Szoftverfolyamat tervezése és kivitelezése 1.
ELTE IK Prog.Inf. Msc 2014. őszi félév

UNTRUSTED című játékprogram fejlesztési és felhasználói dokumentációja

Készítették: Hideg András és Zádory Zsolt
2014. szept.- dec.

1 Követelmény feltárás

1.1 Célkitűzés

A projekt célja egy asztali Java alkalmazás készítése, melyben a felhasználó (játékos) a pályát létrehozó kód átírásával oldhat meg logikai fejtörőket. A program telepíthető formában kerül kiadásra, lehetőleg minden kompatibilis platformra.

Indítás után egy standard főmenü fogadja a felhasználót. Az innen elérhető funkciók közt szerepel az új játék indítása, mentés/betöltés, beállítások, illetve a legjobb eredmények megtekintése.

Új játék kezdetekor egy bevezető szöveg segíti az indulást (esetleg valamilyen kerettörténetbe rejtve). Az első pálya tanító jellegű, melyben bemutatásra kerülnek az alap funkciók, továbbá a navigáció. Többi pályához is ajánlott kísérő szöveg, de nem alapkövetelmény.

Mentés és betöltés a megszokott módon történik. Lehet korlát a mentések darabszámára. Profilok támogatása nincs tervezve, de később beépíthető. Mentések listázásánál szerepel a mentés sorszáma, illetve az aktuális pálya azonosítója.

Beállításoknál távolabbi cél a programozási nyelv átállítása. További fontosabb kikötés nincs ezen opcióra.

Eredménylistára való felkerülés illetve a pontszám mérése történhet többféle módon. Lehet véges élettel ellátni a játékost, és minden sikertelen próbálkozás után elfogyasztani egyet az életpontokból. Másik módszer lehet egy kiinduló maximum pont redukálása minden újrakezdés alkalmával, „végtelen élettel”. Továbbá mérhetők egyéb metrikák, mint például a teljesítés ideje, vagy a megoldó kód hossza/bonyolultsága.

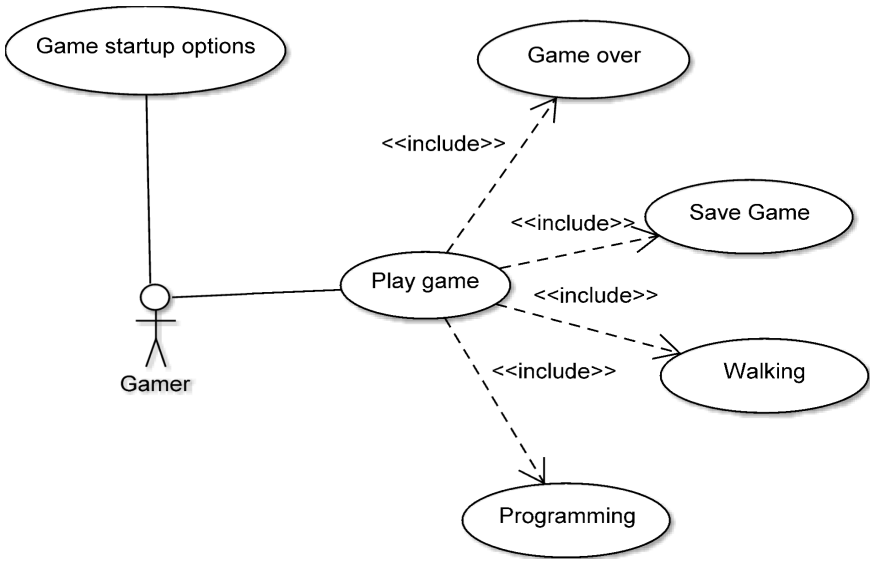
1.2 Szakterületi fogalomjegyzék

A játékfelület különböző elemeire illetve objektumaira a következő kifejezések hivatkoznak:

pálya, labirintus	Az a lezárt terület, ahol a játék folyik. Teljesítése azt jelenti, hogy a játékos eljutott a kijáratig, és új pályára került.
kijárat	A pálya végét és egyben teljesítését jelentő ikon/objektum.
játékos-ikon	A pálcikaemberke, akit a felhasználó irányít.
computer	A pálya átprogramozható. A módosító programot egy grafikus elemen, a számítógépen kell lefuttatni. Ez egy lapotopot szimbolizáló ikon.
mentési pont	A mentési pontba érve a játék elmenthető, de máshol a mentés nem lehetséges.
fal	A játékos és egyéb mozgó objektumok mozgását korlátozó pályaelem.
ellenség	Hozzáérve a felhasználó pontot veszít, illetve meghal.
szellem	Egy ellenség, ami mozog.
csapda	Egy nehezen észrevehető, statikus ellenség.
óra	A játék időre megy, a hátrlévő időt jelzi az óra.
halál	A felhasználó bizonyos ellenfelekkel ütkése után a pálya újra indul, és a játékos pontjai és ideje csökken. Ez nem feltétlenül jelenti a játék végét.
pont	A felhasználó a teljesített pályák, rejtvények stb. után pontot kap.
sprite	A játékhoz tartozó mozgatható, esetleg animált kis ikon / figura.
küldetés (mission)	Pályák sorozata, amelyek teljesítése után a felhasználó nyer.
Főmenü	A bejelentkező menü illetve a kapcsolódó komponensek továbbá a főablak.
Controller	Az MVC architektúra Model komponense (ld. még osztályoknál).
View	Az MVC architektúra Model komponense.
Model	Az MVC architektúra Model komponense.
DAO	Ugyanaz mint a Model
terminál	Ahol a pálya kódot be lehet írni (számítógépként is szerepel).

1.3 HASZNÁLATI ESET MODELLEK

GAMING ALRENDSZER



Use Case Specifikáció	
Use case név	Play game
Alrendszer	gaming
Leírás	Játszma lejátszása: részletek az include-olt esetekben
Előfeltétel	A language alrendszer és a kezelőfelület inicializálása sikeres volt. A játék-kezdési adatok meg lettek adva.
Célok	Játékfelület biztosítása
Kiváltó események	Megfelelő menüelemre kattintás.
Ideális forgatókönyv	(1) A felhasználó - miután megadta az alapbeállításokat - effektíve itt játszik. (2) a játszma egy idő múlva véget ér.
Havaria forgatókönyv(ek)	Hibák léphetnek fel, ekkor a program visszatér a főmenübe

Use Case Specifikáció	
Use case név	Game startup options
Alrendszer	gaming
Leírás	Játszma megkezdése előtti adatok megadás, pl. nehézségi szint, grafikai beállítások, stb.
Előfeltétel	A többi alrendszer inicializálása sikeres volt.
Célok	Alapadatok megadása a játékhoz
Kiváltó események	"játék" menüelemre kattintás.
Ideális forgatókönyv	(1) A Gamer adatokat ad meg
Havaria forgatókönyv(ek)	--

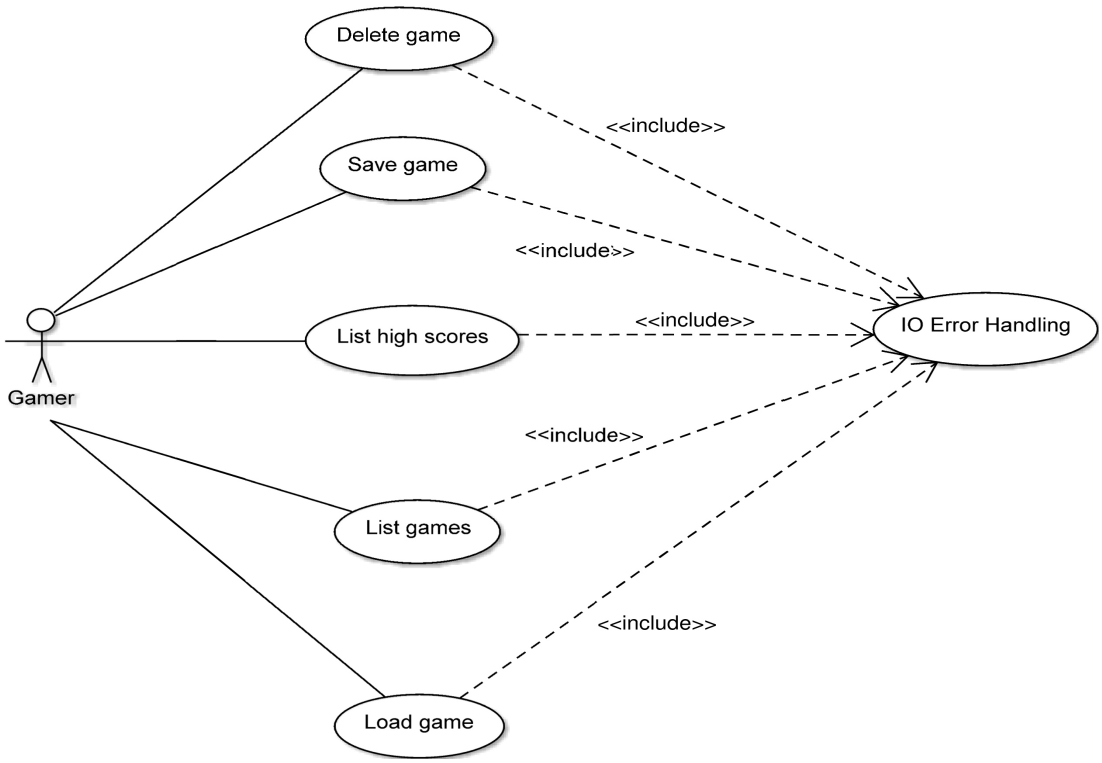
Use Case Specifikáció	
Use case név	Programming
Alrendszer	gaming
Leírás	A játékos átprogramozza a pályát
Előfeltétel	Játszma elindult, computer-nél van a játékos
Célok	Pálya módosítása
Kiváltó események	Computer aktivizálása
Ideális forgatókönyv	Rendben lefut a beirt progam
Havaria forgatókönyv(ek)	Rossz a beírt kód, visszaáll az eredeti pálya

Use Case Specifikáció	
Use case név	Save Game
Alrendszer	gaming
Leírás	Játékos menti az állást a speciális mentési pontoknál
Előfeltétel	A játék elindult
Célok	Állás mentése
Kiváltó események	Mentés gombnál van a játékos és aktivizálja azt
Ideális forgatókönyv	A játszmát elmenti
Havaria forgatókönyv(ek)	IO error

Use Case Specifikáció	
Use case név	Walking
Alrendszer	gaming
Leírás	A játékos mászkál a pályán, közben meghalhat, szüneteltetheti a játékot.
Előfeltétel	A játék elindult
Célok	Általában bejárja a pályát és ki próbálja kerülni az ellenséget. Átlép a következő pályára.
Kiváltó események	Kilépés mentési pontból, computer-től, játszma indítása
Ideális forgatókönyv	A játékos bejárja a pályát
Havaria forgatókönyv(ek)	

Use Case Specifikáció	
Use case név	Game over
Alrendszer	gaming
Leírás	Vége a játéknak: győzelem vagy vereség.
Előfeltétel	Játszott a játékos
Célok	Győzelemről vagy vereségről tájékoztat. esetén
Kiváltó események	Halál, lejár az idő - vagy befejezi az utolsó pályát
Ideális forgatókönyv	Megjelenik az összesítő a képernyőn
Havaria forgatókönyv(ek)	

PERSISTENCE ALRENDSZER



Use Case Specifikáció	
Use case név	Save game
Alrendszer	Persistence
Leírás	A játékos (Gamer) elmenti a játszmákat. Beállíthatja, hogy melyik file-ba menti az adatokat.
Előfeltétel	A persistence alrendszer inicializálása sikeres volt. Játék folyamatban.
Célok	Játszma mentése
Kiváltó események	Megfelelő menüelemre kattintás.
Ideális forgatókönyv	(1) A felhasználó megnyit egy menüt, amelyben megadja a kimeneti fájlt (2) majd elementi a játékot
Havaria forgatókönyv(ek)	Hibák léphetnek fel: I/O hiba

Use Case Specifikáció	
Use case név	Load game
Alrendszer	Persistence
Leírás	A játékos (Gamer) betölti egy mentett játszmát. Beállíthatja, hogy melyik file-ból.
Előfeltétel	A persistence alrendszer inicializálása sikeres volt.
Célok	Játszma betültése
Kiváltó események	Megfelelő menüelemre kattintás.
Ideális forgatókönyv	(1) A felhasználó megnyit egy menüt, amelyben megadja a kimeneti fájlt (2) majd betölti a játékot
Havaria forgatókönyv(ek)	Hibák léphetnek fel: I/O hiba

