KINDERGARTEN

46 – InFoka

Konzulens:

Dady Róbert

Csapattagok:

Bors Alpár Szabolcs	Z0AVHU	alpijoe@sch.bme.hu
Szegedi Tamás	AASWGX	szedzsi@sch.bme.hu
Tóth Tamás	EWYXK4	tommey@freemail.hu

2006. május 15.

2. fejezet
Követelmények, projekt, funkcionalitás

2.1 Követelmény definíció

2.1.1 A program célja, alapvető feladata

A program nem más mint egy ügyességi játék, ahol a feladat az óvodások összegyűjtése. A játék célja az óvó néni vezetésével a gyerekek összegyűjtése, és visszakísérése az óvodába. Részletesebb leírás a játék leírásánál található.

A csoportunk célja egy olyan játék létrehozása, amely megfelelő kikapcsolódást nyújt, illetve minden olyan gépen futtatható, ahol megtalálható a JAVA futtatókörnyezet. A fejlesztés során külön hangsúlyt fektetünk a körültekintő dokumentációnak, ahol különleges hangsúlyt kap az UML modellező rendszer minél tökéletesebb használata, emellett fontos szempont még számunkra a közös munka mikéntjének megismerése és hatékony alkalmazása.

2.1.2 A fejlesztőkörnyezet

A Rational Rose nevű programot használjuk arra, hogy vizuálisan megtervezzük a feladat modelljét, itt a drag and drop módszerrel lehet összerakni a kívánt modellt, majd a program ebből képes különböző nyelveken (C++, JAVA, Oracle, …) kódot generálni. Az így keletkezett forráskód további fejlesztésére a NetBeans, illetve Eclipse nevű fejlesztő környezetet használjunk.
A dokumentumokat WORD formátumban készítjük el, majd a CutePDF nevű program segítségével mentjük le PDF formátumban. A napló készítése is hasonló elven működik.

2.1.3 A futtatáshoz szükséges környezet

Legfontosabb része ennek a Java Runtime Enviroment és egy olyan számítógép, ami ezt futtatni képes (A SUN ajánlása szerint PCre: Pentium 166 MHz vagy gyorsabb processzor és 32 MB memória). További szükséges elem a játék futtatásához a grafikus képernyő és billentyűzet. Ez egy minimum követelmény, tehát ha zökkenő mentesen szeretnék élvezni a játékot kicsit erősebb gép javallott.
Magának a programnak nem lesz nagy memóriaigénye, de mivel grafikus elemeket is tartalmaz, ezért számításaink szerint 1-2 MB között fog mozogni.

2.1.4 A felhasználói felület

A játék végső változata grafikus felhasználói felülettel rendelkezik. A programot a felhasználó billentyűzet segítségével irányítja.

2.1.5 Minőségi tényezők

Ezen belül is kiemelném a Teljesítmény, Újrafelhasználhatóság, Rugalmasság és Felhasználhatóságot, ugyanis ezek teszik ki azokat a legfőbb tulajdonságokat, amiket egy mai szoftvernek teljesíteni-e kell.
Teljesítmény: A cél az, hogy a játék élvezhető legyen, megfelelő kikapcsolódást nyújtva a felhasználónak. Ezért folyamatosan törekedni fogunk a fejlesztés során, hogy egy olyan élmény, illetve hangulati világot alakítsunk ki, ami kellemes kikapcsolódást tud nyújtani.
Újrafelhasználhatóság: A cél az, hogy az Objektum Orientált paradigmának megfelelően úgy hozzuk létre a programunkat, hogy azt később könnyen lehessen módosítani, pl. a grafikus felületet át lehessen ültetni egy másik alkalmazásba, vagy újabb tulajdonságokkal ruházzuk fel azt.
Rugalmasság: A rugalmasság kérdése a játék fejlesztőkörnyezetének hála egyszerű, hiszen a programnak minden olyan környezetben futnia kell, ahol létezik Java futtatókörnyezet.
Felhasználhatóság: Célunk között szerepel az is, hogy a játék használata olyan embernek se okozzon különösebb fennakadást, aki nem jártas a számítástechnikában, persze egy olyan szintű tudást feltételez, mint a számítógép és a játék elindítása.

2.1.6 A software minősítése

A kifejlesztett szoftver akkor megfelelő, ha teljesíti azokat az elvárásokat, amiket fentebb leírtam. Ennek ellenőrzésére külön szervezetek állnak rendelkezésre, de az esetünkben ellenőrizni lehet a játék futtatásával, illetve a forráskód és modell hasonlóságának vizsgálatával.

2.1.7 A kibocsátás

A program kibocsátása először a konzulens felé történik a forráskód csatolásával. Az ellenőrzés és az értékelés után a szoftver hozzáférhető lesz az interneten is.

2.2 Project terv

2.2.1 A fejlesztői csapat

Csapattagok:
Bors Alpár Szabolcs
Szegedi Tamás
Tóth Tamás

A csapatban nincs kifejezett feladatköre senkinek, minden feladatot felosztunk egymás között, mind a dokumentálás, mind a kódolás terén, így is párhuzamosítva a feladatokat a jobb időkihasználás érdekében.

2.2.2 Életciklus modell

Első lépés a program megtervezése, milyen dinamikus- és objektummodelleket fog tartalmazni. Második lépésként a fenti adatokból a skeletont kell implementálni. Ha minden rendesen lett megtervezve az első lépésben itt nem merülhet fel komplikáció a folyamatban.

Következő lépésként egy prototípust kell létrehozni, amely segítségével tesztelhetjük az esetleges hibákat. Ilyenek például olyan logikai vagy funkcionalitásbeli problémák, amelyek a forrásból nem tűntek ki.
A tesztelés befejezése után, mikor a prototípus működése megfelelőnek mondható, akkor kezdődhet a grafikus felület kialakítása és a végleges formára hozás. Ezen folyamat alatt is fontos az új részek folyamatos tesztelése, a hibák következetes javítása. Ha ennek a fejlesztési folyamatnak is a végére értünk, akkor mondhatjuk, hogy elkészült a programunk első teljes változata. Az eddig elkészült részt kell leadni a konzulensnél.

2.2.3 Szervezési struktúra

A csapatunk három főből áll. A tudásunkat nézve mindenkinek vannak kiemelkedő tulajdonságai, mégis úgy döntöttünk, hogy mindenki minden feladatrészből kivegye a maga részét, tehát gondolok itt arra, hogy a dokumentáció különböző pontjait egyszerre írhatjuk, így nagyon sok időt spórolhatunk meg, persze elsődleges szempontok között szerepel az is, hogy mindenki azonos mértékű feladatot kapjon, tehát ne legyen olyan tagja a csapatnak, aki túl sokat dolgozik és olyan se legyen aki tétlenkedne.
Az egyes találkozók alkalmával kerülnek felosztásra a feladatok, illetve ezen időpontok alatt beszéljük meg a játék megvalósításához szükséges részleteket. Erre a találkozóra általában a hét elején kerül sor, de itt meg kell jegyezzem, hogy a csapatból ketten szobatársak vagyunk, így csak a harmadik taggal kell találkozót szervezni. A következő találkozóra már mindenki elkészíti a rá kiszabott feladatot, és ha probléma akad, akkor közösen megoldjuk, illetve, ennél az alkalomnál kerül sor a részfeladatok „összefésülésére”.
A problémák zökkenőmentes megoldása és a kapcsolattartás folyamatos fenntartása miatt, egyéb segédprogramokat is használunk, mint pl:
Skype: Ennek a programnak van egy nagyszerű képessége, ami nem más, mint a konferencia beszélgetés, így könnyen a nap bármely szakában képesek vagyunk egymással beszélni, persze ez nem a személyes találkozók helyett van, hanem hét közben a távolság adta akadály leküzdésére, illetve az előre nem tervezett probléma gyors megoldására.
FTP: A csapatunk egyik tagjának van egy állandóan online FTP servere, ahol létesítettünk egy olyan fiókot, amihez csak nekünk van hozzáférésünk, itt tároljuk a projecthez tartozó dokumentációkat, forrásokat, így ha valaki kész van az ő rész feladatával egyből fel is töltheti.

MSN: Az MSN Messenger szerves része a kapcsolattartásunknak, hiszen ez a legegyszerűbb mondja egy kis probléma gyors megoldásának.

2.2.4 Fejlesztési ütemterv

A fejlesztésnek három főbb lépése van:

1. Skeleton: A cél az, hogy megtaláljuk és összeállítsuk azokat a dinamikus és objektum modelleket, amelyek a feladathoz kellenek. Ha ezzel megvagyunk, és helyes is, akkor már egy biztos alapja van a programunknak.

2. Prototípus: A fejlesztésnek az a része, mikor a kódunkat tesztelhetjük, ennél a résznél a grafikus felhasználó felület még nincs végeleges állapotban, különböző logikai és szemantikai helyességét vizsgálhatjuk a programunknak.

3. Grafikus változat: Az esteleges hibákat tartalmazó prototípus kijavított végleges változata, melyet kiegészítjük a végleges felhasználói felülettel.

2.2.5 Határidők

febr. 20.	Team bejelentkezése
febr. 27.	Követelmény, projekt, funkcionalitás
márc. 6.	Analízis modell kidolgozása 1.
márc. 13.	Analízis modell kidolgozása 2.
márc. 20.	Szkeleton tervezése
márc. 27.	Szkeleton beadása
ápr 3.	Prototípus koncepciója
ápr. 10.	Részletes tervek
ápr. 17.	
ápr. 24.	Prototípus beadása
máj. 2.	Grafikus felület specifikációja
máj. 8.	
máj. 15.	Grafikus változat beadása
máj. 19.	

2.2.6 Szükséges dokumentációk

Kétféle dokumentáció szükséges elkészíteni, az első amelyet a programozó készít, és leírja a program belső szerkezeti modelljét, amely fontos szerepet játszhat egy esetleges továbbfejlesztésnél. A másik fontos dokumentum a felhasználóknak készül, akiknek le kell írni érthető, számítástechnikai szakszavakat mellőzően a program működését, pl: Telepítési és felhasználói útmutató, mivel nem várhatjuk el egy átlag felhasználótól, hogy kiigazodjon a State Chartok és Use Case-k között. A program végül is nekik készül, tehát nem szabad, hogy bármiféle negatív hatást, csalódottságot keltsen bennünk, nekik szórakozást és kikapcsolódást kell hogy nyújtson.

2.3 A követelmények leírása

A játékot elindítva egy menürendszert láthatunk, amely a következő három pontból áll:

- Játék indítása
- Toplista
- Kilépés

Az első menüpontot választva tudjuk megkezdeni a játékot, amely három nehézségi fokozatban indítható: kezdő, közepes és mesteri. A nehézségi szintek közötti különbségről majd részletesebben is szó lesz.

A következő kép már maga a játék helyszíne, vagyis egy játszótér, melyen többek közt utakat, zöld mezőt, csokiautomatákat, játékboltokat, gyerekeket, kutyákat, egy óvó nénit és egy óvodát vélhetünk felfedezni.

Mind közül a legfontosabb az óvó néni, aki játékunk főszereplője, és akit az a megtiszteltetés ért, hogy mi irányíthatjuk. Ezt a billentyűzet segítségével tehetjük meg, azon belül is a 4 billentyűből álló nyilakkal, melyek lenyomásával, értelemszerűen, balra, lefele, jobbra és felfele mozoghatunk. További mozgási irányokra is lehetőség van, mégpedig észak-nyugati, észak-keleti, dél-nyugati és dél-keleti irányban. Ezekbe a megfelelő billentyűk együttes lenyomásának hatására mehetünk (észak=felfele nyíl, dél=lefele nyíl, kelet=jobbra nyíl, nyugat=balra nyíl).

A feladatunk pedig a kisgyerekek összegyűjtése, róluk a következő pontban lesz részletesebben szó. Ami a lényeg, hogy a pályán található csokiautomatákból csokikat vehetünk fel az óvó nénivel, de egyszerre legfeljebb csak 5 db csoki lehet nálunk. Ezek segítségével tudjuk rávenni a kis lurkókat, hogy jöjjenek velünk vonatozni és álljanak be a sorba mögénk. A cél minél több gyerek elvezetése az óvodába, melyet az óvoda felett elhelyezett számlálón számolunk és jelezzük a játékosnak. Azt, hogy egy egész pályányi kígyózó sort formálunk-e s úgy vezetjük el a gyerekeket az óvodába, vagy szépen kis csoportokban, mi döntjük el a stratégiánknak megfelelően.

A következő nagyon fontos szereplők a kisgyerekek, akik – mint már tudjuk - játékunk „célpontjai". Ők véletlenszerűen mozognak a pályán, akár álldogálhatnak egy helyben is. Amit tudni kell róluk, hogy szeretnek bámészkodni a játékboltok előtt és csokit enni. Ez a két tulajdonságuk olyannyira meghatározó, hogy teljesen lekötik a figyelmüket. Ha játékboltról van szó, akkor bizonyos valószínűséggel megállnak előttük, és csak csodálják mit sem törődve a környezetükkel s kiszakadva a sorból. Ha pedig csokiról van szó, bármit megtesznek, akár szót is fogadnak az óvó néniknek, azaz nekünk, hogy beálljanak szépen a sorba, vagy pedig, ha csokiautomatát látnak, ugyanúgy tesznek, mint a játékbolttal kapcsolatban. Amiket nem szeretnek viszont, azok a kutyák, amikor megugatják őket. Attól nagyon félnek, és ha csak egy ilyet is hallanak, megrémülve hagyják el a sort a menekülés reményében.

A helyszínre véletlenszerűen sétálnak be véletlenszerű időközönként. Ha pedig a kisgyerekek a helyszínünk széleihez kóborolnának, akkor irányt változtatva sétálnak tovább.

A harmadik mozgó szereplő a játékban a már említett és rémisztő kutyák, akik ugatásaiktól olyannyira rettegnek az ovisok. A kutyák is véletlenszerűen mozognak a pályán, véletlenszerűen sétálnak be a pályára, s véletlenszerűen ugatja meg az észrevett óvodást is. Amire ügyelnünk kell, hogy a már említett nehézségi szinttől függően több és több kutyával találhatjuk szembe magunkat. Ellenük csak egy megoldás létezik, ha kikerüljük őket. Ez egy nehezítés a játékban, hogy soha ne unhassuk meg.

A helyszínen helyet foglalnak még a játékboltok és a csokiautomaták, amelyek az utak mentén találhatók meg. Játékboltok a nehézségi szinttől függő számban. S a pálya jobb felső sarkában pedig maga az óvoda foglal helyet, ahová a gyerekeket vinni szeretnénk.

A játék menete során, miután az ovisokhoz odamentünk egy csokival, beállnak mögénk, mintha vonatoznának. Ilyenkor az általunk ismert módon, mintha az előttük lévő vállát fognák, haladnak a sorban előttük álló után. Az egész sort pedig mi irányítjuk, legvégül az óvoda fele. Ügyelnünk kell azonban a kutyákra, a játékboltokra és a csokiautomatákra,

hiszen mindhárom a gyerekek sorból való kiszakadását jelentheti. Ha pedig egy gyerek kiszakad a sorból, akkor megszakad a sor, jobban mondva kettészakad. De: amelyik sor elején nincs óvó néni, az a sor felbomlik, mert a gyerekek nem fogják tudni, hogy merre menjenek. Ezért mindenki visszatér véletlenszerű mozgásokból álló életébe és sétálgatnak fel-alá. Jobb esetben csak 1-2 ovis szakad le a sor végéről, de rosszabb esetben az egész sor leszakadhat és szétszéledhet, ha - például - a mögöttünk lévőt ijesztette meg a kutya. Ezért fontos figyelnünk, mert értékes pontokat és időt veszíthetünk el egy kis figyelmetlenséggel. Az óvoda felett elhelyezett számlálón látható pontszámunkat egy név beírása után a játék elmenti egy toplistába, melyet később a kettes menüponttal nézhetünk meg.

A játék akkor ér véget, ha lejár a gyerek begyűjtésére szánt játékidő, azaz 3 perc. Cél ezalatt az idő alatt minél nagyobb pontszám elérése.

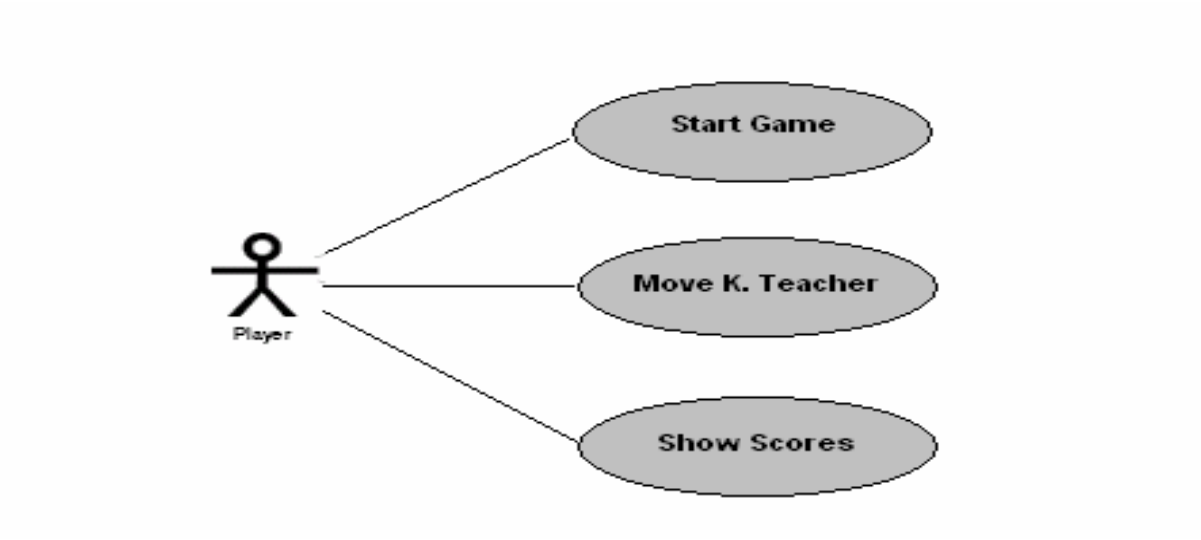
2.4 Szótár

beáll	a gyerekek beállnak a sorba, ha csokival kínálják őket, mindig a meglévő sor végére állnak
besétál	a játéktér széleiből, véletlenszerű pontokból sétálnak be az ovisok a pályára
csoki	ezzel vehetők rá a gyerekek, hogy beálljanak a sorba
csokiautomata	innen vehet fel az óvó néni csokikat, vagy pedig ennek a hatására szakadhatnak ki a gyerekek a sorból
elhagy	a gyerekek elhagyhatják a sort csokiautomata, játékbolt vagy kutyaugatás miatt, ilyenkor felbomlik az egész mögöttük levő sor
felbomlik	ha egy sor elején nincs óvó néni, akkor egy sor felbomlik és a gyerekek tovább sétálgatnak magukban
gyerek	véletlenszerűen mozognak, beállhatnak a sorba vagy felszakíthatják azt
játékbolt	bizonyos valószínűséggel felbomolhatnak a sorok a játékboltok látványának hatására
játszótér	utakból, zöldes részekből álló helyszín, ahol gyerekek, kutyák s egy óvó néni kap szerepet, továbbá helyet

	foglalnak még a csokiautomaták és a játékboltok, lényegében a játék helyszíne
kisgyerek	itt a gyerek szinonimája
kiszakad	itt az elhagy szinonimája
kutya	véletlenszerűen mozognak, megugathatják az ovisokat
lurkó	itt a gyerek szinonimája
megugat	a kutyák megugathatják a gyerekeket, akik ennek hallatán kiszakadnak a sorból
nehézségi fokozat	játék indítás előtt választható, ettől függően egyre több kutyával és játékbolttal lesz dolgunk
nehézségi szint	itt a nehézségi fokozat szinonimája
ovis	itt a gyerek szinonimája
óvó néni	őket irányítjuk, feladatuk a gyerekek összegyűjtése és óvodába vezetése
óvoda	ez a célállomás, ide kell behozni a gyerekeket
összegyűjt	az óvónéni feladata a gyerekek összegyűjtése kisebb vagy nagyobb számban, hogy majd elvezethesse őket
pontszám	a játék célja minél több pont gyűjtése, minden óvodába vezetett gyerekért 1 pont jár, a pontok elmentésre kerülnek majd
számláló	az óvoda felett látható, mutatja a pontszámot
vezetés	az óvónéni feladata a gyerekek óvodába való vezetése, amiért a pontokat kapjuk

2.5 Essential Use Case-ek

2.5.1 Diagramok



2.5.2 Use Case leírások

Use-Case	Start new Game
Actor	Player
Leírás	A játékos új játékot kezd

Use-Case	Move Kindergarten Teacher
Actor	Player
Leírás	A játékos mozgatja az óvónőt

Use-Case	Show High Scores
Actor	Player
Leírás	A játékos megnézi a maximum pontokat

3.fejezet
Analízis modell

3.1 Osztályok leírása

3.1.1 Ovodas

Ez az objektum fogja az óvodásokat reprezentálni a játékban. Van egy (következő óvodásra és egy előző óvodásra mutató) referenciája, illetve egy változója, amivel tudja, hogy éppen milyen állapotban van. Az óvónő beteheti a sorba egy csokiért cserébe. Ezután kiszakadhat a sorból, ha kiszakad, akkor azt egy külső objektum hatására teszi. Kiszakadásának okai lehetnek: kutya, játékbolt/ csokiautomata objektummal való találkozás, illetve ha olyan sorban van, aminek az elején nincs óvónő. Kiszakadásainak a valószínűsége a kutyaugatás esetében 1, míg a többi esetben véletlenszerű. Ha a kutyával találkozik, akkor menekül, ha valamelyik állóobjektummal, akkor megáll és bámészkodik.

3.1.2 Palya

Ez az objektum generálja le majd a pályaelemeket (egymás utáni sorokban), amelyeket egy Vector tömbben fog tárolni, majd minden pályaelemnek beállítja a 4 szomszédját, melyeket a pályaelemek a saját tömbjükben tárolnak el. Ezeket a referenciákkal valósítja meg. Ezek után pedig létrehozza a többi mozgó és nem mozgó objektumokat és hozzárendeli őket egy-egy pályaelemhez véletlenszerűen. Megadhatjuk a pályán generálandó objektumok számát is.

3.1.3 Csokiautomata

Ez az objektum a játékban fontos szerepet játszó csokiautomatát testesíti meg, nehézségi szinttől függően lehet belőle több is, kevesebb is. Az óvónő csokit vehet ki belőle (legfeljebb ötöt), a gyerekeket pedig bámészkodásra késztetheti.

3.1.4 Kutya

Ez az objektum a kutyát reprezentálja. Aktív objektum, mert mozog és megugatja a gyerekeket. Mozgása véletlenszerű. Ugatása által a gyerekeket menekülésre készteti.

3.1.5 Jatek

Ez az osztály a játék fő osztálya, kezeli a menürendszert, s ezen belül az új játék kezdetét, a toplista megtekintését és a kilépést. Ez az osztály tartalmazza a pálya és a toplista osztályt.

3.1.6 Szamlalo

Ez az objektum számolja a pontjainkat. Akkor növekszik az értéke 1-el, ha beviszünk 1 óvodást az óvodába. Az óvoda feletti pályaelemen kap helyet. Függ az óvodába vitt gyerekek számától.

3.1.7 Ovoda

Ez az objektum egy kilépő pontot jelent a játékban, hiszen ide kell hozni az óvónőnek a gyerekeket. Ide jön az óvónő a gyerekekkel.

3.1.8 Jatekbolt

Ez az objektum reprezentálja a játékboltot, pályán való helyfoglalása véletlenszerű, melyet a pálya objektum szab ki azáltal, hogy meghatározza a pályaelem referenciáját, amelyen helyet fog kapni.

3.1.9 Ovono

Ez az objektum az óvónőt testesíti meg a játékban. A játékos egyedül az óvónőt képes irányítani a játékban. Rendelkezik egy változóval, mely a nála lévő csokik számát határozza meg, amelynek értéke csökken az egyes meggyőzött gyerekek sorba állításával. Van egy referenciája, amelyik a mögötte lévő első gyerekre mutat. Maximum öt darab csokijainak számát feltöltheti a csokiautomatán keresztül.

3.1.10 PalyaElem

Ez az objektum a játékban a földet testesíti meg, amelyen az egyéb objektumok foglalnak helyet. Egy vector tömbben tárolja a rajta levő objektumok milyenségét és számát, míg egy másikban a szomszédos pályaelemek referenciáit. Ez mozgatja az összes objektumot az atad/atvesz függvényekkel.

3.1.11 Mozgat -- Interface

Ez az interface szolgál a mozgó objektumok mozgatásához. Magában foglal egy mozog függvényt, melynek segítségével történik a pályaelemeken való lépkedés.

3.1.12 Fajl

Ez az objektum arra való, hogy a toplistát fájlba mentse a játék befejeztével majd egy játék indításkor pedig onnan betöltse.

3.1.13 Ido

Ez az objektum az idő mérését szolgáltatja a játékban. Ez figyeli mikor telik le az idő.

3.1.14 Toplista

Ennek az objektumnak a segítségével tudjuk megjeleníteni azokat az értékeket, amik a high score táblázatban vannak.

3.1.15 Nehezseg

Ennek az objektumnak a játék indításakor van fontos szerepe, hiszen a segítségével tudjuk beállítani, hogy milyen nehézségi szinten játsszuk a játékot, tehát a játékboltok, kutyák.. számát a pályán.

3.1.16 KozosOs

Egy abstract osztály, melyet az egyes objektumok valósítanak meg, segítségével tudnak az egymás mellett álló objektumok „köszönni” egymásnak, így tudjuk, hogy pl. egy kutya objektumnál a OvodasUdvozol függvényét csak egy ovodás hívhat meg, tehát a kutyának ennek megfelelően kell reagálnia.

3.2 Objektum katalógus

3.2.1 Ovodas

Ovodások osztálya. Őket kell összegyűjtenünk a feladat során.

Alaposztályok: KozosOs
Példányok száma: [1...*]
Perzisztencia: dinamikus
Relációk: az UML diagrammokon látszik.

Változók:	unsigned int allapot;	0 - mászkál, 1 - sorban van, 2 - bámészkodik
	KozosOs Elotte	előtte lévő Óvónő vagy Óvodás referenciája
	Ovodas Utana	utána lévő óvodás referenciája
Szolgáltatások:	void Sorbaall();	közli az óvónővel, hogy megfogta a kezét, illetve az óvónő mögöttivel, hogy most már ő fogja a kezét
	void Menekul();	kiszakad() után, ha a kutya megugatja:D véletlenszerűen mozog
	void Bameszkodik();	ha játékbolt, vagy csokiautomata miatt kiszakad
	void OvonoUdvozol(Ovono);	ezzel üdvözli az óvónő az óvodást
	void KutyaMegugat();	ezzel „üdvözli” a kutya az óvódást, s megugatja.

Felelősségek: PályaElem objektummal, lásd UML diagrammok

3.2.2 Palya

Ő építi fel a pályát a pályaelemekből. Megadjuk hány elemből építse fel, illetve miből mennyi legyen fenn a pályán, úgy mint a Csokiautomatá száma, Kutya...

Alaposztályok: Object
Példányok száma: 1
Perzisztencia: dinamikus
Relációk: az UML diagrammokon látszik.
Változók: Vector PalyaElemek; vector tárolóban tároljuk a pályaelemeket
int SorHossz; ennyi pályaelem lesz egy sorban
unsigned int OsszPalyaElem; megadjuk, hogy hány elemből épüljön fel a pálya
unsigned int OsszOvodas; mennyi óvodás legyen összesen
unsigned int OsszCsokiautomata; mennyi csokiautomata legyen összesen
unsigned int OsszKutya; mennyi kutya legyen összesen
unsigned int OsszJatekBolt; mennyi játékbolt legyen összesen
Szolgáltatások: void palyaepit(); a pályaelemeket elrendezi a pályán
void OsszOvodasBeallit(unsigned int); Ovodas-ok száma
void OsszCsokiautomataBeallit(unsigned int) Csokiautomaták száma
void OsszKutyaBeallit(unsigned int) Kutyák száma
void OsszJatekboltBeallit(unsigned int) Játékboltok száma
void SzomszedBeallit() beállítja a pályaelemek szomszédait
Felelősségek:

3.2.3 Csokiautomata:

A Csokiautomaták osztálya. Segítségével juthat az óvónő csokikhoz, amelyek az óvodások láncba fűzéséhez szükségesek.

