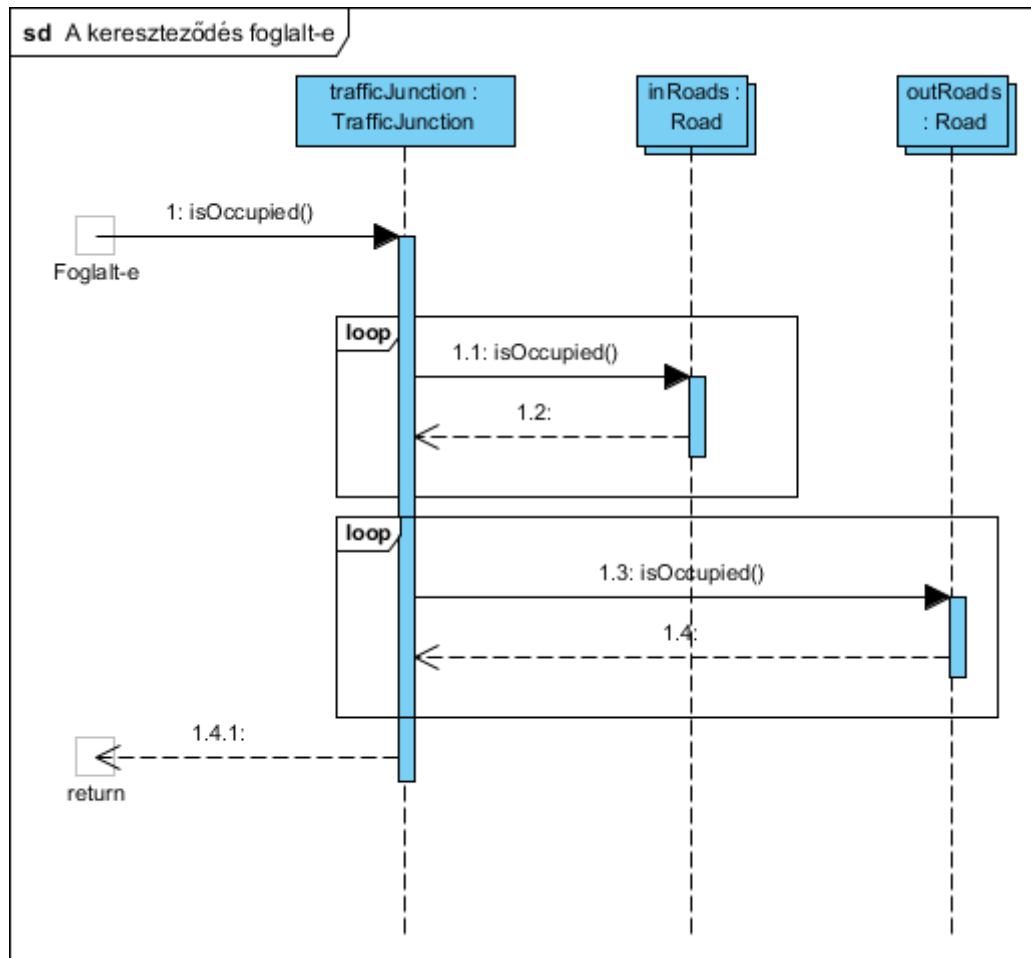
3.4 Szekvencia diagramok

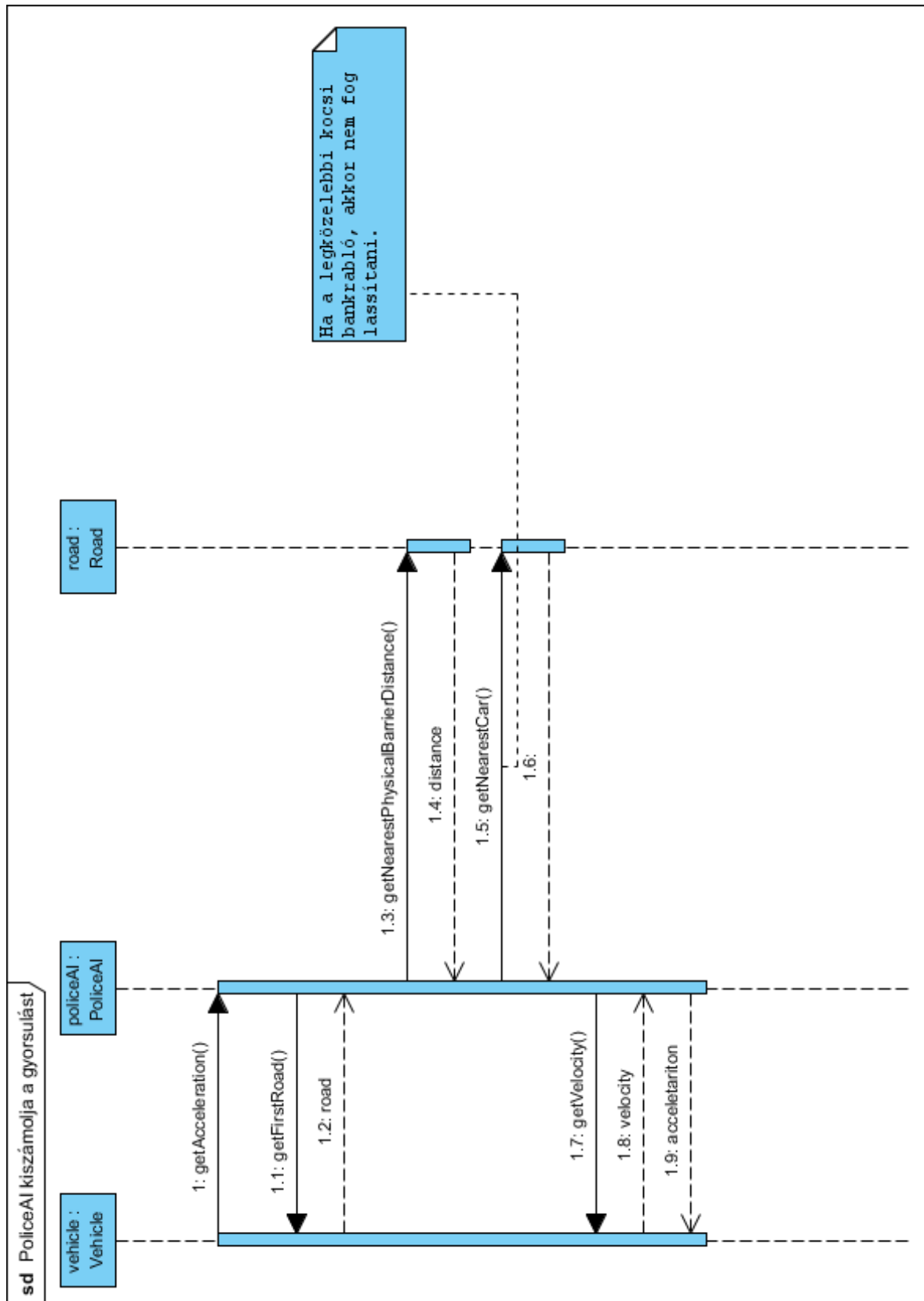


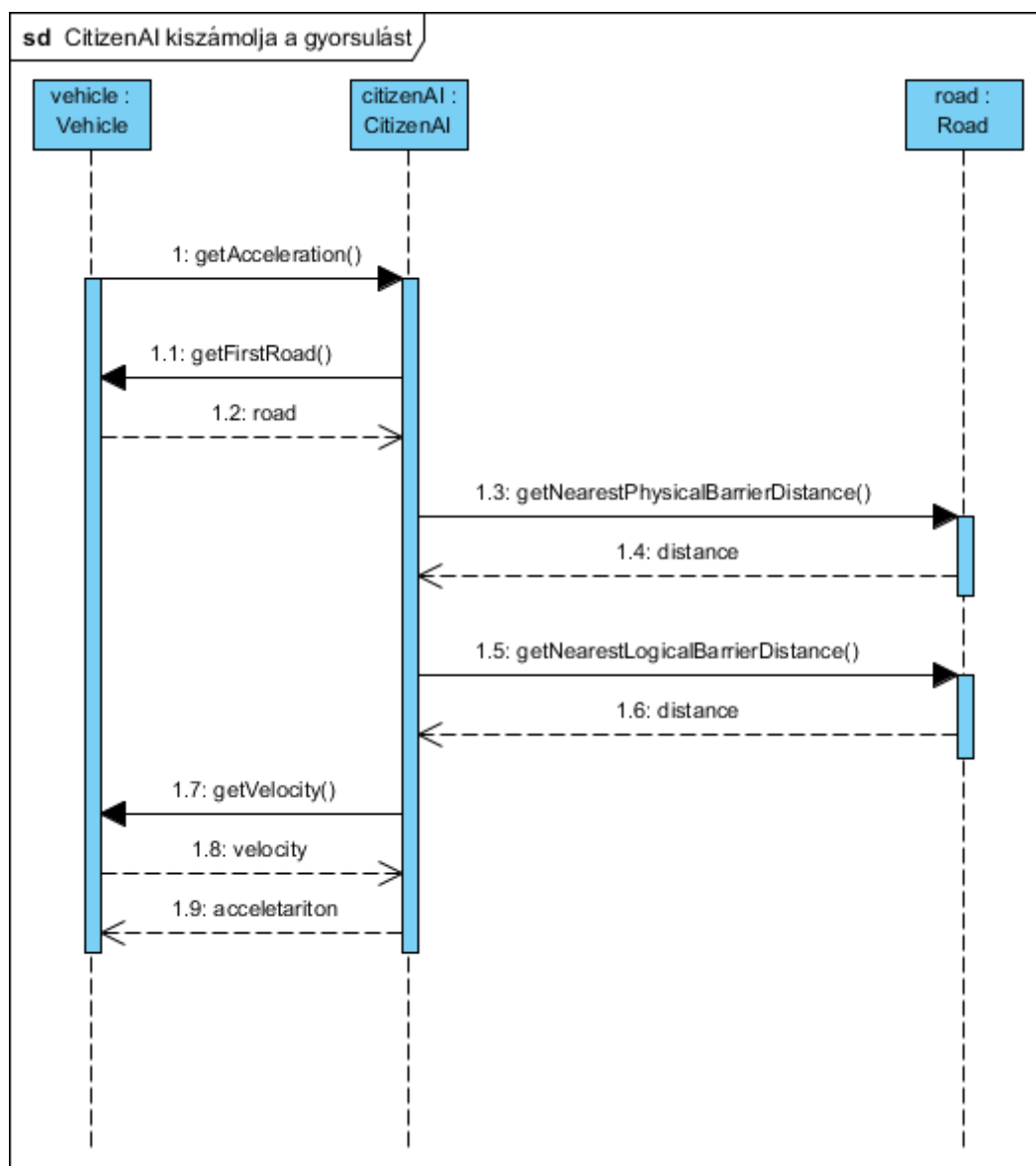


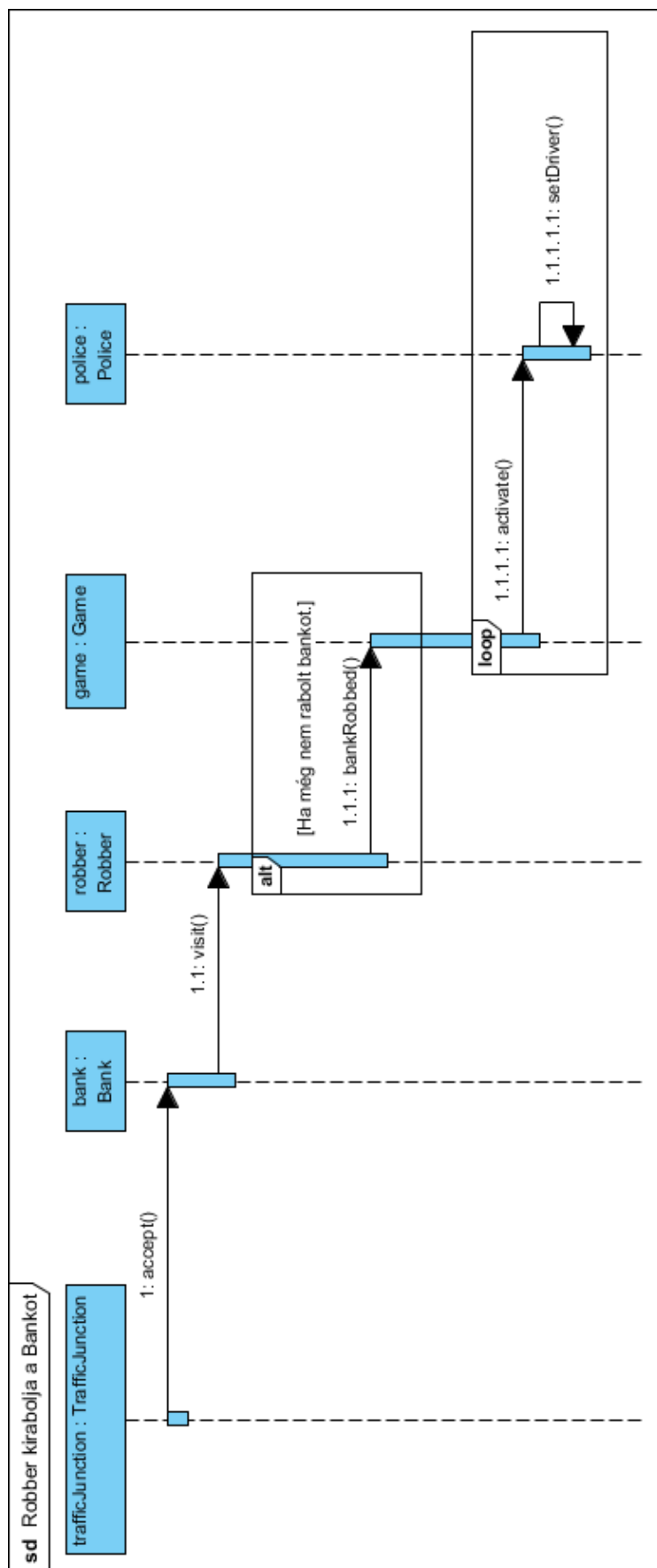


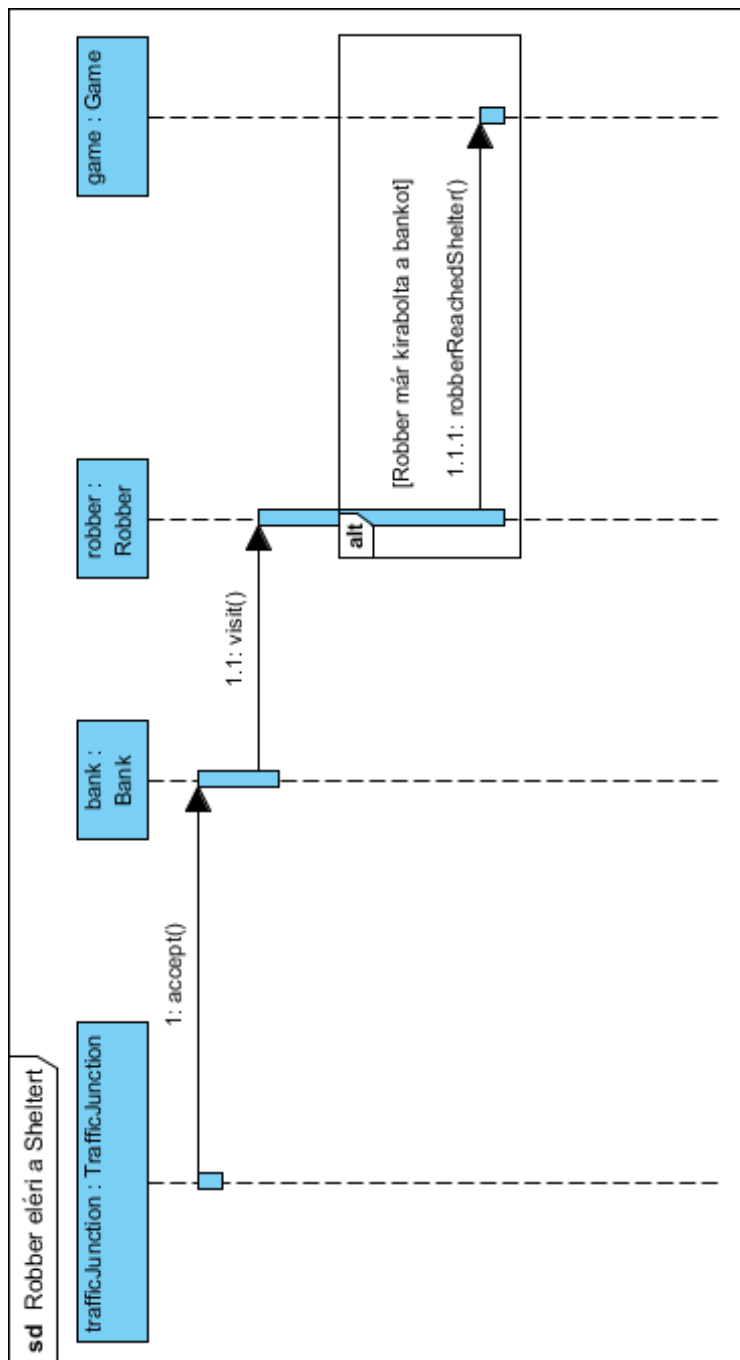


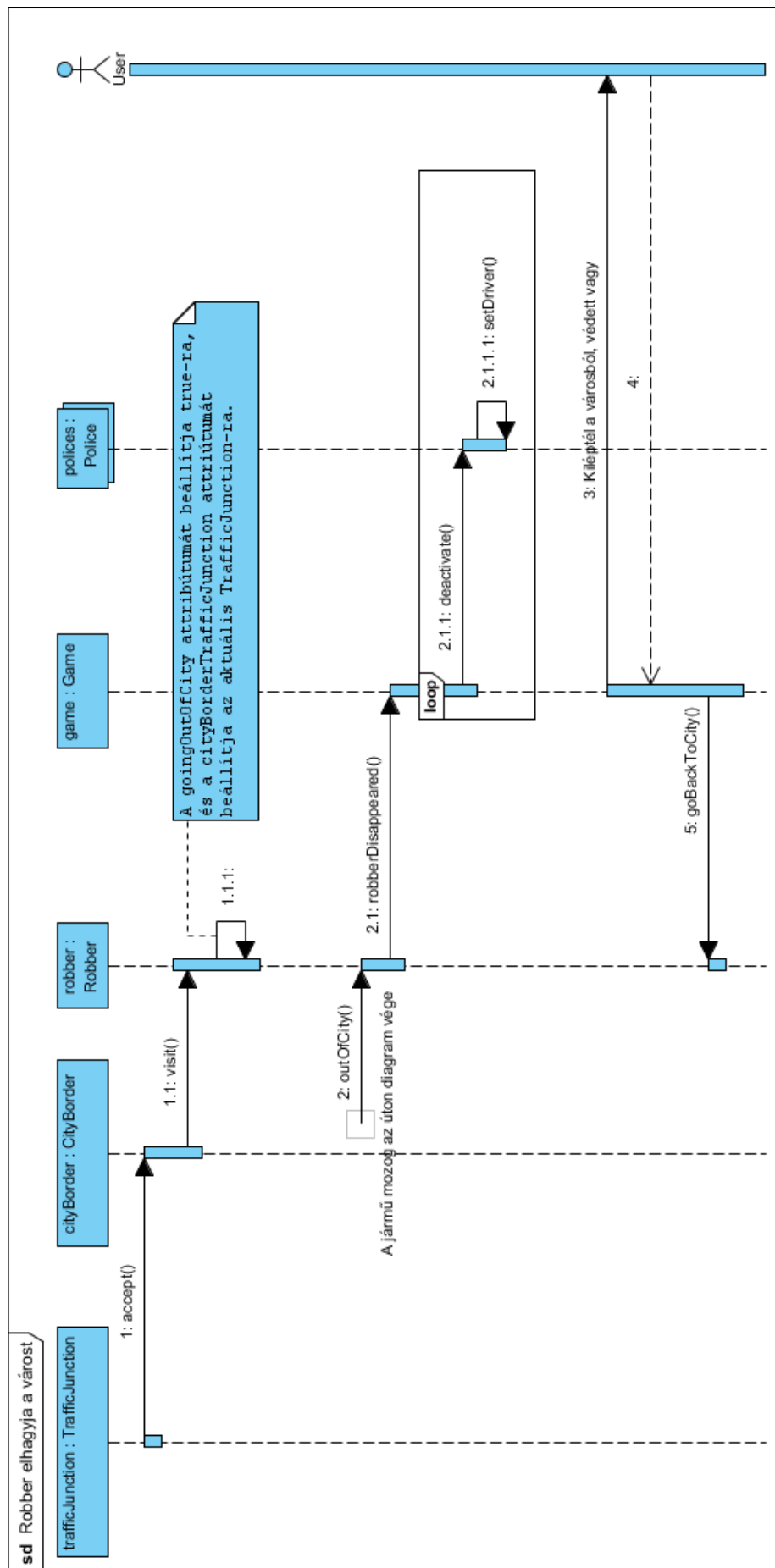


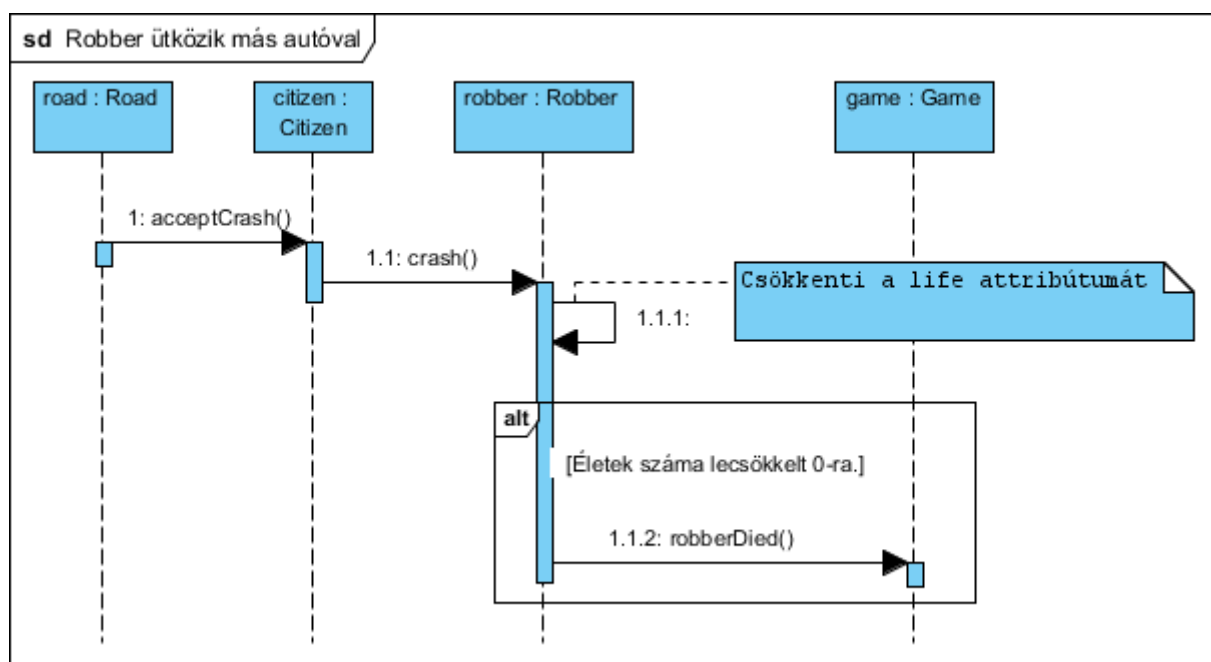


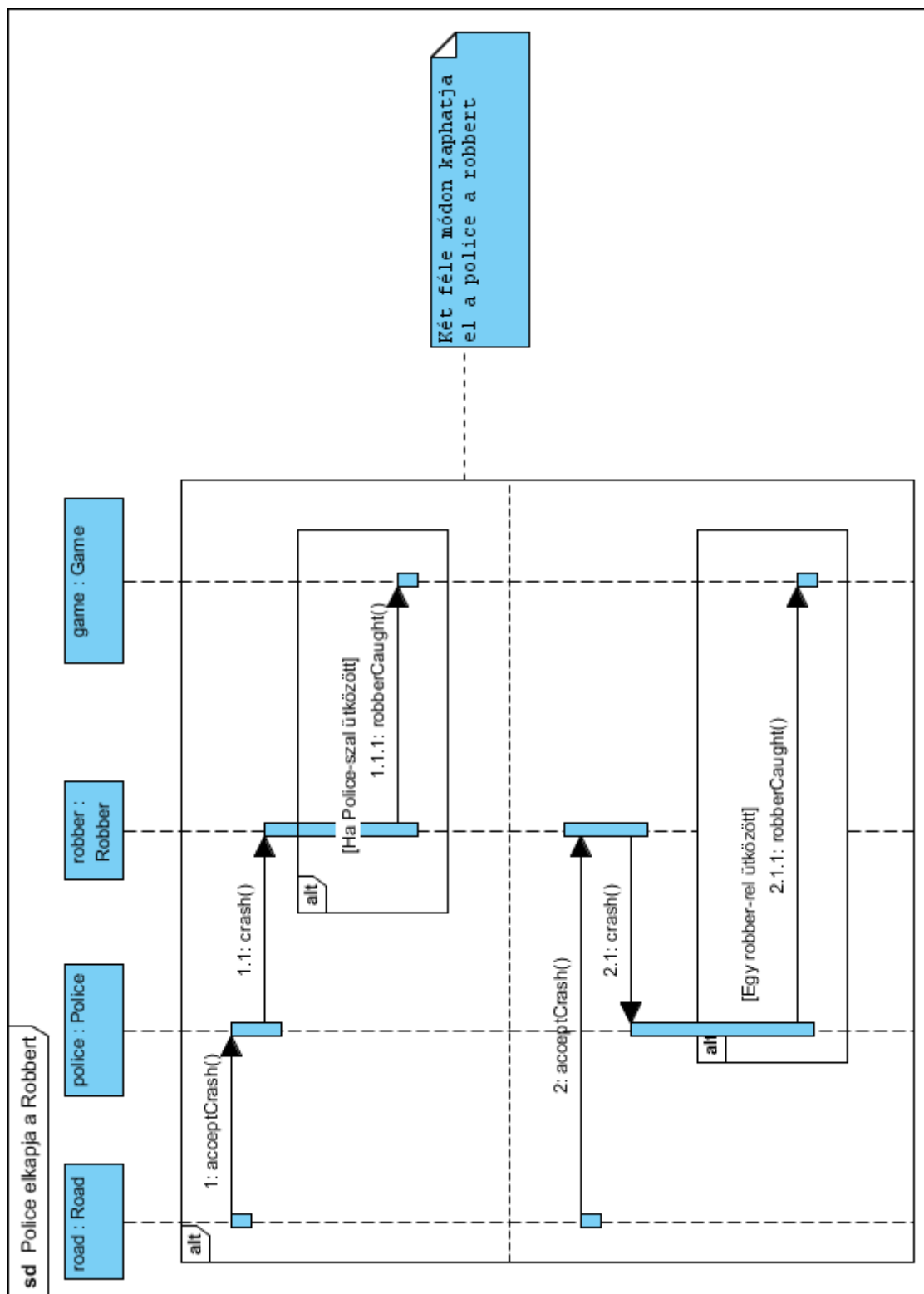


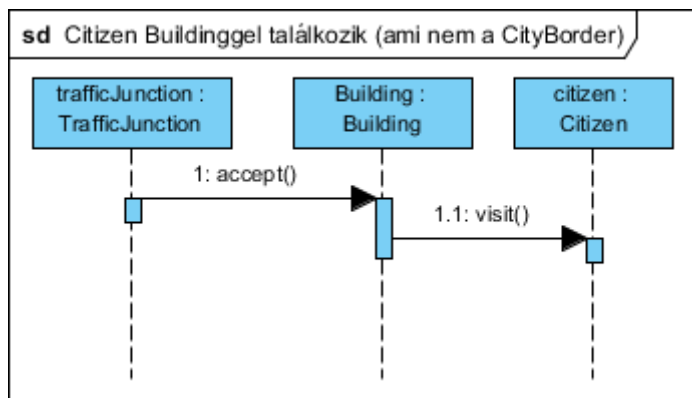
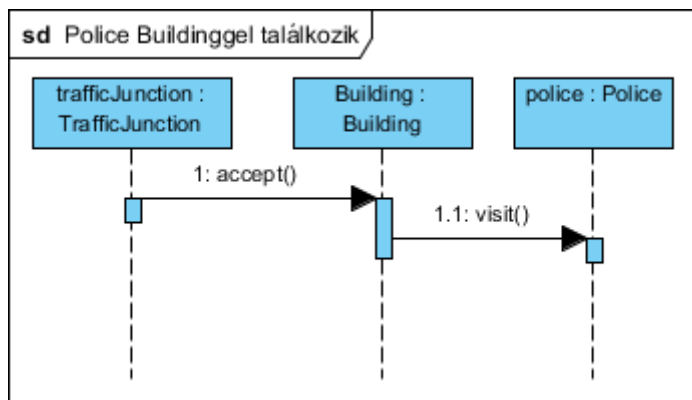


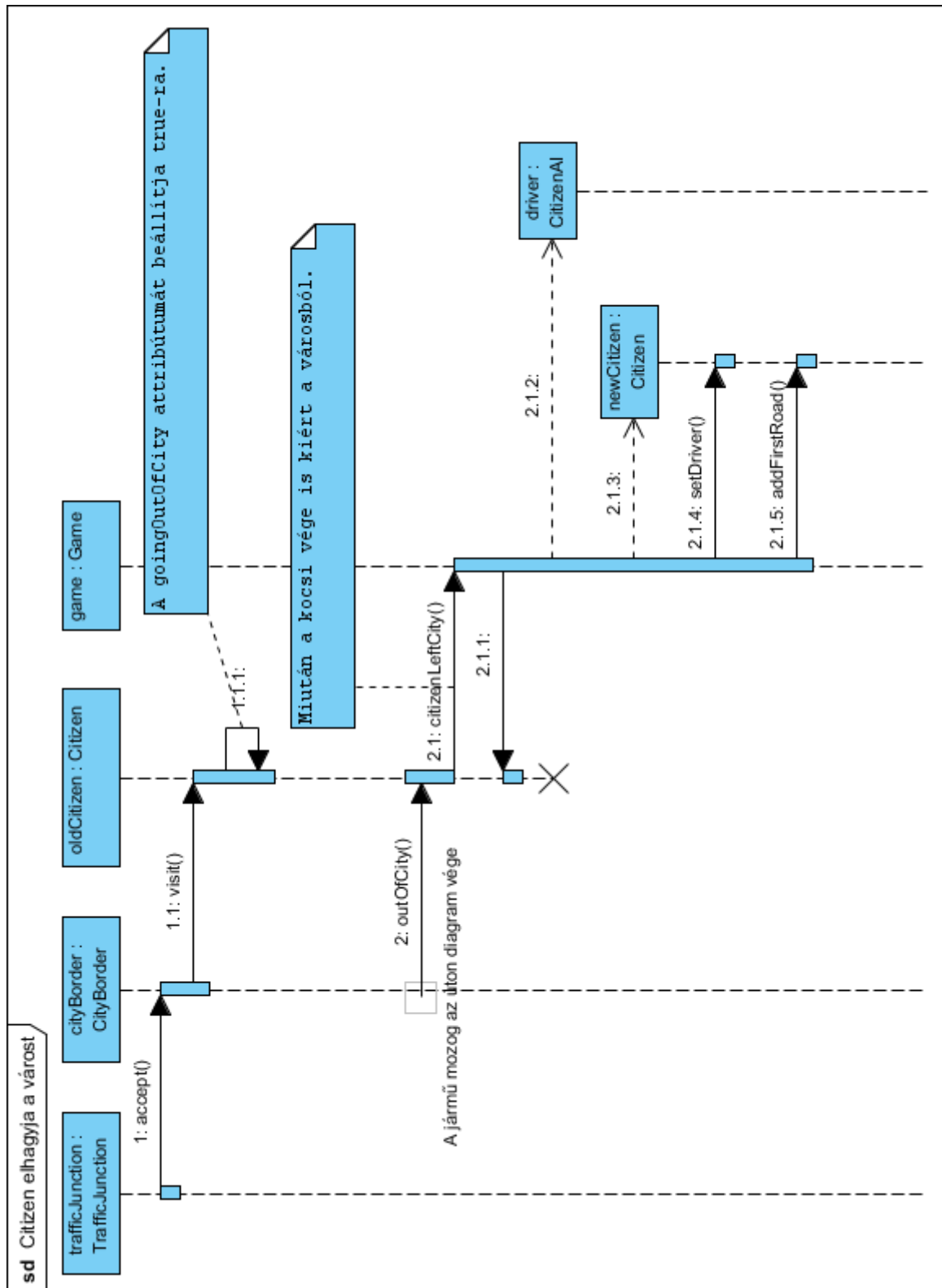




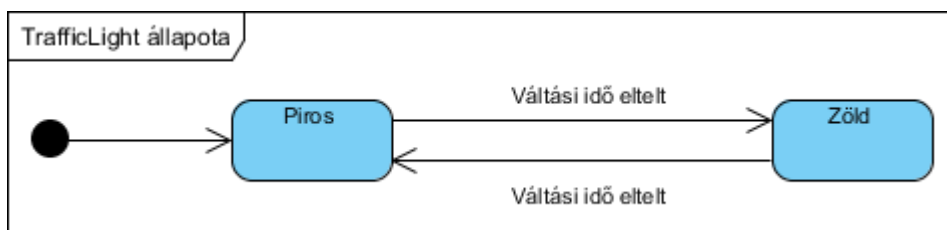
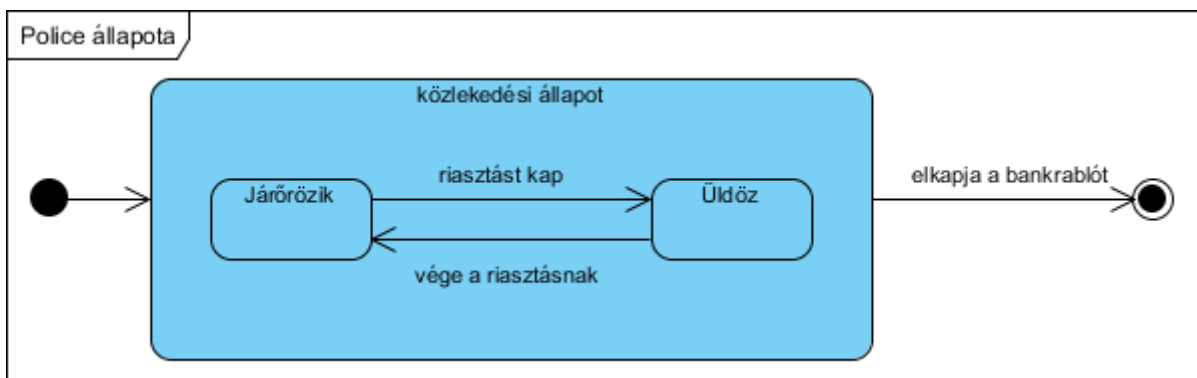
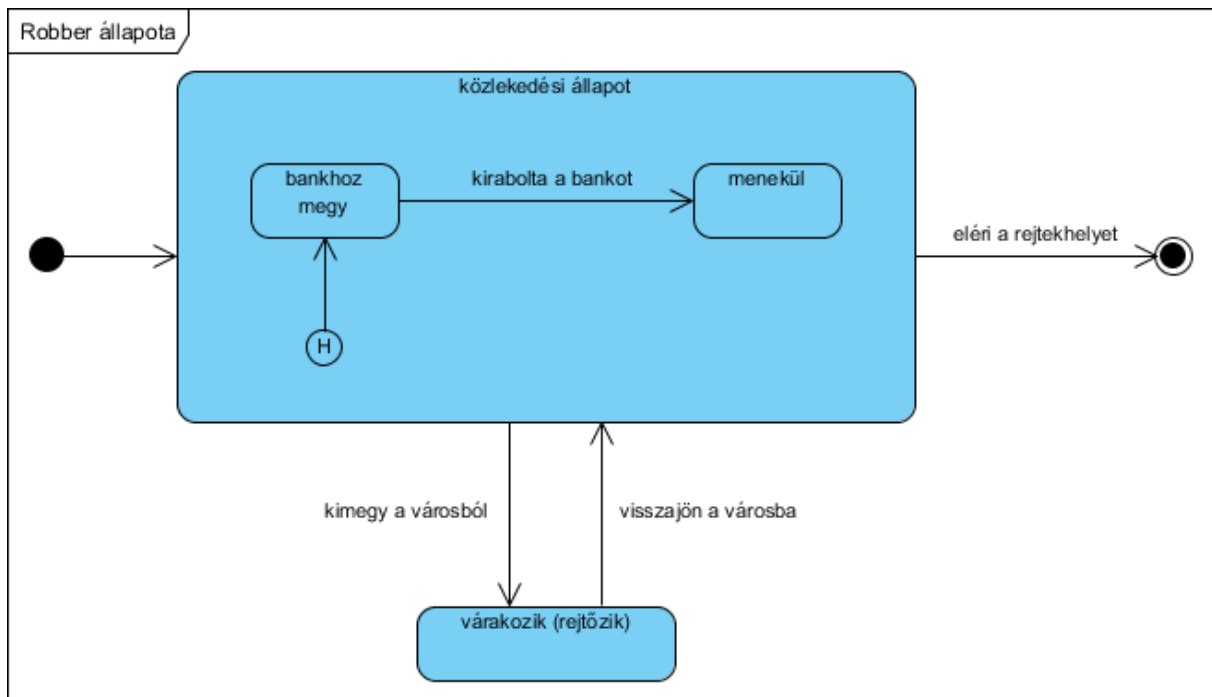








3.5 State-chartok



Napló

Kezdet	Időtartam	Résztevők	Leírás
2010.02.18. 19:00	3 óra	Angeli Koczka Kovács Mikó	Értekezlet. Döntés: Koczka és Kovács tervez, Angeli és Mikó dokumentumot szerkesztenek
2010.02.20. 15:00	5 óra	Koczka Kovács	Osztálydiagram fejlesztése
2010.02.20. 17:30	2 óra	Angeli Mikó	Objektumkatalógus megírása
2010.02.21. 11:00	5 óra 30 perc	Koczka Kovács	Osztálydiagram fejlesztése, Sequence diagramok készítése
2010.02.22. 20:30	2 óra	Kovács	Osztálydiagram fejlesztése, Sequence diagramok készítése
2010.02.23. 9:20	2 óra	Koczka	Osztálydiagram fejlesztése, Sequence diagramok készítése
2010.02.23. 18:00	4 óra	Angeli Mikó	Osztályok leírása fejezet írása, Objektumkatalógus befejezése
2010.02.23. 20:30	4 óra	Kovács	Osztálydiagram fejlesztése, Sequence diagramok készítése
2010.02.23. 23:00	3 óra	Koczka	Osztálydiagram fejlesztése, Sequence diagramok készítése
2010.02.24. 20:30	3 óra	Angeli Mikó	Osztályok leírása fejezet megírása
2010.02.24. 21:00	1 óra	Koczka	State Chartok készítése
2010.02.24. 23:00	1 óra	Mikó	Dokumentáció formázása

4. Analízis modell kidolgozása

4.1 Objektum katalógus

4.1.1 Map

Térkép mely tartalmazza, összefogja a térképen lévő különböző épületeket, utcákat, kereszteződések, táblákat, lámpákat.

4.1.2 TrafficJunction

Egy útkereszteződést testesít meg, ehhez kapcsolódnak Road-ok és más objektumok (Bank, Shelter). Az ő feladata, hogy az autókat egyik útról a másikra irányítsa.

4.1.3 Road

Egy utat testesít meg, amely egyirányú és egysávos, ezen mennek az autók. Több táblát tartalmazhat. Mozgatja a rajta levő kocsikat, megmondja egy kocsinak, hogy előtte milyen messze vannak különféle akadályok, illetve ha a kocsi ütközött.

4.1.4 StopSign

Stop tábla, akkor enged át, ha az előtte lévő autó megállt.

4.1.5 TrafficLight

Közlekedési lámpa, periodikusan átenged/nem enged át.

4.1.6 Vehicle

Egy autót testesít meg, Felelőssége többek között az ütközések kezelése, a kocsi paramétereinek megfelelő mozgás, valamint a saját sofőrével (driver-rel) való kommunikáció megvalósítása.

4.1.7 Robber

A bankrabló autója. Feladata a bank kirablásának majd ezután a rejtekhely elérésének illetve a városhatárból kérés és a rendőrrel való ütközés figyelése.

4.1.8 Police

A rendőr autója, MI irányítja (vagy egy másik játékos), üldözi a Robber-t. Rá nem vonatkoznak a közlekedési táblák szabályai, de nem ütközik a többi autóval, csak a Robberrel. Feladata, hogy elkapja a Robbert.

4.1.9 Citizen

Városi autó, ami nem rendőr és nem bankrabló autója. Nem megy neki az előtte lévőnek, lelassít, ha az előtte lévő lassít. Közlekedési szabályoknak megfelelően vezet.

4.1.10 Bank

A játékban az egyes épületek között megkülönböztetett szereppel bír a bank. Ha a rabló ehhez az épülethez érkezik, akkor értesíti az épületet, hogy itt tartózkodik, és az Bank elkezd lekezelti, hogy elinduljon a rabló elleni körözés, stb.

4.1.11 Shelter

A rejtekhely, mely szintén különleges funkciókat tölt be a játék folyamán. Ha egy

rabló látogatja meg az épületet, a Shelter megkérdi tőle, hogy ő rabolt-e már bankot és ha igen elkezdli lekezeli, hogy véget érjen a játék, a rabló nyer, stb.

4.1.12 CityBorder

A város határa, ha ezen túl megy a játékos, nem üldözik tovább a rendőrök. Persze, ha visszatér a városba, akkor újra üldözőbe veszik. Citizen esetén leveszi a térképről és új Citizen lerakását indítja.

4.2 Osztályok leírása

4.2.1 Bank

- **Felelősség**
 - Ha rabló a bankot tartalmazó mezőre lép akkor jelez a Game-nek hogy kirabolták a bankot, illetve ezt más jármű esetén is lekezeli.
- **Össztályok**
 - Building
- **Interfészek**
 - nincs
- **Attribútumok**
 - Game game - Egy referencia a Game-re.
- **Metódusok**
 - void visit(Robber robber) - Ha egy Robber érkezik az épülethez ez hívódik meg.
 - void visit(Citizen citizen) - Ha egy Citizen érkezik az épülethez ez hívódik meg.
 - void visit(Police police) - Ha egy Police érkezik az épülethez ez hívódik meg.

4.2.2 Building

- **Felelősség**
 - A Mapen elhelyezett kereszteződésekénél lévő épületek. (Shelter, Bank, CityBorder, stb.) Amikor egy jármű ehhez a kereszteződéshez ér, az épületek az ideérkező jármű típusától (Visitor Pattern) függően cselekednek.
- **Össztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - Game game - Egy referencia a Game-re.
- **Metódusok**
 - void visit(Robber robber) - Ha egy Robber érkezik az épülethez ez hívódik meg.
 - void visit(Citizen citizen) - Ha egy Citizen érkezik az épülethez ez hívódik meg.
 - void visit(Police police) - Ha egy Police érkezik az épülethez ez hívódik meg.

4.2.3 Citizen

- **Felelősség**
 - Városi autó, ami nem rendőr és nem bankrabló autója. Nem megy neki az előtte haladó járműnek, lelassít ha ő is lelassít, valamint betartja a közlekedési szabályokat.
 - Egy autót testesít meg. Felelőssége többek között az ütközések kezelése valamint a saját sofőrével (driver-rel) való kommunikáció megvalósítása.
- **Össztályok**
 - Vehicle
- **Interfészek**
 - nincs
- **Attribútumok**
 - boolean goingOutOfCity - Akkor igaz, ha a kocszi eleje már belement egy CityBorderes kereszteződésbe, de a kocszi vége még nem ért ki a városból. Még rajta van valamelyik Road-on.
 - Game game – Egy referencia a Game-re.
- **Metódusok**
 - Road getFrontRoad() - Visszaadja azt a Road-t, amelyiken az autó eleje van.
 - Road getDirection(Road[] roads) - Kiválasztjuk a továbbhaladási irányt.
 - void addFrontRoad(Road road) - Hozzáadja a Road[] listához azt az utat, amire az autó eleje rátért.
 - void removeLastRoad() - Leveszi a Road[] listáról legutolsó utat, amit az autó hátulja éppen elhagy.
 - double getVelocity() - Visszaadja a jármű pillanatnyi sebességét.
 - void crash(Robber robber) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
 - void crash(Citizen citizen) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
 - void crash(Police police) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
 - void acceptCrash(Vehicle vehicle) - Meghívódik, ha a járműbe belementek.
 - void outOfCity() - Meghívódik, ha a jármű letért az összes Roadról (vagyis kiment a városból).
 - VehicleType getType() – Visszaadja az autóra jellemző adatokat tartalmazó objektumot
 - double getMaxVelocity() – Visszaadja az autóra jellemző maximális sebességet
 - double getMaxAcceleration() – Visszaadja az autóra jellemző maximális gyorsulást
 - void setDriver(IDriver driver) – Ezzel lehet kiválasztani az autót vezető AI-t vagy emberi játékost.
 - void cityBorderReacher() – Ez hívódik meg, ha elérünk egy CityBorderes TrafficJunctiont
 - void doTick(int time, int deltatime) – Ezen fv. hatására mozog a jármű az úton. Lekéri a gyorsulást, megnézi az előtte lévő akadályokat stb.
 - void acceptBuilding(Building building) – A trafficjunction hívja meg a jármű ezen függvényét, hogy utána meglátogassa (visit) az ott lévő épületet, és lekezelje majd az épület, hogy mi történjen. Tudomásul adja az adott épületnek látogató jármű típusát (Visitor Pattern).

4.2.4 CitizenAI

- **Felelősség**
 - Ez az osztály azért felelős, hogy a városi autókat vezesse, illetve a rendőr autókat, ha éppen deaktiváltak.
- **Ősosztályok**
 - DriverAdapter
- **Interfészek**
 - nincs
- **Attribútumok**
 - Vehicle vehicle - Az irányított városi autó.
- **Metódusok**
 - double getAcceleration() - Visszaadja a Vehicle gyorsulását az akadályokat figyelembe véve.
 - Road chooseDirection(Road [] roads) – Választ egy utat hogy merre közlekedik tovább a jármű.
 - boolean visit(TrafficLight trafficLight) – Megmondja hogy az adott lámpánál mit tesz a vezető.
 - boolean visit(StopSign stopSign) – Megmondja hogy az adott stoptáblánál mit tesz a vezető.

4.2.5 CityBorder

- **Felelősség**
 - A városhatár. Itt tudnak a Citizen-ek kilépni/belépni a városba, illetve itt tud a bankrabló eltűnni ideiglenesen a rendőrök elől. Ezeket az eseményeket kezeli.
- **Ősosztályok**
 - Building
- **Interfészek**
 - nincs
- **Attribútumok**
 - Game game - Egy referencia a Game-re.
- **Metódusok**
 - void visit(Robber robber) - Ha egy Robber érkezik a CityBorderhez ez hívódik meg.
 - void visit(Citizen citizen) - Ha egy Citizen érkezik a CityBorderhez ez hívódik meg.
 - void visit(Police police) - Ha egy Police érkezik a CityBorderhez ez hívódik meg.

4.2.6 Game

- **Felelősség**
 - Ez az osztály hozza létre és semmisíti meg a térképet és a kocsikat, meghatározza a kocsik viselkedését.
 - Ez az osztály "irányítja" a játék mérőföldköveit
- **Ősosztályok**
 - nincs
- **Interfészek**

- ITick
- **Attribútumok**
 - Map map - A térkép.
 - Timer timer - A különböző objektumok időzítéséért felelős.
 - Robber[0..*] robbers - A játékban résztvevő rabló(k).
 - Police[0..*] polices - A játékban résztvevő rendőrök.
 - Citizen[0..*] citizens - A játékban résztvevő városi autók.
- **Metódusok**
 - void bankRobbed() - Akkor hívódik meg, amikor egy rabló kirabolja az első bankot.
 - void citizenLeftCity(Citizen citizen) - A paraméterül kapott citizen jármű városelhagyásakor hívódik meg, megsemmisíti a citizen-t és újat hoz létre.
 - void robberCaught(Robber robber) - Ha egy rendőr elér egy rablót, ez a függvény hívódik meg. A játék sikertelen végét jelenti.
 - void robberReachedShelter(Robber robber) - Ha a rabló eléri a búvóhelyet, ez a függvény hívódik meg. A játék sikeres végét jelenti.
 - void robberDied(Robber robber) - Ez a függvény kezeli le, ha a rablónak elfogynak az élete és úgy ér véget sikertelenül a játék.
 - void robberDisappeared(Robber robber) - Ez a függvény kezeli le, ha a rabló "kimenekül" a városhatáron kívülre.
 - void doTick(int time, int deltaTime) - Meghívódik, ha éppen egy időegységnyivel később van. Folyamatosan hívódik a Timer osztályon keresztül. Ez "lépteti" a játékot.

4.2.7 IDriver

- **Felelősség**
 - A járművek vezetése ezen az interfacen keresztül történik.
- **Ősosztályok**
 - ITrafficRuleHandler (ős-interface)
- **Interfészek**
 - nincs
- **Attribútumok**
- **Metódusok**
 - Road chooseDirection(Road[] roads) - Visszaadja, hogy merre halad tovább a Vehicle a kereszteződéseknél.
 - double getAcceleration() - Visszaadja a Vehicle gyorsulását az akadályokat figyelembe véve.
 - boolean visit(TrafficLight trafficLight) – Megmondja hogy az adott lámpánál mit tesz a vezető.
 - boolean visit(StopSign stopSign) – Megmondja hogy az adott stoptáblánál mit tesz a vezető.

4.2.8 ITick

- **Felelősség**
 - Az egész játék időzítése ezen az interfacen keresztül történik. Lámpakezelés, autó mozgatása, játék vezérlése.
- **Ősosztályok**
 - nincs
- **Interfészek**
 - nincs

- **Attribútumok**
 - nincs
- **Metódusok**
 - void doTick(int time, int deltatime) - Meghívódik, ha éppen egy időegységnivel később van. Folyamatosan hívódik a Timer osztályon keresztül. Ez "lépteti" a játékot.

4.2.9 ITrafficRuleHandler

- **Felelősség**
 - A járművek a közlekedési objektumokat (lámpa, tábla, stb.) ezen az interfacen keresztül kezelik.
- **Ősosztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - nincs
- **Metódusok**
 - boolean visit(StopSign stopSign) – Meghívódik egy adott jármű Stop táblához való érkezése esetén.
 - boolean visit(TrafficLight trafficLight) – Meghívódik egy adott jármű Lámpához való érkezése esetén.

4.2.10 Map

- **Felelősség**
 - Térkép mely tartalmazza, összefogja a térképen lévő különböző épületeket, utcákat, kereszteződések, táblákat, lámpákat.
- **Ősosztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - TrafficJunction[0..*] trafficJunctions - A kereszteződések tartalmazó tömb.
 - Road[0..*] roads - Az utakat tartalmazó tömb.
 - TrafficSign[0..*] trafficsigns - A táblákat tartalmazó tömb.
 - Building[0..*] buildings - Az épületeket tartalmazó tömb (amikkel interakcióba léphet valamelyik jármű).
- **Metódusok**
 - Buliding[] getBulidings() - Lekéri az épületeket tartalmazó tömböt.
 - TrafficJunction[] getTrafficJunctions - Lekéri a kereszteződések tartalmazó tömböt.
 - Road[] getRoads() - Lekéri az utakat tartalmazó tömböt.
 - TrafficSign[] getTrafficSigns() - Lekéri a táblákat tartalmazó tömböt.

4.2.11 Police

- **Felelősség**
 - A rendőr autója, MI irányítja, üldözi a Robber-t.

- Egy autót testesít meg. Felelőssége többek között az ütközések kezelése valamint a saját sofőrével (driver-rel) való kommunikáció megvalósítása.
- **Összaltályok**
 - Vehicle
- **Interfészek**
 - nincs
- **Attribútumok**
 - IDriver defaultDriver - A rendőrautó alapértelmezett vezetője (irányítója). Legtöbbször a PoliceAI, de ugyanakkor lehetőséget biztosít arra hogy KeyboardPlayer-ként más játékos is irányítsa a rendőrt.
 - boolean active - A rendőr állapotát jelző attribútum. (true - a rendőr üldözi a rablót; false - a rendőr nem üldözi a rablót).
 - boolean goingOutOfCity - Akkor igaz, ha a kocsi eleje már belement egy CityBorderes kereszteződésbe, de a kocsi vége még nem ért ki a városból. Még rajta van valamelyik Road-on.
 - Game game – Egy referencia a Game objektumra.
- **Metódusok**
 - void activate () - Az adott rendőr aktiválásáért felelős függvény. (aktív -> defaultdriver veszi át az irányítást).
 - void deactivate() - Az adott rendőr deaktiválásáért felelős függvény. (deaktív -> citizen-ként viselkedik).
 - boolean getActivate() - A függvény segítségével lekérdezhető az adott rendőr állapota (aktivitása).
 - void setDefaultDriver (IDriver : defaultDriver) - A rendőrautó alapértelmezett vezetőjének (irányítójának) beállításáért felelős függvény.
 - Road getFrontRoad() - Visszaadja azt a Road-t, amelyiken az autó eleje van.
 - Road getDirection(Road[] roads) - Kiválasztjuk a továbbhaladási irányt.
 - void addFrontRoad(Road road) - Hozzáadja a Road[] listához azt az utat, amire az autó eleje rátért.
 - void removeLastRoad() - Leveszi a Road[] listáról legutolsó utat, amit az autó hátulja éppen elhagy.
 - double getVelocity() - Visszaadja a jármű pillanatnyi sebességét.
 - void crash(Robber robber) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
 - void crash(Citizen citizen) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
 - void crash(Police police) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
 - void acceptCrash(Vehicle vehicle) - Meghívódik, ha a járműbe belementek.
 - void outOfCity() - Meghívódik, ha a jármű letért az összes Roadról (vagyis kiment a városból).
 - VehicleType getType() – Visszaadja az autóra jellemző adatokat tartalmazó objektumot
 - double getMaxVelocity() – Visszaadja az autóra jellemző maximális sebességet
 - double getMaxAcceleration() – Visszaadja az autóra jellemző maximális gyorsulást
 - void setDriver(IDriver driver) – Ezzel lehet kiválasztani az autót vezető AI-t vagy emberi játékost.

- void cityBorderReacher() – Ez hívódik meg, ha elérünk egy CityBorderes TrafficJunctiont
- void doTick(int time, int deltatime) – Ezen fv. hatására mozog a jármű az úton. Lekéri a gyorsulást, megnézi az előtte lévő akadályokat stb.
- void acceptBuilding(Building building) – A trafficjunction hívja meg a jármű ezen függvényét, hogy utána meglátogassa (visit) az ott lévő épületet, és lekezelje majd az épület, hogy mi történjen. Tudomásul adja az adott épületnek látogató jármű típusát (Visitor Pattern).
-

4.2.12 PoliceAI

- **Felelősség**
 - Ez az osztály azért felelős, hogy a rendőrt vezérelje, és az elkapja a rablót.
- **Ősosztályok**
 - DriverAdapter
- **Interfészek**
 - nincs
- **Attribútumok**
 - Vehicle vehicle - Az irányított városi autó.
- **Metódusok**
 - double getAcceleration() - Visszaadja a Vehicle gyorsulását az akadályokat figyelembe véve.
 - Road chooseDirection(Road [] roads) – Választ egy utat hogy merre közlekedik tovább a jármű.
 - boolean visit(TrafficLight trafficLight) – Megmondja hogy az adott lámpánál mit tesz a vezető.
 - boolean visit(StopSign stopSign) – Megmondja hogy az adott stoptáblánál mit tesz a vezető.