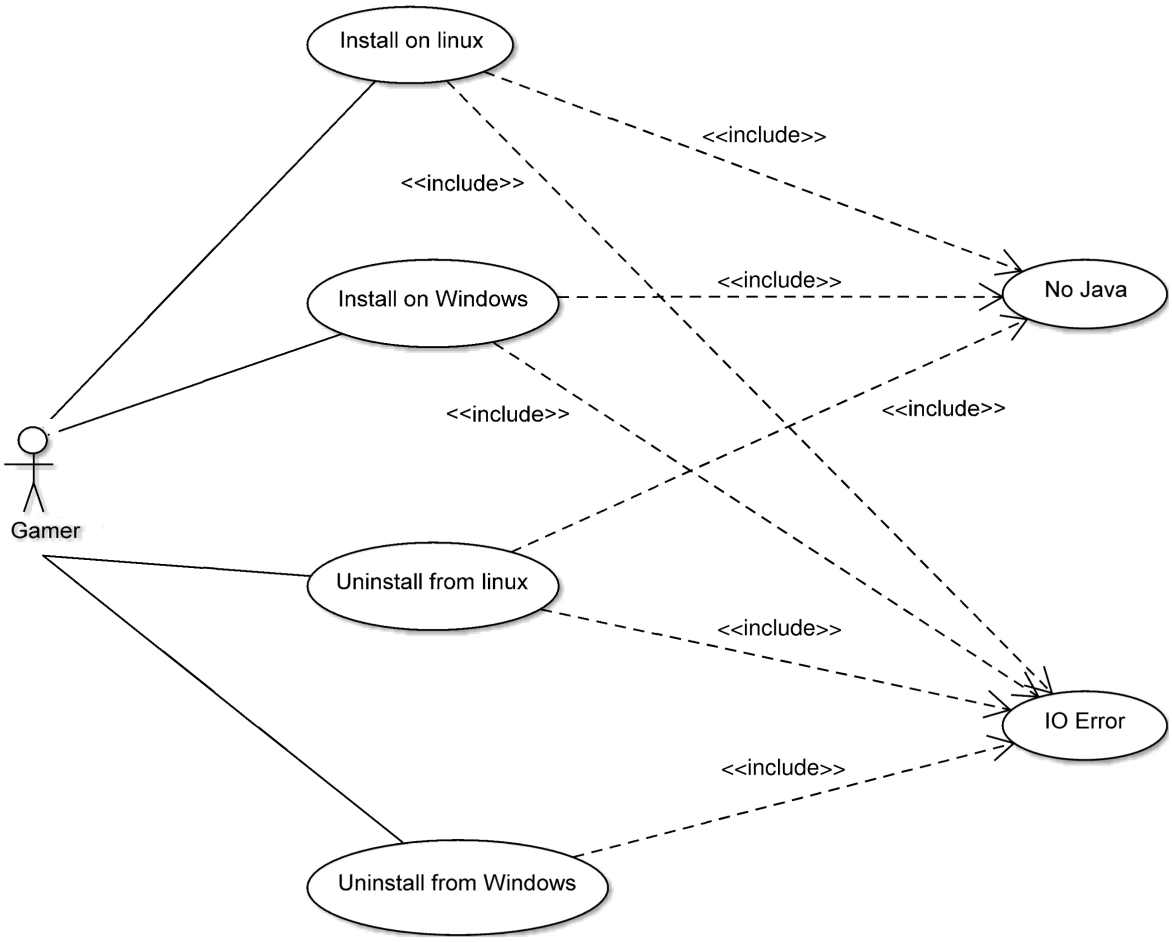
Use Case Specifikáció	
Use case név	Delete game
Alrendszer	Persistence
Leírás	A játékos (Gamer) töröl elmentett játszmákat. Beállíthatja, hogy melyik file-okat.
Előfeltétel	A persistence alrendszer inicializálása sikeres volt.
Célok	Játszma vagy játszmák törlése
Kiváltó események	Megfelelő menüelemre kattintás.
Ideális forgatókönyv	(1) A felhasználó megnyit egy menüt, amelyben megadja a kimeneti fájlt (2) majd töröl a listában megjelenő játszák közül néhányat
Havaria forgatókönyv(ek)	Hibák léphetnek fel: I/O hiba

Use Case Specifikáció	
Use case név	List games
Alrendszer	Persistence
Leírás	A játékos (Gamer) kilistázza a mentett a játszmákat.
Előfeltétel	A persistence alrendszer inicializálása sikeres volt.
Célok	Játszmák listázása
Kiváltó események	Megfelelő menüelemre kattintás.
Ideális forgatókönyv	(1) A felhasználó megnyit egy menüt, amelyben megadja a kimeneti fájlt (2) majd listázza a mentett játszmákat
Havaria forgatókönyv(ek)	Hibák léphetnek fel: I/O hiba

Use Case Specifikáció	
Use case név	List high scores
Alrendszer	Persistence
Leírás	A játékos megjeleníti a legmagasabb pontszámokat
Előfeltétel	A persistence alrendszer inicializálása sikeres volt.
Célok	Legmagasabb pontszámok listázása
Kiváltó események	Megfelelő menüelemre kattintás.
Ideális forgatókönyv	(1) A felhasználó rákkatint a megfelelő gombra (2) majd a rendszer kilistázza a pontszámokat
Havaria forgatókönyv(ek)	Hibák léphetnek fel: I/O hiba

Use Case Specifikáció	
Use case név	IO error handling
Alrendszer	Persistence
Leírás	File írás/olvasás során fellépő hibák esetén megjelenő hibaüzenet
Előfeltétel	IO hiba történt
Célok	Hibajelző felület biztosítása
Kiváltó események	IO hiba történt
Ideális forgatókönyv	Megjelenít egy hiba-ablakot
Havaria forgatókönyv(ek)	

INSTALL ALRENDSZER



Use Case Specifikáció	
Use case név	Install on linux
Alrendszer	install
Leírás	Telepíti a programot Linuxra
Előfeltétel	Megfelelő operációs rendszer álljon rendelkezésre
Célok	A program telepítése
Kiváltó események	Telepítő indítása
Ideális forgatókönyv	(1) A felhasználó elindítja a telepítőt (2) Ha nem talál a telepítő a gépen JRE-t, akkor megkéri a felhasználót, hogy mutassa meg, hol van JRE. Ha nincs JRE, akkor kilép. (3) Ha van JRE, akkor a felhasználói jogosultságtól függően: (a) root esetén bárhova (pl. /opt/games) telepíti a programot (b) ha nem root, akkor csak egy megadott lokális mappába telepíti a programot
Havaria forgatókönyv(ek)	Hibák léphetnek fel: IO error, jogosultsági probléma, Nincs Java

Use Case Specifikáció	
Use case név	Install on linux
Alrendszer	install
Leírás	Telepíti a programot Linuxra
Előfeltétel	
Célok	A program telepítése
Kiváltó események	Telepítő indítása
Ideális forgatókönyv	(1) A felhasználó elindítja a telepítőt (2) Ha nem talál a telepítő a gépen JRE-t, akkor megkéri a felhasználót, hogy mutassa meg, hol van JRE. Ha nincs JRE, akkor kilép. (3) Ha van JRE, akkor a felhasználói jogosultságtól függően: (a) root esetén bárhova (pl. /opt/games) telepíti a programot (b) ha nem root, akkor csak egy megadott lokális mappába telepíti a programot
Havaria forgatókönyv(ek)	Hibák léphetnek fel: IO error, jogosultsági probléma, Nincs Java a gépen

Use Case Specifikáció	
Use case név	Install on Windows
Alrendszer	install
Leírás	Telepíti a programot Windows
Előfeltétel	
Célok	A program telepítése
Kiváltó események	Telepítő indítása
Ideális forgatókönyv	(1) A felhasználó elindítja a telepítőt (2) Ha nem talál a telepítő a gépen JRE-t, akkor megkéri a felhasználót, hogy mutassa meg, hol van JRE. Ha nincs JRE, akkor kilép. (3) Ha van JRE, akkor a felhasználói jogosultságtól függően: (a) root esetén bárhova (pl. c:\Program Files\) telepíti a programot. Nem ir a registry-be. (b) ha nem root, akkor csak egy megadott lokális mappába telepíti a programot
Havaria forgatókönyv(ek)	Hibák léphetnek fel: IO error, jogosultsági probléma, Nincs Java a gépen

Use Case Specifikáció	
Use case név	uninstall on linux
Alrendszer	install
Leírás	Eltávolítja a programot Linuxról
Előfeltétel	Legyen fenn a program
Célok	A program törlése
Kiváltó események	Uninstaller indítása
Ideális forgatókönyv	(1) A felhasználó elindítja az uninstallert (2) Ha vannak fent mentett játszmák, akkor erre figyelmeztet (3) Eltávolítja a programot.
Havaria forgatókönyv(ek)	Hibák léphetnek fel: IO error, jogosultsági probléma, Nincs Java a gépen

Use Case Specifikáció	
Use case név	Uninstall on linux
Alrendszer	install
Leírás	Eltávolítja a programot Windows-ról
Előfeltétel	Legyen fenn a program
Célok	A program törlése
Kiváltó események	Uninstaller indítása
Ideális forgatókönyv	(1) A felhasználó elindítja az uninstallert (2) Ha vannak fent mentett játszmák, akkor erre figyelmeztet (3) Eltávolítja a programot.
Havaria forgatókönyv(ek)	Hibák léphetnek fel: IO error, jogosultsági probléma, Nincs Java

Use Case Specifikáció	
Use case név	No Java
Alrendszer	install
Leírás	Nincs Java a rendszeren
Előfeltétel	Nincs java futtatókörnyezet
Célok	Értesíteni a felhasználót a JRE hiányáról
Kiváltó események	(Un)installer indítása
Ideális forgatókönyv	Értesíti pl. egy hiba-panelen a felhasználót
Havaria forgatókönyv(ek)	

Use Case Specifikáció	
Use case név	IO error
Alrendszer	install
Leírás	Írási/olvasási/jogosultsági hiba történt
Előfeltétel	IO v. jogosultsági hiba
Célok	Értesíteni a felhasználót a hibáról
Kiváltó események	(Un)installer indítása
Ideális forgatókönyv	Értesíti pl. egy hiba-panelen a felhasználót
Havaria forgatókönyv(ek)	

1.4 Szakterületi követelmények

Logikai fejtörők közül kezdetben három csoport kerülhet implementálásra. Elsőnek a labirintus típusú pályákat érdemes megvalósítani. Ezeknek az alapszituációjában a kijárat valamilyen módon el van zárva vagy nincs direkt irányítás alatt a „játékos”. Második csoport a fizika alapú fejtörők. Ezekben értelemszerűen valamilyen való életben felmerülő akadályt kell modellezni. Példának megemlíthető a folyón való átkelés (a játékos alapból nem tud úszni), vagy például egy szakadék áthidalása (gravitáció leküzdése). Utolsóként említendő a valamilyen egyszerűbb gépi ellenséges karakterek legyőzése/elkerülése.

A felhasználó felé a pályának csak a rajzoló része látszódik illetve a megoldáshoz kritikus algoritmusok (a fentebbi gépi ellenfeles példához mondjuk a gondolkodási stratégia), vagyis a teljes implementációhoz nem férhet hozzá. Ennek megfelelően nyújtani kell valamilyen API-t, amivel befolyásolhatja a környezetet. További megszorításokat kell tenni a triviális megoldások kivédésére.

1.5 Nem funkcionális követelmények

Fejlesztési módszertan:

- Egységesített Eljárás

A fejlesztéshez használt hardver:

- CPU: Intel Core i5, RAM: 8 GB, videó: 1600x900

A fejlesztéshez használt szoftverek:

- Operációs rendszer: Arch linux, Windows 7

Követelmény elemzés:

- Libreoffice Writer (4.3.2) szövegszerkesztővel. MS Office

CASE eszköz:

- ArgoUML

Java fejlesztőeszköz:

- NetBeans 7.x, Anache Ant 1.9

A futtatáshoz szükséges operációs rendszer:

- Tetszőleges operációs rendszer, melyhez létezik JRE 7 implementáció

A futtatáshoz szükséges hardver:

- CPU: Pentium 2600, RAM: 2 GB, Videó: 512 MB
  (látható, hogy szerényebb teljesítményű gépen is fut a program)

Egyéb követelmények:

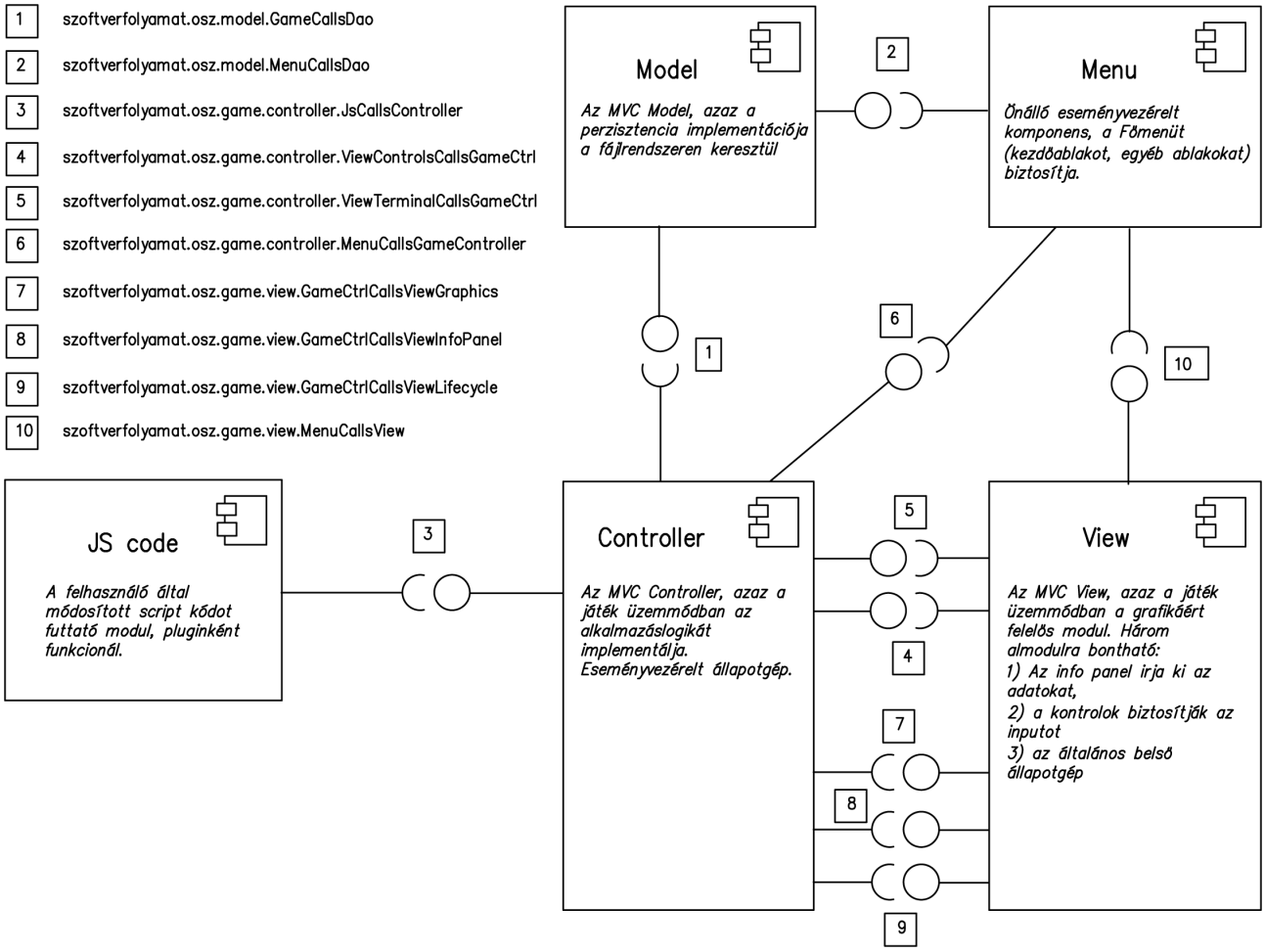
- Intuitív felhasználói felület, könnyű kezelhetőség

2 Tervezés

Ez a fejezet a program tervezésének a legfontosabb koncepcióit és aspesktusait mutatja be.