Alaposztályok: KozosOs
Példányok száma: [1...*]
Perzisztencia: dinamikus
Relációk: az UML diagrammokon látszik.

Változók:	unsigned int CsokiSzam;	a csokiautomatában lévő csokik száma (nehézségi szinttől függ)
Szolgáltatások:	void CsokiSzamCsokkent();	csökkenti a felvett csokik számával a csokiautomataban lévő csokik számát
	void OvonoUdvozoll(Ovono);	ezt hívja meg az óvónő, amikor találkozik a csokiautomatával
	void OvodasUdvozol(Ovodas);	ezt hívja meg az óvódás, amikor találkozik a csokiautomatával

Felelősségek:

3.2.4 Kutya

A kutyák osztálya. Ők az egyik olyan objektumok, melyek felszakítják az óvodások sorrendjét.

Alaposztályok: KozosOs
Példányok száma: [1...*]
Perzisztencia: dinamikus
Relációk: az UML diagrammokon látszik.
Változók: nincs

Szolgáltatások:	void OvodasUdvozol(Ovodas);	ezt hívja meg az óvodás, amikor találkozik a kutyával.

Felelősségek: PályaElem objektummal, UML diagrammon látszik

3.2.5 Jatek

A Kindergarten főosztálya, ez a legelső példányosított osztály, amely a játékot indítja. Illetve ebből érhető el a Toplista.

Alaposztályok: Object
Példányok száma: 1
Perzisztencia: dinamikus

Relációk: az UML diagrammokon látszik.

Változók:	unsigned int szint;	éppen milyen szinten vagyunk: 1 - kezdő, 2 - normál, 3 - mester
	unsigned int ido;	az időkorlát, amin belül teljesítenünk kell
Szolgáltatások:	void JatekIndit();	elindítjuk és létrehozzuk a játékot
	void EredmenytNez();	megtekintjük a toplistát
	void Kilep();	kiléphetünk a programból

Felelősségek:

3.2.6 Szamlalo

Alaposztályok: Object

Példányok száma: 1

Perzisztencia: dinamikus

Relációk: az UML diagrammokon látszik.

Változók:	unsigned static int Eredmeny;	aktuális eredmény tarolása
Szolgáltatások:	unsigned int EredmenytLekerdez();	lekérdezi az eredményt folyamatosan
	void EredmenytNovel(unsigned int);	megnöveli az eredményt a beérkező gyerekek számával arányosan

Felelősségek: ismernie kell az Ovodába bemenő gyerekek számát

3.2.7 Ovoda

A kilépési pont.

Alaposztályok: KozosOs

Példányok száma: 1

Perzisztencia: statikus

Relációk: az UML diagrammokon látszik.

Változók: nincs

Szolgáltatások:	void OvodasLancbolTorol();	ezzel kéne az óvó néni mögül a gyerekeket

		kivenni a láncolt listából, ha ugye beérkeztek az oviba
	void OvonoUdvozol(Ovono);	ezt hívja meg az óvónő, amikor odaérkezik az óvodához.

Felelősségek:

3.2.8 Jatekbolt

A játékboltok osztálya. Ez a másik olyan objektum, amely kiszakíthatja a gyereket a sorból, nagyszerű kínálataival. (Amelyet sajnos nem áll módunkban beleimplementálni a kész programba.)

Alaposztályok: KozosOs
Példányok száma: [1..*]
Perzisztencia: statikus
Relációk: az UML diagrammokon látszik.
Változók: nincs

Szolgáltatások:	void OvodasUdvozol(Ovodas);	ezt hívja meg az óvodás, amikor szemben találja magát a játékbolttal.

Felelősségek:

3.2.9 Ovono

A kedves felhasználó ezen nőszemélyt irányíthatja, s vele kell összegyűjtenie a pályán lézengő Óvodásokat, és az Óvodába juttatnia, adott időn belül és nehezítések között.

Alaposztályok: KozosOs
Példányok száma: 1
Perzisztencia: dinamikus
Relációk: az UML diagrammokon látszik.

Változók:	int CsokiSzam	nála lévő csokik száma
	int MaxCsokiSzam	ennyi lehet nála maximum
	Ovodas ElsoOvodas	az sor elso ovodasara mutato pointer

Szolgáltatások: void OvodasLancbaFuz(Ovodas); ez fűzi láncba az aktuális gyereket

void CsokiSzamCsokkent(); csokik számának 1el való csökkentése

void CsokivalFeltolt(); csokik számának int-el való növelése (mer ugye amennyit felvesz)

unsigned int CsokiSzamLekerdez(); ezzel tudjuk lekérdezni a csokik számát.

void OvodasUdvozol(Ovodas); ezt hívja meg az óvodás, amikor találkozik az óvónővel.

Ovodas OvodasLead(); ezzel a függvénnyel tudjuk az óvodánál letenni az óvódásokat.

Felelősségek: PályaElem objektummal, UML diagrammon látszik

3.2.10 PalyaElem

A pályát felépítő elemek, amelyeken a különböző objektumok tartózkodnak. Ezeken zajlik az élet. Az egyik pályaelemről a másikra vándorlást is ezek végzik (az átad és átvesz függvények segítségével, egy-egy mozgó objektum Mozog függvényével kezdeményezve).

Alaposztályok: Object
Példányok száma: n
Perzisztencia: dinamikus
Relációk: az UML diagrammokon látszik.

Változók: Vector Szomszedok; A szomszédos pályaelemek halmaza.

Vector Objektumok; A pályaelemen tartózkodó objektumok.

Szolgáltatások: PalyaElem Atad(KozosOs,enum); Átadja az Objektumot, a szomszédjának.

Void Atvesz(KozosOs); Átveszi az Objektumot, a szomszédjától.

Vector LekerdezObjektumok(); Visszaadja a rajtalévő objektumokat.

Felelősségek

3.2.11 Mozgat -- Interface

A mozgó osztályokat egybefogó interface.

Alaposztályok: Object
Példányok száma: 0
Perzisztencia: -
Relációk: az UML diagrammokon látszik.
Változók:-
Szolgáltatások: Mozog(); A mozgó objektumok ezt valósítják meg és használják mozgásra.
Felelősségek:

3.1.12 Fajl

Ez a fájlkezelő műveleteket megvalósító osztály.

Alaposztályok: Object
Példányok száma: 1
Perzisztencia: dinamikus
Relációk: az UML diagrammokon látszik.

Változók:	String FajlNev;	A toplistát tároló fájl neve.
Szolgáltatások:	void FajlbaMentes(List)	Kimenti fájlba a toplistát.
	List FajlbolOlvas()	Beolvassa fájlból a toplistát.

Felelősségek:

3.1.13 Ido

Ez az időt megtestesítő osztály.

Alaposztályok: Object
Példányok száma: 1
Perzisztencia: dinamikus

Relációk: az UML diagrammokon látszik.

Változók: unsigned int Ido; Ebben tároljuk a lejáró időt.

Szolgáltatások: void IdoTelik(); Ezzel a függvényel telik az idő.

boolean Vegevan(); Így ellenőrizhetjük, hogy lejárt-e az idő már.

Felelősségek:

3.1.14 Toplista

Toplistát megvalósító osztály.

Alaposztályok: Object

Példányok száma: 1

Perzisztencia: dinamikus

Relációk: az UML diagrammokon látszik.

Változók: List Lista; ebben tároljuk a toplistát átmenetileg

Szolgáltatások: void Megjelenit(); ezzel jeleníthetjük meg a toplistát a képernyőn.

void Mentes(List); ezzel menthetjük el az aktuális toplistánkat.

Felelősségek:

3.1.15 Nehezseg

Ez az osztály a nehézségi szinteket testesíti meg.

Alaposztályok: Object

Példányok száma: 1

Perzisztencia: dinamikus

Relációk: az UML diagrammokon látszik.

Változók: unsigned int Szint; ebben tároljuk a kiválasztott szintet.

Szolgáltatások: void BeallitSzint(unsigned int); ezzel állíthatjuk be a nehézségi szintet.

unsigned int SzintLekerdez();	ezzel tudhatjuk meg, hogy milyen szinten játszunk éppen.
void Konnyu();	a könnyű nehézségi szintet beállító fv.
void Kozepes();	a közepes nehézségi szintet beállító fv.
void Nehez();	a nehéz nehézségi szintet beállító fv.

Felelősségek: Palya objektummal, UML diagrammok látszik.

3.2.16 KozosOs – Abstract

Alaposztályok:-
Példányok száma: 0
Perzisztencia: dinamikus
Relációk: az UML diagrammokon látszik.

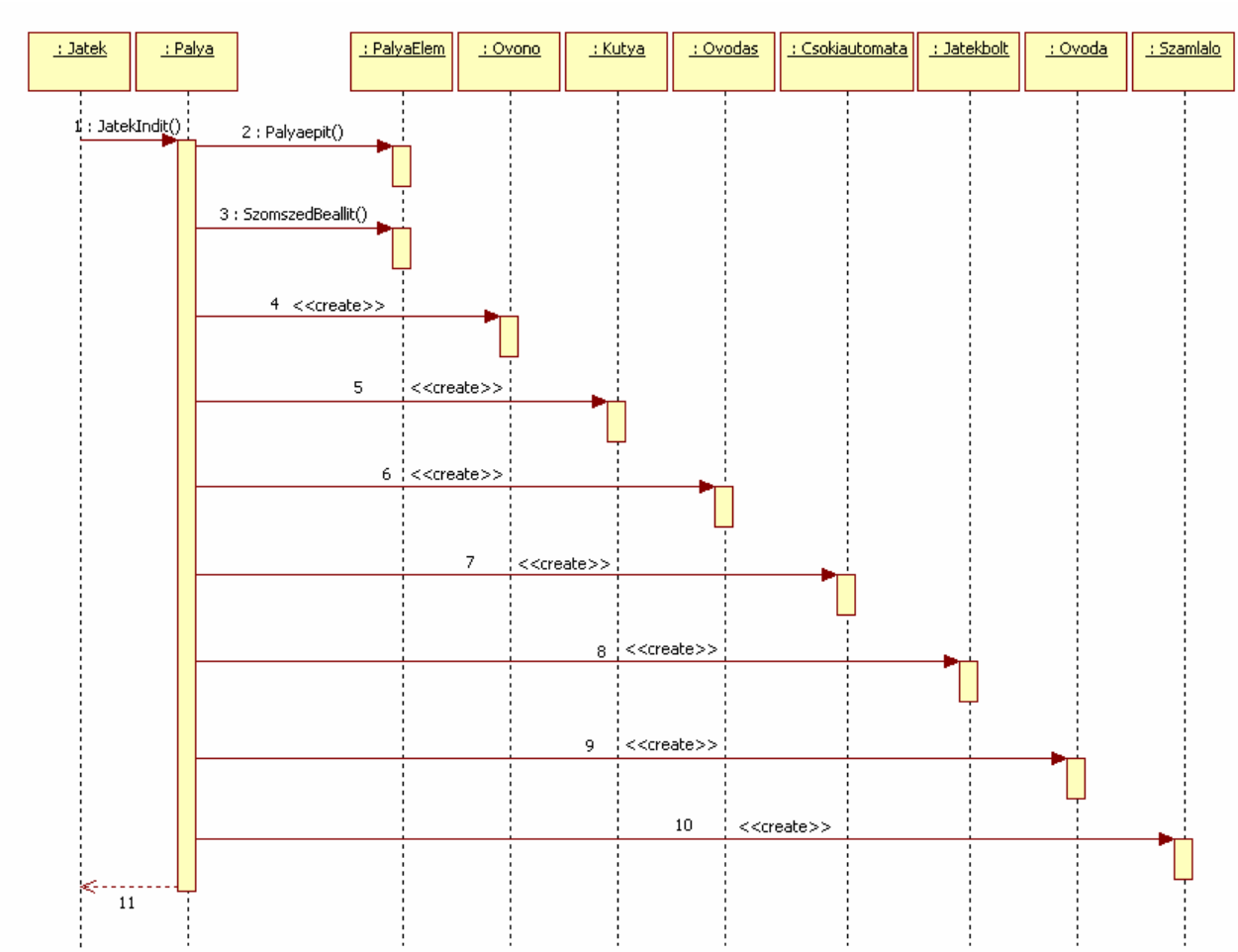
Változók: PalyaElem AktHelyzet()		egy pályaelem aktuális helyzetét kérjük le
Szolgáltatások:	void OvodasUdvozol(Ovodas)	- akkor hívódik meg mikor ovodás kerül a pályaelemre
	Void OvonoUdvozol(Ovono)	- akkor hívódik meg ha egy ovono lép a pályaelemre
	Void KutyaMegugat(Kutya)	- E függvény hatására szakadnak ki az ovodások a sorból

3.3 Statikus struktúra diagrammok

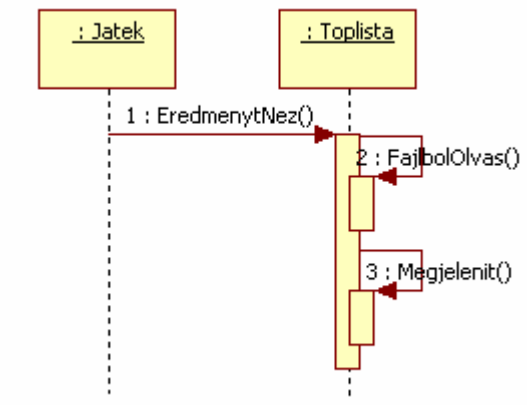
A mellékelt ábrán látható.

3.4 Szekvencia diagramok (inicializálásra, és a use-case-ekre)

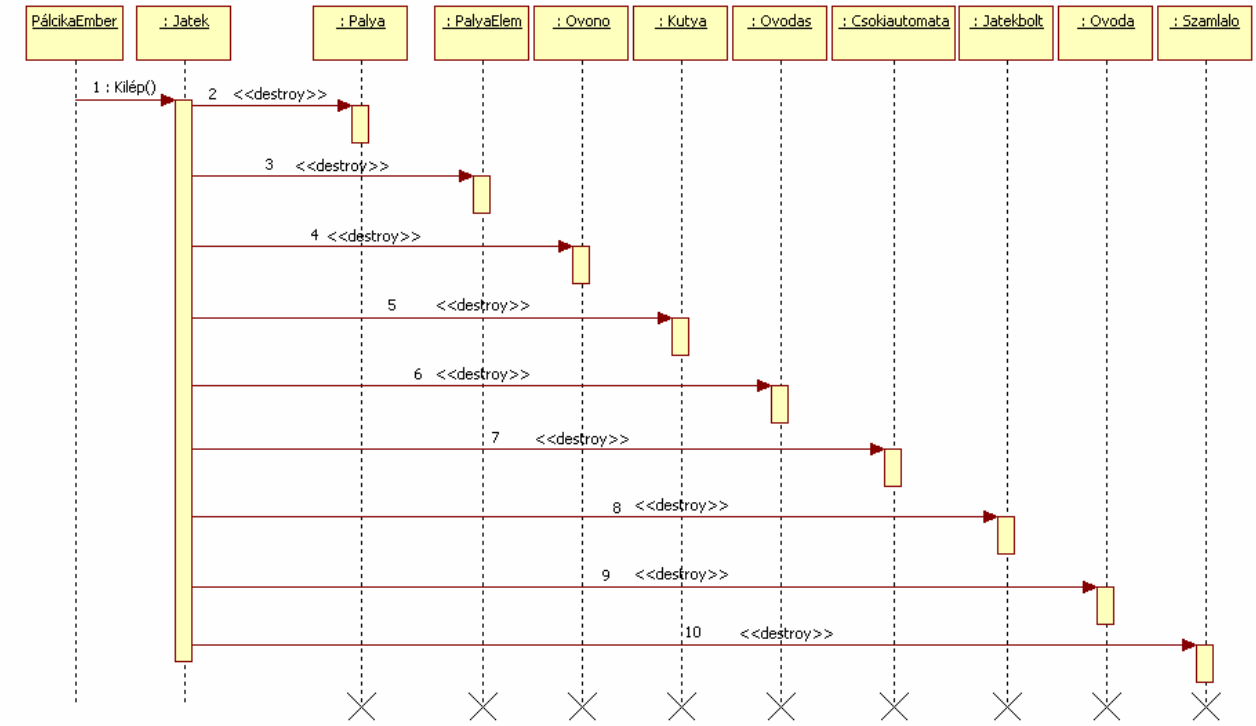
3.4.1 A játék menete



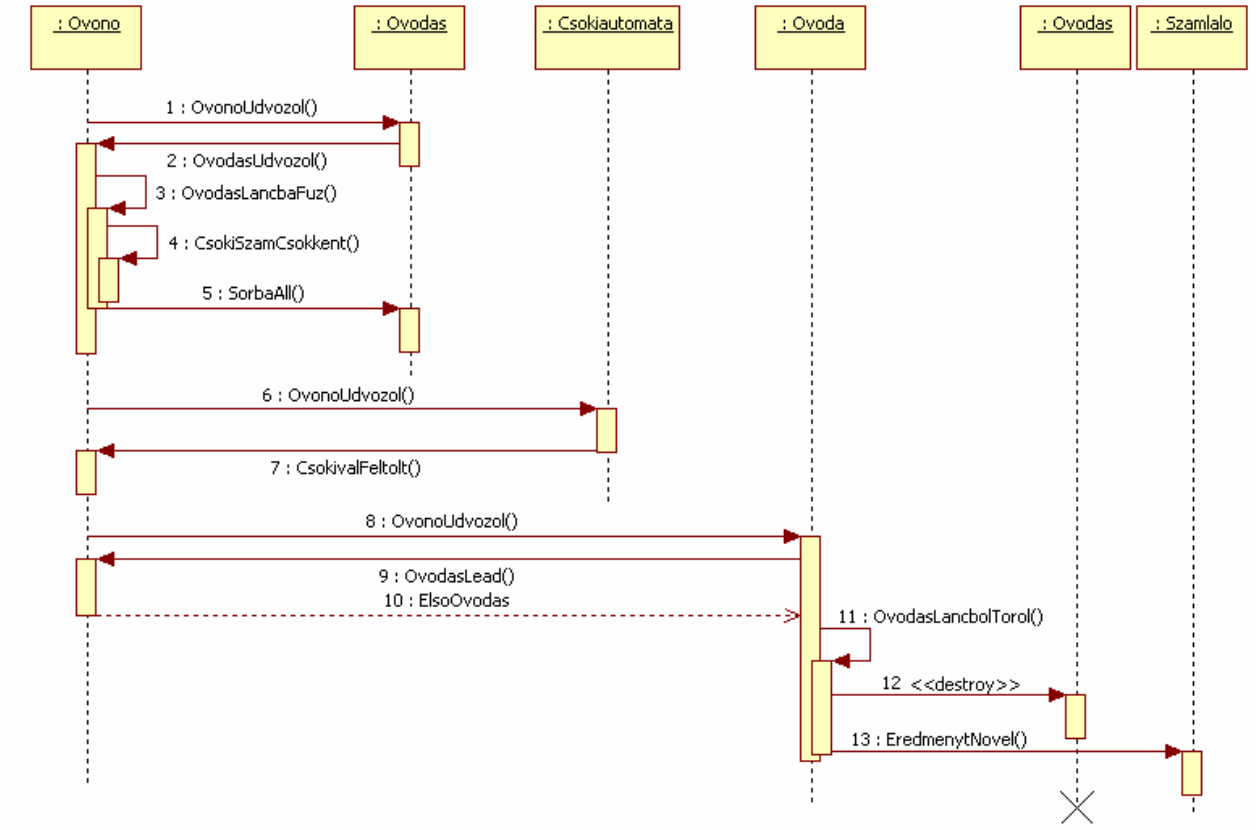
3.4.2 A Toplista megnézése



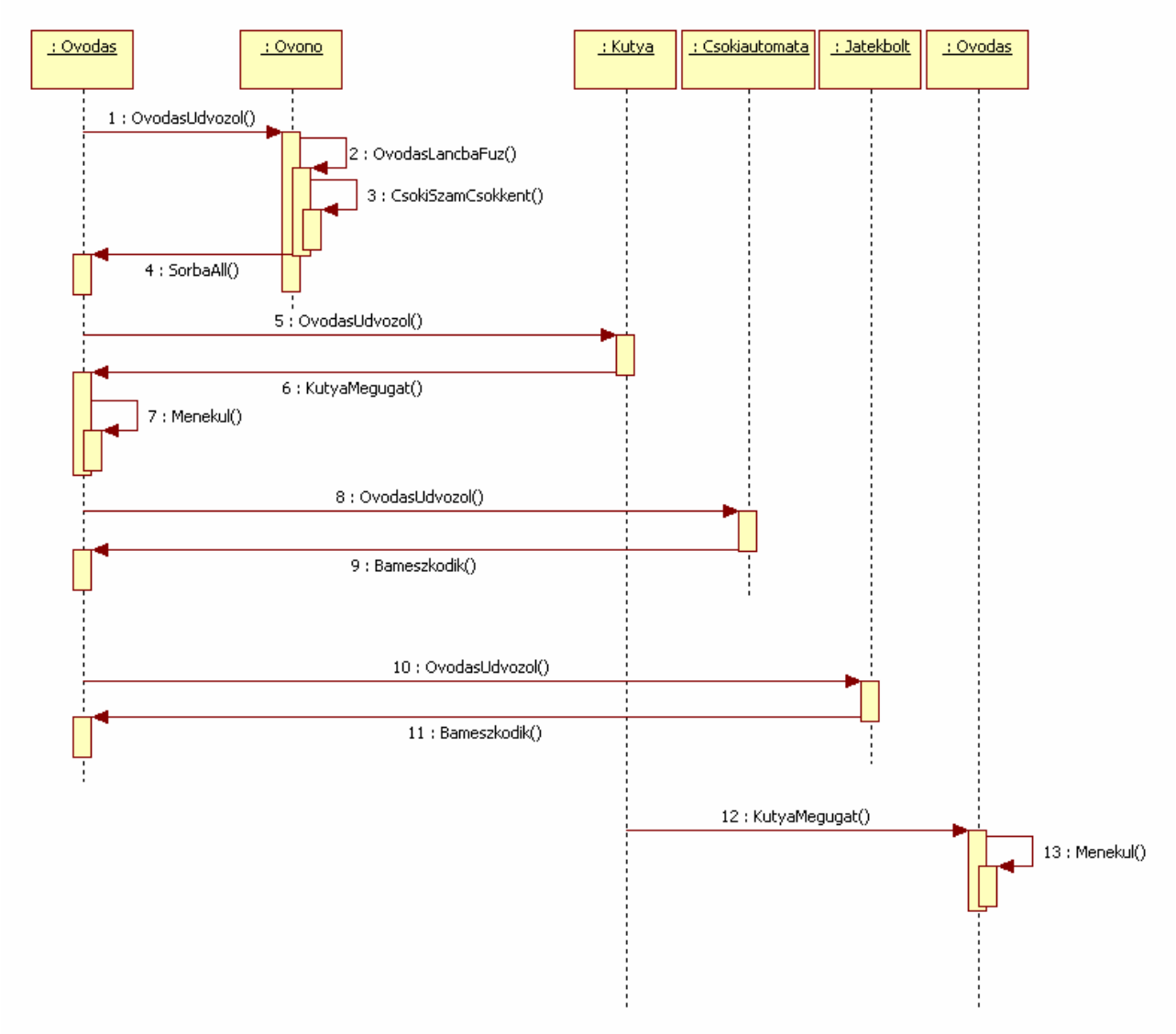
3.4.3 A kilépés menete



3.4.4 Az ovono akciói

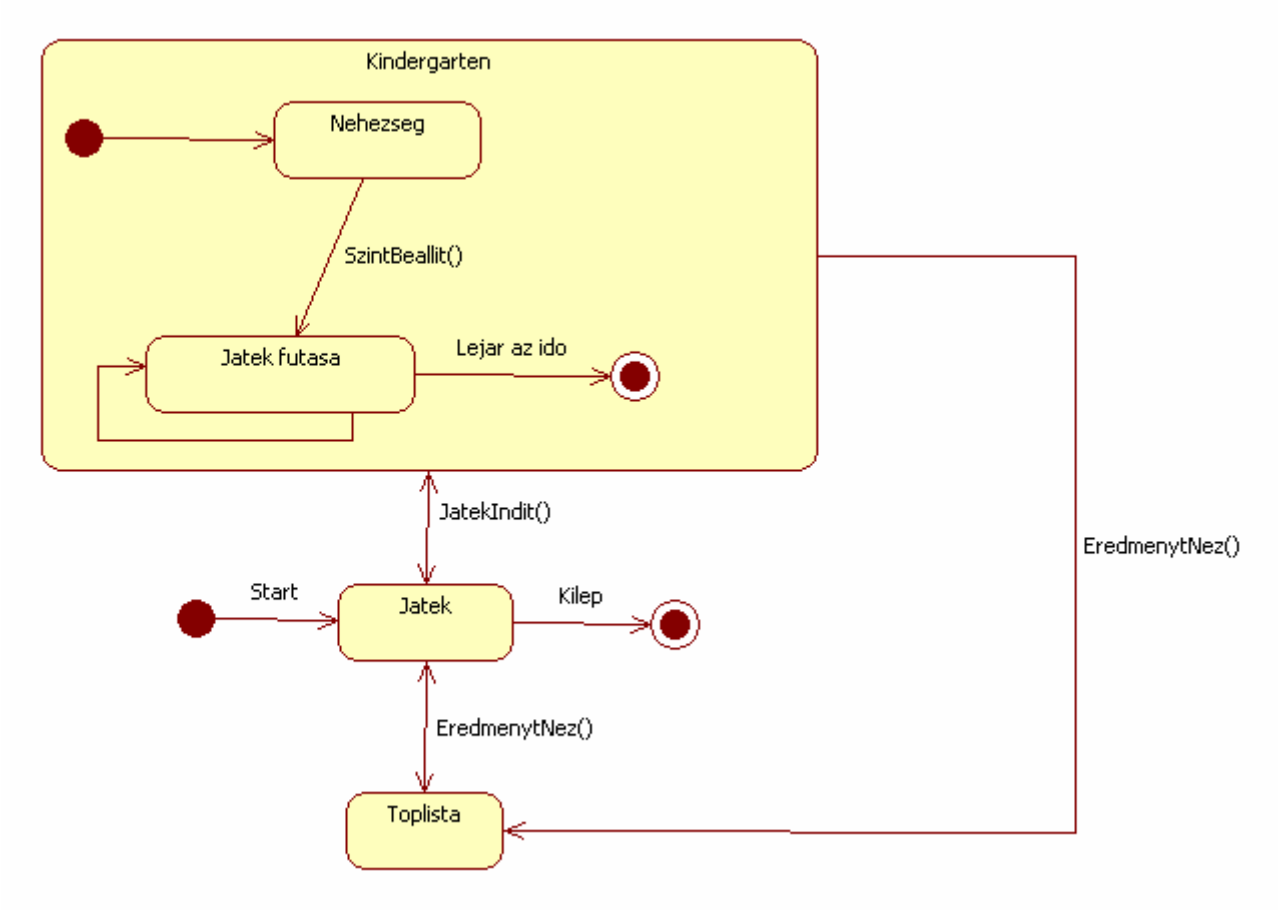


3.4.5 Az ovodas akciói

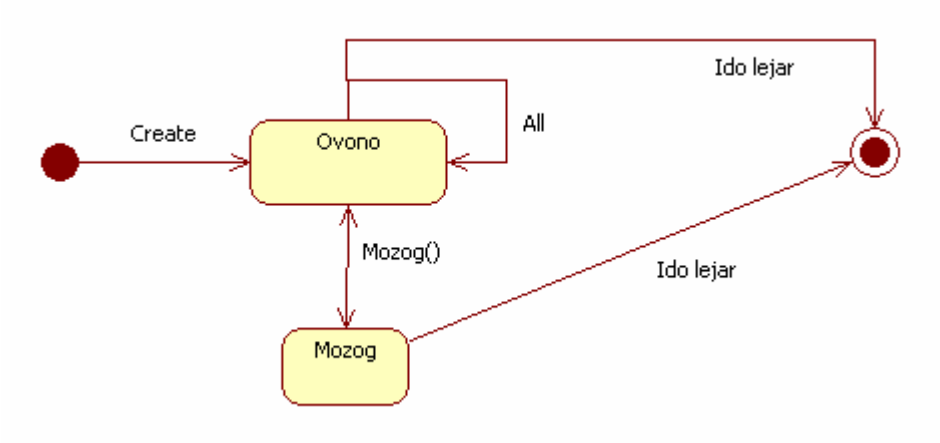


3.5 State-chartok

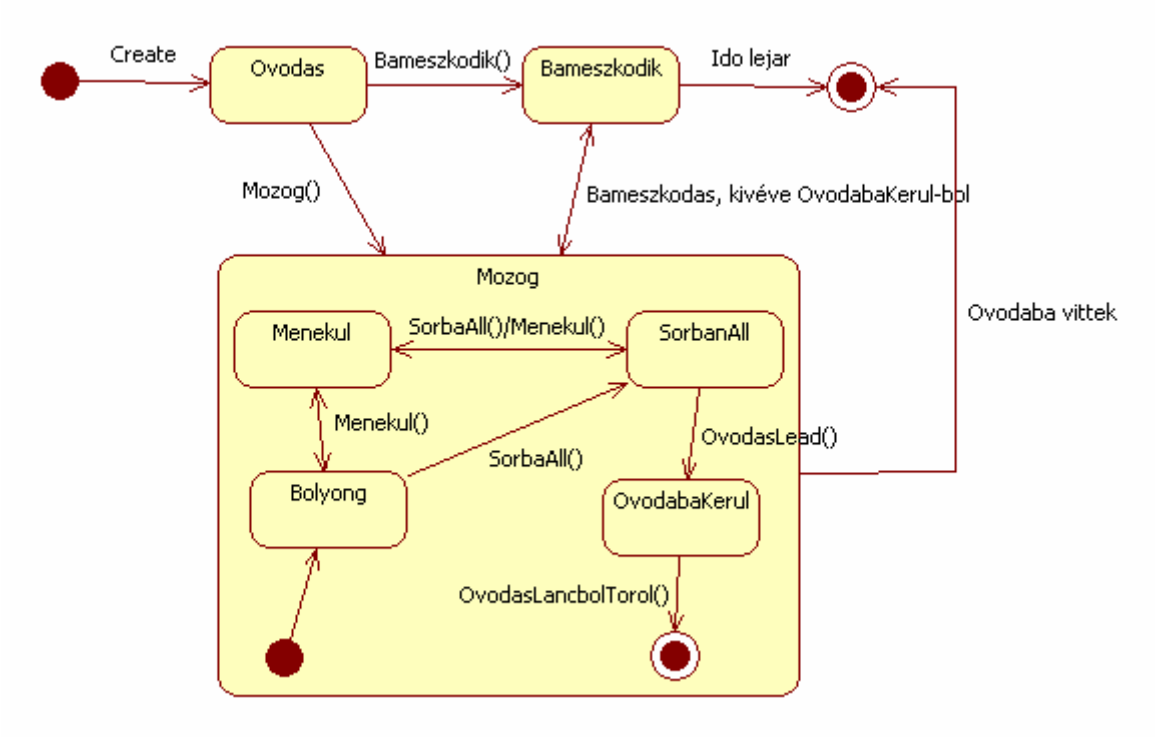
3.5.1 A játék menete



3.5.3 Az ovonő állapotai



3.5.4 Az ovodas állapotai



5. fejezet

Szkeleton tervezése

5.1. A szkeleton modell valóságos use-case-ei

Use Case Új Játék
Actor Játékos
Leírás Ez a use-case azt modellezi, hogy mi történik akkor, amikor új játékot kezdünk.