4.2.13 Road

- **Felelősség**
 - Egy utat testesít meg, amely egyirányú és egysávos, melyen a járművek közlekednek. Tartalmazhat különböző közlekedést szabályozó eszközöket (táblák, lámpa).
- **Ősosztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - VehicleOnRoad[0..*] vehicles - Az úton tartózkodó (közlekedő) járműveket tartalmazó tömb.
 - TrafficJunction[2] trafficJunctions - Az út két végén található keresztezések tartalmazó tömb. Az elsőből a másodikba megy az út.
 - TrafficSign[0..*] trafficSigns - Az úton található különböző közlekedést szabályozó eszközök tömbje (lámpák, táblák). Mindegyik az út végén található (egyirányúság).
- **Metódusok**
 - void addTrafficSign(TrafficSign trafficSign) – A függvény az útra helyez különböző TrafficSignokat (lámpa, tábla)
 - double getLength() - Kiszámolja és visszaadja az út hosszát.

- boolean isOccupied(TrafficJunction trafficJunction) - Az út azon részének foglaltságát jelzi ami még belelóg a kereszteződésbe.
- void addVehicle(Vehicle vehicle) - A függvény az útra helyez egy járművet.
- void removeVehicle(Vehicle vehicle) - A függvény levesz az útról egy járművet.
- double getNearestPhysicalBarrierDistance(Vehicle vehicle) - A függvény a legközelebbi fizikai akadálytól való távolságot (az út irányában) adja vissza (fizikai: olyan dolog amivel ütközni lehet pl.: jármű)
- TrafficSign[] getTrafficSigns(Vehicle vehicle, double distance) - A függvény visszatér a paraméterként kapott járműtől maximum distance távolságra lévő akadályokat)
- Vehicle getNearestCar(Vehicle vehicle) - Egy autótól az adott úton lévő legközelebbi jármű lekérdezésére szolgáló metódus (az út irányában).
- TrafficJunction[] getTrafficJunctions() - A függvény visszatérési értékül adja az adott úthoz tartozó közlekedési csomópontokat.
- void moveVehicle(Vehicle vehicle, double distance) - A függvény az adott járművet a megadott távolságban mozgatja az úton.
- boolean isVehicleIncoming(double time) – Igaz, ha adott időn belül a Road végén lévő TrafficJunction-be érkezik jármű.

4.2.14 Robber

- **Felelősség**
 - A bankrabló autója, a játékos irányítja.
 - Egy autót testesít meg. Felelőssége többek között az ütközések kezelése valamint a saját sofőrével (driver-rel) való kommunikáció megvalósítása.
- **Össztályok**
 - Vehicle
- **Interfészek**
 - nincs
- **Attribútumok**
 - int lives - A játékos életeinek száma.
 - boolean wanted - A játékost körözik-e a rendőrök. A bankrablásnál vált true-ra.
 - boolean goingOutOfCity - Akkor igaz, ha a kocsí eleje már belement egy CityBorderes kereszteződésbe, de a kocsí vége még nem ért ki a városból. Még rajta van valamelyik Road-on.
 - Game game – Egy referencia a Game objektumra.
 - TrafficJunction cityBorderTrafficJunction - Az a kereszteződés, amelyiknél kilépett a rabló a városból.
- **Metódusok**
 - void goBackToCity() - Ez a függvény hívódik meg akkor, ha a rabló visszatér a városba.
 - boolean getWanted() – Ez adja vissza hogy ezt a rablót körözik-e már!
 - void setWanted(boolean wanted) – Beállíthatjuk, hogy az adott rablót körözzék.
 - void setCityBorderTrafficJunction() – Beállíthatjuk, a rabló cityBorderTrafficJunction attribútumát, hogy hol ment ki a városból.
 - Road getFrontRoad() - Visszaadja azt a Road-t, amelyiken az autó eleje van.

- Road getDirection(Road[] roads) - Kiválasztjuk a továbbhaladási irányt.
- void addFrontRoad(Road road) - Hozzáadja a Road[] listához azt az utat, amire az autó eleje rátért.
- void removeLastRoad() - Leveszi a Road[] listáról legutolsó utat, amit az autó hátulja éppen elhagy.
- double getVelocity() - Visszaadja a jármű pillanatnyi sebességét.
- void crash(Robber robber) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
- void crash(Citizen citizen) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
- void crash(Police police) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
- void acceptCrash(Vehicle vehicle) - Meghívódik, ha a járműbe belementek.
- void outOfCity() - Meghívódik, ha a jármű letért az összes Roadról (vagyis kiment a városból).
- VehicleType getType() – Visszaadja az autóra jellemző adatokat tartalmazó objektumot
- double getMaxVelocity() – Visszaadja az autóra jellemző maximális sebességet
- double getMaxAcceleration() – Visszaadja az autóra jellemző maximális gyorsulást
- void setDriver(IDriver driver) – Ezzel lehet kiválasztani az autót vezető AI-t vagy emberi játékost.
- void cityBorderReacher() – Ez hívódik meg, ha elérünk egy CityBorderes TrafficJunctiont
- void doTick(int time, int deltatime) – Ezen fv. hatására mozog a jármű az úton. Lekéri a gyorsulást, megnézi az előtte lévő akadályokat stb.
- void acceptBuilding(Building building) – A trafficjunction hívja meg a jármű ezen függvényét, hogy utána meglátogassa (visit) az ott lévő épületet, és lekezelje majd az épület, hogy mi történjen. Tudomásul adja az adott épületnek látogató jármű típusát (Visitor Pattern).

4.2.15 Shelter

- **Felelősség**
 - A rejtekhely, mely szintén különleges funkciókat tölt be a játék folyamán. Ha egy rabló látogatja meg az épületet, a Shelter megkérdi tőle, hogy ő rabolt-e már bankot és ha igen elkezd lekezelni, hogy véget érjen a játék, a rabló nyer, stb.
- **Összaltályok**
 - Building
- **Interfészek**
 - nincs
- **Attribútumok**
 - Game game - Egy referencia a Game-re.
- **Metódusok**
 - void visit(Robber robber) - Ha egy Robber érkezik a Shelterhez ez hívódik meg.

- void visit(Citizen citizen) - Ha egy Citizen érkezik a Shelterhez ez hívódik meg.
- void visit(Police police) - Ha egy Police érkezik a Shelterhez ez hívódik meg.

4.2.16 **StopSign**

- **Felelősség**
 - Stop tábla, akkor mehet át az autó rajta ha előtte megáll, értesíti az ideérkező járművet a létezéséről.
- **Össztályok**
 - TrafficSign
- **Interfészek**
 - nincs
- **Attribútumok**
 - nincs
- **Metódusok**
 - boolean accept(ITrafficRuleHandler trafficRuleHandler) – Szól az ideérkező jármű Driverének, hogy kezelje le, hogy stoptáblánál van.

4.2.17 **Timer**

- **Felelősség**
 - A játék órája. Az időzítésért felelős.
- **Össztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - ITick[] tickers - Azon objektumok, amelyek függnek az idő múlásától. (pl.: autók, lámpák, stb.) Ezeket kezeljük folyamatosan.
 - double deltaTime - Ez az a delta érték amennyi időnként újra Tick-eljük az objektumokat.
 - int time – A starttól eltelt idő.
 - Thread thread – A szál amit a timer használ.
- **Metódusok**
 - void start() - Ez indítja el az időt (egy szálal).
 - void stop() - Ez megállítja az időt.
 - void run() - Ez futtatja a szálal.
 - void setDeltaTime(double deltaTime) - Itt lehet beállítani a fent leírt időközöket.
 - void addTicker(ITick ticker) – Ezzel lehet hozzáadni egy időre érzékeny objektumot.

4.2.18 **TrafficJunction**

- **Felelősség**
 - Egy útkereszteződést testesít meg, ebbe vannak az Road-ok és más objektumok (Bank, Shelter).
- **Össztályok**

- nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - Road[0..*] inRoads - A kereszteződésbe bemenő utak tömbje.
 - Road[0..*] outRoads - A kereszteződésből kilépő utak tömbje.
 - Building[0..*] buildings - A kereszteződésben található épületek tömbje (amikkel interakcióba léphet valami).
 - double x - A kereszteződés x koordinátája.
 - double y - A kereszteződés y koordinátája.
- **Metódusok**
 - void addBuilding(Building building) – Ezzel lehet a kereszteződéshez hozzáadni buildinget (Shelter, Bank, CityBorder stb.)
 - void addInRoad(Road inRoad) – Ezzel lehet bevezető utat hozzáadni a kereszteződéshez.
 - void addOutRoad(Road outRoad) – Ezzel lehet kivezető utat hozzáadni a kereszteződéshez.
 - boolean isOccupied() - A kereszteződés foglaltságát jelzi (tartózkodik-e jármű a kereszteződésben).
 - void vehicleArrived(Vehicle vehicle) - Meghívódik amint érkezik egy autó a kereszteződésbe.
 - double getX() - Visszatérési értékül adja a kereszteződés x koordinátáját.
 - double getY() - Visszatérési értékül adja a kereszteződés y koordinátáját.
 - Road[] getInRoads() - A függvény a kereszteződésbe belépő utak tömbjét adja vissza.
 - Road[] getOutRoads() - A függvény a kereszteződésből kilépő utak tömbjét adja vissza.
 - void vehicleArrived(Vehicle vehicle) – Ez a függvény kezeli, ha egy jármű jut a kereszteződésbe.

4.2.19 TrafficLight

- **Felelősség**
 - Közlekedés irányító lámpa, periodikusan vált színt és értesíti az ideérkező járművet a létezéséről.
- **Ősosztályok**
 - TrafficSign
- **Interfészek**
 - ITick
- **Attribútumok**
 - int period – Az az idő intervallum, amennyi időnként vált a lámpa.
 - int phase – Fáziseltolás msecben.
 - boolean green – Akkor igaz, ha a lámpa éppen zöld.
- **Metódusok**
 - boolean accept(ITrafficRuleHandler trafficRuleHandler) – Szól az ideérkező jármű Driverének, hogy kezelje le, hogy lámpánál van.
 - boolean isGreen() – Lekérdezhető itt, hogy a lámpa éppen zöld-e.
 - void doTick(int time, int deltaTime) – Ezt a függvényt meghívva történik az idő múlása, lámpa váltása, ha éppen kell.

4.2.20 TrafficSign

- **Felelősség**
 - Közlekedési objektum, amiből öröklődnek a tábla, lámpa stb. osztályok.
 - Egy közlekedési szabályt fogalmaz meg.
- **Ősosztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
- **Metódusok**
 - abstract boolean accept(ITrafficRuleHandler trafficRuleHandler) – Szól az ideérkező jármű Driverének, hogy kezelje le, hogy valamilyen közlekedési objektumhoz ért.

4.2.21 Vehicle

- **Felelősség**
 - Egy autót testesít meg, Felelőssége többek között az ütközések kezelése valamint a saját sofőrével (driver-rel) való kommunikáció megvalósítása.
- **Ősosztályok**
 - nincs
- **Interfészek**
 - ITick
- **Attribútumok**
 - driver: IDriver - Az autó vezetője, ő adja meg, hogy milyen gyorsulással és merre menjen az autó.
 - double velocity - Az autó pillanatnyi sebessége.
 - Road[] roads - Azon utak listája, amelyeken az autó pillanatnyilag rajta van. (Hosszú jármű v. kanyarodó út esetén fontos.) FIFO típusú lista.
 - VehicleType type - Az autó "tulajdonságait" adja meg.
 - Game game – Egy referencia a Game objektumra
 - boolean goingOutOfCity - Akkor igaz, ha a kocsi eleje már belement egy cityborderes kereszteződésbe, de a kocsi vége még nem ért ki a városból. A jármű még rajta van valamelyik Road-on.
- **Metódusok**
 - Road getFrontRoad() - Visszaadja azt a Road-t, amelyiken az autó eleje van.
 - Road getDirection(Road[] roads) - Kiválasztjuk a továbbhaladási irányt.
 - void addFrontRoad(Road road) - Hozzáadja a Road[] listához azt az utat, amire az autó eleje rátért.
 - void removeLastRoad() - Leveszi a Road[] listáról legutolsó utat, amit az autó hátulja éppen elhagy.
 - double getVelocity() - Visszaadja a jármű pillanatnyi sebességét.
 - void crash(Robber robber) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
 - void crash(Citizen citizen) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
 - void crash(Police police) - Meghívódik, ha a jármű ütközik. A jármű belemegy egy másik járműbe. (Visitor pattern)
 - void acceptCrash(Vehicle vehicle) - Meghívódik, ha a járműbe belementek.

- void outOfCity() - Meghívódik, ha a jármű letért az összes Roadról (vagyis kiment a városból).
- VehicleType getType() – Visszaadja az autóra jellemző adatokat tartalmazó objektumot
- double getMaxVelocity() – Visszaadja az autóra jellemző maximális sebességet
- double getMaxAcceleration() – Visszaadja az autóra jellemző maximális gyorsulást
- void setDriver(IDriver driver) – Ezzel lehet kiválasztani az autót vezető AI-t vagy emberi játékost.
- void cityBorderReacher() – Ez hívódik meg, ha elérünk egy CityBorderes TrafficJunctiont
- void doTick(int time, int deltaTime) – Ezen fv. hatására mozog a jármű az úton. Lekéri a gyorsulást, megnézi az előtte lévő akadályokat stb.
- abstract void acceptBuilding(Building building) – A trafficjunction hívja meg a jármű ezen függvényét, hogy utána meglátoassa (visit) az ott lévő épületet, és lekezelje majd az épület, hogy mi történjen (visitor pattern) Tudomásul adja az adott épületnek látogató jármű típusát (Visitor Pattern).

4.2.22 VehicleOnRoad

- **Felelősség**
 - Egy járművet bedobozolunk. Ezen keresztül tárolja a Road osztály a járműveket. Itt tároljuk a jármű helyét az adott úton.
- **Össztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - vehicle Vehicle - A jármű, amit tartalmaz .
 - double position - A jármű pozíciója az adott úton.
- **Metódusok**
 - double getPosition() - visszaadja a jármű position attribútumát.
 - void setPosition(double position) - Beállíthatjuk a jármű pozícióját, például az autó mozgásánál az úton, ezt állítjuk.
 - Vehicle getVehicle() - Visszaadja a tárolt járművet.

4.2.23 VehicleType

- **Felelősség**
 - Egy járműre jellemző tulajdonságokat tároljuk ebben az osztályban.
- **Össztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - double length - A jármű hossza.
 - double maxVelocity - A jármű maximális sebessége.
 - double maxAcceleration - A jármű maximális gyorsulása.
- **Metódusok**

- double getLength() - Visszaadja a jármű hosszát.
- double getMaxVelocity() - Visszaadja a jármű maximális sebességét.

4.2.24 double getMaxAcceleration() - Visszaadja a jármű maximális gyorsulását VehicleType

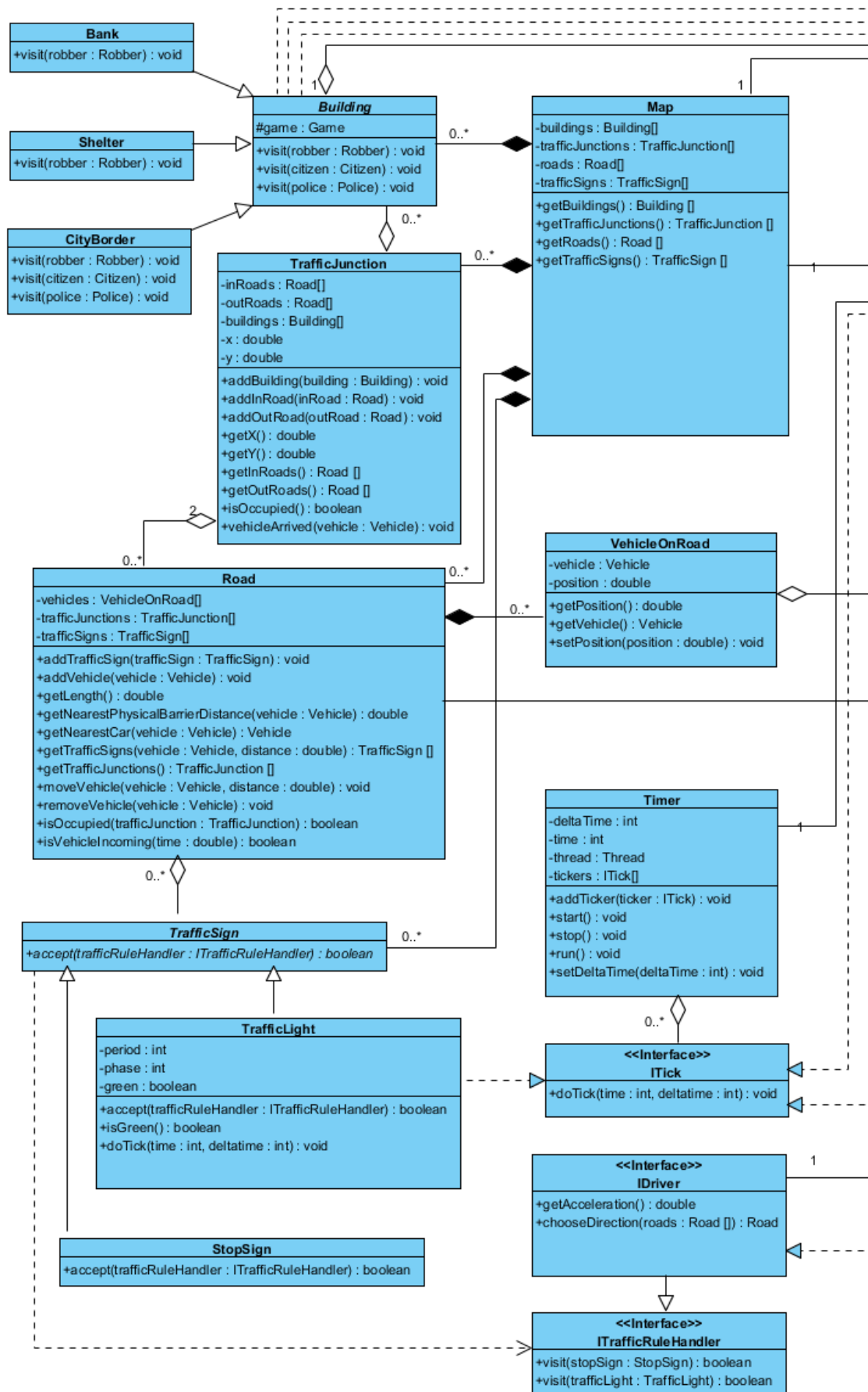
- **Felelősség**
 - Egy járműre jellemző tulajdonságokat tároljuk ebben az osztályban.
- **Össztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - double length - A jármű hossza.
 - double maxVelocity - A jármű maximális sebessége.
 - double maxAcceleration - A jármű maximális gyorsulása.
- **Metódusok**
 - double getLength() - Visszaadja a jármű hosszát.
 - double getMaxVelocity() - Visszaadja a jármű maximális sebességét.
 - double getMaxAcceleration() - Visszaadja a jármű maximális gyorsulását

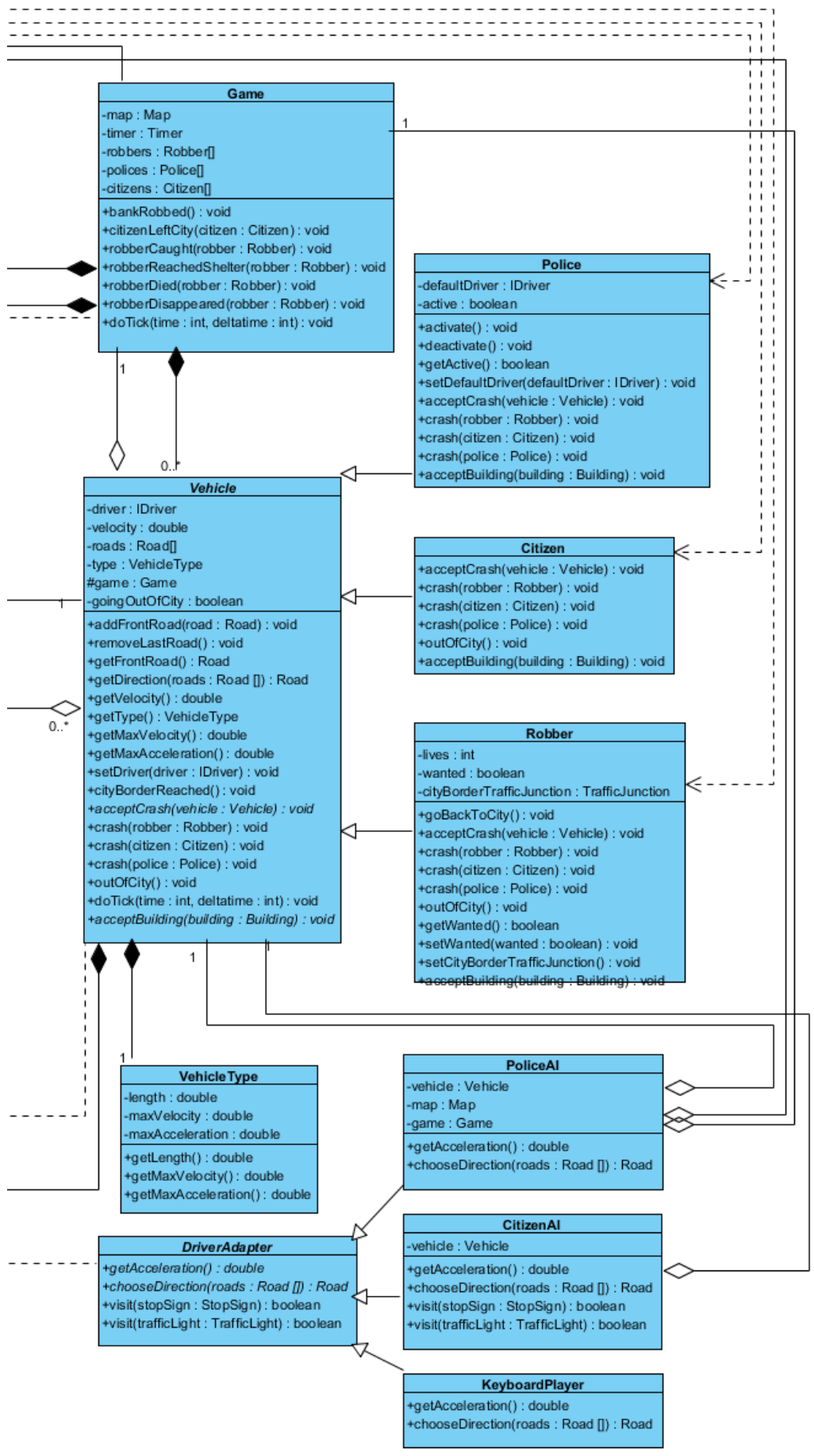
4.2.25 VehicleType

- **Felelősség**
 - Egy járműre jellemző tulajdonságokat tároljuk ebben az osztályban.
- **Össztályok**
 - nincs
- **Interfészek**
 - nincs
- **Attribútumok**
 - double length - A jármű hossza.
 - double maxVelocity - A jármű maximális sebessége.
 - double maxAcceleration - A jármű maximális gyorsulása.
- **Metódusok**
 - double getLength() - Visszaadja a jármű hosszát.
 - double getMaxVelocity() - Visszaadja a jármű maximális sebességét.
 - double getMaxAcceleration() - Visszaadja a jármű maximális gyorsulását

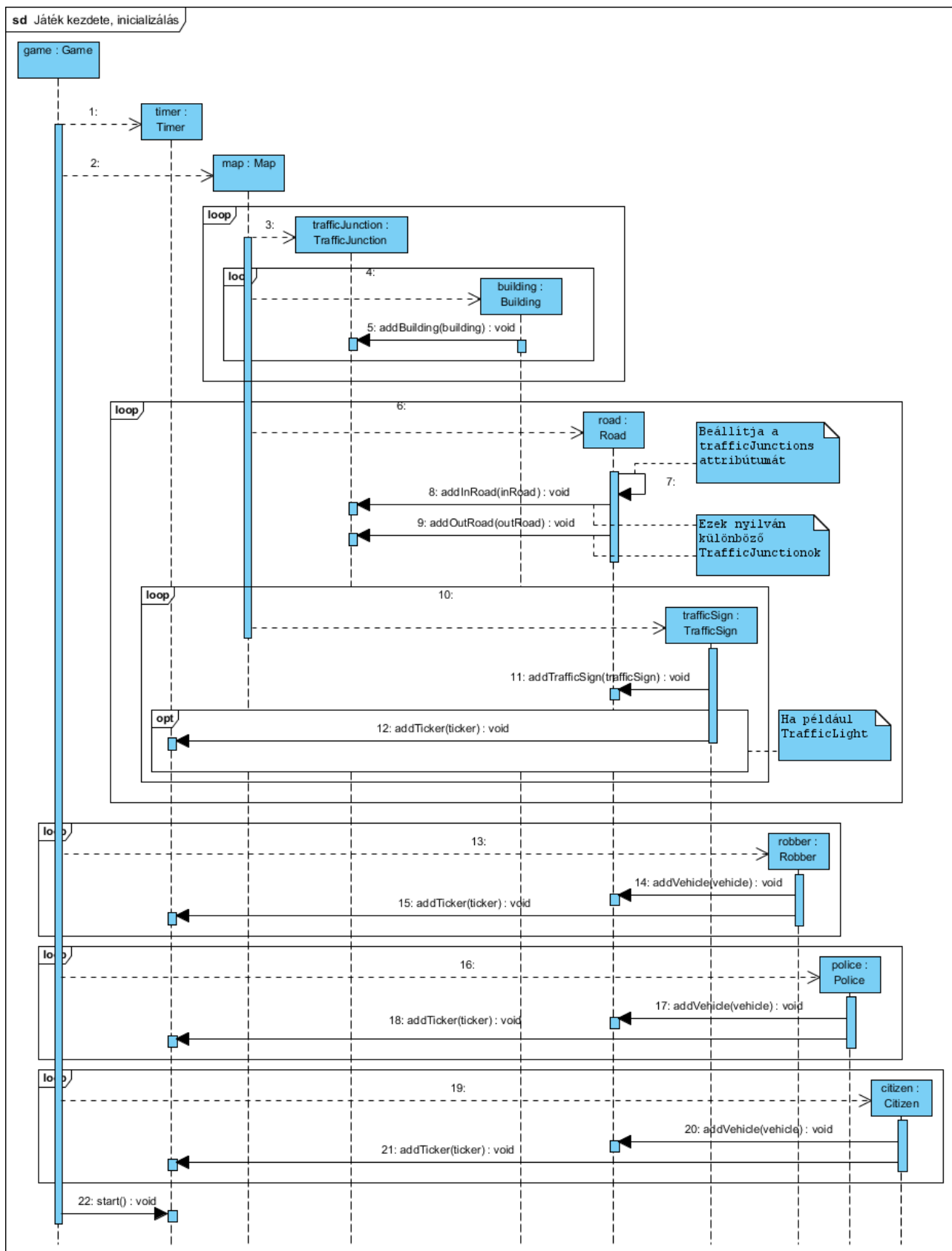
4.3 Statikus struktúra diagramok

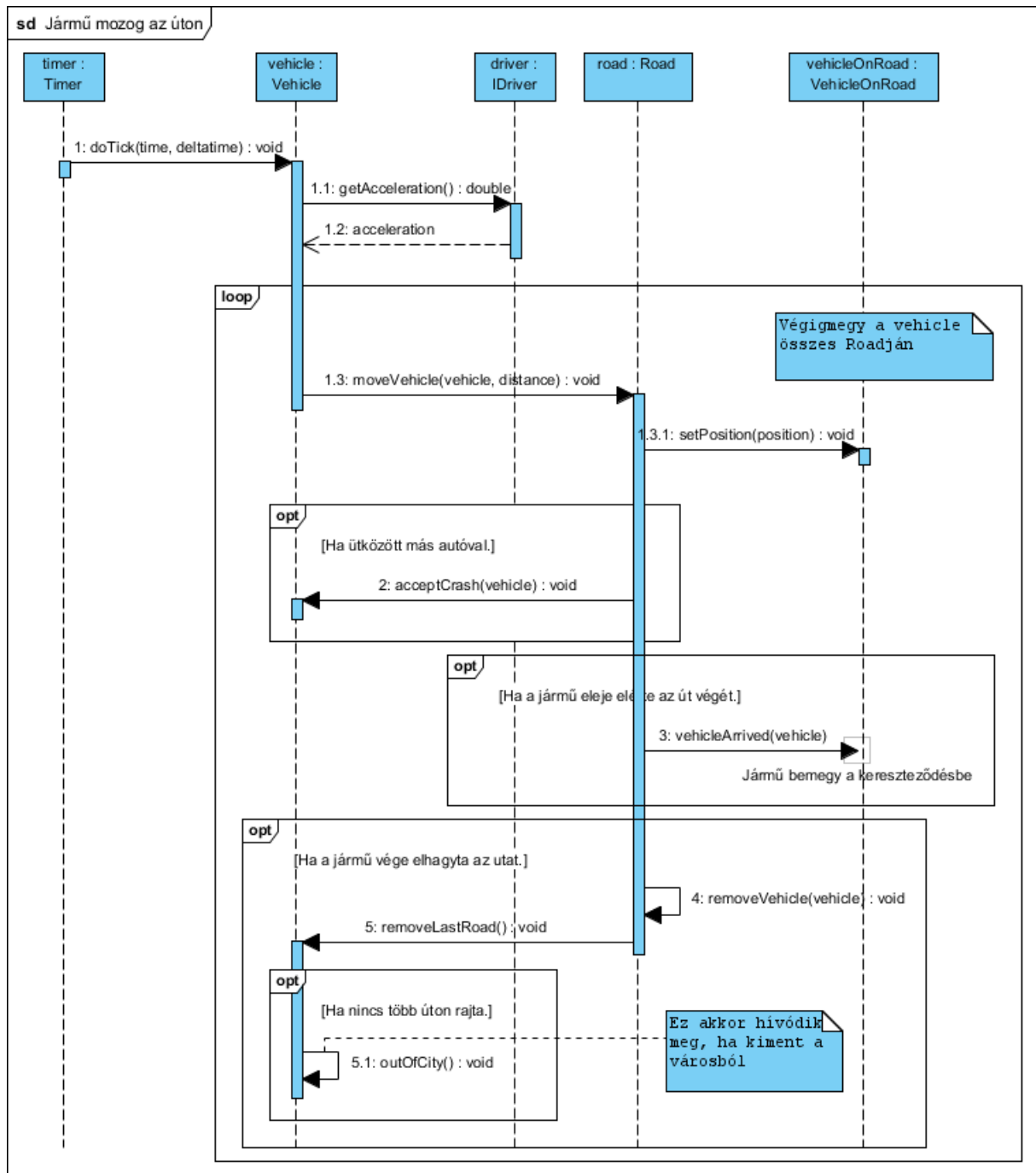
Osztálydiagram

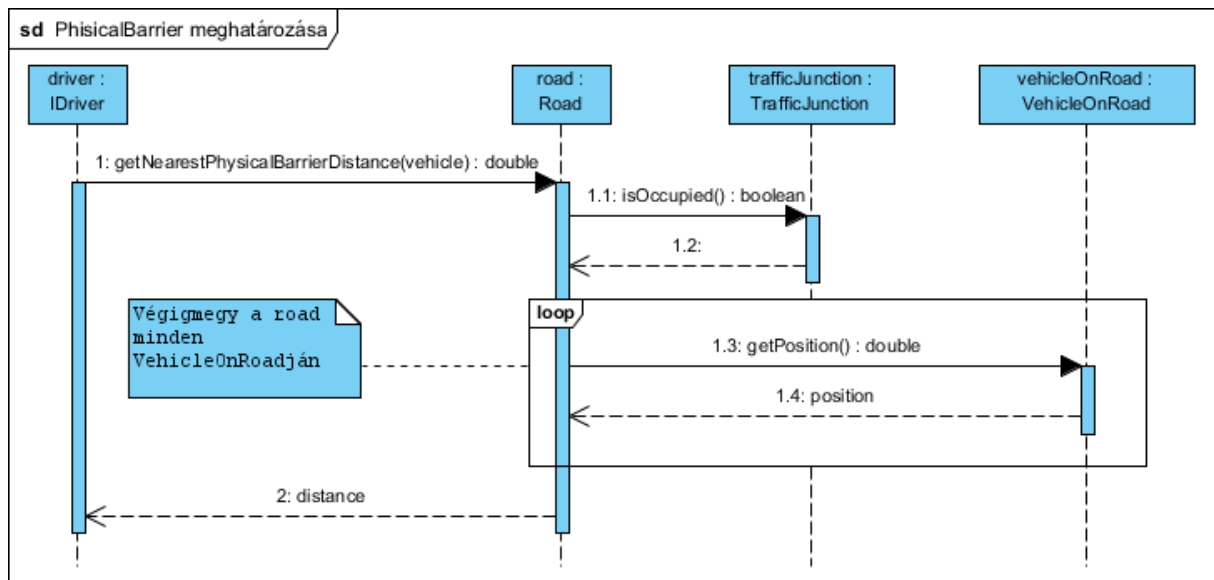
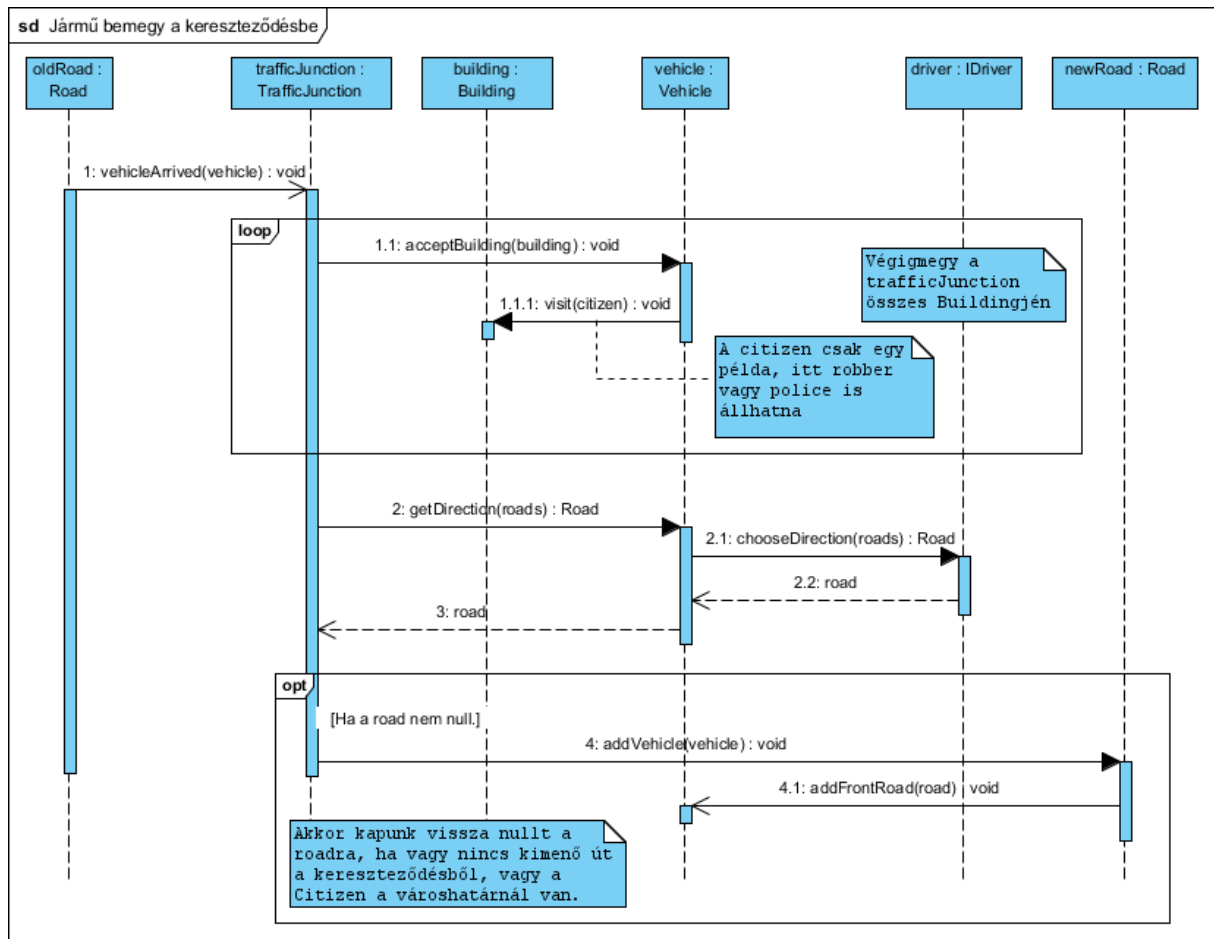


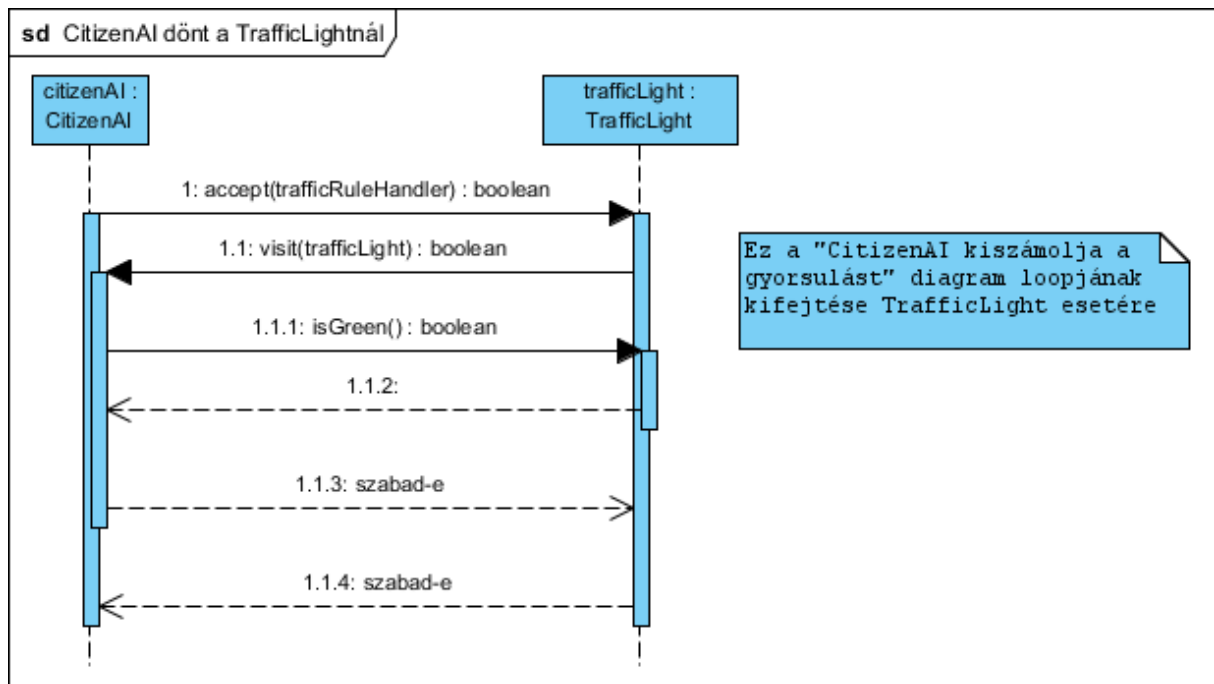
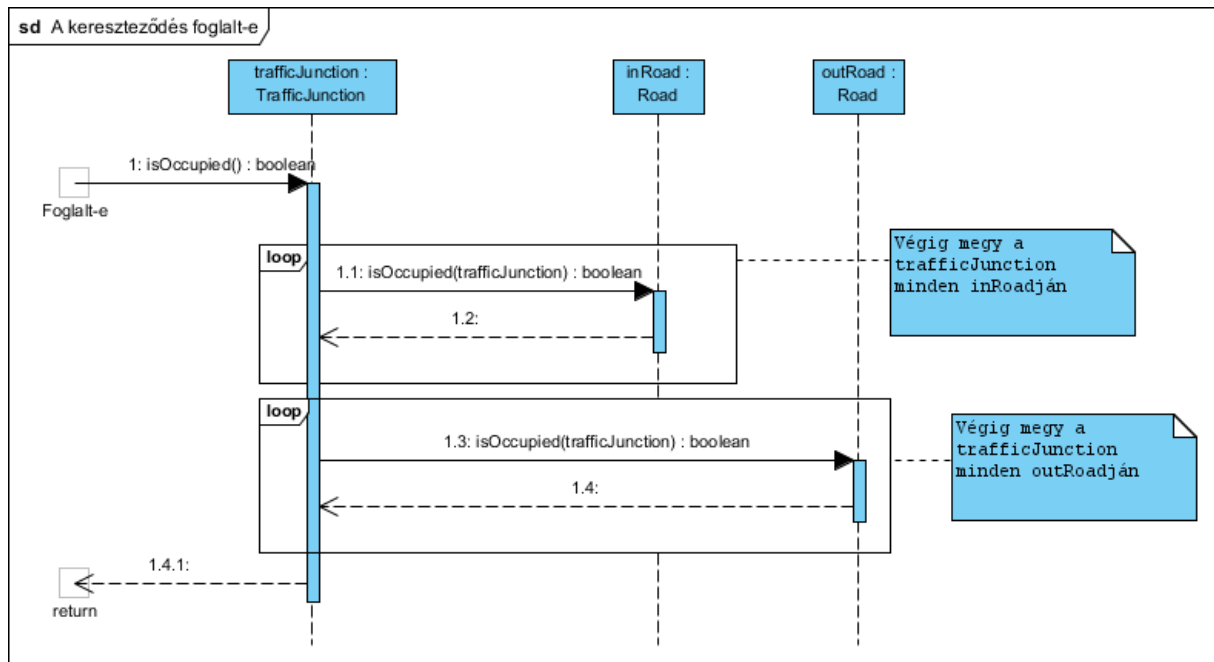


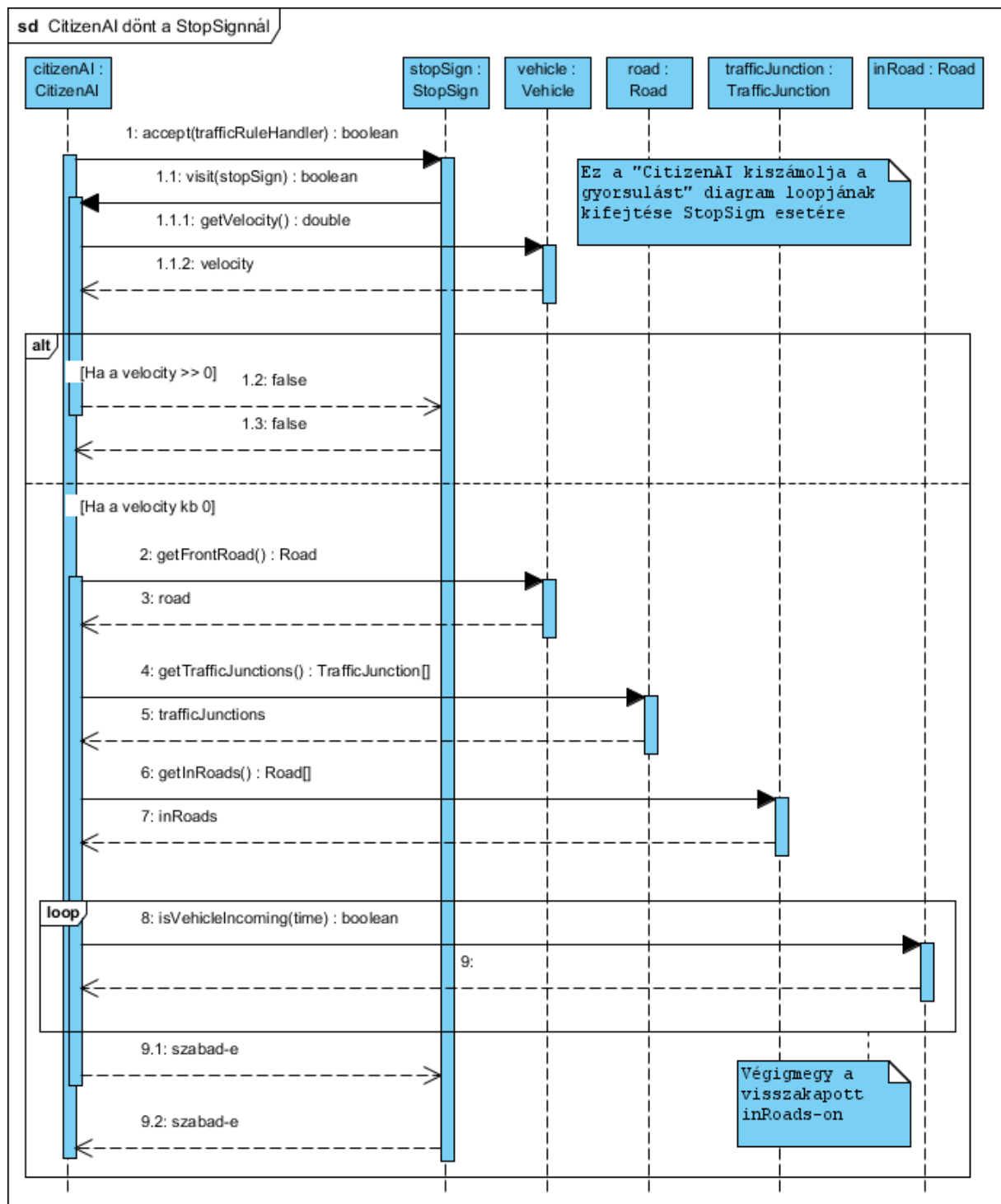
4.4 Szekvencia diagramok

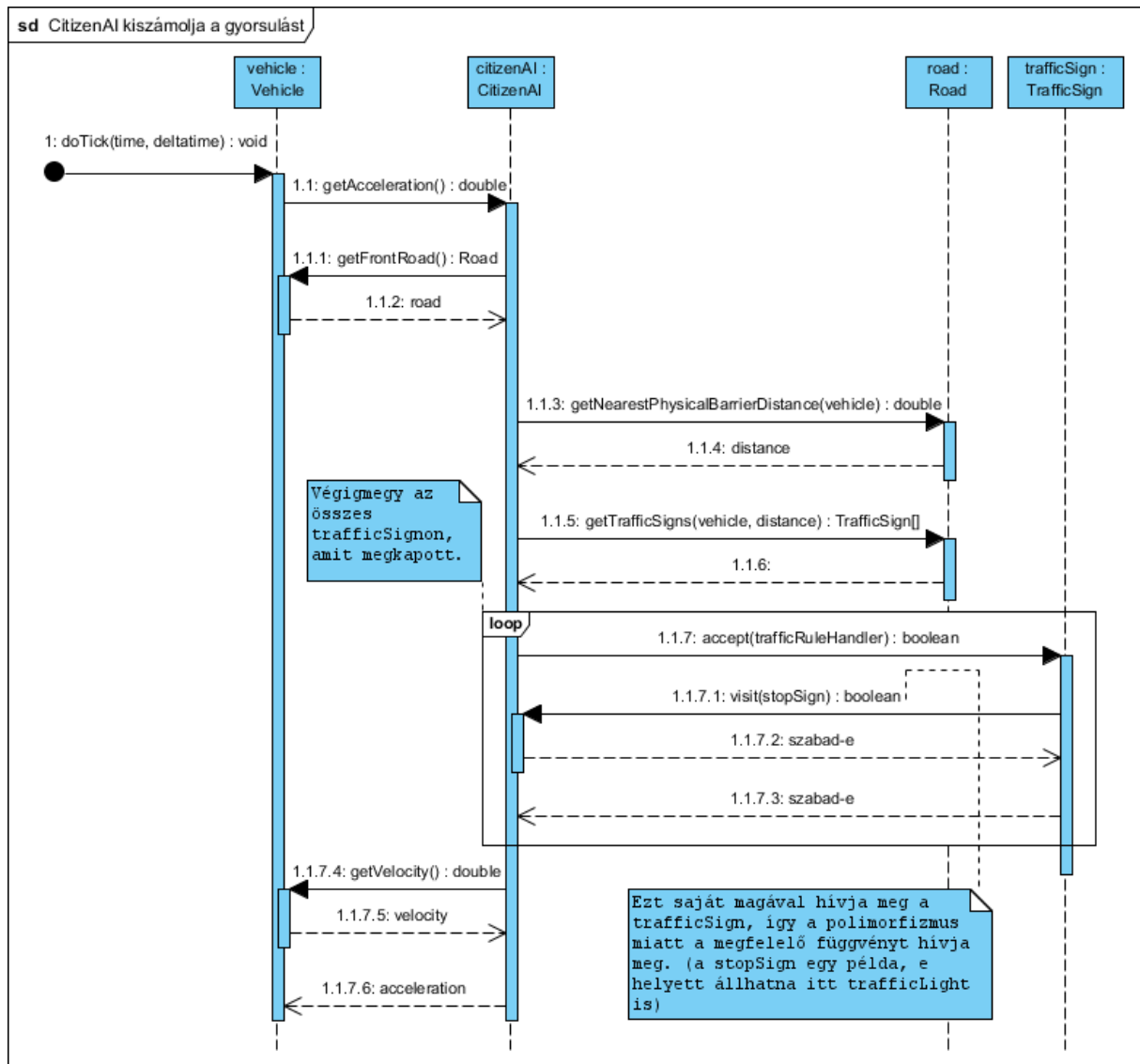


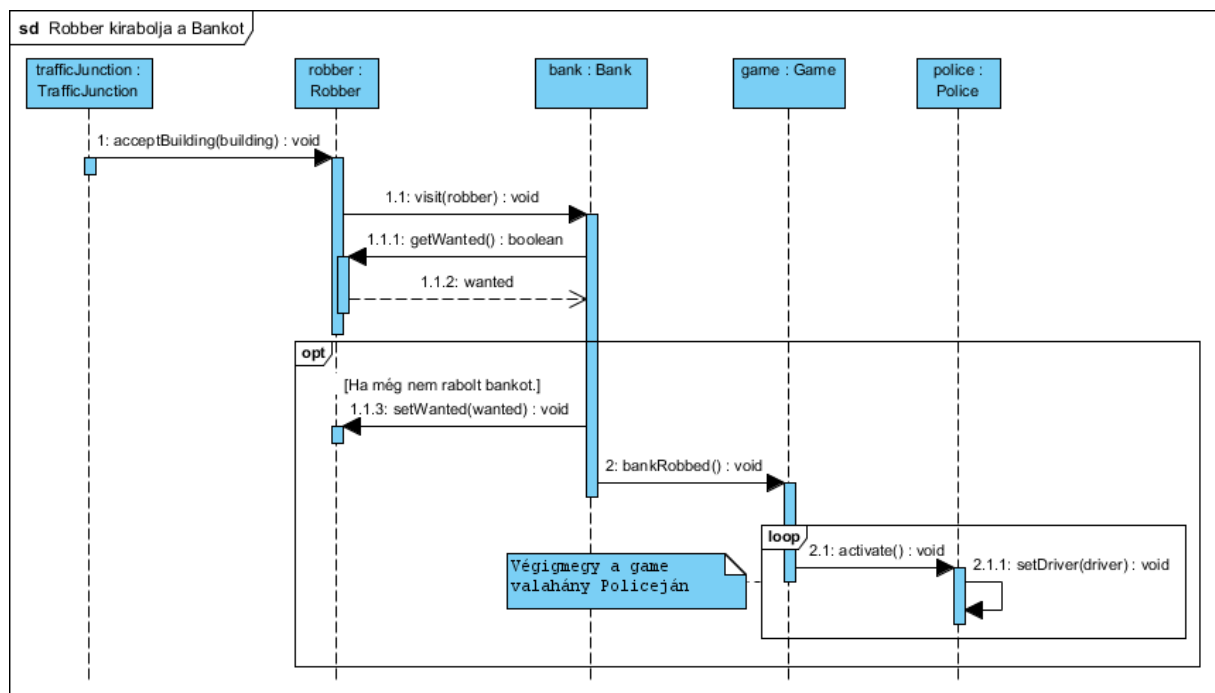
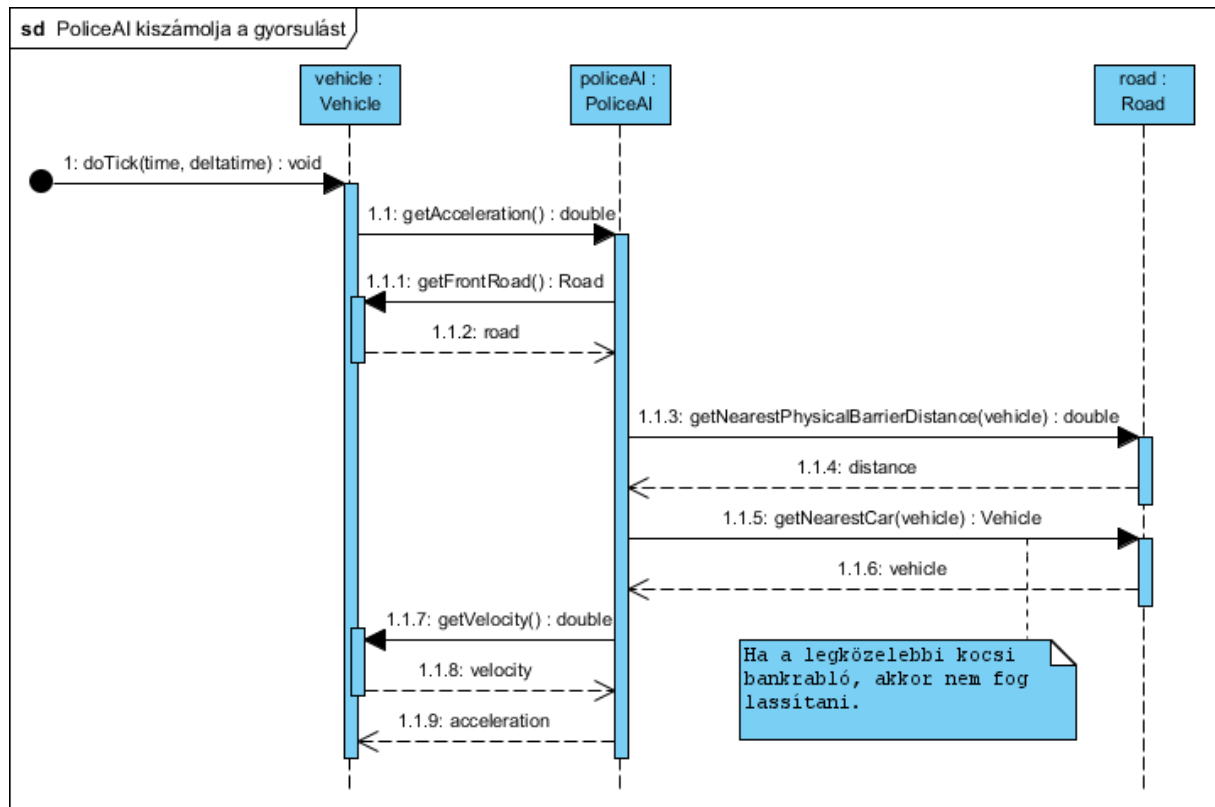