2.1 A program architektúrája

A program két üzemmódban működik: az egyik a menű mód a másik a játék mód. A menű mód a bejelentkező képernyőt és az ehhez kapcsolódó előugró ablakokat jelenti. Ez egy egyszerű eseményvezérelt folyamat, ezért itt nem volt szükség bonyolultabb tervezési minta implementálására. Játék módban a Model View Controller (MVC) mintát implementáltuk, illetve ezen belül szerepel a Singleton, a Decorator, az Observer és a Factory pattern. Erről az osztályleírások részletes tájékoztatást adnak.



A program architektúráját komponens diagramon mutatjuk be, ez a fenti ábrán látható. (A komponensek közötti kommunikációs interfészek a későbbiekben részletesen szerepelnek.) Itt szerepelnek kerülnek a program logikai komponensei, más néven a modulok. A program három modulból áll, ezek a `menu` (fent Menu) modul, a `game` modul (ez a Controller View és JS Code együtt) és a `model` (fent Model) modul. A modulok szerinti felosztás elvben a fejlesztői kompetenciákhoz és szerepekhez próbál igazodni. Elvileg minden modult más fejlesztő készít el és a modulok csak (a lehetőségek szerint minimalizált) interfészeken keresztül érintkeznek.

A `menu` modul jeleníti meg a bejelentkező képernyőt, innen lehet a különböző almenükbe navigálni, például a beállítások megadására vagy egy elmentett játék betöltésére szolgáló menük érhetők el ebben a modulban. Itt tekinthetők meg a legmagassabb pontszámok is. Amikor a felhasználó elindítja a játékot, akkor a `menu` modul `JFrame` -jén belül megjelenik a játékfelület panelje.

A `game` modul biztosítja a játékfelületet, azaz itt lehet játszani a játékkal, és innen lehet állásokat elmenteni. A `game` modul számára a `menu` modul továbbítja az ablak-eseményeket, például a főablak minimalizálásakor (task-ba leküldésekor) a `menu` modul fő `JFrame` −je jelzi a `game` modulnak, hogy a játék szüneteljen. A `game` modul jelzi a `menu` -nek ha egy játszma befejeződött, és a játékos visszalép a főmenübe.

A `model` (Data Access Object) modul biztosítja a perzisztenciát, mint például a pályák vagy játékosok adatbázisát. Ezt a modult mindkét másik modul használja.

A kommunikáció a `game` és a `menu` között kétirányú, a `game` és a `model` között egyirányú, valamint a `menu` és a `model` között is egyirányú (ami azt jelenti, hogy csak a `model` -t hívják, de ő nem hív vissza).

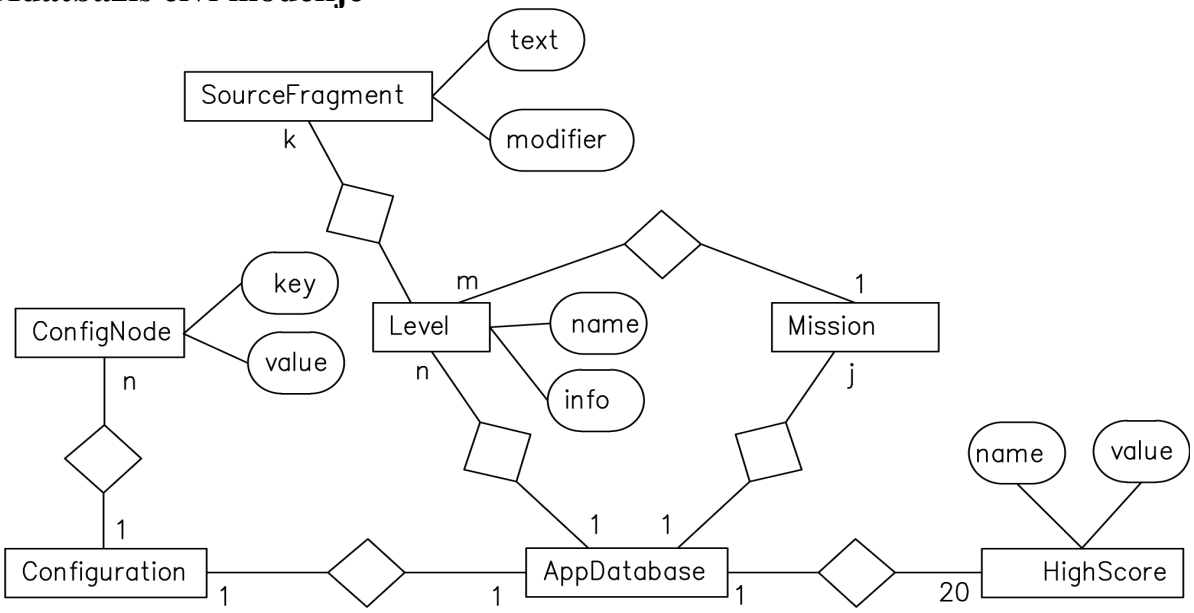
2.2 Osztálymodell

A program osztályairól részletes leírás készült, ezt a 2.6 pont tartalmazza. Továbbá az UML diagramok megtalálhatók a mellékelt class_diagram.zip fájlban .png formátumban (mivel ezek közül néhány igen terjedelmes fájl és A4-es formátumban, PDF-ben nem igazán kényelmes kezelni ezeket).

2.3. Adatbázisterv

Ebben a pontban a program által használt adatbázis modellje és implementációja kerül bemutatásra.

Adatbázis elvi modellje



A program adatbázismodelljének ER diagramja

A fenti ábra az adatbázismodellt mutatja be, ahol
- A `Configuration` a konfigurációs adatbázis. (Egyelőre nem világos, hogy mi kerül ide, de később ez tárolhatná az anyanyelvi beállítást, a játék nehézségi szintjét stb.) Ez egy "dummy" bejegyzés most még.
- A `ConfigNode` egy kulcs/érték pár a konfigurációs bejegyzésekben.
- A `Level` egy pálya a játékban, amelyet a programozó készít el. A `Level` JavaScript forráskódot tartalmaz, elkülönítve a játékos által szerkeszthető, nem szerkeszthető valamint a nem látható állományt.
- A `Mission` egy küldetés, ami `Level` -ek egy sorozatát jelenti. A játékos küldetést választ ki és azt játssza le.
- A `HighScore` a legmagasabb elért pontszámokat tartalmazó, (mondjuk) 20 elemű adatbázis. (játékos neve, pontszám) párokat tartalmaz. A `menu` olvassa, a `game` írja ezeket a bejegyzéseket. A játékosokról külön adatbázis egyelőre nem készül.
- Az adatbázis az `AppDatabase` gyökér node fogja össze.

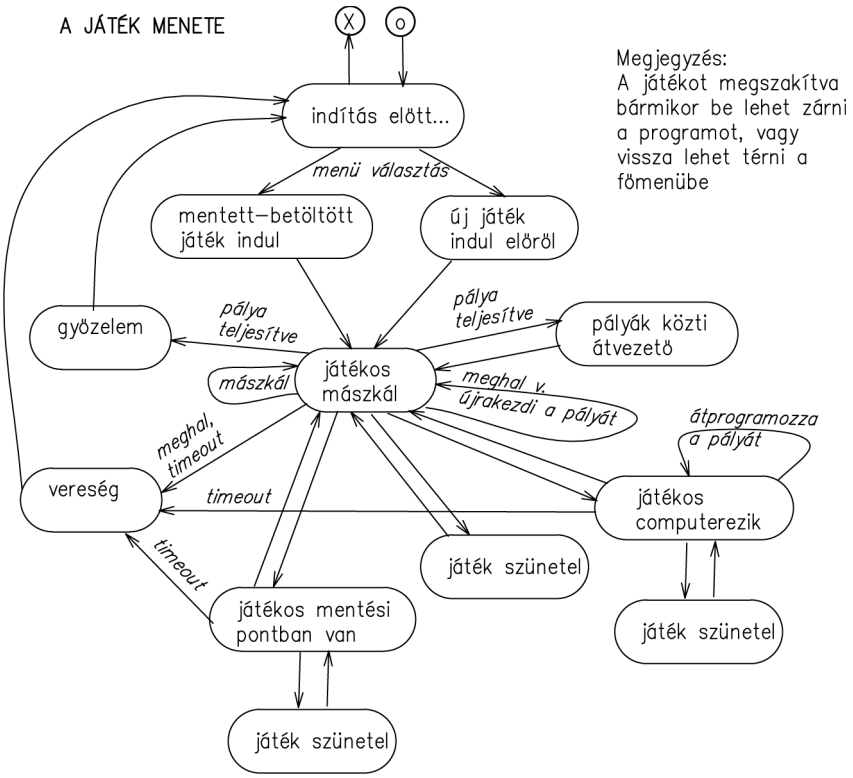
Továbbá:
- A `SavedGames` az elmentett játékok adatbázisa; jelenleg még nincs implementálva, ezért nincs a diagramon.
- A `Game` egy bejegyzés az elmentett játékokat tartalmazó adatbázisban. Ezeket a játékosok mentik el a mentési pontoknál illetve töltik be a játék indításakor. Jelenleg még nincs implementálva.

Megvalósítás a fájlrendszerben

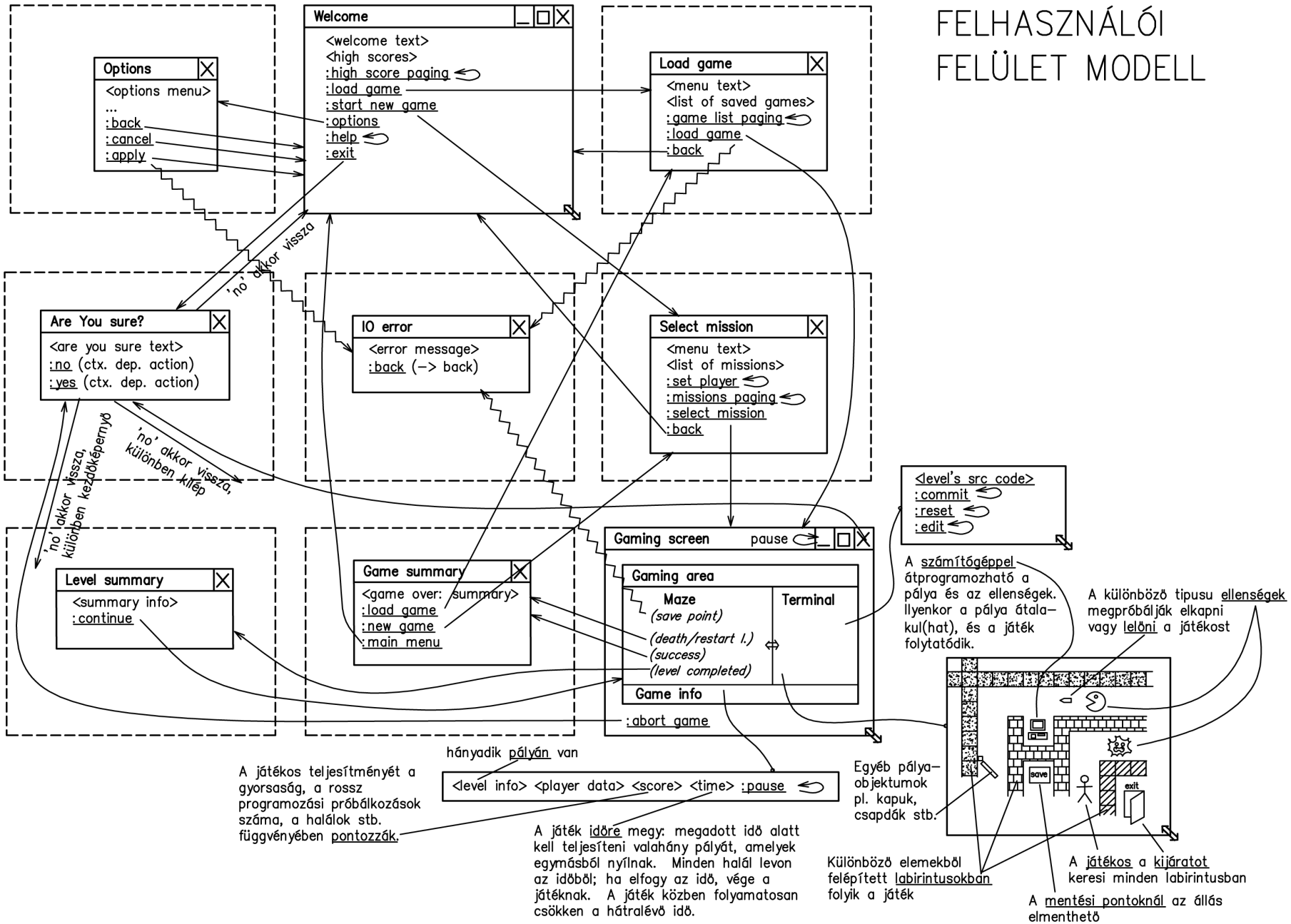
Az `<app-root>/data` könyvtárba kerülnek a perzisztens adatok, a következők szerint:
- Az `<app-root>/data/level*.xml` fájlokba kerülnek a pályák. Ezek XML fájlok, ahol a forráskód-részeket tartalmazó tag-ek jelölhetik, hogy az adott kódrészlet írható illetve csak olvasható. A pályák forráskódja pedig CDATA szekciókban tárolt (lásd a konkrét alkalmazást).
- Az `<app-root>/data/missions.properties` fájlba kerülnek a küldetések, amelyek lényegében pályák sorozatai. Itt nyílván csak a pályákra mutató hivatkozások vannak. Ez egy Java properties fájl, ahol a kulcs a küldetés neve
- Az `<app-root>/data/high_scores.properties` fájlba kerülnek a legmagasabb pontszamok (string, integer) párként. Ez is egy Java properties fájl.
- Az `<app-root>/data/config. properties` fájlba kerülnek a konfigurációs bállítások (string, string) kulcs/érték párként. Ez is egy Java properties fájl.
- Az `<app-root>/data/games` fájlba kerülhetnek majd az elmentett játszmák szerializált Java objektumokként.