Use Case Mozgás
Actor Játékos
Leírás Ez a use-case azt modellezi, hogy mi történik akkor, amikor a billentyűzet segítségével mozgatjuk az óvónőt egyik pályaelemről a másikra.

Use Case Óvodás sorbaállítása
Actor Játékos
Leírás Ez a use-case azt modellezi, hogy mi történik akkor, amikor az óvónővel egy csokiért cserébe beállítjuk az óvodást a sorba.

Use Case Óvónő csokifelvétele
Actor Játékos
Leírás Ez a use-case azt modellezi, hogy mi történik akkor, amikor az óvónővel egy csokiautomatából csokikat veszünk fel.

Use Case Óvodást megugatja a kutya
Actor Játékos
Leírás Ez a use-case azt modellezi, hogy mi történik akkor, amikor az óvodás találkozik a kutyával, s ennek hatására a kutya megugatja.

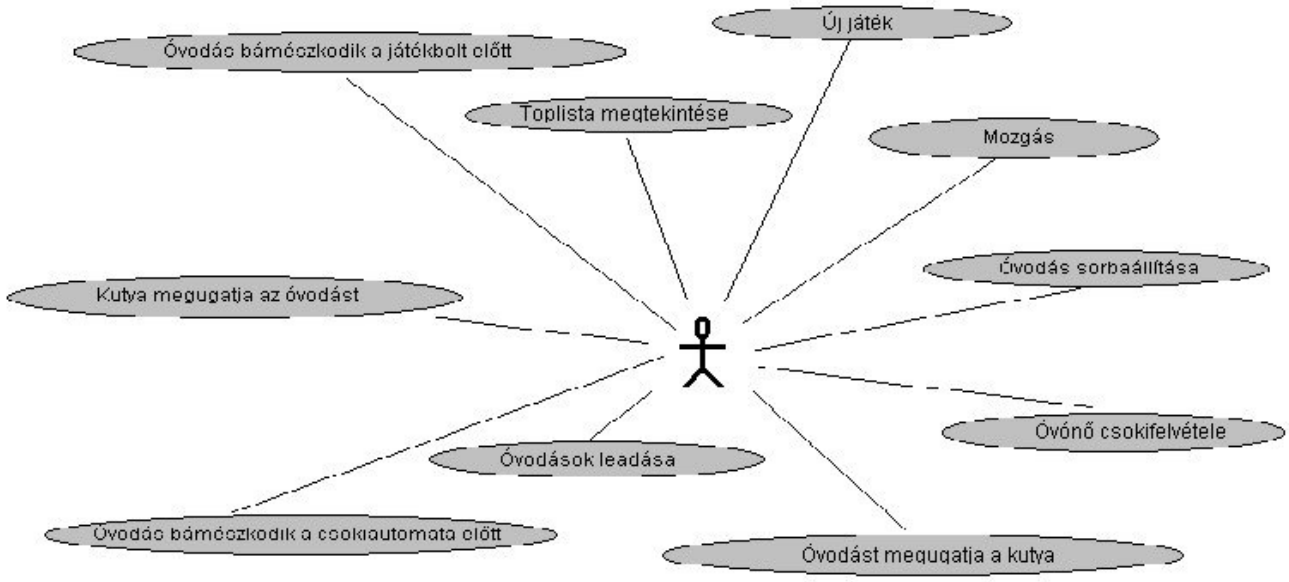
Use Case	Óvodások leadása
Actor	Játékos
Leírás	Ez a use-case azt modellezi, hogy mi történik akkor, amikor az óvónővel a sort az óvodába irányítjuk, s ezáltal leadjuk az óvodásokat.

Use Case	Toplista megtekintése
Actor	Játékos
Leírás	Ez a use-case azt modellezi, hogy mi történik akkor, amikor a menüből a toplista megnézését választjuk.

Use Case	Óvodás bámészkodik a játékbolt előtt
Actor	Játékos
Leírás	Ez a use-case azt modellezi, hogy mi történik akkor, amikor az óvodás a játékbolt előtt elkezd bámészkodni (s ezáltal ha sorban állt, akkor kiszakadni).

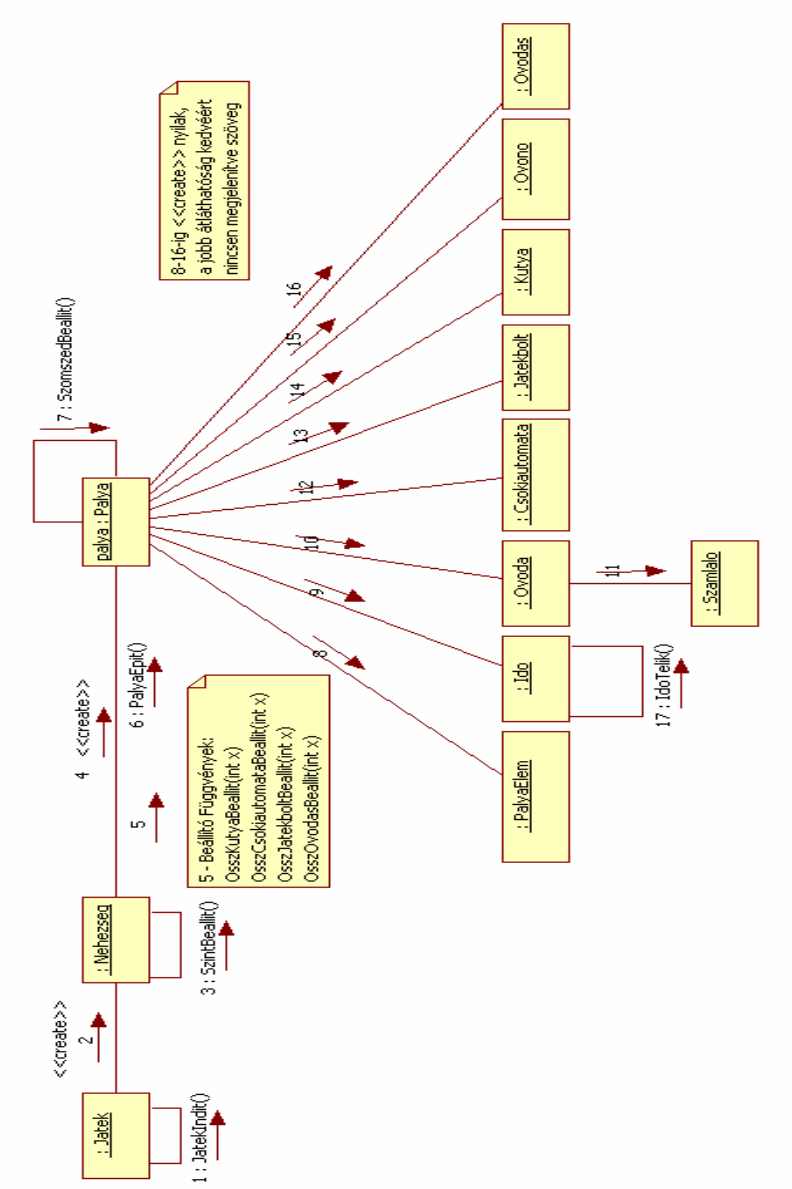
Use Case	Óvodás bámészkodik a csokiautomata előtt
Actor	Játékos
Leírás	Ez a use-case azt modellezi, hogy mi történik akkor, amikor az óvodás a csokiautomata előtt elkezd bámészkodni (s ezáltal ha sorban állt, akkor kiszakadni).

Use Case	Kutya megugatja az óvodást
Actor	Játékos
Leírás	Ez a use-case azt modellezi, hogy mi történik akkor, amikor a kutya találkozik az óvodással s megugatja.

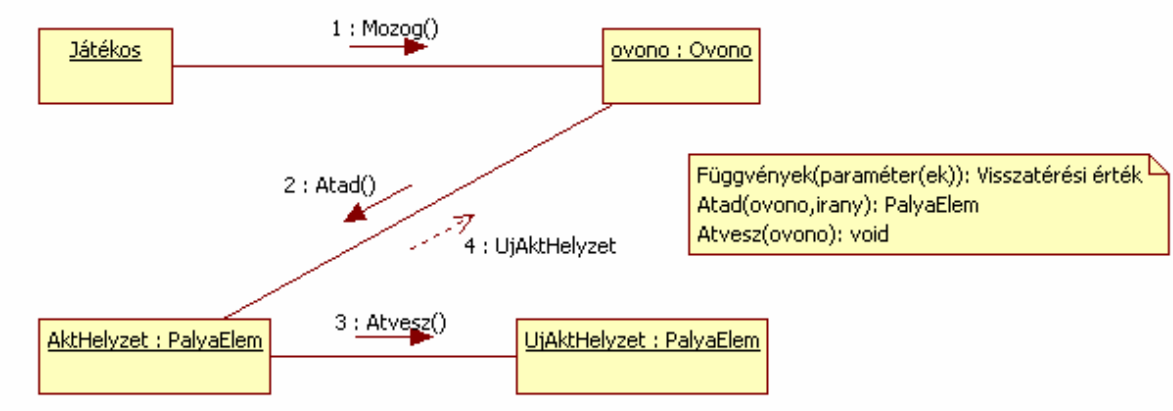


5.2. A kollaborációs diagrammok

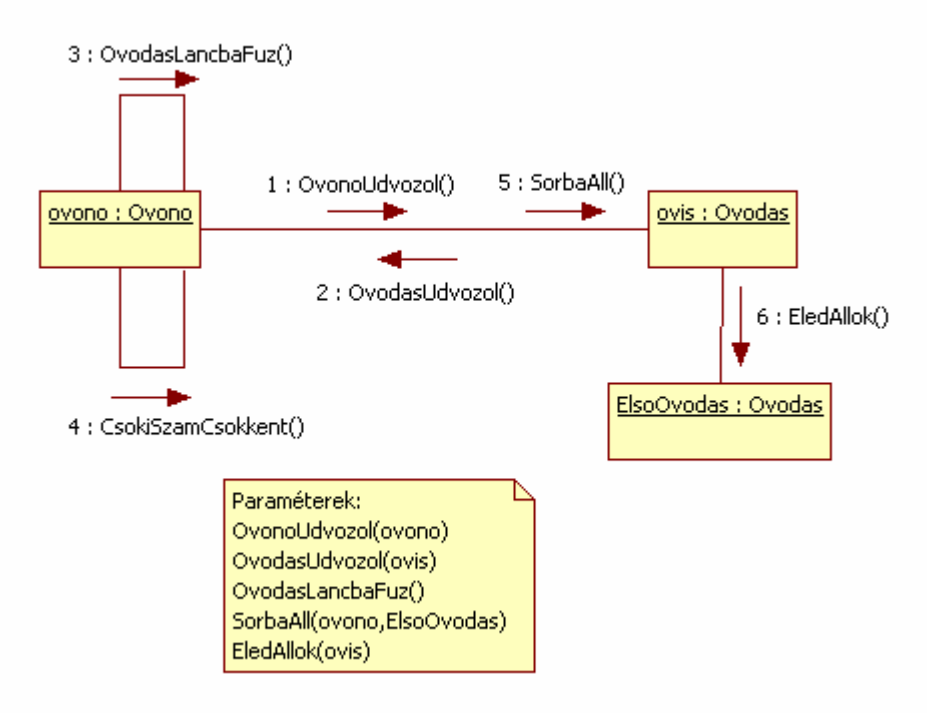
(Mellékeljük az új osztály diagrammot, ami alapján a kollaborációs diagrammok elkészültek.)



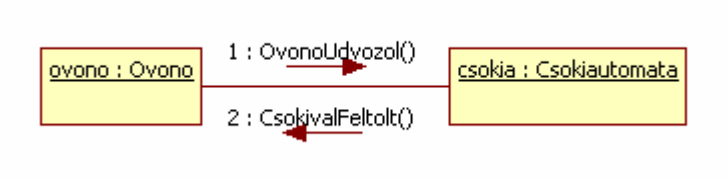
5.1 ábra kollaboráció diagramm az „Új Játék” use-case hoz



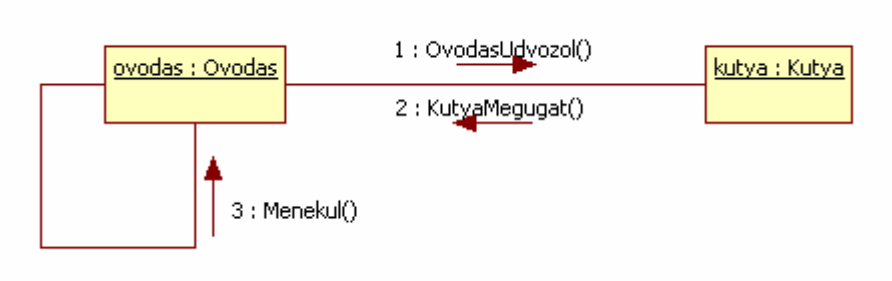
5.2 ábra Kollaborációs diagramm az „Mozgás” use-case hoz



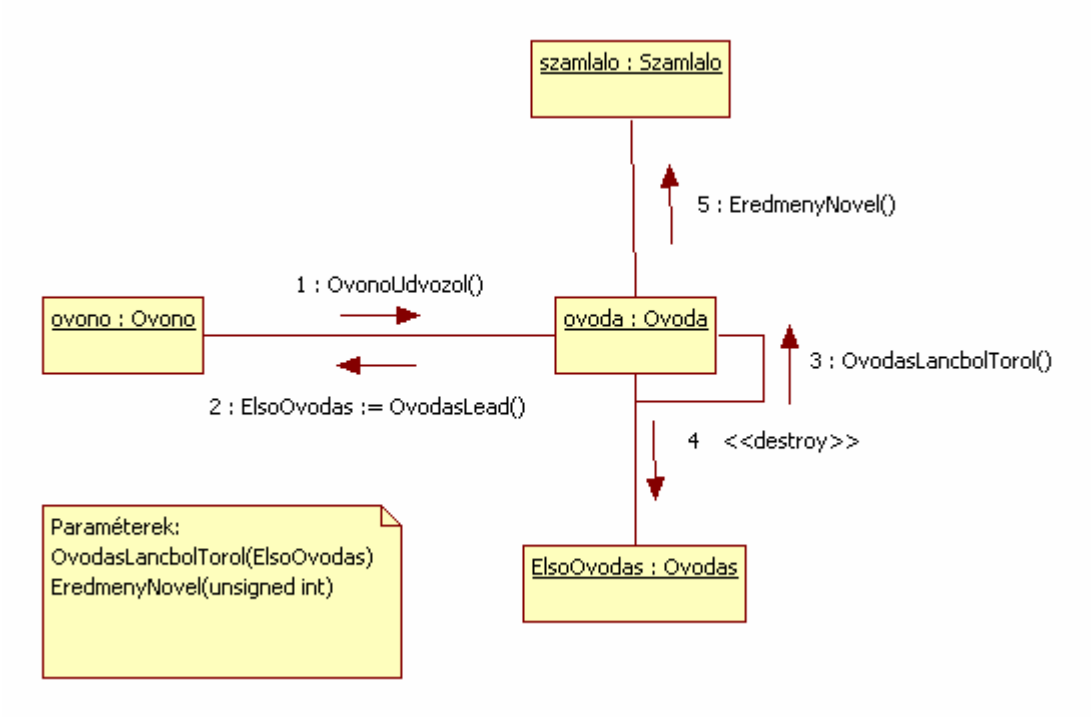
5.3 ábra Kollaborációs diagramm az „Óvodás sorbaállítása” use-case hoz



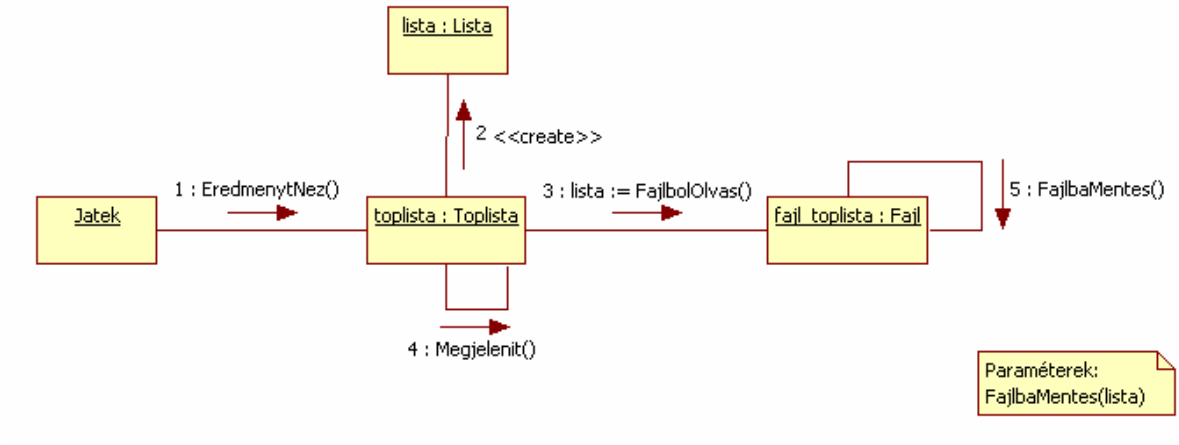
5.4 ábra Kollaborációs diagramm az „Óvónő csokifelvétele” use-case hoz



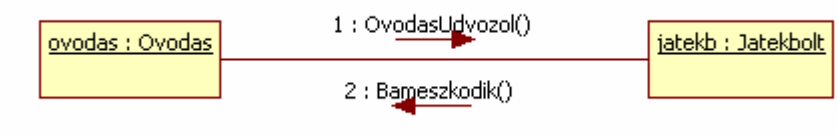
5.5 ábra Kollaborációs diagramm az „Óvodást megugatja a kutya” use-case hoz



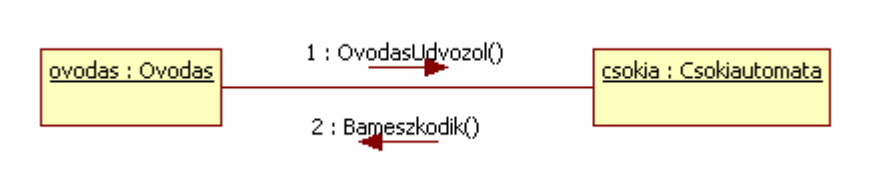
5.6 ábra Kollaborációs diagramm az „Óvodások leadása” use-case hoz



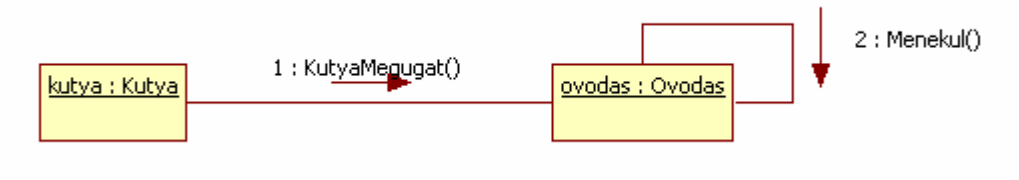
5.7 ábra Kollaborációs diagramm az „Toplista megtekintése” use-case hoz



5.8 ábra Kollaborációs diagramm az „Óvodás bámészkodik a játékbolt előtt” use-case hoz



5.9 ábra Kollaborációs diagramm az „Óvodás bámészkodik a csokiautomata előtt” use-case hoz



5.10 ábra Kollaborációs diagramm az „Kutya megugatja az óvodást” use-case hoz

5.3. A szkeleton kezelői felületének terve, dialógusok

A szkeleton az elkészítendő szoftver első futtatható változata.. A szkeleton modell segítségével ellenőrizhető, hogy a készülő program működése valóban megfelel-e a megrendelő által definiáltaknak. Hogy a működés követhető legyen a felhasználó számára is, célszerű valamilyen megjelenítési formát is kidolgozni. A szkeletonnál az egyik szempont, hogy a megjelenés magában a parancssorban történjen. Tartalmazza a program összes fontos objektumát és a szükséges interfészt. Meg kell említeni, hogy itt az objektumok még nincsenek teljesen kidolgozva. Ezen objektumok metódusainak a megvalósítása a szoftverfejlesztés egy későbbi fázisában kerül sor Mivel a metódusokból még hiányoznak a vezérlést megvalósító algoritmusok, a megfelelő működés érdekében erről nekünk kell gondoskodnunk. Az egyes metódusokban ezért csak a következő funkciókat valósítottuk meg: más metódusok meghívása, vezérlési vagy döntési helyzetben választási lehetőség biztosítása. A programnak, ha bemenetre vár, körül kell írnia a felhasználó számára, hogy pontosan milyen adatra van szüksége. Ennek megvalósítása szintén a parancssorban kell, hogy történjen.

A szkeleton tervezése során törekedtünk arra, hogy a modell működése nyomon követhető legyen, tehát azt kell megvalósítani, hogy minden függvény kiírja, hogyha meghívták, majd kiírja azt is, amikor visszatér. A függvénynek ez a két jelzése körülöleli az adott függvényben belül található többi függvényhívást, ebből következik, hogy függvényenként egy kijelzés nem elegendő. Illetve még egy olyan lehetőségek rakunk bele a szkeletonba, hogyha egy függvényen belül további függvényhívások is szükségesek, akkor azok egy szinttel beljebb

szerepelnek. Ha egy objektum metódusának végrehajtása során vezérlés vagy döntés szükséges, a program egy eldöntendő kérdés formájában kér információt, s a felhasználó válaszától függően folytatja a működést.

5.4. Architektúra, ütemezés

A projecthez tartozó összes Java forrásfájlt egy package-ben helyezzük el. Minden osztály külön fileban kap helyet, így a modellben a logikailag összetartozó definíciók egy implementációs egységbe kerülnek. Ezáltal az egyes modulok fejlesztése a későbbiekben egymástól függetlenül is történhet.
A program kihasználja a Java nyújtotta lehetőségeket, és többszálú futást valósít meg. Az objektumok nagy része – főleg azok, melyek aktív objektumok (saját maguk változtatják állapotukat) – saját szállal rendelkeznek. Ezzel elkerülhetjük, hogy egy közös külső időzítőre legyen szükségünk, emellett az is hatalmas előnyt jelent, hogy az egyes objektumok képesek önálló életet élni, ezzel megkönnyítve a tesztelést.
A többszálú programozás során ügyelni kell a párhuzamos működésből eredő hibák elkerülésére. A szálak használhatnak közös erőforrásokat, melyek módosításakor kölcsönös kizárást kell alkalmazni Igyekszünk kihasználni a Java által nyújtott alapvető szinkronizációs lehetőségeket. Mivel kis számban fordulnak elő szinkronizálást igénylő esetek, a beépített szolgáltatások elégségesek lesznek a hibás működés elkerülésére. A szálak legfőbb felhasználási területe ezek mellett az ütemezés lesz. A szkeletonban az időzítésnek nincs jelentősége, az ütemezés teljesen kézzel, a parancssorból történik.

6.1 A skeleton fordítása és futtatása

6.1.1 DOS parancssorból

1. A mellékelt zip file-t ki kell csomagolni egy könyvtárba.
2. A program fordítás és futtatása a run.bat nevű file segítségével történik.
3 A JavaDoc dokumentációt a doc.bat segítségével állíthatjuk elő. Ekkor létrejön egy Doc nevű könyvtár, amelyből az index.html oldalt webböngészőben megnyitva rendelkezésünkre áll egy jól használható objektumkatalógus. Az itt szereplő leírások a forráskódban is megtalálhatóak kommentek formájában.

6.1.2 DOS parancssor hiányában

1. A mellékelt zip file-t ki kell csomagolni egy könyvtárba.
2. A következő parancsokkal fordíthatjuk le a rendszert:
   javac -d Csontvaz Csontvaz/*.java
3. Futtatás:
   cd Csontvaz
   java Csontvaz/Jatek
4. A JavaDoc előállítása
   javadoc -d javadoc Csontvaz/*.java

6.2 A mellékelt állomány tartalma

A elküldött zip file tartalma az alábbi listában szereplő fileok.

./ tartalma:

A Csontvaz nevű mappa tartalmazza a skeletonhoz szükséges java állományokat, a javadoc nevű mappa pedig a már elkészített JavaDoc dokumentációt. A doc.bat a JavaDoc elkészítéséhez szükséges file, a run.bat segítsével tudjuk lefordítani és futtatni a fileokat.