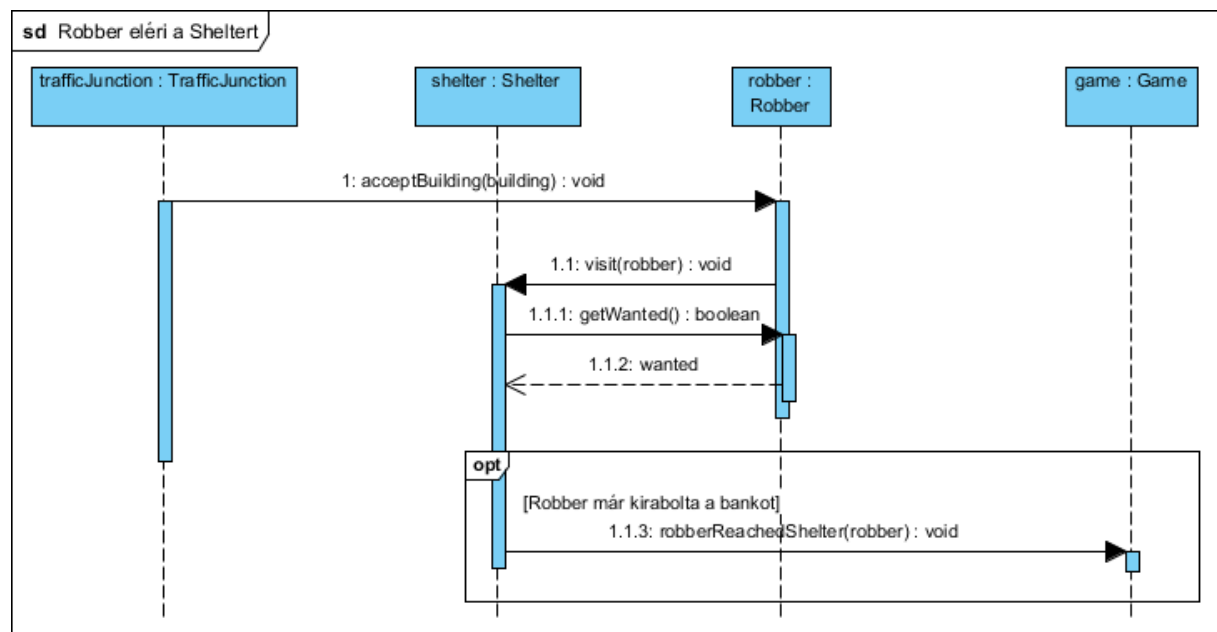
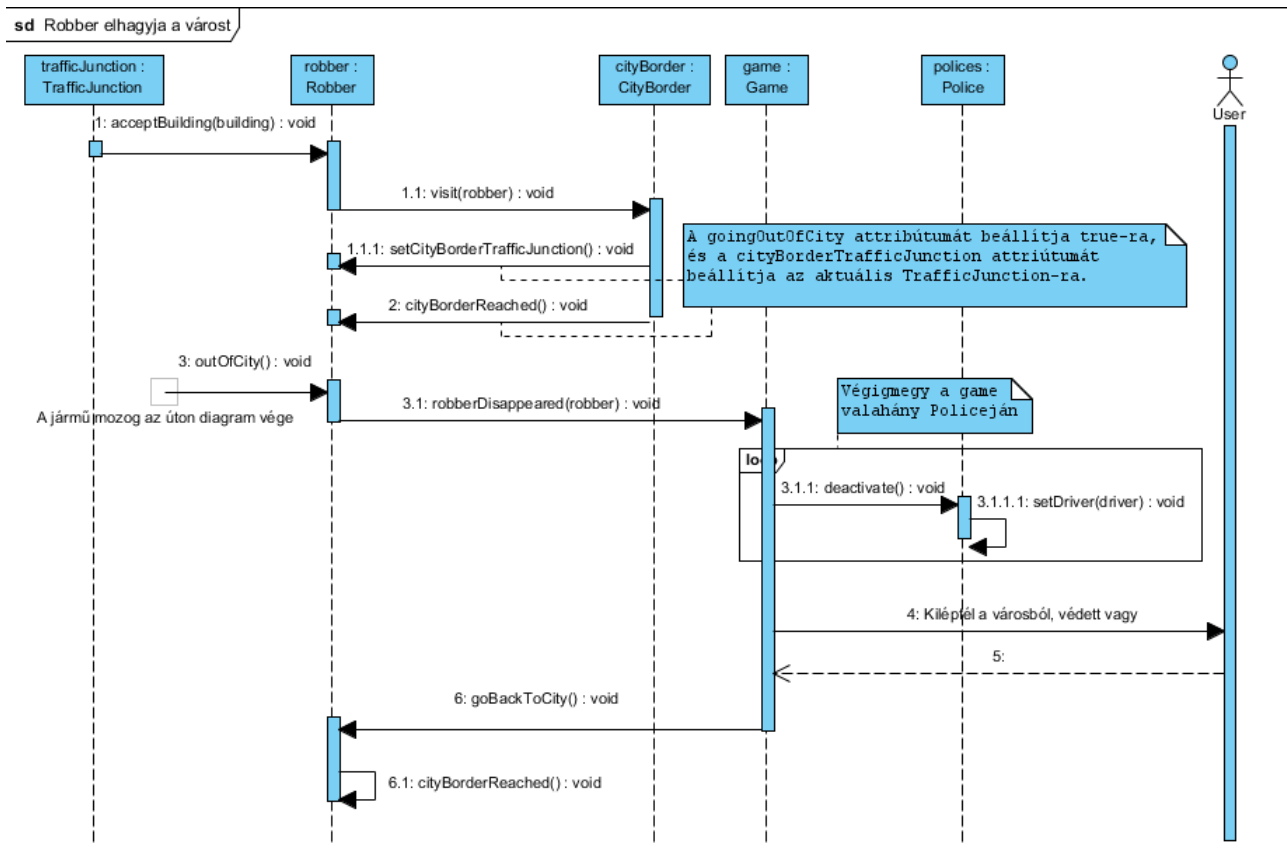


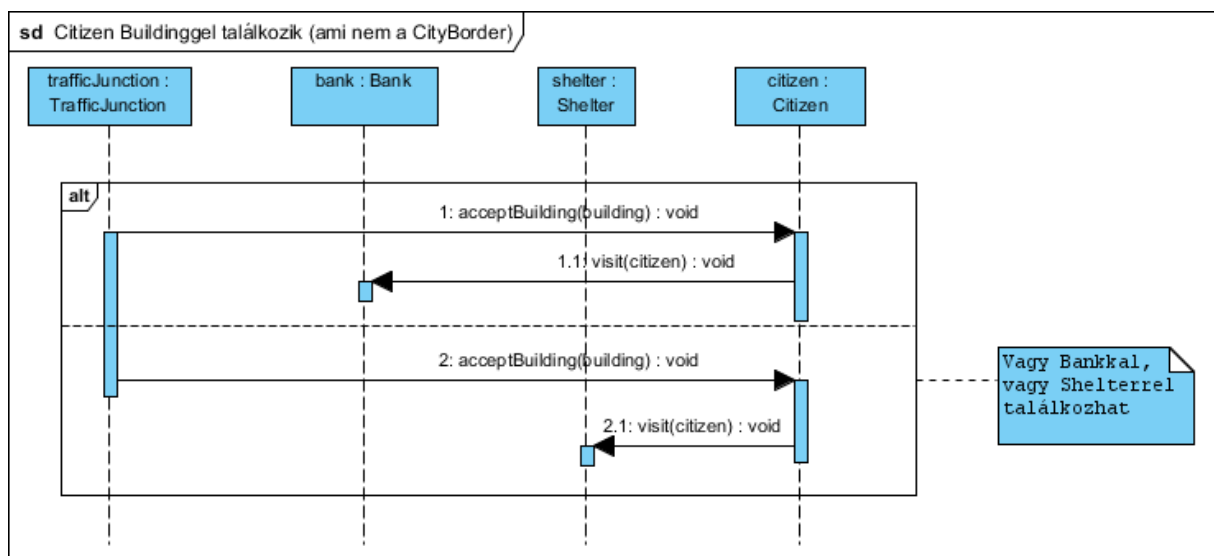
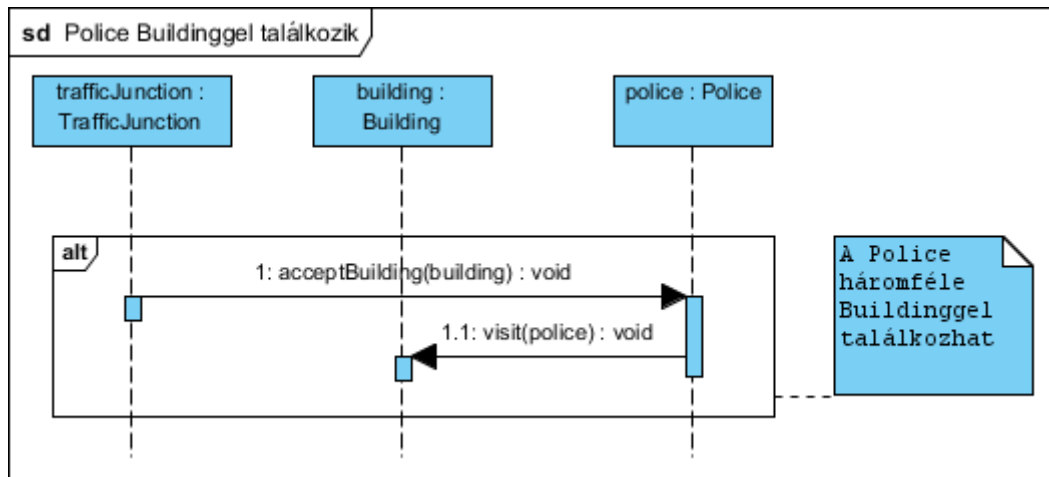
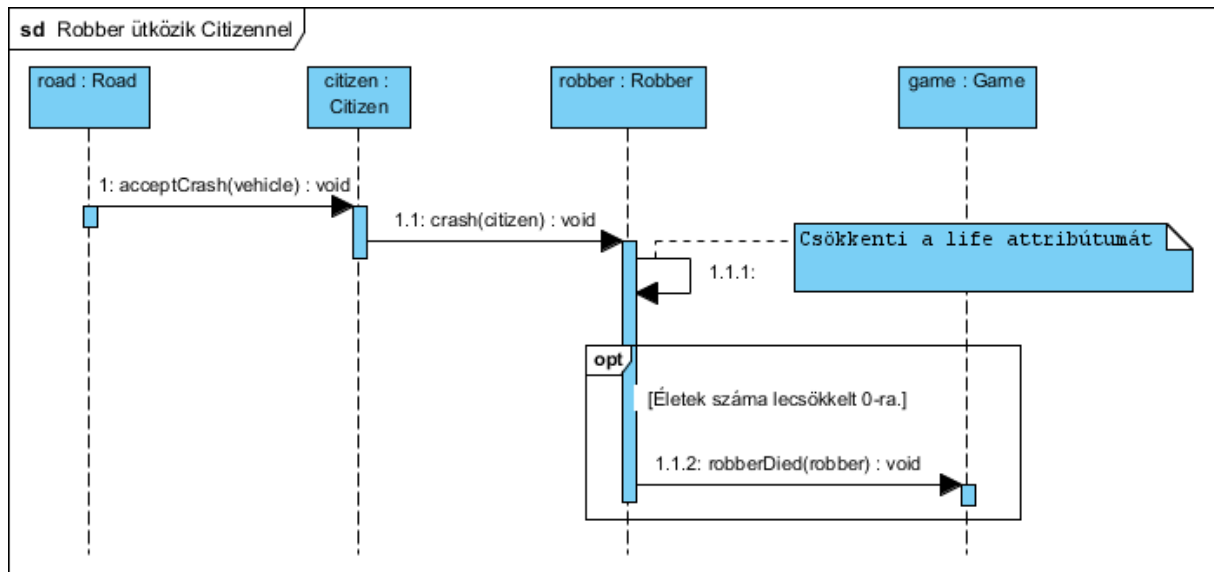


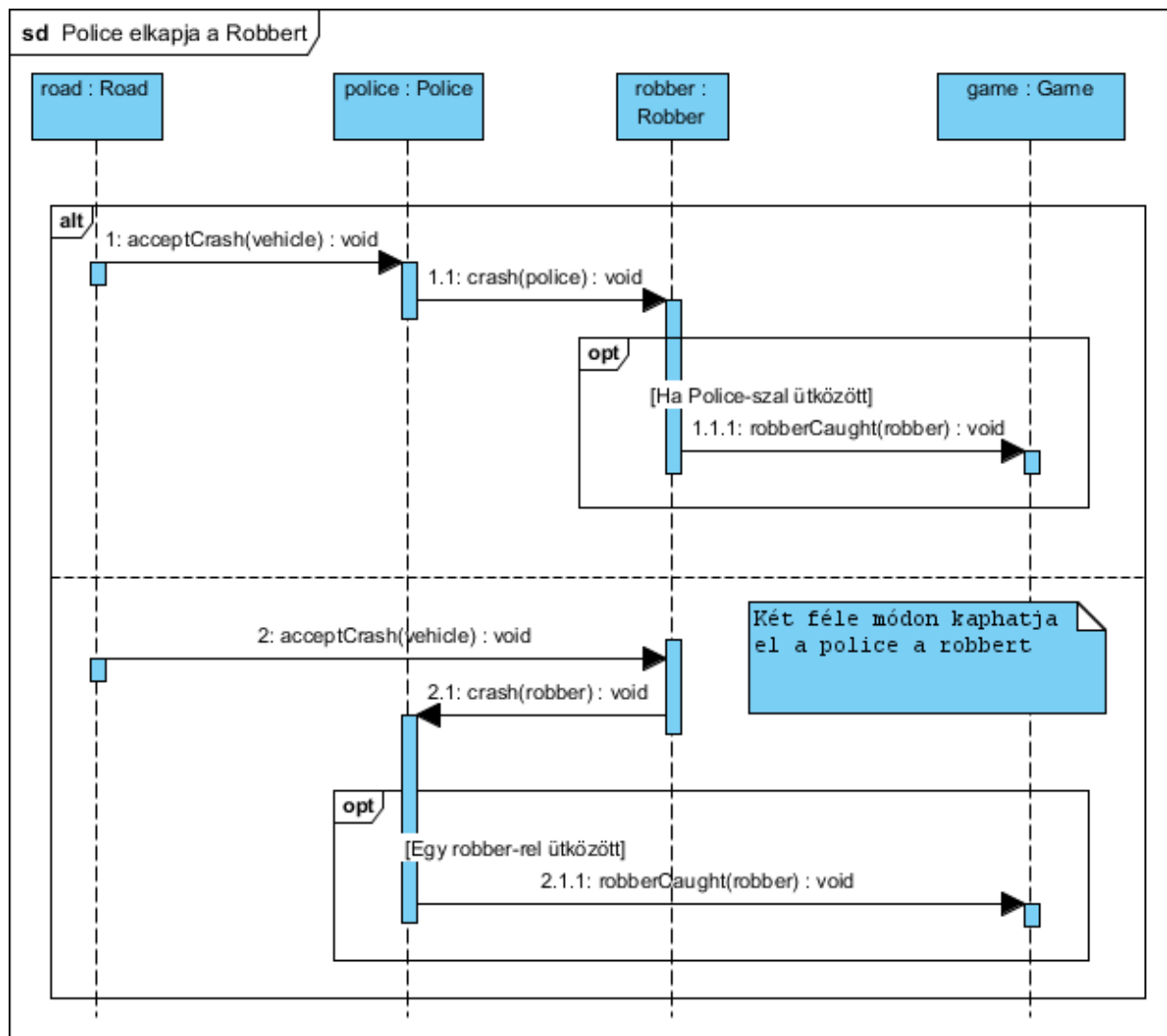


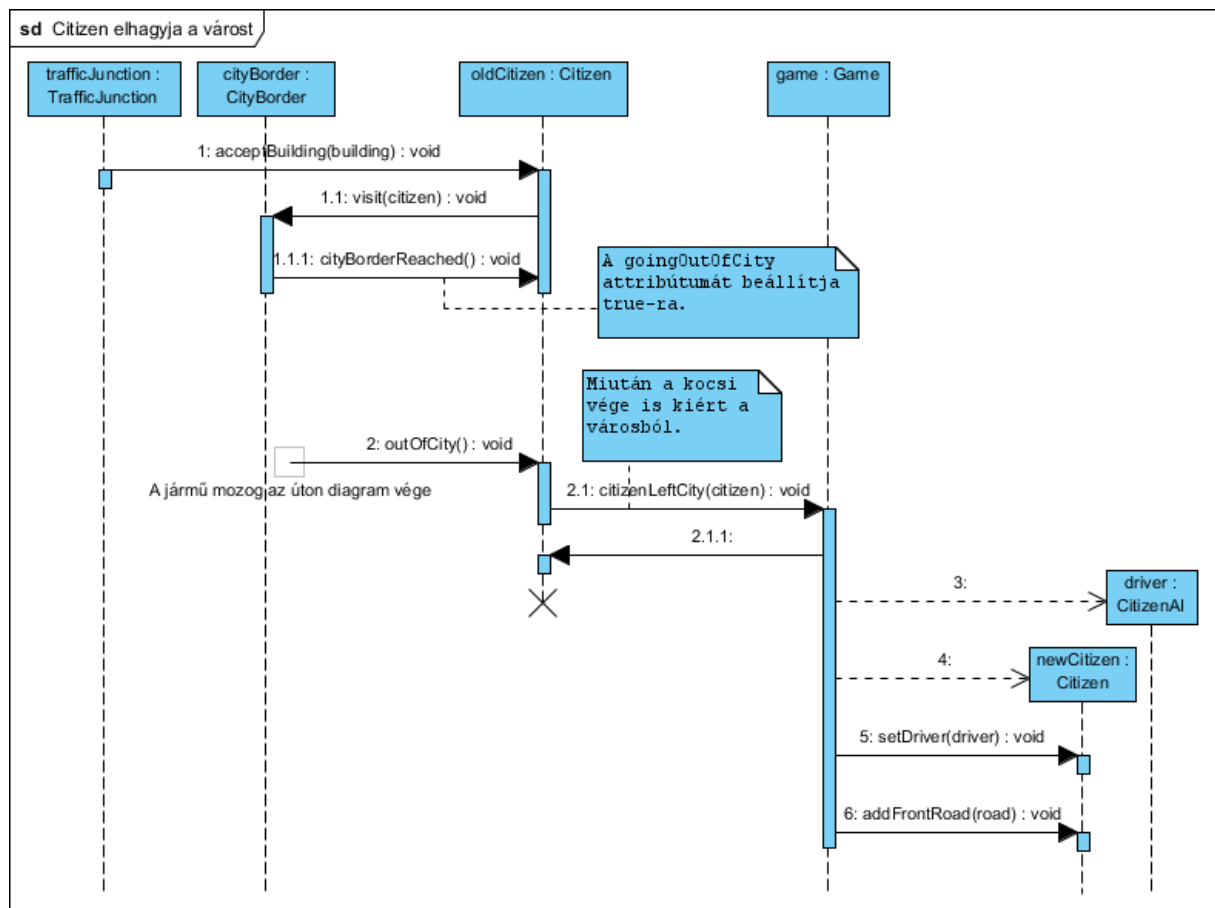




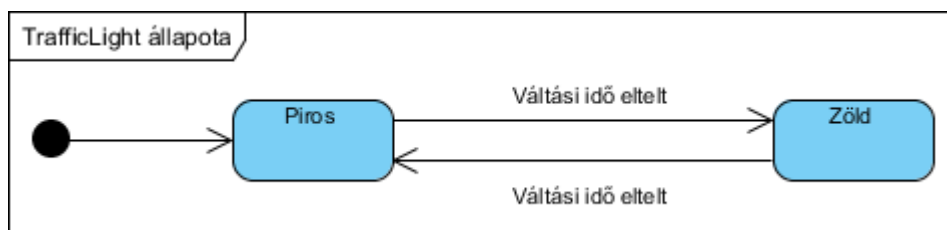
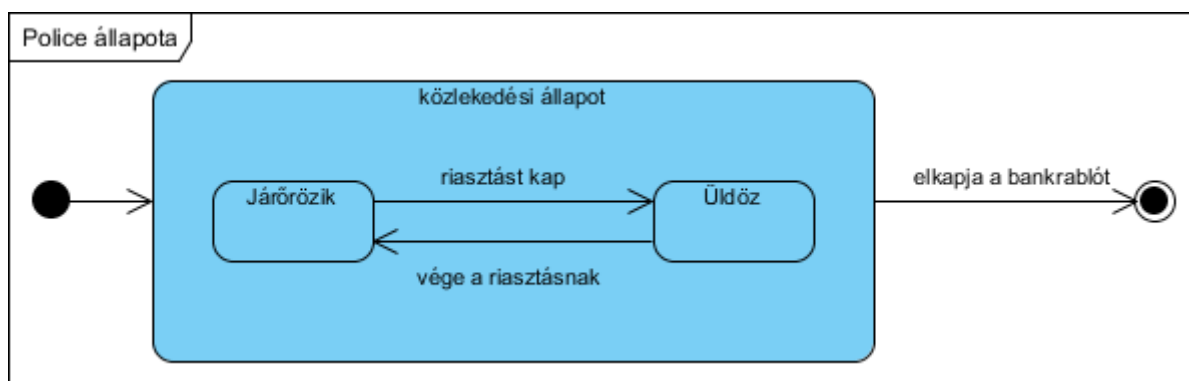
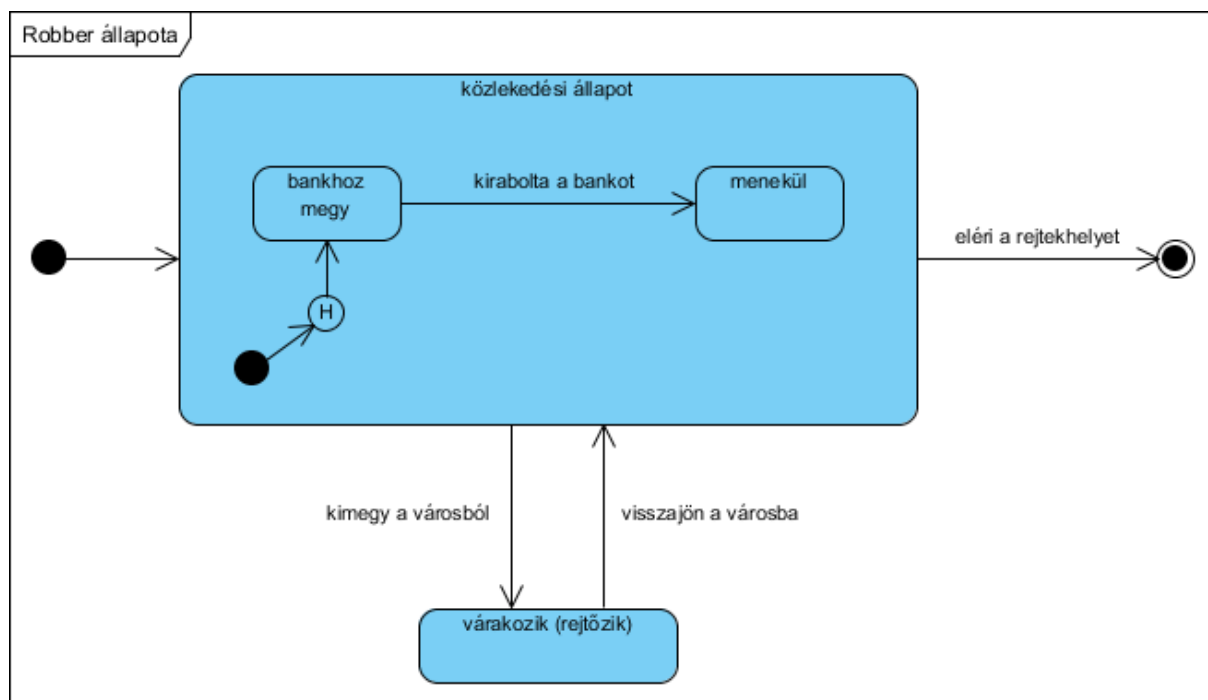








4.5 State-chartok



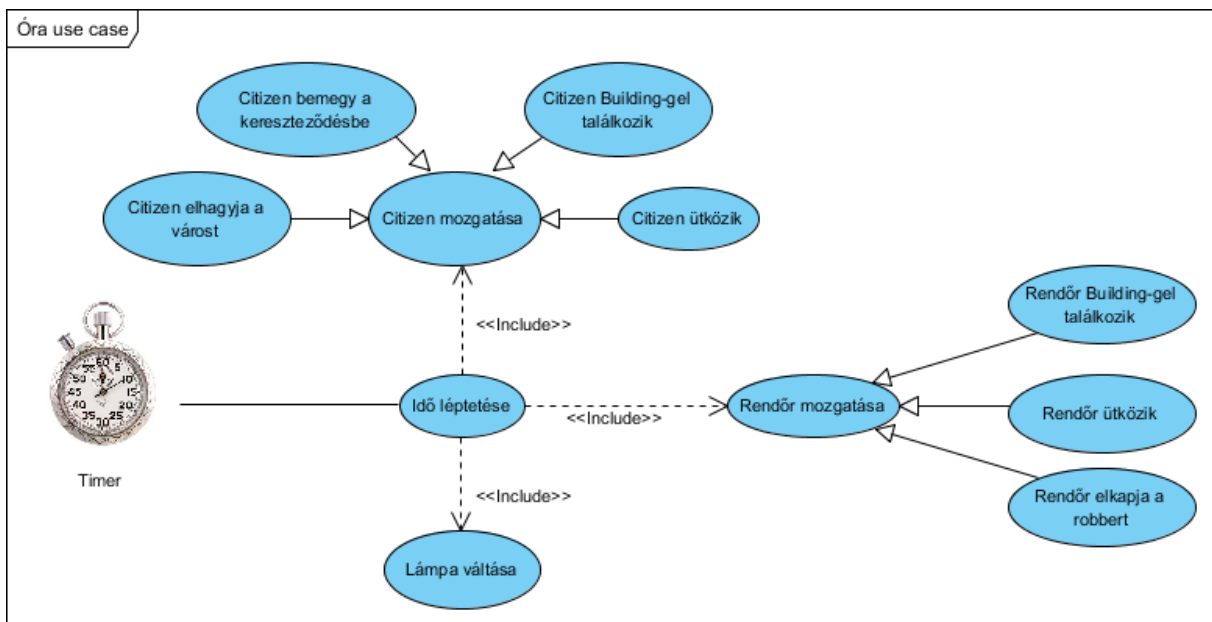
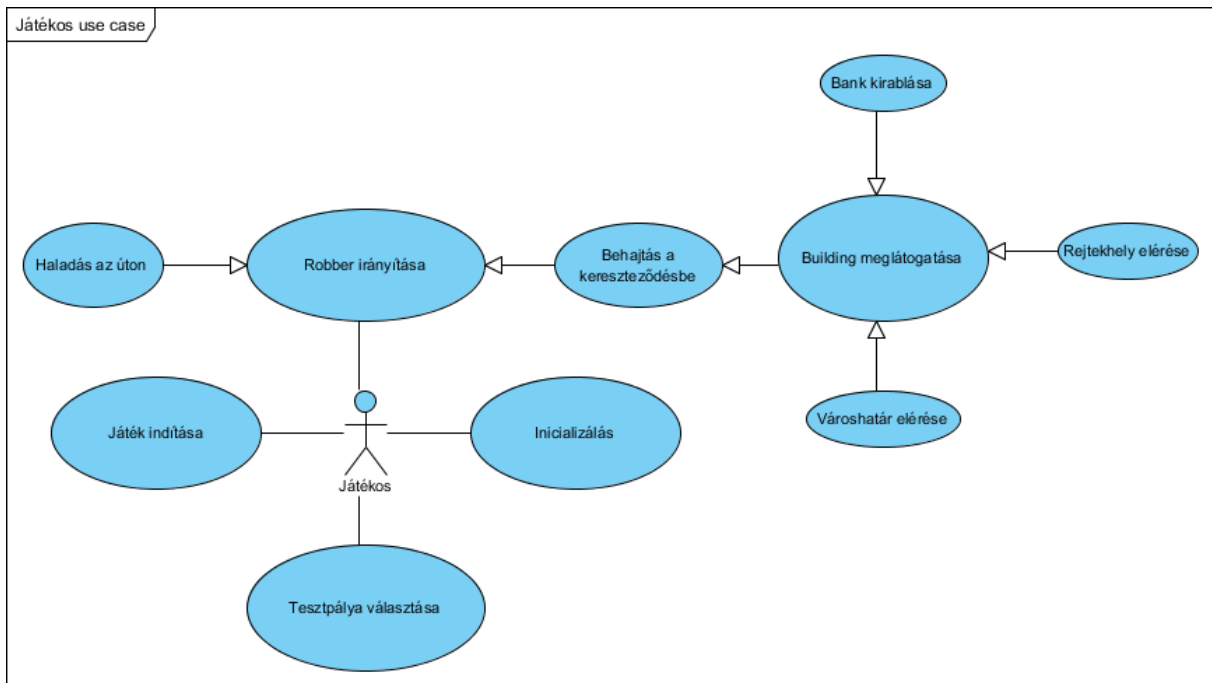
4.6 Napló

Kezdet	Időtartam	Résztevők	Leírás
2010.03.02. 14:30	4 óra	Koczka Kovács	Koczka és Kovács megtervezi a javított táblarendszert, kevés diagram készítés.
2010.03.02. 21:30	2 óra	Kovács	Osztálydiagram fejlesztése, Sequence diagramok készítése
2010.03.02. 22:00	1 óra	Mikó	Osztályok leírása fejezet, Objektumkatalógus javítása, kiegészítése
2010.03.02. 22:00	2 óra	Angeli	Osztályok leírása fejezet, Objektumkatalógus javítása, kiegészítése
2010.03.03. 9:40	1,5 óra	Koczka	Osztálydiagram fejlesztése, Sequence diagramok készítése
2010.03.03. 20:00	2 óra	Angeli Mikó	Osztályok leírása fejezet, Objektumkatalógus javítása, kiegészítése
2010.03.03. 22:30	1 óra	Koczka Kovács	Dokumentáció ellenőrzése, javítása

5. Szkeleton tervezése

5.1 A szkeleton modell valóságos use-case-ei

5.1.1 Use-case diagram



5.1.2 Use-case leírások

Use-case neve	Óra Use Case
Rövid leírás	Az adott UseCase a Timer által vezérelt Lámpát, Rendőrt és Citizen jármű cselekvéseit ábrázolja.
Aktorok	Timer
Forgatókönyv Lámpa cselekvései: 1.1 Lámpa vált: A lámpa a doTick hatására egy előre beállított időközönként a vált Piros / Zöld között. Citizen cselekvései: 2.1 Citizen mozgása: A Timer doTick hatására a jármű lekérdezi a vezetőjétől a gyorsulást, majd ezt visszakapva meghívja a road(ok) getPositionjait, megfelelő értékkel megváltoztatva beállítja őket. Majd megnézi, hogy ütközött-e más járművel. Ha a jármű eleje elér egy kereszteződéshez, akkor meghívja a kereszteződés vehicleArrived fv-ét. Ha a jármű vége épp leér egy útról, akkor meghívja a letérő út removeVehicle fv-ét, majd az út meghívja a vehicle removeLastRoad fv-ét, ami törli az utolsó utat a listájáról, majd ha már egy utat sem tárol, akkor meghívja az outOfCity fv-ét, és elhagyja a várost. 2.2 CitizenAI (a Citizen vezetője) kiszámolja a gyorsulását: doTick hatására a vehicle meghívja a Vezetője getAcceleration fv-ét. Erre a vezető lekéri a jármű első roadját. Majd ennek a roadnak meghívja a getNearestPhysicalBarrierDistance, illetve a getDistanceFromRoadend metódusát. Majd ha elég közel van az út végéhez akkor lekérdezi az ott lévő TrafficSignokat. Majd végigmegy az összes megkapott TrafficSignon és meghívja az accept függvényüket. Ezek után a vezetőn a visit() függvény az adott trafficSignokkal meghívódik. (A poliformizmus miatt a megfelelő függvény fog meghívódni). Így a vezető (CitizenAI) megkapja, hogy szabad-e az átkelés. Ez után történik még egy getVelocity hívás, hogy ne tudjon a maximális sebessége fölé gyorsulni adott helyzetben. Végül visszadjuk a vehiclenek a kiszámított gyorsulást. 2.3 Citizen elhagyja a várost: Ha a jármű egy CityBorderrel rendelkező TJ-ban van akkor, a TJ acceptálja a járművel a CityBordert, majd a jármű meghívja magával a kapott CityBorder visit fv-ét, ami meghívja a jármű cityBorderReached fv-ét. Ha az előző pontban leírt outOfCity meghívódott, a jármű szól a Gamenek (citizenLeftCity), ami törli ezt a járművet és létrehoz egy újat. 2.4 Citizen Building-gel találkozik (a CityBorderes eset részletesebben fentebb): TrafficJunctionba érve, az meghívja a citizen acceptBuilding fv-ét, mely hatására a citizen tudja, hogy melyik épülethez ért, majd a citizen meghívja az adott Building visit(citizen) fv-ét saját magával. Ez után az adott building tud reagálni, mivel tudja, hogy egy citizen érkezett hozzá. 2.5 Citizen ütközik: Az adott úton ha ütközik, akkor az út meghívja a citizennel az ütközött jármű acceptCrash fv-ét, majd ennek hatására az ütközött jármű meghívja a citizen crash fv-ét (önmagával).	

2.6 Citizen bemegy a kereszteződésbe: A régi út (amiről letér a jármű) meghívja a TrafficJunction vehicleArrived fv-ét. Ez után a fent leírt módon megtörénik a Buildingekkel való találkozás. Majd a TrafficJunction lekéri a járműtől a továbbhaladási irányt (getDirection). Erre a jármű lekéri a vezetőjétől (chooseDirection), majd ezt megkapja. A TJ az addVehicle fv-el ráteszi az új útra, ami hozzáadja a vehicle addFrontRoad fv-vel a jármű út listájának az elejére az új utat

Police cselekvései:

3.1 Police mozgatása: Leírás megegyezik a Citizen mozgatásának leírásával azzal a különbséggel hogy a rendőr nem fogja elhagyni a várost. A Timer doTick hatására a jármű lekérdezi a vezetőjétől a gyorsulást, majd ezt visszakapva. meghívja a road(ok) getPositionjait, megfelelő értékkel megváltoztatva beállítja őket. Majd megnézi, hogy ütközött-e más járművel. Ha a jármű eleje elér egy kereszteződéshez, akkor meghívja a kereszteződés vehicleArrived fv-ét. Ha a jármű vége épp leér egy útról akkor meghívja a letérő út removeVehicle fv-ét, majd az út meghívja a vehicle removeLastRoad fv ami törli az utolsó utat a listájáról.

3.2 PoliceAI (a Police vezetője) kiszámolja a gyorsulását: doTick hatására a vehicle meghívja a Vezetője (PoliceAI) getAcceleration fv-ét. Erre a vezető lekéri a jármű első roadját. Majd ennek a roadnak meghívja a getNearestPhysicalBarrierDistance, illetve a getNearestVehicle metódusát. Ha az utóbbi metódussal lekért jármű (azaz a legközelebbi kocs) a bankrabló kocsija akkor a PoliceAI nem fog lassítani (a PoliceAI tudja kit üldöz). Ezek után történik még egy getVelocity hívás, hogy a rendőr ne tudjon a maximális sebessége fölé gyorsulni adott helyzetben. Végül visszadjuk a vehiclenek a kiszámított gyorsulást.

3.3 Police elkapja a rablót: 1. Ha egy rendőr ütközik roberrel: Az adott úton ha ütközik, akkor az út meghívja az ütközött járművel az acceptCrash fv-t majd ennek hatására a rendőr meghívja az ütközött jármű crash fv-ét és ha ez éppen robber-nak a crash(police) fv-e volt, akkor az meghívja a Game robberCaught fv-t.

2. Ha egy robber ütközik rendőrrel: Az adott úton ha ütközik, akkor az út meghívja az ütközött járművel az acceptCrash fv-t majd ennek hatására a robber meghívja az ütközött jármű crash fv-t és ha ez éppen police-nak a crash(robber) fv-e volt akkor az meghívja a Game robberCaught fv-t.

3.4 Police Building-gel találkozik: TrafficJunctionba érve, az meghívja a rendőr acceptBuilding fv-ét, mely hatására a rendőr tudja, hogy melyik épülethez ért, majd a rendőr meghívja az adott Building visit fv-ét saját magával.

3.5 Police ütközik: Az adott úton ha ütközik, akkor az út meghívja az ütközött járművel az acceptCrash fv-t majd ennek hatására a rendőr meghívja az ütközött jármű crash fv-ét.

3.6 Police bemegy a kereszteződésbe:

(Megegyezik a Citizen bemegy a kereszteződésbe UseCase forgatókönyvével.)

A régi út (amiről letér a jármű) meghívja a TrafficJunction vehicleArrived fv-ét. Ez után a fent leírt módon megtörénik a Buildingekkel való találkozás. Majd a TrafficJunction lekéri a járműtől a továbbhaladási irányt (getDirection). Erre a jármű lekéri a vezetőjétől (chooseDirection), majd ezt megkapja. A TJ az addVehicle fv-el

rátesszi az új útra, ami hozzáadja a vehicle addFrontRoad fv-vel a jármű út listájának az elejére az új utat.

Use-case neve	Játékos Use Case
Rövid leírás	Az adott UseCase a játékos által cselekvéseit ábrázolja.
Aktorok	Játékos
Játékos cselekvései:	
4.1 Robber haladása az úton: A leírás megegyezik a Citizen mozgásával. A jármű lekérdezi a vezetőjétől a gyorsulást, majd ezt visszakapva. meghívja a road(ok) getPositionjait, megfelelő értékkel megváltoztatva beállítja őket. Majd megnézi, hogy ütközött-e más járművel. Ha a jármű eleje elér egy kereszteződéshez, akkor meghívja a kereszteződés vehicleArrived fv-ét. Ha a jármű vége épp leér egy útról akkor meghívja a letérő út removeVehicle fv-ét, majd az út meghívja a vehicle removeLastRoad fv ami törli az utolsó utat a listájáról.	
4.2 Robber buildingel találkozik (Shelter elérése): A TrafficJunctionba érve, az meghívja a robber acceptBuilding fv-ét. Ennek hatására a robber tudja, hogy a shelterhez ért, majd a robber meghívja a Shelter visit(robber) fv-ét saját magával. Ez után a shelter lekéri hogy ezt a rablót körözik-e már (getWanted). Ha igen szól a Gamenek, hogy a rabló elérte a sheltert (robberReached Shelter).	
4.3 Robber buildingel találkozik (Bank rablása): A TrafficJunctionba érve, az meghívja a robber acceptBuilding fv-ét. Ennek hatására a robber tudja, hogy a bankhoz ért, majd a robber meghívja az adott Bank visit(robber) fv-ét saját magával. Ez után a bank lekéri hogy ezt a rablót körözik-e már (getWanted). Ha nem akkor beállítja a setWanted fv-el, és szól a Gamenek, hogy kirabolták a bankot (bankRobbed). Ez után a game sorban aktiválja a rendőröket azok activate fv-ével.	
4.4 Robber buildingel találkozik (Város elhagyása): A TrafficJunctionba érve, az meghívja a robber acceptBuilding fv-ét. Ennek hatására a robber tudja, hogy a CityBorderhez ért, majd a robber meghívja az adott CityBorder visit(robber) fv-ét saját magával. Ez után CityBorder beállítja a rabló goingOutOfCity attribútumát és meghívja a cityBorderReached fv-t. Mitán a jármű vége letért erről az útról (outOfCity) akkor a robber szól a Gamenek, hogy most eltűnt (robberDisappeared). Erre a Game sorra deaktiválja a rendőröket és szól a játékosnak, hogy most védett állapotban van.	
4.5 Robber bemegy a kereszteződésbe: (Megegyezik a Citizen bemegy a kereszteződésbe UseCase forgatókönyvével.) Miután a régi út meghívja a TJ vehicleArrived fv-ét, után a fent leírt módon megtörénik a Buildingekkel való találkozás. Majd a TJ lekéri a járműtől a továbbhaladási irányt (getDirection), erre a jármű lekéri a vezetőjétől (chooseDirection), majd ezt visszakapva a TJ az addVehicle fv-el rátesszi az új útra, ami hozzáadja a vehicle addFrontRoad fv-vel a jármű út listájának az elejére az új utat.	
4.6 Játék kezdete, inicializálás: Az inicializálás megkezdése során a Game létrehozza a map-et. A map az általa tárolt objektumokat ezt követően hozza létre, kezdve a következőkkel: timer, map, trafficJunction, building (itt az épületek a kereszteződések addBuilding metódusa segítségével már hozzá is lesznek rendelve a különböző	

trafficJunction-okhoz). Ezután következik a road-ok létrehozása, majd ezek összekötése a megfelelő trafficJunction-okkal. Ha ez megtörtént akkor létrejönnek a trafficSign-ok melyek a különböző utak addTrafficSign metódusainak segítségével elhelyezésre kerülnek rajtuk. (trafficLight esetén még a timer tick-szolgáltatására való feliratkozás is megtörténik). Majd végül létrejönnek a robber, police valamint a citizen objektumok, mindegyik feliratkozása megtörténik a timer tick-szolgáltatására, valamint rákerülnek a különböző utakra a road addVehicle metódusának segítségével.

4.7 Robber ütközik: Alapvetően ugyanaz mint a Citizen ütközik, csak életet veszít. Rendőrrel való ütközéskor játék vége lesz.

5.2 Architektúra.

A szkeleton architektúráját úgy készítettük el, hogy letesztelhessük vele, hogy az előzetesen megtervezett szekvenciadiagramoknak megfelelően működnek-e az program alapvető folyamatai, amelyeket use caseken ábrázoltunk. A tesztpályákat előre inicializáltuk, így nyílik lehetőség arra, hogy letesztelhessük a játék fontos szituációit, hogy azok megfelelően működnek az előző analízisben definiált szekvenciadiagramoknak megfelelően. A skeleton tester szekvenciális lefutású, így itt nincs szükség a konkurens feladatok vizsgálatára. A jelenlegi skeleton a timer általi use case-eket teszteli.

Alapvetően a modell a valóságot olyan mélységben próbálja tükrözni, hogy a járművek mozgás folyamatosnak tűnjön, illetve a járművek minden pillanatban jellemezhetők helyzetükkel, gyorsulásukkal, sebességükkel. Ahhoz, hogy ezt meg lehessen valósítani a Timer doTick függvényhívásai nagyon gyakoriak kellene hogy legyenek. Jelen skeleton tesztben ez azt eredményezné, hogy pl. egy citizen kimegy a városból letesztelése, nagyon sok doTicket venne igénybe, ha az autó akár csak 1 métert akar haladni. Ezért úgy fogunk tesztelni, hogy a doTicket nem a Timer hívja meg, hanem a tesztelő személy. A szituációk, azért hogy ne legyen nagyon sok Tick-re szükség, úgy lettek kitalálva, hogy viszonylag kevés lépésből "látszódn" a lényeg, hogy működik-e az adott funkció.

Az alábbi tesztpálya leírásokban a Tesztelt UseCase számozások, jelen dokumentáció 5.1.2 alszámozásait jelentik. Amely szám mellett csillag van, annak forgatókönyve nagyon hasonló a 'fő' UseCasehez, ezért ahhoz nincs külön tesztpálya.

0. tesztpálya: Inicializalas

Objektumok: minden egy általános futattás során felhasznált objektumtípusból legalább egy darab

(Game, Timer, VehicleType, Map, TrafficJunction, Road, Citizen, CitizenOnRoad, stb.)

Cél: az objektumok létrehozásának és egymással való kapcsolatok létrejöttének bemutatása

Tesztelt UseCase: 4.6

1. tesztpálya: Jarmű mozog az úton

Objektumok:

-Citizen: az autó, amely mozog az úton.

-Road: az út, amelyen mozog az autó.

-TrafficJunction: az útkereszteződés, melyben találkoznak az utak, valamint melyek segítségével a járművek egyik útról a másikra kerülhetnek át.

Tesztelt UseCase: 2.1, 3.1*, 4.1*

2. tesztpálya: Jarmu bemegy a keresztezesbe

Objektumok:

- Citizen: -az autó, amely behajt a kereszteződésbe.
- Building: -az útkereszteződésben található épület.
- TrafficJunction: -az útkereszteződés, melyben találkoznak az utak, valamint melyek

segítségével az ide behajtó jármű egyik útról a másikra kerül át

- oldRoad: a régi út, amiről lemegy az autó, a TrafficJunctionhoz kapcsolódik
- newRoad: az új út, amire rámegy az autó, a TrafficJunctionhoz kapcsolódik

Cél: annak modellezése, miként egy jármű behajt a kereszteződésbe.

Tesztelt UseCase: 2.6, 3.6*, 4.5***3. tesztpálya: Citizen elhagyja a varost**

Objektumok:

- TrafficJunction - az az útkereszteződés, ami a városhatárt is tartalmazza
- 2 Road - egyik az az út, ahol a Citizen eléri a városhatárt, a másik pedig ahol az új

Citizen lépbe a városba

- CityBorder - Building típusú, a városhatár, ahol a Citizen távozik
- 2 Citizen - egy aki kilép, egy aki belép helyette
- Game - értesül, hogy a Citizen elhagyta a várost és rátesz egy új autót a térképre

Tesztelt UseCase: 2.3**4. tesztpálya: CitizenAI kiszámolja a gyorsulást**

Objektumok:

- Citizen - a városi autó
- CitizenAI - a városi autó irányításáért felelős objektum
- Road - az út ahol halad
- TrafficSign - a közlekedési tábla, amik befolyásolhatja a Citizen gyorsulását

Tesztelt UseCase: 2.2, 3.2***5. tesztpálya: PhysicalBarrier meghatározása**

Objektumok:

- 1 TrafficJunction - az út után ez a TrafficJunction következik, ez is jelenthet torlaszt az autó számára

- 2 Road - egyik úton lévő autókra határozódik meg, hogy van-e előttük akadály, a másik útról jövő autó pedig megint csak akadályként funkcionál

- 3 Citizen - 2 egymás mögött haladó autó, illetve még egy, ami a másik útról jön

Tesztelt UseCase: 2.2, 3.2***6. tesztpálya: Robber kirabolja a Bankot**

Objektumok:

- Bank: kirabolt Bank
- Robber: a bankrabló, aki rabolt
- TrafficJunction: az az útkereszteződés, ahol egy Bank helyezkedik el
- Road: ezen az úton éri el a Robber a Bankot
- Police: rendőrautó, ami aktiválódik a Bank kirablására
- Game: értesül, hogy a Robber kirabolta a bankot és tudatja a rendőrökkel ezt

Tesztelt UseCase: 4.3*, 4.2, 3.4*, 2.4**7. tesztpálya: Robber elhagyja a varost**

Objektumok:

- Robber: a bankrabló, aki kimegy a városból
- CityBorder: a városhatár, ahol a Robber távozik
- TrafficJunction: az az útkereszteződés, ami a városhatárt is tartalmazza
- Road: ezen az úton éri el a Robber a városhatárt
- Police: rendőrautó, ami nem üldözi tovább a Robbert miután távozott a városból
- Game: értesül, hogy a Robber elhagyta a várost és a rendőrök üldözését megszünteti

Tesztelt UseCase: 4.4

8. tesztpálya: Police elkapja a Robbert

Objektumok:

- Robber: a bankrabló, akit elkap a Police
- Police: a rendőr, aki elkapja a Robbert
- Road: az út, amin a Police elkapja a Robbert
- Game: értesül, hogy elkapták a Robbert, így vége a játéknak

Tesztelt UseCase: 3.3

9. tesztpálya: Robber utkozik Citizennel

Objektumok:

- Robber: a bankrabló, akivel ütközik a Citizen
- Citizen: a városi autós, akivel ütközik a Robber
- Road: az út, amin a Citizen ütközik a Robberrel
- Game: értesül, ha elfogyott a Robber élete, s így vége a játéknak

Tesztelt UseCase: 4.7

10. tesztpálya: CitizenAI dont a StopSignnal

Objektumok:

- 3 TrafficJunction - Egy az első út elején, egy a két út között, egy a második út végén
- 2 Road - 2 út egymással szemben,
- 2 Citizen - Egyik az egyik úton másik a másik úton.
- StopSign - Stop tábla

Tesztelt UseCase: 2.2

11. tesztpálya: A keresztezodes foglalt-e

Objektumok:

- 3 TrafficJunction: Egy az első út elején, egy a két út között, egy a második út végén
- 2 Road - TJ közti utak

Tesztelt UseCase: 2.2, 3.2

5.3 A szkeleton kezelői felületének terve, dialógusok

A szkeleton kezelői felülete egy egyszerű konzolos megjelenítési felület, amelyen a use case-knek megfelelő tesztelési esetek egyszerűen be vannak számozva és számuk kiválasztásával lefutnak. Lefutásuk közben kiír minden objektumot, amin metódust hívtak meg, a metódus nevét, paraméter listáját, visszatérési értékét (visszatérésakor). Ez már lehetséges, hisz a szkeleton már tartalmazza a kész szoftver minden lényegi osztályát és metódusát, de azoknak csak a deklarációit tartalmazza (változókat fel nem használva), a lényegi logikát már nem. Így egyértelműen eldönthető, hogy a szoftver képes lesz-e az elvárt működését teljesíteni.

A tesztelés folyamán előállhatnak döntési helyzetek, ilyenkor a szkeleton ugyancsak a konzolos felületén a függvényhívások kiírása közben megáll és felteszi a kérdést, majd az arra kapott válasznak megfelelően fut tovább. Ezzel a flexibilitással csökkenthető a tesztesetek száma, de mégis szélesebb körű tesztelhetőséget tesz lehetővé. A még gyorsabb tesztelhetőség elősegítésére a szkeleton tartalmaz automatikusan megválaszolt kérdéseket is, így azokra adandó egyértelmű válasszal már nem kell időt pazarolnia a tesztelőnek (ezek főleg a program belső működéséből adódnak), de mégis láthatja, hogy ott a program egy döntés helyzetet is megfelelően lekezelt.

Így a szkeleton a szekvenciadiagramoknak megfelelő működést fogja generálni. Egyszerű és könnyen összehasonlítható módon írja ki az adatokat, amik így a szekvenciadiagramok megfelelő entitásaival egyértelműen párhuzamba állíthatók.

Egy metódushívás és visszatérés esetén kiírt adatok a következők:

- Metódushívás esetén a CALL szót, konstruktorhívás esetén CREATE szót, míg visszatérésakor a RETURN szót
- Objektum egyedi azonosító száma (a teszt esetben minden résztvevő objektum 1-től számozva egyedi számot kap, így könnyen azonosítható lesz)
- Objektum típusa (osztálya)
- A metódus visszatérési értékének típusa, a metódus neve és paraméter listája (ami tartalmazza a paraméterek típusát, nevét, értékét - objektum esetén érték alatt az objektum típusát és egyedi számát kiírva)
- Visszatérés esetén a visszatérési értéket (objektum esetén a típusát és egyedi azonosító számát)

Formátumra példa:

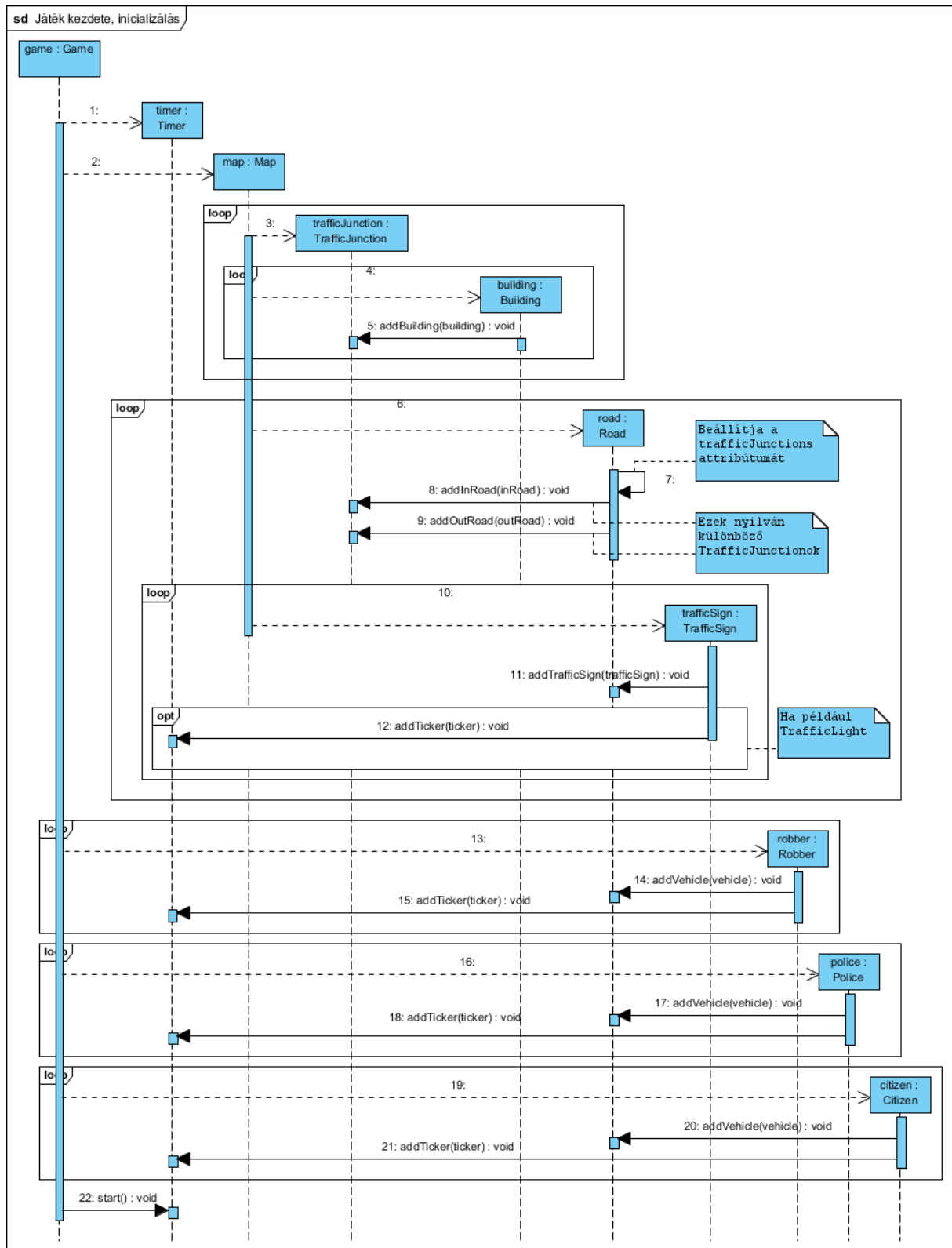
```
CALL|#1:Citizen|void doTick(int time = 5, int deltatime = 5)
  CALL|#2:CitizenAI|double getAcceleration()
  RETURN|#2:CitizenAI|double getAcceleration() = 3
  CALL|#3:Road|void moveVehicle(Vehicle vehicle = #1:Citizen, double distance = 5.0)
    CALL|#4:VehicleOnRoad|double getPosition()
    RETURN|#4:VehicleOnRoad|double getPosition() = 0.0
    CALL|#4:VehicleOnRoad|void setPosition(double position = 0.5)
    RETURN|#4:VehicleOnRoad|void setPosition(double position = 0.5)
    CALL|#5:VehicleOnRoad|Vehicle getVehicle()
    RETURN|#5:VehicleOnRoad|Vehicle getVehicle() = #6:Citizen
  QUESTION Utkozott ezzel az autoval? #6:Citizen (Y/N)
  Y
    CALL|#6:Citizen|void acceptCrash(Vehicle vehicle = #1:Citizen)
    RETURN|#6:Citizen|void acceptCrash(Vehicle vehicle = #1:Citizen)
  QUESTION A jarmu eleje elerte az ut veget? #1:Citizen (Y/N)
  Y
    CALL|#7:TrafficJunction|void vehicleArrived(Vehicle vehicle = #1:Citizen)
    RETURN|#7:TrafficJunction|void vehicleArrived(Vehicle vehicle = #1:Citizen)
  QUESTION A jarmu vege elhagyta az utat? #1:Citizen (Y/N)
  Y
    CALL|#3:Road|void removeVehicle(Vehicle vehicle = #1:Citizen)
    RETURN|#3:Road|void removeVehicle(Vehicle vehicle = #1:Citizen)
    CALL|#1:Citizen|void removeLastRoad()
  QUESTION Nincs tobb utan rajta? #1:Citizen (Y/N)
  Y
    CALL|#1:Citizen|void outOfCity()
    RETURN|#1:Citizen|void outOfCity()
    RETURN|#1:Citizen|void removeLastRoad()
  RETURN|#3:Road|void moveVehicle(Vehicle vehicle = #1:Citizen, double distance = 5.0)
```

5.4 Szekvencia diagramok a belső működésre

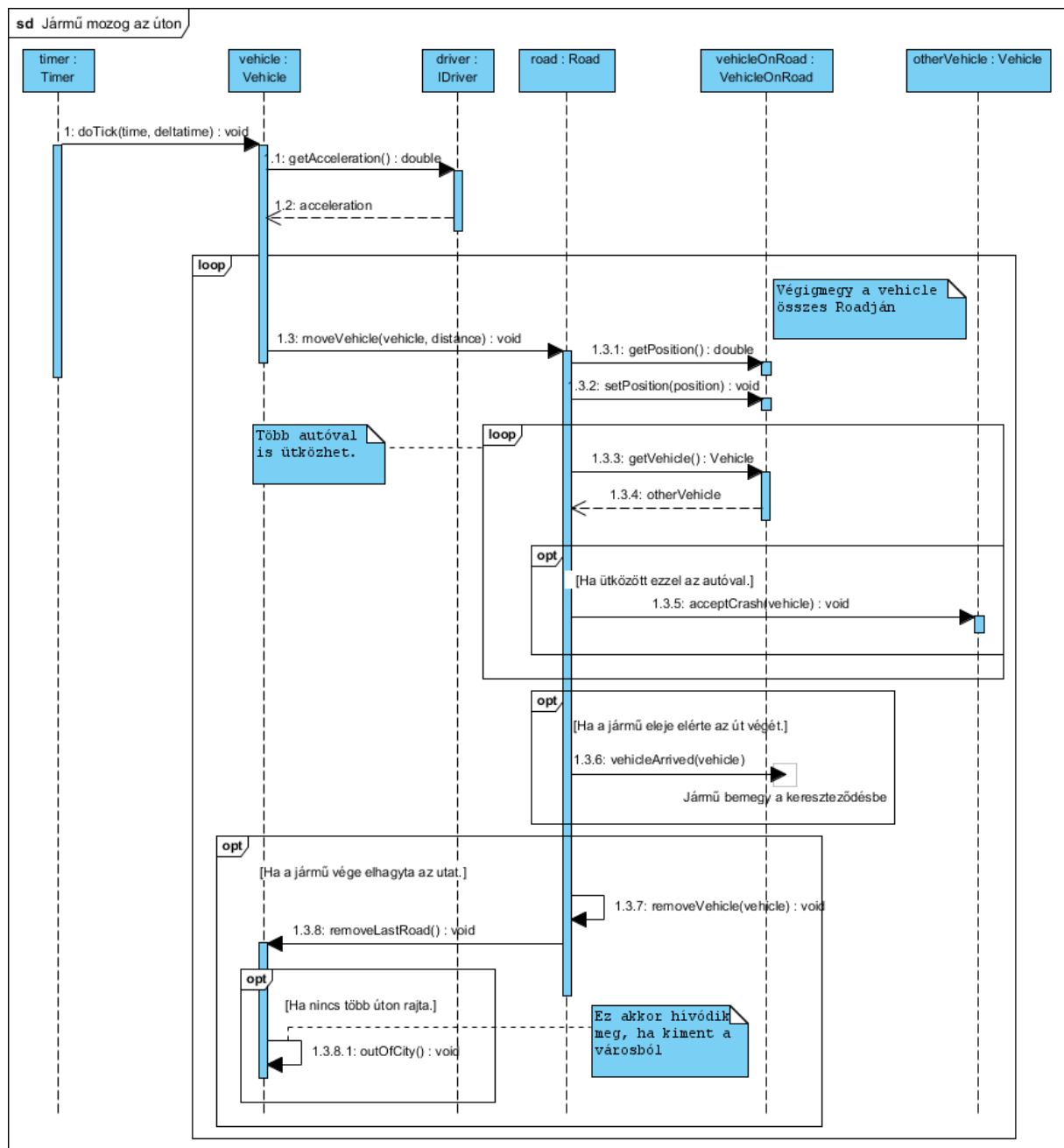
A belső működést reprezentáló use case-ek szekvenciadiagramjai megtalálhatók a dokumentáció korábbi fejezeteiben.