2.4. Dinamikus működés

A program dinamikus működését játék módban mutatjuk be, állapotátmeneti diagrammal.



2.5. UI tervek


FELHASZNÁLÓI FELÜLET MODELL
Welcome
<welcome text>
<high scores>
:high score paging
:load game
:start new game
:options
:help
:exit
Options
<options menu>
...
:back
:cancel
:apply
Load game
<menu text>
<list of saved games>
:game list paging
:load game
:back
'no' akkor vissza
Are You sure?
<are you sure text>
:no (ctx. dep. action)
:yes (ctx. dep. action)
IO error
<error message>
:back (–> back)
Select mission
<menu text>
<list of missions>
:set player
:missions paging
:select mission
:back
'no' akkor vissza, különben kezdőképernyő
'no' akkor vissza, különben kilép
Level summary
<summary info>
:continue
Game summary
<game over: summary>
:load game
:new game
:main menu
Gaming screen
pause
Gaming area
Maze
(save point)
(death/restart l.)
(success)
(level completed)
Terminal
Game info
:abort game
<level's src code>
:commit
:reset
:edit
A számítógéppel átprogramozható a pálya és az ellenségek. Ilyenkor a pálya átala–kul(hat), és a játék folytatódik.
A különböző tipusu ellenségek megpróbálják elkapni vagy lelöni a játékost
save
exit
Egyéb pálya–objektumok pl. kapuk, csapdák stb.
Különböző elemekből felépített labirintusokban folyik a játék
A játékos a kijáratot keresi minden labirintusban
A mentési pontoknál az állás elmenthető
hányadik pályán van
<level info> <player data> <score> <time> :pause
A játékos teljesítményét a gyorsaság, a rossz programozási próbálkozások száma, a halálok stb. függvényében pontozzák.
A játék időre megy: megadott idő alatt kell teljesíteni valahány pályát, amelyek egymásból nyílnak. Minden halál levon az időből; ha elfogy az idő, vége a játéknak. A játék közben folyamatosan csökken a hátralévő idő.

2.6 Részletes osztályleírások

package szoftverfolyamat.osz.commons;
A csomagba grafikus és általános célú segédosztályok kerültek.

---

public class **ServiceAllocator**
Sztereotipia: Helper
Feladat: A service discovery mechaniznussal megkeresi egy interface implementációját.

```
public static <T> T allocateService(Class<T> klass)
```
Adott service specification osztály/interfész implementációjának lekérése.

---

public class **ImageProvider**
Sztereotipia: Helper
Feladat: A jar file-okban tárolt képek betöltését és méretezését segíti.

```
public static Image loadImage(Class klass, String path,
                              int width, int height)

public static Image loadImage(Class klass, String path)

public static ImageIcon loadIcon(Class klass, String path,
                                 int width, int height)

public static ImageIcon loadIcon(Class klass, String path)
```

---

public class **FontProvider**
Sztereotipia: Helper
Feladat: Adott jar file-ban egy elérési úton található fontokat éri el ez az osztály.

```
private static Font regular
private static Font regularItalic
private static Font bold
private static Font boldItalic
private static Font monospace
private static Font monospaceBold
private static AffineTransform transform

public static boolean init(String resourcePath,
                           String r, String ri, String b, String bi,
                           String ms, String msb, double tx, double ty)
```
Meg kell hivni ezt először. Inicializál, betölti a fontokat.

```
public static Font getMonospaceBold(float sizePx)
```
Bold Terminal font lekérése.

```
public static Font getMonospace(float sizePx)
```
Terminal font lekérése.

```
public static Font getRegular(float sizePx)
```
Normál font lekérése.

```
public static Font getRegularItalic(float sizePx)
```
Normál dőlt font lekérése.

```
public static Font getBold(float sizePx)
```
Normál félkövér font lekérése.

```
public static Font getBoldItalic(float sizePx)
```
Félkövér dőlt font lekérése.

```
public static Font getFont(Class klass, String path, float height)
```
Tetszőleges font lekérése.

```
public class ColorProvider
```
Sztereotipia: Helper
Feladat: Színkezelési segédosztály

```
public static Color lighten(Color color, double percent)
```
Világosabbá ill. sotétebbé teszi a color paramétert a percent százalékos arányban.

```
public static Color copyOf(Color c)
```
Másolatot készit az input paraméter szinről.

```
public static Color matten(Color color, double percent)
```
A percent százalékkal mattabbá teszi a color szint.

```
private static Random random = new Random()
```

```
public static Color randomColor()
```
Véletlen szint ad vissza.

```
public static Color colorByRgbString(String s)
```
String alapjan szin

```
public static Color colorByInt(int i)
```
Int alapjan szin

```
package szoftverfolyamat.osz.commons.graphics;
```
A csomagba grafikus és általános célú segédosztályok kerültek.

```
public abstract class ModalDialog extends JDialog
```
Sztereotipia: boundary
Feladat: Specializált modális párbeszédablak-osztály, az alkalmazás betűkészletével és színeivel.

```
public static JFrame jframe
protected final GenericPanel buttonRow
protected final GenericPanel bgPanel
protected final JTextArea textArea
```

```
public ModalDialog(String t, int w, int h)
```

```
public void addButton(GenericButton b)
```
Vezérlőgombot helyez el a panel alján.

```
public void setText(String s)
```
(Az itt következő getter/setter metódusok egyértelműek).

```
public String getText()
```

```
public JTextArea getTextArea()
```

```
public void setColor(int c)
```

```
public void setButtonColor(int c)
```

```
protected void dropDialog()
```
A swing EDT -re helyezi a bezárási akciót, biztosítva így a megfelelő működést.

```
public void setVisibleLater()
```
A swing EDT -re helyezi a láthatóvá tétel akciót, biztosítva így a megfelelő működést.

```
public abstract class MessageBox extends ModalDialog
```
Sztereotipia: boundary
Feladat: Egyszerű üzenetpanel egy OK gombbal. Automatikusan a parent-je közepén jelenik meg.

```
public MessageBox(String t, int w, int h)
```

```
protected abstract void clicked()
```
A panel OK gombjára kattintva mi történjen.

```
public class GraphicConstants
```
Sztereotipia: Konstansok
Feladat: Alkalmazásspecifikus konstansok, a nevek egyértelműek.

```
public final static float STD_FONT_HEIGHT

public final static Font STANDARD_FONT
public final static Font STANDARD_BOLD_FONT
public final static Font BIG_STANDARD_BOLD_FONT
public final static Font MONOSPACE_FONT
public final static Font MONOSPACE_BOLD_FONT
public final static Font BIG_MONOSPACE_BOLD_FONT

public final static int STANDARD_BG_COLOR_INT
public final static Color STANDARD_BG_COLOR
public final static int BUTTON_HILITE_COLOR_INT
public final static int BUTTON_COLOR_INT
public final static int GRAY_BG_COLOR_INT
public final static int GRAY_BUTTON_HILITE_COLOR_INT
public final static int GRAY_BUTTON_COLOR_INT
public final static int GRAY_1_INT
public final static int GRAY_0_INT
public final static int GRAY_2_INT
public final static int LIGHT_TEXT_COLOR_INT
public final static Color LIGHT_TEXT_COLOR
```

```
public class GenericPanel extends JPanel
```
Sztereotipia: Konténer
Feladat: Specializált JPanel leszármazott, gyakran használt segéd-metódusokkal. A metódusok és azok paramétereinek a neve sok esetben egyértelművé teszi a funkciót; az ilyen esetekben a magyarázat elmaradt.

```
public GenericPanel()

public GenericPanel(int color)

public void setNullLayout()
```
Null (szabad) layout-ot ad meg a panelnek.

```
public void setBorderLayout()

public void setFlowLayout()
```

```
public void setBoxLayout(boolean b)
```
Ha b true, akkor vizszintes, különben függőleges.

```
public void setBackground(int color)
```

```
public void setTitledBorder(String title, int color)
```
Feliratos keretet ad a panelnek.

```
public void setEmptyBorder(int width)
```

```
public void setLinedBorder(Color color, int width)
```

```
public void setAllSizes(int x, int y)
```
Minden méretét fixre állítja.

A következő nyolc metódus a BorderLayout kezelését egyszerűsíti.
```
public void addToStart(Component c)
```

```
public void addToEnd(Component c)
```

```
public void addToTop(Component c)
```

```
public void addToBottom(Component c)
```

```
public void addToCenter(Component c)
```

```
public void addToLeft(Component c)
```

```
public void addToRight(Component c)
```

```
public void invokeLater(Runnable r)
```
A swing EDT -re helyezi az `r` akciót.

```
public class GenericLabel extends JLabel
```
Sztereotipia: boundary
Feladat: Specializált JLabel leszármazott, gyakran használt segéd-metódusokkal. A metódusok és azok paramétereinek a neve egyértelművé teszi a funkciókat.

```
public GenericLabel(int color, String caption)
```
A `color` a betűszin.

```
public GenericLabel(String caption)
```

```
public GenericLabel(ImageIcon image)
```

```
public void setForeground(int color)
```

```
public void alignToCenter()
```

```
public void alignToLeft()
```

```
public void alignToRight()
```

```
public abstract class GenericButton extends JButton
```

Sztereotipia: boundary
Feladat: Specializált JButton leszármazott, gyakran használt segéd-metódusokkal. A metódusok és azok paramétereinek a neve több helyen egyértelművé teszi a funkciókat.

```
    private int bgColor
    private int moverColor

    public GenericButton(String caption, int color, int moverColor,
                         int inactiveColor)

    protected abstract void onClick()
```

Kattintás esetén ezen metódus implementációja hívódik meg.

```
    public void setBackground(int color)

    public void setForeground(int color)

    public void invokeLater(Runnable r)

    @Override
    public void setEnabled(boolean b)
```

```
public abstract class AreYouSureDialog extends ModalDialog
```

Sztereotipia: boundary
Feladat: ”Biztos-e a dolgában” célú modális párbeszédpanel, Yes és No gombokkal.

```
    public AreYouSureDialog(String title, int width, int height);

    protected abstract void yesClicked()
```

Yes -re kattintás esetén ezen metódus implementációja hívódik meg.

```
    protected abstract void noClicked()
```

No -ra kattintás esetén ezen metódus implementációja hívódik meg.

---

package szoftverfolyamat.osz.game.view;
Az MVC architektura View komponensét specifikáló interfészek és implementáló osztályok kerültek ebbe a csomagba. Ezeket hívja meg a Controller, hogy rajzolják ki a pályaelemeket.

```
class TerminalPanel extends GenericPanel
```

Sztereotipia: konténer
Feladat: A játék-képernyő jobb oldalán elhelyezett, a prgramkód módosítására illetve megtekintésére szolgáló függőleges panel.

```
    private final QuitTerminalButton quitTerminalButton
    private final ResetCodeButton resetCodeButton
    private final CommitCodeButton commitCodeButton
    private final ScrollableEditorPane sep
    private boolean active

    public TerminalPanel()

    void setActive(boolean b)
```

Ha inaktív, akkor nem szerkeszthető, különben szerkeszthető.

```
    boolean isActive()
```

```
void setSource(Iterable<SourceFragment> source)
```
A forráskódot itt lehet megadni.

```
void setControllerIf(ViewTerminalCallsGameCtrl i)
```
Interfész megadása, ld. még ViewTerminalCallsGameCtrl.

```
class ResetCodeButton extends TerminalButton
```
Sztereotipia: boundary
Feladat: A kezdeti pálya-forráskódot visszaállító gomb.

```
private Iterable<SourceFragment> src

public ResetCodeButton(ScrollableEditorPane sep)

@Override
protected void onClick()

public void setSource(Iterable<SourceFragment> source)
```

```
abstract class TerminalButton extends GenericButton
```
Sztereotipia: boundary
Feladat: A terminál-panel gombajinak az őse, közös segéd-metódusokkal. Ld. még GenericButton.

```
protected final ScrollableEditorPane scrollableEditorPane;
protected ViewControlsCallsGameCtrl gci;
protected ViewTerminalCallsGameCtrl gti;

public TerminalButton(String c, ScrollableEditorPane sep)

@Override
protected void onClick()

void setControllerIf(ViewControlsCallsGameCtrl i)
```
Interfészek megadása, ld. még ViewTerminalCallsGameCtrl és ViewControlsCallsGameCtrl.

```
void setControllerIf(ViewTerminalCallsGameCtrl i)
```

```
class RestrictedEditFilter extends DocumentFilter
```
Sztereotipia: viselkedést szabályoz
Feladat: A forrás szerkesztését korlátozó implementáció, ld. még javax.swing.text.DocumentFilter. Megadjuk, maximum hány soros lehet egy szerkeszthető forráskód fragmentum.

```
final private int maxRows

public RestrictedEditFilter(int t)

@Override
public void insertString(DocumentFilter.FilterBypass fb, int offset,
                         String string, AttributeSet attrs)

@Override
public void remove(DocumentFilter.FilterBypass fb, int offset, int length)

@Override
public void replace(DocumentFilter.FilterBypass fb, int offset,
                    int length, String string, AttributeSet attrs)
```

```
class ScrollableEditorPane extends JScrollPane
```

Sztereotipia: konténer

Feladat: A terminál panel részét képzi. Görgethető panel, amelyen nézegethető illetve szerkeszthető a forráskódja a pályáknak.

```
    private boolean active
    private final GenericPanel panel
    private ArrayList<String> sources
    private ArrayList<JTextArea> textAreas
    private ArrayList<JTextArea> allTextAreas

    public ScrollableEditorPane()

    void setActive(boolean b)
```