2006.03.27. 00:26	<DIR>	Csontvaz
2006.03.27. 00:47	<DIR>	javadoc
2006.03.21. 17:13	34	doc.bat
2006.03.20. 10:36	134	run.bat
	2 fájl	168 bájt

./Csonvaz/ tartalma:

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

2006.03.26. 20:45	<DIR>	Csontvaz
2006.03.26. 23:08	1 827	Csokiautomata.java
2006.03.27. 00:23	779	Ido.java
2006.03.27. 00:33	6 326	Jatek.java
2006.03.26. 23:08	1 135	Jatekbolt.java
2006.03.27. 00:43	1 022	KozosOs.java
2006.03.27. 00:45	1 596	Kutya.java
2006.03.27. 00:23	2 470	Lista.java
2006.03.21. 17:11	285	Mozgat.java
2006.03.27. 00:23	2 993	Nehezseg.java
2006.03.27. 00:47	1 874	Ovoda.java
2006.03.27. 00:44	3 868	Ovodas.java
2006.03.27. 00:44	3 387	Ovono.java
2006.03.27. 00:33	3 610	Palya.java
2006.03.27. 00:23	3 317	PalyaElem.java
2006.03.27. 00:32	1 454	Szamlalo.java
2006.03.27. 00:19	5 141	Szkeleton.java
2006.03.27. 00:46	1 563	Toplista.java
	17 fájl	42 647 bájt

6.3 A skeleton használata

A program a terveknek megfelelően karakteres felületet használ. Egy egyszerű menü segítségével lehet navigálni a programban: kiválasztani, hogy melyik alapvető működést akarjuk megvizsgálni, illetve ha befejeztük a program használatát, a kilépés is itt történhet. Miután kiválasztottuk az egyik menüpontot, egy rövid leírást kapunk, melyben szerepel, hogy mely objektummal, illetve objektumokkal történik az interakció. Rögtön ezután szekvenciadiagram-szerűen jelenik meg, hogy mely objektum melyik másik objektum metódusát hívta meg.

Pelda:

Use-Case: Ovodast megugatja a Kutya Az Ovodas olyan PalyaElemre lep, ahol talalhato Kutya; ekkor udvozli a Kutyat, mire az megugatja, es az Ovodas Menekul()-ni kezd.

```
Kutya::Konstruktor es VEGE
Ovodas::Konstruktor es VEGE
Kutya::OvodasUdvozol(Ovodas ovis)
  Ovodas::KutyaMegugat()
    Ovodas::Menekul()
    Ovodas::Menekul() VEGE
  Ovodas::KutyaMegugat() VEGE
Kutya::OvodasUdvozol(Ovodas ovis) VEGE
```

A program a behúzásokkal jelzi, hogy a függvényhívások mely szintjén állunk. A szkeleton nem ismeri a modell algoritmusait, ezért ha olyan metódust hívunk meg, melyben dönteni kell, akkor a program választási lehetőséget kínál fel. Ez minden esetben egy eldöntendő kérdés, melyre a felhasználónak kell megadnia a választ. A választól függően folytatódik tovább a működés.

6.4 Értékelés

A csapat munkája néhány hét után összehangolttá vált. Arra törekedtünk, hogy a feladatok megfelelő szétosztásával mindenki egyenlő arányban vegyen részt a féléves feladat megoldásában, valamint mindenki olyan feladatot kapjon, ami képességeinek és felkészültségének a legjobban megfelel. Az egyenlő elosztás nem mindig teljesült, de ezt igyekeztünk kompenzálni. Eddig az időbeosztás rendben alakult, mindenki időben elkészült a feladatával. A csapat jól tud együtt dolgozni. A százalékos értékelést közösen beszéltük meg. Bár mindenki más-más típusú feladatot kapott, úgy gondoljuk, a csapat minden tagja egyenlő mértékben vette ki részét a munkából, ezért a pontok egyenlő elosztása mellett döntöttünk.

A feladat megoldásához szükséges fejlesztőkörnyezetet hamar kiválasztottuk, döntésünk egyhangú volt. Hasonlóan egyhangú döntés született az UML diagramok elkészítéséhez használt program esetén is.

7. Prototípus koncepciója

7.1. Prototípus interface definíciója

7.1.1 A prototípus célja

A prototípus célja a program helyes működésének tesztelése. A tesztelés hangsúlya főleg a modellre helyeződik, de az ütemezés is fontos szerepet kap már. A program ezen stádiumában a be- és kimenetet, az ütemezést és a felhasználói felületet leszámítva már minden úgy kell működjön, mint ahogy majd a végleges változatban fog.

7.1.2. Felület megvalósítása

A prototípus az előírásoknak megfelelően karakteres üzemmódot használ a megjelenítésre. Ebből kifolyólag játékra tulajdonképpen alkalmatlan, nem élvezhető program áll elő. Egyszerű kezelhetősége és megjelenése mögött tartalmas üzenetek és eseménynaplózás valósul meg, így a tesztelési esetek megvizsgálására tökéletes lesz. A program normál működésben a billentyűzet segítségével lesz kezelhető.

7.1.3. Ütemezésbeli sajátosságok

A prototípus ütemezése eltér még a végleges változatétól. Lehetőséget biztosítunk ugyanis ütemenkénti végrehajtásra. A mi modellünk több szálból áll, minden objektum önálló életet tud élni, ezért egy globális "engedélyező" kerül beépítésre a programba. Ennek működését talán a digitális technikában alkalmazott órajelhez lehetne hasonlítani.

7.1.4. Be és kimenet

A program a futás során minden ütem végén lehetőséget nyújt felhasználói beavatkozásra, melyek a standard bemenetről érkezhetnek. Az aktuális parancsokról a felhasználó mindig tájékoztatást kap a programtól, melyek többnyire egy-két karakter hosszúak. A parancsokat begépelve és az ENTER billentyűt leütve a program értelmezi őket. Egymás után több

parancs is kiadható, hiszen a bevitt utasításokat a program egy pufferban tárolja, amelyet folyamatosan dolgoz fel. A következő utasításokat tudja értelmezni a program:

parancs: n - Új játék
parancs: r - Szint újrajátszása a játékidő letelése után
parancs: t - Toplista megtekintése
parancs: b - Óvónő irányítása balra
parancs: j - Óvónő irányítása jobbra
parancs: f - Óvónő irányítása fel
parancs: l - Óvónő irányítása le
parancs: k - Kilépés a játékból, vagy a menübe
parancs: c - Csokifelvétel
parancs: o - Óvodások leadása
parancs: s - Óvodás sorbaállítása
parancs: g - adott ideig engedélyezi a szálak működését, így az egyes objektumok állapota két 'g' utasítás között fog megváltozni.

A program kimenete elsősorban a standard kimenet. Ide írja üzeneteit és rajzolja a pálya vázlatos képét. Emellett a modell részletes üzenetei naplózásra kerülnek a háttérben egy fájlba. Ezzel megkönnyítjük a nyomkövetést, és szétválasztjuk a két felületet egymástól. A képernyőn a pálya képe és az esetleges kérdések, választási lehetőségek olvashatók, míg a modell működéséről a naplófájlból kapunk pontos képet.
A naplófájlban minden sor egy eseményt jelöl. Zárójelek között feltüntetjük annak az objektumnak a nevét és típusát, melynek egy tagfüggvénye meghívódott. Ezután az eseményről kapunk egy rövid leírást, melyben a többi résztvevő objektum is szerepel.
A standard bemenet és kimenet használatának előnye ott is megmutatkozik, hogy tesztelési célból tudunk "külső" vezérlőfájlokat készíteni, amit a program feldolgoz és felhasználói beavatkozás nélkül fut. A standard be és kimenet a megszokott módon, DOS alatt a "<" és ">" jelekkel, Unix alatt pedig a pipe alkalmazásával irányítható át.
Például egy bemeneti utasítássorozatot tartalmazó fájl: teszteset3.txt
Ennek alkalmazása pl: java ProtOvoda.Jatek < teszteset3.txt DOS alatt.

7.1.5. Proto fájlformátumok

7.1.5.1. Bemeneti fájlformátum

Az első sor a játékos nevét tartalmazza, a második sor pedig az elvégzendő parancsokat (a fentebb - "Be és kimenet" részben - megadottak szerint). Pl.:

Jánoshidai Jenő
n0ggggggbgfgfglgk

7.1.5.2. Kimeneti fájlformátum

<EsemámylHiba> - <ütem> > <objektum::történés>, történés-t az objektum egy függvénye fogja reprezentálni, tehát amikor meghívódik, illetve futás közben a fontosabb változásoknál üzenetet ír ki, pl.

Esemény - 0. ütem > Ovodas::Sorbaálltam
Hiba - 1. ütem > Ovoda::Nincs leadott óvodás

7.2 Összes részletes use-case

7.2.1 Use-case leírások

Use case:	Új játék
Actor:	Játékos
Leírás:	A játékos új játékot kezd az 'n' betű leütésével. A rendszer feladata az alap-helyzet beállítása: bekéri a játékos nevét, a nehézségi szintet, majd megjeleníti a pályát. A játékos feladata: megadnia a nevét, kiválasztani a nehézséget, majd a végtelenségig játszani.

Use case:	Szint újrajátszása
Actor:	Játékos

Leírás: A játékos az aktuálisan játszott nehézségi szintet újra kezdheti az 'r' betű lenyomásával, miután lejárt az idő. A rendszer ekkor ismételten betölti az aktuális nehézségi szintet, és módosítja a játékos pontszámát: visszaállítja nullára. Miután a rendszer betöltötte a pályát, a játékos újra elkezdhet játszani.

Use case: Toplista megtekintése

Actor: Játékos

Leírás: A játék kezdetén a menüből, vagy a játék befejeztével a játékos megtekintheti az addigi legjobb eredményeket az 't' betű lenyomásával. A rendszer ekkor egy külső fájlban tárolt lista alapján közli az eddigi legjobb eredményeket és játékosokat. A játékosnak a kezdeményezés után nyugtáznia kell a látottakat.

Use case: Óvónő irányítása

Actor: Játékos

Leírás: A játékos irányítja az óvónőt, ezzel befolyásolhatja (ha van mögötte óvodás sor) az óvodások haladási irányát. Ehhez a négy nyilat használhatja és 6 irányba léphet. A rendszer gondoskodik arról, hogy az Óvónő a megfelelő irányba mozduljon, és az esetlegesen utána lévő sor is kövesse.

Use case: Kilépés

Actor: Játékos

Leírás: A játékos befejezheti a játékot bármikor a 'k' betű lenyomásával. A rendszer megvizsgálja, hogy a végeredmény alapján a játékos bekerülhet-e a legjobbak közé a toplistába. Ha bekerül, akkor módosítania kell az eredményeket tároló listát, majd lementenie egy megadott fájlba.

Use-case: Csokifelvétel

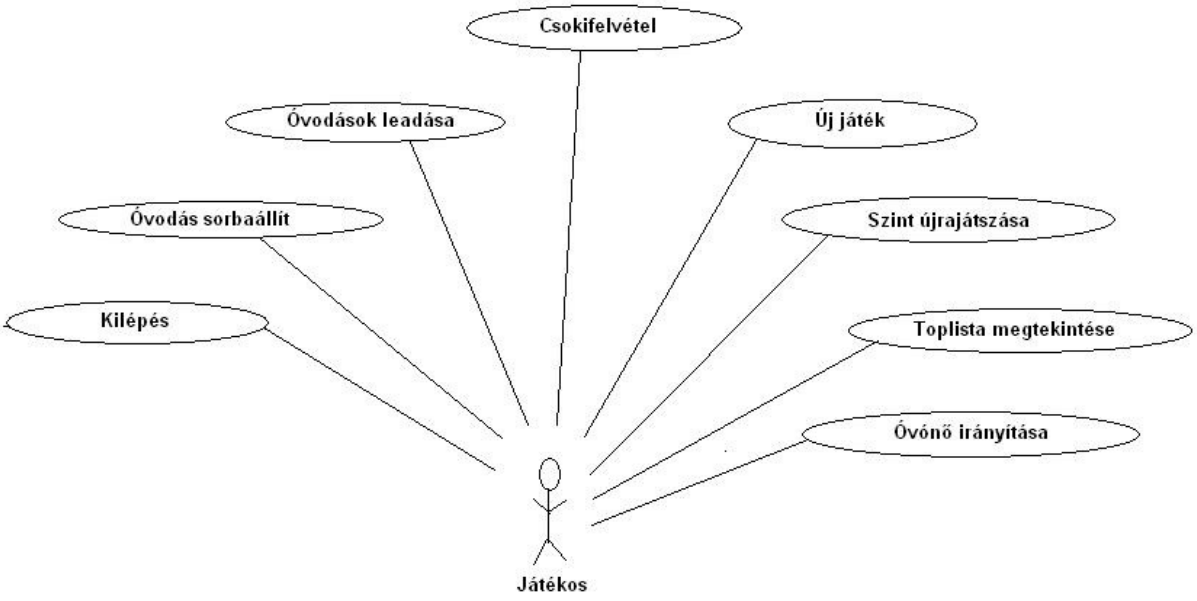
Actor: Játékos

Leírás: A játékos csokit vehet fel az óvónővel. Ehhez az óvónőt az egyik csokiautomatához kell irányítania (az óvónő irányítása c. use-case-ben leírtak szerint), majd pedig a 'c' gombot lenyomni. Ezután az óvónőnél levő csokik száma a maximális lesz.

Use-case:	Óvodások leadása
Actor:	Játékos
Leírás:	Az óvónővel "leadhatjuk" az őt követő óvodásokat úgy, hogy az óvónőt az óvodához irányítjuk (az óvónő irányítása c. use-case-ben leírtak szerint). Majd ha megérkeztünk, akkor a 'o' lenyomásának a hatására a rendszer törli az óvónő mögött álló óvodások referenciáit és a bevitt óvodások számával arányosan növeli a számláló értékét, mely majd végül az eredményt szolgáltatja.

Use-case:	Óvodás sorbaállítása
Actor:	Játékos
Leírás:	Az óvónővel és a nála levő csokik segítségével állíthatjuk sorba az óvodásokat a következő módon: miután az óvónőt az egyik "kiszemelt" óvodáshoz irányítottuk, akkor (ha van nálunk csoki) az 's' betű lenyomásának hatására beállíthatjuk az óvodást az óvónő mögötti sorba, s ezzel egy időben csökken az óvónőnél levő csokik száma, melyeket csokifelvétellel pótolhatunk. Amire ügyelnünk kell, hogy csak fiú, vagy lány sorunk lehet, ugyanis az óvodások nem fogják meg a másik nem kezét. Ha mégis sorbaállítanánk egy ellenkező neműt, akkor az aktuális sorunk felbomlik.

7.2.2 Use-case diagram



7.3 Tesztelési terv

A tesztelés során törekedni kell a program széleskörű vizsgálatára. A teszt során különböző tesztesetek vizsgálatát hajtjuk majd végre az egyes funkciók ellenőrzése végett. A megvalósításunkban lehetőség nyílik az önálló tesztesetek egyenkénti kipróbálására. A bemeneti állományokkal vezérelhető a program a standard bemeneten keresztül, az eredmény pedig a standard kimeneten és a napló fájlban szerepel. A naplófájlba írt üzenetek megvizsgálásával pontosan nyomon követhető az objektumok működése.

A Tesztesetek és rövid leírásuk:

Use-case: Új játék kezdése

Leírás: Új játék indítása a játék elkezdésekor, vagy pályáról való kilépés után. Figyelni kell arra, hogy a pálya betöltődött-e az elvártaknak megfelelően.

Use-case: Kutya megugatja az óvodást

Leírás: Ha egy kutya találkozik, egy óvodással, akár fiú akár lány megugatja. Ha ekkor az óvodások sorban voltak, akkor kiszakadnak.

Use-case: Óvodás meglátja a Kutyát
Leírás: Az óvodás találkozik a kutyával, aki szintén észreveszi és megugatja, mire az óvodás menekülni kezd. Ha a találkozáskor az óvodás sorban állt, akkor kiszakad belőle.

Use-case: Óvodás(lány) bámészkodik a csokiautomata előtt
Leírás: Ha egy óvodás(lány) találkozik egy csokiautomatával, akkor ennek hatására kiszakadhat a sorból. Ellenben a fiú óvodásra nincs hatással.

Use-case: Óvodás(fiú) bámészkodik a játékbolt előtt
Leírás: Ha egy fiú óvodás találkozik egy játékbolt, akkor ennek hatására ha sorban állt, akkor kiszakadhat, a lányokra semmi hatással nincs.

Use-Case: Csokifelvétel
Leírás: Az óvónő az óvodások a sorba való befűzéshez csokira van szüksége. Mivel az óvonőnél, csak 5 csoki lehet, ezért az utánpótlásért a csokiautomatához mehet.

Use-Case: Óvodás sorbaállítása (egynemű)
Leírás: Egy óvodás csak akkor áll sorba, ha a sorban csak vele megegyező nemű óvodások vannak.

Use-Case: Óvodás sorbaállítása (fiú-lány)
Leírás: Egy óvodás nem akar beállni a sorba, ha ellenkező neműekből áll, s társai se akarnak vele egy sorban lenni, ezért ilyenkor az aktuális sor felbomlik.

Use-Case: Óvodások leadása
Leírás: Az óvónő itt tudja leadni az óvodásokat, miután sorbaállította őket.

Use-Case: Számláló növelése

Leírás: Mikor az óvónő bevitte az óvodásokat az óvodába, a számláló növekedni fog annak függvényében, hogy hány óvodást tudott leadni az óvónő. A játék végén a Toplista lekérdezi az értékét, s eszerint kerül majd be a játékos eredménye a toplistába.

Use-Case: Toplista megtekintése a menüből

Leírás: A menüből kiválasztva megtekinthetjük a toplistát, mely a legjobb eredményeket tárolja, amelyeket egy fájlból olvas be.

Use-Case: Toplista megtekintése a játék végén

Leírás: A játék idő leteltével megtekinthetjük a toplistát, mely a legjobb eredményeket tárolja egy fájlban, illetve a játék folyamán egy listában.

Use-Case: Szint újrajátszása

Leírás: Amikor a játékidő végére érünk, választhatunk, hogy a toplista megtekintése után kilépünk, vagy újra játszunk ugyanazon a nehézségi szinten.

Use-Case: Idő lejár

Leírás: Amikor az idő lejár, akkor véget ér a játék és a rendszernek le kell ellenőriznie a számláló értéke alapján, hogy felkerülünk-e a toplistára vagy sem.

7.4 Változtatások a specifikáció módosulása miatt

- Ezentúl nem négy, hanem hat irányban kell tudjunk mozogni az óvónővel és a pályaelemek is ebből kifolyólag hat oldalúak lesznek. Ez az ún. méhkaptáros elrendezés. Egyértelmű tehát, hogy egy pályaelem immár hat referenciát fog tartalmazni.

- Az óvodások objektumot két objektummal valósítjuk meg, amelyet az óvodások objektumból származtatok, a fiúóvodás és a lányóvodás objektumokkal. Mivel másképp reagálnak a külső hatásokra, ezért például a fiú objektumnak lesz

játékboltüdvözöl függvénye, de csokiautomataüdvözöl függvénye nem, míg a lány objektumnál fordítva. Ezzel biztosítjuk, hogy a fiúk nem bámészkodnak a csokiautomata hatására és a lányoknál úgyszintén. Ami probléma még a nemi különbségből fakad az az, hogy a lány objektum nem hajlandó átvenni egy fiú objektum referenciáját s fordítva úgyszintén, azaz a lányok nem állnak be fiú sorba, s a fiúk se a lány sorba.

8.2 Objektumok és metódusok tervei

8.2.1 Csokiautomata

Ős:

-

Attribútumok:

static int csokiautomatakszama

Tagfüggvények:

public Csokiautomata()

A konstruktor, egyelore semmi szerepe.

public void AktHelyzetBeallit(PalyaElem akt)

Az AktHelyzet változót hivatott beállítani.

public void OvonoUdvozol(Ovono ovono)

Amikor az Ovono olyan PalyaElemre lep, ahol talalhato Csokiautomata, akkor ez a függvény hívódik meg, és feltölti az Ovonot csokival, annak CsokiFeltolt függvényének meghívásával.

Paraméterek:

ovono - az Ovonore mutató referencia, amivel elérhetjük az Ovono CsokivalFeltolt függvényét

public void OvodasUdvozol(Ovodas ovis)

Amikor az Ovodas olyan PalyaElemre lep, ahol talalhato Csokiautomata, akkor ez a függvény hívódik meg, és elbámészkodtatja az Ovodast, annak Bameszkodik függvényének meghívásával.

Paraméterek:

ovis - az Ovodasra mutató referencia, amivel elérhetjük az Ovodas Bameszkodik függvényét.

8.2.2 FiuOvodas

Ős:

KozosOs

Ovodas

Attribútumok:

-

Tagfüggvények:

public FiuOvodas()

Konstruktor, az ososztállyal tudatja, hogy Fiú.

public void JatekboltUdvozol()

A játékbolt elbámékoztatja a fiúkat.

8.2.3 Ido

Ös:

-

Attribútumok:

int ido

Proto proto

Tagfüggvények:

public Ido()

A konstruktor, ami beállítja az ido-t 3percre, és meghívja az IdoTelik függvényt.

public void IdoTelik()

Az ido múlását ezzel a függvénnyel biztosítjuk, másodpercenként fogja csökkenteni az ido változót, amikor nullához ér majd valahogy jelez, hogy letelt az ido.

public boolean IdoVege()

Az idot lehet lekérdezni, vagyis, hogy letelt-e már vagy sem.

8.2.4 Jatek

Attribútumok:

public Ido ido
public Proto proto
public Toplista toplista

Tagfüggvények:

public Jatek()

A konstruktor, amely vezérli a Szkeleton projekt muködését, a Szkeleton menüje után, a választott use-case-t valósítja meg.

public void EredmenytNez()

A Toplistat érhetjük el vele.

public void JatekIndit()

A játék indítását végzi, jelen állapotában egy Nehezseg objektumot hoz létre, ez lehetséges, hogy késobb is így marad.

public void Kilep()

Egyszeruen és nagyszeruen kiléphetünk vele a programból; tartozni fog hozzá egy grafikus nyomógomb, ami meg fogja hívni.

public static void main(java.lang.String[] args)

Az egész programot indító main függvény, amely létrehoz saját magából egy példányt.

Paraméterek:

args - a parancssori argumentumokat tartalmazza

8.2.5 Jatekbolt

Ős:

KozosOs

Attribútumok:

static int jatekboltok

int sid

Tagfüggvények:

public Jatekbolt()

A konstruktor, ami nem csinál semmit, valószínuleg nem is fog, de még lehet hogy kelleni fog.

public void AktHelyzetBeallit(PalyaElem akt)

Az AktHelyzet változót hivatott beállítani.

public void OvodasUdvozol(Ovodas ovis)

Az Ovodas ezt a függvényt hívja meg ha a két objektum egy azon PalyaElemen van, ekkor a függvény meghívja az Ovodas Bameszkodik függvényét.

Paraméterek:

ovis - a Ovodas referenciája, aki idetévedt

8.2.6 KozosOs

Attribútumok:

PalyaElem AktHelyzet
Proto proto

Tagfüggvények:

public KozosOs()
A konstruktor, ami jelenleg egy új Szkeleton példányt hoz létre, ezzel biztosítva a kommunikációt program és felhasználó között.

public void AktHelyzetBeallit(PalyaElem akt)
A leszármazott osztályok valósítják meg, az AktHelyzet változót hivatott beállítani.

public void CsokiautomataUdvozol()
Azok a leszármazott osztályok valósítják meg, amelyeknek szükséges.

public void JatekboltUdvozol()
Azok a leszármazott osztályok valósítják meg, amelyeknek szükséges.

public void KutyaMegugat()
Azok a leszármazott osztályok valósítják meg, amelyeknek szükséges.

public void OvodasUdvozol(Ovodas ovis)
Azok a leszármazott osztályok valósítják meg, amelyeknek szükséges.

public void OvonoUdvozol(Ovono ovono)
Azok a leszármazott osztályok valósítják meg, amelyeknek szükséges.

8.2.7 Kutya

Ős:

KozosOs

Attribútumok:

static int kutyak

int sid

Tagfüggvények:

public Kutya()

A konstruktor, egyelore semmit nem csinál, de még lehet hogy fog.

public void AktHelyzetBeallit(PalyaElem akt)

Az AktHelyzet változót hivatott beállítani.

public void Mozog()

A Mozgat interfész Mozog() függvényének megvalósítása.

public void OvodasUdvozol(Ovodas ovis)

Ha Ovodas érkezik ugyanoda ahol a Kutya is van, akkor ez a függvény hívódik meg, ez pedig a paraméterként kapott Ovodasnak a KutyaMegugat függvényét hívja meg.

Paraméterek:

ovis - a beérkezo Ovodas referenciája

8.2.8 LanyOvodas

Ős:

Ovodas

KozosOs

Attribútumok:

-

Tagfüggvények:

public LanyOvodas()

Konstruktor, az ososztállyal tudatja, hogy Lány.

public void CsokiautomataUdvozol()

A csokiautomata elbámékoztatja a lányokat..

8.2.9 Lista

Ős:

-

Attribútumok:

Proto proto

Tagfüggvények:

public Lista()

A konstruktor, amely egyelore egy Szkeleton példányt hoz létre saját magának, a későbbiekben különösebb feladata nem lesz.

public void beszur(java.lang.String nev, int eredmeny, int szint)

A játék végén a felhasználó által elért eredményt ezzel a függvénnyel menthetjük el. Az adatokat, amit paraméterek közt kap, egy új Lista.ListaElembe teszi, majd hozzá is adja a listához, az Ososztályból örökölt add(Object o) függvénnyel. Utoljára pedig meghívja a rendez() függvényt, ami az új Lista.ListaElemmel bovült listát rendezi eredmények, és szint szerint.

Paraméterek:

nev - a felhasználó neve, amit a játék elején megadott

eredmeny - a felhasználó által összeszedett óvodások száma

szint - a felhasználó által teljesített szint, amely a következo lehet:

1 - Könnyu

2 - Közepes

3 - Nehéz

public void rendez()

A beszur függvény hívja meg, az egész listát rendezi az eredmények, és szint szerint.

8.2.10 Mozgat (interface)

Ős:

-

Attribútumok:

-

Tagfüggvények:

public void Mozog()

Ezt valósítják meg a mozgó objektumok, és ezzel történik a mozgásuk.

8.2.11 Nehezseg

Ős:

-

Attribútumok:

Proto proto

int Szint

Tagfüggvények:

public Nehezseg()

A konstruktor, ami egyenlore alapértelmezésben a Konnyu() szintet futtatja, a közeljövoben kicsit komplexebben fog muködni.

public void BeallitSzint(int szint)

A Szint osztályváltozót fogja beállítani, nem biztos hogy szükség lesz rá a jövoben, de egyelore még itt marad, hátha mégis.

Paraméterek:

szint - a felhasználó által választott szint (a Listanál részleteztem)

public void Konnyu()

A nehézségi szint függvénye közül a Könnyu szintet reprezentáló.

public void Kozepes()

A nehézségi szint függvények közül a Közepes szintet reprezentáló.

public void Nehez()

A nehézségi szint függvények közül a Nehéz szintet reprezentáló.

public int SzintLekerdez()

A Toplista fogja majd meghívni ezt a függvényt, hogy meg tudja milyen szinten játszott a felhasználó.

Visszatérési érték:

a felhasználó által, a játék elején kiválasztott szint

8.2.12 Ovoda

Ős:

KozosOs

Attribútumok:

Szamlalo szamlalo

Tagfüggvények:

public Ovoda()

A kontruktor, ami beállítja a Szamlalot.

public void AktHelyzetBeallit(PalyaElem akt)

Az AktHelyzet változót hivatott beállítani.

public void OvodasLancbolTorol(Ovodas ovis)

A függvény megkapja paraméterként az Ovono után álló elso Ovodast, és ezután sorra destroyolja az Ovodasokat, majd a végén növeli a Szamlalo értékét.

public void OvonoUdvozol(Ovono ovono)

Az Ovono hívja meg, amikor egy PalyaElemre kerül az Ovodaval, ekkor az Ovoda lekéri az elso Ovodas referenciáját az Ovonotol és átadja az OvodasLancbolTorol függvénynek.

Paraméterek:

ovono - az Ovono referenciája, aki idelépett

8.2.13 Ovodas

Ős:

KozosOs

Mozgat (implementálja)

Attribútumok:

public int Allapot

public KozosOs Elotte

public static int fiuk

public static int lanyok

public java.lang.String nem

public int sid

public Ovodas Utana

Tagfüggvények:

public Ovodas(java.lang.String n)

A konstruktor, beállítja az osztályváltozókat.

public void AktHelyzetBeallit(PalyaElem akt)

Az AktHelyzet változót hivatott beállítani.

public void Bameszkodik()

Ha a Csokiautomata vagy Jatekbolt közelében van az Ovodas akkor bámészkodik.

public void EledAllok(Ovodas ovis)

Egy másik Ovodas hívja meg saját referenciájával, amely majd az aktuális Ovodas Elotte nevű változójába kerül, tudomásul véve, hogy már nem az Ovono áll előtte.