Korábbi fejezetek: 5.1 (use case diagrammok) illetve 4.4 (szekvencia diagrammok)

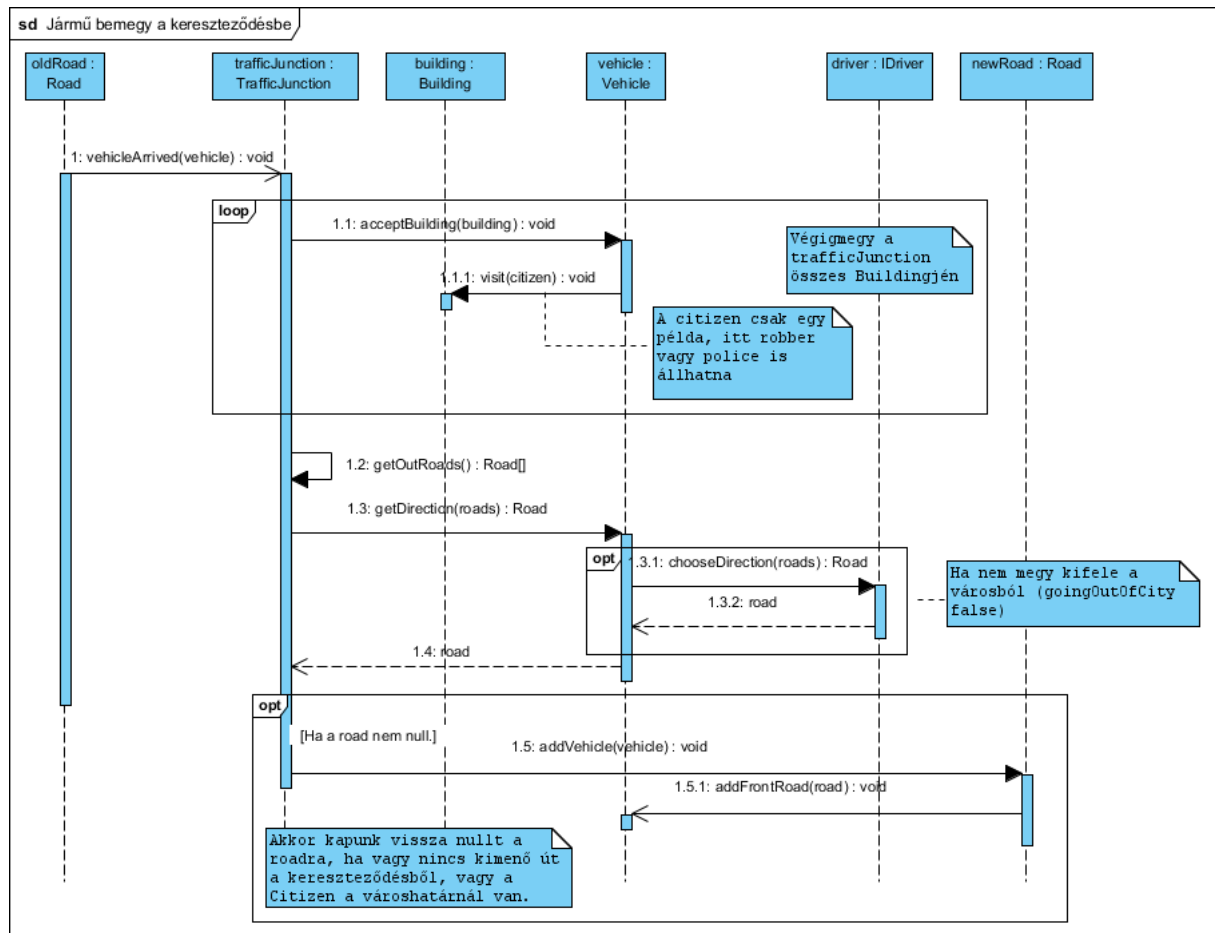
0. tesztet diagramja



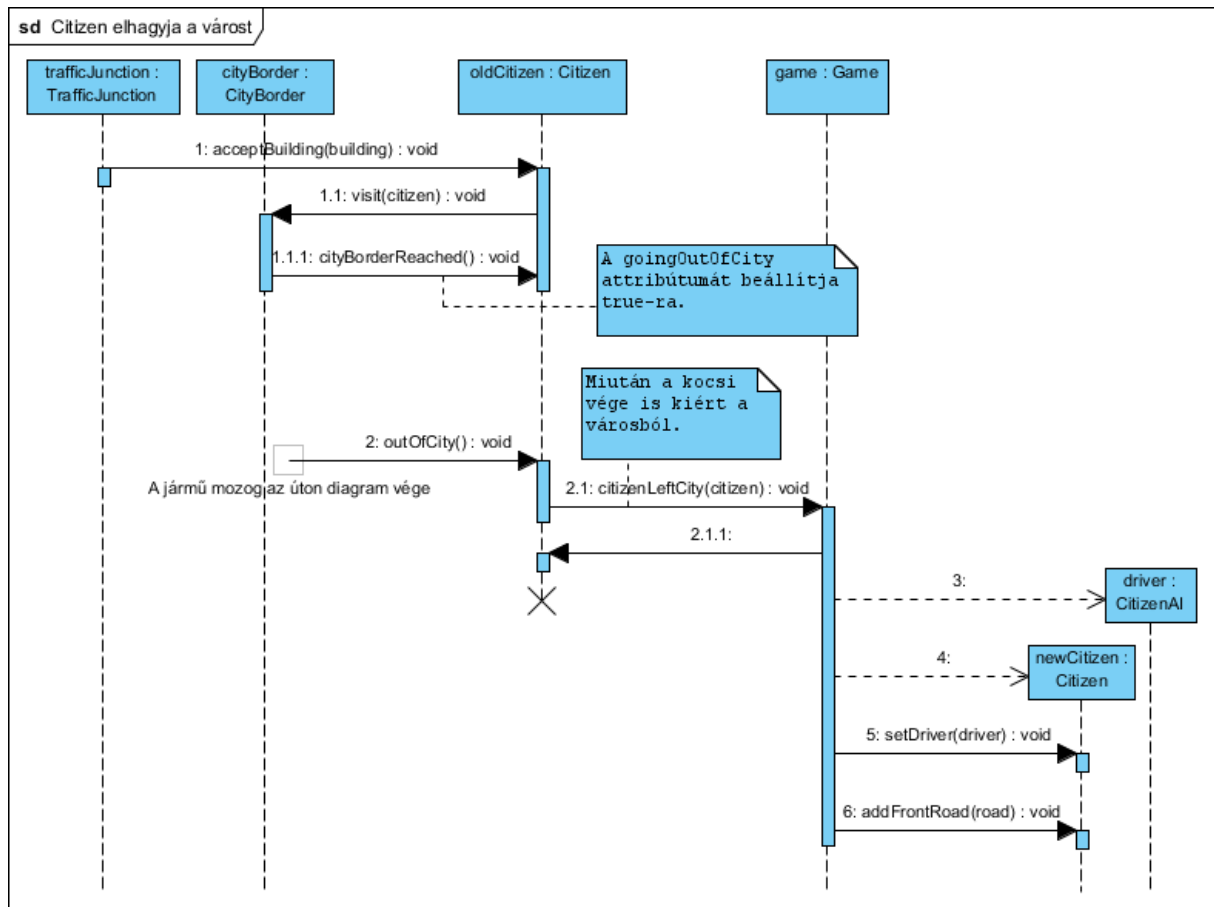
1. teszteset diagramja



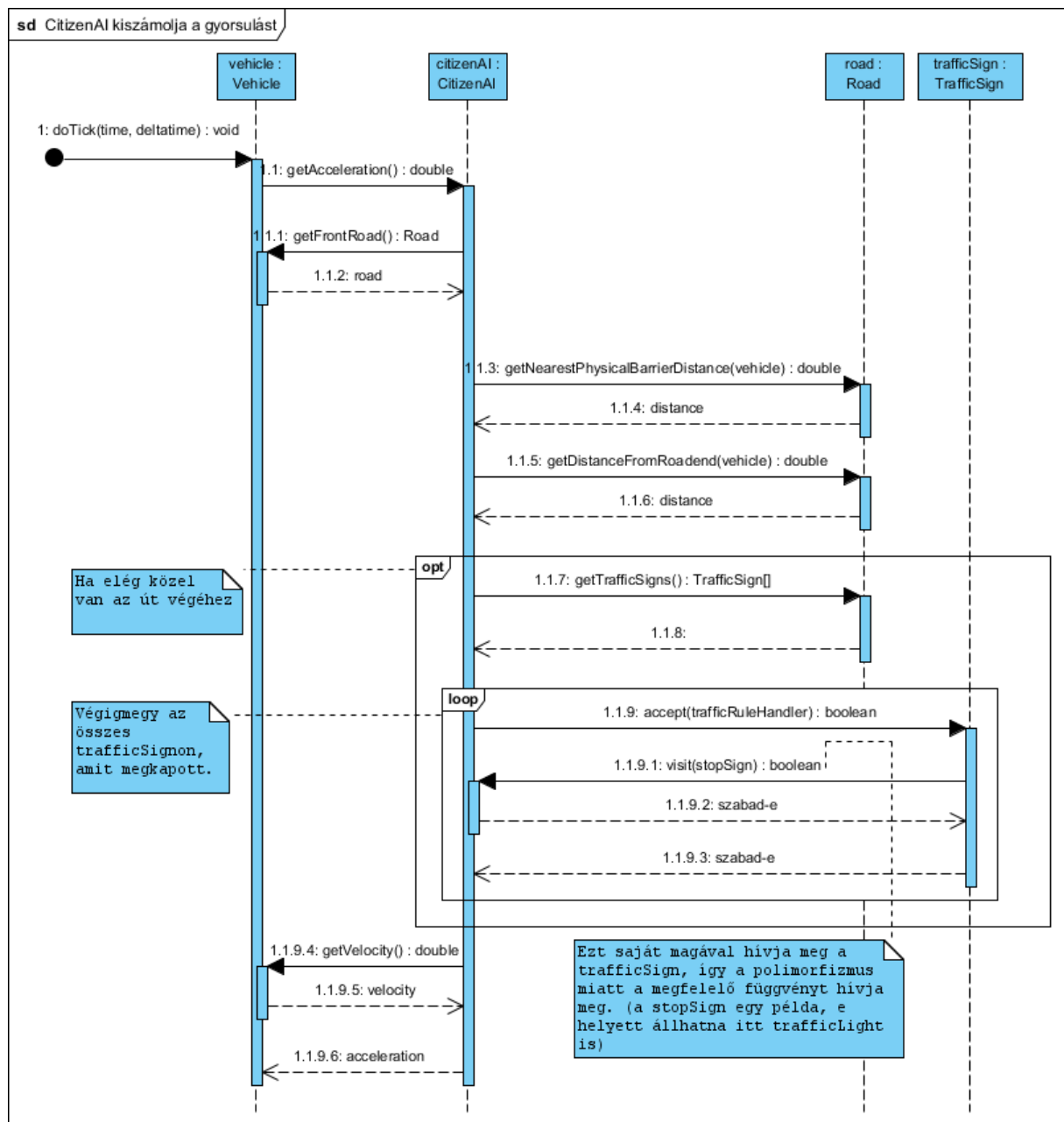
2. teszteset diagramja



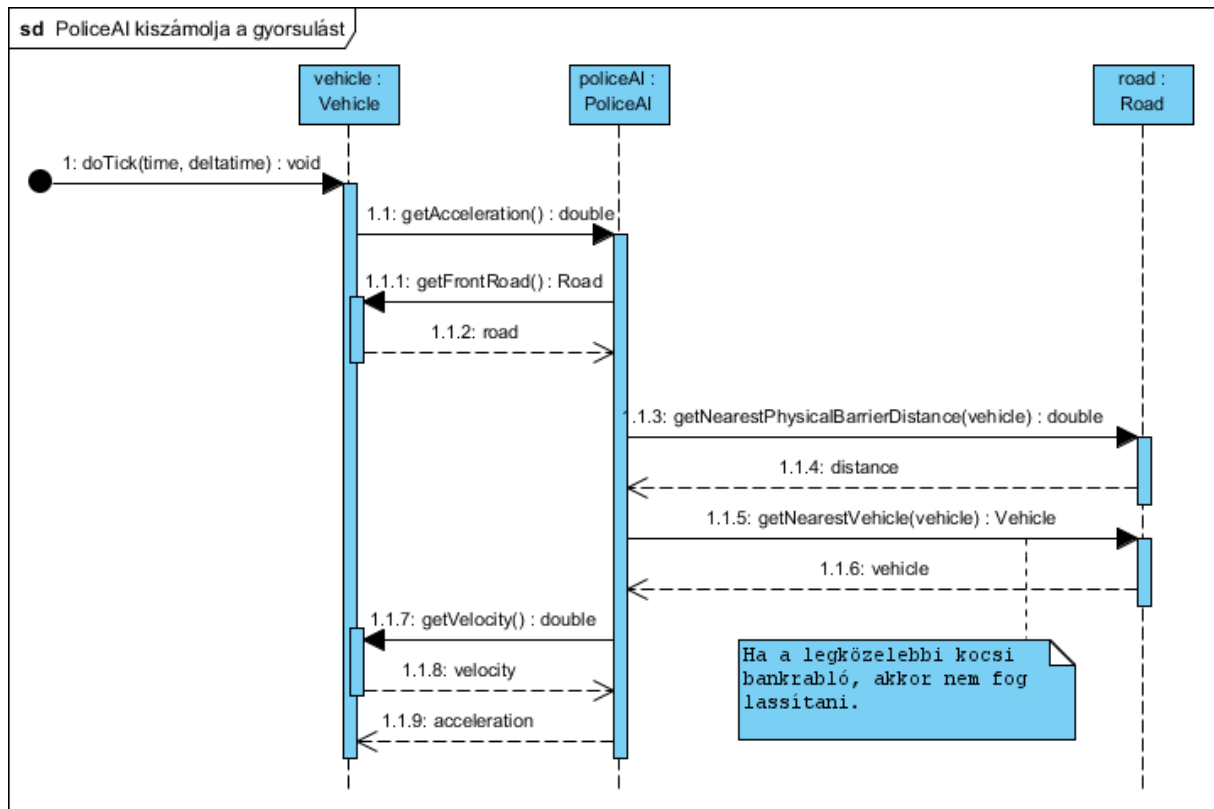
3. teszteset diagramja



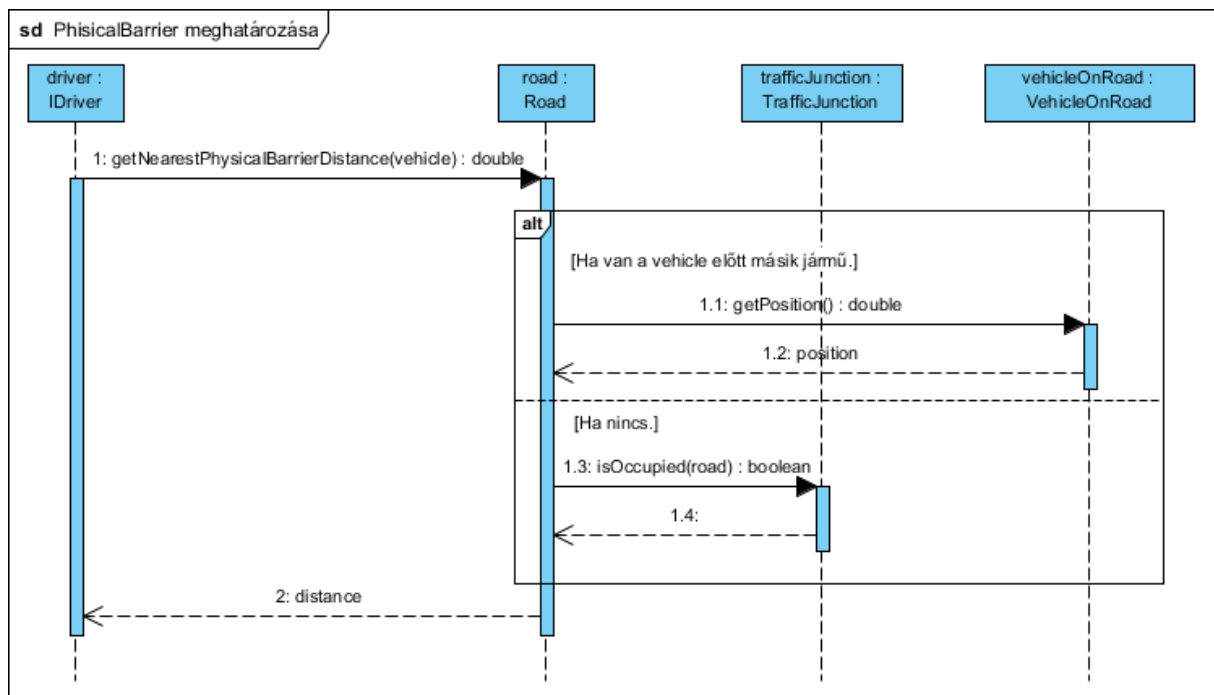
4. teszteset diagramja



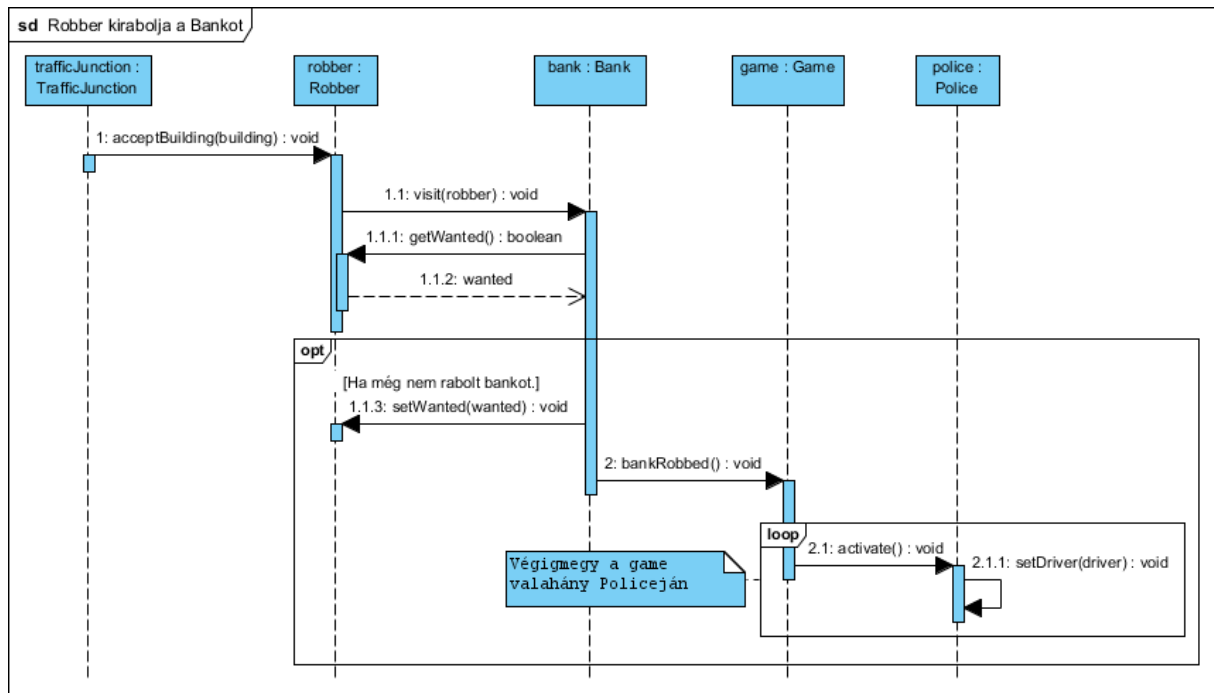
Hasonló esetek:



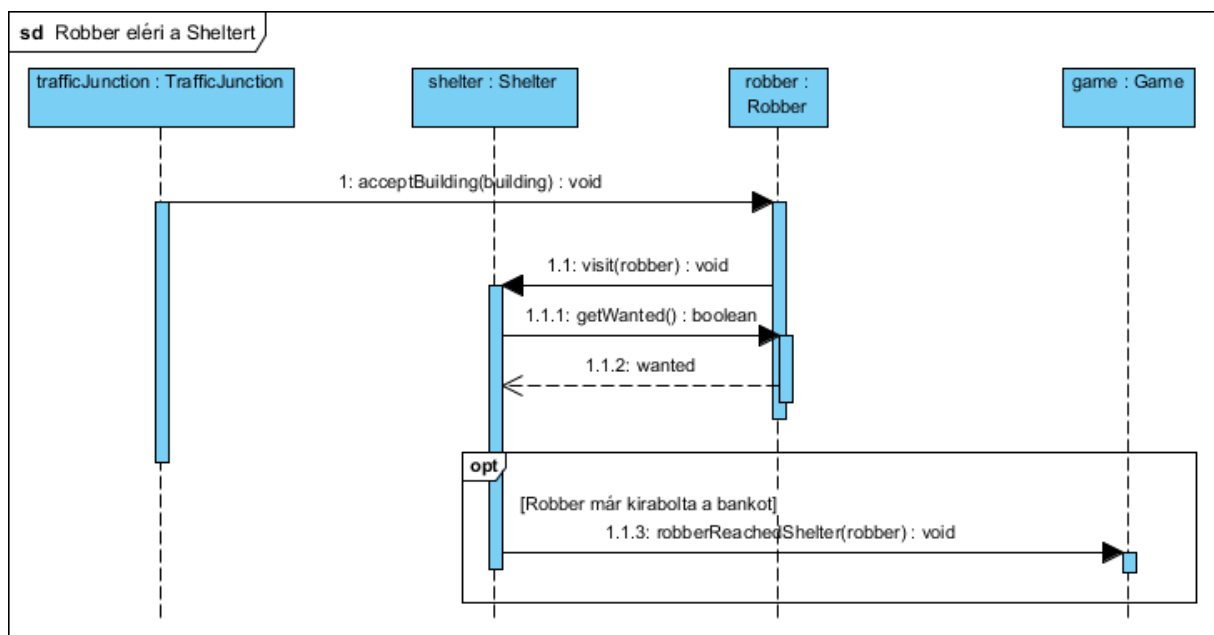
5. teszteset diagramja

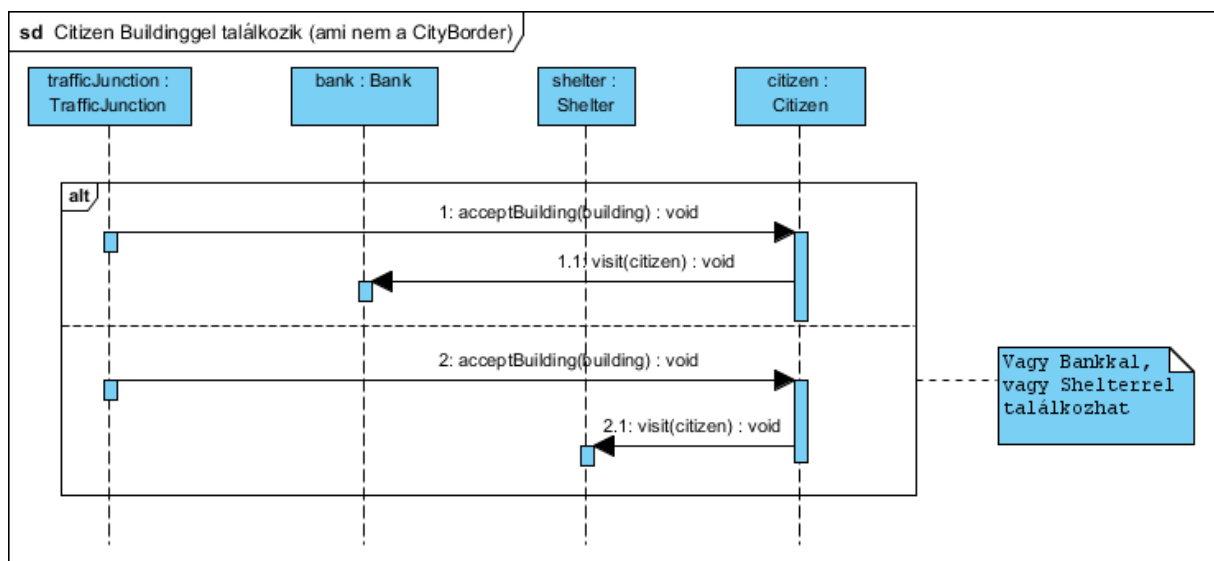
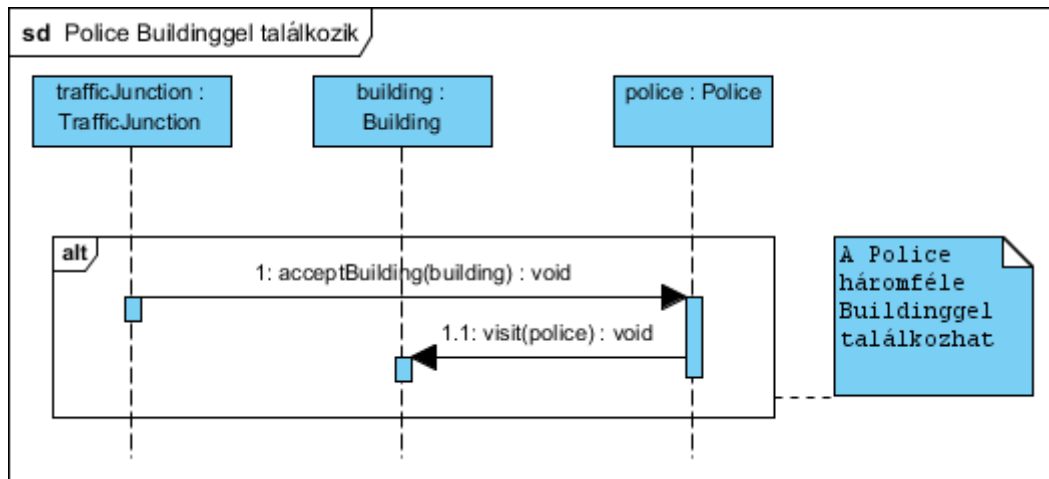


6. teszteset diagramja

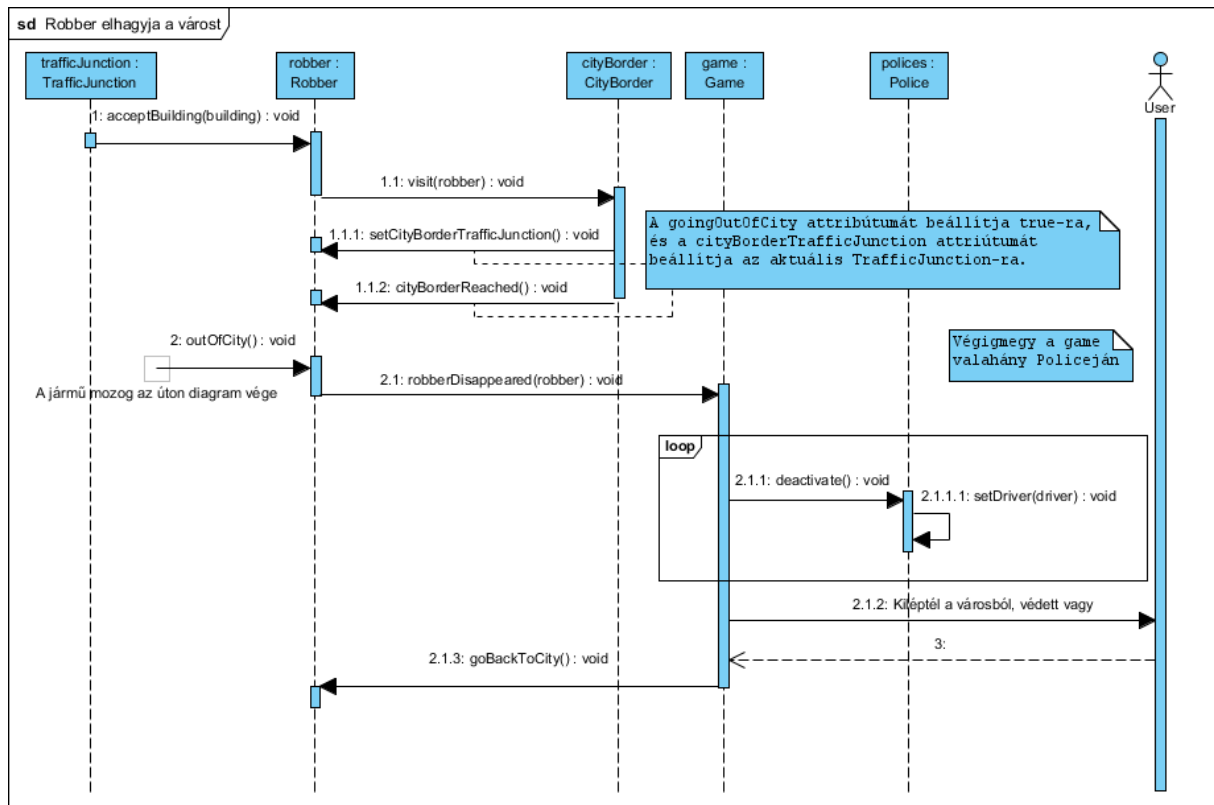


Hasonló esetek:

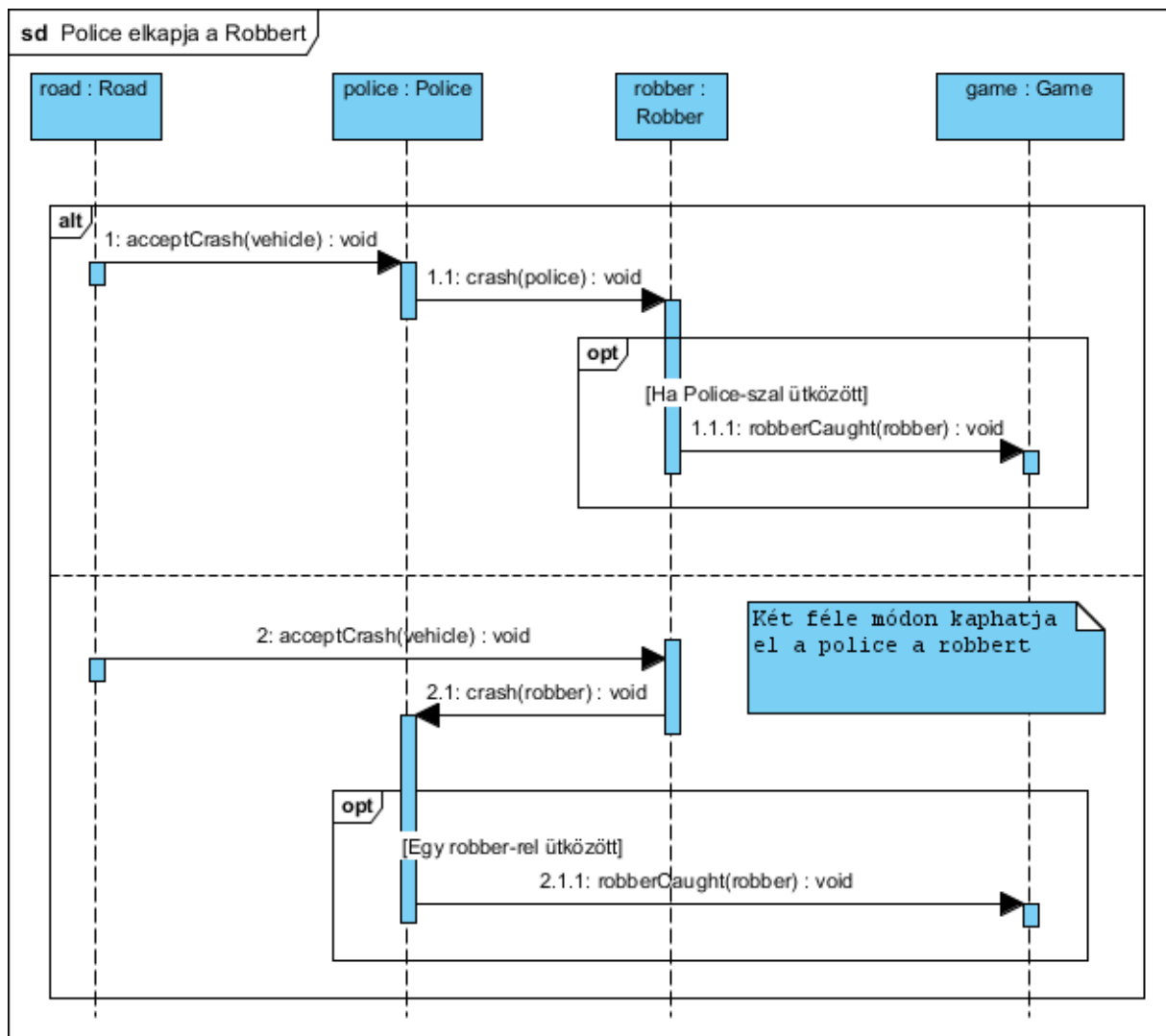




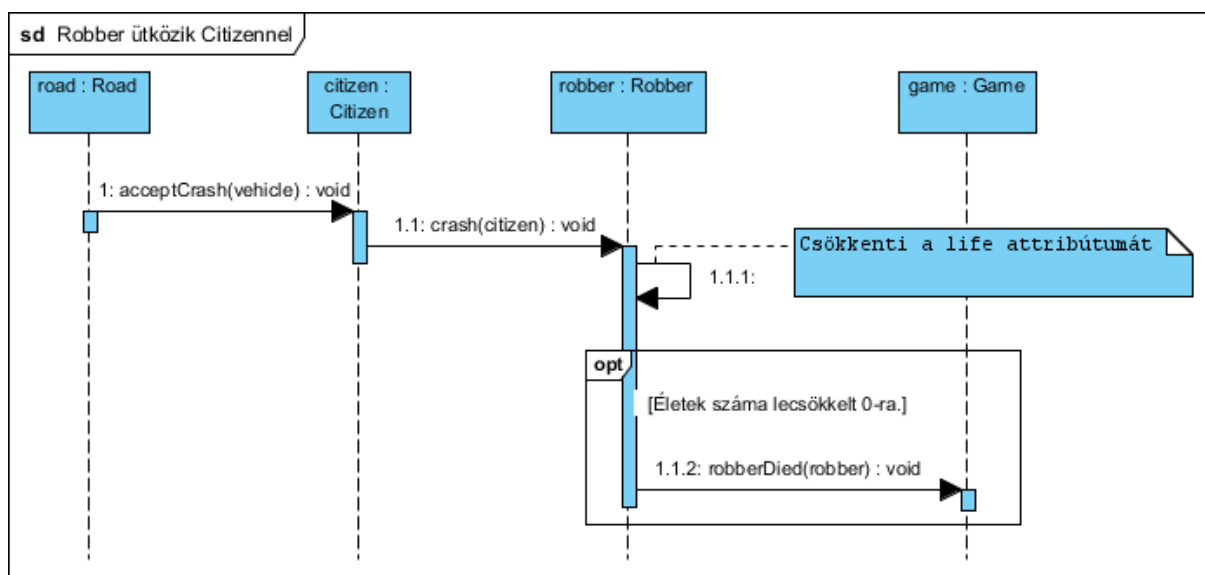
7. teszteset diagramja



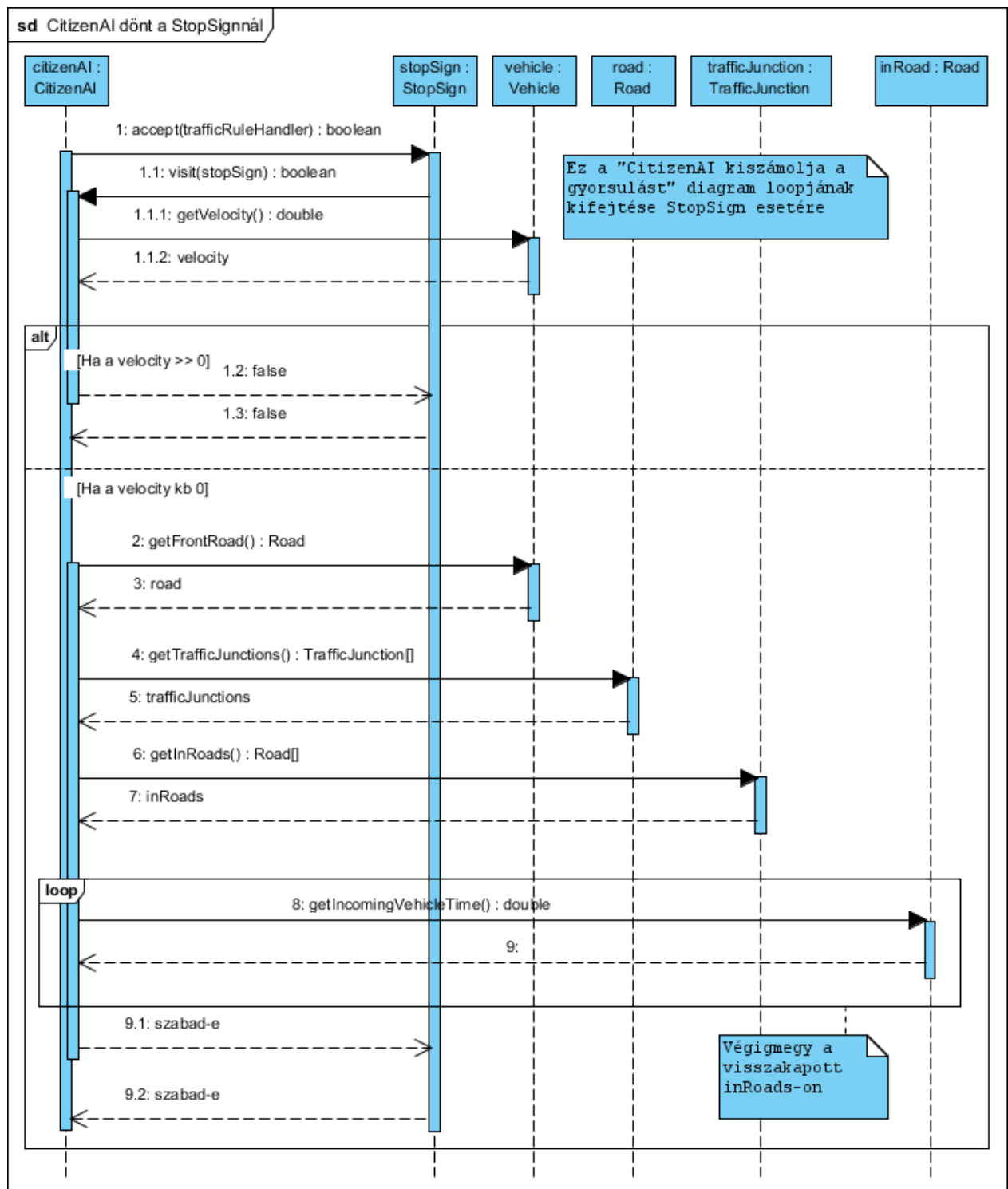
8. teszteset diagramja



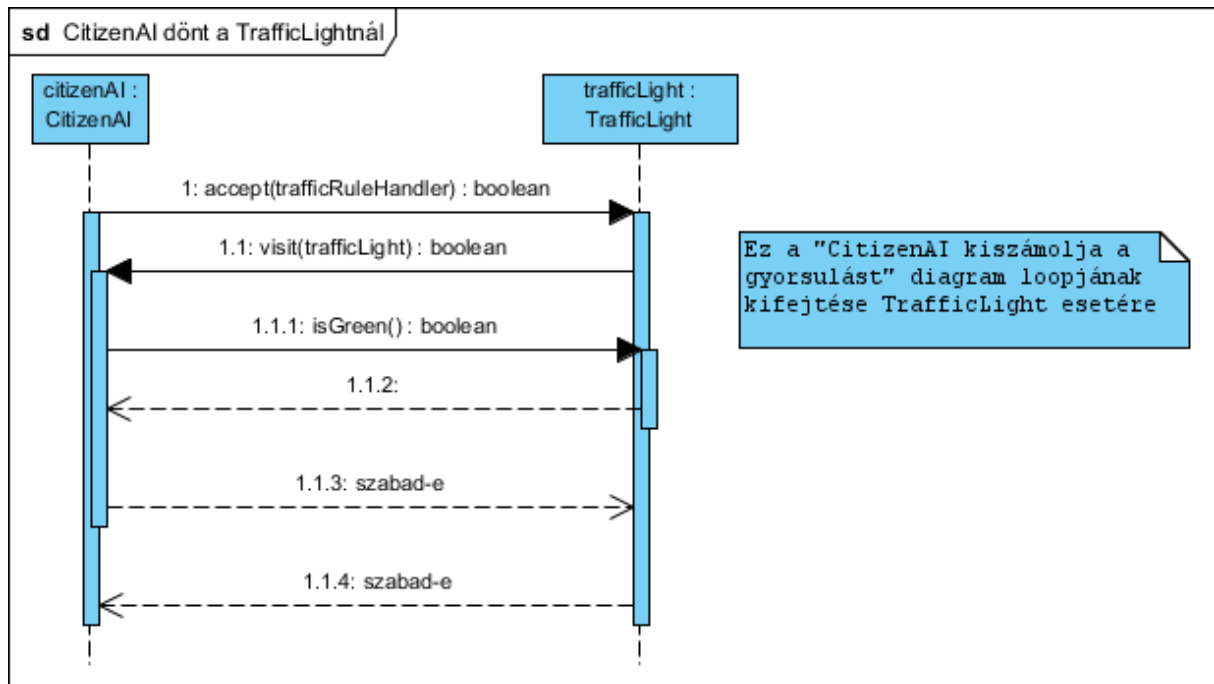
9. teszteset diagramja



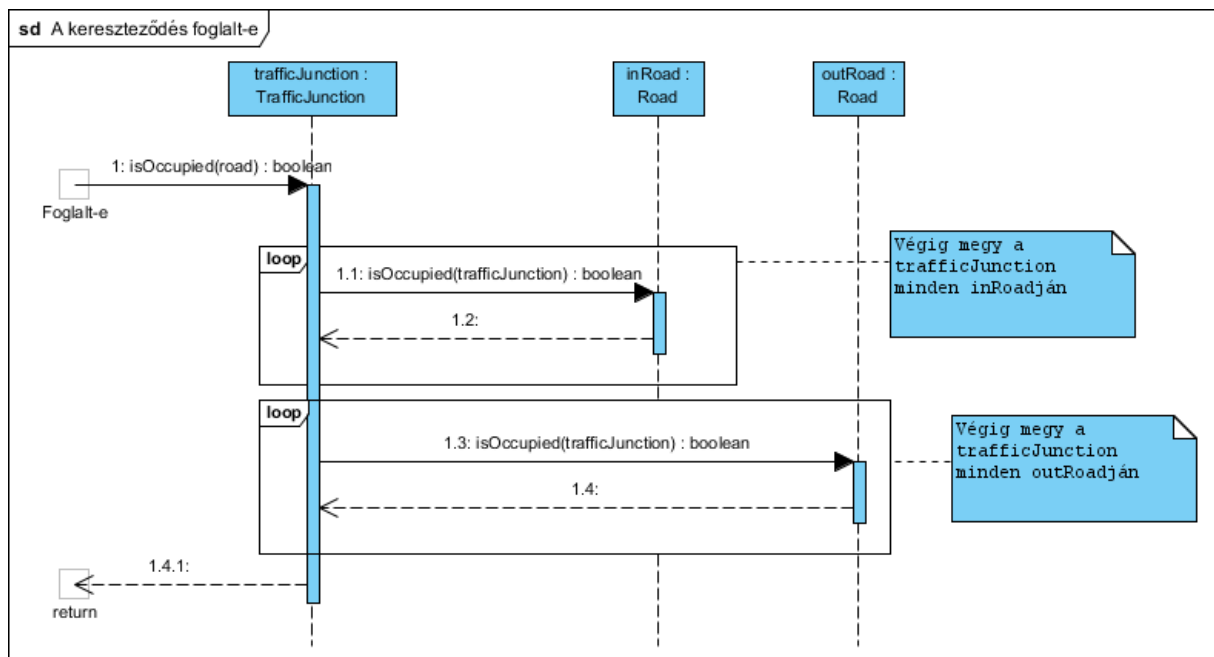
10. teszteset diagramja



Hasonló esetek:



11. tesztet diagramja



5.5 Napló

Kezdet	Idő	Résztevők	Leírás
2010.03.06. 16:00	2 óra	Kovács, Koczka	Szkeleton tervezése
2010.03.07. 12:00	45 perc	Mikó, Angeli	Részletes Use-Case diagramok terv és elkészítés
2010.03.07. 14:00	1 óra	Kovács, Koczka	Szkeleton és kezelői felületének tervezése
2010.03.07. 15:00	30 perc	Koczka	Kezelői felület segédosztály (SkeletonHelper) tervezete
2010.03.08. 12:30	1 óra	Angeli, Mikó, Kovács, Koczka	Tesztesetek összeírása. Megbeszélés.
2010.03.08. 20:00	2 óra	Angeli, Mikó	Dokumentáció készítése
2010.03.09. 18:00	1,5 óra	Mikó	Usecase jegyzőkönyv kibővítése
2010.03.09. 18:00	2 óra	Angeli	Dokumentáció készítése
2010.03.10. 20:00	1,5 óra	Kovács	Dokumentáció ellenőrzése, kiegészítése
2010.03.10. 21:45	30 perc	Mikó	Dokumentum formázása
2010.03.10. 22:30	30 perc	Koczka	Új diagramok berakása, revízió

6. Szkeleton beadás

6.1 Fordítási és futtatási útmutató

6.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
src/GrandTheftBank/Bank.java	1036 b	2010.03.15. 13:03:22	A bankot megvalósító osztály
src/GrandTheftBank/Building.java	1915 b	2010.03.15. 13:03:22	Az épületek alaposztálya
src/GrandTheftBank/Citizen.java	2700 b	2010.03.15. 13:03:22	A városi autót megvalósító osztály
src/GrandTheftBank/CitizenAI.java	5204 b	2010.03.15. 13:03:22	A városi autót irányító algoritmust megvalósító osztály
src/GrandTheftBank/CityBorder.java	1403 b	2010.03.15. 13:03:22	A városhatárt megvalósító osztály
src/GrandTheftBank/DriverAdapter.java	1648 b	2010.03.15. 13:03:22	Segéd adapter az IDriver interfésznek
src/GrandTheftBank/Game.java	19150 b	2010.03.15. 13:03:22	A játék irányítását megvalósító osztály
src/GrandTheftBank/IDriver.java	714 b	2010.03.15. 13:03:22	Az autót irányító algoritmus interfésze
src/GrandTheftBank/ITick.java	714 b	2010.03.15. 13:03:22	Az időt érzékelő elem interfésze
src/GrandTheftBank/ITrafficRuleHandler.java	1066 b	2010.03.15. 13:03:22	A közlekedési szabályt érzékelő elem interfésze
src/GrandTheftBank/Main.java	2577 b	2010.03.15. 13:03:23	A program futását lekezelő osztály
src/GrandTheftBank/Map.java	7357 b	2010.03.15. 13:03:23	A játék térképét tartalmazó osztály
src/GrandTheftBank/Police.java	4516 b	2010.03.15. 13:03:23	A rendőrautót megvalósító osztály
src/GrandTheftBank/Road.java	8387 b	2010.03.15. 13:03:23	Az utat megvalósító osztály
src/GrandTheftBank/Robber.java	5322 b	2010.03.15. 13:03:23	A rabló autóját megvalósító osztály
src/GrandTheftBank/Shelter.java	1011 b	2010.03.15. 13:03:23	A rejtékhelyet megvalósító osztály
src/GrandTheftBank/SkeletonHelper.java	17018 b	2010.03.15. 13:03:23	Skeleton tesztelését megkönnyítő segédosztály
src/GrandTheftBank/StopSign.java	638 b	2010.03.15. 13:03:23	A stop táblát megvalósító osztály
src/GrandTheftBank/Timer.java	1899 b	2010.03.15. 13:03:23	Az időzítőt megvalósító osztály
src/GrandTheftBank/TrafficJunction.java	4444 b	2010.03.15. 13:03:23	Az útkereszteződést megvalósító osztály
src/GrandTheftBank/TrafficLight.java	1761 b	2010.03.15. 13:03:23	A közlekedési lámpát megvalósító osztály
src/GrandTheftBank/TrafficSign.java	494 b	2010.03.15. 13:03:23	A közlekedési szabályok ösosztálya

src/GrandTheftBank/Vehicle.java	6926 b	2010.03.15. 13:03:23	Az autót megvalósító ősosztály
src/GrandTheftBank/VehicleOnRoad.java	1591 b	2010.03.15. 13:03:23	Tároló osztály az autók úton való elhelyezésére
src/GrandTheftBank/VehicleType.java	1726 b	2010.03.15. 13:03:23	Az autómodelleket megvalósító osztály
test0.txt	65 b	2010.03.15. 13:03:23	Teszteset pályája, belső azonosítója: 0
test1.txt	66 b	2010.03.15. 13:03:23	Teszteset pályája, belső azonosítója: 1
test2.txt	52 b	2010.03.15. 13:03:23	Teszteset pályája, belső azonosítója: 2
test6.txt	50 b	2010.03.15. 13:03:23	Teszteset pályája, belső azonosítója: 6
test7.txt	34 b	2010.03.15. 13:03:23	Teszteset pályája, belső azonosítója: 7
test11.txt	80 b	2010.03.15. 13:03:23	Teszteset pályája, belső azonosítója: 11
gtb.bat	60 b	2010.03.15. 13:03:23	Lefordítja a projektet, elkészíti a dokumentációt és elindítja a programot
gtb_compile.bat	58 b	2010.03.15. 13:03:23	Lefordítja a projektet
gtb_javadoc.bat	92 b	2010.03.15. 13:03:23	Elkészíti a dokumentációt
gtb_run.bat	44 b	2010.03.15. 13:03:23	Elindítja a programot

6.1.2 Fordítás

A fordítás megkönnyítése érdekében egy batch fájlt hoztunk létre. Így a projekt főkönyvtárában a parancssorban csak ki kell adni a "gtb_compile" parancsot.

(A projekt főkönyvtára az a könyvtár, ahol megtalálható a "gtb_compile.bat" fájl és a "src" alkönyvtár)

Ez a következő parancsokat hajtja végre:

- 1) `mkdir build`
- 2) `javac ./src/GrandTheftBank/*.java -d ./build/`

Azaz

- 1) létrehoz egy "build" nevű könyvtárat
- 2) lefordítja a "src/GrandTheftBank/" könyvtárban lévő .java kiterjesztésű fájlokat és a "build/" könyvtárba menti az elkészült .class kiterjesztésű fájlokat.

Nagyon fontos, hogy ahhoz, hogy a "javac" parancsot használni tudjuk a PATH környezeti változóban szerepelnie kell a JDK bináris könyvtárának.

Amennyiben a "javac" parancs beírására ismeretlen parancsra panaszkodik a futtatandó környezet, kérjük adják hozzá a PATH környezeti változóhoz a JDK aktuális könyvtárát a következő paranccsal:

```
set PATH=%PATH%;C:\Program Files\Java\jdk1.6.0_18\bin\
```

Fontos, hogy a "C:\Program Files\Java\jdk1.6.0_18\bin\" rész helyett a JDK aktuális könyvtárát kell beírni, ami futtatandó környezetként változhat!