Ha inaktív, akkor nem szerkeszthető, különben szerkeszthető.

```
    boolean isActive()

    void setSource(Iterable<SourceFragment> sfa)
```

A forráskódot itt lehet megadni és lehívni.

```
    String getSource()
```

```
public interface MenuCallsView
```

Ezen az interfészen keresztül hívja a játék-panelt az ő konténere, tipikusan a fő JFrame. Egy eseményt ad át: az átméretezést deltaX, Y -nal.

```
    void resizePanel(int dx, int dy)
```

```
public class GamePanel extends GenericPanel
            implements GameCtrlCallsViewLifecycle, GameCtrlCallsViewGraphics,
                       GameCtrlCallsViewInfoPanel, MenuCallsView
```

Sztereotipia: konténer, boundary

Feladat: A játék-képernyő bal felén/közepén lévő, grafikus játékfelület; ide kerül a játékos, a pálya, a pályaelemek stb. Innen fogad billentyűeseményeket az adapter.

```
    private final InfoPanel infoPanel
    private final GameArea gameArea
    private final TerminalPanel terminalPanel
    private final GameAreaKeyAdapter gaka
    private ViewControlsCallsGameCtrl gccIf
    private boolean terminalActiveBeforePause

    public GamePanel()

    public void setControlInterfaces(ViewTerminalCallsGameCtrl c1,
                                     ViewControlsCallsGameCtrl c2)
```

MenuCallsView implementáció következik, metódusokat lásd az interfész leírásánál.

```
    public void resizePanel(int dx, int dy)
```

GameCtrlCallsViewLifecycle implementáció következik, metódusokat lásd az interfész leírásánál.

```
@Override
public void enterTerminal()

@Override
public void exitTerminal()

@Override
public void exit()

@Override
public void pause()

@Override
public void resume()

@Override
public void playerDied(String msg)

@Override
public void defeat(String msg)

@Override
public void victory(String msg)

@Override
public void levelCompleted(String msg)

@Override
public void gameAborted()

@Override
public void setSource(Iterable<SourceFragment> src)
```

GameCtrlCallsViewGraphics implementáció következik, metódusokat lásd az interfész leírásánál.

```
@Override
public void initStage(int width, int height, int color)

@Override
public Sprite registerSprite(double width, double height, Font f, String s)

@Override
public Sprite registerSprite(ImageIcon[] imgarray)

@Override
public void moveCanvas(double x, double y)
```

GameCtrlCallsViewInfoPanel implementáció következik, metódusokat lásd az interfész leírásánál.

```
@Override
public void setPlayerInfo(String msg)

@Override
public void setLevelInfo(String msg)

@Override
public void setTime(String hour, String min, String sec)
```

```
    @Override
    public void setScore(String msg)

    @Override
    public void gameSaved(String msg)

    @Override
    public void runtimeError(String msg)

    @Override
    public void ping()
```

`public interface` **`Sprite`**

A képernyőn megjelenő mozgatható és animálható ikonokat képviselő interface. A Controller irányítja ezeket.

`void` **`moveTo`**`(double x, double y)`
Áthelyezés az x, y pontba.

`void` **`moveBy`**`(double dx, double dy)`
*Elmozdítás dx, dy, vektorra*l.

`void` **`setImage`**`(int i)`
A regisztrációkor megadott kép-tömb i -dik elemét allítja be látható képként.

`public class` **`SpriteImpl`** `extends JLabel implements Sprite`
Sztereotipia: egyéb
Feladat: A Sprite interfész implementációja.

```
    private final ImageIcon[] icons

    public SpriteImpl(ImageIcon[] ia)

    @Override
    public void moveTo(double x, double y)

    @Override
    public void moveBy(double dx, double dy)

    @Override
    public void setImage(int i)
```

`class` **`InfoPanel`** `extends GenericPanel`
`    implements GameCtrlCallsViewInfoPanel`
Sztereotipia: konténer
Feladat: A játék-képernyő alsó sávja. Itt látható, az idő, a pontszám, a játékos és a pálya neve is.

```
    private final int TEXT_COLOR
    private final int BG_COLOR
    private final InfoCaptionLabel timeCaption;
    private final MonospaceLabel timeValue
    private final InfoCaptionLabel playerCaption
    private final InfoCaptionLabel playerInfo
    private final InfoCaptionLabel levelCaption
    private final InfoCaptionLabel levelValue
    private final InfoCaptionLabel scoreCaption
    private final MonospaceLabel scoreValue
```

```
private final InfoCaptionLabel messageLabel
private final PauseButton pauseButton
private final BackToMainMenuButton bmmButton
private final RestartLevelButton restartLevelButton

public InfoPanel()

void setControllerIf(ViewControlsCallsGameCtrl i)
```

GameCtrlCallsViewInfoPanel implementáció következik, metódusokat lásd az interfész leírásánál.

```
@Override
public void setPlayerInfo(String s)

@Override
public void setLevelInfo(String s)

@Override
public void setTime(String hour, String min, String sec)

@Override
public void setScore(String s)

@Override
public void gameSaved(String s)

@Override
public void ping()

@Override
public void runtimeError(String s)
```

Belső segédosztályok a feliratokhoz.

```
class InfoCaptionLabel extends GenericLabel
    public InfoCaptionLabel(int col, String cap, boolean b)

class MonospaceLabel extends GenericLabel
    public MonospaceLabel(int col, String cap, boolean b)

class CaptionPanel extends GenericPanel
    public CaptionPanel(GenericLabel ul, GenericLabel ll)
```

`class GameArea extends GenericPanel implements GameCtrlCallsViewGraphics`

Sztereotipia: konténer

Feladat: A játékfelület tényleges JPanel -leszármazott komponense, amely Focus és Keyboard eseményeket fogadhat, amennyiben aktív. Egy görgethető belső komponenst is vezérel.

```
private boolean active
private ScrollablePanel panel
private JScrollPane scrollpane
private final GameAreaKeyAdapter gaka

public GameArea(GameAreaKeyAdapter a)

void setActive(boolean b)

boolean isActive()

protected void processFocusEvent(FocusEvent e)
```

```
private void clear(int width, int height)
```
”Takarító” belső segédmetódus.

GameCtrlCallsViewGraphics implementáció következik, metódusokat lásd az interfész leírásánál.

```
@Override
public void initStage(int w, int h, int color)

@Override
public Sprite registerSprite(ImageIcon[] imga)

@Override
public void moveCanvas(double x, double y)

@Override
public Sprite registerSprite(double w, double h, Font f, String s)
```

```
class ScrollablePanel extends GenericPanel implements Scrollable
```
Sztereotipia: konténer
Feladat: Görgethető kopmonens implementáció, lásd a Scrollable Swing komponenst.

```
private int pw;
private int ph;

public ScrollablePanel(int width, int height)

@Override
public Dimension getPreferredSize()


@Override
public Dimension getPreferredScrollableViewportSize()

@Override
public int getScrollableUnitIncrement(Rectangle visibleRect,
                                      int orientation,
                                      int direction)

@Override
public int getScrollableBlockIncrement(Rectangle visibleRect,
                                       int orientation,
                                       int direction)

@Override
public boolean getScrollableTracksViewportWidth()

@Override
public boolean getScrollableTracksViewportHeight()
```

```
public interface GameCtrlCallsViewLifecycle
```
A Controller ezen az interfészen keresztül hívja meg a View életckilusához kötődő funkcióit.

```
void enterTerminal()
```
A játékos belépett a számítógéphez.

```
void exitTerminal()
```
A játékos kijelentkezett.

`void exit()`
A játékos kilép az alkalmazásból.

`void pause()`
A játék szünetel.

`void resume()`
Szünet vége.

`void defeat(String message)`
A játékos kikapott.
`void victory(String message)`
A játékos nyert.

`void levelCompleted(String message)`
A játékos teljesített egy szintet.

`void gameAborted()`
A játékos megszakította a játékot.

`void setSource(Iterable<SourceFragment> src)`
Ez a kód lett le futtattva.

`void playerDied(String message)`
A játékos meghalt.

`public interface` **`GameCtrlCallsViewInfoPanel`**
A Controller ezen az interfészen keresztül hívja meg az InfoPanel információs felületet.

A következő öt metódus nevéből és paraméterezéséből adódik a funkció:
`void` **`setPlayerInfo`**`(String s)`

`void` **`setLevelInfo`**`(String s)`

`void` **`setTime`**`(String hour, String min, String sec)`

`void` **`setScore`**`(String s)`

`void` **`gameSaved`**`(String s)`

`void` **`runtimeError`**`(String s)`
A Controller oldalán bekövetkezett hibát jelzi; főleg tesztelési célból.

`void` **`ping`**`()`
Igazából „refreshPanel()” lenne a jó neve.

`public interface` **`GameCtrlCallsViewGraphics`**
A Controller ezen az interfészen keresztül hívja meg a grafikai felületet.

`void` **`initStage`**`(int width, int height, int bgcolor)`
Inicializálja a megadott méretű és háttérszínű játékfelületet.

`Sprite` **`registerSprite`**`(ImageIcon[] imgarray)`
Regisztrál egy Sprite-ot, ahol az átadott kép-tömb az animációs fázisokat tartalmazza.
A return value -t használja majd a Controller.

```
Sprite registerSprite(double width, double height, Font f, String text)
```
N/A

```
void moveCanvas(double x, double y)
```
N/A

```
class RestartLevelButton extends TerminalButton
```
Sztereotipia: boundary
Feladat: A szint újrakezdésére szolgáló gomb, ld. még TerminalButton.

```
public RestartLevelButton()

@Override
protected void onClick()
```

```
class BackToMainMenuButton extends TerminalButton
```
Sztereotipia: boundary
Feladat: A főmenübe visszatérésre szolgáló gomb, ld. még TerminalButton.

```
public BackToMainMenuButton()

@Override
protected void onClick()
```

```
class PauseButton extends TerminalButton
```
Sztereotipia: boundary
Feladat: A szünet gomb, ld. még TerminalButton.

```
private boolean paused

public PauseButton()

@Override
protected void onClick()
```

```
class CommitCodeButton extends TerminalButton
```
Sztereotipia: boundary
Feladat: A beirt kódot lefuttató gomb, ld. még TerminalButton.

```
public CommitCodeButton(ScrollableEditorPane sep)

@Override
protected void onClick()
```

```
class QuitTerminalButton extends TerminalButton
```
Sztereotipia: boundary
Feladat: A terminálból való kilépésre szolgáló gomb, ld. még TerminalButton.

```
public QuitTerminalButton(ScrollableEditorPane sep)

@Override
protected void onClick()
```

`class` **`GameAreaKeyAdapter`** `extends KeyAdapter`
Sztereotipia: vezérlő, egyéb
Feladat: A billentyűfigyelő komponens, ld. még KeyAdapter.

```
    private final char NORTH_KEY
    private final char SOUTH_KEY
    private final char WEST_KEY
    private final char EAST_KEY
    private final char USE_KEY
    private byte keysDown
    private byte prevKeysDown
    private ViewControlsCallsGameCtrl gci

    public GameAreaKeyAdapter()

    void setControllerIf(ViewControlsCallsGameCtrl i)
```
Ezen az interfészen értesíti a Controllert a billentyűeseményről.

```
    void focusRestored()
```
Fókusz visszanyerésekor hivják meg.

```
    void stopMoving()

    @Override
    public void keyPressed(KeyEvent e)

    @Override
    public void keyReleased(KeyEvent e)

    @Override
    public void keyTyped(KeyEvent e)
```

Két belső segédosztály.

```
    private boolean areThreeOrMoreBitSet(byte b)

    private ViewControlsCallsGameCtrl.Direction getDirectionForByte(byte b)
```

`package szoftverfolyamat.osz.menu;`
Ebbe a csomagba kerül a bejelentkező képernyőt felépítő és kezelő Főmenü komponens. Ez hozza létre a View -et és a Controller-t és ez indítja el őket. Továbbá, használja a perzisztenciát implementáló Model komponenst.

`public interface` **`GameCtrlCallsMenu`**
A Főmenüt a Controller ezen át hívja vissza.

```
    void backToMainMenu()
```
Vége a játéknak, újra menü.

```
    void loadGame()
```
Vége a játéknak, mentett játékot töltünk be.

```
    void startMission()
```
Vége a játéknak, új játékot kezdünk előről.

```
    void error(String errorMsg)
```

`public class` **`Launcher`**
Az indító osztály.

```
public static void main(String[] sa)
```

`class` **`HighScoresPanel`** `extends GenericPanel`
Sztereotipia: konténer
Feladat: A legmagasabb pontszámokat kilistázó panel.

```
public HighScoresPanel(String[] scores)
```

`class` **`ButtonBar`** `extends GenericPanel`
Sztereotipia: konténer
Feladat: Gombokat tartalmazó panel.

```
final MainWindow mainWindow
```

```
public ButtonBar(MainWindow mw)
```
A parent a konstruktor paramétere

```
void addSpacer()
```
Elválasztó térköz hozzáadása.

```
void addButton(GenericButton gb)
```

`abstract class` **`MenuButton`** `extends GenericButton`
Sztereotipia: boundary
Feladat: Menü-gomb specializált megjelenéssel és viselkedéssel.

```
public MenuButton(String caption)
```

`class` **`MainWindow`** `extends JFrame implements GameCtrlCallsMenu`
Sztereotipia: boundary
Feladat: Ez a főablak, egy JFrame, ezen belül funkcionál az egész program. Két üzemmódja van: játék- és menümód.

```
private GenericPanel contentPanel
private final MainWindow thiz
private final MenuCallsDao menuCallsDao
```

```
public MainWindow(int width, int height, MenuCallsDao mcd)
```

```
private void mopup()
```
Belső segédmetódus, ami kitakarít.

```
void setMainMenu()
```
Menümódba lépés

```
void setGaming()
```
Játékmódba lépés.