Paraméterek:

ovis - az eléálló Ovodas referenciája

public void KutyaMegugat()

Ha a Kutya meglátja az Ovodast ezt a függvényt hívja meg, amivel ráveszi az Ovodast a menekülésre.

public void Menekul()

Ha a Kutya megugatja az Ovodast akkor menekül.

public void Mozog()

A Mozgat interfész Mozog() függvényének megvalósítása, a későbbiekben lesz implementálva.

public void OvodabaKerul()

Amikor az Ovodaba kerül a lurkó, akkor hívódik meg, az "eltűnéshez" van rá szükség.

public void OvonoUdvozol(Ovono ovono)

Az Ovono üdvözli az Ovodast ha meglátja, ekkor utóbbi visszaköszön.

Paraméterek:

ovono - az Ovono referenciája

public void SorbaAll(Ovono ovono, Ovodas ovis)

Az Ovono hívja meg ezt a függvényt, amikor az Ovodast beállítja maga mögé a sorba, ha van elég csokija az eseményhez; ekkor átadja saját referenciáját, és az ElsoOvodas változóját, ami referencia az eddig mögötte állóra, lehet üres is. Az új Ovodas mindig a sor elejére áll, így közli az eddigi elsovel, hogy eléállt.

Paraméterek:

ovono - az Ovono referenciája

ovis - az Ovono mögött álló Ovodas referenciája, lehet üres is

8.2.14 Ovono

Ős:

KozosOs

Attribútumok:

-

Tagfüggvények:

public void AktHelyzetBeallit(PalyaElem akt)

Az AktHelyzet változót hivatott beállítani.

public void CsokiSzamCsokkent()

A CsokiSzam változót csökkenti egyel.

public int CsokiSzamLekerdez()

A CsokiSzam változót adja vissza.

Visszatérési érték:

a CsokiSzam-ot adja vissza.

public void CsokivalFeltolt()

A csokiszámot maximálisra állítja.

public void Mozog()

A Mozgat interfész Mozog() függvényének megvalósítása.

public void OvodasLancbaFuz(Ovodas ovis)

Ez a függvény kéri meg az Ovodast, hogy álljon be a sorba.

Paraméterek:

ovis - a sorbafuzendo Ovodas referenciája

public Ovodas OvodasLead()

Az ElsoOvodas változót adja vissza az Ovodanak, amikor vele van interakciója.

Visszatérési érték:

az ElsoOvodas-t adja vissza

public void OvodasUdvozol(Ovodas ovis)

Az Ovodas hívja meg ezt a függvényt, amikor azonos PalyaElemre lép, mint ahol az Ovono található (tehát így köszön egy illedelmes Ovodas).

Paraméterek:

ovis - az Ovodas referenciája, aki meghívja a függvényt

8.2.15 Palya

Attribútumok:

Vector PalyaElemek
final int Sorhossz
final int OsszPalyaElem
int OsszOvodas
int OsszCsokiautomata
int OsszJatekbolt
int OsszKutya
public Proto proto

Tagfüggvények:

public void PalyaEpit()

Létrehozza a PalyaElemek sokaságát, és elindítja a szomszédok beállítását. Létrehozza az objektumokat, mindegyikből adott darabnyit.

public void OsszOvodasBeallit(int x)

A függvény nevében lévo változót állítja be a kapott x értékre.

public void OsszCsokiautomataBeallit(int x)

A függvény nevében lévo változót állítja be a kapott x értékre.

public void OsszJatekboltBeallit(int x)

A függvény nevében lévo változót állítja be a kapott x értékre.

public void OsszKutyaBeallit(int x)

A függvény nevében lévo változót állítja be a kapott x értékre.

public void SzomszedBeallit()

A PalyaElemeket szövi össze egy pályává, mindegyiknek beállítja a 6 szomszédját (méhkaptár elrendezés).

public int getID(int s, int o)

Koordinatak alapjan visszaadja a PalyaElem id-jét a Vectorban. Csak a SzomszedBeallit() használja. Paraméterek: s - a sor sorszáma; o - az oszlop sorszáma.

public Palya()

Új proto objektumot hoz létre a naplózáshoz és új vektort hoz létre a PalyaElemek tárolására.

8.2.16 PalyaElem

Attribútumok:

Vector Szomszedok
Vector Objektumok
static int palyaelemek
Proto proto
java.lang.String sid

Tagfüggvények:

public PalyaElem Atad(KozosOs kos, int irany)

Az Atad függvény az egyik, ami felelos az objektumok mozgásáért, a mozgó objektumok fogják meghívni ezen függvényt, így közlik a PalyaElemmel, amin vannak, hogy merre szeretnének továbbmenni. Ennek hatására a PalyaElem felkeresi a szomszédját és átadja a mozgó objektumot, illetoleg a mozgó objektumnak visszaadja a szomszédja referenciáját. Paraméterek: kos - a mozgó objektum referenciája, amit a szomszéd PalyaElemnek kell átadni; irany - az irány, amerre a mozgó objektum menni kíván, a következok szerint: 0 - balra; 1 - bal-fel; 2 - jobb-fel; 3 - jobbra; 4 - jobb-le; 5 - bal-le. Visszatérési érték: a megadott irányban lévo szomszéd PalyaElem referenciáját adja vissza.

public void Atvesz(KozosOs kos)

Az Atad függvény párja, a vételi oldal. A paraméterként kapott objektumot beteszi a saját objektumai közé (amik rajta vannak). Paraméterek: kos - a mozgó objektum referenciája, amely most lépett át erre a PalyaElemre.

public java.util.Vector LekerdezObjektumok()

A függvény hatására a hívó objektum megkapja, hogy milyen objektumok tartózkodnak a PalyaElemen. Ezt a funkciót a mozgó objektumok Mozog függvénye fogja hívni. Visszatérési érték: a PalyaElemen található Objektumokat adja vissza.

public void SzomszedokBeallit(PalyaElem l, PalyaElem lt, PalyaElem rt, PalyaElem r, PalyaElem rb, PalyaElem lb)

A PalyaElem szomszédait állítja be.

Paraméterek:

l - bal PalyaElem szomszéd

lt - balfelso PalyaElem szomszéd

rt - jobbfelso PalyaElem szomszéd

r - jobb PalyaElem szomszéd

rb - jobbalsó PalyaElem szomszéd

lb - balalsó PalyaElem szomszéd

public java.lang.String make2digit(int x)

Kétjegyuvé teszi az egyjegyu számokat is.

public PalyaElem()

Létrehoz egy proto objektumot, két vektort, illetve beállítja a saját azonosítóját (sid).

8.2.17 Proto

Attribútumok:

static int Tab

static int utem

Tagfüggvények:

public void tab(int a)

A függvény egymásrahatások vizsgálatához könnyítésképpen bevezetett megoldás; ha egy függvényt egy másik hív meg, akkor, annak beljebb kell elhelyezkednie, ezt behúzással jelezzük. Szükségképpen ez a függvény a paraméterként kapott értékkel módosítja a behúzás mértékét. Paraméterek: a - a behúzás mértéke, többnyire 1 vagy -1 értéket kap.

public void Kiir(java.lang.String s)

A Kiir függvény első formája, amely String-et tud kiírni az éppen aktuális behúzás mértékével együtt. Paraméterek: s - a kiírandó String

public void Hiba(java.lang.String s)

A Kiir függvény második formája, amely String-et tud kiírni az éppen aktuális behúzás mértékével együtt. Paraméterek: s - a kiírandó String

public boolean Beker(java.lang.String s)

A beolvasást végzo függvény, a paraméterként kapott szöveget írja ki a felhasználónak, majd igen/nem választ vár tole, amit visszaad a függvényt lehívó objektumnak. Paraméterek: s - kiírandó szöveg (kérdés a felhasználó felé). Visszatérési érték: true-val tér vissza ha Igen választ kapott, false-al, ha Nemleges választ kapott a felhasználótól.

public int SzamotBeker(java.lang.String s)

A beolvasást végzo függvény, a paraméterként kapott szöveget írja ki a felhasználónak, majd igen/nem választ vár tole, amit visszaad a függvényt lehívó objektumnak. Paraméterek: s - kiírandó szöveg (kérdés a felhasználó felé). Visszatérési érték: a számláló által megadott Integer szám.

public void EntertVar(java.lang.String s)

A Beker függvény primitívebb formája, csak egy bármilyen karakter lenyomását várjuk, ami nem lesz visszadva. Paraméterek: s - a felhasználó számára kiírandó szöveg (pl. Üss entert).

public int menu()

A menü függvényünk, amely a felhasználó számára láthatóvá teszi a menüt, illetve beolvassa a választott menüpontot, majd azzal visszatér (a Jatek objektum fogja hívni és kapni az eredményt).

public Proto()

Semmi feladata jelenleg.

8.2.18 Szamlalo

Attribútumok:

int Eredmeny
Proto proto

Tagfüggvények:

public int EredmenytLekerdez()

A játék végén a Toplista fogja lekérdezni a játékos által elért pontszámot, ezt adja vissza.

Visszatérési értéke:

a saját Eredmeny változóját adja vissza - a felhasználó által az Ovodaba juttatott Ovodasok száma

public void EredmenytNovel(int x)

Az Ovoda fogja használni ezt a függvényt, amely az Eredmeny változó növelésére szolgál. Paraméterei: x - ennyivel fogja megnövelni az Eredmeny értékét.

public Szamlalo()

Új proto objektumot hoz létre a naplózáshoz.

8.2.19 Toplista

Attribútumok:

public Lista lista
public Proto proto
public Szamlalo szamlalo

Tagfüggvények:

public void Megjelenit()

A Jatek osztály fogja meghívni ezt a függvényt, vagy a menübol, vagy a játék végén történhet ez. Ha a menübol történik, akkor csak egyszeruen kiíratjuk a listankat, ha a játék végén, akkor a jelenlegi felhasználó adatait beszúrjuk a listánkba, lementjük az új listát az eredményeket tartalmazó fájlunkba, majd megjelenítjuk a felhasználó számára a listát.

public Toplista()

A konstruktor, amely beállítja az osztály attributumok értékeit: létrehoz egy Proto és egy Lista objektumot, illetve átveszi egy Szamlalo tipusu objektum referenciáját.

8.3 A tesztek részletes tervei:

8.3.1 Új játék kezdése

Ezt vizsgáljuk:
Új játék indításának helyessége a játék elkezdésekor, vagy pályáról való kilépés után. Figyelni kell arra, hogy a pálya betöltődött-e az elvártaknak megfelelően.
Megvalósítás:
A pályán minden lehetséges objektumból egyet (vagy többet) létrehozunk és a játékot elindítjuk az 'n' betű lenyomásával.
Várt eredmény:

A pálya felépül és a játék elindul.

Bemeneti fájl:

Gábor Zsazsa

ngk

Hibaforrás:

az esetleges hiba a Jatek konstruktorában keresendő.

8.3.2 Kutya megugatja az óvodást

Ezt vizsgáljuk:

Ha egy kutya találkozik, egy óvodással, akár fiú akár lány meg kell hogy ugassa. Ha ekkor az óvodások sorban voltak, akkor ki kell, hogy szakadjanak.

Megvalósítás:

Több részletben is. Először egy kutya és egy óvodás objektum találkozására várunk. Másodszor pedig már egy óvodás sort hozunk létre az óvónővel, hogy tesztelhessük a kiszakadást.

Várt eredmény:

A kutya megugatja az óvodást, nemétől függetlenül. Egy esetleges szakadást is várunk, ha sorban voltak.

Bemeneti fájl:

Cplusplus Jenő

nggggggggggggggkngbfgblgjlgjgjgggggggggggggk

Hibaforrás:

Hiba a Ovodas.KutyaMegugat() függvényében keresendő.

8.3.3 Óvodás meglátja a Kutyát

Ezt vizsgáljuk:

Az óvodás találkozik a kutyával, aki szintén észreveszi és meg kell hogy ugassa. Ekkor az óvodásnak menekülnie kell. Ha a találkozáskor az óvodás sorban állt, akkor ki kell szakadnia belőle.

Megvalósítás:

Több részletben is. Először egy óvodás és egy kutya objektum találkozására várunk. Másodszor pedig egy óvodás sort hozunk létre, hogy tesztelhessük a kiszakadást.

Várt eredmény:

Ha találkoznak, a kutyának meg kell ugatnia, mire az óvodásnak menekülnie kell. Ha sorban állt, akkor egy kiszakadást is várunk.

Bemeneti fájl:

Java Judit

nggggggggggggggkngbfgblgjlgjgjgggggggggggggggggggggk

Hibaforrás:

Az esetleges hiba az Ovodas.kutyaMegugat() vagy a Kutya.OvodasUdvozol() függvényében keresendő.

8.3.4 Óvodás(lány) bámészkodik a csokiautomata előtt

Ezt vizsgáljuk:

Ha egy óvodás(lány) találkozik egy csokiautomatával, akkor ennek hatására ki kell szakadnia a sorból. Ellenben a fiú óvodásra nem szabad hatással legyen.

Megvalósítás:

Létrehozunk egy lány ovodás sort és a sort a csokiautomatához irányítjuk. Létrehozunk egy fiú óvodásokból álló sort is, ezt is odairányítjuk.

Várt eredmény:

Az óvodás lány üdvözli a csokiautomatát, az viszont üdvözli, mire a lány elbámészkodik.

Bemeneti fájl:

Merengő Márk

nggggggggggggggkngjfgjlgblgbgbgbgbgbgbgggggggggggk

Hibaforrás:

Az esetleges hibák a Ovodas.Bameszkodik() vagy az Ovodas.CsokiautomataUdvozol függvényében keresendő.

8.3.5 Óvodás(fiú) bámészkodik a játékbolt előtt

Ezt vizsgáljuk:

Ha egy fiú óvodás találkozik egy játébolttal, akkor ennek hatására ha sorban állt, akkor ki kell szakadnia, a lányokra nem szabad hatással legyen.

Megvalósítás:

Létrehozunk egy fiú óvodás sort és egy játékbolt elé irányítjuk. Ezután a következő tesztelés miatt létrehozunk egy fiú sort, melyet ugyancsak a játékbolt elé irányítjuk.

Várt eredmény:

Ha egy fiú óvodás találkozik a játékobolttal, akkor ennek hatására bámészkodni fog, ez azt jelenti, hogy ha sorban állt akkor kifog szakadni bizonyos valószínűséggel.

Bemeneti fájl:

Burgenlander Benő

nggggggggggggggkngbfgblgjlgjgbfgbgbgbgbgbgggggggggggk

Hibaforrás:

Az esetleges hiba az Ovodas.Bameszkodik() vagy a Ovodas.JatekboltUdvozol() függvényben keresendő.

8.3.6 Csokifelvétel

Ezt vizsgáljuk:

Az óvónőnek az óvodások sorbafűzéshez csokira van szüksége. Mivel az óvónőnél, csak 5 csoki lehet, ezért az utánpótlásért a csokiautomatához mehet. Ennek a folyamatnak a helyességét vizsgáljuk.

Megvalósítás:

Az óvonőt odairányítjuk a csokiautómatához. A 'c' parancsgomb leütésével kezdeményezzük a csokifeltőltést.

Várt eredmény:

Az óvónő csokiszáma maximális lesz.

Bemeneti fájl:

Almonovics Alfréd

ngbgbgbgbgcggk

Hibaforrás:

Az esetleges hibák az Ovono.CsokivalFeltolt vagy az Ovono.CsokiautomataUdvozol() függvényben keresendő.

8.3.7 Óvodás sorbaállítása (egynemű)

Ezt vizsgáljuk:

Egy óvodás csak akkor állhat sorba, ha a sorban csak vele megegyező nemű ovodások vannak.

Megvalósítás:

Létrehozunk egy egynemű óvodásokból álló sort és az azonos nemű óvodásunkhoz irányítjuk. Az óvodás sorbaállításához az 's' parancsgombot használjuk.

Várt eredmény:

Az óvónő sorbaállítja az óvodást, utóbbi közli a lehetséges mögötte lévővel hogy eléállt, így a sor kibővül egy taggal és nem bomlik szét sehol.

Bemeneti fájl:

Goromba Gerzson

nbfgblgjlgblgblgblggk

Hibaforrás:

Az esetleges hibák az Ovono.OvodasLancbaFuz() vagy az Ovodas.SorbaAllit() függvényben keresendők.

8.3.8 Óvodás sorbaállítása (fiú-lány)

Ezt vizsgáljuk:

Ha ellenkező neműekből áll a sor, akkor az új óvodás lesz a sor egyetlen tagja, mivel ellenkező nemű társai nem akarnak vele egy sorban lenni, ezért ilyenkor az aktuális sornak fel kell bomlania.

Megvalósítás:

Létrehozunk egy egynemű óvodás sort és egy ellenkező nemüt próbálunk meg sorbaállítani az 's' betűvel.

Várt eredmény:

Felbomlik a sor az ellenkező neműség miatt.

Bemeneti fájl:

Furfangos Ferenc

njfgjlgblgjgjgblgbgbgbfgbfggggk

Hibaforrás:

Az esetleges hibák a Ovono.OvodasLancbaFuz() vagy az Ovodas.SorbaAllit() függvényben keresendők.

8.3.9 Óvodások leadása

Ezt vizsgáljuk:

Az óvónő csak itt szabad tudja leadni az óvodásokat, miután sorbaállította őket.

Megvalósítás:

Létrehozunk egy óvodás sort és odavezetjük az óvodához, ahol az 'o' parancsgomb lenyomásával kezdeményezzük az óvodások leadását.

Várt eredmény:

Mikor az óvonő összegyűjtötte az óvodásokat azt az óvodában kell leadni, a leadott óvodások számával fog a számláló értéke növekedni. Ezután törlödnek a sórból az óvosások.

Bemeneti fájl:

Humoros Hallgato

njlgjfgbfgjfgjfgjfgjfgjfgjfgjfgggggggggk

Hibaforrás:

Az esetleges hibákat a Ovono.OvodasLead() vagy az Ovoda.OvodasLancbolTorol() függvényben keresendők.

8.3.10 Számláló növelése

Ezt vizsgáljuk:

Mikor az óvónő bevitte az óvodásokat az óvodába, a számlálónak növekednie kell annak függvényében, hogy hány óvodást tudott leadni az óvónő. A játék végén a Toplistának le kell kérdeznie az értékét, s eszerint kell majd bekerüljön a játékos eredménye a toplistába.

Megvalósítás:

Leadunk egy óvodás sort az óvodában.

Várt eredmény:

Növekedik a számláló értéke a bevitt óvodások számával arányosan.

Bemeneti fájl:

Termetes Tarantula

njlgjfgbfgjfgjfgjfgjfgjfgjfgjfgggggggggggggk

Hibaforrás:

Az esetleges hibákat a Szamlalo.EredmenyNovel() függvényében keresendők.

8.3.11 Toplista megtekintése a menüből

Ezt vizsgáljuk:

A menüből kiválasztva meg kell hogy tekinthessük a toplistát, mely a legjobb eredményeket kell hogy tárolja, amelyeket egy fájlból kell beolvasson.

Megvalósítás:

A toplista megtekintéséhez a 't' parancsgombot használjuk.

Várt eredmény:

A toplista beolvassa a fájlból az eredményeket, egy listát feltölt vele, majd ezt megjeleníti.

Bemeneti fájl:

Remete Renáta

ntgk

Hibaforrás:

Az esetleges hibák a Toplista.Megjelenit() függvényben lehetnek.

8.3.12 Toplista megtekintése a játék végén

Ezt vizsgáljuk:

A játék idő leteltével lehetőségünk kell legyen arra, hogy megtekintsük a toplistát, mely a legjobb eredményeket kell, hogy eltárolja egy fájlban, illetve a játék folyamán egy listában.

Megvalósítás:

A játék végén (az idő leteltével) lehetőség van a toplista megtekintésére a 't' parancsgomb lenyomásával.

Várt eredmény:

Egy toplistát tartalmazó fájl, melyet sikeresen le tudunk menteni és megjeleníteni.

Bemeneti fájl:

ngggggggggggggggggggggggggggggggggggggggggggggggtgk

Hibaforrás:

Az esetleges hibák az Ido.IdoTelik() vagy a Toplista.Megjelenit() függvényeiben lehetnek.

8.3.13: Szint újrajátszása

Ezt vizsgáljuk:

Amikor a játékidő végére érünk, választanunk kell, hogy a toplista megtekintése után kilépünk, vagy újra játszunk ugyanazon a nehézségi szinten. A játék újbóli betöltését vizsgáljuk.

Megvalósítás:

A játék végén leütjük az 'r' billentyűt és kiválasztjuk, hogy mit szeretnénk.

Várt eredmény:

Vagy megjelenik a toplista, vagy pedig sikeresen be kell hogy töltődjön a játék.

Bemeneti fájl:

Szintetikus Szilvia

ngggggggggggggggggggggggggggggggggggggggggggggggtgrggk

Hibaforrás:

Az esetleges hibák az Ido.IdoTelik() függvényben keresendők.

8.3.14 Idő lejár

Ezt vizsgáljuk:

Amikor az idő lejár, akkor véget kell érjen a játék és a rendszernek le kell ellenőriznie a számláló értéke alapján, hogy felkerülünk-e a toplistára vagy sem.

Megvalósítás:

Játszunk, amíg lejár az idő.

Várt eredmény:

Mikor letelik az idő, akkor a játék megszakad, és felkerülünk a toplistára ha a számláló értéke nagyon a legkisebb toplistás értéknél.

Bemeneti fájl:

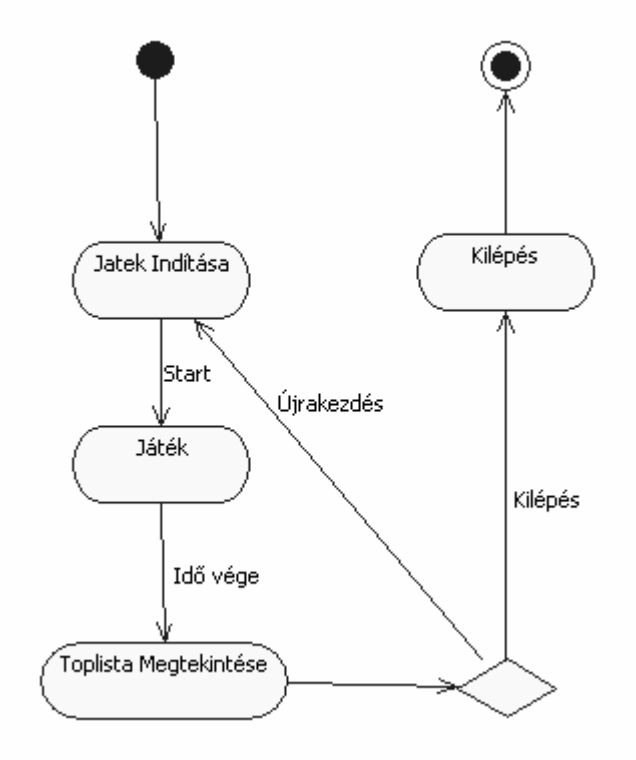
nggggggggggggggggggggggggggggggggggggggggggggggggggggggggggk

Hibaforrás:

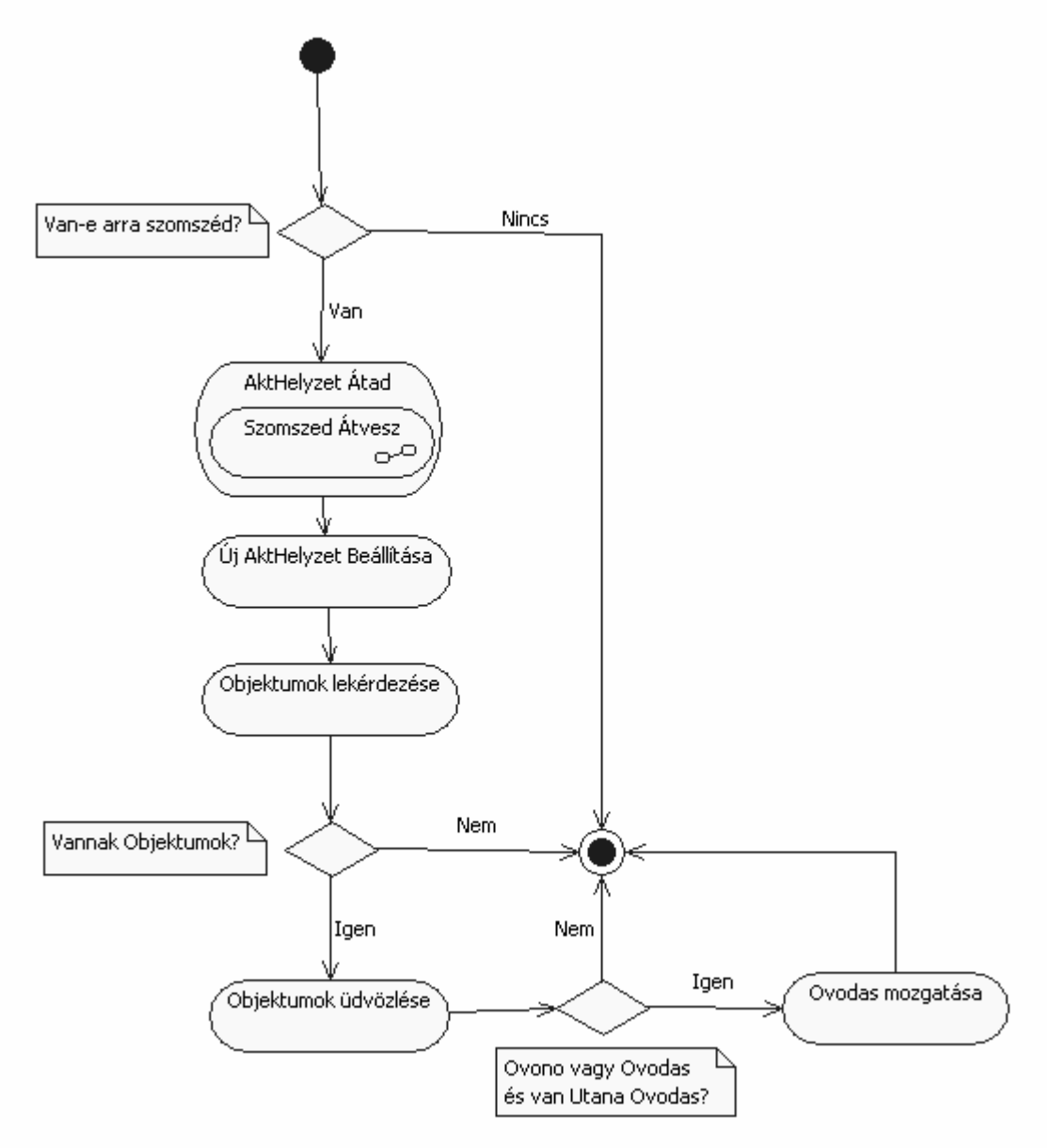
Az esetleges hibák az Ido.IdoTelik() függvényben vagy a Játék konstruktorában keresendők.

8.4 Teszteléshez használt segédprogramok

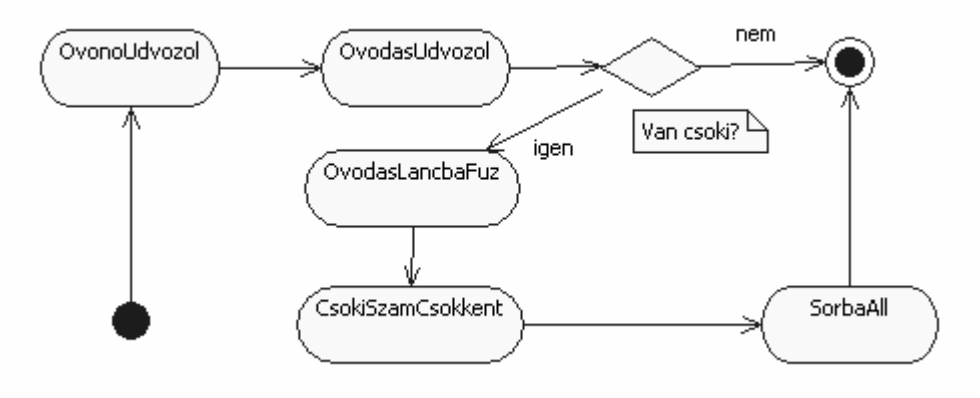
A tesztelés során nem használunk különféle segédprogramokat. A naplófájlok elemzésével végezzük el az egyes tesztesetek ellenőrzését. A viszonylag kis számú teszteset és azok egyszerűsége miatt döntöttünk így.