(Általános esetben csak a 1.6.0_18 verziószámot kell kicserélni a gépen lévő JDK aktuális verziószámára. Amennyiben a JDK megtalálása közben problémába ütközik kérje a gép rendszergazdájának segítségét!)

A PATH beállítása után újra kell indítani a parancssort, mert csak akkor lép érvénybe a változás!

Dokumentáció létrehozása: a fordítás sikeressége után adja ki a *"gtb_javadoc"* parancsot.

Ez a következő parancsokat hajtja végre:

- 1) *javadoc ./src/GrandTheftBank/*.java -d ./javadoc/ -charset utf-8*
- 2) *start ./javadoc/index.html*

Azaz

- 1) legenerálja a dokumentációt a "src/GrandTheftBank/" mappában lévő .java kiterjesztésű fájlokból a "javadoc/" könyvtárba
- 2) megnyitja a dokumentációt a gép alapértelmezett böngészőjében

6.1.3 Futtatás

A futtatás megkönnyítése érdekében egy batch fájlt hoztunk létre. Így a projekt főkönyvtárában a parancssorban csak ki kell adni a *"gtb_run"* parancsot.

Ez a következő parancsot hajtja végre: *"java -classpath ./build/ GrandTheftBank.Main"*, azaz elindítja a "build/"-ból kiindulva a GrandTheftBank package Main classjának belépési pontját.

Figyelem! A szkeleton konzolos programja esetenként hosszú sorokat ír ki, ezért javasoljuk, hogy a parancssorban minél nagyobb oszlopszélesség legyen beállítva. A kiírt kimenetek mentődnek a "out-{teszteset száma}.txt" fájlokba is, így szükség esetén bármikor visszanézhetők.

A szkeleton konzolos programjának kezelői felületének leírását a dokumentáció 5.3-as pontjában találja meg!

6.2 Értékelés

A közös projekt a kezdetektől fogja megfelelően halad, minden az ütemtervhez igazodva elkészült és megfelelő minőségűnek bizonyult.

A csapat stabilan tudja nyújtani az elvárásoknak megfelelő eredményeket.

A projekt első felében nagy figyelmet fordítottunk a modell részletes megalkotására, hisz attól alapjaiban függ a projekt sikeressége, ezért próbáltunk minél könnyebben bővíthető és átlátható modellt alkotni, hogy ezzel minél könnyebbé és gördülékenyebbé tegyük a jövőbeni munkát.

Az optimalizált munkavégzés végett dinamikusan osztjuk fel egymás között a munkafolyamatokat. Mindenki olyan részalmazát végzi el a munkának, amit a leggyorsabban és legjobban ő tud elvégezni.

Legmegfelelőbbnek azt a megoldást találtuk, hogy 2 fős csoportokra válunk szét: Koczka és Kovács a tervezésben és a váz létrehozásában, míg Angeli és Mikó a dokumentációban és a váz kiegészítésében vállal nagyobb szerepet. Így a kommunikáció 2 fő között még nem jelent jelentős felesleges összehangolási idővesztést, de folyamatos ellenőrzést biztosít a közösen megosztott munkafolyamaton és kiküszöböli az egyoldalú rossz döntések veszélyét. A projektet érintő nagyobb volumenű kérdésekben természetesen a csapat minden tagja részt vesz értekezletek formájában, így biztosítva a közös és mindenki számára elfogadott menetrend létrejöttét.

A menetközben felmerülő problémák általában rövidéletűek voltak: a megfelelő csoporton belül gyorsan megoldásra került a sor, illetve ha nagyobb problémába ütköztünk, akkor a következő online egyeztetéskor gyors megoldás született rá.

Tag neve	Munka órában	Munka százalékban
Angeli Róbert	35,25 óra	20,6%
Koczka Tamás	48,5 óra	28,4%
Kovács Balázs	48,5 óra	28,4%
Mikó Norbert	38,75 óra	22,6%

6.3 Napló

Kezdet	Idő	Résztvevők	Leírás
2010.03.11. 20:00	1 óra	Angeli Koczka Kovács Mikó	Értekezlet. Döntés: - Koczka és Kovács leimplementálja a SkeletonHelpert - Kovács és Mikó elkészíti a teszteseket - Angeli ledokumentálja a skeleton forráskódját
2010.03.12. 12:00	2 óra	Koczka	SkeletonHelper implementálása
2010.03.12. 16:00	3 óra	Kovács	SkeletonHelper majd tesztesetek (0-5) implementálása
2010.03.12. 19:00	4 óra	Mikó	Tesztesetek (6-11) implementálása
2010.03.13. 10:00	3,5 óra	Kovács	A program vázának megírása, SkeletonHelper hívások beépítése a kódba
2010.03.13. 15:00	3,5 óra	Angeli	Osztályok dokumentálása forráskódban
2010.03.13. 16:00	2 óra	Mikó	Osztályok dokumentálása forráskódban
2010.03.13. 14:00	2,5 óra	Koczka	Forráskód átnézése, kóddokumentáció kiegészítése
2010.03.14. 10:00	2 óra	Koczka	Fordító és futtató szkriptek megírása, fájlok összesítése Értékelés megírása
2010.03.15. 19:00	1 óra	Mikó	Dokumentáció összeillesztése, formázása

7. Prototípus koncepciója

7.1 Prototípus interface-definíciója

7.1.1 Az interfész általános leírása

A prototípusnak karakteres felületet alakítunk ki, amelyen egyszerűen ellenőrizhető a program algoritmusainak működése. Egy egyszerűsített parancsokból álló bemeneti nyelvet alkalmazunk, amelyen egy parancs és paramétereinek megadására a prototípus végrehajtja a szükséges műveleteket és visszatér az eredményükkel. Így a parancsok egymás után való megadásával felépíthető minden tesztet, és szemügyre vehető, hogy a prototípus az elvárt működést hajtja végre.

Mivel a prototípus lényege, hogy a program helyes működését folyamatosan tesztelhessük, ezért a programot teljesen automatikus tesztelés megvalósítására is felkészítjük. A prototípusnak megadható parancsokat így fájlból is lehet beolvasni, illetve az eredményt fájlba mentődik, így külső programmal könnyedén ellenőrizhető a modell helyes működése az elvárt és generált eredmények összehasonlításával, ami ugyancsak egyszerű szöveges formában történik.

7.1.2 Bemeneti nyelv

A parancsokat a standard bemenetről, illetve fájlból olvassa be a program. Minden parancsot egy sorvége karakter zár le.

tick [num]

Leírás: Lépteti a játékot num db lépéssel.

Opciók: ha nem írunk be num-ot, akkor eggyel léptet.

loadMap [file.txt]

Leírás: Betölti a megadott térképet. (TrafficJunction-ok, Road-ok, Building-ek, TrafficSign-ok)

Opciók: -

showMap

Leírás: Kiírja minden járműről, hogy melyik úton van, milyen pozícióban és milyen sebességgel mozog (a formátumot lásd a Kimeneti nyelv című fejezetben).

Opciók: -

loadCommands [file.txt]

Leírás: Betölti a megadott fájlból a parancsokat, és végrehajtja azokat.

Opciók: -

setDeltaTime [num]

Leírás: Beállítja azt az időt msec-ben (num), ami egy 'tick' meghívása után "eltelik". (a num egy egész szám)

Opciók: -

playerSetDirection [num]

Leírás: Beállítja, hogy ha a Player egy kereszteződéshez ér, merre forduljon majd el. (a megadott Road tömbből a num-adikat választja, a num egy egész szám)

Opciók: -

playerSetAcc [num]

Leírás: Beállítja a Player gyorsulását. (a num egy valós szám)

Opciók: -

setRand [num]

Leírás: Ha num 0, akkor kikapcsolja a program véletlen viselkedését, ha 1, akkor bekapcsolja.

Opciók: -

addVehicle [Vehicle kódja] [Road indexe] [Road-on lévő position 0-tól 1-ig]

Leírás: Hozzáad a megadott indexű Road-hoz a megadott position-re egy megadott típusú Vehicle-t.

Opciók: a Vehicle kódja a következő lehet: c Citizen, r Robber, p Police

quit

Leírás: Befejezi a tesztelést.

Opciók: -

Pályaképet megadó objektumok nyelvtana (a sorok leírva sorban):

[TrafficJunction-ök száma]

[1. TrafficJunction x koordinátája] [1. TrafficJunction y koordinátája]; [1. TrafficJunction 1. Buildingjének kódja] [1. TrafficJunction 2. Buildingjének kódja] stb

<< *az előző sor minden TrafficJunction-re* >>

[Road-ok száma]

[1. Road kiinduló TrafficJunctionének indexe] [1. Road végződő TrafficJunctionének indexe]; [1. Road 1. TrafficSignjának kódja] [1. Road 2. TrafficSignjának kódja] stb

<< *az előző sor minden Road-ra* >>

Building-kódok lehetnek:

b - Bank

s - Shelter

c - CityBorder

TrafficSign-kódok lehetnek:

s - StopSign

l - TrafficLight

7.1.3 Kimeneti nyelv

A standard kimeneten jelennek meg a program történései, illetve ezeket fájlba is kiírja a program. A történések formátuma a következő:

[#ObjektumSzama]:[ObjektumTipusa] has [Cselekmény].

Pld.

#1:Citizen has moved on #2:Road to x position.

#4:Robber has crashed with #3:Citizen.

Illetve a showMap hatására kiíródik az aktuális járművek állapota:

[VehicleSzama]:[VehicleTipusa] is on [RoadSzama]:Road at [Vehicle pozíciója] position, moving with [sebesség] velocity.

A tesztelőről feltételezzük, hogy nem ír be hibás, nem létező parancsokat, ezért a hibaüzeneteknek nincs külön fajtájuk, a program csak jelzi a hibát (konzulenssel egyeztetve).

7.2 Összes részletes use-case

Use-case neve	Teszt inicializálása
Rövid leírás	A tesztprogram kezdeti állapotának megadása, ez után kezdődhet a tesztelés
Aktorok	Tesztelő
Forgatókönyv	A Map betöltése, a járművek elhelyezése a Map-re.

Use-case neve	Teszt indítása
Rövid leírás	A tesztelés megkezdése (az inicializálás után).
Aktorok	Tesztelő
Forgatókönyv	-

Use-case neve	Teszt befejezése
Rövid leírás	A tesztelés befejezése.
Aktorok	Tesztelő
Forgatókönyv	-

Use-case neve	Teszt eredményének ellenőrzése
Rövid leírás	A kimenet összehasonlítása a várt kimenettel.
Aktorok	Tesztelő
Forgatókönyv	Egy script a teszt lefutását követően összehasonlítja a kimeneti fájlt a referencia fájllal és kiírja az eredményt.

Use-case neve	Robber irányítása
Rövid leírás	Parancsok segítségével irányíthatjuk a Robbert.
Aktorok	Tesztelő
Forgatókönyv	A Robber-t a playerSetDirection, playerSetAcc parancsokkal irányíthatjuk

Use-case neve	Véletlen viselkedés állítása.
Rövid leírás	A program véletlen viselkedésének ki/be kapcsolása.
Aktorok	Tesztelő
Forgatókönyv	A setRand paranccsal állítható a véletlen viselkedés.

7.3 Tesztelési terv

Egy script a teszt lefutását követően összehasonlítja a kimeneti fájlt a referencia fájlal és kiírja az eredményt. Ehhez egy, a szövegfájlok tartalmának egyezőségét vizsgáló programot használunk.

Teszt-eset neve	1) A Robber megnyeri a játékot
Rövid leírás	A Robber eléri a Bank épületet (így kirabolja), aminek hatására a Police üldözni kezdi őt. A Robber eléri a Shelter épületet, aminek hatására megnyeri a játékot.
Teszt célja	Teszteli: - A bankrablás és az üldözés létrejöttének tesztelése - A búvóhely elérésének és a játék megnyerésének tesztelése

Teszt-eset neve	2) A Robber elveszti a játékot
Rövid leírás	A Robber eléri a Bank épületet (így kirabolja), aminek hatására a Police üldözni kezdi őt. A Police ütközik a Robberrel, aminek hatására a Robber elveszti a játékot.
Teszt célja	Teszteli: - A bankrablás és az üldözés létrejöttének tesztelése - Az ütközés és a játék elvesztésének tesztelése

Teszt-eset neve	3) Citizen átmegy egy piros lámpás kereszteződésen
Rövid leírás	Egy jármű (Citizen, minden másik fajta is ugyanígy viselkedik, kivéve, hogy nem veszi figyelembe a táblákat) mozog az úton, meglát egy lámpát, ami piros, lelassít, majd ha zöld akkor behajt a kereszteződésbe (nem megy át rajta)
Teszt célja	Teszteli: - A járművek közlekedési szabályok betartását - A lámpa megfelelően vált-e piros és zöld állapot között

Teszt-eset neve	4) Citizen átmegy átmegy egy stoptáblás kereszteződésen
Rövid leírás	A Citizen stoptáblához érkezik és érzékeli, hogy a kereszteződésben jön egy másik autó, ezért megvárja, amíg az áthalad, addig ő nem megy tovább
Teszt célja	- A Citizen megfelelő döntésének tesztelése a Stop táblánál - Teszteli, hogy a Citizen megfelelően érzékeli-e más autók kereszteződéshez közeledését

Teszt-eset neve	5) Citizen kereszteződésnél megvárja amíg egy másik autó áthalad rajta
------------------------	--

Rövid leírás	Egy hármas kereszteződésben egyik útról a másikra halad át a tesztelő (Robber), ennek hatására a harmadik úton lévő Citizen érzékeli, hogy foglalt a kereszteződés, ezért vár és nem megy bele a kereszteződésbe
Teszt célja	Teszteli: - A kereszteződés megfelelően rakja át egyik útról a másikra az autókat - A kereszteződés megfelelően dönti el foglaltságát és ezt a Citizen is megfelelően érzékeli

Teszt-eset neve	6) Citizen nem ütközik az előtte lévő autóval
Rövid leírás	Az úton a tesztelő autója (Robber), majd utána egy Citizen van. A tesztelő lassít, majd megáll és ennek hatására a mögötte lévő Citizen is lelassít, majd megáll, hogy az ütközést elkerülje
Teszt célja	Annak tesztelése, hogy a Citizen nem ütközik, ha az előtte lévő autó akármennyire is lelassít vagy megáll

Teszt-eset neve	7) Citizen ütközik a Robberrel
Rövid leírás	Az úton a tesztelő autója (Robber), előtte egy Citizen van. A tesztelő gyorsít, majd ütközik a Citizennel. A Citizen így tönkremegy és eltűnik az útról, a Robber életet veszít. A Robber halad tovább az úton.
Teszt célja	A Robber és Citizen ütközésének megfelelő lekezelése

Teszt-eset neve	8) Citizen elhagyja a várost
Rövid leírás	Egy Citizen kimegy a városból, aminek hatására a város egy másik pontján egy másik Citizen jön be
Teszt célja	Teszteli: - Autók város elhagyásának helyes érzékelése - Új autók megfelelő lerakásának tesztelése

Teszt-eset neve	9) Robber kimegy a városból, majd visszajön
Rövid leírás	A tesztelő (Robber) eléri a CityBorder épületet, ennek hatására kilép a városból. Amikor kiért, akkor a rendőrök nem üldözik tovább. A Robber újra elkezd gyorsulni, ezzel visszatér a városba ugyanazon a ponton és a rendőrök megint elkezdik őt üldözni
Teszt célja	Annak tesztelése, hogy a modell megfelelően kezeli-e le, amikor a Robber elhagyja a várost, ezzel a rendőrök figyelmét elveszti, majd visszatérve visszakapja azt

Teszt-eset neve	10) Terhelésteszt
Rövid leírás	Különböző szituációkban kipróbálja a programot és megméri, hogy mennyi idő alatt hajtotta végre a műveleteket
Teszt célja	A program megfelelően gyors működésének tesztelése, az algoritmusok komplexitásának ellenőrzése

7.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása

MapEditor:

Pálya szerkeszthető vele: TrafficJunction-ök, Road-ok, Building-ek, TrafficSign-ok elhelyezése. Fájlbba mentés a fent megadott formátumban.

ProtoHelper (a proto részeként beépülve):

Ez értelmezi a parancsokat, amiket beírnak. Kiírja a megadott kimenet formátumban a tesztelés történéseit.

Szövegfájl tartalmát összehasonlító program:

Összehasonlítja 2 szöveges fájlt és kiírja, hogy egyeznek-e. Pl. DiffUtils cmp.exe.

7.5 Napló

Kezdet	Idő	Résztvevők	Leírás
2010.03.20. 16:00	2 óra	Koczka Kovács	Tesztesetek és beviteli parancsok megtervezése
2010.03.23. 18:00	2 óra	Kovács	Use-casek, tesztelési nyelv megírása
2010.03.24. 14:00	2 óra	Koczka	Tesztelési terv és interfész leírás megírása
2010.03.24. 18:30	1 óra	Angeli	Dokumentáció készítés
2010.03.24. 19:30	2 óra	Mikó	Dokumentáció összeillesztése, formázása

8. Részletes tervek

8.1 Osztályok és metódusok tervei.

8.1.1 Bank

- **Felelősség**

Ha rabló a bankot tartalmazó mezőre lép akkor jelez a Game-nek hogy kirabolták a bankot, valamint ha a rabló még nem volt körözve akkor beállítja annak körözöttségét is. Ha más járművek látogatják meg a bankot akkor nem tesz semmit, mivel nekik nincs hatásuk a bankra, se banknak rájuk.

- **Ősosztályok**

Building

- **Interfészek**

nincs

- **Attribútumok**

Game game - Egy referencia a Game-re. Azért van rá szükség, hogy ezen keresztül tudjon jelezni hogy a bankot kirabolták, azáltal hogy meghívja a game.Bankrobbed() függvényt.

- **Metódusok**

+ void visit(Robber robber) - Ha egy Robber érkezik az épülethez ez hívódik meg. A metódus először megvizsgálja hogy a rablót körözik-e azaz hogy rabolt-e már bankot (robber.getWanted()) , és ha még nem akkor beállítja annak körözöttségét (robber.setWanted(true)), majd ezek után jelez a Game-nek a game.Bankrobbed() metóduson keresztül hogy a bankot kirabolták. Ha már a rablót körözték akkor nem tesz semmit.

+ void visit(Citizen citizen) - Ha egy Citizen érkezik az épülethez ez hívódik meg. Ekkor nem történik semmi. Ezt a különböző épületek, melyeknek a Building az őosztálya különböző módon kezelik le. Jelen esetben nincs rá szükség, mivel a játék szempontjából irreleváns, hogy ha egy bankot egy citizen meglátogat.

+ void visit(Police police) - Ha egy Police érkezik az épülethez ez hívódik meg. Ekkor nem történik semmi. Ezt a különböző épületek, melyeknek a Building az őosztálya különböző módon kezelik le. Jelen esetben nincs rá szükség, mivel a játék szempontjából irreleváns, hogy ha egy bankot egy police meglátogat.

8.1.2 Building

- **Felelősség**

A Mapen elhelyezett kereszteződésekénél lévő épületek. (Shelter, Bank, CityBorder, stb.) Amikor egy jármű ehhez a kereszteződéshez ér, az épületek az ideérkező jármű típusától valamint magától az épület típusától függően cselekednek.

- **Ősosztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

Game game - Egy referencia a Game-re. Azért van rá szükség, hogy ezen keresztül tudjon jelezni hogy a bankot kirabolták, azáltal hogy meghívja a game.Bankrobbed() függvényt.

- **Metódusok**

+ void visit(Robber robber) - Ha egy Robber érkezik az épülethez ez hívódik meg. Ezt a különböző épületek, melyeknek a Building az ősosztálya különböző módon kezelik le, ezért ennek a függvénynek a törzse üres, majd a származtatott osztályokban kerül implementálásra

+ void visit(Citizen citizen) - Ha egy Citizen érkezik az épülethez ez hívódik meg. Ezt a különböző épületek, melyeknek a Building az ősosztálya különböző módon kezelik le, ezért ennek a függvénynek a törzse üres, majd a származtatott osztályokban kerül implementálásra

+ void visit(Police police) - Ha egy Police érkezik az épülethez ez hívódik meg. Ezt a különböző épületek, melyeknek a Building az ősosztálya különböző módon kezelik le, ezért ennek a függvénynek a törzse üres, majd a származtatott osztályokban kerül implementálásra

8.1.3 Bunny

- **Felelősség**

A Mapen elhelyezett kereszteződésekénél lévő épületek. (Shelter, Bank, CityBorder, stb.) Amikor egy jármű ehhez a kereszteződéshez ér, az épületek az ideérkező jármű típusától valamint magától az épület típusától függően cselekednek.

- **Ősosztályok**

Building

- **Interfészek**

ITick

- **Attribútumok**

Game game - Egy referencia a Game-re. Azért van rá szükség, hogy ezen keresztül tudjon jelezni hogy a bankot kirabolták, azáltal hogy meghívja a game.Bankrobbed() függvényt.

- int waitTime - Ennyi ideig érvényes a védettség
- int startTime - Az időpillanat amikortól a védettség érvényes
- boolean active - Ha a nyulat elütjük akkor aktív állapotba kerül.
- Robber robber - A rabló aki elütötte a bunnyt
- TrafficJunction tj - A tj ahol elütöttük a bunnyt.

- **Metódusok**

+ void visit(Robber robber) - Ha egy Robber érkezik a Bunnyhoz ez hívódik meg. Ha a Bunny nem aktív beállítja aktívra és meghívja a deactivatePolices függvényt.

+ void doTick(int time, int deltaTime) - Ha nem aktív akkor a startTimert az aktuális időre állítjuk. Ha lejárt a védettségi idő akkor aktiváljuk a rendőrokat. És eltávolítjuk a bunnyt a TJ-ből, levesszük a térképről és levesszük a tickerekről. Majd az active-t false-ra állítjuk.

8.1.4 Citizen

- **Felelősség**

Városi autó, ami nem a rendőr és nem a bankrabló autója. Nem megy neki az előtte haladó járműnek, lelassít ha ő is lelassít, valamint betartja a közlekedési szabályokat.

Felelőssége többek között az ütközések kezelése valamint a saját sofőrével (driver-rel) való kommunikáció megvalósítása.

- **Ősosztályok**

Vehicle

- **Interfészek**

nincs

- **Attribútumok**

Game game – Egy referencia a Game-re. Szükség van rá ha a citizen jelezni szeretné a Game-nek hogy elhagyta a várost. Ezt a game.citizenLeftCity(this) metódushívással teszi meg az adott helyzetben.

- **Metódusok**

+ Road getFrontRoad() - A metódust Vehicle ősosztálytól örököljük. A függvény visszaadja azt a Road-ot, amelyiken az autó eleje van. Ezt a roads.get(0) metódushívással éri el, mely visszaadja a roads tömb első elemét.

+ Road getDirection(Road[] roads) - A metódust Vehicle ősosztálytól örököljük. A függvény segítségével kiválasztjuk a továbbhaladási irányt a következő módon. Először megvizsgáljuk hogy az adott jármű kifelé halad-e a városból, azaz hogy a GoingOutOfCity paramétere true-e. Ha már elkezdte elhagyni a várost akkor null-t adunk vissza, ha viszont nem akkor megkérdezzük a vezetőt, azaz meghívjuk a

driver.chooseDirection(roads) függvényt. Ebben az esetben az ebből származó visszatérési érték lesz a függvény visszatérési értéke. Ez a visszatérési érték egy road, az a road amit a driver megfelelőnek talált a továbbhaladásra.

+ void addFrontRoad(Road road) - A metódust Vehicle őssosztálytól örököljük. A függvény segítségével hozzáadjuk a Road[] listához azt az utat, amire az autó eleje rátért. Ezt a roads.add(0, road) metódushívással érjük el, mely a roads tömb első helyére teszi a paraméterként átadott road-ot.

+ void removeLastRoad() - A metódust Vehicle őssosztálytól örököljük. A függvény segítségével levesszük a Road[] listáról legutolsó utat, amit az autó hátulja éppen elhagy. Ez a roads.remove(roads.size()-1) metódushívással történik. Ezt követően ellenőrzésre kerül hogy az autó esetleg rajta van-e még több úton. Ez a vizsgálat a roads tömb vizsgálatával egyenértékű. Ha úgy találtuk hogy a jármű nincs több úton rajta akkor meghívjuk this.outOfCity() metódusát.

+ double getVelocity() - A metódust Vehicle őssosztálytól örököljük. A függvény egy get-ter metódus megvalósításaként visszaadja a jármű pillanatnyi sebességét (velocity).

+ void crash(Robber robber) - Meghívódik, ha a jármű ütközik, azaz ha a Citizen belement egy Robber-be. Elvileg ez nem történhet meg, mert a citizen-nek le kell lassítani, hogy elkerülje az ütközést, így jobb esetben ez a metódus soha nem kerül meghívásra. A metódus nem csinál egyebet mint meghívja a robber.crash(this) függvényt önmagával ,mely majd ténylegesen lekezeli az ütközést (pl.: a rabló életét csökkenti...stb.)

+ void crash(Citizen citizen) - Meghívódik, ha a jármű ütközik, azaz ha a Citizen belement egy Citizen-be. Elvileg ez nem történhet meg, mert a citizen-nek le kell lassítani, hogy elkerülje az ütközést, így jobb esetben ez a metódus soha nem kerül meghívásra.

+ void crash(Police police) - Meghívódik, ha a jármű ütközik. azaz ha a Citizen belement egy Police-be. Elvileg ez nem történhet meg, mert a citizen-nek le kell lassítani, hogy elkerülje az ütközést, így jobb esetben ez a metódus soha nem kerül meghívásra.

+ void acceptCrash(Vehicle vehicle) - Meghívódik, ha a járműbe belementek. Ebben az esetben a citizen saját magát adja át annak aki őbelé belement. Ezt a következő metódus meghívásával viszi véghez: vehicle.crash(this). Ez az imént emghívott függvény fogja ténylegesen lekezelní az ütközést annak megfelelően, hogy kik voltak az ütköző felek.

+ void outOfCity() - Meghívódik, ha a jármű letért az összes Roadról (vagyis kiment a városból). A függvény jelzi a Game-nek az imént említett tényt, melyet a game.citizenLeftCity(this) hívással teszi meg. E hívásnak az eredményeként többek között a paraméterként átadott citizen lekerül citizens listáról, és helyette egy új generálódik.

+ `VehicleType getType()` – A metódust `Vehicle` őssosztálytól örököljük. A függvény egy `get`-ter metódus megvalósításaként visszaadja az autóra jellemző adatokat tartalmazó objektumot. (`type`)

+ `double getMaxVelocity()` – A metódust `Vehicle` őssosztálytól örököljük. Visszaadja az autóra jellemző maximális sebességet, ezt azonban csak úgy tudja megtenni hogy előtte lekérdezi az adott jármű típusát, amire a maximális sebesség jellemző, majd ennek lekérdezi a maximális sebességét a következőképpen: `this.getType().getMaxVelocity()`

+ `double getMaxAcceleration()` – A metódust `Vehicle` őssosztálytól örököljük. Visszaadja az autóra jellemző maximális gyorsulást, ezt azonban csak úgy tudja megtenni hogy előtte lekérdezi az adott jármű típusát, amire a maximális gyorsulás jellemző, majd ennek lekérdezi a maximális gyorsulását a következőképpen: `this.getType().getMaxAcceleration()`

+ `void setDriver(IDriver driver)` – A metódust `Vehicle` őssosztálytól örököljük. E függvény segítségével lehet kiválasztani az autót vezető AI-t vagy emberi játékost. Ezt a függvény a `driver` paraméter átállításával viszi véghez. Az így kiválasztott `driver` fogja irányítani a továbbiakban a járművet.

+ `void cityBorderReacher()` – A metódust `Vehicle` őssosztálytól örököljük. Ez hívódik meg, ha elérünk egy `CityBorderes TrafficJunction`-t. A metódus az adott `citizen goingOutOfCity` paraméterét állítja be `true`-ra, ezzel jelezve hogy az adott `citizen` már elkezdte elhagyni a várost.

+ `void doTick(int time, int deltatime)` – A metódust `Vehicle` őssosztálytól örököljük. Ezen függvény hatására mozog a jármű az úton. Ez úgy történik hogy először lekérdezésre kerül a `driver` által meghatározott gyorsulás `driver.getAcceleration()`, majd ezután végigmegyünk a `roads` tömb elemein, azaz azokon az utakon melyeken a jármű rajta van, majd mindegyiken léptetjük a járművet az adott `road moveVehicle()` függvényének meghívásával.

+ `void acceptBuilding(Building building)` – A `trafficjunction` hívja meg a jármű ezen függvényét, hogy utána meglátogassa (`visit`) az ott lévő épületet, és lekezelje majd az épület, hogy mi történjen. Tudomásul adja az adott épületnek látogató jármű típusát. Ezt a `building.visit(this)` meghívásával érjük el. Az esetleges történéseket majd ez a függvény fogja lekezelni.

+ `void die()` - Öröklött metódus. A függvény meghívódik, ha az adott jármű meghal. Ekkor a függvény meghívása következtében az adott járművet a metódus leszedi az őt tartalmazó utakról. Ezt a `this.roads.get(0).removeVehicle(this)` végzi el.

8.1.5 CitizenAI

- **Felelősség**

Ez az osztály azért felelős, hogy a városi autókat vezesse, illetve a rendőr autókat, ha éppen deaktiváltak.

- **Össztályok**

- DriverAdapter

- **Interfészek**

- nincs

- **Attribútumok**

- Vehicle vehicle - Az irányított városi autó.
 - double breakDistance - Az a távolság amin belül már lassít a jármű ha van előtte valami (másik jármű, tábla)
 - double maxLITJTime - Az az idő amihez képest viszonyítja, hogy a többi útról ugyanabba a TJ-ba érve mint ez a jármű előbb történik-e meg vagy sem.

- double stopDistance - Megállás esetén ekkora távolsággal fog az előttünk lévő akadály előtt megállni.

- boolean hasStopped - Megállt-e már a jármű a stopptáblánál flag.

- **Metódusok**

- + double getAcceleration(int deltaTime) - Visszaadja a Vehicle gyorsulását az akadályokat figyelembe véve. Itt az történik, hogy az autó FrontRoadján meghívjuk a getNearestPhysicalBarrierDistance függvényt. A megkapott értékből még levonjuk a beállított stopdistance-t. Ez az a távolság ahol meg kell állni. Letároljuk az út végétől való távolságot (distance2).

Ha ez kisebb mint 0 akkor a hasStopped flaget false-ra állítjuk és a distance2-t lehető legnagyobb értékre állítjuk.

Ha közelebb van az út vége mint a köv fizikai akadály, vagy a féktávolság akkor az előttünk lévő TJ-nak végighívjuk az összes TrafficSign-ját. Ha szabad az út akkor isFree flag true.

Ha nagyobb a távolság mint a féktávolság, akkor maximális gyorsulást választunk, ha nem akkor kellőképpen gyorsítunk vagy lassítunk.

- + Road chooseDirection(Road [] roads) – Választ egy utat hogy merre közlekedik tovább a jármű. Ezt úgy csinálja, hogy ha a paraméter nem null vagy nem üres tömb akkor random függvényel választ az utak közt és azt az utat adja vissza.

- + boolean visit(TrafficLight trafficLight) – Megmondja hogy az adott lámpánál mit tesz a vezető. Meghívja a paraméterül kapott lámpa isGreen függvényét és annak igazságértékével tér vissza.

- + boolean visit(StopSign stopSign) – Megmondja hogy az adott stopptáblánál mit tesz a vezető. Itt az történik, hogy ha az autó nem állt meg és még van sebessége akkor false. Csinálunk egy listát a TJ inRoadjaiból és meghívjuk a rajtuk kívüliekre a getIncomingVehicleTime fv-t és ha ez kevesebb időt ad vissza mint a beállított értékünk akkor szintén false. Egyéb esetben true.

8.1.6 CityBorder

- **Felelősség**

A városhatár. Itt tudnak a Citizen-ek kilépni/belépni a városba, illetve itt tud a bankrabló eltűnni ideiglenesen a rendőrök elől.

Az osztály felelőssége hogy ezeket az eseményeket kezelje le.

- **Ősosztályok**

Building

- **Interfészek**

nincs

- **Attribútumok**

Game game - Egy referencia a Game-re. A referenciát az osztály a Building űosztályától öröklí. Ezt az attribútumot CityBorder nem használja fel, ellenben egyéb Building leszármazottakkal akik viszont igen, ezáltal szükségesszerűen a CityBorder is megöröklí az adott referenciát..

- **Metódusok**

+ void visit(Robber robber) - Ha egy Robber érkezik a CityBorderhez ez hívódik meg. A metódus lefolyása során először a robber.setCityBorderTrafficJunction() függvényhívással a rabló eltárolja hogy mely kereszteződésénél hagyta el a várost, mely a rabló városba való visszajövetelénél lesz hasznos. Majd ezt követően robber.cityBorderReached() metódus a robber goingOutOfCity paraméterét true-ra állítja, ezzel jelezve hogy az adott rabló elkezdte elhagyni a várost.

+ void visit(Citizen citizen) - Ha egy Citizen érkezik a CityBorderhez ez hívódik meg. Ebben az esetben meghívásra kerül a citizen.cityBorderReached() metódus, mely az adott citizen goingOutOfCity paraméterét állítja be true-ra, ezzel jelezve hogy az adott citizen már elkezdte elhagyni a várost.

+ void visit(Police police) - Ha egy Police érkezik a CityBorderhez ez hívódik meg. Ekkor meghívásra kerül a police.cityBorderReached() metódus, mely az adott police goingOutOfCity paraméterét állítja be true-ra. Elvileg a rendőrök nem hagyhatják el a várost, így ez a metódus nem kerülhet meghívásra.

8.1.7 DriverAdapter

- **Felelősség**

Ez az adapter osztály azért felelős, hogy összefogja a vezető osztályokat (PoliceAI, CitizenAI, KeyboardPlayer)

- **Ősosztályok**

nincs

- **Interfészek**

IDriver

- **Attribútumok**

nincs

- **Metódusok**

double getAcceleration(int deltaTime) - Visszaadja a Vehicle gyorsulását az akadályokat figyelembe véve. A függvény pontos implementációja a származtatott osztályban található mivel e egy abstract függvény.

+ *Road chooseDirection(Road [] roads)* – Választ egy utat hogy merre közlekedik tovább a jármű. A függvény pontos implementációja a származtatott osztályban található mivel e egy abstract függvény.

+ *boolean visit(TrafficLight trafficLight)* – Megmondja hogy az adott lámpánál mit tesz a vezető. Igazzal tér vissza

+ *boolean visit(StopSign stopSign)* – Megmondja hogy az adott stoptáblánál mit tesz a vezető. Igazzal tér vissza

8.1.8 Game

- **Felelősség**

Ez az osztály hozza létre és semmisíti meg a térképet és a kocsikat, meghatározza a kocsik viselkedését. Ezt az osztályt használjuk például inicializáláskor.

Ez az osztály "irányítja" a játék mérföldköveit. Ilyen mérföldkönek számít például a bank kirablása, a város elhagyása, a rejtekhely elérés, a rabló elkapása stb.

- **Össztályok**

nincs

- **Interfészek**

ITick

- **Attribútumok**

Map map - Térkép mely tartalmazza, összefogja a térképen lévő különböző épületeket, utcákat, kereszteződéseket, táblákat, lámpákat.

Timer timer - A különböző objektumok időzítéséért felelős.

Robber[0..] robbers* - A játékban résztvevő rabló(k). Ezeken keresztül tudjuk őket irányítani, pl ha eltűnik, meghal, elkapják stb.

Police[0..] polices* - A játékban résztvevő rendőrök. Ezen keresztül tudjuk őket irányítani, aktiválás, deaktiválás.

Citizen[0..] citizens* - A játékban résztvevő városi autók. Ezt használjuk pl. ha elhagyja a várost egy ilyen jármű

- **Metódusok**

+ *void bankRobbed()* - Akkor hívódik meg, amikor egy rabló kirabolja az első bankot. Meghívja a *activatePolices* függvényt.

+ *void citizenLeftCity(Citizen citizen)* - A paraméterül kapott citizen jármű városelhagyásakor hívódik meg, megsemmisíti a citizent és újat hoz létre. Ezt úgy teszi, hogy a régi citizenre meghívja a *killVehicle* függvényt, majd létrehoz egy új Citizent, beállítja a tulajdonságait. Lekéri a *CitiBorderes TJ*-ket és ezek közül random

választ egyet, és annak a kivezető útjára rátesszi az új járművet. Végül hozzáadja a Timerhez a járművet.

+ void robberCaught(Robber robber) - Ha egy rendőr elér egy rablót, ez a függvény hívódik meg. A játék sikertelen végét jelenti. Ezt úgy teszi, hogy meghívja a killVehicle függvényt a robberre. A a robbers tömb mérete 0-ra csökkent akkor megállítja a timert.

+ void robberReachedShelter(Robber robber) - Ha a rabló eléri a búvóhelyet, ez a függvény hívódik meg. A játék sikeres végét jelenti. Megállítja a timert.

+ void robberDied(Robber robber) - Ez a függvény kezeli le, ha a a rablónak elfogynak az élete és úgy ér véget sikertelenül a játék. Ezt úgy teszi, hogy meghívja a killVehicle függvényt a robberre. A a robbers tömb mérete 0-ra csökkent akkor megállítja a timert.

+ void robberDisappeared(Robber robber) - Ez a függvény kezeli le, ha a rabló "kimenekül" a városhatáron kívülre. Végigmegy a robbers tömbön és megvizsgálja, hogy üldözik-e bármelyik rablót. Ha igen, akkor nem csinál semmit, ha nem akkor meghívja a deactivatePolices függvényt.

+ void robberGoBackToCity(Robber robber) - Meghívja a robber goBackToCity függvényét.

+ activatePolices - Aktiválja az összes rendőrt. Végigmegy a polices tömbön és minden rendőrnek meghívja az activate függvényét.

+ deactivatePolices - Deaktiválja az összes rendőrt. Végigmegy a polices tömbön és minden rendőrnek meghívja a deactivate függvényét.

+ killVehicle(Citizen citizen) - Eltávolítja a citizen az összes útról a citizen die függvényével. Kiveszi a citizens tömbből, és leveszi a timerről is.

+ killVehicle(Police police) - Eltávolítja a policet az összes útról a police die függvényével. Kiveszi a polices tömbből, és leveszi a timerről is.

+ killVehicle(Robber robber) - Eltávolítja a robbert az összes útról a robber die függvényével. Kiveszi a robbers tömbből, és leveszi a timerről is.

+ void doTick(int time, int deltaTime) - Minden tick esetén valamekkora valószínűséggel valamelyik TJ-ra hozzáad egy Bunny-t.

8.1.9 IDriver

- **Felelősség**

A járművek vezetése ezen az interfacen keresztül történik.

- **Ősosztályok**

ITrafficRuleHandler (ős-interface)

- **Interfészek**

nincs

- **Attribútumok**

nincs

- **Metódusok**

+ Road chooseDirection(Road[] roads) - Visszaadja, hogy merre halad tovább a Vehicle a kereszteződéseknél. Mivel ez egy interface, a függvény implementációja az interfacet használó osztályokban található.

+ double getAcceleration(int deltaTime) - Visszaadja a Vehicle gyorsulását az akadályokat figyelembe véve. Mivel ez egy interface, a függvény implementációja az interfacet használó osztályokban található.

+ boolean visit(TrafficLight trafficLight) – Megmondja hogy az adott lámpánál mit tesz a vezető. Mivel ez egy interface, a függvény implementációja az interfacet használó osztályokban található.

+ boolean visit(StopSign stopSign) – Megmondja hogy az adott stoptáblánál mit tesz a vezető. Mivel ez egy interface, a függvény implementációja az interfacet használó osztályokban található.

8.1.10 ITick

- **Felelősség**

Az egész játék időzítése ezen az interfacen keresztül történik. Lámpakezelés, autó mozgatása, játék vezérlése.

- **Ősosztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

nincs

- **Metódusok**

+ void doTick(int time, int deltaTime) - Meghívódik, ha éppen egy időegységnyivel később van. Folyamatosan hívódik a Timer osztályon keresztül. Ez "lépteti" a játékot. Mivel ez egy interface, a függvény implementációja az interfacet használó osztályokban található.

8.1.11 ITrafficRuleHandler

- **Felelősség**

A járművek a közlekedési objektumokat (lámpa, tábla, stb.) ezen az interfacen keresztül kezelik.

- **Ősosztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

nincs

- **Metódusok**

+ boolean visit(StopSign stopSign) – Meghívódik egy adott jármű Stop táblához való érkezése esetén. Mivel ez egy interface, a függvény implementációja az interfacet használó osztályokban található.

+ boolean visit(TrafficLight trafficLight) – Meghívódik egy adott jármű Lámpához való érkezése esetén. Mivel ez egy interface, a függvény implementációja az interfacet használó osztályokban található.

8.1.12 Map

- **Felelősség**

Térkép mely tartalmazza, összefogja a térképen lévő különböző épületeket, utcákat, keresztezéseket, táblákat, lámpákat. A térképtől lekérdezhetőek többek között az általa tárolt objektumok az olyan osztályok számára akiknek erre szükségük van.

- **Ősosztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

- ArrayList<Building> buildings - Az épületeket tartalmazó lista. A Map osztály szolgáltatásaként jelentkező get-ter függvénnyel kérdezhető le (getBuildings()) azok számára az osztályok számára kiknek szükségük van a térkép ismeretére.

- ArrayList<TrafficJunction> trafficJunctions - Az útkeresztezéseket tartalmazó lista. A Map osztály szolgáltatásaként jelentkező get-ter függvénnyel kérdezhető le (getTrafficJunctions()) azok számára az osztályok számára kiknek szükségük van a térkép ismeretére.

- ArrayList<Road> roads Az utakat tartalmazó lista. A Map osztály szolgáltatásaként jelentkező get-ter függvénnyel kérdezhető le (getRoads()) azok számára az osztályok számára kiknek szükségük van a térkép ismeretére.

- ArrayList<TrafficSign> trafficSigns A közlekedési táblákat, lámpákat tartalmazó lista. A Map osztály szolgáltatásaként jelentkező get-ter függvénnyel kérdezhető le (getTrafficSigns()) azok számára az osztályok számára kiknek szükségük van a térkép ismeretére.

- ArrayList<TrafficJunction> cityBorderTrafficJunctions Azokat az útkeresztezéseket tartalmazó lista melyek egyben városhatárok is. A Map osztály szolgáltatásaként jelentkező get-ter függvénnyel kérdezhető le (getTrafficJunctions()) azok számára az osztályok számára kiknek szükségük van a térkép ismeretére.

- **Metódusok**

- void LoadFromFile(Timer timer, Game game, String fileName) A függvény egy külső fájlból betölti a fájlban tárolt térképet. A paraméterként átadott string tartalmazza a fájl nevét, melyből először megtörténik a csomópontok beolvasása, majd az utak beolvasása valamint ezek összeillesztése. Timer paraméterre azért van szükség mert azoknak az objektumoknak melyek működéséhez elengedhetetlen a timer tick szolgáltatása, azok így feliratkozhatnak erre az eseményre. pl.: timer.addTicker(trafficLight) A game paraméter a speciális épületek létrehozásánál szükséges, hogy ezek az épületek jelezni tudják a Game-nek különböző események bekövetkeztét. pl: Bank->a bankot kirabolták.

- + Buliding[] getBulidings() -A függvény a térképen található épületek lekérdezhetőségéért felelős egy get-ter típusú metódus megvalósításaként.

- + TrafficJunction[] getTrafficJunctions -A függvény a térképen található útkereszteződések lekérdezhetőségéért felelős egy get-ter típusú metódus megvalósításaként.

- + Road[] getRoads() -A függvény a térképen található utak lekérdezhetőségéért felelős egy get-ter típusú metódus megvalósításaként.

- + TrafficSign[] getTrafficSigns() -A függvény a térképen található közlekedési táblák valamint lámpák lekérdezhetőségéért felelős egy get-ter típusú metódus megvalósításaként.

- + TrafficJunction[] getCityBorderTrafficJunctions() - A függvény a térképen található városhatárként funkcionáló útkereszteződések lekérdezhetőségéért felelős egy get-ter típusú metódus megvalósításaként.

8.1.13 Police

- **Felelősség**

A rendőr autója, ami nem a városi autó és nem a bankrabló autója. Ha nem köröz senkit akkor nem megy neki az előtte haladó járműnek, lelassít ha ő is lelassít, valamint betartja a közlekedési szabályokat.

Ha azonban egy rablóra körözöttséget adnak ki akkor ezek a tulajdonságai megváltoznak. Ugyancsak nem megy neki senkinek kivéve ha rablót lát maga előtt, mivel azzal hogy ütközik vele, elkapja, és ezzel teljesítette a rendőr saját feladatát, mely a rabló játékanak sikertelen végét jelenti. Körözés esetén a rendőrök nem tartják be a közlekedési szabályokat.

A rendőrautó felelőssége többek között az ütközések kezelése valamint a saját sofőrével (driver-rel) (PoliceAI) való kommunikáció megvalósítása.

- **Ősosztályok**

Vehicle

- **Interfészek**

nincs

- **Attribútumok**

- IDriver defaultDriver - A rendőrautó alapértelmezett vezetője (irányítója). Legtöbbször a PoliceAI, de ugyanakkor lehetőséget biztosít arra hogy KeyboardPlayer-ként más játékos is irányítsa a rendőrt. Azért van rá szükség hogy külön lehessen kezelni azokban az esetekben a rendőr viselkedését amikor üldözi a rablót vagy amikor nem üldözi. Az activate() függvény meghívásának hatására a defaultdriver veszi át az irányítást.

- boolean active - A rendőr állapotát jelző attributum. (true - a rendőr üldözi a rablót; false - a rendőr nem üldözi a rablót). Az activate() valamint a deactivate() függvények segítségével változtatható az attribútum értéke.

- # boolean goingOutOfCity - Akkor igaz, ha a kocsí eleje már belement egy CityBorderes kereszteződésbe, de a kocsí vége még nem ért ki a városból. Még rajta van valamelyik Road-on. Értékének true-ra történő beállítása a cityBorderReached() metódus segítségével történik.

- # Game game – Egy referencia a Game objektumra. Szükségessége akkor jelentkezik amikor a rendőr ütközik a rablóval, hiszen ekkor a game.robberCaught(robber) metódushívással jelezzük a Game-nek hogy a rablót elkapták, azaz vége a játéknak.

- **Metódusok**

- + void activate () - Az adott rendőr aktiválásáért felelős függvény. (aktív -> defaultdriver veszi át az irányítást). Az active attribútumot true-ra állítja, valamint a driver átállítását a this.setDriver(defaultDriver) hívással végzi. A defaultDriver általában a PoliceAI így a függvényhívás eredményeként megkezdődik a rabló üldözése.

- + void deactivate() - Az adott rendőr deaktiválásáért felelős függvény. (deaktív - > citizen-ként viselkedik). Az active attribútumot false-ra állítja, valamint a driver átállítását a this.setDriver(new CitizenAI(this)) hívással végzi. Így a rendőr citizenként viselkedik, azaz nem üldözi a rablót, és betartja a közlekedési szabályokat.

- + boolean getActivate() - A függvény segítségével lekérdezhető az adott rendőr állapota (aktivitása). A függvény egy get-ter metódus megvalósításaként adja vissza a active attributum értékét.

- + void setDefaultDriver (IDriver : defaultDriver) - A rendőrautó alapértelmezett vezetőjének (irányítójának) beállításáért felelős függvény. A függvény egy set-ter típusú metódust valósít meg. Az alapértelmezett vezető legtöbbször a PoliceAI, de ugyanakkor lehetőséget biztosít arra hogy KeyboardPlayer-ként más játékos is irányítsa a rendőrt. Azért van rá szükség hogy külön lehessen kezelni azokban az esetekben a rendőr viselkedését amikor üldözi a rablót vagy amikor nem üldözi. Az activate() függvény meghívásának hatására a defaultdriver veszi át az irányítást.

- + Road getFrontRoad() -A függvény visszaadja azt a Road-ot, amelyiken az autó eleje van. Ezt a roads.get(0) metódushívással éri el, mely visszaadja a roads tömb első elemét.

+ `Road getDirection(Road[] roads)` -A függvény segítségével kiválasztjuk a továbbhaladási irányt a következő módon. Először megvizsgáljuk hogy az adott jármű kifelé halad-e a városból, azaz hogy a `GoingOutOfCity` paramétere `true`-e (elvileg a rendőr nem haladhat kifelé a városból) . Ha már elkezdte elhagyni a várost akkor `null`-t adunk vissza, ha viszont nem akkor megkérdezzük a vezetőt, azaz meghívjuk a `driver.chooseDirection(roads)` függvényt. Ebben az esetben az ebből származó visszatérési érték lesz a függvény visszatérési értéke. Ez a visszatérési érték egy `road`, az a `road` amit a `driver` megfelelőnek talált a továbbhaladásra.

+ `void addFrontRoad(Road road)` - A függvény segítségével hozzáadjuk a `Road[]` listához azt az utat, amire az autó eleje rátért. Ezt a `roads.add(0, road)` metódushívással érjük el, mely a `roads` tömb első helyére teszi a paraméterként átadott `road`-ot.

+ `void removeLastRoad()` - A függvény segítségével levesszük a `Road[]` listáról legutolsó utat, amit az autó hátulja éppen elhagy. Ez a `roads.remove(roads.size()-1)` metódushívással történik. Ezt követően ellenőrzésre kerül hogy az autó esetleg rajta van-e még több úton. Ez a vizsgálat a `roads` tömb vizsgálatával egyenértékű. Ha úgy találtuk hogy a jármű nincs több úton rajta akkor meghívjuk `this.outOfCity()` metódusát.

+ `double getVelocity()` - A függvény egy `get`-ter metódus megvalósításaként visszaadja a jármű pillanatnyi sebességét (`velocity`).

+ `void crash(Robber robber)` - Meghívódik, ha a jármű ütközik, azaz ha a jármű belement egy `Robber`-be. Ha ez az eset fennállna akkor mind a két `robber` meghal, melyet jelzünk a `Game`-nek is. `game.robberDied(this)`, `game.robberDied(robber)` metódushívásokkal.

+ `void crash(Citizen citizen)` - Meghívódik, ha a jármű ütközik, azaz ha a jármű belement egy `Citizen`-be. Ekkor a rabló élete lecsökken, valamint a vele ütközött `citizen` meghal, melyet jelzünk a `game`-nek `this.game.killVehicle(citizen)` metódushívással. Végül ellenőrizzük, hogy a `robber` élete nem csökkent a megengedett minimum alá, mert különben a `robber` meghal és ezt szintén jeleznünk kell a `game`-nek `game.robberDied(this)`

+ `void crash(Police police)` - Meghívódik, ha a jármű ütközik, azaz ha a jármű belement egy `Police`-be. Ez az eset egyenértékű azzal mintha elkapták volna a rablót, így ezt jeleznünk kell a `game`-nek `game.robberCaught(this)` metódushívással.

+ `void acceptCrash(Vehicle vehicle)` - Meghívódik, ha a járműbe belementek. Ebben az esetben az adott jármű saját magát adja át annak aki őbelé belement, így az ütközés további menetét ő fogja lekezelni. A függvényben ezért a `vehicle.crash(this)` függvényhívás történik.

+ `void outOfCity()` - Meghívódik, ha a jármű letért az összes `Road`-ról (vagyis kiment a városból). -----

+ `VehicleType getType()` – A függvény egy `get`-ter metódus megvalósításaként visszaadja az autóra jellemző adatokat tartalmazó objektumot. (`type`)

+ `double getMaxVelocity()` –Visszaadja az autóra jellemző maximális sebességet, ezt azonban csak úgy tudja megtenni hogy előtte lekérdezi az adott jármű típusát, amire a maximális sebesség jellemző, majd ennek lekérdezi a maximális sebességét a következőképpen: `this.getType().getMaxVelocity()`

+ `double getMaxAcceleration()` –Visszaadja az autóra jellemző maximális sebességet, ezt azonban csak úgy tudja megtenni hogy előtte lekérdezi az adott jármű típusát, amire a maximális gyorsulás jellemző, majd ennek lekérdezi a maximális gyorsulását a következőképpen: `this.getType().getMaxAcceleration()`

+ `void setDriver(IDriver driver)` – E függvény segítségével lehet kiválasztani az autót vezető AI-t vagy emberi játékost. Ezt a függvény a driver paraméter átállításával viszi véghez. Az így kiválasztott driver fogja irányítani a továbbiakban a járművet.

+ `void cityBorderReacher()` –Ez hívódik meg, ha elérünk egy CityBorderes TrafficJunctiont. A metódus az adott citizen goingOutOfCity paraméterét állítja be true-ra, ezzel jelezve hogy az adott citizen már elkezdte elhagyni a várost.

+ `void doTick(int time, int deltatime)` –Ezen függvény hatására mozog a jármű az úton. Ez úgy történik hogy először lekérdezésre kerül a driver által meghatározott gyorsulás `driver.getAcceleration()`, ezt összevetjük az adott jármű maximális gyorsulásával `this.getMaxAcceleration()`. Ezután kiszámítjuk a sebességet az addigi sebességgel és a kiszámított gyorsulással kombinálva. Ezt is meg kell vizsgálni hogy nem haladtuk-e meg az adott járműre meghatározott maximális sebességet `this.getMaxVelocity()`. Eztkövetően végigmegyünk a roads tömb elemein, azaz azokon az utakon melyeken a jármű rajta van, majd mindegyiken léptetjük a járművet az adott road `road.moveVehicle(this, velocity*deltatime/1000)` függvényének meghívásával, melynek átadjuk magát a járművet, valamint a számított sebességet kombinálva a deltatime-mal mely meghatározza a mozgás mértékét.

+ `void acceptBuilding(Building building)` – A trafficjunction hívja meg a jármű ezen függvényét, hogy utána meglátogassa (visit) az ott lévő épületet, és lekezelje majd az épület, hogy mi történjen. Tudomásul adja az adott épületnek látogató jármű típusát. Ezt a `building.visit(this)` meghívásával érjük el. Az esetleges történéseket majd ez a függvény fogja lekezelni.

+ `void die()` -Öröklött metódus. A függvény meghívódik, ha az adott jármű meghal. Ekkor a függvény meghívása következtében az adott járművet a metódus leszedi az őt tartalmazó utakról. Ezt a `this.roads.get(0).removeVehicle(this)` végzi el.

8.1.14 PoliceAI

- **Felelősség**

Ez az osztály azért felelős, hogy a rendőrt vezérelje, és az elkapja a rablót.

- **Összaltályok**

DriverAdapter

- **Interfészek**
nincs
- **Attribútumok**
 - Vehicle vehicle - Az irányított városi autó.
- **Metódusok**
 - + double getAcceleration(int deltaTime) - Visszaadja a Vehicle gyorsulását az akadályokat figyelembe véve. Nullát ad vissza
 - + Road chooseDirection(Road [] roads) – Választ egy utat hogy merre közlekedik tovább a jármű. Nullal tér vissza.
 - + boolean visit(TrafficLight trafficLight) – Megmondja hogy az adott lámpánál mit tesz a vezető. Igazsal tér vissza
 - + boolean visit(StopSign stopSign) – Megmondja hogy az adott stoptáblánál mit tesz a vezető. Igazsal tér vissza

8.1.15 ProtoDriver

- **Felelősség**
A Robber irányítását teszi lehetővé konzolos felületről.
- **Ősosztályok**
DriverAdapter
- **Interfészek**
örökölt IDriver
- **Attribútumok**
 - double acceleration - Az autó gyorsulását adja meg ('a vezető ennyire nyomja a gázpedált')
 - int direction - Az autóval a következő útkereszteződésnél ebbe az irányba fog menni az autós
- **Metódusok**
 - + void setAcceleration(double value) - Beállítja az "acceleration" tagváltozót
 - + void setDirection(int value) – Beállítja a "direction" tagváltozót.
 - + double getAcceleration(int deltaTime) – A megvalósított IDriver interfész része: meghatározza a gyorsulást. A ProtoDriver az előre megadott "acceleration" tagváltozót adja vissza.
 - + Road chooseDirection(Road[] roads) – A megvalósított IDriver interfész része: meghatározza merre menjen az útkereszteződésnél. A ProtoDriver az előre megadott "direction" tagváltozónak megfelelő utat adja vissza.