GameCtrlCallsMenu implementáció következik, metódusokat lásd az interfész leírásánál.

```
@Override
public void backToMainMenu()

@Override
public void loadGame()

@Override
public void startMission()

@Override
public void error(String s)
```

`class` **`BogusPi`** `extends ModalDialog`
Sztereotipia: boundary
Feladat: Folyamatindikátor, ami nem indikál semmit, csak úgy csinál, és egy megadott idő múlva leáll.

```
private int secs;

public BogusPi(int secs)
```

`package szoftverfolyamat.osz.model;`
Ebbe a csomagba kerülnek a perzisztenciát definiáló interfészek, és azok megvalósításai.

`public interface` **`SourceFragment`**
Forráskod töredék. Minden pálya forrása ilyen töredékekből áll.

`boolean` **`isVisible`**`()`
Ha rejtett, akkor a játékos nem látja.

`boolean` **`isReadonly`**`()`
Irásvédett-e.

`String` **`getCode`**`()`
Maga a kod szövege.

`int` **`getMaxRows`**`()`
Legfeljebb ennyi soros lehet.

`public interface` **`Configuration`**
A konfiguracios beallitasokat jelkepező interface. Jelenleg a nyelvet, az ablak szélességet es magassagat lehet megadni a getterekkel es a setterekkel. Ezeket az implementacio perzisztensen kezeli.

```
public String getLanguage();

public void setLanguage(String l)
Nyelv megadasa es lekerdezese

public int getWindowWidth();
Ablak szélesség megadasa es lekerdezese

public void setWindowWidth(int w)
```

```
    public int getWindowHeight()
    Ablak magassag megadasa es lekerdezese

    public void setWindowHeight(int h)
```

```
public class ConfigurationFile extends PropertiesReader
    implements Configuration
```

Sztereotipia: entity
Feladat: A konfigurációs beállításokat jelképező interface megvalósítása.

```
    public ConfigurationFile()

    protected void checkConfigurationFile(String path)

    @Override
    public String getLanguage()

    @Override
    public void setLanguage(String l)

    @Override
    public int getWindowWidth()

    @Override
    public void setWindowWidth(int w)

    @Override
    public int getWindowHeight()

    @Override
    public void setWindowHeight(int h)
```

```
public interface Mission
```

Egy betoltheto kuldetes, amelyben az Iterator felsőrolja az egymas utan kovetkezo pályakat.

```
    String getName()

    String getInfo()

    Iterable<Level> getLevels()
```

felsőrolja az egymas utan kovetkezo pályakat. Egyik pálya teljesitese utan jon a kovetkezo.

```
public interface GameStub
```

Egy elmentett játék bejegyzese. Eloszor csak ezek lesznek betoltve, a játékos ezek kozul valaszt, aztan az id alapjan betolti a tenyleges Game -et. NINCS EGYELŐRE IMPLEMENTÁLVA!

```
    String getPlayerName()

    int getScore()

    String getNote()

    String getDate()

    long getId()
```

`public interface` **`Level`**

Egy pályat reprezentalo interface; a tenyleges pályat a pálya-programozo keszíti el.

`String` **`getName`**`()`

`String` **`getInfo`**`()`

`Iterable<SourceFragment>` **`getFragments`**`()`
A forraskod-töredékek sorozata.

`public interface` **`Game`**

Egy elmentett es betoltheto jatszma. NINCS IMPLEMENTÁLVA MÉG!

`Mission` **`getMission`**`()`

`Object` **`getCurrentLevel`**`()`

`public interface` **`MenuCallsDao`**

Ezen az interfészen át hívja meg a Főmenü a Modelt (A Modellre korábban DAO-két hivatkoztunk)

`Iterable<Mission>` **`getMissions`**`()`
Elérhető küldetések listázása.

`String[][]` **`getHighScores`**`()`
Legmagasabb pontszamokat listazza ki.

`<Iterable>GameStub` **`getSavedGames`**`(String player)`
Mentett jatszmak listajanak lekerese; ha a player==null, akkor az osszeset kilistazza, kulonben csak az adott nevu játékoset.

`Game` **`loadGame`**`(long gameId)`
Adott azonositoju jatszma betoltese.

`int` **`errno`**`()`
IO error, 0 ha OK.

`public interface` **`GameCallsDao`**

Ezen az interfészen át hívja meg a Controller a Modelt (A Modellre korábban DAO-két hivatkoztunk)

`void` **`saveGame`**`(Game game)`
Folyamatban levo jatszma elmentese.

`void` **`saveScore`**`(int score, String player)`
játék vegen a pontszam es a játékos nevenek mentese.

`int` **`errno`**`()`
IO error, 0 ha OK.

---

public class **MissionImpl** implements Mission

Sztereotipia: entity

Feladat: Egy Mission (= küldetés) implementációja

```
    private final Iterable<Level> al
    private final String name
    private final String info

    public MissionImpl(Iterable<Level> l, String n, String i)

    @Override
    public String getName()

    @Override
    public String getInfo()

    @Override
    public Iterable<Level> getLevels()
```

---

public class **LevelImpl** implements Level

Sztereotipia: entity

Feladat: Egy Level (= pálya) implementációja

```
    private final String info
    private final String name
    private final Iterable<SourceFragment> fragments

    public LevelImpl(String n, String i, Iterable<SourceFragment> f)

    @Override
    public String getName()

    @Override
    public String getInfo()

    @Override
    public Iterable<SourceFragment> getFragments()
```

---

class **MissionsReader** extends PropertiesReader

Sztereotipia: entity / helper

Feladat: Beolvassa a rendelkezésre álló küldetéseket.

```
    private final Vector<Mission> missions

    public MissionsReader()

    Iterable<Mission> getMissions()
```

---

class **LevelsReader**

Sztereotipia: entity / segéd

Feladat: Beolvassa a rendelkezésre álló pályákat.A binaries/data/level.xml fajlokban vannak a pályak javascript kodjai; ezeket tolti be a program inditaskor. Alapertelmezetten az elso Mission indul el, lasd missions.properties fajl. Ezek a fajlok szerkeszthetok.*

```
    public int errno
    private DocumentBuilderFactory dbf
```

```
    public LevelsReader()

    public Level getLevel(String id)

    private String stripLeadingAndTrailingNewlines(String s)
```

---

class **PropertiesReader**

Sztereotipia: entity / segéd

Feladat: Java properties file-ok irására és olvasására szolgáló segédosztály. A metódusok nevei és paraméterei utalnak a funkciókra. Saját konstans IO-hibakódokat is definiál.

```
    public static final int OK
    public static final int FILE_NOT_FOUND
    public static final int FILE_NOT_WRITABLE
    public static final int FILE_NOT_READABLE
    public static final int UNSPECIFIED_ERROR
    public static final String SEPARATOR
    public static final String USER_DIR
    public int errno
    protected String path

    public PropertiesReader()

    protected Properties loadProperties()
    protected void writeProperties(Properties prop)

    protected String getStringProperty(String key)

    protected void writeStringProperty(String key, String s)

    protected int getIntProperty(String key)

    protected void writeIntProperty(String key, int t)
```

---

public class DaoForMenu implements MenuCallsDao

Sztereotipia: entity / segéd

Feladat: MenuCallsDao implementáció, a metódusokat lásd az interfész leírásánál.

```
    private int _errno;

    public DaoForMenu()

    @Override
    public Iterable<Mission> getMissions()
```
N/A
```
    @Override
    public String[][] getHighScores()

    @Override
    public <Iterable>GameStub getSavedGames(String player)
```
N/A
```
    @Override
    public Game loadGame(long gameId)
```
N/A
```
    @Override
    public int errno()
```

---

package szoftverfolyamat.osz.game.controller;

Ebbe a csomagba kerülnek az MVC Controller komponensét definiáló interfészek és az őket megvalósító osztályok. A Controller eseményvezérelt állapotgépként működik.

---

public interface **JsCallsController**
A Javascript kód ezen az interfészen keresztül hívja meg az eseményeket feldolgozó Controllert.

void **initialize**(int width, int height, int bgColor)
Inicializalja a pályát; legelőször ezt kell meghivni.

void **setPlayer**(int x, int y, int w, int h)
Elhelyezi a játékost a pályán.
w: objektum szélesség
h: objektum magassaga
x: objektum bal felső sarkanak x koordinataja
y: objektum bal felső sarkanak y koordinataja

void **createWall**(String klass, int x, int y, int w, int h)
Fal-elemet (~teglat) rak ki a pályára.
klass: YELLOW_BRICK, RED_BRICK, DARK_BRICK, STONE_BRICK valamelyike
w: objektum szélesség
h: objektum magassaga
x: objektum bal felső sarkanak x koordinataja
y: objektum bal felső sarkanak y koordinataja

void **createComputer**(int x, int y, int w, int h)
Szamitogepet hoz létre a pályán.
w: objektum szélesség
h: objektum magassaga
x: objektum bal felső sarkanak x koordinataja
y: objektum bal felső sarkanak y koordinataja

void **createExit**(int x, int y, int w, int h)
Kijaratot hoz létre a pályán.
w: objektum szélesség
h: objektum magassaga
x: objektum bal felső sarkanak x koordinataja
y: objektum bal felső sarkanak y koordinataja

void **createSavePoint**(int x, int y, int w, int h)
Mentesi pontot hoz létre a pályán. A mentes meg nem elerheto.
w: objektum szélesség
h: objektum magassaga
x: objektum bal felső sarkanak x koordinataja
y: objektum bal felső sarkanak y koordinataja

```
void createGhost(int w, int h, int pxs[], int pys[], int speed)
```
Mozgo szellemet hoz létre a pályán.
w: objektum szélesség
h: objektum magassaga
psx: a szellem altal bejart pontok x koordinatai
psy: a szellem altal bejart pontok y koordinatai
speed: mozgas gyorsasaga (1 gyors, 10 lassu)

```
void createTrap(int x, int y, int w, int h)
```
Csapdat hoz létre a pályán. Ha ralep a játékos, akkor meghal.
w: objektum szélesség
h: objektum magassaga
x: objektum bal felső sarkanak x koordinataja
y: objektum bal felső sarkanak y koordinataja

```
void registerTickHandler(Object o)
```
Nincs implementalva.

```
void registerCollisionHandler(Object o)
```
Nincs implementalva.

```
public interface MenuCallsGameCtrl
```
A Főmenü ezen át hívja a Controllert; a metódusok neve és paraméterezése jelzi a funkciókat.

```
void exitProgram()

void pause()

void resume()

void start(Mission m, String playerName, GameCtrlCallsMenu cbi)

void start(Game m, String playerName, GameCtrlCallsMenu cbi)
```

```
public interface ViewControlsCallsGameCtrl
```
A View komponens kontrolljai (billentyűzet, buttonok) ezen át hívja a Controllert; a metódusok neve és paraméterezése jelzi a funkciókat.

```
enum Direction
```
Az irányító billentyűk által megadott mozgási iránya a játékosnak.

```
final static int BACK_TO_MAIN_MENU
final static int LOAD_GAME
final static int SELECT_MISSION
```
Mi történjen game over után?

```
void abortGame()

void movePlayer(Direction d)

void activate()
```
Pálya-elem aktiválása.

```
void pause()

void levelCompletionConfirmed()
```

```
void gameoverConfirmed(int furtherAction)

void restartLevel()
```

`public interface` **`ViewTerminalCallsGameCtrl`**

A View komponens Terminálja ezen át hívja a Controllert.

```
void executeCode(String code)
```
Kód futtatása.

```
void exitTerminal()
```
Kilépés a terminálból

`class` **`SpriteFarm`** `implements BoundingShapeObserver`

Sztereotipia: entity / vezérlő

Feladat: Sprite-okat tároló és mozgató adatbázis, ami a mozgatás illetve átméretezés hatására jelzi az ütközéseket. Ld még BoundingShapeObserver.

```
private final Hashtable<AbstractSprite, Point2D.Double> sprites
private final Vector<CollisionHandler> collisionHandlers
private final Vector<CollisionHandler> nocollisionHandlers

public SpriteFarm()

@Override
public synchronized void changed(BoundingShape shape)

synchronized public Collection<AbstractSprite>
moveBy(AbstractSprite s, Vector2D v)
```
Relatív mozgatás, illetve felvétel az adatbázisba.

```
synchronized public Collection<AbstractSprite>
moveTo(AbstractSprite s,  double x, double y)
```
Abszolut mozgatás, illetve felvétel az adatbázisba.

```
synchronized void addCollisionHandler(CollisionHandler h)

synchronized void addNocollisionHandler(CollisionHandler h)
```
Ha egy esemény hatására nincs ütközés, akkor ezek lesznek meghívva.

```
synchronized void remove(AbstractSprite s)
```
Törlés az adatbázisból

```
private Collection<AbstractSprite>
checkCollisions(AbstractSprite sp, int ctype)
```

`interface` **`CollisionHandler`**

Ütközéseket figyelő objektumpéldányok implementálják.

```
final int MOVED
final int SHAPE_CHANGED
```
Esemény-azonosító

```
void collision(AbstractSprite s, AbstractSprite changed, int t)
```
elzi, hogy s ütközött a mozgó vagy. megváltozott changed-del, t tipusu eseménykor.

---

class **Vector2D** extends Point2D.Double
Segédosztály - pont helyett vektorra.

```
public Vector2D()

public Vector2D(double x, double y)

public void setCoordinates(double x, double y)

public Vector2D getReverse()

public Vector2D getScaledCopy(double r)
```

---

class **JavascriptHost**
Sztereotipia: vezérlő
Feladat: Javascriptet futtató osztály.

```
private final ScriptEngine engine

public JavascriptHost()

void addReference(String s, Object o)
```
Ezzel lehet átadni a javascriptnek Java-ban implementált objektumokat, amelyeket a JS meg tud hívni runtime.

```
void removeReference(String s)

String runScript(String s)
```
Script futtatása.