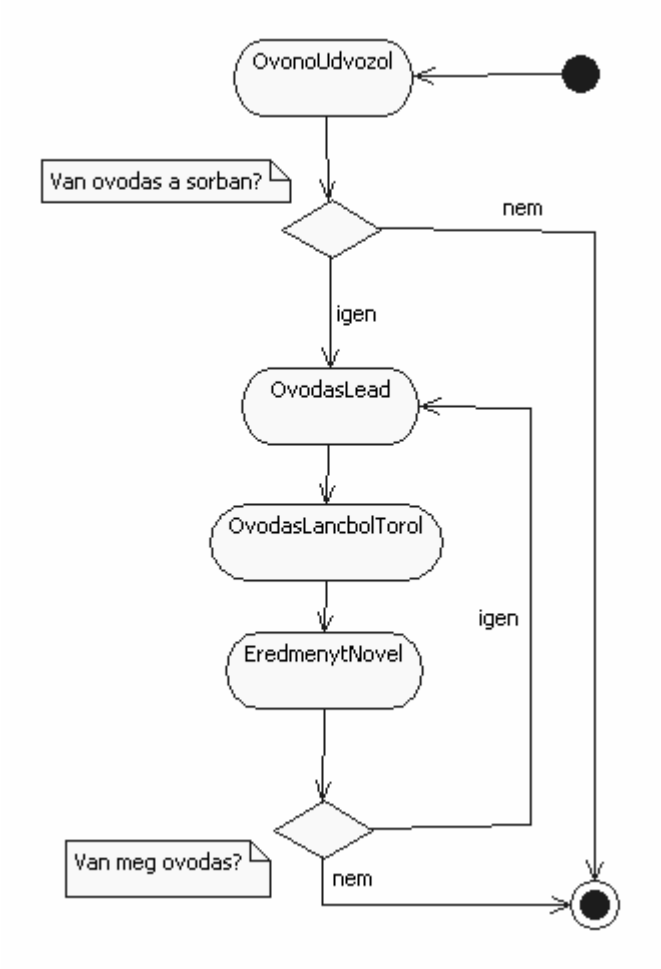
1. ábra. A játék újrakezdése



2. ábra Mozgás modellezése



3. ábra A sorbaállás modellezése



4. ábra Az óvodás leadásának modellezése

10. Prototípus beadása

10.1 A tesztelés eredményeinek összefoglalása

A program írása során törekedtünk a folyamatos tesztelésre, ellenőrzésre. Így a tesztelés során kevés hiba adódott.

A teszt során a képernyő kimenetet és a napló fájlt egyaránt tanulmányoztuk. A várt és kapott eredmény ennek megfelelően van leírva.

10.2 A mellékelt állomány tartalma

Az InFoka\Bemenetek könyvtár tartalmazza a teszteléshez szükséges bemeneti fileokat, listázva:

2006.04.24. 14:58	<DIR>	.
2006.04.24. 14:58	<DIR>	..
2006.04.24. 09:33	11	test1.in
2006.04.24. 11:24	18	test10.in
2006.04.24. 11:25	10	test11.in
2006.04.24. 14:18	55	test12.in
2006.04.24. 14:08	55	test13.in
2006.04.24. 12:27	55	test14.in
2006.04.24. 11:21	52	test2.in
2006.04.24. 11:21	52	test3.in
2006.04.24. 11:21	52	test4.in
2006.04.24. 11:21	52	test5.in
2006.04.24. 11:22	14	test6.in
2006.04.24. 14:32	38	test7.in
2006.04.24. 11:23	31	test8.in
2006.04.24. 11:24	18	test9.in
	14 fájl	513 bájt

Az Infoka\Javadoc nevű könyvtár tartalmazza a Java dokumentációt:

2006.04.24. 14:58	<DIR>	.
2006.04.24. 14:58	<DIR>	..
2006.04.24. 14:54	2 475	allclasses-frame.html
2006.04.24. 14:54	2 095	allclasses-noframe.html
2006.04.24. 14:54	4 599	constant-values.html
2006.04.24. 14:54	4 483	deprecated-list.html
2006.04.24. 14:54	7 717	help-doc.html
2006.04.24. 14:54	58 719	index-all.html
2006.04.24. 14:54	821	index.html
2006.04.24. 14:54	7 795	overview-tree.html
2006.04.24. 14:54	11	package-list
2006.04.24. 14:54	706	packages.html
2006.04.24. 14:58	<DIR>	ProtOvoda
2006.04.24. 14:58	<DIR>	resources
2006.04.24. 14:54	5 309	serialized-form.html
2006.04.24. 14:54	1 231	stylesheet.css
	12 fájl	95 961 bájt

Az InFoka\Kimenetek könyvtár tartalmazza a System.Out-ra érkező üzenetek, a pálya kirajzolás, a kommunikáció a userral, tartalma:

2006.04.24. 14:58	<DIR>	.
2006.04.24. 14:58	<DIR>	..
2006.04.24. 11:30	416	test1.out
2006.04.24. 11:37	3 557	test10.out
2006.04.24. 12:15	338	test11.out
2006.04.24. 14:19	15 817	test12.out
2006.04.24. 14:53	9 730	test13.out
2006.04.24. 12:29	15 762	test14.out
2006.04.24. 11:31	14 370	test2.out
2006.04.24. 11:32	14 370	test3.out

2006.04.24. 11:33	14 370	test4.out
2006.04.24. 11:34	14 370	test5.out
2006.04.24. 11:35	2 161	test6.out
2006.04.24. 14:33	10 822	test7.out
2006.04.24. 11:36	8 208	test8.out
2006.04.24. 11:37	3 728	test9.out
	14 fájl	128 019 bájt

Az Infoka\ProtOvoda tartalma a forrásfileok és két almappa:

2006.04.24. 14:58	<DIR>	.
2006.04.24. 14:58	<DIR>	..
2006.04.24. 14:58	<DIR>	Naplozas
2006.04.24. 14:58	<DIR>	ProtOvoda
2006.04.24. 06:34	2 125	Csokiautomata.java
2006.04.24. 06:28	850	FiuOvodas.java
2006.04.24. 14:15	1 543	Ido.java
2006.04.24. 13:48	1 866	Jatek.java
2006.04.24. 06:34	1 488	Jatekbolt.java
2006.04.24. 05:48	1 486	KozosOs.java
2006.04.24. 09:36	2 168	Kutya.java
2006.04.24. 06:28	856	LanyOvodas.java
2006.04.24. 12:15	4 597	Lista.java
2006.04.24. 02:05	295	Mozgat.java
2006.04.24. 09:38	3 188	Nehezseg.java
2006.04.24. 08:10	2 218	Ovoda.java
2006.04.24. 09:25	5 730	Ovodas.java
2006.04.24. 09:37	3 686	Ovono.java
2006.04.24. 14:38	14 031	Palya.java
2006.04.24. 06:34	4 078	PalyaElem.java
2006.04.24. 13:54	4 920	Proto.java
2006.04.24. 06:28	1 369	Szamlalo.java
2006.04.24. 14:43	169	toplista
2006.04.24. 06:34	1 426	Toplista.java

2006.04.24. 14:43	137 726	user.log
	21 fájl	195 815 bájt

Az Infoka\ProtOvoda\Naplozas ide kerültek a teszt esetek alatt naplózásra került történések, minden testhez külön file:

2006.04.24. 14:58	<DIR>	.
2006.04.24. 14:58	<DIR>	..
2006.04.24. 11:30	15 424	test1.log
2006.04.24. 11:37	45 193	test10.log
2006.04.24. 12:15	489	test11.log
2006.04.24. 14:19	131 406	test12.log
2006.04.24. 14:43	137 726	test13.log
2006.04.24. 12:29	133 991	test14.log
2006.04.24. 11:31	104 170	test2.log
2006.04.24. 11:32	94 878	test3.log
2006.04.24. 11:33	98 965	test4.log
2006.04.24. 11:34	117 971	test5.log
2006.04.24. 11:35	28 729	test6.log
2006.04.24. 14:33	108 832	test7.log
2006.04.24. 14:25	134 443	test8.log
2006.04.24. 11:37	41 344	test9.log
	14 fájl	1 193 561 bájt

Az Infoka\ProtOvoda\ProtOvoda a lefordított program, futtatható verzió, tartalma:

2006.04.24. 14:58	<DIR>	.
2006.04.24. 14:58	<DIR>	..
2006.04.24. 14:41	1 725	Csokiautomata.class
2006.04.24. 14:41	805	FiuOvodas.class
2006.04.24. 14:41	1 408	Ido.class
2006.04.24. 14:41	1 497	Jatek.class
2006.04.24. 14:41	1 498	Jatekbolt.class
2006.04.24. 14:41	1 021	KozosOs.class

2006.04.24. 14:41	2 295	Kutya.class
2006.04.24. 14:41	804	LanyOvodas.class
2006.04.24. 14:41	722	Lista$ListaElem.class
2006.04.24. 14:41	2 828	Lista.class
2006.04.24. 14:41	126	Mozgat.class
2006.04.24. 14:41	1 930	Nehezseg.class
2006.04.24. 14:41	1 584	Ovoda.class
2006.04.24. 14:41	4 431	Ovodas.class
2006.04.24. 14:41	2 873	Ovono.class
2006.04.24. 14:41	7 611	Palya.class
2006.04.24. 14:41	2 390	PalyaElem.class
2006.04.24. 14:41	4 291	Proto.class
2006.04.24. 14:41	874	Szamlalo.class
2006.04.24. 14:41	977	Toplista.class
	20 fájl	41 690 bájt

10.3 Használati Útmutató

A mellékelt zip fájlt (ProtOvoda_InFoka.zip) ki kell csomagolni egy tetszőleges mappába, majd ki kell adni a run parancsot a fordításhoz és futtatáshoz. A run script futtatásához szükségeltetik, hogy a java fordító benne legyen az elérési útban.

A paraméter nélküli run script futtatása elindítja a programot, és a manuálisan használható, akár az igazi játék. Amennyiben paraméterrel hívjuk meg, ami lehet 1..14-ig terjedő egész szám, akkor az annak mefelelő teszteset fut le. Ezekhez segítség, illetve futtatási példa: javac -d ProtOvoda ProtOvoda*.java

cd ProtOvoda

java ProtOvoda.Jatek user // ez a normál használat

java ProtOvoda.Jatek Naplozas\test1 < Bemenetek\test1.in > Kimenetek\test1.out // tehát bemeneti és kimeneti fájlokkal

10.4 A tesztelés eredményeinek összefoglalása

A program írása során törekedtünk a folyamatos tesztelésre, ellenőrzésre. Így a tesztelés során kevés hiba adódott.
A teszt során a képernyő kimenetet és a napló fájlt egyaránt tanulmányoztuk. A várt és kapott eredmény ennek megfelelően van leírva.
A következő teszteket hajtottuk végre:

10.4.1 Új játék kezdése

A teszt célja:
Új játék indításának helyessége a játék elkezdésekor, vagy pályáról való kilépés után. Egy pálya betöltésének nyomon követése.
Várt eredmény:
A pálya felépül és a játék elindul.
Kapott eredmény:
A játék látható módon építi fel magát, előbb a nehézségi szint állítódik be, majd létrejön a pálya, mely létrehozza a rajta lévő objektumok valamennyi példányát, illetve összeszövi a PályaElemeket egy Pályává.
Naplófájl:

Log: 0.utem -> Jatek::Jatek inditasa
Log: 0.utem -> Nehezseg::Letrehozva
Log: 0.utem -> Nehezseg::Konnyu szint inditasa
Log: 0.utem -> Palya::Letrehozva
Log: 0.utem -> Palya::Osszes ovodas szamanak beallitasa -> 8
Log: 0.utem -> Palya::Osszes jatekbolt szamanak beallitasa -> 3
Log: 0.utem -> Palya::Osszes csokiautomata szamanak beallitasa -> 3
Log: 0.utem -> Palya::Osszes kutya szamanak beallitasa -> 1
Log: 0.utem -> Palya::Palya epites elkezdve
Log: 0.utem -> Palya::PalyaElem-ek letrehozasa
Log: 0.utem -> PalyaElem[00]::Letrehozva
Log: 0.utem -> PalyaElem[01]::Letrehozva

```
Log: 0.utem -> PalyaElem[02]::Letrehozva
Log: 0.utem -> .....
Log: 0.utem -> PalyaElem[92]::Letrehozva
Log: 0.utem -> PalyaElem[93]::Letrehozva
Log: 0.utem -> PalyaElem[94]::Letrehozva
Log: 0.utem -> Palya::PalyaElem-ek osszelancolasanak inditasa
Log: 0.utem -> PalyaElem[00]::Szomszedok beallitva
Log: 0.utem -> PalyaElem[01]::Szomszedok beallitva
Log: 0.utem -> PalyaElem[02]::Szomszedok beallitva
Log: 0.utem -> .....
Log: 0.utem -> PalyaElem[92]::Szomszedok beallitva
Log: 0.utem -> PalyaElem[93]::Szomszedok beallitva
Log: 0.utem -> PalyaElem[94]::Szomszedok beallitva
Log: 0.utem -> Palya::PalyaElem-ek osszelancolasa befejezve
Log: 0.utem -> Palya::Ovono letrehozasa
Log: 0.utem -> Ovono::Letrehozva
Log: 0.utem -> Ovono::AktHelyzet beallitva
Log: 0.utem -> PalyaElem[52]::Objektum atveve
Log: 0.utem -> Palya::Ovoda letrehozasa
Log: 0.utem -> Ovoda::Letrehozva
Log: 0.utem -> Ovoda::AktHelyzet beallitva
Log: 0.utem -> PalyaElem[09]::Objektum atveve
Log: 0.utem -> Palya::Ovodasok letrehozasa
Log: 0.utem -> FiuOvodas[0]::Letrehozva
Log: 0.utem -> FiuOvodas[0]::AktHelyzet beallitva
Log: 0.utem -> PalyaElem[87]::Objektum atveve
Log: 0.utem -> FiuOvodas[1]::Letrehozva
Log: 0.utem -> FiuOvodas[1]::AktHelyzet beallitva
Log: 0.utem -> PalyaElem[79]::Objektum atveve
Log: 0.utem -> FiuOvodas[2]::Letrehozva
Log: 0.utem -> FiuOvodas[2]::AktHelyzet beallitva
Log: 0.utem -> PalyaElem[40]::Objektum atveve
Log: 0.utem -> FiuOvodas[3]::Letrehozva
Log: 0.utem -> FiuOvodas[3]::AktHelyzet beallitva
```

Log: 0.utem -> PalyaElem[80]::Objektum atveve

Log: 0.utem -> LanyOvodas[0]::Letrehozva

Log: 0.utem -> LanyOvodas[0]::AktHelyzet beallitva

Log: 0.utem -> PalyaElem[45]::Objektum atveve

Log: 0.utem -> LanyOvodas[1]::Letrehozva

Log: 0.utem -> LanyOvodas[1]::AktHelyzet beallitva

Log: 0.utem -> PalyaElem[33]::Objektum atveve

Log: 0.utem -> LanyOvodas[2]::Letrehozva

Log: 0.utem -> LanyOvodas[2]::AktHelyzet beallitva

Log: 0.utem -> PalyaElem[26]::Objektum atveve

Log: 0.utem -> LanyOvodas[3]::Letrehozva

Log: 0.utem -> LanyOvodas[3]::AktHelyzet beallitva

Log: 0.utem -> PalyaElem[35]::Objektum atveve

Log: 0.utem -> Palya::Csokiautomata(k) letrehozasa

Log: 0.utem -> Csokiautomata[0]::Letrehozva

Log: 0.utem -> Csokiautomata[0]::AktHelyzet beallitva

Log: 0.utem -> PalyaElem[10]::Objektum atveve

Log: 0.utem -> Csokiautomata[1]::Letrehozva

Log: 0.utem -> Csokiautomata[1]::AktHelyzet beallitva

Log: 0.utem -> PalyaElem[11]::Objektum atveve

Log: 0.utem -> Csokiautomata[2]::Letrehozva

Log: 0.utem -> Csokiautomata[2]::AktHelyzet beallitva

Log: 0.utem -> PalyaElem[22]::Objektum atveve

Log: 0.utem -> Palya::Jatekbolt(ok) letrehozasa

Log: 0.utem -> Jatekbolt[0]::Letrehozva

Log: 0.utem -> Jatekbolt[0]::AktHelyzet beallitva

Log: 0.utem -> PalyaElem[42]::Objektum atveve

Log: 0.utem -> Jatekbolt[1]::Letrehozva

Log: 0.utem -> Jatekbolt[1]::AktHelyzet beallitva

Log: 0.utem -> PalyaElem[77]::Objektum atveve

Log: 0.utem -> Jatekbolt[2]::Letrehozva

Log: 0.utem -> Jatekbolt[2]::AktHelyzet beallitva

Log: 0.utem -> PalyaElem[17]::Objektum atveve

Log: 0.utem -> Palya::Kutya(k) letrehozasa

Log: 0.utem -> Kutya[0]::Letrehozva

Log: 0.utem -> Kutya[0]::AktHelyzet beallitva

Log: 0.utem -> PalyaElem[48]::Objektum atveve

Log: 0.utem -> Palya::Palya epites befejezve

Log: 0.utem -> Ido::Letrehozva

Log: 0.utem -> Ido::Az ido mulasa elkezdodott - 45másodperc a tesztekhez

Log: 0.utem -> .....

Log: 1.utem -> Palya::Utasitas: k

Log: 1.utem -> Jatek vege, kilepes, Viszlat!

10.4.2 Kutya megugatja az óvodást

A teszt célja:

Ha egy kutya találkozik, egy óvodással, akár fiú akár lány meg kell hogy ugassa. Ha ekkor az óvodások sorban voltak, akkor ki kell, hogy szakadjanak.

Várt eredmény:

A kutya megugatja az óvodást, nemétől függetlenül. Egy esetleges szakadást is várunk, ha sorban voltak.

Kapott eredmény:

A kutya véletlenszerűen mozog a pályán, jelen esetben a 9. ütem során sikerült megtámadnia egy fiút. A folyamat helyesen lefolytatódott.

Naplófájl:

Log: 9.utem -> Kutya[0]::Mozogni kezd

Log: 9.utem -> PalyaElem[83]::Atadas masik PalyaElemnek, visszateres annak referenciajaval

Log: 9.utem -> PalyaElem[74]::Objektum atveve

Log: 9.utem -> PalyaElem[74]::Objektumok lekerdezese

Log: 9.utem -> FiuOvodas[2]::A kutya megugatott

Log: 9.utem -> FiuOvodas[2]::Menekulok

Log: 9.utem -> Kutya[0]::Mozgast befejezi

10.4.3 Óvodás meglátja a Kutyát

A teszt célja:

Az óvodás találkozik a kutyával, aki szintén észreveszi és meg kell hogy ugassa. Ekkor az óvodásnak menekülnie kell. Ha a találkozáskor az óvodás sorban állt, akkor ki kell szakadnia belőle.

Várt eredmény:

Ha találkoznak, a kutyának meg kell ugatnia, mire az óvodásnak menekülnie kell. Ha sorban állt, akkor egy kiszakadást is várunk.

Kapott eredmény:

Az előzőhöz hasonló az eset, csak itt egy lány találkozik össze a kutyával. A folyamat rendben lezajlik.

Naplófájl:

Log: 11.utem -> LanyOvodas[2]::Mozogni kezd
Log: 11.utem -> PalyaElem[40]::Atadas masik PalyaElemnek, visszateres annak referenciajaval
Log: 11.utem -> PalyaElem[50]::Objektum atveve
Log: 11.utem -> PalyaElem[50]::Objektumok lekerdezese
Log: 11.utem -> Kutya[0]::Ovodas udvozolt
Log: 11.utem -> LanyOvodas[2]::A kutya megugatott
Log: 11.utem -> LanyOvodas[2]::Menekulok
Log: 11.utem -> LanyOvodas[2]::Mozgast befejezi

10.4.4 Óvodás(lány) bámészkodik a csokiautomata előtt

A teszt célja:

Ha egy óvodás(lány) találkozik egy csokiautomatával, akkor ennek hatására ki kell szakadnia a sorból. Ellenben a fiú óvodásra nem szabad hatással legyen.

Várt eredmény:

Az óvodás lány üdvözli a csokiautomatát, az viszont üdvözli, mire a lány elbámészkodik.

Kapott eredmény:

A folyamat gond nélkül működött már elsőre.

Naplófájl:

Log: 1.utem -> LanyOvodas[0]::Mozogni kezd

Log: 1.utem -> PalyaElem[39]::Atadas masik PalyaElemnek, visszateres annak referenciajaval

Log: 1.utem -> PalyaElem[29]::Objektum atveve

Log: 1.utem -> PalyaElem[29]::Objektumok lekerdezese

Log: 1.utem -> Csokiautomata[1]::Ovodas udvozolt

Log: 1.utem -> LanyOvodas[0]::WOW, Csokiautomata!!! De szeeep!

Log: 1.utem -> LanyOvodas[0]::Bameszkodok

Log: 1.utem -> LanyOvodas[0]::Mozgast befejezi

10.4.5 Óvodás(fiú) bámészkodik a játékbolt előtt

A teszt célja:

Ha egy fiú óvodás találkozik egy játébolttal, akkor ennek hatására ha sorban állt, akkor ki kell szakadnia, a lányokra nem szabad hatással legyen.

Várt eredmény:

Ha egy fiú óvodás találkozik a játékbolttal, akkor ennek hatására bámészkodni fog, ez azt jelenti, hogy ha sorban állt akkor kifog szakadni bizonyos valószínűséggel.

Kapott eredmény:

A több szál miatt kuszának tűnik elsőre, de jól látható, hogy minden jól működik, ahogy kell.

Naplófájl:

Log: 6.utem -> FiuOvodas[2]::Mozogni kezd

Log: 6.utem -> PalyaElem[08]::Atadas masik PalyaElemnek, visszateres annak referenciajaval

Log: 6.utem -> PalyaElem[48]::Atadas masik PalyaElemnek, visszateres annak referenciajaval

Log: 6.utem -> PalyaElem[17]::Objektum atveve

Log: 6.utem -> PalyaElem[38]::Objektum atveve

Log: 6.utem -> PalyaElem[17]::Objektumok lekerdezese

Log: 6.utem -> PalyaElem[38]::Objektumok lekerdezese

Log: 6.utem -> LanyOvodas[2]::Mozogni kezd

Log: 6.utem -> Jatekbolt[2]::Ovodas udvozolt

Log: 6.utem -> LanyOvodas[0]::Mozgast befejezi

Log: 6.utem -> PalyaElem[44]::Atadas masik PalyaElemnek, visszateres annak referenciajaval

Log: 6.utem -> FiuOvodas[2]::WOW, Jatekbolt!!! Megnezem van-e G.I.Joe!

Log: 6.utem -> PalyaElem[45]::Objektum atveve

Log: 6.utem -> FiuOvodas[2]::Bameszkodok

Log: 6.utem -> PalyaElem[45]::Objektumok lekerdezese

Log: 6.utem -> FiuOvodas[2]::Mozgast befejezi

10.4.6 Csokifelvétel

A teszt célja:

Az óvónőnek az óvodások sorbafűzéshez csokira van szüksége. Mivel az óvónőnél, csak 5 csoki lehet, ezért az utánpótlásért a csokiautomatához mehet. Ennek a folyamatnak a helyességét vizsgáljuk.

Várt eredmény:

Az óvónő csokiszáma maximális lesz.

Kapott eredmény:

A várttal azonos eredmény, az egyik lány ovis nem is restelt egyből ellopni egyet az újonnan megszerzett csokik közül. Minden rendben történt.

Naplófájl:

Log: 4.utem -> Ovono::Mozogni kezd

Log: 4.utem -> PalyaElem[50]::Atadas masik PalyaElemnek, visszateres annak referenciajaval

Log: 4.utem -> PalyaElem[49]::Objektum atveve

Log: 4.utem -> PalyaElem[49]::Objektumok lekerdezese

Log: 4.utem -> Csokiautomata[3]::Ovono udvozolt

Log: 4.utem -> Ovono::Csokival feltoltoltodtem

Log: 4.utem -> LanyOvodas[0]::Ovono udvozolt

Log: 4.utem -> Ovono::Ovodas udvozolt

Log: 4.utem -> Ovono::Ovodast sorbaallitom

Log: 4.utem -> Ovono::CsokiSzam csokkentve 1-gyel

Log: 4.utem -> LanyOvodas[0]::Sorba allok

Log: 4.utem -> Ovono::Mozgast befejezi

10.4.7 Óvodás sorbaállítása (egynemű)

A teszt célja:

Egy óvodás csak akkor állhat sorba, ha a sorban csak vele megegyező nemű ovodások vannak.

Várt eredmény:

Az óvónő sorbaállítja az óvodást, utóbbi közli a lehetséges mögötte lévővel hogy eléállt, így a sor kibővül egy taggal és nem bomlik szét sehol.

Kapott eredmény:

Egymás mögé áll be két fiú ovis. Jól látható, hogy az óvónő először a 0-as IDjű ovist fogta sorba, aztán utóbb állt be az 1-es IDjű, és ahogy kell szólt is a már sorban állónak, hogy kicsit fáradjon hátrébb, tehát minden rendben ment.

Naplófájl:

Log: 0.utem -> Ovono::Ovodas udvozolt

Log: 0.utem -> Ovono::Ovodast sorbaallitom

Log: 0.utem -> Ovono::CsokiSzam csokkentve 1-gyel

Log: 0.utem -> LanyOvodas[0]::Sorba allok

.....

Log: 19.utem -> Ovono::Ovodas udvozolt

Log: 19.utem -> FiuOvodas[2]::Mozogni kezd

Log: 19.utem -> Ovono::Ovodast sorbaallitom

Log: 19.utem -> PalyaElem[88]::Atadas masik PalyaElemnek, visszateres annak referenciajaval

Log: 19.utem -> Ovono::CsokiSzam csokkentve 1-gyel

Log: 19.utem -> PalyaElem[88]:: !* Abban az iranyban nem mehetsz tovabb

Log: 19.utem -> FiuOvodas[1]::Sorba allok

Log: 19.utem -> FiuOvodas[2]::Mozgast befejezi

Log: 19.utem -> FiuOvodas[0]::Elem alltak

10.4.8 Óvodás sorbaállítása (fiú-lány)

A teszt célja:

Ha ellenkező neműekből áll a sor, akkor az új óvodás lesz a sor egyetlen tagja, mivel ellenkező nemű társai nem akarnak vele egy sorban lenni, ezért ilyenkor az aktuális sornak fel kell bomlania.

Várt eredmény:

Felbomlik a sor az ellenkező neműség miatt.

Kapott eredmény:

Hosszabb futtatások után sikerült összehoznia a véletlennek ezt a teljesen korrekt eredményt, a lány beáll a sorba a fiú ki áll onnan, és semmi gubanc.

Naplófájl:

```
Log: 20.utem -> LanyOvodas[0]::Mozogni kezd
Log: 20.utem -> PalyaElem[24]::Atadas masik PalyaElemnek, visszateres annak referenciajaval
Log: 20.utem -> PalyaElem[23]::Objektum atveve
Log: 20.utem -> PalyaElem[23]::Objektumok lekerdezese
Log: 20.utem -> Ovono::Ovodas udvozolt
Log: 20.utem -> Ovono::Ovodast sorbaallitom
Log: 20.utem -> Ovono::CsokiSzam csokkentve 1-gyel
Log: 20.utem -> LanyOvodas[0]::Sorba allok
Log: 20.utem -> FiuOvodas[0]::Elem allt egy ellenkezo nemu, ugyhogy lelepek
Log: 20.utem -> LanyOvodas[0]::Kileptek mogullem
Log: 20.utem -> LanyOvodas[0]::Mozgast befejezi
```

10.4.9 Óvodások leadása

A teszt célja:

Az óvónő csak itt szabad tudja leadni az óvodásokat, miután sorbaállította őket.

Várt eredmény:

Mikor az óvonő összegyűjtötte az óvodásokat azt az óvodában kell leadni, a leadott óvodások számával fog a számláló értéke növekedni. Ezután törlödnek a sórból az óvodások.

Kapott eredmény:

A véletlenszerűség adta nehézségek a 8. ütemben oldódtak meg, egy lány ovis ráköszönt az óvónőre, mire utóbbi sorba állította és le is adta az ovodába, amit kicsit idegesen fogadott, de belátta sorsát. Minden rendben zajlott, kidolgozás közben voltak problémák az ovisok közti kapcsolódások miatt, végtelen ciklusban hívták egymást mindenfelé, de egy-két kifelejetett feltétel beillesztésével megoldódott a gond.