8.1.16 ProtoGame

- **Felelősség**

A játék vezérlését úgy egészíti ki, hogy hozzáférést enged a játék belső változóihoz, hogy azokat a proto kezelői felületéről módosítani és lekérni lehessen. Ezenkívül lehetővé teszi a játékmenetbe való beavatkozást, pl. az idő múlásának szabályozását vagy a véletlengenerálás kikapcsolását.

- **Ősosztályok**

Game

- **Interfészek**

örökölt: ITick

- **Attribútumok**

< Az osztály több attribútumot is örököl a szülőosztályától! Ezeket lásd a Game osztály leírásánál! >

ProtoDriver protoDriver - A Robber-t irányító osztály. Ez a proto konzolos képernyőjéről történő irányításhoz szükséges.

randomIsOn: boolean - A játék véletlenszám generátorát módosítja. Ha true, akkor történik véletlenszám generálás, ha false, akkor véletlenszám helyett mindig 0-at ad vissza a random() függvény.

- **Metódusok**

+ tick(int num): void - Lépteti a játékban lévő időt. A "num" paraméterben megadott db lépést hajt végre, mindegyik lépés a timer "deltaTime" tagváltozójában tárolt ezredmásodpercet léptet a játék idején. A timer tick metódusát hívja meg "num"-nyi szor.

+ showMap(): void - Kiírja a játékban lévő autók helyét és sebességét. A pontos kimenethez lásd a 7.1.3-as pontot.

+ setDeltaTime(int num): void - Meghívja a timer setDeltaTime függvényét az adott paraméterrel. Lásd a Timer osztály setDeltaTime függvényét pontosabb leíráshoz.

+ playerSetDirection(int num): void - Meghívja a protoDriver setDirection függvényét az adott paraméterrel. Lásd a ProtoDriver osztály setDirection függvényét pontosabb leíráshoz.

+ playerSetAcc(double num): void - Meghívja a protoDriver setAcceleration függvényét az adott paraméterrel. Lásd a ProtoDriver osztály setAcceleration függvényét pontosabb leíráshoz.

+ playerSetWanted(int num): void - Beállítja a Robber üldözöttségét. 1, ha üldözik a rendőrök, ilyenkor meghívja a Game activatePolices metódusát, 0, ha nem üldözik a rendőrök, ilyenkor meghívja a Game deactivatePolices metódusát.

+ `playerGoBackToCity(): void` - Meghívja a `robberGoBackToCity` függvényt. Lásd a `Game` osztály `robberGoBackToCity` függvényét pontosabb leíráshoz.

+ `setRand(int num): void` - Ha 1, akkor a "randomIsOn" tagváltozó true lesz, és a program nem lesz determinisztikus, egyéb esetben a "randomIsOn" false lesz és a program determinisztikusan fog működni.

+ `random(): double` - Visszaad egy 0 és 1 közötti véletlen törtszámot, ha a "randomIsOn" tagváltozó true, ellenkező esetben 0-at ad vissza.

+ `addVehicle(char vehicleCode, int roadIndex, double position): void` - A megadott útra a megadott pozícióba a megadott típusú autót helyezi le. A `vehicleCode` lehet: "r" (Robber), "c" (Citizen), "p" (Police). A `roadIndex` a mapnek megfelelő számozással annak a Roadnak az indexe, ahova az autót le kell rakni. A `position` egy 0 és 1 közötti tört szám, ami meghatározza, hogy az autó a Roadnak melyik részére fog kerülni. 0 jelenti az út elejét (ahol az autó ráhajt az egyirányú útra), míg az 1 jelenti az út végét (ahol lehajt róla).

8.1.17 ProtoHelper

- **Felelősség**

Felelőssége a felhasználó által adott proto konzolos képernyőjén bekért és/vagy fájlból beolvasott parancsok feldolgozása, eredmények megjelenítése a képernyőn és/vagy fájlba mentése

- **Össztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

- simpleCode: `HashMap<Integer, Integer>` - Az objektumok `hashCode`-ját átalakítja egy sokkal egyszerűbben nyomkövethető `>= 1` számra. A kulcs az objektum `hashCode`-ja, még az érték az egyszerűsített szám.

- appending: `boolean` - A log fájlhoz hozzáfűnk (`appending==true`), vagy új fájlt nyitunk (`appending==false`). A program futása során első íráskor false, utána végig true. Így programfutásonként új fájlt nyit (`out.txt`).

- ProtoGame: `pGame` - A jelenlegi játék objektum, amit éppen tesztelünk. Kezdetben null, majd a `loadMap` hatására felépül a játéktér és ezen keresztül érhető el.

- **Metódusok**

- loadMap(String fileName): void - Betölti a "fileName" paraméterben megadott fájlnevű 7.1.2-es pontban ismertetett fájlformátumú fájlból a játéktér pályáját. Létrehoz egy új játéktér, majd létrehozza a pályát. A simpleCode-t törli, mert új pályán az összes objektumnak a számozását előlről kell kezdeni.

- + executeCommand(String command): boolean - Értelmezi a szövegesen megadott parancsot, majd meghívja a megfelelő parancshoz tartozó metódust. A parancsokhoz tartozó metódusok a ProtoHelper és ProtoGame osztályokban találhatók. Ha a ProtoHelper metódusát hívja, akkor az statikus metódus, ha a ProtoGame metódusát hívja, akkor a "pGame" tagváltozónak a tagmetódusát hívja meg. Ezért a ProtoGame metódusokat csak akkor tudja hívni, ha már létezik a pGame tagváltozó, aminek előfeltétele, hogy történt már legalább 1 loadMap parancsvégrehajtás. Visszatérési értéke true, ha folytatni kell a futást, false, ha ki kell lépni a programból. A használható parancsok listájáért lásd a 7.1.2-es pontot!

- + loadCommands(String fileName): boolean - Az adott fájlból soronként betölti a parancsokat és végrehajtja őket. Amennyiben hibás a bemeneti fájl, jelzi. Egyébként az executeCommand paranccsal végrehajtja az összes beolvasott parancsot. Visszatérési értéke true, ha folytatni kell a futást, false, ha ki kell lépni a programból

- ObjectToString(Object o): String - Egy objektumot szöveges, kiírható formába alakítja át. Az java beépített osztályokat (a "java" packagebe tartozókat) a saját toString() metódusukkal. Double kivétel, mert azt csak 2 tizedesponossággal adja vissza. A nem java beépített osztályokat (pl. a projekt keretein belül írtakat) pedig az egyszerűsített objektumkóddal (lásd simpleCode attribútum leírása) és az osztályfajtajával jellemzi a következő formátumban: "#<objektumkód>:<osztály>".

- + Log(String s, Object... args): void - Kiírja az "s" paraméterben szereplő formátum alapján az "args" paraméterlistában szereplő objektumokat. Az "s" formátumleíró felépítése: a kiírandó szöveget tartalmazza, de a "{n}" szövegrészeket (ahol n = 0 .. args tömb hossza -1) lecseréli az "args" tömbben található objektum ObjectToString függvény által megformázott szöveges reprezentációjára. A kiírást a Log(String s, boolean printOut) függvénnyel végzi.

- Log(String s, boolean printOut): void - Kiírja az "s" paraméterben foglalt információt az "out.txt" fájlba, illetve a képernyőre is, ha a printOut paraméter true. Ha nem sikerül a kiírás, akkor a képernyőn hibaüzenetet jelenít meg.

- getSimpleCode(int hashCode): int - Visszaadja az egyszerűsített objektumkódot (lásd simpleCode attribútum). Ha a paraméterként átadott "hashCode"-hoz még nem tartozik egyszerűsített objektumkód, akkor generál hozzá.

8.1.18 Road

- **Felelősség**

Egy utat testesít meg, amely egyirányú és egysávos, melyen a járművek közlekednek. Az út két végén mindenképpen van TJ. Tartalmazhat különböző

közlekedést szabályozó eszközöket (táblák, lámpa), mindenképpen az út végén. Illetve tartalmazhatja a rajta lévő autókat bedobozolva.

- **Ősosztályok**
nincs

- **Interfészek**
nincs

- **Attribútumok**

- VehicleOnRoad[0..*] vehicles - Az úton tartózkodó (közlekedő) járműveket tartalmazó tömb. Akkor van jelentősége, amikor a Timer hatására az úton lévő Vehicle mozogni fog. Ez úgy valósul meg, hogy meghívja a doTick fv meghívja a Road moveVehicle függvényét, és az lekéri ebből a tömbből a járműt bedobozolva és majd annak a pozícióját állítja.

- TrafficJunction[2] trafficJunctions - Az út két végén található kereszteződéseket tartalmazó tömb. Az elsőből a másodikba megy az út. Például akkor használjuk, amikor az autó elér az út végére akkor meghívja a TJ vehicleArrived függvényét a járművel.

- TrafficSign[0..*] trafficSigns - Az úton található különböző közlekedést szabályozó eszközök tömbje (lámpák, táblák). Mindegyik az út végén található (egyirányúság).

- **Metódusok**

- + void addTrafficSign(TrafficSign trafficSign) – A függvény az útra helyez különböző TrafficSignokat (lámpa, tábla). Inicializálásnál használhatjuk, amikor létrehozunk egy Roadot és annak a paramétereit beállítjuk.

- + double getLength() - Kiszámolja és visszaadja az út hosszát. Mindezt úgy teszi, hogy az utat lezáró TJ-k getX() ill. getY() függvényeivel meghatározzuk, az út végeinek fizikai koordinátáit, majd a Pitagorasz tétel alkalmazásával kiszámolja a Road hosszát, és visszaadja.

- + boolean isOccupied(TrafficJunction trafficJunction) - Az út azon részének foglaltságát jelzi ami még belelóg a kereszteződésbe. 2 féle lehetőség van: 1: Azt nézi hogy az út elején lévő TJ-hoz közel van-e jármű Ezt úgy csinálja, hogy az lekéri a legutolsó járműt a tömbből (ez lesz a legelső jármű az úton) a getPosition függvénnyel megkapja a jármű pozícióját. 2: Azt nézi hogy az út végén lévő TJ-hoz közel van-e jármű Ezt úgy csinálja, hogy az lekéri a legelső járműt a tömbből (ez lesz a legutolsó jármű az úton) a getPosition függvénnyel megkapja a jármű pozícióját. Majd a megkapott távolságot felszorozzuk az út hosszával, majd leellenőrizzük, hogy ez nagyobb-e a roadEndLength. Ha nagyobb, akkor false -> tehát nem foglalt. Ha kisebb akkor true, tehát foglalt.

- + void addVehicle(Vehicle vehicle) - A függvény az útra helyez egy járművet. Ezt úgy teszi, hogy a Road jármű tömbjéhez az add függvénnyel hozzáad egy új VehicleOnRoad-ot majd az eredeti jármű addFrontRoad() függvényével beállítja ezt az utat, hogy a jármű eleje itt található.

+ void removeVehicle(Vehicle vehicle) - A függvény levesz az útról egy járművet. Ezt úgy teszi, hogy egy for ciklussal végigmegy a járműket tároló vehicle tömbön. A bedobozolt járműeknek a getVehicle függvényével lekéri a járművet és megnézi, hogy ez-e a keresett jármű. Ha ez igaz, akkor meghívja a jármű remove függvényét.

+ double getNearestPhysicalBarrierDistance(Vehicle vehicle) - A függvény a legközelebbi fizikai akadálytól való távolságot (az út irányában) adja vissza (fizikai: olyan dolog amivel ütközni lehet pl.: jármű). Ezt úgy teszi, hogy meghatározzuk a járműhöz tartozó VehicleOnRoad példányt a vehicle tömb get függvényével. Majd ha nem ez a legutolsó autó az úton (vehicle tömbben nem ő a legelső) akkor lekéri az előtte lévő egyel kisebb indexű járműt. Mindkét járműnek meghatározza a pozícióját a getPosition fv-el, és a két távolságot kivonja egymásból, és ezzel tér vissza. Ha ő a legutolsó, akkor meghívja a kereszteződés végére az isOccupied függvényt. Ha igaz akkor megnézi, levonja az ő pozícióját az út hosszából, ezzel megkapjuk, milyen messze van az út végétől, végül ebből levonjuk a roadEndLength-t.

+ TrafficSign[] getTrafficSigns() - Egyszerű getter függvény, visszaadja a Roadban tárolt trafficSigns tömböt.

+ Vehicle getNearestVehicle(Vehicle vehicle) - Egy autótól az adott úton lévő legközelebbi jármű lekérdezésére szolgáló metódus (az út irányában). Ezt úgy teszi, hogy ha ez a jármű a vehicle tömb első eleme, akkor null-t ad vissza, mivel ez a jármű van legelőbbre az úton. Ha nem az első elem, akkor lekéri az egyel kisebb indexű járműt és azt adja vissza.

+ TrafficJunction[] getTrafficJunctions() - A függvény visszatérési értékül adja az adott úthoz tartozó közlekedési csomópontokat.

+ void moveVehicle(Vehicle vehicle, double distance) - A függvény az adott járművet a megadott távolságban mozgatja az úton. Először eltároljuk a régi pozíciót, kiszámoljuk az új pozíciót és ezt is eltároljuk. Beállítjuk a jármű setPosition függvényével ezt az új pozíciót, majd egy for ciklussal végigmegyünk az össze előttünk lévő járművön, hogy a jármű pozíciójával levonják a jármű hosszát és ha ez a hely kisebb mint az új pozícionk, akkor ütközünk a járművel (acceptCrash).

Ha a jármű végigment az úton (pos>1) és a régi pozíció kisebb mint 1 akkor meghívja az elől lévő TJ vehicleArrived függvényét. Ha a jármű vége is leért az útról, akkor meghívjuk a removeLastRoad függvényt.

+ double getIncomingVehicleTime() - Lekéri a jármű tömb első járművét, majd meghívja a getDistanceFromRoadend fv-t és ezt elosztja a jármű pillanatnyi sebességével.

+ double getDistanceFromRoadend(Vehicle vehicle) - Lekéri a jármű pozícióját és ezt levonja az út hosszából majd ebből levonja a roadEndLength-t.

8.1.19 Robber

- **Felelősség**

A bankrabló autója, a játékos irányítja, az autó vezetőjén (driverén) keresztül.

Az autó felelőssége többek között az ütközések kezelése valamint a saját sofőrrel (driver-rel) való kommunikáció megvalósítása.

- **Ősosztályok**

Vehicle

- **Interfészek**

nincs

- **Attribútumok**

- int lives - A játékos életeinek száma. Kezdetben 3, azonban ütközések alkalmával ez a szám csökken míg el nem éri a 0-t, ami egyben a játék sikertelen végét is jelenti a rabló számára.

- boolean wanted - A játékost körözik-e a rendőrök. A bankrablásnál vált true-ra, melynek a beállításáért a setWanted(boolean wanted) metódus felel.

- # boolean goingOutOfCity - Akkor igaz, ha a kocsi eleje már belement egy cityborderes kereszteződésbe, de a kocsi vége még nem ért ki a városból. A jármű még rajta van valamelyik Road-on. Értékének true-ra történő beállítása a cityBorderReached() metódus segítségével történik.

- # Game game – Egy referencia a Game objektumra. Szükség van rá akkor is mikor a robber elhagyja a várost és ezt jelezni szeretnénk a Game-nek, valamint mikor a robbert elkapták, vagy életének száma minimumra csökkent és ezt szeretnénk közölni a Game-mel.

- TrafficJunction cityBorderTrafficJunction - Az a kereszteződés, amelyiknél kilépett a rabló a városból. Szükséges hogy megjegyezzük, hiszen a visszatéréskor ide kell visszahelyeznünk a robbert a goBackToCity() függvénnyel.

- **Metódusok**

- + void goBackToCity() - Ez a függvény hívódik meg akkor, ha a rabló visszatér a városba. A függvény visszahelyezi a várost elhagyott rablót arra az útkereszteződésre ahol elhagyta a térképet, előtte viszont megvizsgálja az esetleges ütközések elkerülése végett az adott kereszteződés foglaltságát. Ha mindent rendben talált akkor egy az útkereszteződést elhagyó útra helyezi a rablót.

- + boolean getWanted() – A függvény egy get-ter metódus megvalósításaként visszatérési értékül szolgáltatja a wanted paraméter értékét, azaz hogy a rablót körözik-e már.

- + void setWanted(boolean wanted) –A függvény egy set-ter metódus megvalósításaként a wanted attribútum értékét állítja be a paraméterben átadott értéknek megfelelőre.

+ void setCityBorderTrafficJunction() – Beállíthatjuk, a rabló cityBorderTrafficJunction attributumát, hogy hol ment ki a városból. Ezt egy tipikus set-ter metódus végzi.

+ Road getFrontRoad() - A függvény visszaadja azt a Road-ot, amelyiken az autó eleje van. Ezt a roads.get(0) metódushívással éri el, mely visszaadja a roads tömb első elemét.

+ Road getDirection(Road[] roads) -A függvény segítségével kiválasztjuk a továbbhaladási irányt a következő módon. Először megvizsgáljuk hogy az adott jármű kifelé halad-e a városból, azaz hogy a GoingOutOfCity paramétere true-e (elvileg a rendőr nem haladhat kifelé a városból) . Ha már elkezdte elhagyni a várost akkor null-t adunk vissza, ha viszont nem akkor megkérdezzük a vezetőt, azaz meghívjuk a driver.chooseDirection(roads) függvényt. Ebben az esetben az ebből származó visszatérési érték lesz a függvény visszatérési értéke. Ez a visszatérési érték egy road, az a road amit a driver megfelelőnek talált a továbbhaladásra.

+ void addFrontRoad(Road road) -A függvény segítségével hozzáadjuk a Road[] listához azt az utat, amire az autó eleje rátért. Ezt a roads.add(0, road) metódushívással érjük el, mely a roads tömb első helyére teszi a paraméterként átadott road-ot.

+ void removeLastRoad() -A függvény segítségével levesszük a Road[] listáról legutolsó utat, amit az autó hátulja éppen elhagy. Ez a roads.remove(roads.size()-1) metódushívással történik. Ezt követően ellenőrzésre kerül hogy az autó esetleg rajta van-e még több úton. Ez a vizsgálat a roads tömb vizsgálatával egyenértékű. Ha úgy találtuk hogy a jármű nincs több úton rajta akkor meghívjuk this.outOfCity() metódusát.

+ double getVelocity() - A függvény egy get-ter metódus megvalósításaként visszaadja a jármű pillanatnyi sebességét (velocity).

+ void crash(Robber robber) - Meghívódik, ha a jármű ütközik, azaz ha a jármű belement egy Robber-be. Ha ez az eset fennállna akkor mind a két robber meghal, melyet jelzünk a Game-nek is. game.robberDied(this), game.robberDied(robber) metódushívásokkal.

+ void crash(Citizen citizen) - Meghívódik, ha a jármű ütközik, azaz ha a jármű belement egy Citizen-be. Ekkor a rabló élete lecsökken, valamint a vele ütközött citizen meghal, melyet jelzünk a game-nek this.game.killVehicle(citizen) metódushívással. Végül ellenőrizzük, hogy a robber élete nem csökkent a megengedett minimum alá, mert különben a robber meghal és ezt szintén jeleznünk kell a game-nek.game.robberDied(this)

+ void crash(Police police) - Meghívódik, ha a jármű ütközik, azaz ha a jármű belement egy Police-be. Ez az eset egyenértékű azzal mintha elkapták volna a rablót, így ezt jeleznünk kell a game-nek game.robberCaught(this) metódushívással.

+ void acceptCrash(Vehicle vehicle) - Meghívódik, ha a járműbe belementek. Ebben az esetben az adott jármű saját magát adja át annak aki őbelé belement, így

az ütközés további menetét ő fogja lekezelni. A függvényben ezért a `vehicle.crash(this)` függvényhívás történik.

+ `void outOfCity()` - Meghívódik, ha a jármű letért az összes Roadról (vagyis kiment a városból). Ebben az esetben a `goingOutOfCity` attribútum `false`-ra állítódik be, azaz már nem csak folyamatban van a város elhagyása hanem meg is történt az. Ezekután jelezzük a Game-nek hogy a rabló eltűnt `game.robberDisappeared(this)`, melynek következtében a körözés ezidőre megszűnik.

+ `VehicleType getType()` –A függvény egy `get`-ter metódus megvalósításaként visszaadja az autóra jellemző adatokat tartalmazó objektumot. (type)

+ `double getMaxVelocity()` –Visszaadja az autóra jellemző maximális sebességet, rabló esetén viszont ebbe beleszámít az életek száma, így kevesebb élet esetén kisebb sebességet fogunk kapni.

+ `double getMaxAcceleration()` – Visszaadja az autóra jellemző maximális gyorsulást, rabló esetén viszont ebbe beleszámít az életek száma, így kevesebb élet esetén kisebb gyorsulást fogunk kapni.

+ `void setDriver(IDriver driver)` – Ezzel lehet kiválasztani az autót vezető AI-t vagy emberi játékost. Jellemzően a rablót egy játékos irányítja.

+ `void cityBorderReacher()` –Ez hívódik meg, ha elérünk egy `CityBorderes TrafficJunction`-t. A metódus az adott citizen `goingOutOfCity` paraméterét állítja be `true`-ra, ezzel jelezve hogy az adott robber már elkezdte elhagyni a várost. Az adott útkereszteződést is eltároljuk.

+ `void doTick(int time, int deltatime)` – Ezen függvény hatására mozog a jármű az úton. Ez úgy történik hogy először lekérdezésre kerül a driver által meghatározott gyorsulás `driver.getAcceleration()`, ezt összevetjük az adott jármű maximális gyorsulásával `this.getMaxAcceleration()`. Ezután kiszámítjuk a sebességet az addigi sebességgel és a kiszámított gyorsulással kombinálva. Ezt is meg kell vizsgálni hogy nem haladtuk-e meg az adott járműre meghatározott maximális sebességet `this.getMaxVelocity()`. Eztkövetően végigmegyünk a roads tömb elemein, azaz azokon az utakon melyeken a jármű rajta van, majd mindegyiken léptetjük a járművet az adott road `road.moveVehicle(this, velocity*deltatime/1000)` függvényének meghívásával, melynek átadjuk magát a járművet, valamint a számított sebességet kombinálva a `deltatime`-mal mely meghatározza a mozgás mértékét.

+ `void acceptBuilding(Building building)` – A `trafficjunction` hívja meg a jármű ezen függvényét, hogy utána meglátogassa (`visit`) az ott lévő épületet, és lekezelje majd az épület, hogy mi történjen. Tudomásul adja az adott épületnek látogató jármű típusát. `building.visit(this)`

+ `void die()` -Öröklött metódus. A függvény meghívódik, ha az adott jármű meghal. Ekkor a függvény meghívása következtében az adott járművet a metódus leszedi az őt tartalmazó utakról. Ezt a `this.roads.get(0).removeVehicle(this)` végzi el.

8.1.20 Shelter

- **Felelősség**

A rejtekhely, mely különleges funkciókat tölt be a játék folyamán. Ha egy olyan rabló látogatja meg az épületet, mely már rabolt bankot akkor a rejtekhely felelőssége hogy elkezdje lekezelni, hogy véget érjen a játék, a rabló nyerjen, stb. Ha más járművek illetve még bankot nem rabolt robber-ek látogatják meg a rejtekhelyet, akkor nem történik semmi.

- **Össztályok**

Building

- **Interfészek**

nincs

- **Attribútumok**

Game game - Egy referencia a Game-re. Akkor van rá szükség ha már egy olyan robber látogatja meg az adott rejtekhelyet, mely már rabolt bankot. Ekkor ezt jelezni kell a Game-nek a game.robberReachedShelter(robber) metódus meghívásával.

- **Metódusok**

+ void visit(Robber robber) - Ha egy Robber érkezik a Shelterhez ez hívódik meg. Először ellenőrzésre kerül hogy a beérkezett rabló rabolt-e már ki bankot. Ennek ellenőrzése a robber.getWanted() függvényhívással történik. Ha kiderül hogy a rablót már körözték, azaz rabolt már bankot, akkor a game.robberReachedShelter(robber) metódus kerül meghívásra, ellenben ha még nem állt körözés alatt a rabló akkor nem történik semmi. Az utóbbi függvény meghívása a játék sikeres végét jelenti.

+ void visit(Citizen citizen) - Ha egy Citizen érkezik a Shelterhez ez hívódik meg. Ekkor nem történik semmi. Ezt a különböző épületek, melyeknek a Building az osztálya különböző módon kezelik le. Jelen esetben nincs rá szükség, mivel a játék szempontjából irreleváns, hogy ha egy rejtekhelyet egy citizen meglátogat.

+ void visit(Police police) - Ha egy Police érkezik a Shelterhez ez hívódik meg. Ekkor nem történik semmi. Ezt a különböző épületek, melyeknek a Building az osztálya különböző módon kezelik le. Jelen esetben nincs rá szükség, mivel a játék szempontjából irreleváns, hogy ha egy rejtekhelyet egy rendőr meglátogat.

8.1.21 StopSign

- **Felelősség**

Stop tábla, akkor mehet át az autó rajta ha előtte megáll, értesíti az ideérkező járművet a létezéséről.

- **Össztályok**
TrafficSign
- **Interfészek**
nincs
- **Attribútumok**
nincs
- **Metódusok**
 - + boolean accept(ITrafficRuleHandler trafficRuleHandler) – Szól az ideérkező jármű Driverének, hogy kezelje le, hogy stoptáblánál van. Ezt úgy teszi, hogy meghívja a jármű vezetőjének (ITrafficRuleHandler) visit függvényét

8.1.22 Timer

- **Felelősség**
A játék órája. Az időzítésért felelős. A protoban a threadet nem használjuk, mert a tesztesetek direkt úgy lettek kitalálva, hogy a tickek manuálisan történnek, így lehet tökéletesen látni a tesztesetek lényegét.
 - **Össztályok**
nincs
 - **Interfészek**
nincs
 - **Attribútumok**
 - ITick[] tickers - Azon objektumok, amelyek függnnek az idő múlásától. (pl.: autók, lámpák, stb.) Ezeket kezeljük folyamatosan.
 - double deltaTime - Ez az a delta érték amennyi időnként újra Tick-eljük az objektumokat.
 - int time – A starttól eltelt idő.
 - Thread thread – A szál amit a timer használ.
 - **Metódusok**
 - + void start() - Ez indítja el az időt (egy szálát). Meghívja a thread start függvényét. A time értékét 0-ra állítja.
 - + void stop() - Ez megállítja az időt. Meghívja a thread interrupt függvényét
 - + void run() - Ez futtatja a szálát. Egy végtelen ciklusban meghívja a tick függvényt majd a deltaTime időnyire elaltatja a szálát.
 - + void setDeltaTime(double deltaTime) - Itt lehet beállítani a fent leírt időközöket.
- A deltaTime setter függvénye
- + void addTicker(ITick ticker) – Ezzel lehet hozzáadni egy időre érzékeny objektumot. A ticker listához hozzáadja a paraméterben megadott tickert.
 - + void removeTicker(ITick ticker) - A ticker listáról leveszi a paraméterben megadott tickert.

+ void tick - megnöveli a time-t deltatime értékével. Végigmegy a tickers listán majd meghívja mindegyikre a doTick függvényt.

8.1.23 TrafficJunction

- **Felelősség**

Egy útkereszteződést testesít meg, ebbe vannak az Road-ok és más objektumok (Bank, Shelter).

- **Ősosztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

- Road[0..*] inRoads - A kereszteződésbe bemenő utak tömbje.
- Road[0..*] outRoads - A kereszteződésből kilépő utak tömbje.
- Building[0..*] buildings - A kereszteződésben található épületek tömbje (amikkel interakcióba léphet valami).
- double x - A kereszteződés x koordinátája.
- double y - A kereszteződés y koordinátája.

- **Metódusok**

+ void addBuilding(Building building) – Ezzel lehet a kereszteződéshez hozzáadni buildinget (Shelter, Bank, CityBorder stb.) A buildings tömbhöz hozzáadja a paraméterül kapott buildinget.

+ void addInRoad(Road inRoad) – Ezzel lehet bevezető utat hozzáadni a kereszteződéshez. A inroads tömbhöz hozzáadja a paraméterül kapott roadot.

+ void addOutRoad(Road outRoad) – Ezzel lehet kivezető utat hozzáadni a kereszteződéshez. A outroads tömbhöz hozzáadja a paraméterül kapott roadot.

+ boolean isOccupied() - A kereszteződés foglaltságát jelzi (tartózkodik-e jármű a kereszteződésben). Ezt úgy teszi, hogy csinál egy új listát az összes be és kivezető úttal, és végigmegy az összes úton kivéve amiről meghívtuk. Mindegyikre meghívja az isOccupied függvényt és ha valamelyik foglalt akkor igazgal tér vissza, különben hamissal.

+ void vehicleArrived(Vehicle vehicle) - Meghívódik amint érkezik egy autó a kereszteződésbe. Azt csinálja, hogy a TJ összes tartalmazott buildingjével meghívja a jármű acceptBuilding függvényét. Majd meghívja a jármű getDirection függvényét az outRoads tömbbel. Ha nem null ezek után a kiválasztott road akkor ahhoz hozzáadjuk a vehiclet.

+ double getX() - Visszatérési értékül adja a kereszteződés x koordinátáját.

+ double getY() - Visszatérési értékül adja a kereszteződés y koordinátáját.

+ Road[] getInRoads() - A függvény a kereszteződésbe belépő utak tömbjét adja vissza. Ez az inRoads getter függvénye.

+ Road[] getOutRoads() - A függvény a kereszteződésből kilépő utak tömbjét adja vissza. Ez az outRoads getter függvénye

+ void removeBuilding(Bulidingg building) - A buildings tömbből kitörlni a paraméterül kapott bulidinget.

+ Building[] getBuildings() - A függvény a kereszteződésben található Bulidingek tömbjét adja vissza. Ez az buildings getter függvénye.

8.1.24 TrafficLight

- **Felelősség**

Közlekedés irányító lámpa, periodikusan vált színt és értesíti az ideérkező járművet a létezéséről.

- **Ősosztályok**

TrafficSign

- **Interfészek**

ITick

- **Attribútumok**

- int period – Az az idő intervallum, amennyi időnként vált a lámpa.
- int startTime - Az az utolsó váltáskori idő.
- boolean green – Akkor igaz, ha a lámpa éppen zöld.

- **Metódusok**

+ boolean accept(ITrafficRuleHandler trafficRuleHandler) – Szól az ideérkező jármű Driverének, hogy kezelje le, hogy lámpánál van. Ezt úgy teszi, hogy meghívja a jármű vezetőjének (ITrafficRuleHandler) visit függvényét

+ boolean isGreen() – Lekérdezhető itt, hogy a lámpa éppen zöld-e. Lényegében egy Getter függvény. Ha lámpa zöld -> true ha piros -> false-al tér vissza.

+ void doTick(int time, int deltaTime) – Ezt a függvényt meghívva történik az idő múlása, lámpa váltása, ha éppen kell. Ha a time és a startTime különbsége kisebb mint a period akkor váltunk és updateljük a startTimet.

8.1.25 TrafficSign

- **Felelősség**

Közlekedési objektum, amiből öröklődnek a tábla, lámpa stb. osztályok.

Egy közlekedési szabályt fogalmaz meg. Ezt akkor használjuk amikor az autót mozgatjuk és ellenőrizzük, hogy van-e előttünk valamilyen közlekedési akadály ami alapján változtatni kell a sebességünket.

- **Ősosztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

nincs

- **Metódusok**

+ *boolean accept(ITrafficRuleHandler trafficRuleHandler)* – Szól az ideérkező jármű Driverének, hogy kezelje le, hogy valamilyen közlekedési objektumhoz ért. A konkrét implementáció a megfelelő származtatott osztályban található meg.

8.1.26 Vehicle

- **Felelősség**

Egy autót testesít meg, Felelőssége többek között az ütközések kezelése valamint a saját sofőrrel (driver-rel) való kommunikáció megvalósítása. Belőle származnak a különböző járművek (police, robber, citizen), melyek az itt megvalósított egyes függvényeket használják.

- **Ősosztályok**

nincs

- **Interfészek**

ITick

- **Attribútumok**

- driver: IDriver - Az autó vezetője, ő adja meg, hogy milyen gyorsulással és merre menjen az autó. Ha azt szeretnénk megtudni hogy mely uton haladjon tovább a jármű azaz hogy merre menjen akkor ezt a driver.chooseDirection(roads) hívással tudhatjuk meg. Ha a gyorsulást szeretnénk lekérdezni, akkor ehhez a driver.getAcceleration() függvényhívás szükséges. E paraméter határozza meg tulajdonképpen hogy ki vezeti az autót pl.: CitizenAI, PoliceAI vagy a játékos.

- double velocity - Az autó pillanatnyi sebessége. Az értéke a getVelocity() függvénnyel kérdezhető le.

- Road[] roads - Azon utak listája, amelyeken az autó pillanatnyilag rajta van. (Hosszú jármű v. kanyarodó út esetén fontos.) FIFO típusú lista. Szükség van rá többek között a getFrontRoad(), addFrontRoad(Road road), removeLastRoad() metódusok lefutásánál.

- VehicleType type - Az autó "tulajdonságait" adja meg, mely tulajdonképpen egy adott objektumra mutató referenciaként tárolódik. Értékének lekérdezéséhez a getType() metódus szükséges.

Game game – Egy referencia a Game objektumra. Szükség rá az adott osztályon belül nincsen, fontossága viszont a származtatott osztályok körében jelentkezik. pl.: ha a citizen jelezni szeretné a Game-nek hogy elhagyta a várost.

- boolean goingOutOfCity - Akkor igaz, ha a kocsi eleje már belement egy cityborderes kereszteződésbe, de a kocsi vége még nem ért ki a városból. A jármű még rajta van valamelyik Road-on. Értékének true-ra történő beállítása a cityBorderReached() metódus segítségével történik.

- **Metódusok**

- + `Road getFrontRoad()` - A függvény visszaadja azt a `Road`-ot, amelyiken az autó eleje van. Ezt a `roads.get(0)` metódushívással éri el, mely visszaadja a `roads` tömb első elemét.

- + `Road getDirection(Road[] roads)` - A függvény segítségével kiválasztjuk a továbbhaladási irányt a következő módon. Először megvizsgáljuk hogy az adott jármű kifelé halad-e a városból, azaz hogy a `GoingOutOfCity` paramétere `true`-e. Ha már elkezdte elhagyni a várost akkor `null`-t adunk vissza, ha viszont nem akkor megkérdezzük a vezetőt, azaz meghívjuk a `driver.chooseDirection(roads)` függvényt. Ebben az esetben az ebből származó visszatérési érték lesz a függvény visszatérési értéke. Ez a visszatérési érték egy `road`, az a `road` amit a `driver` megfelelőnek talált a továbbhaladásra.

- + `void addFrontRoad(Road road)` - A függvény segítségével hozzáadjuk a `Road[]` listához azt az utat, amire az autó eleje rátért. Ezt a `roads.add(0, road)` metódushívással érjük el, mely a `roads` tömb első helyére teszi a paraméterként átadott `road`-ot.

- + `void removeLastRoad()` - A függvény segítségével levesszük a `Road[]` listáról legutolsó utat, amit az autó hátulja éppen elhagy. Ez a `roads.remove(roads.size()-1)` metódushívással történik. Ezt követően ellenőrzésre kerül hogy az autó esetleg rajta van-e még több úton. Ez a vizsgálat a `roads` tömb vizsgálatával egyenértékű. Ha úgy találtuk hogy a jármű nincs több úton rajta akkor meghívjuk `this.outOfCity()` metódusát.

- + `double getVelocity()` - A függvény egy `get`-ter metódus megvalósításaként visszaadja a jármű pillanatnyi sebességét (`velocity`).

- + `void crash(Robber robber)` - Meghívódik, ha a jármű ütközik, azaz ha a jármű belement egy `Robber`-be. A függvény megvalósítása a származtatott osztályokra van bízva mivel ők tudják lekezelni az ütközést annak fényében hogy milyen fajta járművek ütköztek. Így ennek a metódusnak a törzse üresen van hagyva.

- + `void crash(Citizen citizen)` - Meghívódik, ha a jármű ütközik, azaz ha a jármű belement egy `Citizen`-be. A függvény megvalósítása a származtatott osztályokra van bízva mivel ők tudják lekezelni az ütközést annak fényében hogy milyen fajta járművek ütköztek. Így ennek a metódusnak a törzse üresen van hagyva.

- + `void crash(Police police)` -Meghívódik, ha a jármű ütközik, azaz ha a jármű belement egy `Police`-be. A függvény megvalósítása a származtatott osztályokra van bízva mivel ők tudják lekezelni az ütközést annak fényében hogy milyen fajta járművek ütköztek. Így ennek a metódusnak a törzse üresen van hagyva.

- + `abstract void acceptCrash(Vehicle vehicle)` -Meghívódik, ha a járműbe belementek. Ebben az esetben az adott jármű saját magát adja át annak aki őbelé belement. A függvény megvalósítása a származtatott osztályokra van bízva mivel ők tudják lekezelni az ütközést annak fényében hogy milyen fajta járművek ütköztek.

+ `void outOfCity()` - Meghívódik, ha a jármű letért az összes Roadról (vagyis kiment a városból). A függvény megvalósítása a származtatott osztályokra van bízva, mivel a különböző Vehicle-ből származtatott osztályok máshogy kezelik le fenti eseményt. Ebből kifolyólag a függvény itt nem kerül implementálásra.

+ `VehicleType getType()` – A függvény egy get-ter metódus megvalósításaként visszaadja az autóra jellemző adatokat tartalmazó objektumot. (type)

+ `double getMaxVelocity()` – Visszaadja az autóra jellemző maximális sebességet, ezt azonban csak úgy tudja megtenni hogy előtte lekérdezi az adott jármű típusát, amire a maximális sebesség jellemző, majd ennek lekérdezi a maximális sebességét a következőképpen: `this.getType().getMaxVelocity()`

+ `double getMaxAcceleration()` – Visszaadja az autóra jellemző maximális sebességet, ezt azonban csak úgy tudja megtenni hogy előtte lekérdezi az adott jármű típusát, amire a maximális gyorsulás jellemző, majd ennek lekérdezi a maximális gyorsulását a következőképpen: `this.getType().getMaxAcceleration()`

+ `void setDriver(IDriver driver)` – E függvény segítségével lehet kiválasztani az autót vezető AI-t vagy emberi játékost. Ezt a függvény a driver paraméter átállításával viszi véghez. Az így kiválasztott driver fogja irányítani a továbbiakban a járművet.

+ `void cityBorderReacher()` –Ez hívódik meg, ha elérünk egy CityBorderes TrafficJunctiont. A metódus az adott citizen goingOutOfCity paraméterét állítja be true-ra, ezzel jelezve hogy az adott citizen már elkezdte elhagyni a várost.

+ `void doTick(int time, int deltaTime)` – Ezen függvény hatására mozog a jármű az úton. Ez úgy történik hogy először lekérdezésre kerül a driver által meghatározott gyorsulás `driver.getAcceleration()`, ezt összevetjük az adott jármű maximális gyorsulásával `this.getMaxAcceleration()`. Ezután kiszámítjuk a sebességet az addigi sebességgel és a kiszámított gyorsulással kombinálva. Ezt is meg kell vizsgálni hogy nem haladtuk-e meg az adott járműre meghatározott maximális sebességet `this.getMaxVelocity()` .Eztkövetően végigmegyünk a roads tömb elemein, azaz azokon az utakon melyeken a jármű rajta van, majd mindegyiken léptetjük a járművet az adott road `road.moveVehicle(this, velocity*deltaTime/1000)` függvényének meghívásával, melynek átadjuk magát a járművet, valamint a számított sebességet kombinálva a deltaTime-mal mely meghatározza a mozgás mértékét.

+ `void acceptBuilding(Building building)` – A trafficjunction hívja meg a jármű ezen függvényét, hogy utána meglátogassa (visit) az ott lévő épületet, és lekezelje majd az épület, hogy mi történjen. Tudomásul adja az adott épületnek látogató jármű típusát. A függvény megvalósítása a származtatott osztályokra van bízva mivel a különböző járművektől függenek az adott épület meglátogatása által kiváltott történések. Ebből fakadóan a függvény itt nem került implementálásra.

+ `void die()` - A függvény meghívódik, ha az adott jármű meghal. Ekkor a függvény meghívása következtében az adott járművet a metódus leszedi az őt tartalmazó utakról. Ezt a `this.roads.get(0).removeVehicle(this)` végzi el.

8.1.27 VehicleOnRoad

- **Felelősség**

Egy járművet bedobozolunk. Ezen keresztül tárolja a Road osztály a járműveket. Itt tároljuk a jármű helyét az adott úton.

- **Ősosztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

- vehicle Vehicle - A jármű, amit tartalmaz a doboz. Ez azért fontos, hogy tudjuk, hogy melyik járművet mozgatjuk az úton.
- double position - A jármű pozíciója az adott úton. Azért kell, hogy ebben tároljuk, hogy minden egyes tick után éppen hol tart az úton a jármű

- **Metódusok**

+ double getPosition() - Getter függvény. Visszaadja a jármű position attribútumát.

+ void setPosition(double position) - Setter függvény. Beállíthatjuk a jármű pozícióját, például az autó mozgásánál az úton, ezt állítjuk.

+ Vehicle getVehicle() - Getter függvény. Visszaadja a tárolt járművet.

8.1.28 VehicleType

- **Felelősség**

Egy járműre jellemző tulajdonságokat tároljuk ebben az osztályban.

- **Ősosztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

- double length - A jármű hossza. Akkor szoktuk használni például amikor egy autót mozgatunk, akkor ezt használjuk annak számítására, hogy ütköztünk-e valakivel. Illetve ezt is használjuk akkor amikor ellenőrizzük, hogy leért-e már a jármű az útról.
- double maxVelocity - A jármű maximális sebessége. Ha megvan a gyorsulás akkor ezt használjuk, hogy nehogy túl gyorsra állítsuk a jármű sebességét.
- double maxAcceleration - A jármű maximális gyorsulása. A jármű gyorsulásának beállításakor használjuk.

- **Metódusok**

+ double getLength() - Getter függvény. Visszaadja a jármű length attribútumát.

- + double getMaxVelocity() - Getter függvény. Visszaadja a jármű maxVelocity attribútumát.
- + double getMaxAcceleration() - Getter függvény. Visszaadja a jármű maxAcceleration attribútumát.

8.2 A tesztek részletes tervei, leírásuk a teszt nyelvén

A TrafficJunction, illetve Road osztályokat igazából szinte minden teszt esetben teszteljük, de csak ott soroljuk fel a tesztelt osztályok között, ahol különös figyelmet fordítunk működésükre. Ezek a tesztesetek a program alapvető működését hivatottak tesztelni. Minden teszt esethez megadjuk a pályát leíró text fájlt, illetve ezt a pályát ábrázoló képet. Jelölések a képeken:

- Piros pont - TrafficJunction
- Fekete vonal - Road
- Cyan pont - A Road végét jelöli (egyirányú utcák vannak)
- Rózsaszín pont - StopSign
- Magenta pont - TrafficLight
- Narancssárga B - Bank
- Narancssárga S - Shelter
- Narancssárga C - CityBorder
- Narancssárga N - Bunny

A tesztek leíró nyelvhez még három parancsot hozzávettünk a tesztelés megkönnyítése érdekében:

playerSetWanted [0 vagy 1]

Leírás: Ha 1, akkor üldözik a Robbert a Police-ok, ha 0, akkor nem üldözik.

Opciók: -

playerGoBackToCity

Leírás: Ha a rabló kiment a városból, ezzel a paranccsal lehet visszamenni.

Opciók: -

tickMap [num]

Leírás: Olyan a hatása, mintha egy tick-et, majd egy showMap-et írnánk be, ezt ismétli num-szor.

Opciók: ha nem írunk be num-ot, akkor eggyel léptet.

Apró változtatás a showMap parancsban: Ha egy jármű a városon kívül van (tehát nincs semmilyen úton rajta), akkor "[VehicleSzám]:[VehicleTípus] is out of city." üzenet kapjuk. A tick, illetve a tickMap parancsnál kiírja: <<Tick [a tick sorszáma]>> minden lépés előtt.

A konzolos kimeneten (amit a tesztelő lát) nem csak az üzenetek jelennek meg, hanem a bemeneti parancsok is. Így a teszt eredményei áttekinthetőbbek, mivel látjuk, hogy az adott üzenet melyik parancs hatására íródott ki, viszont az out.txt-ben (ebbe is kiíródik a kimenet, ezt fogja összehasonlítani a komparáló program a referenciával) ezek nem jelennek meg.

8.2.1 Teszteset0

- **Leírás**

Egy Citizen tesz egy "lépést" az úton egy tick hatására, a program legalapvetőbb működésének tesztelése.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

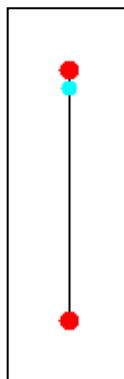
A járművek mozgásának tesztelése. Road, Citizen (és ugyanígy Robber, Police) osztályok tesztelése. A Road.moveVehicle metódus bonyolultabb logikájában esetleg lehet hiba.

- **Bemenet**

```
loadMap ptest0map.txt
setRand 0
addVehicle C 0 0.3
showMap
setDeltaTime 200
tickMap 4
```

ptest0map.txt:

```
2
312 188;
312 313;
1
1 0;
```



- **Elvárt kimenet**

```
#1:Citizen has turned to #2:Road.
#1:Citizen is on #2:Road at 0,30 position, moving with 0,00 velocity.
<< Tick 1 >>
#1:Citizen is on #2:Road at 0,31 position, moving with 10,00 velocity.
<< Tick 2 >>
#1:Citizen is on #2:Road at 0,32 position, moving with 20,00 velocity.
<< Tick 3 >>
#1:Citizen is on #2:Road at 0,35 position, moving with 30,00 velocity.
<< Tick 4 >>
#1:Citizen is on #2:Road at 0,38 position, moving with 40,00 velocity.
```

8.2.2 Teszteset1

- **Leírás**

A Robber eléri a Bank épületet (így kirabolja), aminek hatására a Police üldözni kezdi őt. A Robber eléri a Shelter épületet, aminek hatására megnyeri a játékot.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

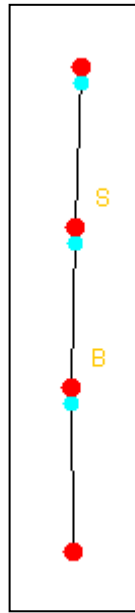
A bankrablás és az üldözés létrejöttének tesztelése, a búvóhely elérésének és a játék megnyerésének tesztelése. Robber, Bank, Police, PoliceAI osztályok tesztelése. A PoliceAI "üldöző" algoritmusában várható hiba, mivel ez bonyolult logikát tartalmaz.

- Bemenet**

```
loadMap ptest1map.txt
setRand 0
addVehicle R 0 0.99
addVehicle P 0 0.00
showMap
playerSetAcc 60
playerSetDirection 0
setDeltaTime 500
tick
setDeltaTime 1500
tick 2
```

ptest1map.txt:

```
3
299 414;
298 332;B
300 252;S
2
0 1;
1 2;
```



- Elvárt kimenet**

```
#1:Robber has turned to #2:Road.
#3:Police has turned to #2:Road.
#1:Robber is on #2:Road at 0,99 position, moving with 0,00 velocity.
#3:Police is on #2:Road at 0,00 position, moving with 0,00 velocity.
<< Tick 1 >>
#1:Robber's front has reached the end of the #2:Road.
#1:Robber has robbed the #4:Bank.
#3:Police has been activated.
#1:Robber has turned to #5:Road.
#1:Robber's end has reached the end of the #2:Road.
<< Tick 1 >>
#1:Robber has left #6:TrafficJunction.
#3:Police has left #7:TrafficJunction.
<< Tick 2 >>
#1:Robber has entered #8:TrafficJunction.
#1:Robber's front has reached the end of the #5:Road.
#1:Robber has reached the #9:Shelter.
#1:Robber has won (reached the Shelter).
#1:Robber has turned to #10:Road.
#1:Robber's end has reached the end of the #5:Road.
#3:Police has entered #6:TrafficJunction.
```

8.2.3 Teszteset2

- **Leírás**

A Robbert körözik, aminek hatására a Police üldözni kezdi őt. A Police ütközik a Robberrel (mivel a Robber áll), aminek hatására a Robber elveszti a játékot.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

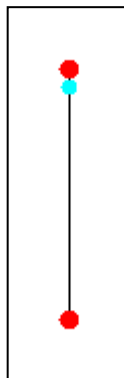
Az üldözés létrejöttének tesztelése, az ütközés és a játék elvesztésének tesztelése. A Robber, Police (Police-on belül az ütközés pluszban, az előző tesztesethez képest), PoliceAI osztályok tesztelése. A PoliceAI "üldöző" algoritmusában várható hiba (getAcceleration illetve chooseDirection fv), mivel ez bonyolult logikát tartalmaz.

- **Bemenet**

```
loadMap ptest0map.txt
setRand 0
addVehicle R 0 0.3
addVehicle P 0 0.2
showMap
playerSetWanted 1
setDeltaTime 1000
tick
```

ptest0map.txt:

```
2
312 188;
312 313;
1
1 0;
```



- **Elvárt kimenet**

```
#1:Robber has turned to #2:Road.
#3:Police has turned to #2:Road.
#1:Robber is on #2:Road at 0,30 position, moving with 0,00 velocity.
#3:Police is on #2:Road at 0,20 position, moving with 0,00 velocity.
#3:Police has been activated.
<< Tick 1 >>
#3:Police has crashed with #1:Robber.
#1:Robber has lost (been caught).
#1:Robber has died.
```

8.2.4 Teszteset3

- **Leírás**

Egy jármű (Citizen, minden másik fajta is ugyanígy viselkedik, kivéve, hogy nem veszi figyelembe a táblákat) mozog az úton, meglát egy lámpát, ami piros, lelassít, majd ha zöld akkor behajt a kereszteződésbe (nem megy át rajta).

- **Ellenőrzött funkcionalitás, várható hibahelyek**

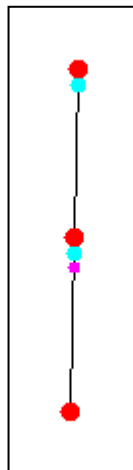
Teszteli, hogy a járművek a közlekedési szabályokat betartják-e, a Citizen hogyan lassít le, ha akadály van előtte, a lámpa megfelelően vált-e piros és zöld állapot között. A Citizen, CitizenAI, TrafficLight, TrafficJunction osztályok tesztelése. A CitizenAI getAcceleration, illetve a TrafficJunction vehicleArrived függvényében várható hiba.

- **Bemenet**

```
loadMap ptest3map.txt
setRand 0
addVehicle C 0 0.7
showMap
setDeltaTime 500
tickMap 4
```

ptest3map.txt:

```
3
307 402;
309 315;
311 231;
2
0 1;L
1 2;
```



- **Elvárt kimenet**

```
#1: Citizen has turned to #2: Road.
#1: Citizen is on #2: Road at 0,70 position, moving with 0,00 velocity.
<< Tick 1 >>
#1: Citizen is on #2: Road at 0,77 position, moving with 25,00 velocity.
<< Tick 2 >>
#1: Citizen is on #2: Road at 0,83 position, moving with 18,78 velocity.
<< Tick 3 >>
#1: Citizen is on #2: Road at 0,85 position, moving with 9,83 velocity.
<< Tick 4 >>
#3: TrafficLight has switched to GREEN.
#1: Citizen has entered #4: TrafficJunction.
#1: Citizen is on #2: Road at 0,95 position, moving with 34,83 velocity.
```

8.2.5 Teszteset4

- **Leírás**

A Citizen stoptáblához érkezik és érzékeli, hogy a kereszteződésben jön egy másik autó, ezért megvárja, amíg az áthalad, addig ő nem megy tovább.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

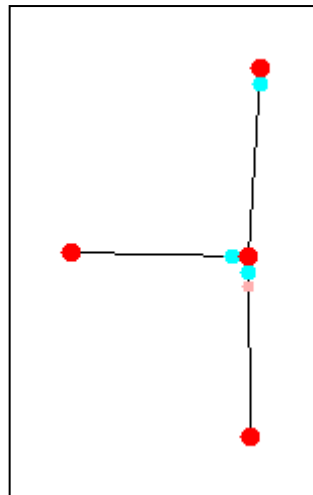
A Citizen megfelelő döntésének tesztelése a Stop táblánál, illetve teszteli, hogy a Citizen megfelelően érzékeli-e más autók kereszteződéshez közeledését. A StopSign, Citizen, CitizenAI osztályok tesztelése. A CitizenAI visit(StopSign) fvében várható hiba.

- **Bemenet**

```
loadMap ptest4map.txt
setRand 0
addVehicle C 0 0.7
addVehicle C 1 0.3
showMap
setDeltaTime 500
tickMap 2
setDeltaTime 1000
tickMap 3
setDeltaTime 500
tickMap 2
```

ptest4map.txt:

```
4
311 330;
310 240;
222 238;
316 146;
3
0 1;S
2 1;
1 3;
```



- **Elvárt kimenet**

```
#1:Citizen has turned to #2:Road.
#3:Citizen has turned to #4:Road.
#1:Citizen is on #2:Road at 0,70 position, moving with 0,00 velocity.
#3:Citizen is on #4:Road at 0,30 position, moving with 0,00 velocity.
<< Tick 1 >>
#1:Citizen is on #2:Road at 0,77 position, moving with 25,00 velocity.
#3:Citizen is on #4:Road at 0,37 position, moving with 25,00 velocity.
<< Tick 2 >>
#1:Citizen is on #2:Road at 0,84 position, moving with 26,88 velocity.
#3:Citizen is on #4:Road at 0,51 position, moving with 50,00 velocity.
<< Tick 1 >>
#1:Citizen is on #2:Road at 0,88 position, moving with 6,20 velocity.
#3:Citizen is on #4:Road at 0,80 position, moving with 50,00 velocity.
<< Tick 2 >>
#3:Citizen has entered #5:TrafficJunction.
#3:Citizen's front has reached the end of the #4:Road.
```

```

#3:Citizen has turned to #6:Road.
#3:Citizen's end has reached the end of the #4:Road.
#1:Citizen is on #2:Road at 0,89 position, moving with 1,43 velocity.
#3:Citizen is on #6:Road at 0,00 position, moving with 50,00 velocity.
<< Tick 3 >>
#3:Citizen has left #5:TrafficJunction.
#1:Citizen is on #2:Road at 0,89 position, moving with 0,00 velocity.
#3:Citizen is on #6:Road at 0,27 position, moving with 50,00 velocity.
<< Tick 1 >>
#1:Citizen has entered #5:TrafficJunction.
#1:Citizen is on #2:Road at 0,96 position, moving with 25,00 velocity.
#3:Citizen is on #6:Road at 0,40 position, moving with 50,00 velocity.
<< Tick 2 >>
#1:Citizen's front has reached the end of the #2:Road.
#1:Citizen has turned to #6:Road.
#1:Citizen's end has reached the end of the #2:Road.
#1:Citizen is on #6:Road at 0,00 position, moving with 50,00 velocity.
#3:Citizen is on #6:Road at 0,53 position, moving with 50,00 velocity.