---

abstract class **Ticker**
Sztereotipia: vezérlő
Feladat: Időzítő - óra. Előre magadott időt számolja vissza, szüneteltethető, a belső ideje módosítható.

```
private enum State
private Timer timer
private State state
private int runningTime
private long interval

public Ticker()

synchronized void start(int secs, long decrement)
```
A sec -rol csokkenti decrement -tel. Decrement-enként egy tick();

```
synchronized void pause()

synchronized void kontinue()

synchronized void dispose()

synchronized void decreaseTime(int s)
```
A *belső ideje csökken s -másodperccel.*

```
int getSecondsLeft()
```

```
    protected abstract void timeout()
```
Ez hívódik meg, ha lejárt az idő.

```
    protected abstract void tick()
```
Ez hívódik meg minden egyes tick -nél.

```
interface TickListener
    void tick()
```

```
class TickListenerSet implements TickListener
```
Sztereotipia: konténer
Feladat: TickListener interface-eket regisztrál és értesít a tick eseményekről.

```
    private HashSet<TlEntry> entries
```
A bejegyzésekben a listenerek.

```
    public TickListenerSet()

    synchronized void add(TickListener tl, int pc)
```
TickListener -t regisztrál: pc tick-enként hívja meg.

```
    @Override
    synchronized public void tick()

    class TlEntry implements TickListener
```
Bejegyzés egy TickListener számára.

```
        private final TickListener tickListener
        private final int perCount
        private int counter

        public TlEntry(TickListener l, int pc)

        @Override
        public void tick()
```

```
class PlayerTickListener implements TickListener
```
Sztereotipia: egyed
Feladat: A játékos számára fenttartott TickListener.

```
    private final double delta
    private final SpriteFarm spriteFarm
    private final Sprite playerSprite
    private final AbstractSprite player
    private ViewControlsCallsGameCtrl.Direction direction

    public PlayerTickListener(SpriteFarm sf, Sprite ps, AbstractSprite p)

    void setDirection(ViewControlsCallsGameCtrl.Direction d)

    @Override
    public void tick()

    private boolean containsWall(Collection<AbstractSprite> c)
```
Belső segédosztály.

```
class ScoreTickListener implements TickListener
```

Sztereotipia: egyed
Feladat: A pontszám alakítására fenttartott TickListener.

```
    private final Ticker ticker
    private final InternalEvents ieif
    private final GameCtrlCallsViewInfoPanel vipIf
    private int score

    public ScoreTickListener(Ticker t, GameCtrlCallsViewInfoPanel i,
                             InternalEvents ie)

    @Override
    public synchronized void tick()

    public void decreaseScore(int t)

    private void checkScore()

    private void setScore(int sc)
```

```
abstract class AbstractSpriteImpl implements AbstractSprite
```

Sztereotipia: egyed
Feladat: AbstractSprite implementáció. Ld. még AbstractSprite.

```
    private Dimension primaryBoundingBox
    private BoundingShapeObserver boundingShapeObserver

    protected void setPrimaryBoundingBox(Dimension p)

    int getHeight()

    int getWidth()

    @Override
    public Dimension getPrimaryBoundingBox()

    @Override
    public Rectangle2D[] getRefinementBoundingBoxes()

    @Override
    public void setBoundingShapeObserver(BoundingShapeObserver o)
```

```
abstract class BitmapSpriteImpl extends AbstractSpriteImpl
    implements BitmapSprite
```

Sztereotipia: egyed
Feladat: BitmapSprite implementáció, ikon/képbetöltési funkcióval.

```
    private ImageIcon[] icons

    static ImageIcon[] loadIcons(Class klass, String[] filenames, int w, int h)

    static ImageIcon[] loadSingleIcon(Class klass, String fn, int w, int h)

    protected void setIcons(ImageIcon[] p)

    protected ImageIcon getIcon()

    @Override
    public ImageIcon[] getIcons()
```

```
final class DefaultWallFactory
```

Sztereotipia: Helper
Feladat: Különbözően dekorált Wall (fal) objektumokat legyártó factory. Ld. még Wall.

```
    private static Hashtable<Dimension, ImageIcon> redBricksTable
    private static Hashtable<Dimension, ImageIcon> darkBricksTable
    private static Hashtable<Dimension, ImageIcon> yellowBricksTable
    private static Hashtable<Dimension, ImageIcon> stoneBricksTable

    private static BitmapSpriteImpl createBrickWall(String path,
            Hashtable<Dimension, ImageIcon> ht, int width, int height)

    static BitmapSpriteImpl createRedBrickWall(int width, int height)

    static BitmapSpriteImpl createYellowBrickWall(int w, int h)

    static BitmapSpriteImpl createDarkBrickWall(int w, int h)

    static BitmapSpriteImpl createStoneBrickWall(int w, int h)

    static class BrickWall extends BitmapSpriteImpl
                            implements BitmapSprite, Wall
```

Belső implementációs osztály.

```
final class Computer extends BitmapSpriteImpl implements Device, AllowsCollision
```

Sztereotipia: egyed
Feladat: Ld. még Device, AllowsCollision.

```
    public Computer(int width, int height)

    @Override
    public void activate()
```

```
final class SavePoint extends BitmapSpriteImpl implements Device
```

Sztereotipia: egyed
Feladat: Ld. még Device.

```
    public SavePoint(int width, int height)

    @Override
    public void activate()
```

```
final class Exit extends BitmapSpriteImpl implements AllowsCollision
```

Sztereotipia: egyed
Feladat: Ld. még AllowsCollision.

```
    public Exit(int w, int h)
```

`class` **`Ghost`** `extends BitmapSpriteImpl implements Foe, Moving, Animated`

Sztereotipia: egyed

Feladat: Ld. még Foe, Moving, Animated.

```
    private final int[] frames
    private int seq

    public Ghost(int width, int height)

    @Override
    public int scoreDecrease()

    @Override
    public boolean instantlyKills()

    @Override
    public Dimension getTranslationVector()

    @Override

    public int getNextFrame()
```

`final class` **`Trap`** `extends BitmapSpriteImpl implements Foe`

Sztereotipia: egyed

Feladat: Csapdát reprezentál.

```
    public Trap(int w, int h)

    @Override
    public int scoreDecrease()

    @Override
    public boolean instantlyKills()
```

`final class` **`Player`** `extends BitmapSpriteImpl implements Moving`

Sztereotipia: egyed

Feladat: A játékos figurát implementáló sprite. Ld. még Moving.

```
    public Player(int w, int h)

    @Override
    public Dimension getTranslationVector()
```

`interface` **`InternalEvents`**

A Controller állapotgép belső eseményei; ezeket nem külső hívások váltják ki.

`void` **`playerDied`**`()`
A játékos meghalt.

`void` **`levelCompleted`**`()`
Szint teljesítve.

`String` **`initLevel`**`(String s)`
Inicializálása a pályának.

`void` **`gameOver`**`(boolean victory)`
Játék vége.

`public class` **`EventProcessor`** `implements`

Sztereotipia: vezérlő

Feladat: A Controller oldalon ez az osztály végzi az események feldolgozását.

```
        MenuCallsGameCtrl, ViewTerminalCallsGameCtrl,
ViewControlsCallsGameCtrl, JsCallsController, InternalEvents

private GameCtrlCallsViewInfoPanel vipIf
private GameCtrlCallsViewGraphics vgIf
private GameCtrlCallsViewLifecycle vlcIf
private final GameCtrlCallsMenu mif
private final int TICKS_PER_SECOND = 35
private Ticker ticker
private volatile MainState state
private volatile PlayingState playingState
private Iterator<Level> levels
private Level currentLevel
private TickListenerSet tickListenerSet
private SpriteFarm spriteFarm
private PlayerTickListener playerTickListener
private ScoreTickListener scoreTickListener

private enum MainState
    PRE_START,
    PLAYING,
    AFTERGAME,
    BUILDING_LEVEL,
    INTERMEZZO,
    HALTED


private enum PlayingState
    STROLLING,
    AT_SAVEPOINT,
    AT_COMPUTER,
    PROGRAMMING,
    PAUSED


public EventProcessor(GameCtrlCallsMenu m)

public void setViewInterfaces(GameCtrlCallsViewInfoPanel v1,
                              GameCtrlCallsViewGraphics v2,
                              GameCtrlCallsViewLifecycle v3)

private synchronized void clockTicked()

private synchronized void clockTimeout()
```

`MenuCallsGameCtrl` implementáció következik, metódusokat lásd az interfész leírásánál.

```
@Override
public synchronized void exitProgram()

private PlayingState prePauseState

@Override
public synchronized void pause()
```

```
@Override
public synchronized void resume()

@Override
public synchronized void start(Mission m, String pn, GameCtrlCallsMenu cbi)

@Override
public synchronized void start(Game m, String pn, GameCtrlCallsMenu cbi)

private void startGame(String playerName)
```
Belső segédek.

```
private String getCurrentLevelCode()
```

InternalEvents implementáció következik, metódusokat lásd az interfész leírásánál.

```
public synchronized String initLevel(String source)
@Override
public synchronized void playerDied()

@Override
public synchronized void levelCompleted()

@Override
public synchronized void gameOver(boolean victory)
```

ViewTerminalCallsGameCtrl implementáció következik, metódusokat lásd az interfész leírásánál.

```
@Override
public synchronized void executeCode(String code)

@Override
public synchronized void exitTerminal()
```

ViewControlsCallsGameCtrl implementáció következik, metódusokat lásd az interfész leírásánál.

```
@Override
public synchronized void restartLevel()

@Override
public synchronized void abortGame()

@Override
public synchronized void movePlayer(Direction d)

@Override
public synchronized void activate()

@Override
public synchronized void levelCompletionConfirmed()

@Override
public synchronized void gameoverConfirmed(int furtherAction)
```

`JSCallsController` implementáció következik, metódusokat lásd az interfész leírásánál.

```
@Override
public synchronized void initialize(int w, int h, int bgColor)

@Override
public synchronized void setPlayer(int x, int y, int w, int h)

@Override
public synchronized void createWall(String k, int x, int y, int w, int h)

@Override
public synchronized void createComputer(int x, int y, int w, int h)

@Override
public synchronized void createSavePoint(int x, int y, int w, int h)

@Override
public synchronized void createExit(int x, int y, int w, int h)

@Override
public synchronized void createTrap(int x, int y, int w, int h)

@Override
public synchronized void createGhost(int w, int h, int pxs[],
                                     int pys[], int speed)

@Override
public synchronized void registerTickHandler(Object o)

@Override
public synchronized void registerCollisionHandler(Object o)
```

---

package szoftverfolyamat.osz.game.controller.sprites

A sprite-ok alapvető tulajdonságait reprezentáló interface-ek kerültek ebbe a csomagba.

---

public interface **AbstractSprite** extends BoundingShape

Alap interfész, azt a tulajdonságot mondja ki, hogy az implementáló egyed körvonala megfigyelhető.

```
void setBoundingShapeObserver(BoundingShapeObserver o)
```

---

public interface **BoundingShapeObserver**

Akkor hívják vissza, ha a megfigyelt egyed körvonala módosul.

```
void changed(BoundingShape s)
```

---

public interface **Wall** extends AbstractSprite

Fal, ugyanazokkal az alaptulajdonságokkal mint az AbstractSprite.

---

public interface **Resizable**

Jelzi, hogy az implementáló egyed átméretezehető.

`public interface` **`Animated`**

Jelzi, hogy az implementáló egyed animálható. Az animációs esemény hatására a következő fázist veszi fel az egyed, a metódus meghívására.

```
int getNextFrame()
```

`public interface` **`AllowsCollision`**

Jelzi, hogy az implementáló egyed átfedésbe kerülhet egy másik egyeddel (elg, ha csak az egyik implementálja).

`public interface` **`Moving`**

Jelzi, hogy az implementáló egyed mozog, tehát fenntart egy aktuális mozgási irány vektort.

```
Dimension getTranslationVector()
```

`public interface` **`GlyphSprite`**

Még nem implementált (kép helyett Font alapú sprite lenne).

```
String getFontName()

String getText()

boolean isBackgroundTransparent()

int getBackgroundColor()
```

`public interface` **`Foe`**

Jelzi, hogy az implementáló egyed ellenség, azaz ütközéskor levon a játékos pontjaiból, de meg is ölheti azonnal a játékost.

```
boolean instantlyKills()

int scoreDecrease()
```

`public interface` **`Device`** `extends AbstractSprite, AllowsCollision`

Jelzi, hogy az implementáló egyed egy aktiválható eszköz.

```
void activate()
```

`public interface` **`BoundingShape`**

Jelzi, hogy az implementáló egyed rendelkezik körvonallal.

```
   Dimension getPrimaryBoundingBox()

   Rectangle2D[] getRefinementBoundingBoxes()
```

`public interface` **`BitmapSprite`**

Jelzi, hogy az implementáló egyed képét bitmap adja meg.

```
ImageIcon[] getIcons()
```

3. Implementáció

3.1 Fejlesztőeszközök

A program Java 7-es környezetben lett fejlesztve. A program fejlesztéséhez a NetBeans IDE-t használatuk, illetva az Apache Ant build tool 1.9 -es sorozatát. A képek elkészítéséhez a Gimp-et és az InkScape-et használtuk. A perzisztens adatokat a program közvetlenül a fájlrendszerben tárolja, ezért nem volt szükség relációs vagy objektum elvű adatbázisra.
Az XML és egyéb szöveges fájlok szerkesztése a JEdit szövegszerkesztővel történt.

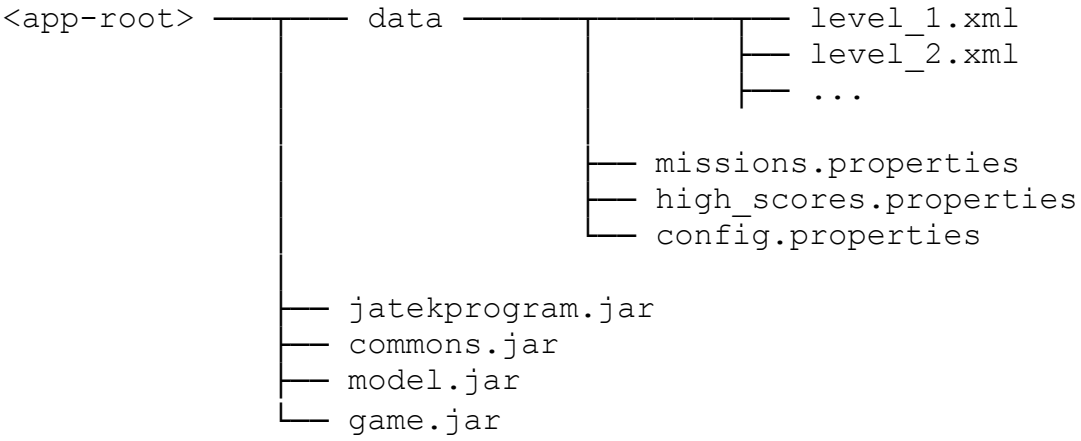
3.2 Forráskód, futtatható kód és egyéb erőforrások

A forráskód a mellékelt `forras.zip` állomány `src` mappájában található. A statikus erőforrások, azaz a betűkészletek és a képek a `res` mappába kerültek, és innen lettek befordítva. A pályák, küldetések és egyéb szöveges fájlok a `misc` mappában találhatók.