Naplófájl:

```
Log: 8.utem -> Ovono::Mozogni kezd
Log: 8.utem -> PalyaElem[18]::Atadas masik PalyaElemnek, visszateres annak referenciajaval
Log: 8.utem -> PalyaElem[09]::Objektum atveve
Log: 8.utem -> PalyaElem[09]::Objektumok lekerdezese
Log: 8.utem -> Ovoda::Ovono udvozolt
Log: 8.utem -> Ovono::ElsoOvodas leadasa
Log: 8.utem -> Ovoda::Ovodas(ok) lancbol torolese
Log: 8.utem -> LanyOvodas[1]::Mozogni kezd
Log: 8.utem -> PalyaElem[08]::Atadas masik PalyaElemnek, visszateres annak referenciajaval
Log: 8.utem -> PalyaElem[08]:: !* Abban az iranyban nem mehetsz tovabb
Log: 8.utem -> LanyOvodas[1]::Mozgast befejezi
Log: 8.utem -> LanyOvodas[1]::Mozogni kezd
Log: 8.utem -> PalyaElem[08]::Atadas masik PalyaElemnek, visszateres annak referenciajaval
Log: 8.utem -> PalyaElem[09]::Objektum atveve
Log: 8.utem -> PalyaElem[09]::Objektumok lekerdezese
Log: 8.utem -> Ovono::Ovodas udvozolt
Log: 8.utem -> LanyOvodas[1]::Mozgast befejezi
Log: 8.utem -> LanyOvodas[1]::Hagyjatok beken!!!
```

Log: 8.utem -> LanyOvodas[1]::Ovodaba kerultem, ezzel vege eletemnek

Log: 8.utem -> Szamlalo::Eredmeny megnovelve -> +1

10.4.10 Számláló növelése

A teszt célja:

Mikor az óvónő bevitte az óvodásokat az óvodába, a számlálónak növekednie kell annak függvényében, hogy hány óvodást tudott leadni az óvónő. A játék végén a Toplistának le kell kérdeznie az értékét, s eszerint kell majd bekerüljön a játékos eredménye a toplistába.

Várt eredmény:

Növekedik a számláló értéke a bevitt óvodások számával arányosan.

Kapott eredmény:

Az előzővel megegyezően szerencsés eset, és a véletlen jól látszik, hiszen most egy fiú ovis akadt az óvónő útjába, és meg is kapta érte a +1 pontját, rendben ahogy kell.

Naplófájl:

Log: 8.utem -> Ovono::Mozogni kezd

Log: 8.utem -> PalyaElem[18]::Atadas masik PalyaElemnek, visszateres annak referenciajaval

Log: 8.utem -> PalyaElem[09]::Objektum atveve

Log: 8.utem -> PalyaElem[09]::Objektumok lekerdezese

Log: 8.utem -> Ovoda::Ovono udvozolt

Log: 8.utem -> Ovono::ElsoOvodas leadasa

Log: 8.utem -> Ovoda::Ovodas(ok) lancbol torolese

Log: 8.utem -> FiuOvodas[0]::Mozogni kezd

Log: 8.utem -> PalyaElem[17]::Atadas masik PalyaElemnek, visszateres annak referenciajaval

Log: 8.utem -> PalyaElem[08]::Objektum atveve

Log: 8.utem -> PalyaElem[08]::Objektumok lekerdezese

Log: 8.utem -> FiuOvodas[0]::Mozgast befejezi

Log: 8.utem -> FiuOvodas[0]::Mozogni kezd

Log: 8.utem -> PalyaElem[08]::Atadas masik PalyaElemnek, visszateres annak referenciajaval

Log: 8.utem -> PalyaElem[09]::Objektum atveve

Log: 8.utem -> PalyaElem[09]::Objektumok lekerdezese

Log: 8.utem -> Ovono::Ovodas udvozolt

Log: 8.utem -> Ovono::Ovodast sorbaallitom

Log: 8.utem -> Ovono::CsokiSzam csokkentve 1-gyel

Log: 8.utem -> FiuOvodas[0]:: !* Mar sorba allok *!

Log: 8.utem -> FiuOvodas[0]::Mozgast befejezi

Log: 8.utem -> FiuOvodas[0]::Hagyjatok beken!!!

Log: 8.utem -> FiuOvodas[0]::Ovodaba kerultem, ezzel vege eletemnek

Log: 8.utem -> Szamlalo::Eredmeny megnovelve -> +1

10.4.11 Toplista megtekintése a menüből

A teszt célja:

A menüből kiválasztva meg kell hogy tekinthessük a toplistát, mely a legjobb eredményeket kell hogy tárolja, amelyeket egy fájlból kell beolvasson.

Várt eredmény:

A toplista beolvassa a fájlból az eredményeket, egy listát feltölt vele, majd ezt megjeleníti.

Kapott eredmény:

A végleges verzióban grafikusan fog megjelenni, így nem sokat foglalkoztam a kinézetével. A teszt bebizonyította hogy jól működik a fájlbeolvasás és a rendezés is, bár utóbbi eleinte kevésszer futott le, így nem adott teljes sorrendezést, egy két plusz kör a ciklusban megoldotta a problémát. A rendezés alapgondolata: buborékos rendezés.

Naplófájl:

Log: 0.utem -> Jatek::EredmenytNez

Log: 0.utem -> Toplista::Megjelenit()

Log: 0.utem -> Lista::Eredmenyek -> 1. XXxXXxxXxx P: 99 Sz: 4

Log: 0.utem -> Lista::Eredmenyek -> 2. Mr. X P: 67 Sz: 3

Log: 0.utem -> Lista::Eredmenyek -> 3. Tommey P: 45 Sz: 1

Log: 0.utem -> Lista::Eredmenyek -> 4. Tommey P: 30 Sz: 1

Log: 0.utem -> Lista::Eredmenyek -> 5. Mr. T P: 15 Sz: 3

Log: 0.utem -> Lista::Eredmenyek -> 6. Sanyi P: 12 Sz: 2

Log: 0.utem -> Toplista::Megjelenit() VEGE

10.4.12 Toplista megtekintése a játék végén

A teszt célja:

A játék idő leteltével lehetőségünk kell legyen arra, hogy megtekintsük a toplistát, mely a legjobb eredményeket kell, hogy eltárolja egy fájlban, illetve a játék folyamán egy listában.

Várt eredmény:

Egy toplistát tartalmazó fájl, melyet sikeresen le tudunk menteni és megjeleníteni.

Kapott eredmény:

Megfelel az elvárásoknak, és szándékosan van kétszer rendezve, és az utolsó 3 eredmény is azért azonos mivel már harmadjára futott le a teszt, nincs vele különösebb gond, ami volt azt már az előző pontban kifejtettem.

Naplófájl:

Log: 44.utem -> Ido::Az ido letelt

Log: 44.utem -> Ido::Ido vegehez ert-e? -> igen

Log: 44.utem -> Toplista::Megjelenit()

Log: 44.utem -> Szamlalo::Eredmeny lekerdezes -> 0

Log: 44.utem -> Lista::Uj elem felveve

Log: 44.utem -> Lista::Rendezve

Log: 44.utem -> Lista::Rendezve

Log: 44.utem -> Lista::Eredmenyek -> 1. XXxXXxxXxx P: 99 Sz: 4

Log: 44.utem -> Lista::Eredmenyek -> 2. Mr. X P: 67 Sz: 3

Log: 44.utem -> Lista::Eredmenyek -> 3. Tommey P: 45 Sz: 1

Log: 44.utem -> Lista::Eredmenyek -> 4. Tommey P: 30 Sz: 1

Log: 44.utem -> Lista::Eredmenyek -> 5. Mr. T P: 15 Sz: 3

Log: 44.utem -> Lista::Eredmenyek -> 6. Sanyi P: 12 Sz: 2

Log: 44.utem -> Lista::Eredmenyek -> 7. Tommey P: 0 Sz: 0

Log: 44.utem -> Lista::Eredmenyek -> 8. Tommey P: 0 Sz: 0

Log: 44.utem -> Lista::Eredmenyek -> 9. Tommey P: 0 Sz: 0

Log: 44.utem -> Toplista::Megjelenit() VEGE

10.4.13: Szint újrajátszása

A teszt célja:

Amikor a játékidő végére érünk, választanunk kell, hogy a toplista megtekintése után kilépünk, vagy újra játszunk ugyanazon a nehézségi szinten. A játék újbóli betöltését vizsgáljuk.

Várt eredmény:

Vagy megjelenik a toplista, vagy pedig sikeresen be kell hogy töltődjön a játék.

Kapott eredmény:

Bemeneti fájllal elég körülményes megoldani a tesztet, mivel az óránk igazi időt mér, nincs a teszthez igazítva, így nehéz eltalálni, hogy meddig kell menni és hol jön pont az a bekérés, ahol máshogy kéne válaszolni, a test13.out, és test13.log fájl tehát egy úgymond manuális úton keletkeztek, de látható, tökéletesen működik mind a toplista megjelenítése, mind az új játék kezdése, illetve ugyanaz játszása.

Naplófájl:

Log: 11.utem -> Ido::Ido vegehez ert-e? -> igen

Log: 11.utem -> Toplista::Megjelenit()

Log: 11.utem -> Szamlalo::Eredmeny lekerdezes -> 0

Log: 11.utem -> Lista::Uj elem felveve

Log: 11.utem -> Lista::Rendezve

Log: 11.utem -> Lista::Rendezve

Log: 11.utem -> Lista::Eredmenyek -> 1. XXxXXxxXxx P: 99 Sz: 4

Log: 11.utem -> Lista::Eredmenyek -> 2. Mr. X P: 67 Sz: 3

Log: 11.utem -> Lista::Eredmenyek -> 3. Tommey P: 45 Sz: 1

Log: 11.utem -> Lista::Eredmenyek -> 4. Tommey P: 30 Sz: 1

Log: 11.utem -> Lista::Eredmenyek -> 5. Mr. T P: 15 Sz: 3

Log: 11.utem -> Lista::Eredmenyek -> 6. Sanyi P: 12 Sz: 2

Log: 11.utem -> Lista::Eredmenyek -> 7. Tommey P: 0 Sz: 0

```
Log: 11.utem -> Lista::Eredmenyek -> 8. Tommey      P: 0 Sz: 0
Log: 11.utem -> Lista::Eredmenyek -> 9. Tommey      P: 0 Sz: 0
Log: 11.utem -> Lista::Eredmenyek -> 10. Thomas     P: 0 Sz: 0
Log: 11.utem -> Lista::Eredmenyek -> 11. Tom        P: 0 Sz: 0
Log: 11.utem -> Toplista::Megjelenit() VEGE
.....................................
Log: 12.utem -> Ido::Ujrainditva
Log: 12.utem -> Palya::Palya epites elkezdve
Log: 12.utem -> Palya::PalyaElem-ek letrehozasa
Log: 12.utem -> PalyaElem[95]::Letrehozva
Log: 12.utem -> PalyaElem[96]::Letrehozva
.....................................
Log: 25.utem -> Ido::Ido vegehez ert-e? -> igen
Log: 25.utem -> Toplista::Megjelenit()
Log: 25.utem -> Szamlalo::Eredmeny lekerdezes -> 1
Log: 25.utem -> Lista::Uj elem felveve
Log: 25.utem -> Lista::Rendezve
Log: 25.utem -> Lista::Rendezve
Log: 25.utem -> Lista::Eredmenyek -> 1. XXxXXxxXxx  P: 99 Sz:
4
Log: 25.utem -> Lista::Eredmenyek -> 2. Mr. X       P: 67 Sz: 3
Log: 25.utem -> Lista::Eredmenyek -> 3. Tommey      P: 45 Sz: 1
Log: 25.utem -> Lista::Eredmenyek -> 4. Tommey      P: 30 Sz: 1
Log: 25.utem -> Lista::Eredmenyek -> 5. Mr. T       P: 15 Sz: 3
Log: 25.utem -> Lista::Eredmenyek -> 6. Sanyi       P: 12 Sz: 2
Log: 25.utem -> Lista::Eredmenyek -> 7. Tommey      P: 0 Sz: 0
Log: 25.utem -> Lista::Eredmenyek -> 8. Tommey      P: 0 Sz: 0
Log: 25.utem -> Lista::Eredmenyek -> 9. Tommey      P: 0 Sz: 0
Log: 25.utem -> Lista::Eredmenyek -> 10. Tom        P: 1 Sz: 0
Log: 25.utem -> Lista::Eredmenyek -> 11. Thomas     P: 0 Sz: 0
Log: 25.utem -> Lista::Eredmenyek -> 12. Tom        P: 0 Sz: 0
Log: 25.utem -> Toplista::Megjelenit() VEGE
.....................................
Log: 26.utem -> Jatek vege, kilepes, Viszlat!
```

10.4.14 Idő lejár

A teszt célja:

Amikor az idő lejár, akkor véget kell érjen a játék és a rendszernek le kell ellenőriznie a számláló értéke alapján, hogy felkerülünk-e a toplistára vagy sem.

Várt eredmény:

Mikor letelik az idő, akkor a játék megszakad, és felkerülünk a toplistára ha a számláló értéke nagyon a legkisebb toplistás értéknél.

Kapott eredmény:

Bemeneti fájlból való tesztelés során ezt az eredményt látjuk, mikor letelik az idő, normál játék közben lehet választani a toplista, illetve új játék kezdése közül is a kilépésen kívül, tehát tulajdonképpen ez is jól működik, feltételezve hogy a kész játékot nem input file-okkal akarja valaki játszani.

Naplófájl:

```
Log: 44.utem -> Ido::Az ido letelt
Log: 44.utem -> Szamlalo::Eredmeny lekerdezes -> 0
Log: 44.utem -> Ido::Ido vegehez ert-e? -> igen
Log: 45.utem -> Jatek vege, kilepes, Viszlat!
```

10.5 Értékelés

Mindenki kellő részt vállalt a munkából a félév során eddig, így hármunk között oszlik el a 100% teljesítmény.

11. Grafikus felület specifikálása

11.1. Menürendszer, a kezelői felület grafikus képe

A játék egy ablakot fog használni, melyben legfelül egy menüsor található, alatta pedig a játék elemeit grafikusan megjelenítő játéktér és egy a játékosnak szóló üzeneteket tartalmazó fejléc. A játék egy egyszerű menüből és billentyűzet, illetve egér segítségével vezérelhető, mely a játék folyamán állandóan elérhető lesz. A menüben a use case-eknek megfelelően a következő menüpontok szerepelnek majd:

Új játék

A játékos bármikor elkezdheti a játékot a kiválasztott szintről indulva, aktuális pontjai innentől kezdődve számlálódnak.

Szint újrakezdése

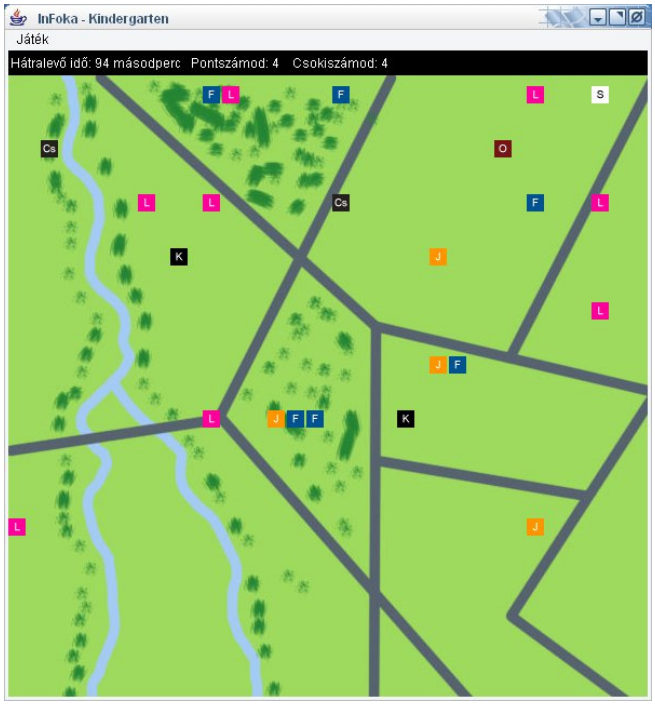
A játékos bármikor újrakezdheti az aktuálisan játszott szintet, ekkor eddig szerzett pontjai elvesznek.

Toplista

A játék során bármikor lekérheti a játékos az eddigi legjobb eredményeket. Míg a játékos az eredményeket nézi, a játék áll. Amennyiben a játékos eddigi eredményei alapján bekerül a legjobbak közé, akkor az adatok módosítása ekkor történik meg.

Kilépés

A játékból való kilépésre szolgáló menüpont.



1. ábra: Egy pálya előzetes kinézete; rajta minden objektumból legalább 1 található

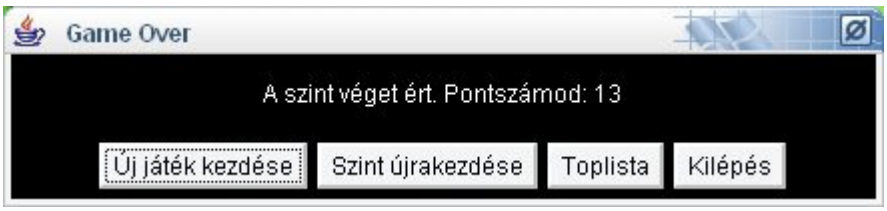
A játékosnak szóló információk a játéktér fejlécében jelennek meg. Ezek: az aktuális szinten elért pontszám, a szint teljesítésének hátralévő ideje, az óvónő csokiszáma. Ez még leendő pálya alapképe a játékos és a többi objektum képei a közeljövőben lesznek kidolgozva.

További ablakokat is használunk: a játékos nevének bekérésére, a nehézségi szint kiválasztására, a toplista kiíratására és a szint végén a további műveletek bekérésére.

2. ábra: A név bekéréséhez használt ablak

3. ábra: A nehézség kiválasztását szolgáló ablak

4. ábra: A toplista megjelenítését szolgáló ablak



5. ábra: A szintvége, azaz a játék végén előjövő ablak, amiből kiválaszthatjuk, hogyan tovább

A grafikus felület a Java AWT csomagját fogja használni.

11.2. A felület működésének elve, a grafikus rendszer architektúrája (struktúra diagramok)

A grafikus megvalósítás lényege a következő: a modellben szereplő objektumok önállóan változtatják állapotukat: egy adott időbeli felbontással megváltozik az objektumok helyzete, frissül a hátralevő idő, stb. A modelltől függetlenül működik a megjelenésért felelős objektum, melynek feladata, hogy kellően sűrű időközzel lekérdezze az objektumok állapotát, és ennek megfelelően elvégezze a megjelenítéssel kapcsolatos feladatokat. A viszonylag kevés számú és egyszerű esemény kezelése nem igényelte külön osztály létrehozását e feladat megoldására. Ezért az eseménykezelés is a Grafika osztályban található. Egy-egy esemény közvetlenül a modell egy objektumának metódusát fogja meghívni. Törekszünk arra, hogy a modell és a grafikus megjelenítést elvégző réteg között a lehető legkisebb legyen a csatolás, ennek részeként az egyik cél az, hogy a modellt a lehető legkisebb mértékben változtassuk meg. A prototípushoz képest a következő változtatásokra mégis szükség van:

" A grafikus megjelenítésért felelős új osztály - Grafika - létrehozása. A modell által szolgáltatott adatokat leképezi egy megjeleníthető grafikus felületre.

1. ábra: A grafikus objektumokat tartalmazó osztálydiagram

11.3. A grafikus objektumok felsorolása, kapcsolatuk az alkalmazói rendszerrel (szekvencia diagramok)

11.3.1. Grafikus elemek

Grafika osztály: a java.awt.Frame osztály leszármazottja. A grafikus megjelenítés ennek az osztálynak a felelőssége. Ez tartalmazza a játékteret a fejléccel együtt. Itt található a menü is. Ezen kívül ez az osztály kezeli az eseményeket is.

LegjobbakAblak osztály: szintén a java.awt.Frame osztály leszármazottja. A legjobb eredményeket elért játékosok nevét és eredményét tüntetjük fel rajta.

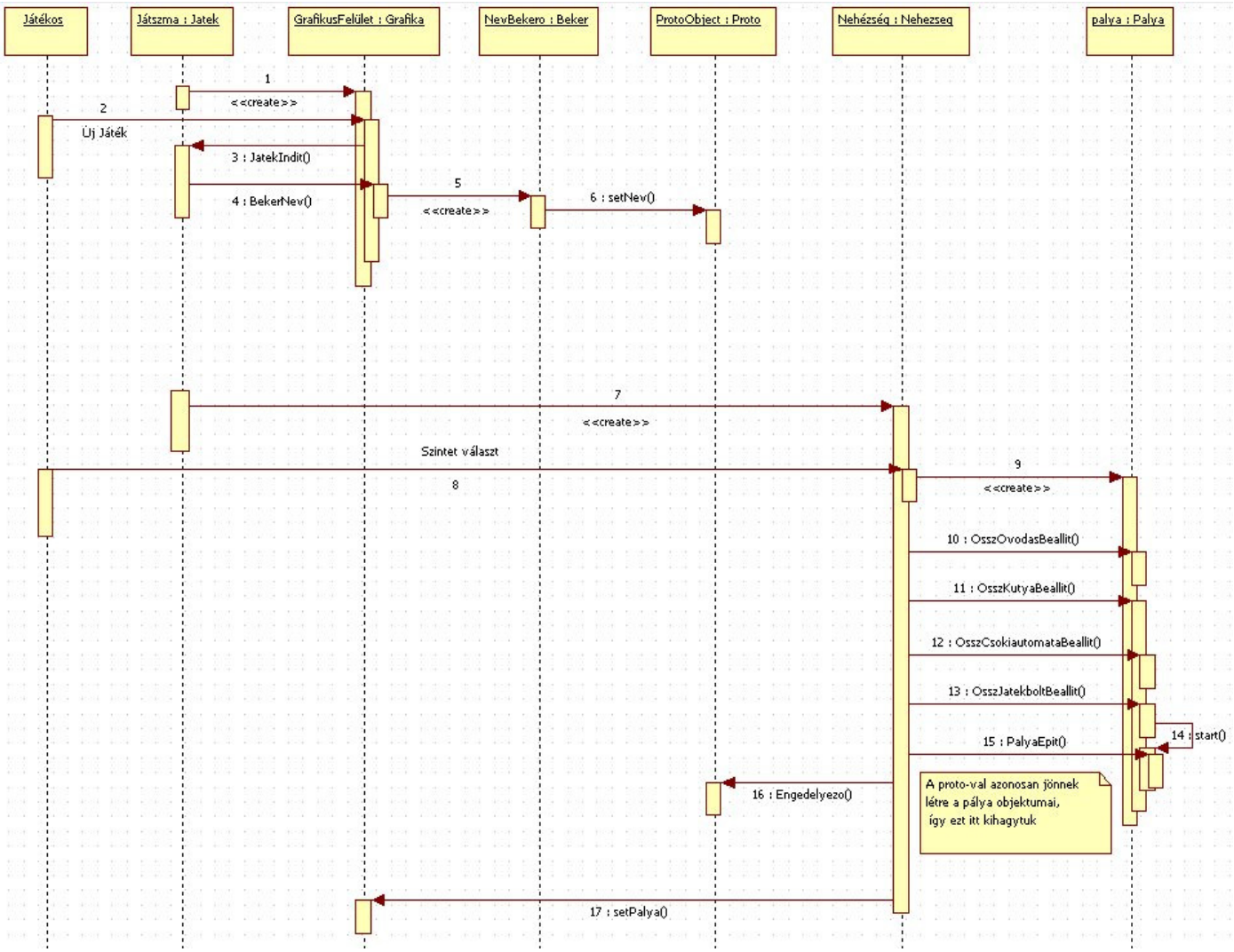
SzintVegeAblak osztály: ez is egy Frame, amikor egy szint véget ér, a játékos dönthet, hogy tovább lép, újrajátsza a szintet esetleg a legjobb eredményeket tekinti meg. A lehetőségek közül ezen az osztályon keresztül tud a felhasználó választani.

Idozito osztály: a Grafika osztály használja fel a szolgáltatásait: adott időközönként lekéri a modell objektumok adatait és frissíti a képet.

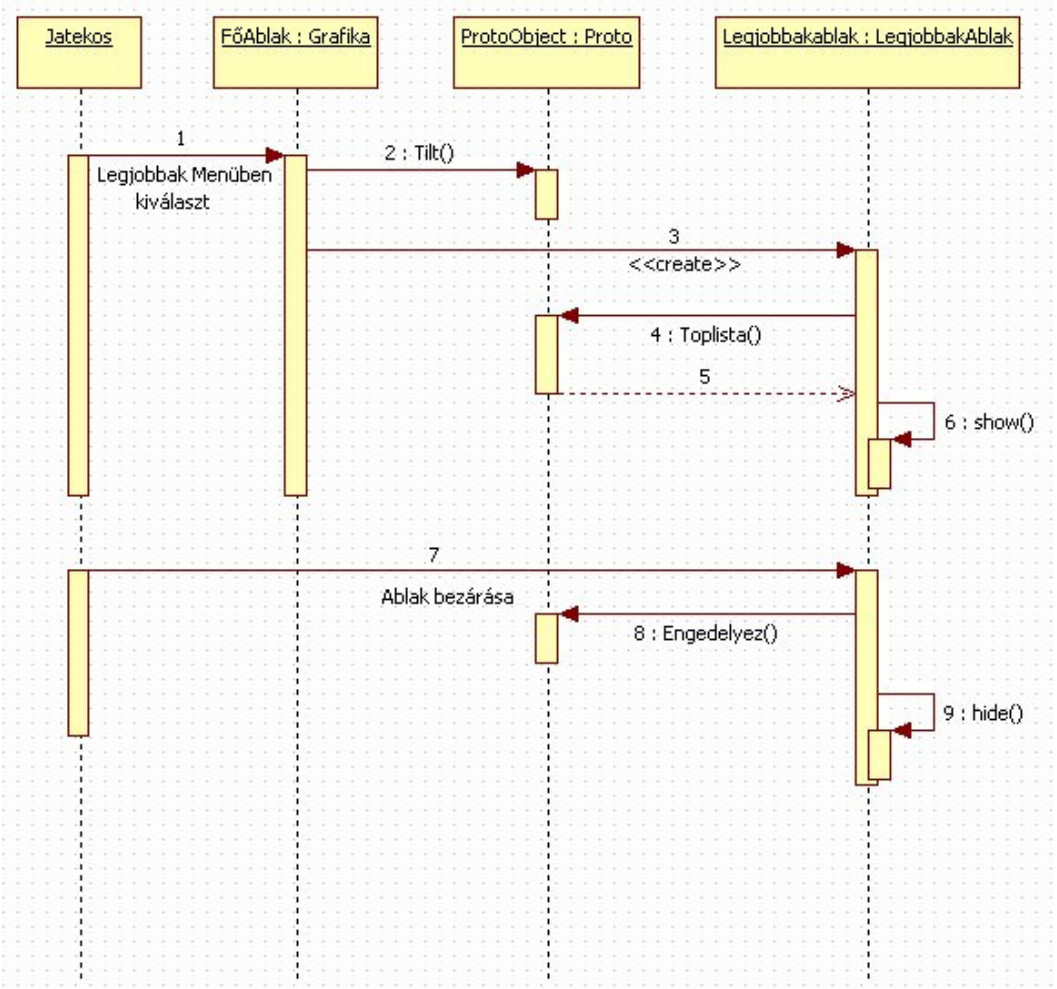
Nehezseg osztály: ez is egy Frame leszármazott, új játék kezdésénél jeleníti meg a választási lehetőségeket [Könnyű/Közepes/Nehéz], majd indítja a játékot.