```

8.2.6 Teszteset5

- Leírás**

Egy hármas kereszteződésben egyik útról a másikra halad át a tesztelő (Robber), ennek hatására a harmadik úton lévő Citizen érzékeli, hogy foglalt a kereszteződés, ezért vár és nem megy bele a kereszteződésbe.

- Ellenőrzött funkcionalitás, várható hibahelyek**

Teszteli, hogy a kereszteződés megfelelően rakja-e át egyik útról a másikra az autókat, illetve a kereszteződés megfelelően dönti-e el foglaltságát és ezt a Citizen is megfelelően érzékeli-e. Citizen, CitizenAI, TrafficJunction osztályok tesztelése. A TrafficJunction isOccupied metódusában várható hiba.

- Bemenet**

```

loadMap ptest5map.txt
setRand 0
addVehicle R 0 0.8
addVehicle C 1 0.7
showMap
playerSetAcc 40
playerSetDirection 0
setDeltaTime 500
tickMap 4

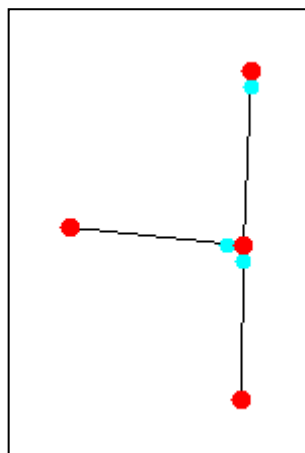
```

ptest5map.txt

```

4
306 345;
307 268;
311 181;
221 259;
3
3 1;
0 1;
1 2;

```



- **Elvárt kimenet**

```
#1:Robber has turned to #2:Road.
#3:Citizen has turned to #4:Road.
#3:Citizen is on #4:Road at 0,70 position, moving with 0,00 velocity.
#1:Robber is on #2:Road at 0,80 position, moving with 0,00 velocity.
<< Tick 1 >>
#3:Citizen is on #4:Road at 0,78 position, moving with 25,00 velocity.
#1:Robber is on #2:Road at 0,86 position, moving with 20,00 velocity.
<< Tick 2 >>
#1:Robber has entered #5:TrafficJunction.
#3:Citizen is on #4:Road at 0,81 position, moving with 8,29 velocity.
#1:Robber is on #2:Road at 0,97 position, moving with 40,00 velocity.
<< Tick 3 >>
#1:Robber's front has reached the end of the #2:Road.
#1:Robber has turned to #6:Road.
#1:Robber's end has reached the end of the #2:Road.
#3:Citizen is on #4:Road at 0,82 position, moving with 4,34 velocity.
#1:Robber is on #6:Road at 0,00 position, moving with 60,00 velocity.
<< Tick 4 >>
#1:Robber has left #5:TrafficJunction.
#3:Citizen is on #4:Road at 0,83 position, moving with 2,27 velocity.
#1:Robber is on #6:Road at 0,17 position, moving with 60,00 velocity.
```

8.2.7 Teszteset6

- **Leírás**

Az úton a tesztelő autója (Robber), majd utána egy Citizen van. A tesztelő lassít, majd megáll és ennek hatására a mögötte lévő Citizen is lelassít, majd megáll, hogy az ütközést elkerülje.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Annak tesztelése, hogy a Citizen nem ütközik, ha az előtte lévő autó akármennyire is lelassít vagy megáll. Citizen, CitizenAI osztályok tesztelése. A CitizenAI getAcceleration függvényében várható hiba.

- **Bemenet**

```
loadMap ptest0map.txt
setRand 0
addVehicle R 0 0.4
addVehicle C 0 0.05
showMap
playerSetAcc 40
playerSetDirection 0
setDeltaTime 1000
tick
playerSetAcc -15
setDeltaTime 900
tickMap 6
```



ptest0map.txt:

```
2
312 188;
312 313;
1
1 0;
```

- **Elvárt kimenet**

```
#1:Robber has turned to #2:Road.
#3:Citizen has turned to #2:Road.
#3:Citizen is on #2:Road at 0,05 position, moving with 0,00 velocity.
#1:Robber is on #2:Road at 0,40 position, moving with 0,00 velocity.
<< Tick 1 >>
#3:Citizen has left #4:TrafficJunction.
<< Tick 1 >>
#3:Citizen is on #2:Road at 0,43 position, moving with 50,00 velocity.
#1:Robber is on #2:Road at 0,66 position, moving with 26,50 velocity.
<< Tick 2 >>
#3:Citizen is on #2:Road at 0,61 position, moving with 50,00 velocity.
#1:Robber is on #2:Road at 0,70 position, moving with 13,00 velocity.
<< Tick 3 >>
#3:Citizen is on #2:Road at 0,63 position, moving with 6,44 velocity.
#1:Robber is on #2:Road at 0,70 position, moving with 0,00 velocity.
<< Tick 4 >>
#3:Citizen is on #2:Road at 0,64 position, moving with 1,80 velocity.
#1:Robber is on #2:Road at 0,70 position, moving with 0,00 velocity.
<< Tick 5 >>
#3:Citizen is on #2:Road at 0,64 position, moving with 0,50 velocity.
#1:Robber is on #2:Road at 0,70 position, moving with 0,00 velocity.
<< Tick 6 >>
#3:Citizen is on #2:Road at 0,64 position, moving with 0,00 velocity.
#1:Robber is on #2:Road at 0,70 position, moving with 0,00 velocity.
```

8.2.8 Teszteset7

- **Leírás**

Az úton a tesztelő autója (Robber), előtte egy Citizen van. A tesztelő gyorsít, majd ütközik a Citizennel. A Citizen így tönkremegy és eltűnik az útról, a Robber életet veszít. A Robber halad tovább az úton.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A Robber és Citizen ütközésének megfelelő lekezelése. A Robber, Citizen osztályok tesztelése. A Robber crash fvében várható hiba.

- **Bemenet**

```
loadMap ptest0map.txt
setRand 0
addVehicle R 0 0.3
addVehicle C 0 0.5
showMap
playerSetAcc 60
playerSetDirection 0
setDeltaTime 1000
tick 2
showMap
```

ptest0map.txt:

```
2
312 188;
312 313;
1
1 0;
```



- **Elvárt kimenet**

```
#1:Robber has turned to #2:Road.
#3:Citizen has turned to #2:Road.
#3:Citizen is on #2:Road at 0,50 position, moving with 0,00 velocity.
#1:Robber is on #2:Road at 0,30 position, moving with 0,00 velocity.
<< Tick 1 >>
#1:Robber has crashed with #3:Citizen, and has 2 lives left.
#3:Citizen has died.
<< Tick 2 >>
#1:Robber is on #2:Road at 0,74 position, moving with 50,00 velocity.
```

8.2.9 Teszteset8

- **Leírás**

Egy Citizen kimegy a városból, aminek hatására a város egy másik pontján egy másik Citizen jön be.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

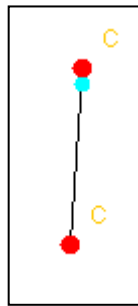
Teszteli, hogy az autók város elhagyását helyesen érzékeleli a program, és azt, hogy jól rakja-e le a térképre az új autókat. Citizen, Game osztályok tesztelése. A Game citizenLeftCity fvében várható hiba.

- **Bemenet**

```
loadMap ptest8map.txt
setRand 0
addVehicle C 0 0.8
showMap
setDeltaTime 1000
tickMap
```

ptest8map.txt:

```
2
303 324;C
309 236;C
1
0 1;
```



- **Elvárt kimenet**

```
#1:Citizen has turned to #2:Road.
#1:Citizen is on #2:Road at 0,80 position, moving with 0,00 velocity.
<< Tick 1 >>
#1:Citizen has entered #3:TrafficJunction.
#1:Citizen's front has reached the end of the #2:Road.
#1:Citizen is going out of city.
#1:Citizen's end has reached the end of the #2:Road.
#1:Citizen left city.
#1:Citizen has died.
#4:Citizen has turned to #2:Road.
#4:Citizen is on #2:Road at 0,00 position, moving with 0,00 velocity.
```

8.2.10 Teszteset9

- **Leírás**

A tesztelőt (Robber) üldözik, ekkor eléri a CityBorder épületet, ennek hatására kilép a városból. Amikor kiért, akkor a rendőrök nem üldözik tovább. A Robbet visszatér a városba ugyanazon a ponton és a rendőrök megint elkezdik őt üldözni.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

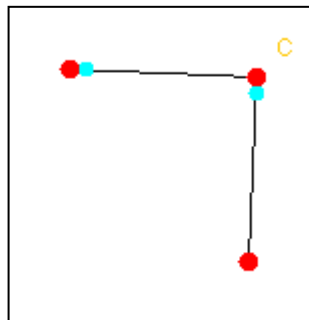
Annak tesztelése, hogy a modell megfelelően kezeli-e le, amikor a Robber elhagyja a várost, ezzel a rendőrök figyelmét elveszti, majd visszatérve visszakapja azt. Robber, CityBorder, Game osztályok tesztelése. A Game robberDisappeared fvében és a Robber goBackToCity fvében várható hiba.

- **Bemenet**

```
loadMap ptest9map.txt
setRand 0
addVehicle R 0 0.95
addVehicle P 0 0.2
showMap
playerSetAcc 40
playerSetDirection 0
playerSetWanted 1
setDeltaTime 1000
tick
showMap
playerGoBackToCity
```

ptest9map.txt:

```
3
313 327;
317 235;C
224 231;
2
0 1;
1 2;
```



- **Elvárt kimenet**

```
#1:Robber has turned to #2:Road.
#3:Police has turned to #2:Road.
#1:Robber is on #2:Road at 0,95 position, moving with 0,00 velocity.
#3:Police is on #2:Road at 0,20 position, moving with 0,00 velocity.
#3:Police has been activated.
<< Tick 1 >>
#1:Robber's front has reached the end of the #2:Road.
#1:Robber has reached the #4:CityBorder.
#1:Robber is going out of city.
#1:Robber's end has reached the end of the #2:Road.
#1:Robber has left city.
#3:Police has been deactivated.
#1:Robber is out of city.
#3:Police is on #2:Road at 0,47 position, moving with 50,00 velocity.
#1:Robber is going back to city.
#1:Robber has turned to #5:Road.
#3:Police has been activated.
```

8.2.11 Teszteset10

- **Leírás**

Egy meglehetősen nagy pályán kipróbálja a programot és megméri, hogy mennyi idő alatt hajtotta végre a műveleteket.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A program megfelelően gyors működésének tesztelése, az algoritmusok komplexitásának ellenőrzése.

- **Bemenet**

Ettől eltekintünk (rengeteg Citizen, több Police, egy Robber van a térképen).

- **Elvárt kimenet**

Akkora időtartam, ami még a kényelmes használatot garantálja.

8.2.12 Teszteset11

- **Leírás**

A Robber-t (tesztelő) üldözik, ekkor rámegy a nyuszira, ami miatt egy ideig védett lesz, vagyis nem üldözik a Police-ok tovább. Miután lejárt a védelem ideje a Police-ok ismét üldözőbe veszik (a védettségi időtartam rövid lesz a tesztelés megkönnyítése érdekében).

- **Ellenőrzött funkcionalitás, várható hibahelyek**

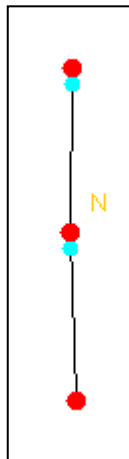
A nyuszi funkcióinak tesztelése. A Bunny, Robber, Police osztályok tesztelése. A Bunny osztály visit, illetve doTick fvében várható hiba.

- **Bemenet**

```
loadMap ptest11map.txt
setRand 0
addVehicle R 0 0.95
addVehicle P 0 0.2
showMap
playerSetWanted 1
playerSetAcc 60
playerSetDirection 0
setDeltaTime 500
tick 3
showMap
```

ptest11map.txt:

```
3
297 350;
294 266;N
295 184;
2
0 1;
1 2;
```



- **Elvárt kimenet**

```
#1:Robber has turned to #2:Road.
#3:Police has turned to #2:Road.
#1:Robber is on #2:Road at 0,95 position, moving with 0,00 velocity.
#3:Police is on #2:Road at 0,20 position, moving with 0,00 velocity.
#3:Police has been activated.
<< Tick 1 >>
#1:Robber's front has reached the end of the #2:Road.
#1:Robber met #4:Bunny, #1:Robber is protected now!
#3:Police has been deactivated.
#1:Robber has turned to #5:Road.
<< Tick 2 >>
#1:Robber has left #6:TrafficJunction.
#1:Robber's end has reached the end of the #2:Road.
<< Tick 3 >>
Bunny's protection is over, #1:Robber is no longer protected.
#3:Police has been activated.
#1:Robber is on #5:Road at 0,37 position, moving with 60,00 velocity.
#3:Police is on #2:Road at 0,57 position, moving with 50,00 velocity.
```

8.3 A tesztelést támogató programok tervei

A tesztadatokat a Proto állítja elő, úgy hogy parancssorból megadott fájlból ("gtb_run.bat <tesztfájl neve>", pl: "gtb_run ptest0com.txt") legyártja az "out.txt"-t, ami már csak a teszt kimenetét tartalmazza. Ezt a fájlt összehasonlítjuk egy előre legyártott fájlal, ami a helyes kimenetet tartalmazza. A két szöveges fájl összehasonlítását a DiffUtils diff parancsával végezzük, ami kiírja, hogy mi a különbség a 2 fájl között.

A tesztelés elősegítése érdekében egy automatikus tesztelő parancsfájlt hozunk létre, ami leteszteli az összes tesztfájlt és leellenőrzi a kimeneteket a helyes értékekkel. Ennek a parancsfájlnak 'gtb_test.bat' lesz a neve. Ezt elindítva a következő kimenetet kapjuk:

```
***** <teszteset száma>. teszt *****
<teszt eredménye>
```

Ahol a <teszteset száma> = 0 .. 11

Míg a <teszt eredménye> =

- a) "A teszt sikeresen lezajlott!", ha a teszt sikeresen zárult.
- b) Minden más kiírás esetén hiba lépett fel, ami a "diff" program hibaüzenete, erről a program honlapján[1] lehet tájékozódni.

A hibaüzeneteken kívül a "diff" program még a tesztek közötti eltérést is mutatja, aminek a formátuma a következő:

```
"
< #1:Citizen is on #2:Road at 0,30 position, moving with 0,00 velocity.
---
> #1:Citizen is on #2:Road at 1,30 position, moving with 0,00 velocity.
"
```

Ahol a "<"-lal kezdődő sorok a helyesek, míg a ">"-lal kezdődő sorok a proto által generált eltérő sorok (a példa egy olyan esetet mutat, ahol az autó az útnak a 130%-án halad, tehát lement az útról és ezt feltételezhetően nem kezelte le megfelelően a program).

[1] http://www.gnu.org/software/diffutils/manual/html_node/Output-Formats.html#Output%20Formats

MapEditor

Pálya szerkeszthető vele: TrafficJunction-ök, Road-ok, Building-ek, TrafficSign-ok elhelyezése. Fájlba mentés a fent megadott formátumban. A pályák szerkesztése egy grafikus interfészen, egérrel, gombokkal történik a fent megadott képszimbólumokkal. A program egy Canvasra rajzolja a pálya képét, a képet fájlba is tudja menteni.

8.4 A specifikáció változása miatti korrekció

A változás szerint: *A húsvéti nyuszi időnként feltűnik az utakon. Ha a rabló elüti a nyuszt, akkor rövid időre immunitást élvez, vagyis a rendőrök ekkor nem tudják elkapni, bárhogyan próbálkoznak is.*

Ezt úgy oldottuk meg, hogy a felvettünk egy új Building-et, amit Bunny névre kereszteltünk. Ezt ugyanúgy meg lehet látogatni, mint a többit. Ez a Building a Robber látogatása esetén deaktiválja az összes Police-t és eltűnik a térképről, majd a védeltségi idő elteltével újra aktiválja őket, ez után megsemmisíti önmagát. Az időt úgy számolja, hogy implementálja a ITick interfészt.

8.5 Napló

Kezdet	Idő	Résztvevők	Leírás
2010.03.28. 11:00	30 perc	Angeli Koczka Kovács Mikó	Értekezlet. Döntés: Angeli és Mikó megírja az osztályok és metódusok terveit, Koczka és Kovács megírja a tesztek részletes terveit
2010.03.28. 11:30	1 óra	Kovács	Tesztesetek felvázolása
2010.03.28. 15:00	2 óra	Koczka Kovács	Tesztesetek részletes terveinek írása
2010.03.28. 18:00	4 óra	Angeli	Osztályok és metódusok terveinek írása
2010.03.29. 16:00	2,5 óra	Koczka Kovács	Tesztesetek részletes terveinek, Specifikáció változása miatti korrekció, Tesztelést támogató programok tervei írása
2010.03.29. 20:00	2 óra	Mikó	Osztályok és metódusok terveinek írása
2010.03.30. 10:00	2 óra	Koczka Kovács	Tesztesetek részletes terveinek írása
2010.03.30. 10:00	3 óra	Angeli Mikó	Osztályok és metódusok terveinek írása
2010.03.30. 17:00	1 óra	Koczka Kovács	Tesztesetek részletes terveinek írása
2010.03.31. 11:00	3 óra	Angeli Mikó	Osztályok és metódusok terveinek írása
2010.03.31. 22:00	1 óra	Mikó	Dokumentum formázása

10. Prototípus beadása

10.1 Fordítási és futtatási útmutató

10.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
src/GrandTheftBank/Bank.java	855 b	2010.03.29. 13:29:31	A bankot megvalósító osztály
src/GrandTheftBank/Bunny.java	2727 b	2010.04.15. 01:21:28	A Bunnyt megvalósító osztály
src/GrandTheftBank/Building.java	1533 b	2010.03.29. 13:29:31	Az épületek alaposztálya
src/GrandTheftBank/Citizen.java	2283 b	2010.03.31. 16:36:14	A városi autót megvalósító osztály
src/GrandTheftBank/CitizenAI.java	7093 b	2010.04.14. 11:00:47	A városi autót irányító algoritmust megvalósító osztály
src/GrandTheftBank/CityBorder.java	1055 b	2010.03.29. 13:29:31	A városhatárt megvalósító osztály
src/GrandTheftBank/DriverAdapter.java	1445 b	2010.03.31. 14:33:28	Segéd adapter az IDriver interfésznek
src/GrandTheftBank/Game.java	7665 b	2010.04.15. 01:21:28	A játék irányítását megvalósító osztály
src/GrandTheftBank/IDriver.java	743 b	2010.03.31. 14:33:28	Az autót irányító algoritmus interfésze
src/GrandTheftBank/ITick.java	714 b	2010.03.29. 13:29:31	Az időt érzékelő elem interfésze
src/GrandTheftBank/ITrafficRuleHandler.java	1066 b	2010.03.29. 13:29:31	A közlekedési szabályt érzékelő elem interfésze
src/GrandTheftBank/Main.java	3248 b	2010.04.14. 12:29:46	A program futását lekezelő osztály
src/GrandTheftBank/Map.java	9847 b	2010.04.14. 11:00:47	A játék térképét tartalmazó osztály
src/GrandTheftBank/Police.java	4251 b	2010.04.15. 01:21:28	A rendőrautót megvalósító osztály
src/GrandTheftBank/PoliceAI.java	9486 b	2010.04.14. 11:00:47	A rendőr autót irányító algoritmust megvalósító osztály
src/GrandTheftBank/ProtoDriver.java	1406 b	2010.04.14. 11:00:47	A rabló konzolos felületről való irányítását lehetővé tevő osztály.
src/GrandTheftBank/ProtoGame.java	6832 b	2010.04.15. 01:21:28	A prototípus konzolos, parancssor alapú játékként való működését lehetővé tevő osztály.
src/GrandTheftBank/ProtoHelper.java	8225 b	2010.04.15. 01:21:28	Konzolra, fájlba való kiírást, konzolról való parancsértelmezést segítő osztály.
src/GrandTheftBank/	10482 b	2010.04.15.	Az utat megvalósító osztály

Road.java		01:21:28	
src/GrandTheftBank/ Robber.java	5093 b	2010.04.15. 01:21:28	A rabló autóját megvalósító osztály
src/GrandTheftBank/ Shelter.java	849 b	2010.03.29. 13:29:31	A rejtékhelyet megvalósító osztály
src/GrandTheftBank/ StopSign.java	511 b	2010.03.29. 13:29:31	A stop táblát megvalósító osztály
src/GrandTheftBank/ Timer.java	2135 b	2010.04.15. 01:21:28	Az időzítőt megvalósító osztály
src/GrandTheftBank/ TrafficJunction.java	4091 b	2010.04.15. 01:21:28	Az útkereszteződést megvalósító osztály
src/GrandTheftBank/ TrafficLight.java	2683 b	2010.04.15. 01:21:28	A közlekedési lámpát megvalósító osztály
src/GrandTheftBank/ TrafficSign.java	494 b	2010.03.29. 13:29:31	A közlekedési szabályok ösosztálya
src/GrandTheftBank/ Vehicle.java	6397 b	2010.04.14. 11:00:47	Az autót megvalósító ösosztály
src/GrandTheftBank/ VehicleOnRoad.java	1192 b	2010.03.29. 13:29:31	Tároló osztály az autók úton való elhelyezésére
src/GrandTheftBank/ VehicleType.java	1294 b	2010.03.29. 13:29:31	Az autómódelleket megvalósító osztály
gtb.bat	61 b	2010.03.14. 20:58:47	Lefordítja a projektet, elkészíti a dokumentációt és leteszteli a teszteseteket
gtb_compile.bat	58 b	2010.03.14. 20:16:01	Lefordítja a projektet
gtb_javadoc.bat	92 b	2010.03.14. 20:33:03	Elkészíti a dokumentációt
gtb_run.bat	47 b	2010.03.14. 20:17:26	Elindítja a programot
gtb_test.bat	288 b	2010.03.31. 12:22:25	Leteszteli a teszteseteket
diff.exe	150528 b	2010.03.31. 12:10:24	Segédprogram: összehasonlítja a helyes kimenetet a teszt futásának eredményével
libiconv2.dll	898048 b	2010.03.31. 12:08:18	diff.exe futtatásához szükséges fájl
libintl3.dll	92672 b	2010.03.31. 12:08:18	diff.exe futtatásához szükséges fájl
OLVASSEL.txt	5390 b	2010.03.15. 21:13:51	Fordítási és futtatási utasítások, egyszerű szöveges formában
pctest0com.txt	90 b	2010.04.14. 10:57:38	A 0. teszteset parancsai
pctest0map.gif	407 b	2010.03.30. 21:39:40	A 0. teszteset térképe (kép)
pctest0map.txt	32 b	2010.03.30. 21:39:40	A 0. teszteset térképe (adat)
pctest1com.txt	175 b	2010.04.14. 11:00:59	Az 1. teszteset parancsai

pctest1map.gif	661 b	2010.03.30. 21:39:40	Az 1. teszteset térképe (kép)
pctest1map.txt	68 b	2010.03.30. 21:39:40	Az 1. teszteset térképe (adat)
pctest1out.txt	848 b	2010.04.14. 11:00:59	Az 1. teszteset referencia kimenete
pctest2com.txt	127 b	2010.04.14. 11:00:59	Az 2. teszteset parancsai
pctest2out.txt	381 b	2010.04.14. 11:15:47	Az 2. teszteset referencia kimenete
pctest3com.txt	90 b	2010.04.14. 11:00:59	Az 3. teszteset parancsai
pctest3map.gif	523 b	2010.03.30. 21:39:40	Az 3. teszteset térképe (kép)
pctest3map.txt	50 b	2010.03.30. 21:39:40	Az 3. teszteset térképe (adat)
pctest3out.txt	533 b	2010.04.14. 11:00:59	Az 3. teszteset referencia kimenete
pctest4com.txt	171 b	2010.04.14. 11:00:59	Az 4. teszteset parancsai
pctest4map.gif	741 b	2010.03.30. 21:39:40	Az 4. teszteset térképe (kép)
pctest4map.txt	66 b	2010.03.30. 21:39:40	Az 4. teszteset térképe (adat)
pctest4out.txt	1734 b	2010.04.14. 11:00:59	Az 4. teszteset referencia kimenete
pctest5com.txt	149 b	2010.04.01. 01:21:04	Az 5. teszteset parancsai
pctest5map.gif	686 b	2010.03.30. 21:39:40	Az 5. teszteset térképe (kép)
pctest5map.txt	64 b	2010.03.30. 21:39:40	Az 5. teszteset térképe (adat)
pctest5out.txt	1020 b	2010.04.14. 11:00:59	Az 5. teszteset referencia kimenete
pctest6out.txt	1199 b	2010.04.14. 11:00:59	Az 6. teszteset referencia kimenete
pctest6com.txt	193 b	2010.04.01. 01:21:04	Az 6. teszteset parancsai
pctest7com.txt	158 b	2010.04.14. 11:00:59	Az 7. teszteset parancsai
pctest7out.txt	393 b	2010.04.14. 11:00:59	Az 7. teszteset referencia kimenete
pctest8com.txt	91 b	2010.04.14. 11:00:59	Az 8. teszteset parancsai
pctest8map.gif	409 b	2010.03.30. 21:39:40	Az 8. teszteset térképe (kép)
pctest8map.txt	36 b	2010.03.30. 21:39:40	Az 8. teszteset térképe (adat)
pctest8out.txt	459 b	2010.04.14.	Az 8. teszteset referencia kimenete

		11:00:59	
pctest9com.txt	196 b	2010.04.01. 01:21:04	Az 9. teszteset parancsai
pctest9map.gif	569 b	2010.03.30. 21:39:40	Az 9. teszteset térképe (kép)
pctest9map.txt	50 b	2010.03.30. 21:39:40	Az 9. teszteset térképe (adat)
pctest9out.txt	692 b	2010.04.14. 11:00:59	Az 9. teszteset referencia kimenete
pctest10com.txt	1334 b	2010.04.14. 11:00:59	Az 10. teszteset parancsai
pctest10map.gif	8410 b	2010.04.08. 20:10:31	Az 10. teszteset térképe (kép)
pctest10map.txt	880 b	2010.04.08. 20:10:31	Az 10. teszteset térképe (adat)
pctest10out.txt	b	1970.01.01. 01:00:00	Az 10. teszteset referencia kimenete
pctest11com.txt	178 b	2010.04.14. 11:00:59	Az 11. teszteset parancsai
pctest11map.gif	538 b	2010.03.30. 21:39:40	Az 11. teszteset térképe (kép)
pctest11map.txt	50 b	2010.03.30. 21:39:40	Az 11. teszteset térképe (adat)
pctest11out.txt	785 b	2010.04.14. 11:00:59	Az 11. teszteset referencia kimenete

10.2 Fordítás

A fordítás megkönnyítése érdekében egy batch fájlt hoztunk létre. Így a projekt főkönyvtárában a parancssorban csak ki kell adni a "gtb_compile" parancsot.

(A projekt főkönyvtára az a könyvtár, ahol megtalálható a "gtb_compile.bat" fájl és a "src" alkönyvtár)

Ez a következő parancsokat hajtja végre:

- 1) `mkdir build`
- 2) `javac ./src/GrandTheftBank/*.java -d ./build/`

Azaz

- 1) létrehoz egy "build" nevű könyvtárat
- 2) lefordítja a "src/GrandTheftBank/" könyvtárban lévő .java kiterjesztésű fájlokat és a "build/" könyvtárba menti az elkészült .class kiterjesztésű fájlokat.

Nagyon fontos, hogy ahhoz, hogy a "javac" parancsot használni tudjuk a PATH környezeti változóban szerepelnie kell a JDK bináris könyvtárának.

Amennyiben a "javac" parancs beírására ismeretlen parancsra panaszkodik a futtatandó környezet, kérjük adják hozzá a PATH környezeti változóhoz a JDK aktuális könyvtárát a következő paranccsal:

```
set PATH=%PATH%;C:\Program Files\Java\jdk1.6.0_18\bin\
```

Fontos, hogy a "C:\Program Files\Java\jdk1.6.0_18\bin\" rész helyett a JDK aktuális könyvtárát kell beírni, ami futtatandó környezetenként változhat!

(Általános esetben csak a 1.6.0_18 verziószámot kell kicserélni a gépen lévő JDK aktuális verziószámára. Amennyiben a JDK megtalálása közben problémába ütközik kérje a gép rendszergazdájának segítségét!)

A PATH beállítása után újra kell indítani a parancssort, mert csak akkor lép érvénybe a változás!

Dokumentáció létrehozása: a fordítás sikeressége után adja ki a "gtb_javadoc" parancsot.

Ez a következő parancsokat hajtja végre:

- 1) `javadoc ./src/GrandTheftBank/*.java -d ./javadoc/ -charset utf-8`
- 2) `start ./javadoc/index.html`

Azaz

- 1) legenerálja a dokumentációt a "src/GrandTheftBank/" mappában lévő .java kiterjesztésű fájlokból a "javadoc/" könyvtárba
- 2) megnyitja a dokumentációt a gép alapértelmezett böngészőjében

10.3 Futtatás

1. Tesztesetek ellenőrzésének futtatása

A futtatás megkönnyítése érdekében egy batch fájlt hoztunk létre. Így a projekt főkönyvtárában a parancssorban csak ki kell adni a `"gtb_test"` parancsot.

Ez minden egyes tesztesetre (számszerűleg a következő sorszámúakra: 0 1 2 3 4 5 6 7 8 9 11) lefuttatja a proto-t, ami végrehajtja a `pTestXcom.txt` fájlt (ahol X a teszteset száma), majd az eredményt az `out.txt`-be írja. Ezek után a batch fájl az `out.txt`-t összehasonlítja a `pTestXout.txt`-vel (ami az előre elkészített helyes kimenetet tartalmazza és ahol X a teszteset száma).

A részletes leírásért lásd a „*Emlékeztető a 8.3-as fejezetből*” részt, illetve a tesztesetek is leíró, még részletesebb leírásért a 8. számú Részletes tervek című fejezet „8.2 A tesztek részletes tervei, leírásuk a teszt nyelvén” című részt.

2. A prototípus futtatása

A futtatás megkönnyítése érdekében egy batch fájlt hoztunk létre. Így a projekt főkönyvtárában a parancssorban csak ki kell adni a `"gtb_run"` parancsot.

Ez a következő parancsot hajtja végre: `"java -classpath ./build/ GrandTheftBank.Main"`, azaz elindítja a "build/"-ból kiindulva a GrandTheftBank package Main classjának belépési pontját.

Figyelem! A proto konzolos programja esetenként hosszú sorokat ír ki, ezért javasoljuk, hogy a parancssorban minél nagyobb oszlopszélesség legyen beállítva. A kiírt kimenetek mentődnek a "out.txt" fájlba is, így szükség esetén bármikor visszanezhetők.

A proto konzolos programjának kezelői felületének leírását (interface-definícióját) a dokumentáció 7.1-es pontjában találja meg!

3. Emlékeztető a 8.3-as fejezetből

A tesztadatokat a Proto állítja elő, úgy hogy parancssorból megadott fájlból (`"gtb_run.bat <tesztfájl neve>"`, pl: `"gtb_run pTest0com.txt"`) legyártja az `"out.txt"`-t, ami már csak a teszt kimenetét tartalmazza. Ezt a fájlt összehasonlítjuk egy előre legyártott fájlal, ami a helyes kimenetet tartalmazza. A két szöveges fájl összehasonlítását a DiffUtils diff parancsával végezzük, ami kiírja, hogy mi a különbség a 2 fájl között.

A tesztelés elősegítése érdekében egy automatikus tesztelő parancsfájlt hozunk létre, ami leteszteli az összes tesztfájlt és leellenőrzi a kimeneteket a helyes értékekkel. Ennek a parancsfájlnak `'gtb_test.bat'` lesz a neve. Ezt elindítva a következő kimenetet kapjuk:

```
***** <teszteset száma>. teszt *****
<teszt eredménye>
```

Ahol a `<teszteset száma> = 0 .. 11`

Míg a `<teszt eredménye> =`

- a) "A teszt sikeresen lezajlott!", ha a teszt sikeresen zárult.
- b) Minden más kiírás esetén hiba lépett fel, ami a "diff" program hibaüzenete, erről a program honlapján[1] lehet tájékozódni.

A hibaüzeneteken kívül a "diff" program még a tesztek közötti eltérést is mutatja, aminek a formátuma a következő:

```
"  
< #1:Citizen is on #2:Road at 0,30 position, moving with 0,00 velocity.  
---  
> #1:Citizen is on #2:Road at 1,30 position, moving with 0,00 velocity.  
"
```

Ahol a "<"-lal kezdődő sorok a helyesek, míg a ">"-lal kezdődő sorok a proto által generált eltérő sorok (a példa egy olyan esetet mutat, ahol az autó az útnak a 130%-án halad, tehát lement az útról és ezt feltételezhetően nem kezelte le megfelelően a program).

[1] http://www.gnu.org/software/diffutils/manual/html_node/Output-Formats.html#Output%20Formats

10.4 Új parancsok

A tesztelés megkönnyítése érdekében felvettünk néhány új parancsot, ezek a következők:

setLog [num]

Leírás: Ha num 0, akkor kikapcsolja a fájlba logolást, ha 1, akkor bekapcsolja. Alapértelmezetten fájlba logol a program.

Opciók: -

tickMap [num]

Leírás: Egyenértékű num db tick + showMap utasítás beírásával.

Opciók: ha nem írunk be num-ot, akkor eggyel léptet.

playerGoBackToCity

Leírás: Ha a tesztelő (Robber) éppen kinnt van a városból, akkor visszamegy.

Opciók: -

10.5 Tesztek jegyzőkönyvei

10.5.1 Teszteset0 - JÓ - Egy Citizen tesz egy "lépést" az úton egy tick hatására, a program legalapvetőbb működésének tesztelése

Tesztelő neve	Angeli Róbert
Teszt időpontja	2010.04.10.

10.5.2 Teszteset1 - JÓ - A Robber eléri a Bank épületet (így kirabolja), aminek hatására a Police üldözni kezdi őt. A Robber eléri a Shelter épületet, aminek hatására megnyeri a játékot.

Tesztelő neve	Angeli Róbert
Teszt időpontja	2010.04.10.

10.5.3 Teszteset3 - JÓ - Egy jármű (Citizen, minden másik fajta is ugyanígy viselkedik, kivéve, hogy nem veszi figyelembe a táblákat) mozog az úton, meglát egy lámpát, ami piros, lelassít, majd ha zöld akkor behajt a kereszteződésbe (nem megy át rajta).

Tesztelő neve	Mikó Norbert
Teszt időpontja	2010.04.10.

10.5.4 Teszteset4 - JÓ - A Citizen stoptáblához érkezik és érzékeli, hogy a kereszteződésben jön egy másik autó, ezért megvárja, amíg az áthalad, addig ő nem megy tovább

Tesztelő neve	Koczka Tamás
Teszt időpontja	2010.04.10.

10.5.5 Teszteset5 - JÓ - Egy hármass kereszteződésben egyik útról a másikra halad át a tesztelő (Robber), ennek hatására a harmadik úton lévő Citizen érzékeli, hogy foglalt a kereszteződés, ezért vár és nem megy bele a kereszteződésbe.

Tesztelő neve	Koczka Tamás
Teszt időpontja	2010.04.10.

10.5.6 Teszteset6 - JÓ - Az úton a tesztelő autója (Robber), majd utána egy Citizen van. A tesztelő lassít, majd megáll és ennek hatására a mögötte lévő Citizen is lelassít, majd megáll, hogy az ütközést elkerülje.

Tesztelő neve	Kovács Balázs
Teszt időpontja	2010.04.10.

10.5.7 Teszteset7 - JÓ - Az úton a tesztelő autója (Robber), előtte egy Citizen van. A tesztelő gyorsít, majd ütközik a Citizennel. A Citizen így tönkremegy és eltűnik az útról, a Robber életet veszít. A Robber halad tovább az úton.

Tesztelő neve	Kovács Balázs
Teszt időpontja	2010.04.10.

10.5.8 Teszteset8 - JÓ - Egy Citizen kimegy a városból, aminek hatására a város egy másik pontján egy másik Citizen jön be.

Tesztelő neve	Mikó Norbert
Teszt időpontja	2010.04.10.

10.5.9 Teszteset9 - JÓ - A tesztelőt (Robber) üldözik, ekkor eléri a CityBorder épületet, ennek hatására kilép a városból. Amikor kiért, akkor a rendőrök nem üldözik tovább. A Robbet visszatér a városba ugyanazon a ponton és a rendőrök megint elkezdik őt üldözni.

Tesztelő neve	Koczka Tamás
Teszt időpontja	2010.04.10.

10.5.10 Teszteset10 - JÓ - Egy meglehetősen nagy pályán kipróbálja a programot és megméri, hogy mennyi idő alatt hajtotta végre a műveleteket.

Tesztelő neve	Angeli Róbert
Teszt időpontja	2010.04.10.

10.5.11 Teszteset11 - JÓ - A Robber-t (tesztelő) üldözik, ekkor rámegy a nyuszira, ami miatt egy ideig védett lesz, vagyis nem üldözik a Police-ok tovább. Miután lejárt a védelem ideje a Police-ok ismét üldözőbe veszik (a védettségi időtartam rövid lesz a tesztelés megkönnyítése érdekében).

Tesztelő neve	Koczka Tamás
Teszt időpontja	2010.04.10.

10.5.12 Teszteset2 - ROSSZ - A Robbert körözik, aminek hatására a Police üldözni kezdi őt. A Police ütközik a Robberrel (mivel a Robber áll), aminek hatására a Robber elveszti a játékot.

Tesztelő neve	Kovács Balázs
Teszt időpontja	2010.04.10.
Teszt eredménye	Rossz
Lehetséges hibaok	Ha nincs több rabló, akkor deaktiválni kell az összes rendőrt. Ezt a Részletes tervek elkészítésénél nem vettük figyelembe, szóval azon dokumentációban az elvárt kimenet a rossz.
Változtatások	A fenti leírás alapján a tesztkimenet felülbírálása és - helytelenségének megállapítása után - módosítása. A helyes tesztkimenet: <pre>#1:Robber has turned to #2:Road. #3:Police has turned to #2:Road. #1:Robber is on #2:Road at 0,30 position, moving with 0,00 velocity. #3:Police is on #2:Road at 0,20 position, moving with 0,00 velocity. #3:Police has been activated. << Tick 1 >> #3:Police has crashed with #1:Robber. #1:Robber has lost (been caught). #1:Robber has died. #3:Police has been deactivated.</pre>

10.6 Értékelés

A közös projekt a kezdetektől fogja megfelelően halad, minden az ütemtervhez igazodva elkészült és megfelelő minőségűnek bizonyult.

A csapat stabilan tudja nyújtani az elvárásoknak megfelelő eredményeket.

Az optimalizált munkavégzés végett dinamikusan osztjuk fel egymás között a munkafolyamatokat. Mindenki olyan részhalmazát végzi el a munkának, amit a leggyorsabban és legjobban ő tud elvégezni.

A Prototípus implementálásában Kovács és Koczka nagyobb részt vettek ki.

Tag neve	Munka órában	Munka százalékban
Angeli Róbert	53,25 óra	21,2%
Koczka Tamás	69 óra	27,4%
Kovács Balázs	72 óra	28,6%
Mikó Norbert	57,25 óra	22,8%

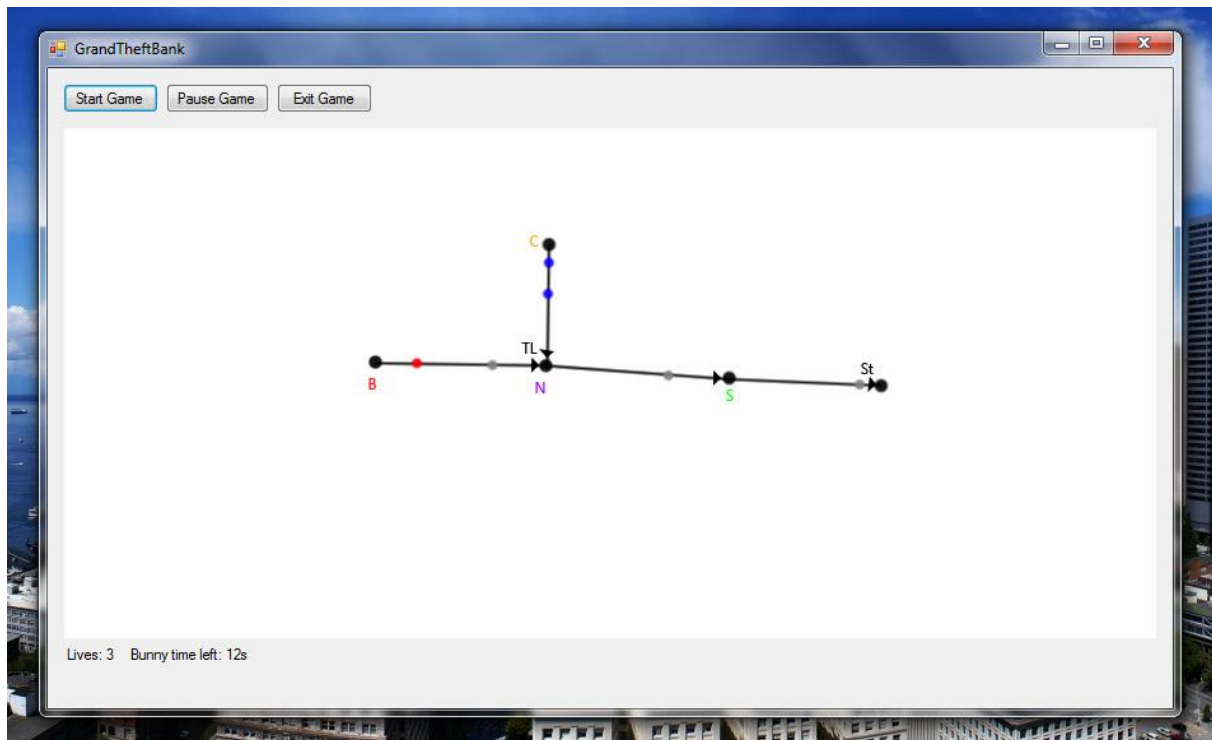
10.7 Napló

Kezdet	Idő	Résztvevők	Leírás
2010.04.02. 20:00	1 óra	Angeli Koczka Kovács Mikó	Értekezlet. Döntés: - Koczka és Kovács leimplementálja a protot - Angeli és Mikó dokumentál és kommentel - Mindenki tesztl
2010.04.08. 16:00	2 óra	Koczka	Proto implementálása
2010.04.08. 16:30	3,5 óra	Kovács	Proto implementálása
2010.04.09. 17:00	2,5 óra	Koczka	Proto implementálása
2010.04.09. 18:00	3 óra	Kovács	Proto implementálása
2010.04.10. 15:00	3 óra	Angeli Koczka Kovács Mikó	Tesztelés
2010.04.13. 10:00	2 óra	Mikó	Kommentelés és dokumentálás
2010.04.13. 16:00	2,5 óra	Angeli	Kommentelés és dokumentálás
2010.03.15. 19:00	1 óra	Mikó	Dokumentáció összeillesztése, formázása

11. Grafikus felület specifikációja

11.1 A grafikus interfész

A program tervezett grafikus felülete:



Menürendszer: Start Game (játék indítása), Pause Game (folyamatban lévő játék megállítása), Exit Game (játék befejezése).

Jelölések:

Road – fekete vonal nyíllal a végén jelezve a haladási irányt

TrafficJunction – Fekete

Robber – Piros kör

Police – Kék kör

Citizen – Szürke kör

TrafficLight – A kellő kereszteződéshez egy TL felirat odaírva

StopSign – A kellő kereszteződéshez egy St felirat odaírva

Bank – A kellő kereszteződéshez egy piros B odaírva

Shelter – A kellő kereszteződéshez egy zöld S odaírva

CityBorder – A kellő kereszteződéshez egy sárga C odaírva

Bunny – A kellő kereszteződéshez egy lila N odaírva

A fent leírt elemek egy Canvasra rajzolva jelennek meg. A Canvas alatt található a Robber életeinek száma, illetve a Bunny hatás (ha van) hátralevő ideje.



A program működése során 3 üzenetablakot kaphatunk:

- 1: Ha nyert a Robber
- 2: Ha veszített a Robber
- 3: Ha a Robber kiment a városból. Ilyenkor az OK nyomására visszamegy a városba.

11.2 A grafikus rendszer architektúrája

11.2.1 A felület működési elve

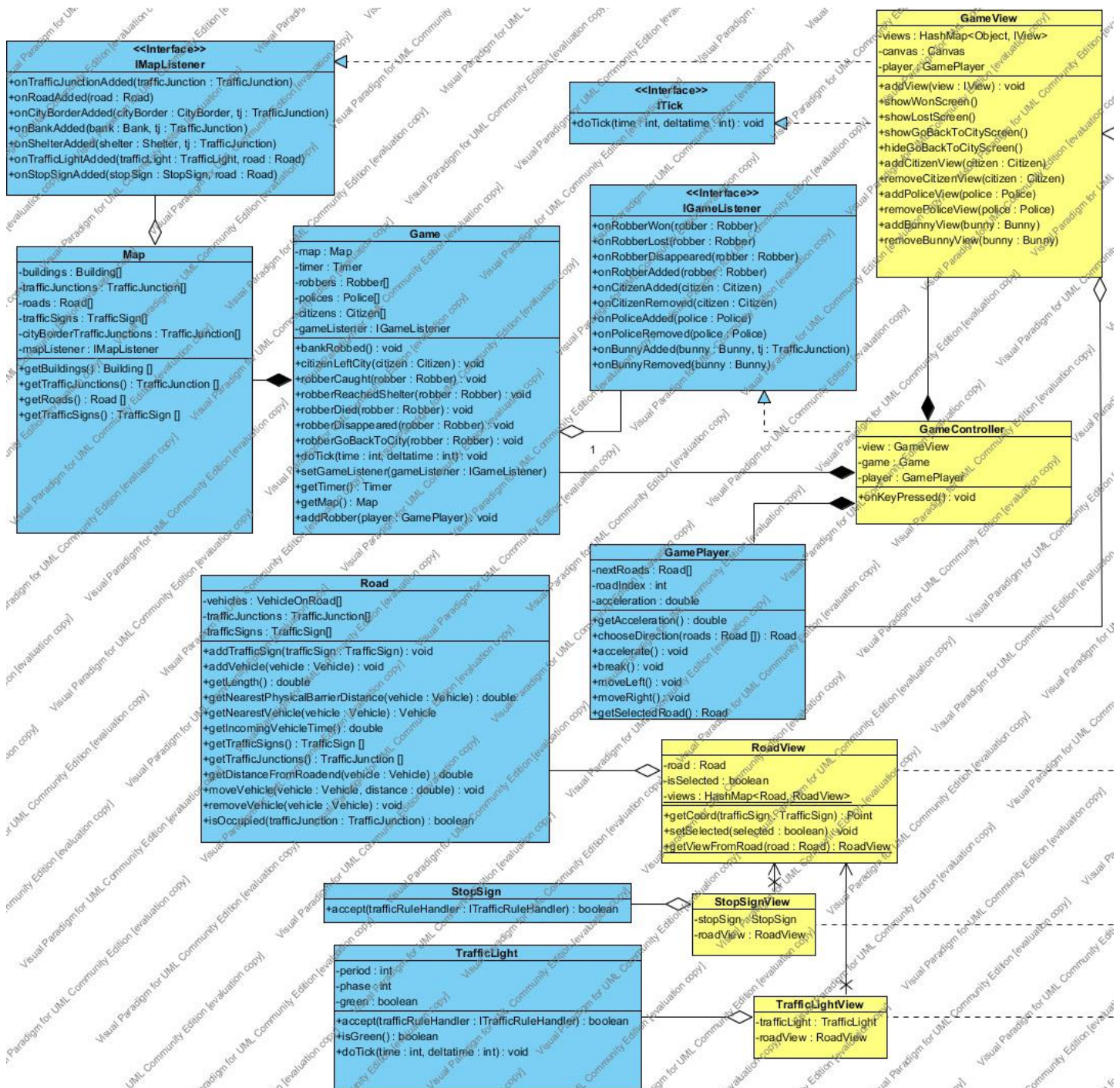
Az MVC alapelveinek megfelelő a modellt a grafikus felülettől teljesen különállóan készítettük el, így a felület modellhez kapcsolása nem jár a modell nagymértékű módosításával. A modellben történt módosítások, csak a változások eseményként való érzékelését valósítják meg a grafikus felülettől független interfészen keresztül.

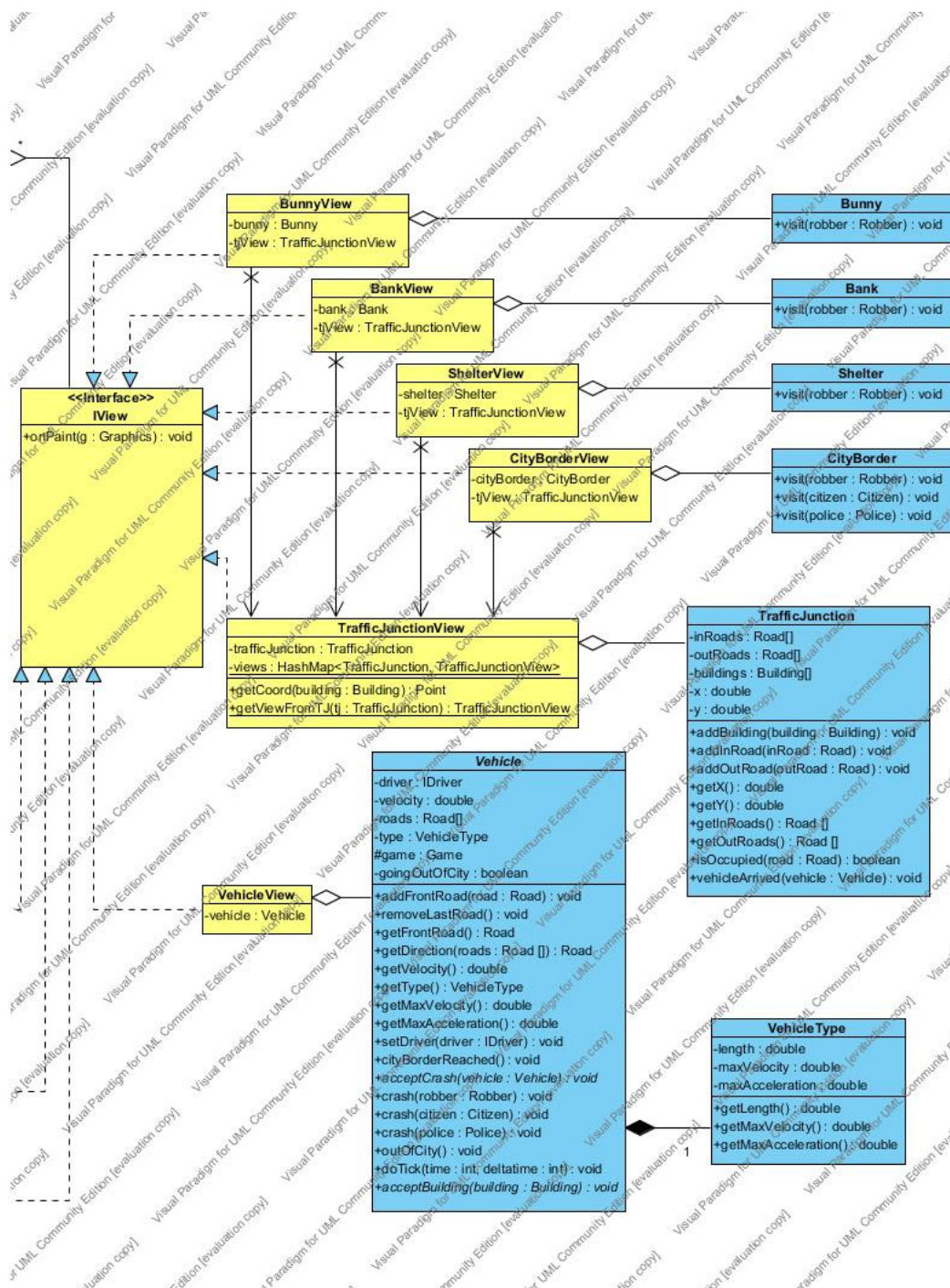
Minden egyes osztályhoz, vagy szükségszerűen osztálycsoporthoz létrehoztunk egy külön View-t, ami a modell objektum megjelenítéséért felelős. A kapcsolat egyirányú: a view ismeri a modellt, de a modell (az MVC alapelveinek megfelelően) nem ismeri a view-t. Így minden modellben lévő objektumhoz gyakorlatilag legenerálódik a megjelenítéséért felelős view objektum. A view objektum közvetlenül ismeri azt a modell objektumot, amihez ő kapcsolódott, míg a modell -> view hozzárendelést egy statikus, az adott view osztályához tartozó kulcs szerint indexelt tömbbel (Javában: HashMap-pal) oldjuk meg.

Az MVC alkalmazásának pull alapú megvalósítást hoztunk létre: tekintve, hogy a modell gyakorlatilag folyamatosan változik, sokkal célravezetőbb a kirajzoláskor újra lekérni a modelltől az aktuális állapotát és annak megfelelően frissíteni a felületet. Ennek ellenére a modellben történnek jelentősebb változások is, amelyek általában új objektumok létrejöttét kísérik, amikhez a megjelenítő felületnek új viewkat kell legyártania, ezért ilyenkor a modell eseményként értesíti a hozzákapcsolódott controllert az akció szükségességéről és ennek hatására a controller a játékot vezérlő fő view-val (GameView) végrehajtja a szükséges intézkedéseket.

Így a játékkal való interakcióért, illetve a grafikus alapvető működéséért is a controller felelős, ő fogja össze a modell és a view működését.

11.2.2 A felület osztály-struktúrája





11.3 A grafikus objektumok felsorolása

11.3.1 Game

- **Felelősség**

A játék irányításáért felelős

- **Ősosztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

- *gameListener* : *IGameListener* - A játék grafikus felületével ezen az interfacen keresztül kommunikál

- **Metódusok**

+ **void setGameListener(gameListener : IGameListener)**: A *gameListener* attributum beállítása.

11.3.2 GameController

- **Felelősség**

Az osztály felelőssége, hogy megteremtse a kapcsolatot a billentyűk segítségével játszó játékos és maga a játék között, azaz az osztály kezeli le az itt folyó interakciót, miszerint a különböző billentyűk lenyomására minek kell történnie a játék folyamán, valamint felelős továbbá azért, hogy a grafikus felületen a különböző kitüntetett események megjelenítésre kerüljenek

- **Ősosztályok**

nincs

- **Interfészek**

IGameListener

- **Attribútumok**

-view : GameView - A játék grafikus felületének eléréséhez szükséges attribútum
-game : Game - A Game osztály eléréséhez szükséges változó
-player : GamePlayer - A gép előtt ülő játékos az itt hivatkozott objektumot tudja irányítani a különböző billentyűk lenyomásával

- **Metódusok**

+ void onKeyPressed() - A különböző billentyűlenyomásokra reagál oly módon, hogy meghívja a GamePlayer osztály adott billentyű lenyomásának megfelelő függvényét

+ **void OnRobberWon()**: A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy a rabló megnyerte a játékot, így ezáltal ez valamilyen formában megjelenítésre kerüljön.

+ **void OnRobberLost()**: A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy a rabló elvesztette a játékot, így ezáltal ez valamilyen formában megjelenítésre kerüljön

+ **void OnRobberDisappeared()**: A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy a rabló elhagyta várost, így ezáltal ez valamilyen formában megjelenítésre kerüljön

+ void onRobberAdded(robber : Robber) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy új rabló került a játékba, így ezáltal ez valamilyen formában megjelenítésre kerüljön

+ void onCitizenAdded(citizen : Citizen) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy új citizen került játékba, így ezáltal ez valamilyen formában megjelenítésre kerüljön.