A futtatható kódot a telepítő állomány tartalmazza, de külön megtalálhatók a JAR file-ok a `dist.zip` file-ban. A JAR fájlok tartalmaznak minden szükséges erőforrást.

3.2.1 Fizikai architektúra

Ebben a pontban bemutatásra kerülnek a program fizikai komponensei, azaz a programot alkotó fájlok:



A fenti ábra az `<app-root>` gyökérkönyvtár tartalmát mutatja be, ahol
- Az `<app-root>` könyvtár az alkalmazás gyökérkönyvtára, a program csak ebbe a könyvtárba ─ és az alkönyvtárakba ─ ír illetve csak innen olvas ki adatokat. E könyvtár neve lehet például `untrusted_0.1` (ahol a 0.1 a verziószám). Tehát a program Windows-on nem használja a Registry-t és a Linuxokon sem a különböző csomagkezelő vagy rendszerleíró adatbázisokat.
- Az `<app-root>/data` könyvtárba kerülnek a perzisztens adatok, mint például az elmentett játszmák, pályák stb.
- Az `<app-root>/game.jar` fájl a `game` modul.
- Az `<app-root>/model.jar` fájl a perzisztenciáért felelős `model` modul.
- Az `<app-root>/graphics.jar` fájl a grafikai és egyéb segédosztályokat tartalmazza, például a betűkészletek és a képek kezelését végző Helper-eket. Ezt egyaránt használhatja a `menu` és a `game` modul.
- Az `<app-root>/jatekprogram.jar` fájl a főprogram, ez a `menu` modul. Ennek a fájlnak a Manifest-jében szerepel a MainClass bejegyzés, vagyis egy futtatható Java alkalmazásról van szó.

4 Tesztelés

A program típusából kiindulva a specifikáció alapú technika lett felhasználva, azon belül is a használati eset teszt. Ezzel biztosítani lehet a program keretének, a főmenünek és az alapvető játékmenetbeli funkcióknak a működését.

4.1 Új játék kezdés

Elvárt viselkedés:

A programablak nézete átvált a játéktérre. A játéktér a kiértékelés alapján előállított pályát, a módosítandó kódot és a funkciógombokat tartalmazza.

Valódi viselkedés:

Egy felugró ablak jelenik meg ismertetve a funkciót, amit tovább lehet nyomni. Ezután játéktér betöltődik az összes elemével együtt.

Eredmény:

SIkeres.

4.2 Betöltés

Elvárt viselkedés:

Létező mentés esetén új játék kezdődik a mentett sorszámú pályán. Ennek menete az új játék funkcióval megegyező. Nem létező mentés esetén felugró ablak figyelmeztet.

Valódi viselkedés:

Nem létező mentés esetén van figyelmeztetés. Mentést a teszt pillanatában nem lehet létrehozni, így a tényleges betöltés nem tesztelhető.

Eredmény:

Nem működik. A mentés nincs implementálva, így a betöltés nem tesztelhető érdemben.

4.3 Beállítások

Elvárt viselkedés:

Nézet átvált a beállítás menübe, ahol különböző globális és játékmenetbeli dolgokat lehet módosítani. Értékek módosításakor a változás életbe lép.

Valódi viselkedés:

Felugró ablak értesít a funkcióról.

Eredmény:

Nem működik. A beállító menü nincs implementálva.

4.4 Súgó

Elvárt viselkedés:

Nézet átvált a súgó ablakba, ahol a játék rövid leírása és az irányításhoz szükséges gombok felsorolása szerepel.

Valódi viselkedés:

Felugró ablak értesít a funkcióról.

Eredmény:

Nem működik. A beállító menü nincs implementálva.

4.5 Kilépés

Elvárt viselkedés:

Felugró ablak jelenik meg, ahol a kilépési szándékot lehet megerősíteni. Ha az igen gombra nyomunk, a főablak bezáródik. Ellenkező esetben visszatérünk a főmenübe.

Valódi viselkedés:

Megegyezik az elvárt viselkedéssel.

Eredmény:

Sikeres.

4.6 Kód kiértékelés

Elvárt viselkedés:

Helyes kód esetén fusson le a kód és az eredmény jelenjen meg a pályán. Hibás kód esetén jelenjen meg a trace.

Valódi viselkedés:

Megegyezik az elvárt viselkedéssel.

Eredmény:

Sikeres.

4.7 Kódszerkesztés

Elvárt viselkedés:

A kód pályát létrehozó része alap esetben szerkeszthető, ellenkező esetben a letiltott blokkokat elszínezett háttérrel jelzi a program.

Valódi viselkedés:

Az elvárt viselkedés teljesül, a jelzett kódrészlet szerkeszthető, a több pedig nem.

Eredmény:

A funkció működik.

Untrusted kézikönyv

Bevezető

Az évekig fejlesztett mesterséges intelligencia elszabadult szakértő kezeink közül, és elrejtette az egyetlen Algoritmust amellyel meg lehet állítani. Mivel azonban nem tudta teljesen elzárni a kódot, így agyafúrt akadályokat tervezett a rendszerbe. A mi feladatunk, hogy a még hozzáférhető kód átírásával megoldjuk ezeket a fejtörőket és visszaszerezzük az Algoritmust.

A játék azokat a felhasználókat célozza, akik valamilyen szinten tisztában vannak a programozás alapjaival. Azonban a pályák úgy lettek kialakítva, hogy ne legyen szükséges semmilyen nyelvspecifikus tudás. A módosítandó kód Javascript nyelven íródott, így ennek mélyebb ismeretében bővülhet a megoldások tárháza (nem feltétlenül csak egyetlen megoldás létezik).

Környezet

Hardver követelmények

Processzor: 1Ghz

Memória: 1Gb

Videó: openGL 3.0 képes kártya

Szoftveres követelmények

Operációs rendszer: Windows Vista+ / Debian alapú linux disztribúció *

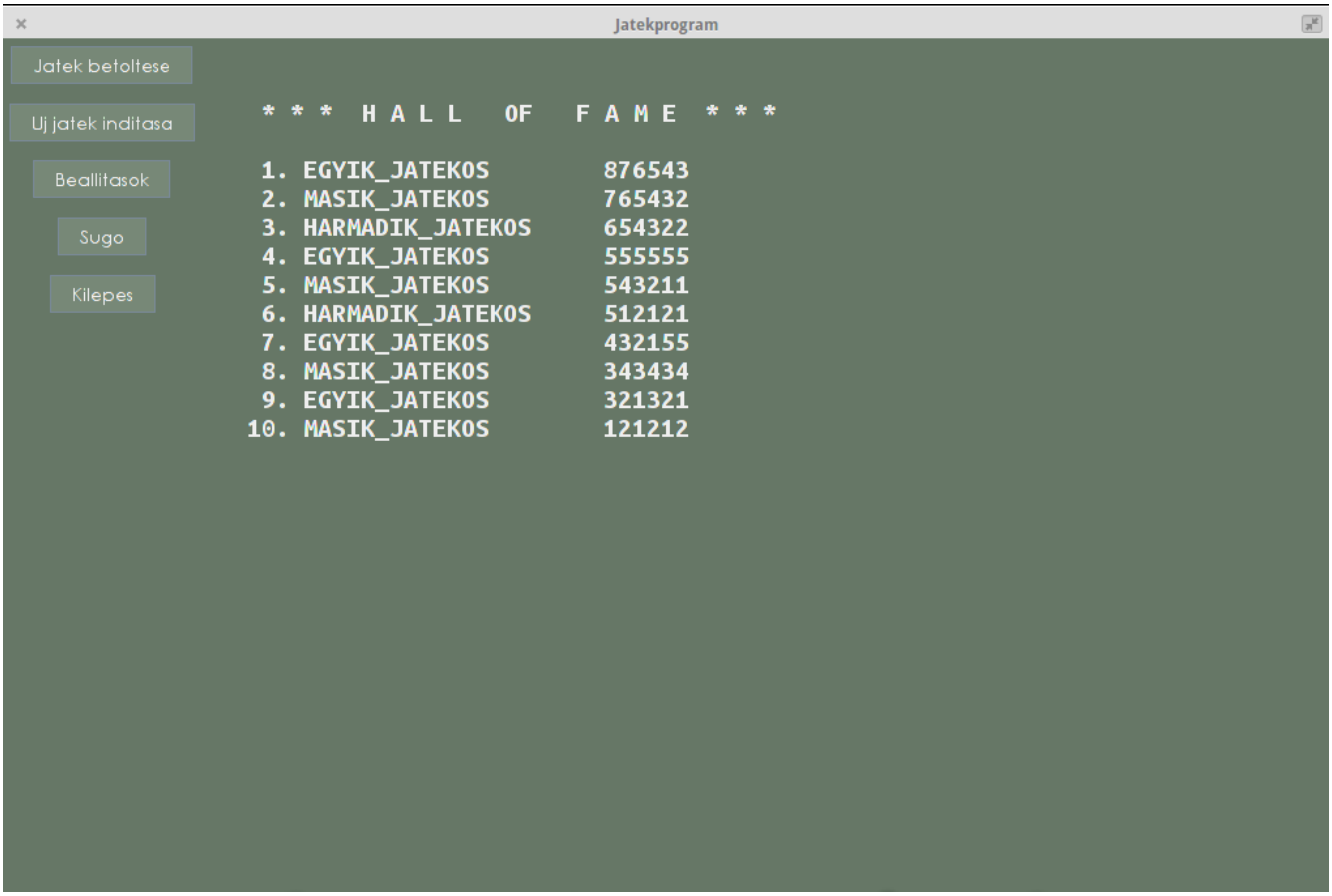
Java: Java SE 7

* A program bármilyen operációs rendszeren képes futni, ahol elérhető a megfelelő Java futtató környezet. Ezekhez azonban nem készült telepítő.

Használat

Főmenü

A program indítását követően a főmenübe kerül a játékos. Innen érhetők el a fontosabb funkciók, melyek részletes leírása alább található.



Betöltés

Ez a menüpont a korábban elmentett játékállást hivatott visszatölteni. Csak pályák visszatöltése lehetséges, tehát a módosított kód nincs elmentve.

Új játék indítása

Nevéhez illően új játékot indít, vagyis betölti az első pályát és nulla pontszámmal elindul a játék. A nézet átvált a játéktérre, ez a későbbiekben lesz ismertetve bővebben

Beállítások

Itt lehet paraméterezni a programot, illetve a játékmenetet.

Súgó

Az alapvető program leírást, illetve az irányításhoz szükséges billentyűk felsorolását tartalmazza a menüpont.

Kilépés

Kilép a programból. Figyelem, mentsük el az állást kilépés előtt!

Játéktér

Játék indításakor új nézetbe vált a program, mely a szerkeszthető kódot és a kiértékelés utáni állapotot tartalmazza (illetve néhány funkciógombot).

Szerkesztő ablak

Az ablak jobb felső részén található a kód. Amint megszereztük a számítógépet ez a szöveg szerkeszthetővé válik. Pontosabban csak a nem kiemelt részek (zöld a kiemelés). A kód egyes részei csupán azt a célt szolgálják, hogy rávezessenek a megoldásra így ezeket nem lehet írni. A beírt kódnak szintaktikusan helyesnek kell lennie, különben kiértékeléskor csak a fordító üzenete lesz a jutalom. Továbbá mivel a főprogram Java nyelven íródott, ezért a megjelenítésre csak a megadott interfészt lehet használni.

Pálya

A pálya a szerkesztőben megadott kód kiértékelésének eredménye. A fenti képen a bal felső sarokban látható. Minden pályán el van helyezve egy kijárat, amelynek elérése a végső cél mivel ezzel lehet befejezni az adott szintet. Szintén elérhető állandó jelleggel egy mentési hely. Ez a két elem konstans, nem módosítható és nem többszörözhető. Minden más objektum a pályán a programozó játékszere lehet, amennyiben a hozzá tartozó kód módosítható. Saját objektumok is létrehozhatók természetesen, a nyelvi elemek nincsenek korlátozva.

Információs sáv

Pár hasznos információ megjelenik a pálya alatt. Ezek közt szerepel az aktuális játékos neve, illetve az aktuális pálya. Fontosabb lehet azonban a megszerzett pontszám, amellyel fel lehet kerülni a legjobbak közé a főmenüben található eredmény listán. Továbbá a hátralevő időt is érdemes figyelni. Ez ugyan általában bőséges teret enged a gondolkodásnak, de ha túl sokáig tétlenkedünk az Algoritmus örökre elveszik.

Menüsáv

A nézet legalján található pár hasznos gomb. Ezek közül első a szüneteltetés funkciót rejti. Mivel a pálya teljesítése időre megy ezért érdemes szünetet tartani ha valami fontosabb dolgunk támad. Újra is lehet kezdeni az adott pályát ha tiszta lappal szeretnénk kezdeni. Továbbá elhagyhatjuk a játékot ha belefáradtunk a sok kódolásba, ezzel visszalépve a főmenübe.

6. Felhasznált irodalom

Sike S., Varga L.: Szoftverfejlesztés és UML (ELTE-Eötvös, 2003)

Sike S. : Tervezés és elemzés elmélete (Egyetemi jegyzet, 2014)
http://people.inf.elte.hu/sike/TervElem/tervelem.html

Java kódolási konvenciók
http://www.oracle.com/technetwork/java/codeconv-138413.html

Java 7 nyelv specifikációja
http://docs.oracle.com/javase/specs/jls/se7/jls7.pdf