11.3.2 Grafikus objektumok és az alkalmazói rendszer kapcsolata

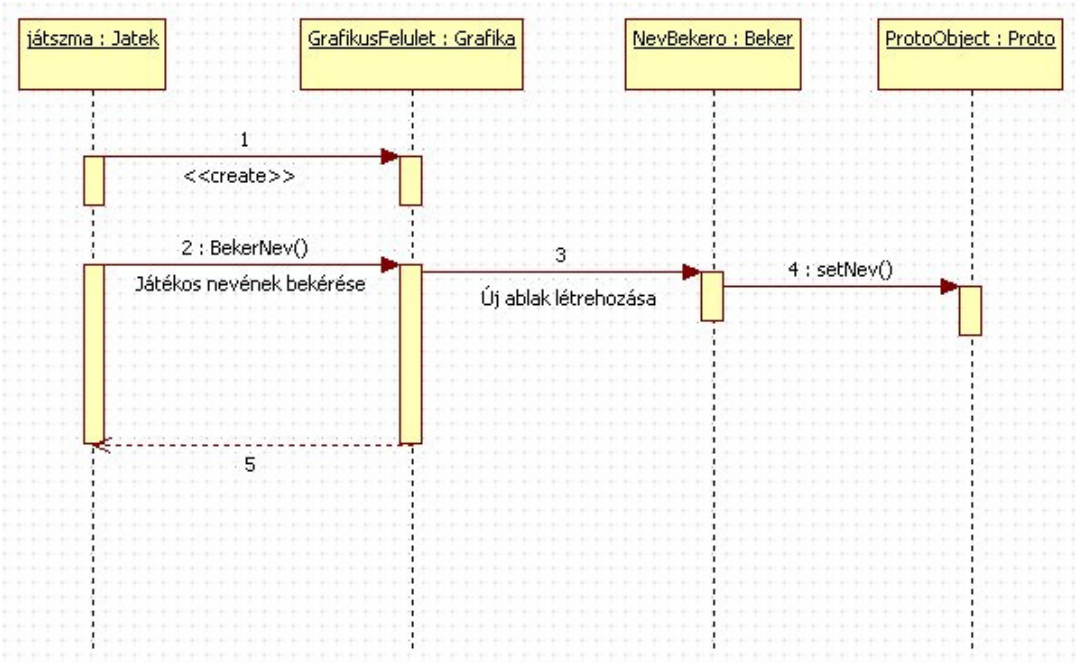
Az alábbi szekvenciadiagramon a játék elindulása követhető nyomon. Azonban hiányoznak bizonyos részek, melyek a prototípusban részletesen láthatók. Itt próbáltunk a grafikus objektumokat érintő üzenetekre koncentrálni. A játszma objektum létrehozza a grafika objektumot. Játék indításakor a játszma bekéri a nevet, és létrehoz egy nehézség objektumot, amely a kiválasztás után létrehozza a pályát, és beállítja az objektumszámokat és elindítja a pályaépítést. A pálya létrehozza az objektumokat. Az alap objektumok elindítja a pálya szálját. Ez a szál felelős a játék vége észlelésében. A nehézség összerendeli a pálya-t és a grafika-t a Grafika setPalya metódusával. Ez a metódus hozza létre és indítja el az idozitot, mely a modell objektumoktól kéri le azok állapotait, majd frissíti a képet a grafika segítségével. A szint végeztével ismét a pálya aktivizálódik. A felhasználó döntésétől függően üzen a pálya-nak, vagy a játszma-nak az aktuális szint újrakezdésére, új játék kezdésére; a toplista megtekintése vagy kilépés. Az alábbi diagramon mind az inicializálás (grafikus felülethez kapcsolódóan) mind pedig a kirajzolással kapcsolatos üzenetváltások megfigyelhetők.



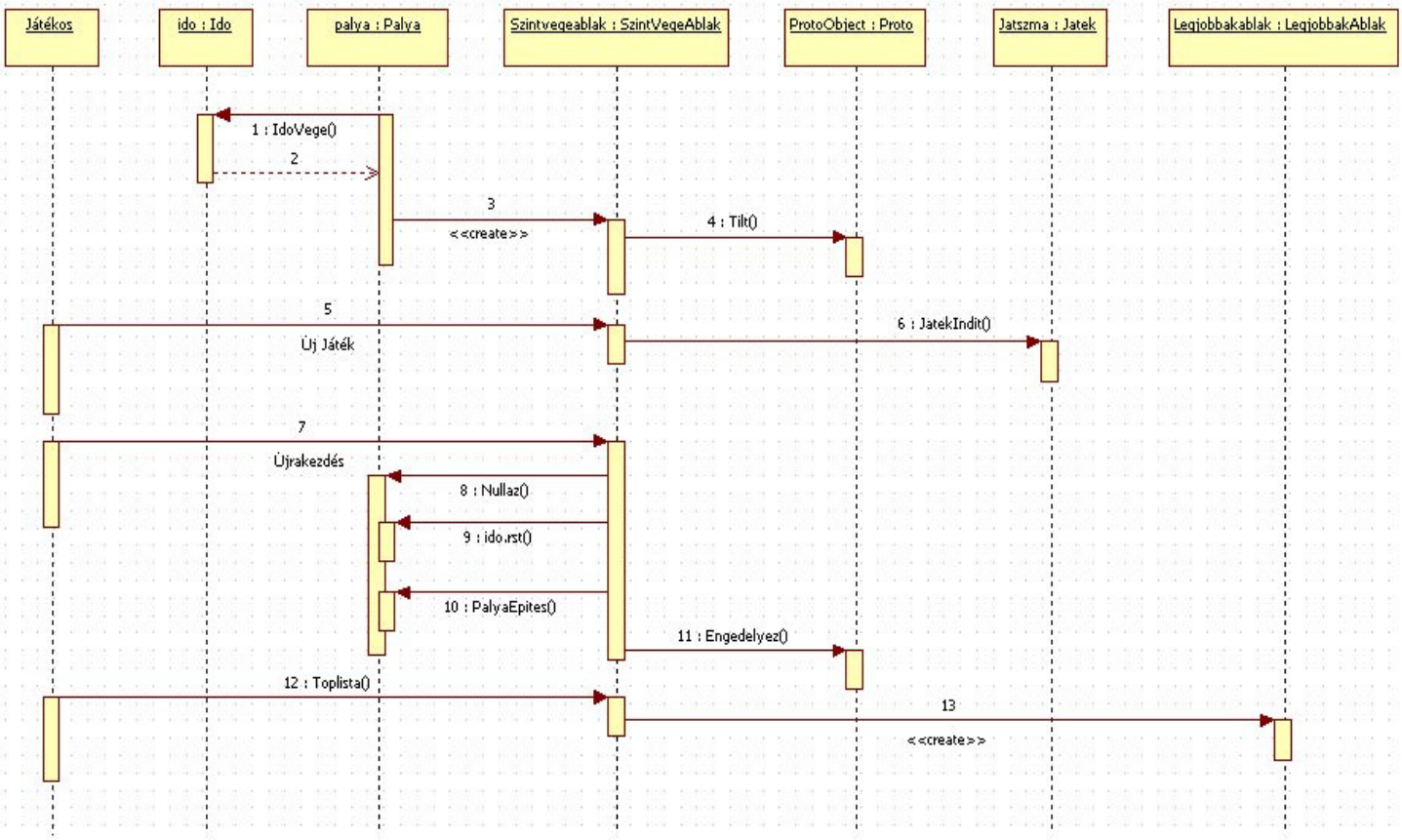
Az alábbiakban az látható, amint a játékos lekéri az eddigi legjobb eredményeket. A menüből kiválasztva a megfelelő menüpontot a Grafika osztály kezeli le az eseményt. Létrehoz egy LegjobbakAblak típusú objektumot, mely a főprogramból lekéri az adatokat. Majd ezeket megjeleníti.



Az előzőekhez hasonló módon zajlik le a játékos nevének bekérése. A kezdeményező azonban maga a játszma. Miután létrehozta a Grafika típusú objektumot, elindítja a névbekérést a BekerNev() metódus segítségével. A grafika ekkor létrehoz egy Beker típusú objektumot, mely valójában egy ablak megjelenését eredményezi. Addig nem folytatódhat a végrehajtás a játszmában, míg a játékos meg nem adta a nevét. Ha azonban ez megtörtént, a grafika-n keresztül ismét a játszmához kerül az irányítás.



Ha a játék közben lejárt az idő, akkor a játékos dönthet hogyan folytatja játékot. Azt hogy az idő lejárt, a pálya észleli másodpercenként lekérdezve az ido objektum IdoVege() függvényét, a prototípusban alkalmazott megoldást használva nem engedélyezi a szálak működését, majd létrehoz egy SzintVegeAblak típusú objektumot. Ez az előző esetekhez hasonlóan a játékos számára csupán egy ablak megjelenését jelenti. A rendszer ekkor felhasználói inputra vár: hogyan folytatódjon a játék. Az inputtól függően a SzintVegeAblak üzen a pálya-nak vagy játszma-nak, ami leállítja a szálak futását, hogy új szint indításakor azok már ne működjenek, majd a Pálya üzenetet küld a jateknak. Ekkor a jatek a felhasználó döntésétől függően ha kell, betölti a kiválasztott szintet vagy kilép.


Játékos
ido : Ido
palya : Palya
Szintvegeablak : SzintVegeAblak
ProtoObject : Proto
Jatszma : Jatek
Legjobbakablak : LegjobbakAblak
1 : IdoVege()
2
3
<<create>>
4 : Tilt()
5
Új Játék
6 : JatekIndit()
7
Újrakezdés
8 : Nullaz()
9 : ido.rst()
10 : PalyaEpites()
11 : Engedelyez()
12 : Toplista()
13
<<create>>

13. Grafikus változat beadása

13.1. Útmutatás

A játék elindítása:

A fő könyvtárba található compile.bat segítségével tudjuk lefordítani a forrás fileokat, a run.bat segítségével automatikusan le is fordítjuk, illetve el is indítjuk a játékot, a doc.bat a javadocumentációt készíti el.

A játék irányítása:

A játék alatt egy óvónőt kell irányítani, hogy összeszedjük az elkóborolt óvodásokat, segítségünkre vannak a csokik, az ellenségeink a különböző játékboltok, kutyák, csokiautomaták. Az óvonő irányítása kétféleképpen lehetséges: vagy a numerikus billentyűzetet használjuk a 6 iránynak megfelelően a 741963 gombokkal, vagy a bal kezesek számára lehetőség van a 'qayedc' gombokkal történő irányításra.

A beküldött zip file tartalma:

A végső beküldött project tartalma: A javadoc nevű könyvtár tartalmazza a java dokumentációt az elkészült projectről, a Kindergarten nevű könyvtár tartalmazza a forrásfileokat, bővebben:

2006.05.15. 12:50	<DIR>	.
2006.05.15. 12:50	<DIR>	..
2006.08.15. 01:38	<DIR>	javadoc
2006.08.15. 01:38	<DIR>	Kindergarten
2006.05.01. 21:22	57	compile.bat
2006.08.15. 01:38	38	doc.bat
2006.05.01. 21:22	124	run.bat
	3 fájl	219 bájt

A /javadoc tartalma:

2006.05.15. 12:50	<DIR>	.
2006.05.15. 12:50	<DIR>	..
2006.08.15. 01:38	<DIR>	Kindergarten
2006.08.15. 01:38	<DIR>	resources
2006.08.15. 01:38	2 910	allclasses-frame.html
2006.08.15. 01:38	2 470	allclasses-noframe.html
2006.08.15. 01:38	4 599	constant-values.html
2006.08.15. 01:38	4 483	deprecated-list.html
2006.08.15. 01:38	7 717	help-doc.html
2006.08.15. 01:38	68 207	index-all.html
2006.08.15. 01:38	827	index.html
2006.08.15. 01:38	8 816	overview-tree.html
2006.08.15. 01:38	14	package-list
2006.08.15. 01:38	709	packages.html
2006.08.15. 01:38	9 454	serialized-form.html
2006.08.15. 01:38	1 231	stylesheet.css
	12 fájl	111 437 bájt

A /javadoc/Kindergarten tartalma, az osztályokról generált html fileok:

2006.05.15. 12:50	<DIR>	.
2006.05.15. 12:50	<DIR>	..
2006.08.15. 01:38	20 027	Csokiautomata.html
2006.08.15. 01:38	16 939	FiuOvodas.html
2006.08.15. 01:38	14 100	GPS.html
2006.08.15. 01:38	22 148	Grafika.html
2006.08.15. 01:38	13 711	Ido.html
2006.08.15. 01:38	14 776	Jatek.html
2006.08.15. 01:38	17 950	Jatekbolt.html
2006.08.15. 01:38	22 113	KozosOs.html
2006.08.15. 01:38	20 220	Kutya.html
2006.08.15. 01:38	16 977	LanyOvodas.html

2006.08.15. 01:38	17 586	Lista.html
2006.08.15. 01:38	10 943	Lista.ListaElem.html
2006.08.15. 01:38	7 786	Mozgat.html
2006.08.15. 01:38	21 914	Nehezseg.html
2006.08.15. 01:38	20 524	Ovoda.html
2006.08.15. 01:38	34 062	Ovodas.html
2006.08.15. 01:38	24 713	Ovono.html
2006.08.15. 01:38	2 996	package-frame.html
2006.08.15. 01:38	12 196	package-summary.html
2006.08.15. 01:38	8 940	package-tree.html
2006.08.15. 01:38	24 355	Palya.html
2006.08.15. 01:38	19 731	PalyaElem.html
2006.08.15. 01:38	24 539	Proto.html
2006.08.15. 01:38	12 930	Szamlalo.html
2006.08.15. 01:38	18 411	SzintVegeAblak.html
2006.08.15. 01:38	12 451	Toplista.html
	26 fájl	453 038 bájt

A /Kindergarten tartalma a java fileok:

2006.05.15. 12:50	<DIR>	.
2006.05.15. 12:50	<DIR>	..
2006.08.15. 00:47	<DIR>	grafika
2006.08.15. 00:57	<DIR>	Kindergarten
2006.05.07. 07:51	250	ActionAdapter.java
2006.08.14. 17:04	1 706	Beker.java
2006.08.14. 01:03	2 104	Csokiautomata.java
2006.05.07. 07:51	853	FiuOvodas.java
2006.08.14. 17:19	3 040	GPS.java
2006.08.15. 00:47	14 367	Grafika.java
2006.08.14. 10:32	1 501	Ido.java
2006.05.07. 07:52	1 648	Jatek.java
2006.08.14. 01:05	1 475	Jatekbolt.java
2006.08.14. 01:09	2 449	KozosOs.java

2006.08.14. 10:34	2 259	Kutya.java
2006.05.07. 07:52	859	LanyOvodas.java
2006.08.14. 17:25	2 016	LegjobbakAblak.java
2006.08.14. 17:24	4 552	Lista.java
2006.05.07. 07:52	298	Mozgat.java
2006.08.14. 21:14	3 282	Nehezseg.java
2006.08.14. 01:42	2 748	Ovoda.java
2006.08.15. 00:57	6 360	Ovodas.java
2006.08.14. 21:13	3 897	Ovono.java
2006.08.14. 21:21	10 710	Palya.java
2006.08.14. 19:25	3 683	PalyaElem.java
2006.05.07. 07:52	5 141	Proto.java
2006.08.14. 17:31	1 312	Szamlalo.java
2006.05.10. 00:30	3 096	SzintVegeAblak.java
2006.08.15. 01:37	177	toplista
2006.05.07. 07:52	1 393	Toplista.java
	26 fájl	81 176 bájt

A /Kindergarten/grafika tartalma a játékos során elpfordulő „szereplők” képei:

2006.05.15. 12:50	<DIR>	.
2006.05.15. 12:50	<DIR>	..
2006.08.15. 00:20	1 647	csokiautomata.gif
2006.08.14. 11:17	1 052	fiuovis_0.gif
2006.08.14. 11:21	1 043	fiuovis_1.gif
2006.08.14. 11:19	1 042	fiuovis_2.gif
2006.08.14. 11:18	1 053	fiuovis_3.gif
2006.08.14. 11:18	1 042	fiuovis_4.gif
2006.08.14. 11:20	1 045	fiuovis_5.gif
2006.08.15. 00:21	1 706	jatekbolt.gif
2006.08.15. 00:29	573	kutya_0.gif
2006.08.15. 00:29	572	kutya_1.gif
2006.08.15. 00:29	573	kutya_2.gif
2006.08.15. 00:29	574	kutya_3.gif

2006.08.15. 00:29	568	kutya_4.gif
2006.08.15. 00:29	579	kutya_5.gif
2006.08.14. 11:12	1 049	lanyovis_0.gif
2006.08.14. 11:16	1 048	lanyovis_1.gif
2006.08.14. 11:14	1 050	lanyovis_2.gif
2006.08.14. 11:12	1 051	lanyovis_3.gif
2006.08.14. 11:13	1 046	lanyovis_4.gif
2006.08.14. 11:15	1 041	lanyovis_5.gif
2006.08.14. 11:09	1 310	ovoda.gif
2006.08.14. 11:04	1 055	ovono_0.gif
2006.08.14. 11:07	1 082	ovono_1.gif
2006.08.14. 11:03	1 083	ovono_2.gif
2006.08.14. 11:04	1 055	ovono_3.gif
2006.08.14. 11:02	1 086	ovono_4.gif
2006.08.14. 11:05	1 087	ovono_5.gif
2006.08.15. 00:46	57 329	palya_0.jpg
2006.08.15. 00:41	135 601	palya_1.jpg
2006.08.15. 00:45	214 235	palya_2.jpg
2006.05.01. 23:14	101 386	ws.jpg
	31 fájl	535 663 bájt

A /Kindergarten/Kindergarten tartalma a lefordított class fileok:

2006.05.15. 12:50	<DIR>	.
2006.05.15. 12:50	<DIR>	..
2006.08.15. 00:57	341	ActionAdapter.class
2006.08.15. 00:57	483	Beker$1.class
2006.08.15. 00:57	781	Beker$MyActionListener.class
2006.08.15. 00:57	1 537	Beker.class
2006.08.15. 00:57	1 761	Csokiautomata.class
2006.08.15. 00:57	817	FiuOvodas.class
2006.08.15. 00:57	1 526	GPS.class
2006.08.15. 00:57	491	Grafika$1.class
2006.08.15. 00:57	495	Grafika$2.class

2006.08.15. 00:57	676	Grafika$3.class
2006.08.15. 00:57	785	Grafika$4.class
2006.08.15. 00:57	727	Grafika$5.class
2006.08.15. 00:57	1 566	Grafika$6.class
2006.08.15. 00:57	6 578	Grafika$Idozito.class
2006.08.15. 00:57	779	Grafika$WelcomeScreen.class
2006.08.15. 00:57	3 826	Grafika.class
2006.08.15. 00:57	1 409	Ido.class
2006.08.15. 00:57	1 529	Jatek.class
2006.08.15. 00:57	1 528	Jatekbolt.class
2006.08.15. 00:57	1 207	KozosOs.class
2006.08.15. 00:57	2 547	Kutya.class
2006.08.15. 00:57	816	LanyOvodas.class
2006.08.15. 00:57	526	LegjobbakAblak$1.class
2006.08.15. 00:57	530	LegjobbakAblak$2.class
2006.08.15. 00:57	2 705	LegjobbakAblak.class
2006.08.15. 00:57	525	Lista$ListaElem.class
2006.08.15. 00:57	3 011	Lista.class
2006.08.15. 00:57	129	Mozgat.class
2006.08.15. 00:57	496	Nehezseg$1.class
2006.08.15. 00:57	626	Nehezseg$2.class
2006.08.15. 00:57	627	Nehezseg$3.class
2006.08.15. 00:57	625	Nehezseg$4.class
2006.08.15. 00:57	2 552	Nehezseg.class
2006.08.15. 00:57	1 792	Ovoda.class
2006.08.15. 00:57	5 032	Ovodas.class
2006.08.15. 00:57	2 968	Ovono.class
2006.08.15. 00:57	5 492	Palya.class
2006.08.15. 00:57	1 961	PalyaElem.class
2006.08.15. 00:57	4 832	Proto.class
2006.08.15. 00:57	811	Szamlalo.class
2006.08.15. 00:57	519	SzintVegeAblak$1.class
2006.08.15. 00:57	705	SzintVegeAblak$2.class
2006.08.15. 00:57	848	SzintVegeAblak$3.class

2006.08.15. 00:57	633	SzintVegeAblak$4.class
2006.08.15. 00:57	1 156	SzintVegeAblak$5.class
2006.08.15. 00:57	2 501	SzintVegeAblak.class
2006.08.15. 00:57	1 099	Toplista.class
	47 fájl	74 906 bájt

13.2. A projekt tapasztalatai

A féléves feladat nagyon hasznosnak bizonyult. Meg kellett tanulnunk együtt dolgozni, felosztani egymás közt a feladatokat és a csapatmunkán belül meg kellett tanulnunk egyedül dolgozni oly módon, hogy az általunk elkészült feladatrészt később más megértse és fel tudja használni. Fontos tapasztalatokat szereztünk továbbá a dokumentáció készítésről. Nagy előnyt jelentett egy olyan dokumentáció, melyben jól láthatók az összefüggések és egymásra épülések, jelentősen megkönnyítette a későbbi munkát. Nagy segítséget jelentett ebben az UML. Ez a leírónyelv biztosította az áttekinthetőséget, hamar láthatóak voltak a segítségével a különböző összefüggések. Hasznos volt az előző féléves tanulmányaink után ezt gyakorlatban is alkalmazni.
Nem volt mindig egyszerű a feladatok megfelelő szétosztása. Néha problémát okozott, hogy egyenlő mértékben és mindenkinek a képességeihez mérten osszunk feladatot. Az összhangot viszont hamar megtalálta a csapat. Nem okozott különösebb nehézséget az sem, mikor egy újabb ponttal bővült a feladat specifikációja a félév közben.
Az idő mindig elégnek bizonyult, bár nem mindig sikerült optimálisan beosztani. A pontszámok nagyrészt arányosak voltak. Ami kevésbé tűnt arányosnak, az a kredit pontok száma. A belefektetett munka mennyisége alapján ez a tárgy inkább 5 kreditet kellene, hogy érjen.

13.3. Értékelés

A csapat ismét úgy döntött, hogy a százalékokat egyenlő arányban osztja el.

NAPLÓ

Dátum	Alany	Tárgy
2006.02.23. 19:15- 2006.02.23. 20:30	Alpi, Szedzsi Tomi	Megbeszélés a feladatokról: - Alpi gyűléslogot vezet - Szedzsi teszteli az UML-es programokat - Tomi kitalál egy jó működést a játékra
2006.02.23. 22:00- 2006.02.23. 23:45	Alpi, Szedzsi Tomi	Értelmezzük a feladat kiírását és ötleteket gyűjtünk
2006.02.25. 18:30- 2006.02.25. 19:45	Alpi, Szedzsi Tomi	Egyéni feladatok fixálása: - Alpi leírást és szótárat készít - Szedzsi követelményeket, projekt tervet - Tomi kitalálja és megírja a use case-eket
2006.02.26. 22:00- 2006.02.26. 23:30	Tomi	Use Case-ek kitalálása, dokumentum megírásának elkezdése és befejezése
2006.02.26. 23:00- 2006.02.27. 03:15	Alpi, Szedzsi	Dokumentumok elkészítése: - Alpi elkezdi és befejezi a szótárat és a leírást - Szedszi megírja és befejezi a követelményeket és a projekt tervet
2006.02.27. 03:30- 2006.02.27. 03:45	Alpi, Szedzsi Tomi	Kész dokumentumok feltöltése FTP-re és összevágása egy dokumentummá
2006.03.02. 21:15- 2006.03.02. 22:00	Szedzsi, Tomi Alpi	Skype videókonferencia beszélgetés, Döntések: - holnap be kell menni konzultációra - holnap be kell menni konzultációra - analízis modellben az objektumok megbeszélése (kezdetlegesen) - leszedni és alkalmazni egy online dok. szerk. programot (MoonEdit)
2006.03.03. 11:00- 2006.03.03 12:00 2006.03.03. 14:00- 2006.03.03. 17:45 2006.03.05. 13:30- 2006.03.05. 14:45	Alpi Szedzsi, Alpi közösen Tomi, Alpi	Konzultáció az I-ben konzulensünkkel (kérdések-válaszok) Analízis modell dokumentációjának elkezdése, osztályok leírásának szerkesztése a MoonEdit programmal Objektum katalógus megírásának elkezdése és az előző dokumentációk kiegészítése
2006.03.05 22:15 2006.03.06 08:00 2006.03.08 19.30- 2006.03.08 21.45	Alpi, Szedzsi Tomi Alpi, Szedzsi Tomi	- A dokumentum végleges alakra hozása - az analízis modell, state-chartok Megrajzolása - A hibák közös értékelése, javításának lehetséges módjait Kerestünk - különböző megoldási útvonalak kijelőlése, pénteki konzultáción Való megbeszélés
2006.03.10 16.00- 2006.03.10 18.00	Alpi, Szedzsi Tomi	- konzultáción való részvétel az I épületben, majd az aulában Megbeszéltük a hibákat.
2006.03.12 22.25 2006.03.13 04.10	Alpi, Szedzsi Tomi	- a helyesnek vélt megoldások analízis modelljének és State Chartjainak a megvalósítása - az objektum modell és katalógusának kijavítása
2006.03.13. 23:30- 2006.03.14. 01:00	Tomi	A szkeleton menüjének elkészítése

2006.03.16. 11:00- 2006.03.16. 15:00	Tomi	Szkeleton preBeta elkészítése
2006.03.16. 20:15- 2006.03.16. 21:30	Alpi, Szedzsi Tomi	Use-case-ek megbeszélése Kollaborációs diagramok felelevenítése
2006.03.17. 13:00- 2006.03.17. 15:30	Alpi, Szedzsi	Use-case lista megbeszélése és megírása
2006.03.18. 12:15- 2006.03.18. 14:00	Alpi, Szedzsi	Architektúra és ütemezés megbeszélése és megírása
2006.03.19. 22:45- 2006.03.20. 03:00	Alpi, Szedzsi Tomi	Kollaborációs diagramok megbeszélése és elkészítése, a dokumentum összevágása
2006.03.21. 20:00- 2006.03.22. 00:45	Tomi	Szkeleton véglegesítés
2006.03.24. 20:15- 2006.03.24. 22:45	Alpi, Szedzsi	Szkeleton tesztelése, konzulens által felvetett problémák Kijavítása
2006.03.26. 23:00- 2006.03.27. 02:30	Alpi, Szedzsi Tomi	Végleges szkeleton változat elkészítése, kommentezés (javadoc) és dokumentáció megírása
2006.03.30. 19:15- 2006.03.30. 21:30	Alpi, Szedzsi Tomi	A főbb usa-case-ek megbeszélése és leírásai
2006.03.31. 15:00- 2006.03.31. 16:15	Alpi, Szedzsi Tomi	Tesztelés megbeszélése
2006.04.02. 22:45- 2006.04.03. 01:45	Alpi, Szedzsi Tomi	- A dokumentum megszerkesztése - Projekt koncepció megbeszélése és megírása - A változtatásokra reagálás
2006.04.04. 13:00 2006.04.04 19:00	Tomi	- ProtOvoda készítése (Szkeleton osztály átalakítása, Fiu, lány ovodás létrehozás)
2006.04.07 16:00 2006.04.06 17:45	Alpi, Szedzsi	- A következő dokumentációhoz szükséges objektumok és Metódusok terveinek megbeszélése
2006.04.09 22:00 2006:04:10 04:45	Alpi, Szedzsi, Tomi	- A részletes tervek szerkesztése, és a diagram elkészítése
2006.04.23. 21:00 2006.04.24. 10:00	Tommey	ProtOvoda készítése 3. Végleges forma
2006.04.23. 21:00 2006.04.23. 22:00	Alpi, Szedzsi	Dokumentáció vázlatának megszerkesztése és jegyzőkönyv Előkészítése
2006.04.24. 10:00 2006.04.24. 16:00	Alpi, Szedzsi, Tommey	- Tesztesetek bemeneti fájljainak tervezése, lefuttatása, ellenőrzése - Dokumentáció összeállítása, nyomtatása, program feltöltése
2006.05.02. 10:00 2006.05.02. 18:00	Alpi, Szedzsi, Tommey	-A grafikus kezelőfelület specifikálása
2006.05.13. 20:00- 2006.05.14. 04:00	Tommey	Grafikus változat véglegesítése
2006.05.14. 09:00- 2006.05.14. 11:30	Szedzsi: Alpi:	Képek keresése a megfelelő objektumokhoz Képek átalakítása, méretezés, kivágás, irányok…
2006.05.14. 22:00- 2006.05.15. 02:30	Alpi, Szedzsi: Tommey:	Képek véglegesítése, formázása Grafikus változat képekkel kiegészített verzójának készítése
2006.05.15. 11:00- 2006.05.15. 13:30	Alpi, Szedzsi	Dokumentáció készítése, a maradék összevágása, feltöltés