+ void onCitizenRemoved(citizen : Citizen) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy citizen eltávolításra került a játékban, így ezáltal ez valamilyen formában megjelenítésre kerüljön

+ void onPoliceAdded(police : Police) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy új police került a játékba, így ezáltal ez valamilyen formában megjelenítésre kerüljön

+ void onPoliceRemoved(police : Police) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy police eltávolításra került a játékban, így ezáltal ez valamilyen formában megjelenítésre kerüljön

+ void onBunnyAdded(bunny : Bunny, tj : TrafficJunction) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy új bunny került a játékba, így ezáltal ez valamilyen formában megjelenítésre kerüljön

+ void onBunnyRemoved(bunny : Bunny) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy bunny eltávolításra került a játékban, így ezáltal ez valamilyen formában megjelenítésre kerüljön

11.3.3 GuiController

- **Felelősség**

Az osztály kezeli azt le, hogy a grafikus felületen a kitüntetett események megjelenjenek (Rabló nyer, Rabló veszít, Rabló eltűnik)

- **Ősosztályok**

nincs

- **Interfészek**

IGameListener

- **Attribútumok**

-view : GameView - A játék grafikus felülete

-game : Game - A játék (modell) osztályra egy referencia, hogy tudjon kommunikálni az osztállyal, pl. a rabló cselekszik.

- robberOutOfCity : boolean - Flag ami azt tárolja, hogy a játékos éppen városon kívül van-e.

- **Metódusok**

void OnRobberWon(): A Game hívja meg, ha a rabló nyer. Meghívja a GameView ShowWonScreen() fv-ét.

void OnRobberLost(): A Game hívja meg, ha a rabló veszít. Meghívja a GameView ShowLostScreen() fv-ét.

void OnRobberDisappeared(): A Game hívja meg, ha a rabló kimegy a városból. Meghívja a GameView ShowGoBackToCityScreen() fv-ét.

11.3.4 IGameListener

- **Felelősség**

A Game ezen az interfacen keresztül tud kommunikálni a GUI-val. Szól a GUI-nak ha elkapták a rablót, ha nyert stb.

- **Ősosztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

nincs

- **Metódusok**

void OnRobberWon(): A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy a rabló megnyerte a játékot, így ezáltal ez valamilyen formában megjelenítésre kerüljön.

void OnRobberLost(): A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy a rabló elvesztette a játékot, így ezáltal ez valamilyen formában megjelenítésre kerüljön

void OnRobberDisappeared(): A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy a rabló elhagyta várost, így ezáltal ez valamilyen formában megjelenítésre kerüljön

+onRobberAdded(robber : Robber) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy új rabló került a játékba, így ezáltal ez valamilyen formában megjelenítésre kerüljön

+onCitizenAdded(citizen : Citizen) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy új citizen került játékba, így ezáltal ez valamilyen formában megjelenítésre kerüljön.

+onCitizenRemoved(citizen : Citizen) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy citizen eltávolításra került a játékban, így ezáltal ez valamilyen formában megjelenítésre kerüljön

+onPoliceAdded(police : Police) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy új police került a játékba, így ezáltal ez valamilyen formában megjelenítésre kerüljön

+onPoliceRemoved(police : Police) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy police eltávolításra került a játékban, így ezáltal ez valamilyen formában megjelenítésre kerüljön

+onBunnyAdded(bunny : Bunny, tj : TrafficJunction) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy új bunny került a játékba, így ezáltal ez valamilyen formában megjelenítésre kerüljön

+onBunnyRemoved(bunny : Bunny) A függvény felelőssége, hogy tudassa a GameView-val a Game-ben történt változásokat, jelen esetben hogy egy bunny eltávolításra került a játékban, így ezáltal ez valamilyen formában megjelenítésre kerüljön.

11.3.5 GameView

- **Felelősség**

A játék grafikus felülete. Ő felel a megjelenítésért, hogy mindig helyes képet lásson a játékos.

- **Össztályok**

nincs

- **Interfészek**

ITick

IMapListener

- **Attribútumok**

-views : HashMap<Object, IView> - A játékban azon objektumok amelyeket kirajzolunk. Ezeket tároljuk ebben a listában

-canvas : Canvas - Erre a felületre rajzoljuk ki az objektumokat. Ez a festővászon.

-player : GamePlayer - A gép előtt ülő játékos által irányított objektum

- **Metódusok**

+ **void addView(view : IView)** - Ezzel a függvénnyel tudunk új kirajzolandó objektumot hozzáadni a views listához.

+ **void showWonScreen()**: A rabló sikeres feladatteljesítésekor megjelenő ablak megjelenítésért felelős metódus

+ **void showLostScreen()**: A rabló sikertelen feladatteljesítésekor megjelenő ablak megjelenítésért felelős metódus

+ **void showGoBackToCityScreen()**: A rabló városhagyásakor megjelenő ablak megjelenítésért felelős metódus

+ **void hideGoBackToCityScreen()**: A rabló városba való visszatérésekor az ablak elrejtéséért felelős metódus

+ **void doTick(time : int, deltatime : int)** - Tick hatására a GameView meghívja az összes tárolt objektumra (views-ból) az onPaint-et és újrarajzolja a canvas-t.

+**addCitizenView(citizen : Citizen)** - A függvénnyel a paraméterül adott citizen-t lehet a views listához adni. A tick eseményre meghívásra kerül a lista elemeinek az onPaint függvénye

+**removeCitizenView(citizen : Osztalydiagram.Citizen)** - A függvénnyel a paraméterül adott citizen-t lehet a views listából eltávolítani.

+**addPoliceView(police : Police)** - A függvénnyel a paraméterül adott police-t lehet a views listához adni. A tick eseményre meghívásra kerül a lista elemeinek az onPaint függvénye

+**removePoliceView(police : Police)** - A függvénnyel a paraméterül adott police-t lehet a views listából eltávolítani.

+**addBunnyView(bunny : Bunny)** - A függvénnyel a paraméterül adott bunny-t lehet a views listához adni. A tick eseményre meghívásra kerül a lista elemeinek az onPaint függvénye

+**removeBunnyView(bunny : Bunny)** - A függvénnyel a paraméterül adott bunny-t lehet a views listából eltávolítani.

+**onTrafficJunctionAdded(trafficJunction : TrafficJunction)** A függvénnyel a paraméterül adott kereszteződést lehet a views listához adni. A tick eseményre meghívásra kerül a lista elemeinek az onPaint függvénye

+**onRoadAdded(road : Road)** A függvénnyel a paraméterül adott utat lehet a views listához adni. A tick eseményre meghívásra kerül a lista elemeinek az onPaint függvénye

+**onCityBorderAdded(cityBorder : CityBorder, tj : TrafficJunction)** A függvénnyel a paraméterül adott városhatárt lehet a views listához adni. A tick eseményre meghívásra kerül a lista elemeinek az onPaint függvénye

+**onBankAdded(bank : Bank, tj : TrafficJunction)** A függvénnyel a paraméterül adott bank épületét lehet a views listához adni. A tick eseményre meghívásra kerül a lista elemeinek az onPaint függvénye

+**onShelterAdded(shelter : Shelter, tj : TrafficJunction)** A függvénnyel a paraméterül adott rejtékhely épületét lehet a views listához adni. A tick eseményre meghívásra kerül a lista elemeinek az onPaint függvénye

+**onTrafficLightAdded(trafficLight : TrafficLight, road : Road)** A függvénnyel a paraméterül adott közlekedési lámpát lehet a views listához adni. A tick eseményre meghívásra kerül a lista elemeinek az onPaint függvénye

+**onStopSignAdded(stopSign : StopSign, road : Road)** A függvénnyel a paraméterül adott stoptáblát lehet a views listához adni. A tick eseményre meghívásra kerül a lista elemeinek az onPaint függvénye

11.3.6 IMapListener

- **Felelősség**

Az interface felelőssége, hogy a függvényei hívásának segítségével megteremtse a kapcsolatot a Map és a GameView között, azaz hogy a térképen történt változások azok a grafikus felületen is megjelenítésre kerüljenek majd

- **Ósosztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

nincs

- **Metódusok**

+onTrafficJunctionAdded(trafficJunction : TrafficJunction) A függvény felelőssége, hogy tudassa a GameView-val a Map-ben történt változásokat, jelen esetben hogy új útkereszteződés lett hozzáadva a térképhez, így ezáltal ez valamilyen formában megjelenítésre kerüljön.

+onRoadAdded(road : Road) A függvény felelőssége, hogy tudassa a GameView-val a Map-ben történt változásokat, jelen esetben hogy új út lett hozzáadva a térképhez, így ezáltal ez valamilyen formában megjelenítésre kerüljön.

+onCityBorderAdded(cityBorder : CityBorder, tj : TrafficJunction) A függvény felelőssége, hogy tudassa a GameView-val a Map-ben történt változásokat, jelen esetben hogy új városhatár lett hozzáadva a térképhez, így ezáltal ez valamilyen formában megjelenítésre kerüljön.

+onBankAdded(bank : Bank, tj : TrafficJunction) A függvény felelőssége, hogy tudassa a GameView-val a Map-ben történt változásokat, jelen esetben hogy új bank épület lett hozzáadva a térképhez, így ezáltal ez valamilyen formában megjelenítésre kerüljön.

+onShelterAdded(shelter : Shelter, tj : TrafficJunction) A függvény felelőssége, hogy tudassa a GameView-val a Map-ben történt változásokat, jelen esetben hogy új rejtkehely épület lett hozzáadva a térképhez, így ezáltal ez valamilyen formában megjelenítésre kerüljön.

+onTrafficLightAdded(trafficLight : TrafficLight, road : Road) A függvény felelőssége, hogy tudassa a GameView-val a Map-ben történt változásokat, jelen esetben hogy új közlekedési lámpa lett hozzáadva a térképhez, így ezáltal ez valamilyen formában megjelenítésre kerüljön.

+onStopSignAdded(stopSign : StopSign, road : Road) A függvény felelőssége, hogy tudassa a GameView-val a Map-ben történt változásokat, jelen esetben hogy új stoptábla lett hozzáadva a térképhez, így ezáltal ez valamilyen formában megjelenítésre kerüljön..

11.3.7 IView

- **Felelősség**

A GameView ezen az interfacen keresztül tudja elérni a kirajzolandó objektumokat. Az interface-t ezeknek a kirajzolando objektumoknak meg kell valósítaniuk.

- **Össztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

nincs

- **Metódusok**

+ **void onPaint(g : Graphics)** - Ezt a függvény az ezt implementáló osztály fogja tartalmazni, melyben megvalósításra kerül hogy hogyan és hova kell kirajzolni az adott objektumot.

11.3.8 TrafficJunctionView

- **Felelősség**

A TrafficJunction, azaz az útkereszteződés kirajzolásáért felelős osztály

- **Össztályok**

nincs

- **Interfészek**

IView

- **Attribútumok**

-trafficJunction : TrafficJunction - Az az útkereszteződés objektum amelyet ki szeretnénk rajzolni.

-views : HashMap<TrafficJunction, TrafficJunctionView> - Egy lista melyben a különböző útkereszteződések, TrafficJunction-ok és a hozzájuk tartozó TrafficJunctionView-k vannak tárolva

- **Metódusok**

+ **void onPaint(g : Graphics)** - Ez a függvény tudja kirajzolni az adott TJ-t

+ **Point getCoord(building : Building)** - Ez a függvény felelős azért, hogy a TrafficJunctionban tárolt Buildingek kirajolásukkor a koordinátaikat ezen a függvényen keresztül tudjuk meg. Ez például akkor is fontos ha egy TJ-ben több Buildinget tárolunk. (Nehogy egymásra rajzoljuk őket.)

+**getViewFromTJ(tj : TrafficJunction)** : TrafficJunctionView - Az adott TrafficJunction-hoz tartozó TrafficJunctionView-t adja vissza

11.3.9 BunnyView

- **Felelősség**

A Bunny, azaz a nyúl kirajolásáért felelős osztály

- **Össztályok**

nincs

- **Interfészek**

IView

- **Attribútumok**

-tjView : TrafficJunctionView - Az a TJ ahol az objektum található. Koordináta elérés miatt kell.

- bunny : Bunny - A kirajzolandó bunny.

- **Metódusok**

+ **void onPaint(g : Graphics)** - Ez a függvény tudja kirajzolni az adott Bunny-t.

11.3.10 BankView

- **Felelősség**

A Bank kirajzolásáért felelős osztály

- **Ősosztályok**

nincs

- **Interfészek**

IView

- **Attribútumok**

-tjView : TrafficJunctionView - Az a TJ ahol az objektum található. Koordináta elérés miatt kell.

- bank : Bank - A kirajzolandó bank.

- **Metódusok**

+ **void onPaint(g : Graphics)** - Ez a függvény tudja kirajzolni az adott Bankot.

11.3.11 ShelterView

- **Felelősség**

A Shelter, azaz a rejtekhely kirajzolásáért felelős osztály

- **Ősosztályok**

nincs

- **Interfészek**

IView

- **Attribútumok**

-tjView : TrafficJunctionView - Az a TJ ahol az objektum található. Koordináta elérés miatt kell.

- shelter : Shelter - A kirajzolandó shelter.

- **Metódusok**

+ **void onPaint(g : Graphics)** - Ez a függvény tudja kirajzolni az adott Sheltert.

11.3.12 CityBorderView

- **Felelősség**

A CityBorder, azaz a városhatár kirajzolásáért felelős osztály

- **Ősosztályok**

nincs

- **Interfészek**

IView

- **Attribútumok**

-tjView : TrafficJunctionView - Az a TJ ahol az objektum található. Koordináta elérés miatt kell.

- cityBorder : CityBorder - A kirajzolandó CityBorder.

- **Metódusok**

+ void onPaint(g : Graphics) - Ez a függvény tudja kirajzolni az adott CityBordert.

11.3.13 VehicleView

- **Felelősség**

A különböző járművek kirajzolásáért felelős osztály

- **Ősosztályok**

nincs

- **Interfészek**

IView

- **Attribútumok**

- vehicle : Vehicle - A kirajzolandó jármű objektum.

.

- **Metódusok**

+ void onPaint(g : Graphics) - Ez a függvény tudja kirajzolni az adott járművet.

11.3.14 RoadView

- **Felelősség**

Az utak kirajzolásáért felelős osztály

- **Ősosztályok**

nincs

- **Interfészek**

IView

- **Attribútumok**

- road : Road - Az az objektumok amelyet kirajzolunk.
- isSelected : boolean
- views : HashMap<Road, RoadView> - Egy lista, mely tartalmazza a különböző utakat, Road-okat, és a hozzájuk tartozó RoadView-kat

- **Metódusok**

- + **void onPaint(g : Graphics)** - Ez a függvény tudja kirajzolni az adott utat.
- + **Point getCoord(trafficSign : TrafficSign)** - Ez a függvény felelős azért, hogy a Roadon tárolt Stoptáblák és Lámpák majd amikor kirajzolásra kerülnek akkor ők ezen a függvényen keresztül tudják meg a koordinátaikat. Ez azért fontos, hogy elkerüljük az egymásra rajzolást.
- + **setSelected(selected : boolean) : void** - Az osztály Selected attribútumát állítja be a paraméternek megfelelő értékre
- + **getViewFromRoad(road : Road) : RoadView** - Visszaadja visszatérési értékül a paraméterül adott road-hoz tartozó RoadView-t

11.3.15 StopSignView

- **Felelősség**

A Stoptábla kirajzolásáért felelős osztály

- **Ősosztályok**

nincs

- **Interfészek**

IView

- **Attribútumok**

- roadView : RoadView - Az a Road ahol az objektum található. Koordináta elérés miatt kell.
- stopSign : StopSign - A kirajzolandó stopptábla.

- **Metódusok**

- + **void onPaint(g : Graphics)** - Ez a függvény tudja kirajzolni az adott Stop táblát.

11.3.16 TrafficLightView

- **Felelősség**

A közlekedési lámpa kirajzolásáért felelős osztály

- **Ősosztályok**

nincs

- **Interfészek**

IView

- **Attribútumok**

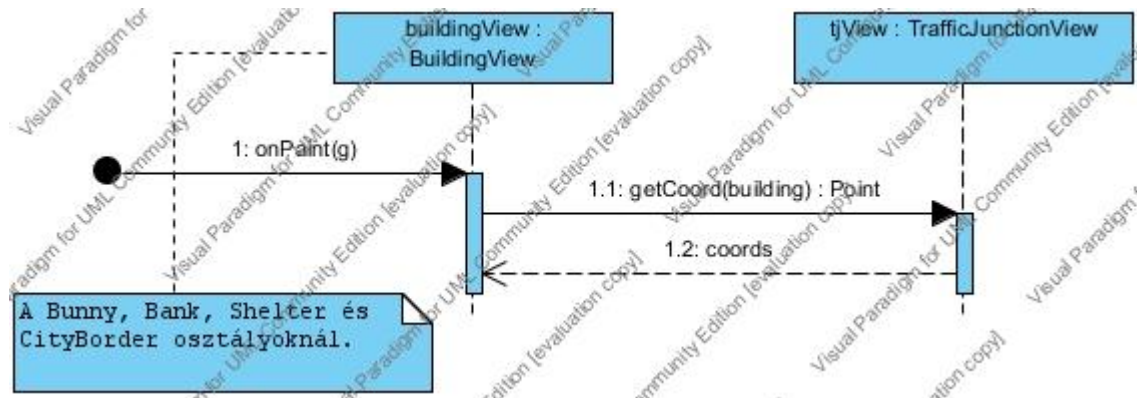
- roadView : RoadView - Az a Road ahol az objektum található. Koordináta elérés miatt kell.
- trafficLight : TrafficLight - A kirajzolandó közlekedési lámpa..

- **Metódusok**

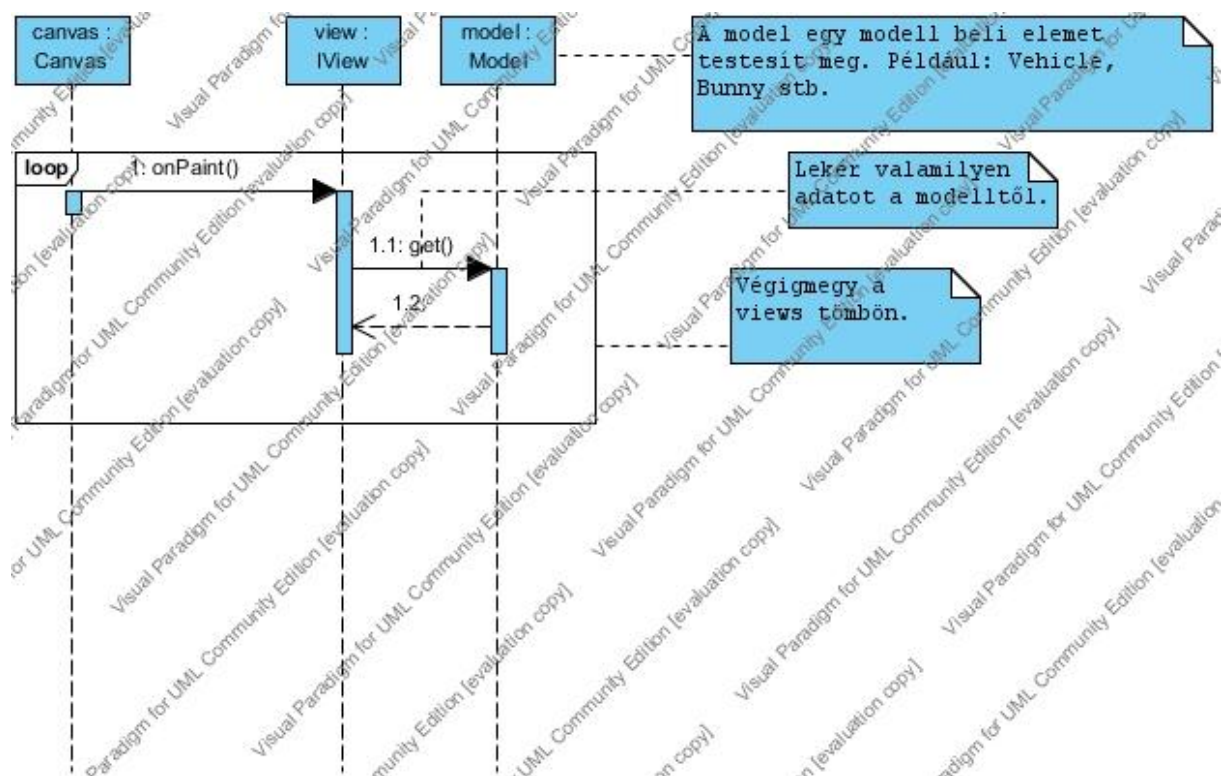
+ **void onPaint(g : Graphics)** - Ez a függvény tudja kirajzolni az adott közlekedési lámpát.

11.4 Kapcsolat az alkalmazói rendszerrel

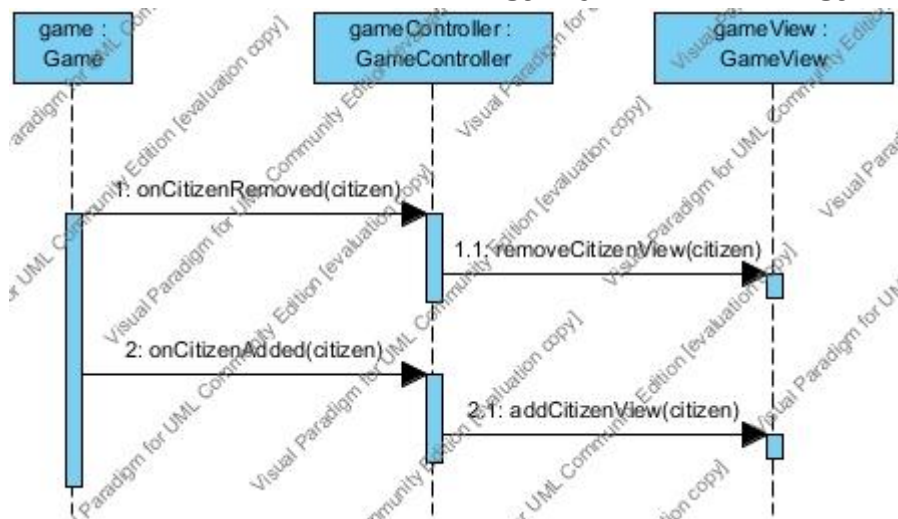
11.4.1 Graf - BuildingView-ok kérdezik a TJView-ot



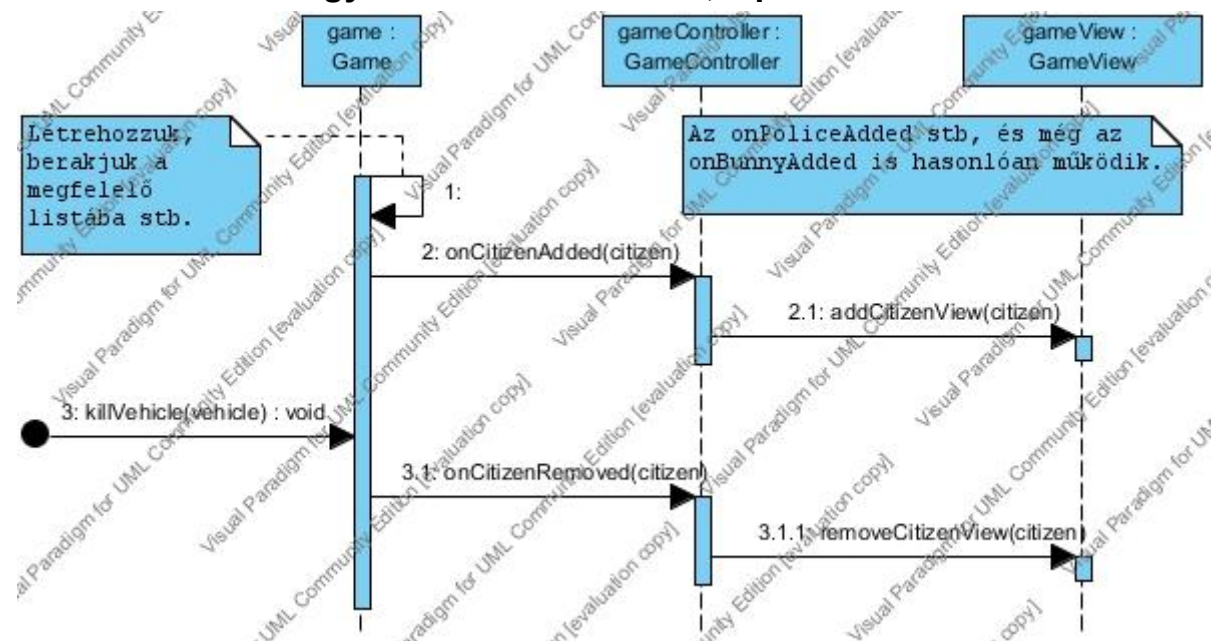
11.4.2 Graf - Canvas rajzol

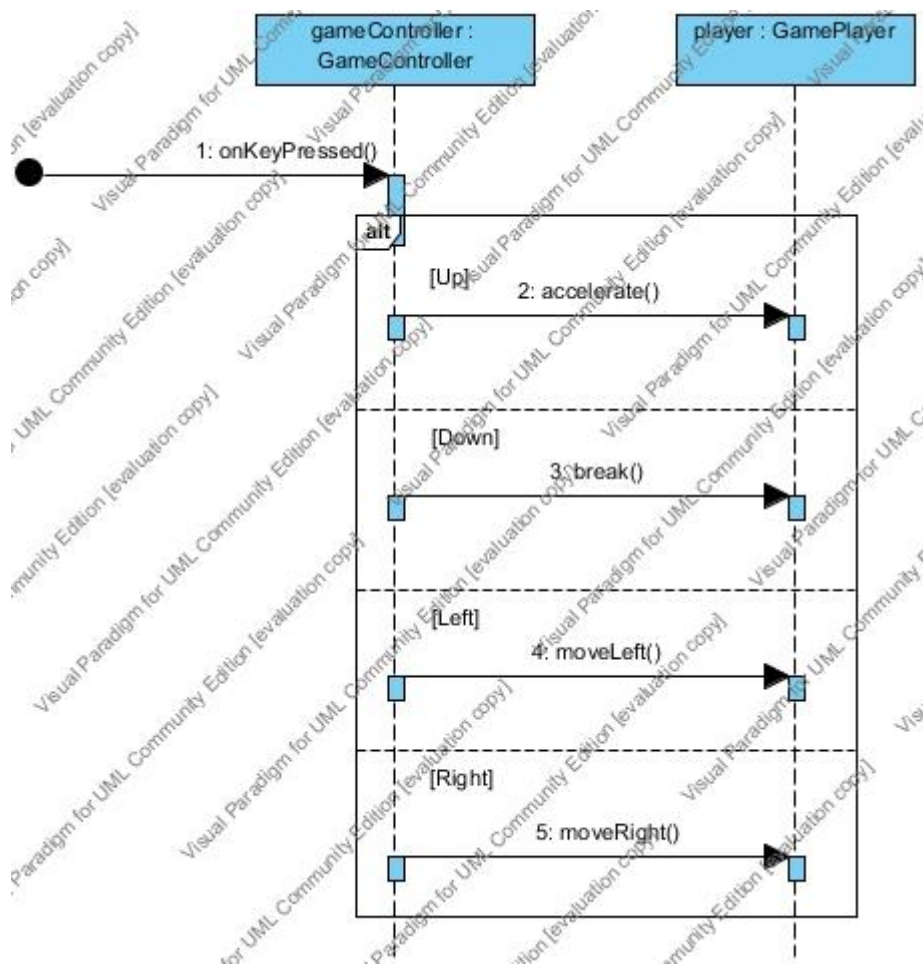


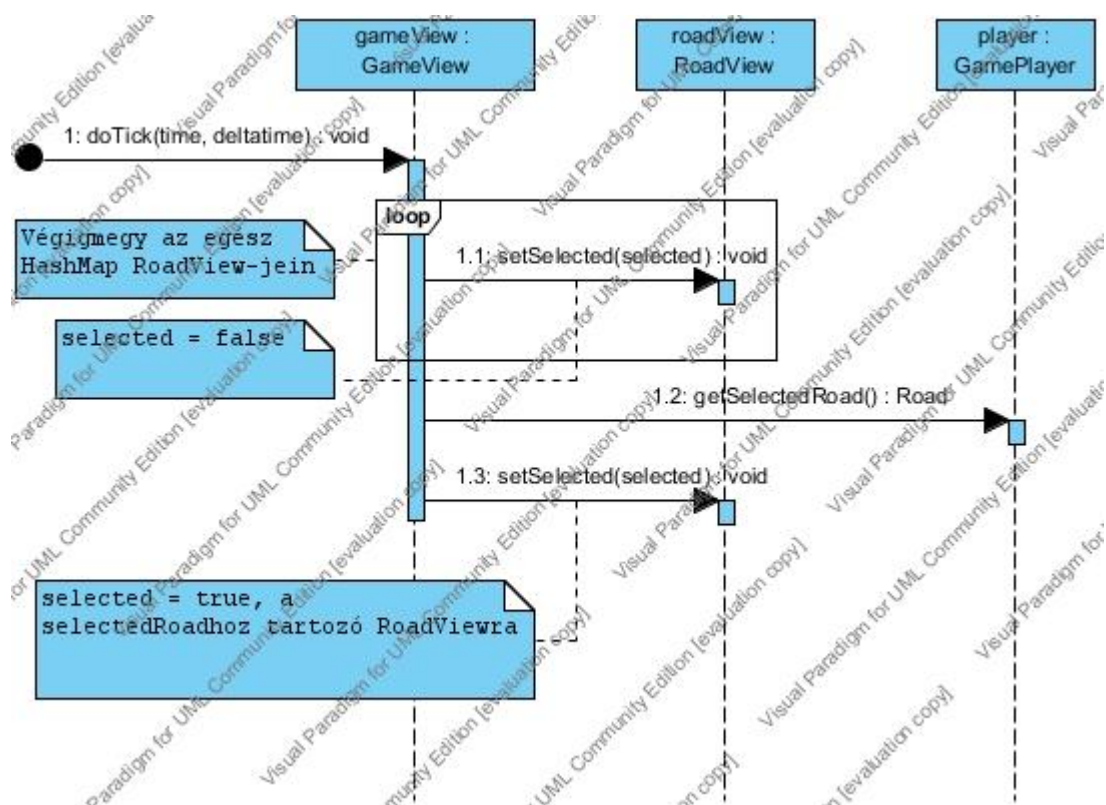
11.4.3 Graf - Citizen kimegy, új Citizen bemegy



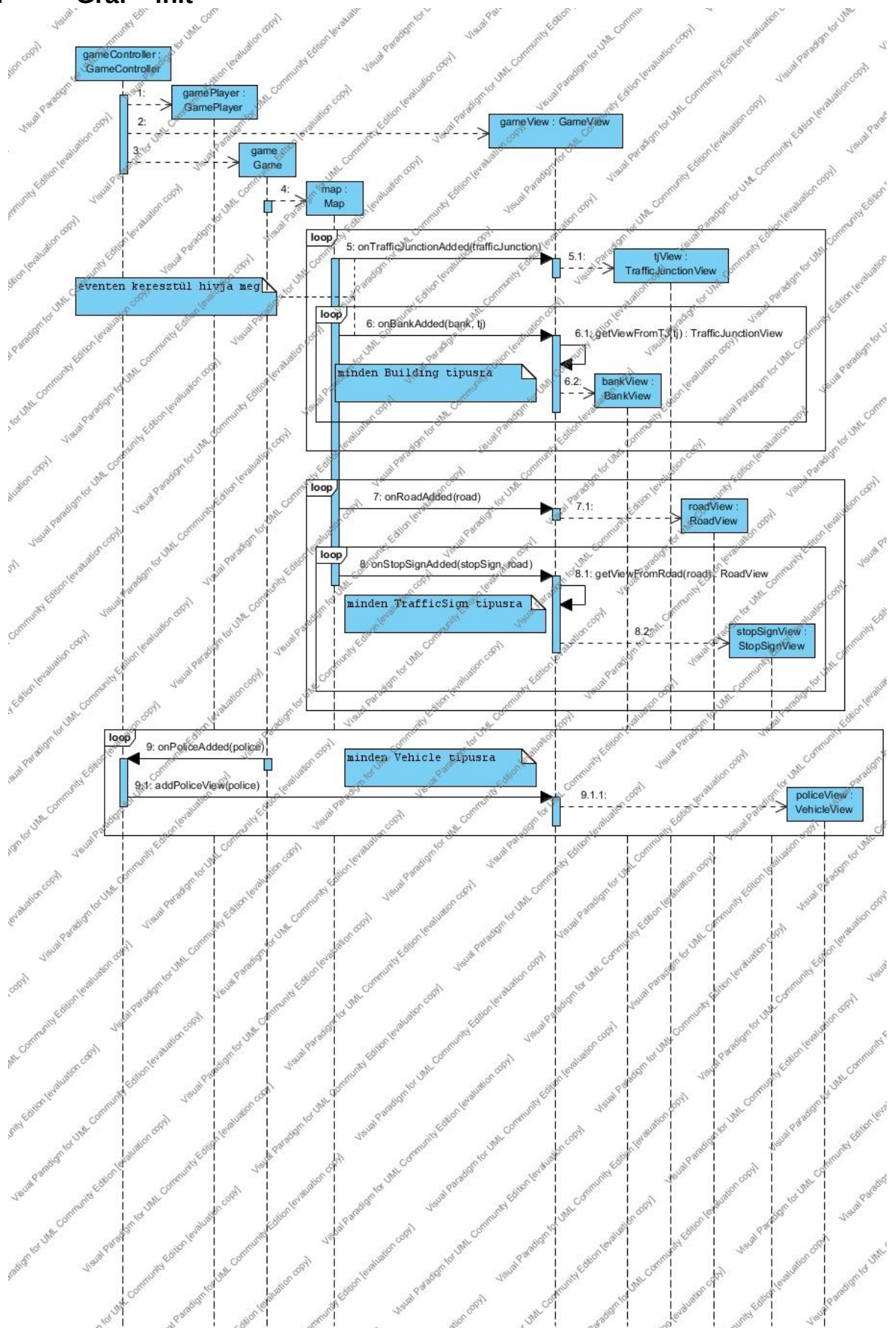
11.4.4 Graf - Egy vehiclet létrehozunk, elpusztítunk



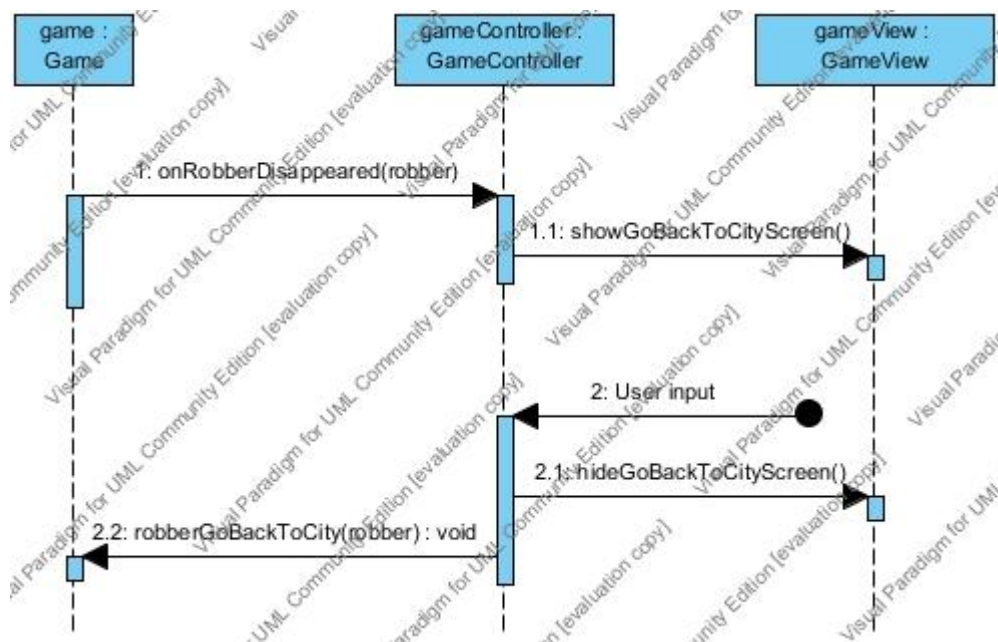
11.4.5 Graf - GameController billentyűlenyomásra reagál**11.4.6 Graf - GameView doTick**



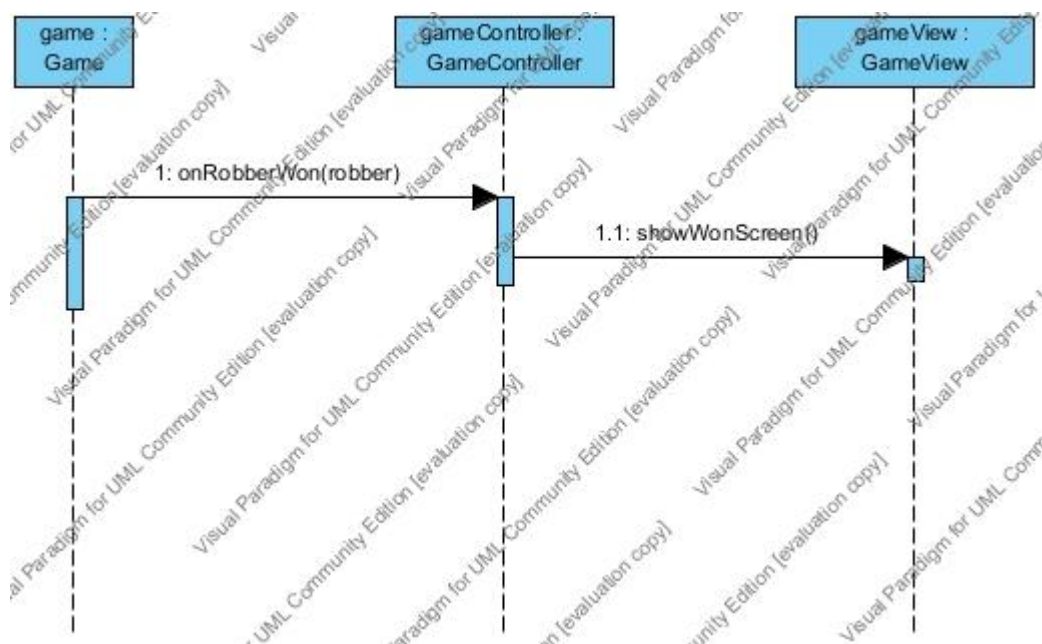
11.4.7 Graf – init



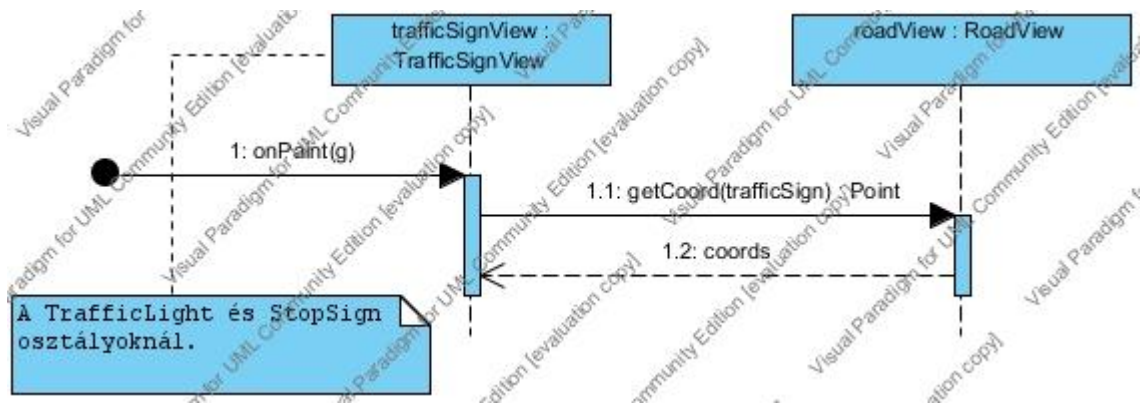
11.4.8 Graf - Player kimegy majd visszajön a városba



11.4.9 Graf - Player nyer (a Player veszít hasonló)



11.4.10 Graf - TrafficSignView-ok kérdezik a RoadView-ot



Napló

Kezdet	Idő	Részv evők	Leírás
2010.04.16. 20:00	1 óra	Angeli Koczka Kovács Mikó	Értekezlet. Döntés: - Koczka és Kovács tervez és diagramot készít - Angeli, Koczka és Mikó dokumentál - Mikó GUI-t rajzol
2010.04.17. 16:00	3 óra	Kovács	Osztály diagram tervezése
2010.04.17. 16:00	3 óra	Koczka	Szekvencia diagram tervezése
2010.04.17. 16:00	2 óra	Mikó	Dokumentálás
2010.04.19. 18:00	3 óra	Kovács	Szekvencia diagramok tervezése
2010.04.18. 17:00	1 óra	Mikó	GUI felület rajzolása
2010.04.19. 15:00	3 óra	Koczka	Dokumentálás és diagram tervezése
2010.04.19. 10:00	2 óra	Angeli	Dokumentálás
2010.04.20. 16:00	3 óra	Angeli	Dokumentálás
2010.03.21. 19:00	1,5 óra	Mikó	Dokumentáció összeillesztése, formázása

13. Grafikus változat beadása

13.1 Fordítási és futtatási útmutató

13.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
src/GrandTheftBank/Bank.java	778 b	2010.05.05 23:13	A bankot megvalósító osztály
src/GrandTheftBank/BankView.java	1166 b	2010.05.05 23:38	A Bank megjelenítéséért felelős osztály
src/GrandTheftBank/Building.java	1533 b	2010.05.05 16:41	Az épületek alaposztálya
src/GrandTheftBank/Bunny.java	3568 b	2010.05.05 23:13	A Bunnyt megvalósító osztály
src/GrandTheftBank/BunnyView.java	1275 b	2010.05.05 23:38	A Bunny megjelenítéséért felelős osztály
src/GrandTheftBank/Citizen.java	2063 b	2010.05.05 23:13	A városi autót megvalósító osztály
src/GrandTheftBank/CitizenAI.java	8403 b	2010.05.05 16:41	A városi autót irányító algoritmust megvalósító osztály
src/GrandTheftBank/CitizenView.java	2661 b	2010.05.05 23:38	A Citizen autó megjelenítéséért felelős osztály
src/GrandTheftBank/CityBorder.java	983 b	2010.05.05 23:13	A városhatárt megvalósító osztály
src/GrandTheftBank/CityBorderView.java	1255 b	2010.05.05 23:38	A CityBorder megjelenítéséért felelős osztály
src/GrandTheftBank/DriverAdapter.java	1445 b	2010.05.05 16:41	Segéd adapter az IDriver interfésznek
src/GrandTheftBank/Game.java	10545 b	2010.05.06 1:04	A játék irányítását megvalósító osztály
src/GrandTheftBank/GameController.java	6688 b	2010.05.06 1:04	Az MVC Controll osztálya
src/GrandTheftBank/GamePlayer.java	2522 b	2010.05.05 23:38	A Játékos irányítását megvalósító osztály
src/GrandTheftBank/GameView.java	10967 b	2010.05.06 1:04	A MVC View osztálya
src/GrandTheftBank/IDriver.java	743 b	2010.05.05 16:41	Az autót irányító algoritmus interfésze
src/GrandTheftBank/IGameListener.java	815 b	2010.05.05 23:46	A Game és GameController közti kommunikációt segítő interface
src/GrandTheftBank/IMapListener.java	765 b	2010.05.05 16:41	A Map és View közti kommunikációt segítő interface
src/GrandTheftBank/ITick.java	714 b	2010.05.05 16:41	Az időt érzékelő elem interfésze
src/GrandTheftBank/ITrafficRuleHandler.java	1066 b	2010.05.05 16:41	A közlekedési szabályt érzékelő elem interfésze
src/GrandTheftBank/IView.java	238 b	2010.05.05 16:41	A játékban kirajzolandó osztályok interface-e

src/GrandTheftBank/ Main.java	3510 b	2010.05.05 16:41	A program futását lekezelő osztály
src/GrandTheftBank/ Map.java	10749 b	2010.05.05 16:41	A játék térképét tartalmazó osztály
src/GrandTheftBank/ Police.java	3903 b	2010.05.05 23:13	A rendőrautót megvalósító osztály
src/GrandTheftBank/ PoliceAI.java	10313 b	2010.05.05 16:41	A rendőr autót irányító algoritmust megvalósító osztály
src/GrandTheftBank/ PoliceView.java	4348 b	2010.05.05 23:46	A Police megjelenítéséért felelős osztály
src/GrandTheftBank/ Road.java	9720 b	2010.05.05 23:13	Az utat megvalósító osztály
src/GrandTheftBank/ RoadView.java	3955 b	2010.05.05 23:46	A Road megjelenítéséért felelős osztály
src/GrandTheftBank/ Robber.java	5085 b	2010.05.05 23:13	A rabló autóját megvalósító osztály
src/GrandTheftBank/ RobberView.java	2461 b	2010.05.06 0:00	A Robber megjelenítéséért felelős osztály
src/GrandTheftBank/ Shelter.java	771 b	2010.05.05 23:13	A rejtékhelyet megvalósító osztály
src/GrandTheftBank/ ShelterView.java	1174 b	2010.05.06 0:00	A Shelter megjelenítéséért felelős osztály
src/GrandTheftBank/ StopSign.java	511 b	2010.05.05 16:41	A stop táblát megvalósító osztály
src/GrandTheftBank/ StopSignView.java	1041 b	2010.05.06 0:00	A Stop Sign megjelenítéséért felelős osztály
src/GrandTheftBank/ Timer.java	3291 b	2010.05.05 23:38	Az időzítőt megvalósító osztály
src/GrandTheftBank/ TrafficJunction.java	4614 b	2010.05.05 16:41	Az útkereszteződést megvalósító osztály
src/GrandTheftBank/ TrafficJunctionView.java	2319 b	2010.05.05 23:46	A Traffic Junction megjelenítéséért felelős osztály
src/GrandTheftBank/ TrafficLight.java	2457 b	2010.05.05 23:13	A közlekedési lámpát megvalósító osztály
src/GrandTheftBank/ TrafficLightView.java	1054 b	2010.05.06 0:00	A Traffic Light megjelenítéséért felelős osztály
src/GrandTheftBank/ TrafficSign.java	494 b	2010.05.05 16:41	A közlekedési szabályok őosztálya
src/GrandTheftBank/ Vehicle.java	6329 b	2010.05.05 23:13	Az autót megvalósító őosztály
src/GrandTheftBank/ VehicleOnRoad.java	1192 b	2010.05.05 16:41	Tároló osztály az autók úton való elhelyezésére
src/GrandTheftBank/ VehicleType.java	1294 b	2010.05.05 16:41	Az autómódelleket megvalósító osztály
gtb.bat	60 b	2010.05.05 16:41	Lefordítja a projektet, elkészíti a dokumentációt és leteszteli a teszteseteket
gtb_compile.bat	77 b	2010.05.05 16:41	Lefordítja a projektet
gtb_javadoc.bat	92 b	2010.05.05 16:41	Elkészíti a dokumentációt

gtb_run.bat	47 b	2010.05.05 16:41	Elindítja a programot
mapper16.txt	2026 b	2010.05.05 23:46	A végleges térkép
OLVASSEL.txt	5390 b	2010.05.05 16:41	Fordítási és futtatási utasítások, egyszerű szöveges formában

13.1.2 Fordítás és telepítés

A fordítás megkönnyítése érdekében egy batch fájlt hoztunk létre. Így a projekt főkönyvtárában a parancssorban csak ki kell adni a "gtb_compile" parancsot.

(A projekt főkönyvtára az a könyvtár, ahol megtalálható a "gtb_compile.bat" fájl és a "src" alkönyvtár)

Ez a következő parancsokat hajtja végre:

- 1) `mkdir build`
- 2) `javac ./src/GrandTheftBank/*.java -d ./build/`

Azaz

- 1) létrehoz egy "build" nevű könyvtárat
- 2) lefordítja a "src/GrandTheftBank/" könyvtárban lévő .java kiterjesztésű fájlokat és a "build/" könyvtárba menti az elkészült .class kiterjesztésű fájlokat.

Nagyon fontos, hogy ahhoz, hogy a "javac" parancsot használni tudjuk a PATH környezeti változóban szerepelnie kell a JDK bináris könyvtárának.

Amennyiben a "javac" parancs beírására ismeretlen parancsra panaszkodik a futtatandó környezet, kérjük adják hozzá a PATH környezeti változóhoz a JDK aktuális könyvtárát a következő paranccsal:

```
set PATH=%PATH%;C:\Program Files\Java\jdk1.6.0_18\bin\
```

Fontos, hogy a "C:\Program Files\Java\jdk1.6.0_18\bin\" rész helyett a JDK aktuális könyvtárát kell beírni, ami futtatandó környezetenként változhat!

(Általános esetben csak a 1.6.0_18 verziószámot kell kicserélni a gépen lévő JDK aktuális verziószámára. Amennyiben a JDK megtalálása közben problémába ütközik kérje a gép rendszergazdájának segítségét!)

A PATH beállítása után újra kell indítani a parancssort, mert csak akkor lép érvénybe a változás!

Dokumentáció létrehozása: a fordítás sikeressége utána adja ki a "gtb_javadoc" parancsot.

Ez a következő parancsokat hajtja végre:

- 1) `javadoc ./src/GrandTheftBank/*.java -d ./javadoc/ -charset utf-8`
- 2) `start ./javadoc/index.html`

Azaz

- 1) generálja a dokumentációt a "src/GrandTheftBank/" mappában lévő .java kiterjesztésű fájlokból a "javadoc/" könyvtárba
- 2) megnyitja a dokumentációt a gép alapértelmezett böngészőjében

13.1.3 Futtatás

A futtatás megkönnyítése érdekében egy batch fájlt hoztunk létre. Így a projekt főkönyvtárában a parancssorban csak ki kell adni a "gtb_run" parancsot.

Ez a következő parancsot hajtja végre: "java -classpath ./build/ GrandTheftBank.Main", azaz elindítja a "build/"-ból kiindulva a GrandTheftBank package Main classjának belépési pontját.

Figyelem! A játék grafikus megjelenítéséhez 1200x700felbontású monitor szükséges.

A játék irányítása a következő-képpen módosult:

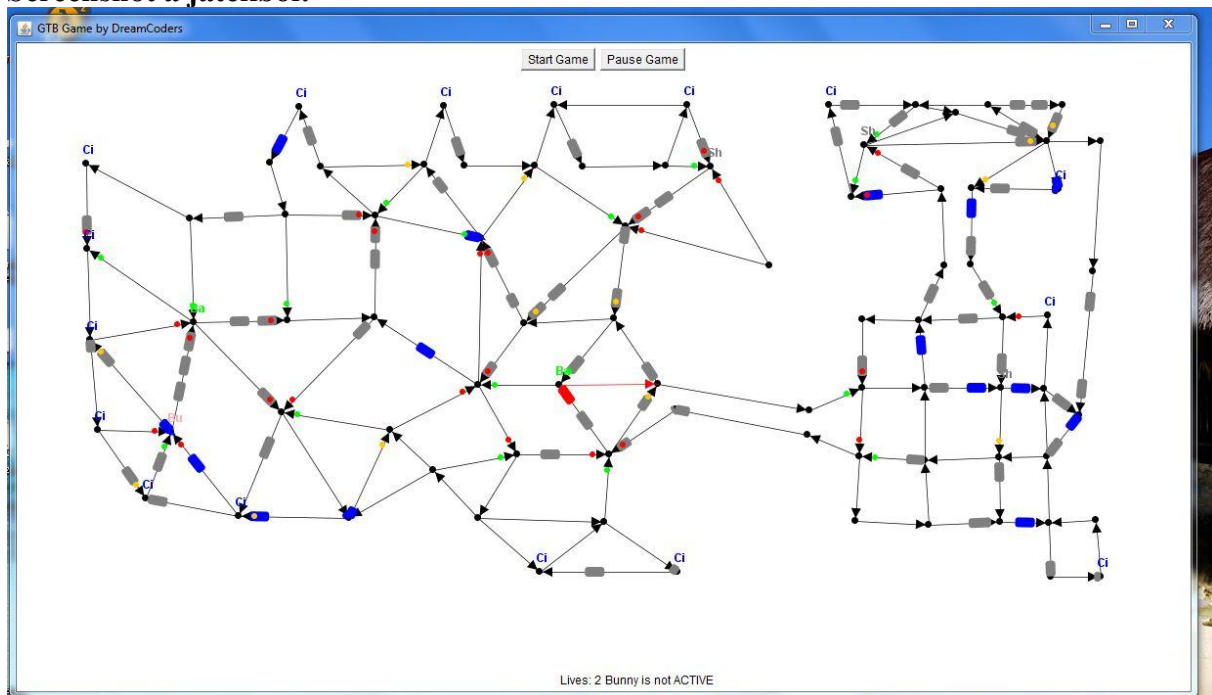
UP	Gyorsít
DOWN	Fékez
LEFT	A kereszteződés utáni utcák közti léptetés
RIGHT	A kereszteződés utáni utcák közti léptetés
B	Ha a robber kiment a városból, a „B” hatására („back”) jön vissza.

A játékban nyereség illetve veszteség esetén MessageBox jelenik meg és ebben értesül a játékos. A város elhagyása esetén nem jelenik meg MessageBox, mivel a tesztelés során ezt a funkciót zavarónak éreztük.

A képernyő alján a robber élete(i) illetve a Bunny ideje és állapota látszik szövegesen.

Sok sikert kíván a DreamCoders csapat a játékban!

Screenshot a játékból:



13.2 Értékelés

A közös projekt a kezdetektől fogja megfelelően halad, minden az ütemtervhez igazodva elkészült és megfelelő minőségűnek bizonyult.

A csapat stabilan tudja nyújtani az elvárásoknak megfelelő eredményeket.

Az optimalizált munkavégzés végett dinamikusán osztjuk fel egymás között a munkafolyamatokat. Mindenki olyan részhalmazát végzi el a munkának, amit a leggyorsabban és legjobban ő tud elvégezni.

A Grafikus felület implementálásában Angeli és Mikó nagyobb részt vettek ki.

A projektet eredményesnek és tanulságosnak érezzük. A terveinket sikerült betartani a munkavégzés során teljes kooperáció és racionális egyetértés volt.

Tag neve	Munka órában	Munka százalékban
Angeli Róbert	77,25 óra	24,3%
Koczka Tamás	81 óra	25,5%
Kovács Balázs	82 óra	25,8%
Mikó Norbert	77,25 óra	24,3%

13.3 Napló

Kezdet	Idő	Résztvevők	Leírás
2010.04.27. 20:00	1 óra	Angeli Koczka Kovács Mikó	Értekezlet. Döntés: - Angeli és Mikó leimplementálja a GUI-t - Koczka és Kovács segít a debuggolásban, szükség esetén korrigálja a modelt. - Mindenki teszteli a játékot
2010.04.29. 16:00	2 óra	Mikó	GUI implementálása
2010.04.08. 16:30	2 óra	Angeli	GUI implementálása
2010.05.01. 14:00	5 óra	Mikó Angeli	GUI implementálása
2010.05.01. 18:00	2 óra	Kovács	Debuggolás és korrekció
2010.05.01. 18:00	2 óra	Koczka	Debuggolás és korrekció
2010.04.10. 15:00	2 óra	Angeli Koczka Mikó	Tesztelés
2010.05.02. 18:00	3 óra	Angeli	GUI korrekció
2010.04.13. 10:00	2 óra	Mikó	Kommentelés és dokumentálás
2010.05.05. 16:00	5 óra	Angeli	Kommentelés és dokumentálás
2010.05.05. 19:00	2,5 óra	Mikó	Dokumentáció összeillesztése, formázása